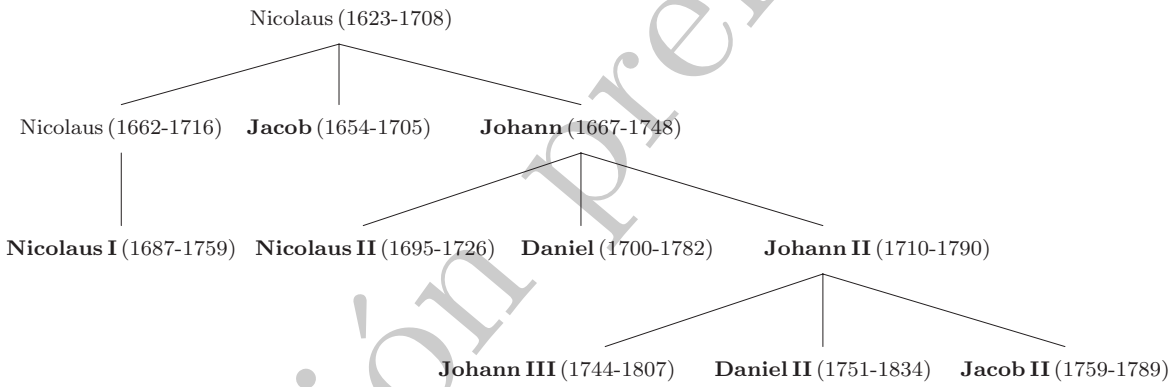
Capítulo 9

Árboles

La de los Bernoulli de Basilea es, quizás, la familia más famosa de las Matemáticas.



Famosa por la cantidad de excelentes matemáticos que "produjo" (hasta ocho, los que aparecen en negrita en el esquema anterior, en tres generaciones distintas) y, también, por la especial personalidad de algunos de ellos. De algunos de los más destacados ya hemos hablado en capítulos anteriores (Daniel, página 508, y Jacob, página 535). Lo que aquí nos interesa es, precisamente, la manera en que hemos exhibido la información sobre la familia, su *árbol genealógico*. Es un grafo, donde los vértices van etiquetados con los nombres de los componentes de la familia, que tiene una estructura especial.

En muchas otras cuestiones utilizamos estas estructuras arbóreas para representar información. La imagen que mostramos a la izquierda resultará, sin duda, familiar: se trata de la descripción que del contenido de un ordenador muestra el Explorador de Windows. Aquí, las etiquetas de los vértices son los nombres de los distintos dispositivos. Pero de nuevo la estructura es especial: se trata de un **árbol**. El capítulo que aquí comienza está dedicado al estudio de estos objetos y a su aplicación a diversas cuestiones: problemas de optimización (sección 9.2), diseño de algoritmos (sección 9.3), análisis de juegos (sección 9.4), etc.

9.1. Definición y características

La primera definición de la noción de árbol (de las varias que daremos) es la sugerida por los ejemplos vistos anteriormente:

Definición 9.1 *Un **árbol** es un grafo **conexo** y **sin ciclos**.*

En el mismo tono botánico, se define un **bosque** como un grafo sin ciclos (si es conexo, será un árbol; si no lo es, sus componentes conexas serán árboles). Por ejemplo, los grafos lineales L_n son árboles, mientras que los circulares C_n o los completos K_n no lo son en cuanto $n \geq 3$. Los bipartitos completos $K_{r,s}$, que son siempre conexos, sólo son árboles si $s = 1$ ó $r = 1$ (si $r \geq 2$ y $s \geq 2$ hay al menos un ciclo de orden cuatro).

Esta primera definición no recoge una de las características fundamentales de los árboles, que los hace especialmente útiles en ciertas cuestiones: son los conexos "más baratos" (en cuanto al número de aristas) que podemos tener. Los siguientes enunciados nos proporcionan caracterizaciones alternativas que recogen esta idea:

Proposición 9.1 *Un grafo G es un árbol (un conexo sin ciclos) $\Longleftrightarrow$ Es conexo y tiene la propiedad de que al eliminar una arista cualquiera del grafo, éste deja de ser conexo.*

DEMOSTRACIÓN. En un sentido, supongamos que tenemos un grafo G conexo y sin ciclos. Queremos probar que se desconecta al quitar una arista cualquiera.

Sea a una arista de G y formemos el grafo $G \setminus \{a\}$ eliminándola. Si $G \setminus \{a\}$ fuera conexo, podríamos conectar en $G \setminus \{a\}$ los vértices de a. Pero añadiendo la arista a, lo que tendríamos sería un ciclo en G (contradicción). Así que $G \setminus \{a\}$ es no conexo (sea cual sea la arista a de G que elijamos).

En el otro sentido, supongamos que G es un grafo conexo que se desconecta si quitamos cualquier arista. Si el grafo contuviera un ciclo, siempre podríamos quitar una arista de ese hipotético ciclo sin que el grafo dejara de ser conexo, lo que una contradicción. Luego ese tal ciclo no puede existir[1]. ■

Proposición 9.2 *Un grafo G es un árbol (un conexo sin ciclos) $\Longleftrightarrow$ No tiene ciclos y, si añadimos una arista cualquiera, se forma un ciclo.*

DEMOSTRACIÓN. En un sentido, sea G un grafo conexo y sin ciclos. Consideremos dos vértices cualesquiera que no estén unidos por arista alguna en G. Por estar en un grafo conexo, existirá un paseo que los conecte en G. Al añadir una arista entre los vértices, se formará un ciclo.

En el otro sentido, sea G un grafo sin ciclos para el que añadir una arista cualquiera supone la formación de un ciclo. Supongamos que no fuera conexo. En este caso, al menos existirían dos vértices que no podríamos conectar en G. Pero entonces todavía podríamos añadir la arista que los une sin que se nos formara un ciclo, contradicción. ■

[1] Como sabemos, véase el lema 8.2, si quitamos una arista de un grafo conexo y éste se desconecta, lo hace en exactamente dos componentes conexas. En el caso de un árbol, al quitar una arista cualquiera se formará un bosque con dos componentes conexas. Obsérvese que lo que nos dice esta proposición es que toda arista de un árbol es un puente.

Sabemos (recuérdese la proposición 8.3) que en un grafo conexo el número de aristas $|A(G)|$ no puede ser menor que $|V(G)| - 1$. La igualdad se alcanza para los árboles, como nos dice el siguiente resultado:

Proposición 9.3 *Un grafo G es un árbol (un conexo sin ciclos) $\Longleftrightarrow$ Es conexo y $|A(G)| = |V(G)| - 1$.*

Demostración. En un sentido, vamos a proceder por inducción (en su versión fuerte) sobre el número de aristas, $|A|$:

- Si G es un árbol con una arista, $|A(G)| = 1$, sólo cabe la posibilidad de que sea un L_2, para el que $|V(G)| = 2$.
- Supongamos cierto que para todo árbol con $|A(G)| \leq d$ se tiene que $|A(G)| = |V(G)|-1$.
- Consideremos un árbol cualquiera G con $|A(G)| = d+1$. Si nos fijamos en una arista e de G, sabemos que

  $$G \setminus \{e\} = G_1 \cup G_2,$$

  donde G_1 y G_2 son árboles con número de aristas $|A(G_1)| \leq d$ y $|A(G_2)| \leq d-1$ (de hecho, $|A(G_1)| + |A(G_2)| = d$). A estos dos árboles podemos aplicarles la hipótesis de inducción para deducir que

  $$\begin{array}{rrcl} & |A(G_1)| & = & |V(G_1)| - 1 \\ + & |A(G_2)| & = & |V(G_2)| - 1 \\ \hline & |A(G)| - 1 & = & |V(G)| - 2\,, \end{array}$$

  de donde obtenemos lo que queríamos.

En el otro sentido, tomemos un grafo conexo G tal que $|A(G)| = |V(G)| - 1$. Si contuviera un ciclo, podríamos quitar una arista e de ese ciclo sin que el grafo se desconectara. Pero habríamos llegado a un grafo, $G \setminus \{e\}$, conexo con

$$|A(G \setminus \{e\})| = |A(G)| - 1 \qquad \text{y} \qquad |V(G \setminus \{e\})| = |V(G)|\,.$$

Utilizando que $|A(G)| = |V(G)| - 1$,

$$|A(G \setminus \{e\})| = |V(G)| - 2 = |V(G \setminus \{e\})| - 2\,;$$

y esto contradice (¡nos faltan aristas!) el que $G \setminus \{e\}$ fuera conexo. ■

Los ejercicios 9.1.1 y 9.1.2 recogen algunas otras caracterizaciones. Resumamos todas las características que hacen de un grafo G un árbol: es conexo, sin ciclos, tiene $|A(G)| = |V(G)| - 1$ aristas y, si quitamos una arista cualquiera, se desconecta; y si añadimos una arista cualquiera, se forma un ciclo. Es decir, como ya adelantábamos, es el grafo conexo más barato (en el sentido de que no sobra ni falta arista alguna) que podemos construir.

Como parece claro, la sucesión de grados de los vértices de un árbol ha de ser muy especial. Ya sabemos que en cualquier grafo $G = (V, A)$ se tiene que

$$\sum_{v \in V} gr(v) = 2\,|A|\,.$$

Pero si además G es un árbol, como $|A| = |V| - 1$, se tendrá que

$$\sum_{v \in V} gr(v) = 2\,|V| - 2\,.$$

Obsérvese que ahora sólo tenemos un "grado de libertad", el número de vértices $|V|$, pues el de aristas ya queda fijado. De esta igualdad podemos deducir el siguiente resultado.

Proposición 9.4 *Todo árbol con $|V| \geq 2$ tiene, al menos, dos vértices de grado 1.*

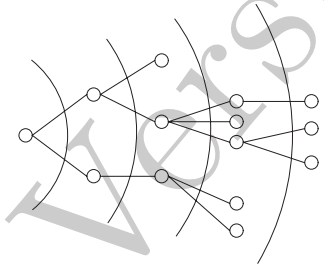
DEMOSTRACIÓN. Supongamos que no hay vértices de grado 1, es decir, que $gr(v) \geq 2$, para todo $v \in V$. Entonces,

$$2|V| - 2 = \sum_{v \in V} gr(v) \geq \sum_{v \in V} 2 = 2\,|V|\,,$$

lo que resulta imposible. Pero tampoco puede ocurrir que haya un sólo vértice w de grado 1, porque tendríamos

$$2|V| - 2 = \sum_{v \in V} gr(v) = gr(w) + \sum_{v \neq w} gr(v) \geq 1 + 2\,(|V| - 1) = 2|V| - 1$$

Así que al menos ha de haber dos de grado 1. ■



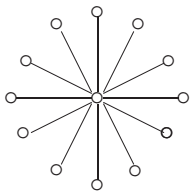
En muchas ocasiones conviene señalar un vértice especial en un árbol. Lo que queda es lo que llamaremos un **árbol con raíz**, donde la raíz, por supuesto, es ese vértice especial, que sirve de origen de coordenadas. Muchos de los algoritmos que veremos en estas páginas producen, de manera natural, un árbol con raíz, y en la sección 9.3 estudiaremos con detalle algunas aplicaciones de estos objetos. Por supuesto, cualquier árbol se convierte en uno con raíz en cuanto decidamos qué vértice actúa como raíz (la elección de este vértice especial puede hacer cambiar las propiedades del árbol con raíz, como veremos más adelante). En un árbol con raíz, los vértices se agrupan por generaciones: la primera contendría sólo a la raíz, la segunda estaría formada por todos los vértices vecinos de la raíz; la tercera, por los vecinos de estos últimos (salvo la raíz); la cuarta, por los vecinos de los de la tercera generación (excepto los que ya estaban en la segunda)... y así sucesivamente. Obsérvese que los vértices de la generación k sólo pueden estar unidos a vértices de las generaciones anterior y posterior, porque si no, tendríamos ciclos. Lo que la proposición anterior afirma es que al menos hay dos vértices terminales (que luego llamaremos hojas) en un árbol como éste. El caso límite es el de un L_n.

EJEMPLO 9.1.1 *¿Cómo son los árboles con n vértices que tienen el menor y el mayor número posible de vértices de grado 1?*

Sabemos que el mínimo número de vértices de grado 1 es 2. Así que, si un árbol con n vértices tiene exactamente dos vértices, digamos w y u, de grado 1, se cumplirá que

$$2n-2=\sum_{v\in V} gr(v)=1+1+\sum_{v\neq u,w} gr(v).$$

En la suma final tenemos $n-2$ términos, todos ellos mayores o iguales que 2; la única forma de conseguir la igualdad será que $gr(v)=2$ para todo $v\in V$, $v\neq u,w$. Y esta configuración de grados es la del grafo lineal con n vértices, L_n.



En el otro extremo, es imposible que todos los vértices tengan grado 1 (pues no se cumpliría la fórmula de los grados). Pero sí podría ocurrir que hubiera $n-1$ de grado 1. El vértice restante, w, tendría grado

$$2n-2=\sum_{v\in V} gr(v)=(n-1)+gr(w)\Longrightarrow gr(w)=n-1\,,$$

que, por cierto, es el máximo grado que puede tener un vértice en un grafo con n vértices. En este caso, tenemos el grafo estrellado de la izquierda. ♣

9.1.1. Contando el número de árboles distintos

Cayley[2] estaba muy interesado en la cuestión de enumerar ciertos compuestos orgánicos, quizás uno de los primeros intentos serios de aplicar las Matemáticas a la Química. Hoy en día, cuando gran parte de la Química moderna se escribe en términos de modelos matemáticos, enternece recordar el comentario que el filósofo positivista Auguste Comte hacía allá por 1830:

> [...] cualquier intento de emplear métodos matemáticos en el estudio de la Química debe ser considerado como profundamente irracional y contrario al propio espíritu de la Química.

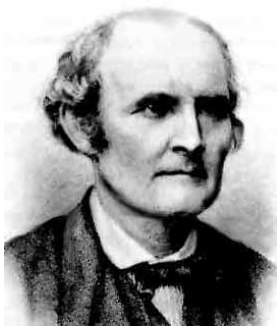


FIGURA 9.1: Cayley

Pero volvamos a los esfuerzos de Cayley, que estaba interesado, por ejemplo, en la cuestión de enumerar los isómeros de hidrocarburos saturados, cuya fórmula es $C_k\,H_{2k+2}$: los átomos de carbono tienen valencia 4, mientras que los de hidrógeno, valencia 1. Se trata de un grafo con $3k+2$ vértices, cuya suma de grados es $4k+(2k+2)=6k+2$, así que el grafo

[2]El inglés Arthur Cayley (1821-1895) fue, durante parte de su vida, un "matemático aficionado". Tras su paso por Cambridge, ejerció durante 14 años como abogado, periodo en el que llegó a publicar más de 200 artículos de investigación en Matemáticas. En su tarea coincidió con James Joseph Sylvester (1814-1897), otro ilustre matemático y abogado, con quien gustaba de discutir sobre casos judiciales y teoremas. Algunas de esas discusiones matemáticas estaban dedicadas a la Teoría de grafos. A partir de 1863 se dedicaría en exclusiva a las Matemáticas, desde su puesto en Cambridge. Cayley es uno de los padres de la teoría de grupos y el creador del cálculo matricial. También son fundamentales sus aportaciones a la geometría no euclídea y la geometría proyectiva.

debe tener $3k+1$ aristas. ¡Ah!, el número de aristas es el de vértices menos 1: se trata de un árbol (si suponemos, como parece razonable, que el grafo es conexo).

Enumerar los grafos no isomorfos con n vértices y una cierta estructura (por ejemplo, ser árboles, tener una determinada sucesión de grados) es, en general, una cuestión muy complicada; en los ejemplos de la subsección 8.1.2 ya lo empezamos a sospechar. Cayley sólo consiguió resolver algunos casos particulares y, años después, Pólya desarrollaría su teoría enumerativa (véase el capítulo 14) para dar respuesta a los problemas de este tipo. Sin embargo, si etiquetamos los vértices, las cosas se simplifican a veces. Por ejemplo, sabemos que, dado el conjunto $\{v_1, \ldots, v_n\}$, hay $2^{\binom{n}{2}}$ grafos distintos con ese conjunto de vértices. Cayley obtuvo[3], en 1889, una fórmula, sorprendentemente simple, para el caso de los árboles con n vértices. Veamos qué nos sugieren algunos ejemplos sencillos.

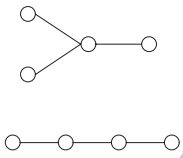
EJEMPLO 9.1.2 *Contemos el número de árboles distintos con 2, 3 y 4 vértices.*

Si tenemos dos vértices, sólo cabe una posibilidad, que el árbol sea isomorfo a un L_2. Y si el conjunto de vértices es $\{1,2\}$, hay también un único árbol (los dos posibles etiquetados de los vértices dan el mismo resultado).

Los árboles con tres vértices también han de ser isomorfos al grafo lineal correspondiente, L_3. Si el conjunto de vértices es $\{1,2,3\}$, basta con decidir qué símbolo va, por ejemplo, en la posición central (cuál es el vértice de grado 2). Esto se puede hacer de tres formas distintas, así que hay 3 árboles distintos con vértices $\{1,2,3\}$.

Vamos con el caso de cuatro vértices. Para asegurarnos de no olvidarnos ningún caso, ayudémonos de la relación

$$gr(v_1) + gr(v_2) + gr(v_3) + gr(v_4) = 6\,.$$



Ninguno de los cuatro números puede ser ≥ 4 (no puede haber vértices de grado 4) y un simple análisis nos lleva a concluir que sólo puede haber dos sucesiones de grados aceptables, $(1,1,1,3)$ y $(1,1,2,2)$. La primera de ellas se corresponde con el grafo que dibujamos a la izquierda y arriba, mientras que la segunda se traduce en el debajo. Para etiquetar el primer grafo con $\{1,2,3,4\}$, observamos que basta con decidir el símbolo que va en la posición central; así que hay cuatro maneras distintas de hacerlo. El etiquetado del grafo de la derecha es un poco más delicado: elegimos primero los dos vértices de grado 2 (se puede hacer de $\binom{4}{2}$ formas); y para cada elección de éstas, hay luego dos posibilidades para elegir los vecinos. En total, 12 maneras distintas. En resumen, con 4 vértices hay dos árboles no isomorfos y 16 árboles distintos con vértices $\{1,2,3,4\}$. ♣

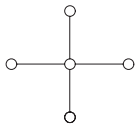
EJEMPLO 9.1.3 *Un poco más difícil: árboles con 5 vértices.*

Se debe cumplir que

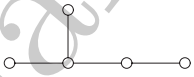
$$\sum_{i=1}^{5} gr(v_i) = 8\,,$$

[3]Cayley sólo enunció el resultado, lo que también había hecho Sylvester años antes, en 1857. De hecho, Borchardt había publicado una demostración en 1860.

así que no puede haber vértices de grado 5 o mayor. Si hay de grado 4, la sucesión de grados ha de ser $(1,1,1,1,4)$, a la que le corresponde un único árbol, el que aparece a la derecha. Etiquetarlo con $\{1,2,3,4,5\}$ es muy sencillo, pues basta decidir qué situamos en el vértice central: en total, 5 posibilidades.



Si no hay de grado 4, pero sí de grado 3, la única sucesión de grados posible es (1,1,1,2,3), y tenemos el grafo de la derecha. Para etiquetarlo con $\{1,2,3,4,5\}$, fijamos el símbolo del vértice de grado 3 (5 posibilidades), el del vértice de grado 2 (4 posibilidades) y, finalmente, elegimos el vecino de grado 1 del vértice de grado 2 (3 posibilidades, las mismas que obtendríamos eligiendo los dos vecinos de grado 1 del vértice de grado 3). En total, $5 \times 4 \times 3 = 60$ posibilidades.



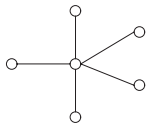
Por último, si no hay vértices de grado 3, entonces sólo podremos tener la sucesión de grados $(1,1,2,2,2)$, que corresponde a un L_5. Para etiquetarlo, elegimos el símbolo del vértice central (5 posibilidades), luego los otros dos de grado 2 ($\binom{4}{2} = 6$ posibilidades) y ya sólo quedan dos posibilidades para etiquetar los vértices finales. En total, $5 \times 6 \times 2 = 60$ formas distintas.

Resumiendo, con 5 vértices hay tres árboles no isomorfos, y 125 árboles distintos con vértices $\{1,2,3,4,5\}$. ♣

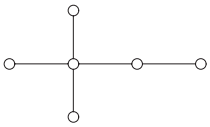
Ejemplo 9.1.4 *Más difícil todavía: árboles con seis vértices.*

De nuevo nos dejamos guiar por la relación $gr(v_1) + \cdots + gr(v_6) = 10$, que nos dice que no puede haber vértices de grado 6 o mayor.

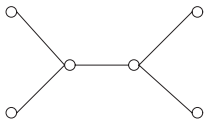
Si hay vértices de grado 5, sólo podrá haber uno, y la sucesión de grados será $(1,1,1,1,1,5)$. El único grafo con estas características es el que aparece a la derecha. Y el etiquetado de los vértices con $\{1,\ldots,6\}$ sólo requiere decidir el símbolo del vértice central: seis posibilidades.



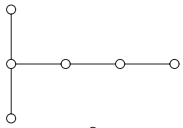
Si no hay de grado 5, pero hay de grado 4, sólo cabe que la sucesión sea $(1,1,1,1,2,4)$. El árbol correspondiente aparece a la derecha. Dejamos como ejercicio para el lector la comprobación de que hay 120 árboles con vértices $\{1,\ldots,6\}$ asociados a esta estructura.

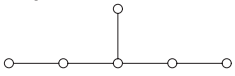


Si no hay vértices de grado 4, pero sí de grado 3, las cosas se ponen interesantes. Tenemos, por un lado, la sucesión de grados $(1,1,1,1,3,3)$, cuyo árbol asociado es el de la figura. El lector debería comprobar que hay 90 formas distintas de etiquetar los vértices de este árbol.



Pero, por otro lado, tenemos la sucesión de grados $(1,1,1,2,2,3)$, que tiene un único vértice de grado 3. Como ya vimos en el ejemplo 8.1.4, hay **dos** árboles no isomorfos a esta sucesión, los que se muestran a la derecha. Es, de nuevo, un sencillo ejercicio comprobar que hay 180 etiquetados distintos para el de arriba y 360 para el de abajo.





Por último, si no hay de grado 3, entonces sólo cabe tener cuatro vértices de grado 2, $(1,1,2,2,2,2)$, y el grafo es isomorfo a un L_6. Y hay 360 distintos árboles que podemos construir con esta estructura y vértices $\{1,\ldots,6\}$.

En resumen, hay seis árboles no isomorfos con 6 vértices, y 1296 árboles distintos con vértices $\{1,\dots,6\}$. ♣

Los resultados que hemos ido obteniendo hasta ahora son especialmente "sospechosos"; y más, si como aparece en la tabla, calculamos los dos siguientes casos:

n	árboles no isomorfos	árboles distintos
2	1	$1 = 2^0$
3	1	$3 = 3^1$
4	2	$16 = 4^2$
5	3	$125 = 5^3$
6	6	$1296 = 6^4$
7	11	$16807 = 7^5$
8	23	$32768 = 8^6$

Parece razonable suponer que la respuesta general es n^{n-2}.

Teorema 9.5 (Cayley) *El número de árboles distintos que se pueden formar con el conjunto de vértices* $\{1,\dots,n\}$ *es* n^{n-2}.

Existen diversas pruebas de este resultado. La que aquí veremos es especialmente sencilla y sugerente y se debe a Zeilberger. En los ejercicios 9.1.7 y 9.1.8 se proponen un par de ellas más; la última de ellas, además, proporciona un mecanismo para codificar la información que determina un árbol, mediante el llamado código de Prüfer (véanse también el ejemplo 9.2.3 y la sección 13.3).

DEMOSTRACIÓN. La escribiremos con el lenguaje (siempre simpático) que utiliza Zeilberger, aunque luego lo reinterpretaremos en términos de grafos.

En una cierta empresa hay m jefes y k empleados. Los queremos organizar de manera que cada empleado tenga un *único* supervisor (que pudiera ser otro empleado o quizás un jefe). Al número de maneras distintas en que esto se puede hacer lo llamaremos $P(m,k)$.

Digamos que $m \geq 1$ y que $k \geq 0$. Es claro que $P(m,0) = 1$ para cada $m = 1, 2, \dots$. Para obtener otros valores valor de $P(m,k)$ seguimos el siguiente proceso:

1) Distinguimos a los empleados cuyos supervisores son jefes. A estos empleados, de los que habrá un cierto número s, con $1 \leq s \leq k$, se les asigna, desde este momento, y como no podía ser de otro modo, el papel de *jefecillos*.

2) Decidimos, pues, qué s empleados son jefecillos; esto se puede hacer de $\binom{k}{s}$ maneras.

3) Y ahora asignamos el jefe ante el que responde cada jefecillo: m^s maneras.

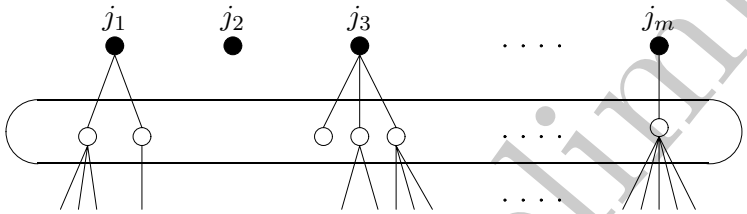
4) Queda organizar a los restantes $k-s$ empleados. Pero, para éstos, los s jefecillos actúan como jefes. Así que los podremos organizar de tantas maneras como nos diga $P(s,k-s)$.

En total,

$$P(m,k) = \sum_{s=1}^{k} \binom{k}{s} m^s \, P(s,k-s)\,.$$

Esta regla de recurrencia, junto con los valores iniciales de antes, determina de manera única el valor de los $P(m,k)$, como el lector podrá comprobar construyéndose la correspondiente tabla de valores.

Cambiando al lenguaje de los grafos, lo que cuenta $P(m,k)$ es el número de bosques (con $m+k$ vértices) formados por m árboles con raíz, donde las raíces llevan, como etiquetas, los nombres de los jefes. El argumento que llevaba a la regla de recurrencia, en estos nuevos términos, consiste en decidir qué vértices van en la segunda generación (por debajo de las raíces), y cómo se distribuyen en esa segunda generación:



El caso que nos interesa realmente es cuando $m=1$ y $k=n-1$. Tenemos entonces un árbol con raíz con n vértices, en el que la raíz ya va etiquetada. Si partimos de los símbolos $\{1,\ldots,n\}$, hay n maneras de etiquetar la raíz. Así que $nP(1,n-1)$ cuenta cuántos árboles con raíz podemos formar con n vértices. Pero por cada árbol con n vértices hay n árboles con raíz distintos. Así que concluimos que la cantidad que nos interesa, el número de árboles con n vértices, coincide con $P(1,n-1)$.

Sólo queda resolver la recurrencia de antes para obtener el valor particular de los $P(m,k)$ que perseguimos. Aquí llega el ingrediente ingenioso de la prueba. La ecuación de antes nos recuerda vagamente a las sumas que se obtienen en el teorema del binomio. El lector podrá comprobar, utilizando este teorema, que la función $f(m,k)=m(m+k)^{k-1}$ cumple la misma regla de recurrencia y los mismos valores iniciales. Así que $P(m,k)=f(m,k)$ para todo m y k. En particular,

$$P(1,n-1)=f(1,n-1)=1\times(1+(n-1))^{(n-1)-1}=n^{n-2}\,,$$

y el resultado de Cayley queda demostrado. ■

EJERCICIOS.

9.1.1 *Probar que un grafo G es un árbol (conexo sin ciclos) si y sólo si no tiene ciclos y $|A(G)|=|V(G)|-1$.*

9.1.2 *Probar que en un árbol existe un único camino que conecta dos vértices cualesquiera.*

9.1.3 *Probar que el grafo G con n vértices es un árbol si y sólo si su polinomio cromático es $p_G(k)=k\,(k-1)^{n-1}$.*

Sugerencia. Para la implicación $\Rightarrow$, probar por inducción, y para ello, considerar un vértice de grado 1 del árbol y aplicar el algoritmo de "quitar aristas". Para la implicación $\Leftarrow$, usar el teorema del binomio y la caracterización de árboles como grafos conexos con $n-1$ aristas.

9.1.4 *Sea G un grafo con n vértices cuyo polinomio cromático P_G cumple que $P'_G(0) \neq 0$ y que $P_G^{(n-1)}(0) = (1-n)(n-1)!$. Demostrar que G es un árbol.*

Sugerencia. Utilizar el significado de los coeficientes b_{n-1} y b_1 del polinomio cromático de cualquier grafo. Observar que el recíproco es también cierto (otra caracterización de un árbol).

9.1.5 *(a) ¿Existen árboles de siete vértices y con cinco vértices de grado 1 y dos de grado 2? (b) ¿Y con siete vértices de grados $1, 1, 1, 2, 2, 2, 3$?*

Sugerencia. Para árboles sabemos relacionar el valor de la suma de los grados con el número de vértices.

Solución. (a) No, (b) Sí (construir un ejemplo).

9.1.6 *Si G es árbol con p vértices de grado 1, q vértices de grado 4 y ningún otro vértice, ¿qué relación hay entre p y q?*

Solución. $p = 2q + 2$.

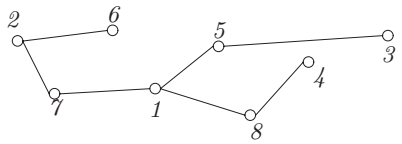
9.1.7 *Denotemos por $N(d_1, d_2, \ldots, d_n)$ el número de árboles distintos que se pueden formar con el conjunto de vértices $\{1, 2, \ldots, n\}$, donde $gr\,(j) = d_j + 1$.*
(a) Observar que si $\sum_{j=1}^{n} d_j \neq n-2$ entonces $N(d_1, d_2, \ldots, d_n) = 0$.
(b) Probar la siguiente fórmula de recurrencia:

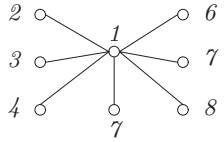
$$N(d_1, d_2, \ldots, d_{n-1}, 0) = N(d_1-1, d_2, \ldots, d_{n-1}) + N(d_1, d_2-1, \ldots, d_{n-1}) + \cdots + N(d_1, d_2, \ldots, d_{n-1}-1)\,,$$

donde en la suma anterior el término i-ésimo no aparece si $d_i = 0$.
(c) Deducir que si $\sum_{j=1}^{n} d_j = n-2$, entonces $N(d_1, d_2, \ldots, d_n) = \binom{n-2}{d_1, d_2, \ldots, d_n}$.
(d) Deducir finalmente la fórmula de Cayley: el número de árboles distintos que se pueden formar con los vértices $\{1, 2, \ldots, n\}$ es n^{n-2}.

Sugerencia. Para el primer apartado, observar que la condición es justamente la necesaria para que tengamos un árbol. Para el segundo, clasificar los árboles dependiendo de a qué vértice está unido el de grado 1. Para el tercero, probar que ambos objetos comparten la misma recurrencia y las mismas condiciones iniciales. Para d), aplicar la fórmula del multinomio.

9.1.8 *Sea T un árbol con n vértices etiquetados con los números $\{1, \ldots, n\}$. Localizamos el vértice de grado 1 con menor etiqueta, que llamamos b_1, y apuntamos quién es su único vecino, a_1. Borramos entonces b_1 y su arista. En el nuevo árbol, repetimos el procedimiento: localizamos el vértice de grado 1 con menor etiqueta, b_2, apuntamos quién es su vecino a_2, y borramos b_2 y su arista. Y así, sucesivamente, hasta quedarnos con sólo un vértice. Lo que hemos ido anotando forma una lista $(a_1, \ldots, a_{n-1})$, el llamado **código de Prüfer** del árbol T. Esta lista puede tener símbolos repetidos y $a_{n-1} = n$.*
(a) Hallar los códigos de Prüfer de los árboles siguientes:





(b) Comprobar que, a partir del código de Prüfer $(a_1, \ldots, a_{n-1})$ *de un cierto árbol, podemos recuperar el árbol en sí. Es decir, la sucesión* $(b_1, \ldots, b_{n-1})$ *de los vértices que son borrados por el algoritmo.*

(c) Demostrar que, dada una lista de números $(a_1, \ldots, a_{n-1})$*, con* $1 \leq a_j \leq n$ *para cada* $j = 1, \ldots, n-2$ *y* $a_{n-1} = n$*, existe un único árbol cuyo código de Prüfer es, precisamente, esa lista.*

(d) Deducir la fórmula de Cayley.

Sugerencia. Para el apartado (b), comprobar que b_1 es el menor elementos que no esté en el código de Prüfer. Para los restantes, comprobar que

$$b_j = \text{mín}\left\{k \,:\, k \notin \{b_1, \ldots, b_{j-1}, a_j, \ldots, a_{n-1}\}\right\}.$$

Las aristas son siempre de la forma $\{a_j, b_j\}$.

Para el apartado (c), construir, a partir de la lista, el grafo T correspondiente, con la regla del apartado (b).

Solución. (a) $(5, 8, 1, 2, 7, 1, 8)$ y $(1, 1, 1, 1, 1, 1, 8)$.

9.1.9 *Hallar el número de árboles distintos que se pueden formar con los vértices* $\{1, 2, \ldots, n\}$ *si (a)* $n = 6$ *y cuatro vértices tienen grado 2; (b)* $n = 5$ *y exactamente tres vértices tienen grado* 1*.*

Solución. (a) 360, (b) 60.

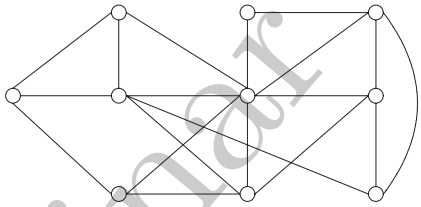
9.1.10 *¿Cuántos árboles distintos se pueden formar con un conjunto de ocho vértices,* $\{1, 2, 3, \ldots, 8\}$*, de manera que 2 de los vértices tengan grado 4 y los 6 restantes tengan grado 1?*

Solución. 560.

9.1.11 *¿Cuántos árboles con vértices* $\{1, 2, \ldots, n\}$ *tienen los grados de todos sus vértices menores o iguales a* 2*?*

Solución. $n!/2$.

9.2. Algoritmos en árboles y optimización



Los árboles son unas estructuras especialmente adaptadas a los problemas de optimización. Una de las cuestiones que planteamos al principio del capítulo 8 era la siguiente: queremos construir una red que conecte una serie de puntos (por ejemplo, podría ser un oleoducto que conecte la refinería con unas cuantas ciudades, o una red de ordenadores) de la forma más barata (en cuanto a número de tramos) a partir de un diseño previo, como el de la figura. Si el objetivo es mantener conectadas todas las ciudades, hay tramos que se podrían eliminar; el objetivo es quitar el mayor número posible de aristas de manera que el grafo siga siendo conexo. O, dicho de otra manera, quedarnos con el número mínimo de aristas que garantizan la conexión del grafo. Estamos buscando, en definitiva, un árbol que incluya a todos los vértices.

Definición 9.2 *Consideremos un grafo $G = (V, A)$. Diremos que un árbol H es* ***árbol abarcador***[4] *de G si cumple que:*

- $V(H) = V(G)$ *(tiene los mismos vértices que G).*
- $A(H) \subseteq A(G)$ *(tiene algunas —o todas— las aristas de G).*

Es decir, es un subgrafo abarcador del grafo inicial que, además, es un árbol. Asegurémonos primero de que tales árboles existen si, como es razonable, partimos de un grafo conexo.

Proposición 9.6 *Un grafo G es conexo si y sólo si tiene, al menos, un árbol abarcador.*

DEMOSTRACIÓN. En un sentido, si un grafo tiene un árbol abarcador, que por definición es conexo, es obvio que G también lo será.

En el otro: consideremos un grafo G conexo. Si es un árbol, ya hemos acabado (G es su propio árbol abarcador). Pero si no es árbol, como es conexo, podemos todavía quitar una cierta arista e sin que se desconecte (recordemos el carácter extremal de los árboles en este sentido). Así que $G \setminus \{e\}$ es conexo. Si es árbol, hemos acabado; pero si no lo es, podremos quitar una arista f de forma que $(G \setminus \{e\}) \setminus \{f\}$ siga siendo conexo. Y así, sucesivamente. En cada paso, los sucesivos grafos son conexos; cuando el número de aristas llegue a $|V(G)| - 1$, habremos llegado a un árbol abarcador. ■

De esta demostración deducimos que si el grafo G es conexo y $|A(G)|+1 = |V(G)|+k$, con $k \geq 0$, entonces podemos quitar k aristas (convenientemente escogidas) y quedarnos con un árbol abarcador. Se trata de ir "eliminando aristas" hasta quedarnos con el número adecuado de ellas. El criterio para eliminarlas sería el de ir "rompiendo" los ciclos del grafo. Pero este proceso requiere identificar los distintos ciclos, algo difícil de implementar en el ordenador.

Existen otros algoritmos para construir un árbol abarcador, que van "haciendo crecer" el árbol en pasos sucesivos, y que resultan ser más sencillos desde el punto de vista computacional. Los veremos en la subsección 9.2.1, aunque allí, por el mismo precio, resolveremos

[4] *Spanning tree*, en la terminología anglosajona. De nuevo, nuestra traducción difiere de la habitual ("árbol generador"), para reflejar que el árbol, efectivamente, contiene todos los vértices del grafo.

una cuestión más general, pues permitiremos que las aristas tengan "peso". Como veremos, la cuestión de hallar un árbol abarcador en un grafo sin pesos no es sino un caso particular de este planteamiento más general.

El número de árboles abarcadores de un árbol

Que un grafo conexo tiene, en general, más de un árbol abarcador es bastante evidente. Dado un grafo G, ¿cuántos hay, exactamente? La respuesta, claro, dependerá, del tipo de grafo considerado.

EJEMPLO 9.2.1 *El número de árboles abarcadores de los grafos C_n, L_n y K_n.*

Consideremos el grafo circular con n vértices, C_n, para el que $|A(C_n)| = |V(C_n)|$. Formar un árbol abarcador consiste, simplemente, en quitar una arista; y cualquiera de las n que hay vale para ello. Así que C_n tiene n posibles árboles abarcadores.

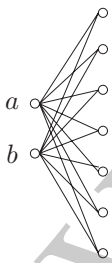
El grafo lineal con n vértices, L_n, es ya un árbol. Así que L_n es el único árbol abarcador que tenemos. Esto es un resultado general: si G es un árbol, sólo tiene un árbol abarcador (él mismo).

Por último, consideremos el grafo completo con n vértices, K_n. Buscamos subgrafos H de K_n que sean árboles, que tengan $V(H) = V(K_n) = \{1, \ldots, n\}$. Pero en K_n están todas las aristas posibles, así que

$$\#\left\{\begin{array}{c}\text{árboles}\\ \text{abarcadores de } K_n\end{array}\right\} = \#\left\{\begin{array}{c}\text{árboles distintos con el conjunto}\\ \text{de vértices } \{1,\ldots,n\}\end{array}\right\} = n^{n-2}\,.$$

K_n es, claro, el grafo con n vértices que más árboles abarcadores tiene. ♣

EJEMPLO 9.2.2 *¿Cuántos árboles abarcadores distintos contiene un $K_{r,s}$?*



Empecemos con el grafo bipartito completo $K_{2,s}$, $s \geq 2$. El grafo tiene $2s$ aristas y $s+2$ vértices, así que tendremos que quitar $s-1$ aristas sin que se desconecte el grafo. Es complicado localizar los ciclos del grafo para ir rompiéndolos, así que pensemos de otra manera. Un árbol abarcador del grafo contendrá a una serie de vértices de los de la derecha que se conectan al vértice a y otros que se conectan al vértice b. Pero ha de haber al menos uno que se conecte a ambos, para que sea conexo. Y sólo uno, porque si hubiera dos (o más) vértices a la derecha que conservaran sus dos aristas, se formaría un ciclo. Con esta información podemos contar el número de árboles abarcadores distintos. Llamemos

$$\begin{aligned} A &= \{\text{vértices de } \{1,\ldots,s\} \text{ que comparten arista con } a \text{ en el árbol abarcador}\}\\ B &= \{\text{vértices de } \{1,\ldots,s\} \text{ que comparten arista con } b \text{ en el árbol abarcador}\}\end{aligned}$$

Lo que buscamos son conjuntos A y B tales que entre los dos tengamos todos los vértices de la derecha y de forma que su intersección conste de un único vértice:

$$\begin{array}{c}\text{Elegir un árbol}\\ \text{abarcador de } K_{2,s}\end{array} \iff \begin{array}{c}\text{Elegir dos conjuntos, } A \text{ y } B \text{ de manera que}\\ A \cup B = \{1,\ldots,s\} \text{ y de forma que } |A \cap B| = 1\end{array}$$

Para contar el número de maneras de elegir A y B, sigamos el siguiente proceso:

1. Elegimos el elemento de la intersección (hay s posibilidades).

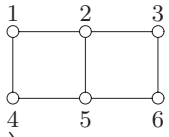
2. Una vez elegido el elemento especial, basta decidir si el resto de los elementos están en A ó en B. O, lo que es equivalente, basta formar una lista de longitud $s-1$ con repetición permitida con dos símbolos (uno corresponde a estar en A y el otro a estar en B). Esto se puede hacer de 2^{s-1} formas distintas.

En total tenemos que hay $s\,2^{s-1}$ árboles abarcadores de $K_{2,s}$. El resultado general es el siguiente: un grafo $K_{r,s}$ tiene $s^{r-1}\,r^{s-1}$ árboles abarcadores distintos (en el ejercicio 9.2.3 se pide estudiar el caso del $K_{3,s}$). ♣

Si un grafo conexo G tiene n vértices, el número de árboles abarcadores que contiene va desde 1 (si G es ya un árbol) hasta n^{n-2}, cuando G sea el grafo completo con n vértices. Este número mide, en cierta manera, la *complejidad* del grafo. El siguiente resultado, que enunciamos sin demostración, nos proporciona una manera de calcular esa medida de complejidad en un grafo arbitrario.

Teorema 9.7 (Kirchhoff) *El número de árboles abarcadores de un grafo G coincide con un cofactor*[5] *cualquiera de la matriz $D-M$, donde M es la matriz de vecindades del grafo y D es la matriz diagonal cuyas entradas son los grados de los vértices.*

Obsérvese que, en la matriz $D-M$, todas sus filas y columnas suman 1, así que es una matriz singular. Este resultado permite, por ejemplo, calcular el número de árboles abarcadores del grafo escalera que aparece a la derecha. Las matrices M y D son



$$M=\begin{pmatrix} 0&1&0&0&0&1\\ 1&0&1&0&1&0\\ 0&1&0&1&0&0\\ 0&0&1&0&1&0\\ 0&1&0&1&0&1\\ 1&0&0&0&1&0 \end{pmatrix} \quad \text{y} \quad D=\begin{pmatrix} 2&0&0&0&0&0\\ 0&3&0&0&0&0\\ 0&0&2&0&0&0\\ 0&0&0&2&0&0\\ 0&0&0&0&3&0\\ 0&0&0&0&0&2 \end{pmatrix}.$$

El cálculo de un cofactor cualquiera de $D-M$ (a mano o bien con ayuda de un paquete matemático) nos lleva a concluir que hay 15 árboles abarcadores en el grafo. ♣

EJEMPLO 9.2.3 *Kirchhoff y Cayley.*

Queremos contar cuántos árboles abarcadores tiene un grafo completo K_n, cuyos vértices

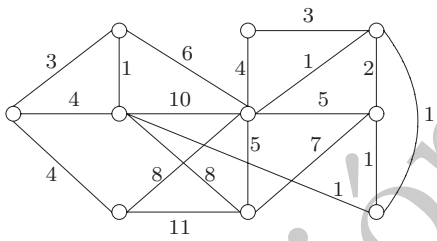
[5] Dada una matriz A $n\times n$, el menor $M_{i,j}$ es el determinante de la matriz A, a la que le hemos quitado la fila i y la columna j. El cofactor correspondiente es $C_{i,j}=(-1)^{i+j}M_{i,j}$.

tienen todos grado $n-1$:

$$M = \begin{pmatrix} 0 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 1 & \cdots & 1 \\ 1 & 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & 0 \end{pmatrix} \quad \Longrightarrow \quad D - M = \begin{pmatrix} n-1 & -1 & -1 & \cdots & -1 \\ -1 & n-1 & -1 & \cdots & -1 \\ -1 & -1 & n-1 & \cdots & -1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & -1 & \cdots & n-1 \end{pmatrix}$$

La matriz de la derecha es $n \times n$, y la respuesta que buscamos está, salvo un signo, en el determinante de la matriz que es idéntica a $D - M$, pero de dimensiones $(n-1) \times (n-1)$. El valor de ese determinante, recordemos, no cambia con las operaciones de eliminación gaussiana. El lector podrá comprobar que, con las operaciones adecuadas (sumar la primera fila a las restantes, sacar un factor n de cada una de ellas, y sumarle finalmente todas las filas a la primera), el determinante original resulta ser $n^{n-2} \det(I)$, donde I es la matriz identidad $(n-1) \times (n-1)$. El resultado de Cayley queda, de nuevo, probado. ♣

9.2.1. Árbol abarcador de menor peso



Vamos a considerar de nuevo el problema de la red de conducción, pero ahora añadiremos un ingrediente nuevo. El estudio previo de ingeniería nos informa de qué tramos es posible construir pero, además, disponemos de la información sobre el *coste* de cada uno de esos tramos. Queremos, claro, elegir una red que conecte todas las ciudades con el menor coste posible. La información de los costes se traduce en que cada arista lleva asociado un número. Lo que buscamos es un árbol abarcador del grafo, pero justo aquél (o aquéllos) para el que la suma de los costes de las aristas elegidas sea mínimo.

Para modelar esta situación, necesitamos una generalización del concepto de grafo. Un **grafo con pesos** (o grafo ponderado) será un grafo G en el que, además, cada arista a tenga asociado lo que llamaremos su **peso**, $p(a)$, un número real no negativo[6]. La matriz de vecindades de un grafo con pesos será simétrica, con ceros en la diagonal, y sus entradas serán los pesos de las aristas (o 0 si no hay tales aristas). Recordemos que en la sección 8.7 ya manejamos un concepto todavía más general, el de grafos dirigidos con pesos (redes).

La cantidad que nos interesa minimizar es lo que llamaremos el **peso del grafo**,

$$p(G) = \sum_{a \in A(G)} p(a).$$

El problema es el siguiente: elegir, de entre todos los posibles árboles abarcadores de G, uno,

[6]Esta restricción a pesos no negativos, aunque suene muy razonable, no es en realidad necesaria para la discusión que haremos sobre árboles abarcadores de peso mínimo. Sin embargo, sí que será fundamental cuando el problema que nos interese sea la construcción de caminos más cortos (véase la subsección 9.2.3).

H, de forma que

$$p(H) = \sum_{a \in A(H)} p(a)$$

sea lo menor posible. Vamos a describir un par de algoritmos sencillos que consiguen este objetivo. Ambos son *algoritmos austeros*, como el que describimos para la coloración de un grafo: en cada paso, elegiremos la mejor decisión posible (que sea compatible con las restricciones del problema). Obsérvese que estos algoritmos servirán también para resolver la cuestión cuando se trate de un grafo sin pesos. Bastará, por ejemplo, asignar peso 1 a cada arista. En estas condiciones, todos los árboles abarcadores tienen el mismo peso (exactamente $|V| - 1$), y el resultado del algoritmo será, simplemente, uno de estos árboles abarcadores.

Algoritmo de Prim

Partimos de un grafo $G = (V, A)$ con pesos, y vamos a ir construyendo, sucesivamente, conjuntos de vértices $V_0, V_1, \ldots \subseteq V$ y de aristas $A_0, A_1, A_2, \ldots \subseteq A$ de la siguiente manera:

- Elegimos un vértice v cualquiera del grafo y formamos $V_0 = \{v\}$. El conjunto de aristas es, en este paso, $A_0 = \emptyset$.
- Examinamos todas las aristas que tienen a v como uno de sus extremos y, de entre ellas, seleccionamos la que tenga un peso menor. Digamos que es la arista a, y sea w su otro extremo. Formamos entonces $V_1 = \{v, w\}$ y $A_1 = \{a\}$.
- Supongamos que hemos construido los conjuntos V_{i-1} (que contendrá i vértices) y A_{i-1} (con $i-1$ aristas). De entre las aristas que unan vértices de V_{i-1} con vértices de $V \setminus V_{i-1}$, escogemos la de menor peso, que añadimos a A_{i-1} para formar A_i. El "nuevo" vértice se añade a los de V_{i-1} para formar V_i.

Si en el paso t no podemos encontrar una arista para continuar el procedimiento, el algoritmo acaba y la salida es el grafo $G_t = (V_{t-1}, A_{t-1})$.

Observemos que, tal como está diseñado el algoritmo, los sucesivos grafos $G_j = (V_j, A_j)$ no pueden contener ciclos (y además son conexos). Si cuando el proceso acaba hemos incluido todos los vértices del grafo, el resultado es un árbol abarcador. Sobre el hecho de que sea de peso mínimo reflexionamos más adelante. Observemos que no hemos exigido que el grafo G fuera conexo. Si lo fuera, el algoritmo produciría un árbol abarcador; pero si no, el resultado es un árbol abarcador de la componente conexa en la que se encuentre el vértice de partida v.

Algoritmo de Kruskaal

El procedimiento comienza ordenando las aristas en orden creciente de pesos: $(a_1, \ldots, a_m)$, donde $p(a_i) \leq p(a_j)$ si $i \leq j$. En cada paso se van determinando conjuntos de aristas $A_0, A_1, A_2, \ldots \subseteq A$ de la siguiente manera:

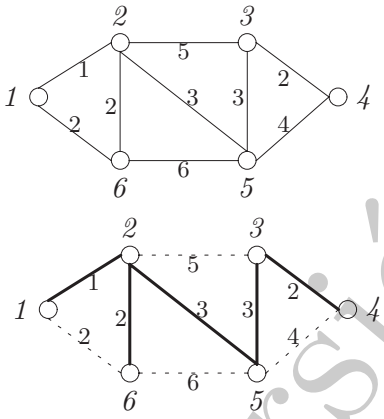
- Al principio, $A_0 = \emptyset$.
- La arista a_1 es la de menor peso del grafo, y la incluimos para formar $A_1 = \{a_1\}$.

- Supongamos que ya hemos determinado A_{i-1}. El conjunto A_i será $A_{i-1} \cup \{a_i\}$ si es que el grafo $(V, A_{i-1} \cup \{a_i\})$ no contiene ciclos. Es decir, si la adición de la arista a_i no hace que se forme un ciclo en el grafo producido hasta el momento. En caso contrario, $A_i = A_{i-1}$.

El algoritmo acaba cuando cuando A_i contenga $|V| - 1$ aristas o bien cuando hayamos examinado todas las aristas (esto es, en el paso $i = m$). Obsérvese que, a diferencia del algoritmo de Prim, los sucesivos grafos producidos por este procedimiento no tienen por qué ser árboles (aunque sí bosques). Si el grafo de partida es conexo, entonces acaba necesariamente en un árbol abarcador (aunque el que sea de peso mínimo no es tan claro).

Observaciones

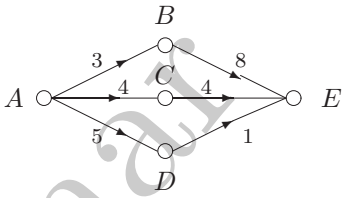
Empecemos insistiendo en que, si G es conexo, ambos algoritmos producen árboles abarcadores. Así que estos procedimientos sirven de comprobación de la conexión de un grafo arbitrario. En el de Prim, los sucesivos grafos son siempre árboles, mientras que en el de Kruskal no necesariamente: son bosques y sólo al final, si G es conexo, tendremos con seguridad un árbol abarcador.



En el grafo con pesos que aparece a la izquierda, el algoritmo de Prim, empezando por ejemplo en el vértice 3, iría incluyendo, sucesivamente, las aristas $\{3,4\}$, $\{3,5\}$, $\{5,2\}$ y $\{2,1\}$. En el último paso, se pueden elegir tanto la $\{2,6\}$ como la $\{1,6\}$. Uno de los posibles resultados es el árbol abarcador, que es de peso mínimo, que dibujamos abajo a la izquierda. El de Prim partiría de una ordenación de las aristas según sus pesos. Podría ser, por ejemplo, la siguiente: $(\{1,2\},\{3,4\},\{2,6\},\{1,6\},\{2,5\},\{3,5\},\{4,5\},\{2,3\},\{5,6\})$. El algoritmo iría incluyendo las aristas en el orden que aparecen en esa lista, saltándose $\{1,6\}$ (pues se formaría un ciclo), y acabando con la elección de la arista $\{3,5\}$ (ya tendríamos cinco aristas, las que necesitamos). El resultado sería el mismo árbol abarcador de antes, Si cambiamos la ordenación de las aristas, respetando el que aparezcan en orden creciente de pesos, obtendríamos un árbol abarcador distinto (lo mismo ocurriría con el de Prim en cada momento en que tengamos varias elecciones posibles).

En un grafo G conexo con pesos, siempre existe, al menos, un árbol abarcador de peso mínimo, pero podría haber varios con ese peso mínimo. Piénsese, por ejemplo, en un grafo completo K_n con pesos 1 en cada una de sus aristas: hay n^{n-2} árboles abarcadores distintos, y todos tienen el mismo peso, $n-1$. Los dos algoritmos que aquí hemos descrito pertenecen a los que hemos dado en llamar la clase de los *algoritmos austeros*, porque en cada etapa toman la decisión óptima, la más "barata" en este caso, que sea compatible con las restricciones del problema (en el de Prim, mantener en cada paso el carácter de árbol, en el de Kruskal, que no se formen ciclos). No siempre estas estrategias "austeras" garantizan que el resultado obtenido sea el mejor posible (recuérdese el algoritmo de coloreado de grafos). Pero, en este caso, se puede demostrar que, dado un grafo G conexo con pesos, el resultado de ambos

ambos algoritmos, tanto el de Prim como el de Kruskaal, es un árbol abarcador de peso mínimo (si hay varios, producen uno de ellos).



El siguiente ejemplo nos ilustra sobre las limitaciones que pueden tener métodos *localmente* óptimos. En el grafo dirigido de la derecha, queremos encontrar un camino barato de A a E, siguiendo el sentido de las flechas. Un análisis "global" (por ejemplo, enumerar todos los posibles caminos y determinar cuál es el más barato de todos ellos) nos llevaría a la respuesta correcta: $A \to D \to E$. Pero si procediéramos "austeramente", partiendo de A elegiríamos la de menor peso, la que lleva a C. Una vez tomada esta decisión, ya no hay marcha atrás: no nos queda más alternativa que escoger la arista que lleva a B. El camino sería $A \to B \to E$, que no es el mejor posible.

EJERCICIOS.

9.2.1 *Sean H y G dos grafos con los mismos vértices y de forma que toda arista de H lo sea también de G. Probar que $p_G(x) \leq p_H(x)$, para todo x, x natural.*

Sugerencia. Observar que en G hay más prohibiciones (habrá menos coloraciones válidas).

9.2.2 *Deducir del ejercicio anterior que si G es un grafo conexo con n vértices entonces*

$$p_G(x) \leq x(x-1)^{n-1}, \text{ para cada } x \text{ natural}.$$

Sugerencia. Por ser conexo, G tiene al menos un árbol abarcador.

9.2.3 *Calcular el número de árboles abarcadores distintos de un grafo isomorfo a $K_{3,s}$.*

Sugerencia. Construir los árboles abarcadores a la manera del caso $K_{2,s}$. Es decir, si llamamos a, b y c a los tres vértices de la izquierda, se deben formar tres conjuntos de vértices A, B y C (los vértices que quedan unidos en el árbol abarcador a a, b y c, respectivamente) de forma que $A \cup B \cup C = \{1, \ldots s\}$ pero de manera que estos tres conjuntos estén conectados. Y obsérvese que ahora hay dos maneras de realizar esta conexión, a través de un único vértice o a través de dos vértices distintos. Con los resultados de $K_{2,s}$ y $K_{3,s}$, parecería que el resultado general para $K_{r,s}$ sería $s^{r-1}\, r^{s-1}$. Nótese que los papeles de r y s son intercambiables.

Solución. $s^2\, 3^{s-1}$

9.2.4 *Consideremos el grafo que se obtiene al tomar n triángulos con exactamente un vértice común. (El número total de vértices es $2n+1$ y el número de aristas es $3n$.) ¿Cuántos árboles abarcadores tiene?*

Sugerencia. Hay que quitar una arista en cada triángulo.

Solución. 3^n.

9.2.5 *Sea H un árbol abarcador de peso mínimo de un grafo ponderado G. Sea f una arista de G que no está en H. Y sea e cualquier arista del camino (único) en H que une los vértices de f. Probar que el peso de e es menor o igual que el peso de f.*

Sugerencia. Obsérvese que si no se cumpliera esa condición podríamos escoger un árbol abarcador con menor peso.

9.2.6 *Sea $\mathcal{G}$ el grafo con vértices $\{a, b, 1, 2, \ldots, 10\}$ (12 vértices) y aristas $\{\{a, j\}, j = 1, 2, \ldots, 10\}$ y $\{\{b, j\}, j = 1, 2, \ldots, 10\}$ (20 aristas). Las aristas que tienen a a como extremo pesan 1 y las aristas que tiene a b como extremo pesan 2. ¿Cuál es el peso mínimo de entre los árboles abarcadores de $\mathcal{G}$?*

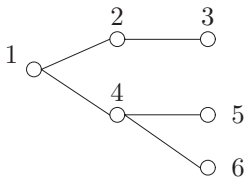
Sugerencia. Hay que mantener una (y sólo una) arista de las que llegan a b.

Solución. 12.

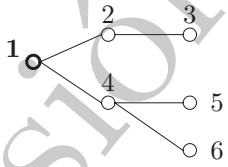
9.3. Árboles con raíz y diseño de algoritmos

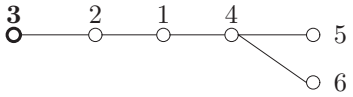
En la sección 9.1 y hablamos de árboles con raíz. se trata de un árbol en el que designamos un vértice especial, la **raíz**, que sirve de origen de coordenadas. Algunos de los algoritmos de optimización que hemos visto anteriormente producen estructuras de este tipo, donde la raíz es el vértice en el que comienza el algoritmo. En un árbol con raíz, los vértices se agrupan en niveles:

$$\begin{array}{lcl}
\text{Nivel } 0 & = & \{\text{raíz}\} \\
\text{Nivel } 1 & = & \{\text{vecinos de la raíz}\} \\
\text{Nivel } 2 & = & \{\text{vecinos de los vértices del nivel } 1\} \setminus \{\text{raíz}\} \\
\text{Nivel } 3 & = & \{\text{vecinos de los vértices del nivel } 2\} \setminus \{\text{Nivel } 1\} \\
\vdots & & \vdots \\
\text{Nivel } j & = & \{\text{vecinos de los vértices del nivel } j-1\} \setminus \{\text{Nivel } j-2\} \\
\vdots & & \vdots
\end{array}$$



Llamaremos a, la **altura** del árbol, al máximo nivel no vacío. Es importante recordar que el valor de a depende de la raíz elegida. Por ejemplo, si partimos del árbol que aparece dibujado a la izquierda, cualquiera de sus vértices puede servir como raíz. Elegir, por ejemplo, el vértice 1 o el 3 lleva a que la altura del árbol sea 2 ó 4, como sugieren los siguientes esquemas:





Definamos algunos conceptos:

- Los **descendientes** de un vértice v son los vértices del nivel siguiente al de v que sean vecinos suyos (al vértice v se le dice **progenitor** de sus descendientes).
- Un vértice es **hoja** de un árbol con raíz si no tiene descendientes.
- Un árbol con raíz será **q-ario** si cada progenitor tiene exactamente q descendientes (es decir, el número de descendientes es 0 si el vértice es hoja y q si es progenitor). Será **casi q-ario** si el número de descendientes de cada vértice está comprendido entre 0 y q.

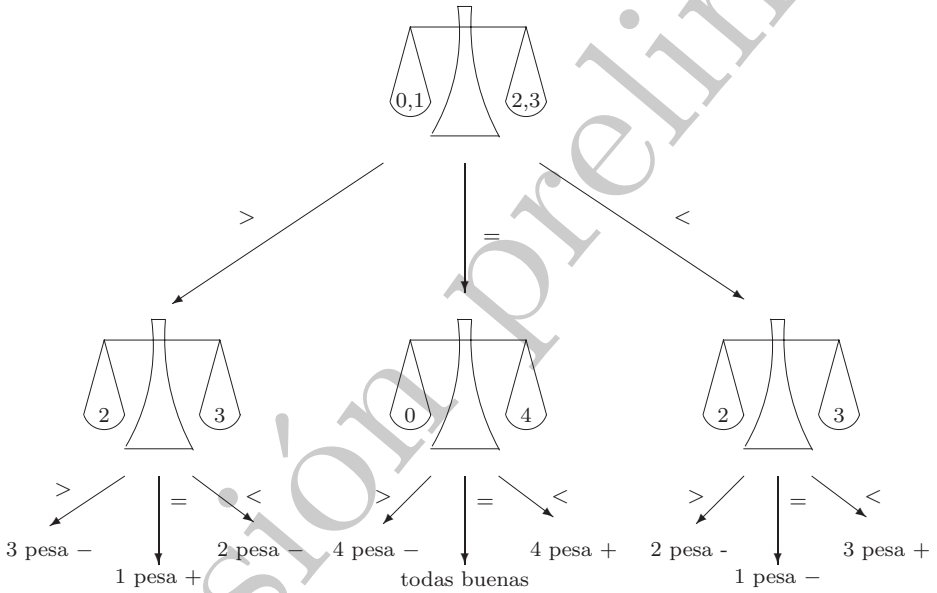
Los parámetros que manejaremos en un árbol con raíz serán

- el número de vértices, n;
- la altura del árbol, a;
- el número de hojas, h;
- y el tipo de árbol, definido por el entero positivo q (podrá ser q-ario o casi q-ario).

La importancia de estos árboles con raíz radica en que se utilizan para representar algoritmos en los que intervienen operaciones binarias (o q-arias) sucesivas. Veamos algunos ejemplos.

EJEMPLO 9.3.1 *Un árbol de decisión.*

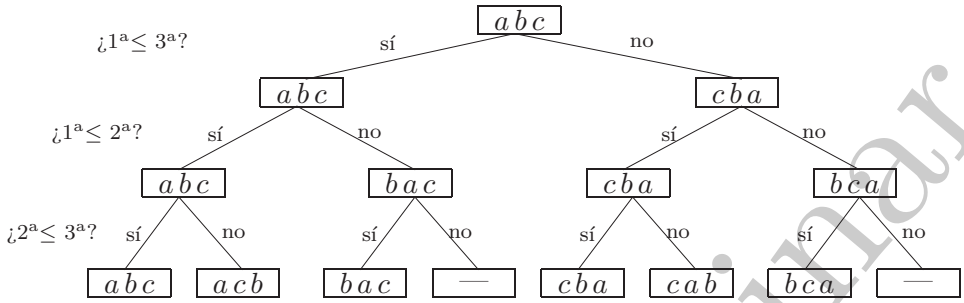
Tenemos 4 monedas, $\{1, 2, 3, 4\}$, y, a lo sumo, una de ellas no tiene el peso correcto (no es legal). Disponemos además de una moneda patrón, la 0, con el peso correcto. El problema es el siguiente: con una balanza, que tiene tres resultados posibles, queremos averiguar de la manera más económica posible (con menos usos de la balanza) cuál es la no legal. Podemos, por supuesto, comparar sucesivamente la moneda patrón con las otras cuatro. Este procedimiento emplea, en el peor de los casos, cuatro pesadas para obtener la respuesta (aunque a veces la podamos obtener con menos). Diseñemos un procedimiento alternativo, el que se recoge en el siguiente esquema:



Los símbolos casi se explican por sí mismos: $<$ significa que la balanza se vence hacia la derecha, $>$ hacia la izquierda, e $=$, se queda equilibrada. El árbol que hemos diseñado es ternario ($q = 3$), tiene altura $a = 2$ (el número de pesadas) y número de hojas $h = 9$. Lo fundamental en este caso es que el algoritmo funciona porque el árbol tiene 9 hojas, justo el número de respuestas posibles (1 pesa más, 2 pesa menos, etc). ♣

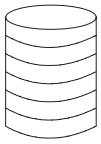
EJEMPLO 9.3.2 *Un algoritmo de ordenación.*

Dada una lista de tres números cualesquiera, (a, b, c), queremos ordenarlos con comparaciones binarias. Diseñamos el siguiente algoritmo: en el primer paso, comprobamos si el primer elemento es menor o igual que el tercero. Si la respuesta es sí, los dejamos tal como están; si es no, los permutamos. En el segundo, miramos si el primero es menor o igual que el segundo, y de nuevo procedemos como antes, dependiendo de la respuesta. Por último, en el tercer paso investigamos si el segundo es menor o igual que el tercero (observamos que aquí hay respuestas que son incompatibles con las obtenidas anteriormente). Si representamos este proceso en un árbol, obtenemos

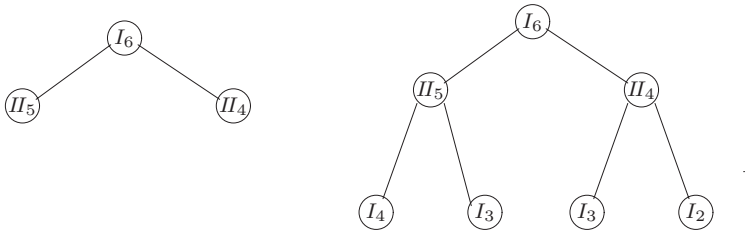


El árbol así construido es casi binario, con $a = 3$ y $h = 6$, que de nuevo coincide con el número de resultados posibles (las $3! = 6$ posibles ordenaciones). Podríamos preguntarnos si se podrían ordenar estos tres números, con comparaciones binarias, en menos pasos, por ejemplo dos. Veremos que no, porque en un árbol casi binario se cumplirá que $h \leq 2^a$; si a es 2, obtenemos que $h \leq 4$, y con ese número de hojas no podríamos cubrir todos los resultados posibles, que recordamos que eran 6. ♣

EJEMPLO 9.3.3 *El juego de Nim con seis monedas.*

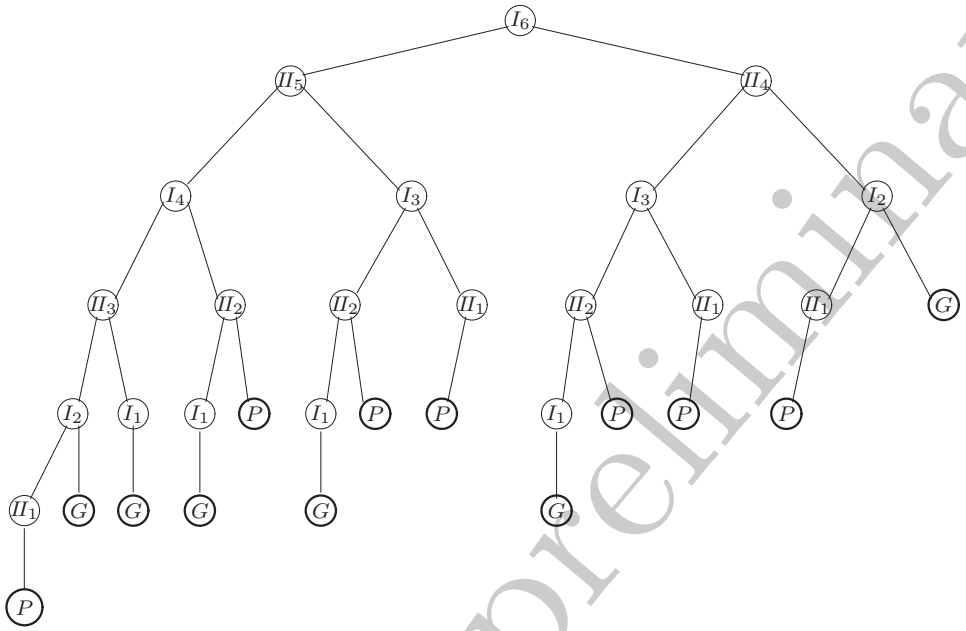


Consideremos el siguiente juego: de un montón inicial de monedas, dos jugadores, que llamaremos I y II, van retirando, alternativamente, una o dos monedas del montón de seis. Gana el jugador que retira las últimas monedas de la mesa. Ésta es una de las muchas variantes del llamado **juego de Nim**, que analizaremos detalladamente más adelante (sección 9.4). Lo que ahora nos interesa es que podemos describir los posibles desarrollos de la partida con árboles: representaremos cada configuración de monedas con un vértice, que etiquetaremos con un I o un II, dependiendo de cuál sea el jugador al que le toca jugar, y un número, para recordar cuántas monedas quedan en el montón. Así, para el montón de partida que dibujamos a la derecha, el primer vértice vendrá etiquetado con un I_6, para recordar que juega I y que hay seis monedas en el montón. El jugador I tiene dos opciones, quitar una o dos monedas; estas dos acciones vendrán representadas por aristas, hacia la derecha si quita dos monedas, hacia la izquierda si quita una. Para cada una de estas elecciones, el jugador II se encuentra con dos configuraciones distintas y en cada una de ellas puede tomar, a su vez, dos decisiones:



Obsérvese que, en cada paso, lo que obtenemos es un árbol. Además, como en cada movimiento se reduce el tamaño del montón, el juego es finito y, por tanto, el árbol que representa los diferentes desarrollos del mismo también lo será. Para el caso que nos ocupa, un montón inicial de seis monedas, el árbol completo del juego es el que aparece debajo de

estas líneas. Como preparación al análisis que haremos más adelante, etiquetamos las hojas del árbol con G si en esa hoja el jugador I ha ganado la partida y con P si ha perdido:

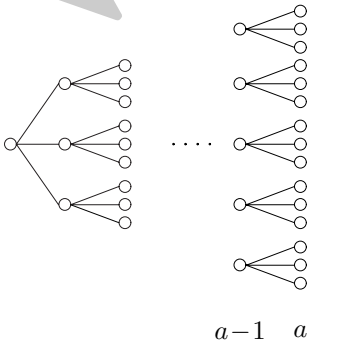


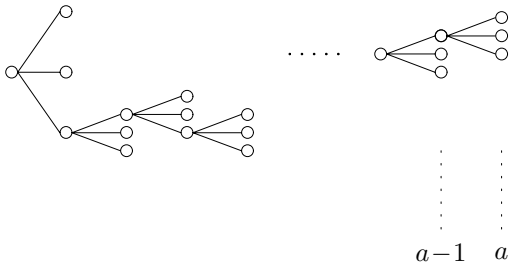
♣

9.3.1. Relación entre a y h en un árbol q-ario

En los ejemplos anteriores, la clave para que los algoritmos (representados por árboles q-arios, o casi q-arios) funcionaran era que el número de hojas cubriera todas las posibles respuestas. Estos dos parámetros, la altura y el número de hojas, no son independientes, y las cotas que obtendremos nos permitirán establecer estimaciones *a priori* sobre, por ejemplo, el mínimo número de pasos que puede tener un cierto algoritmo.

Supongamos fijados a y q. Queremos estimar el número de hojas que puede tener un árbol con raíz con esas características. Parece que la configuración con mayor número de hojas es aquélla en la que *todas* las hojas están en el último nivel. En el otro extremo, el árbol q-ario con menor número de hojas (para a fijo) sería aquél en el que las hojas van apareciendo lo antes posible. Los dos siguientes dibujos representan ambas situaciones:

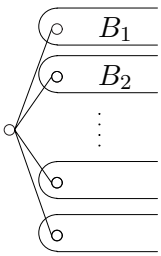




Esto nos sugiere, por un lado, que $h \leq q^a$. Para la otra situación extrema, como en cada nivel, desde el 1 hasta el $a-1$, aparecen $q-1$ hojas nuevas y en el nivel a hay q hojas, sospechamos que $h \geq (a-1)(q-1)+q = (q-1)a+1$.

Proposición 9.8 *En todo árbol con raíz casi q-ario, $h \leq q^a$.*

DEMOSTRACIÓN. Nótese que hemos relajado las condiciones sobre el tipo de árbol, basta con que sea casi q-ario (por supuesto, un árbol q-ario es también casi q-ario; pero no al revés, en general).



Lo probaremos por inducción en a, la altura del árbol. Si $a = 1$, es claro que $h \leq q$. Sea entonces un árbol A casi q-ario con raíz y llamemos B_i a los árboles que tienen como raíz a los vecinos de la raíz del árbol A. Los B_i son más pequeños (en altura) que A y son todos árboles con raíz casi q-arios. La hipótesis de inducción nos diría que para los B_i se tiene, llamando a_i y h_i a la altura y el número de hojas de cada sub-árbol B_i, respectivamente, que

$$h_i \leq q^{a_i} \quad \text{(la misma } q \text{ para todos)}.$$

Nótese que hay a lo sumo q sub-árboles B_i, y que $a_i \leq a-1$, para cada i (el peor caso correspondería a los sub-árboles que se extendieran hasta altura $a-1$). Así que

$$h = \sum_i h_i \leq \sum_i q^{a_i} \leq q q^{a-1} = q^a\,.$$

■

En el otro sentido, se puede probar la siguiente cota:

Proposición 9.9 *En todo árbol con raíz q-ario, $h = s\,(q-1)+1$, donde s es el número de vértices interiores del árbol (esto es, con descendientes).*

Como $a \geq s$ (porque en cada nivel, desde el 0 hasta el $a-1$, ha de haber al menos un vértice interior), deducimos que $h \geq a\,(q-1)+1$, como afirmábamos antes. La prueba de este resultado la dejamos como ejercicio (véase el ejercicio 9.3.1).

EJEMPLO 9.3.4 *Apliquemos las cotas obtenidas al problema de la balanza, pero ahora con r monedas (en lugar de cuatro), de las cuales a lo suma una es falsa, y la moneda patrón.*

Es claro que el número de posibles resultados es $2r+1$, porque para cada moneda hay dos posibles (pesa más o menos de lo legal) y hay un resultado extra, que es que todas sean legales. Un algoritmo que permita detectar la moneda falsa debe recoger todos estos resultados. Si lo representamos por un árbol ternario, la altura será el número máximo de pesadas necesarias para alcanzar todos los posibles resultados; y el número de hojas debe poder cubrir todos los resultados. Como $h \leq 3^a$, necesitaremos

$$2r+1 \leq 3^a \quad \Longrightarrow a \geq \log_3(2r+1).$$

Recordemos que con r pesadas siempre lo podemos hacer; aquí comprobamos que, sea cual sea el algoritmo, si está basado en comparaciones ternarias, entonces requerirá, como mínimo,

un número de pasos del orden de $\log r$. Por ejemplo, si $r = 4$, como en el ejemplo que proponíamos, tendremos que

$$r = 4 \quad \Longrightarrow \quad a \geq \log_3 9 = 2\,,$$

así que al menos se requieren dos pesadas. De la misma manera obtendríamos que para r entre 3 y 13 necesitaríamos al menos necesitaremos 3 pesadas. Y se requerirían cuatro pesadas, al menos, para cualquier r entre 14 y 40. Pero éstas son cotas teóricas; otra cosa, por supuesto, es diseñar un algoritmo que permita hacerlo con ese número mínimo de pesadas. ♣

Ejemplo 9.3.5 *Apliquémoslas al algoritmo para ordenar n números con comparaciones binarias.*

El número de resultados posibles es $n!$, así que necesitaremos que

$$2^{\#\text{ pasos}} \geq n! \Longrightarrow \#\text{ pasos} \geq \log_2(n!)$$

Por ejemplo,

$$\begin{aligned} n = 3 \quad &\longrightarrow \quad \#\,\text{pasos} \geq \log_2(3!) = \log_2(6) \Longrightarrow \#\text{pasos} \geq 3 \\ n = 4 \quad &\longrightarrow \quad \#\text{pasos} \geq \log_2(4!) = \log_2(24) \Longrightarrow \#\text{pasos} \geq 5 \\ n = 5 \quad &\longrightarrow \quad \#\text{pasos} \geq \log_2(5!) = \log_2(120) \Longrightarrow \#\text{pasos} \geq 7 \end{aligned}$$

Utilizando las estimaciones sobre el tamaño de $n!$ que obtuvimos en la subsección 2.4.4, podemos asegurar que

$$\#\text{ pasos} \geq \log_2(n!) > \log_2\left(\left(\frac{n}{e}\right)^n\right) = n\,\log_2\left(\frac{n}{e}\right)\,.$$

♣

Ejemplo 9.3.6 *Contemos el número de hojas en el juego de Nim.*

En el ejemplo que hemos descrito, el juego de Nim con seis monedas, podemos contar en el árbol que hay exactamente 13 hojas. Aunque hemos etiquetado las hojas sólo con G y P, dependiendo de si el jugador I gana o pierde, en realidad cada hoja representa un posible final del juego (y de hecho, a cada hoja le corresponde un desarrollo del juego distinto). Si el juego de Nim parte de n monedas, es fácil convencerse de que la altura del árbol (que es casi-binario) es siempre n, porque el camino de mayor longitud en el árbol es aquél en el que vamos quitando una sola moneda en cada turno (la rama de la izquierda), así que podemos estimar

$$\#\,\{\text{ posibles partidas en un Nim con } n \text{ monedas}\} = \#\{\text{hojas del árbol}\} \leq 2^n\,.$$

También podemos observar que el último nivel sólo tiene una hoja, así que podemos mejorar la estimación. Si llamamos P_n al número de hojas que tiene el árbol del juego de Nim con n monedas tendremos que

$$P_n \leq 2^{n-1} + 1\,.$$

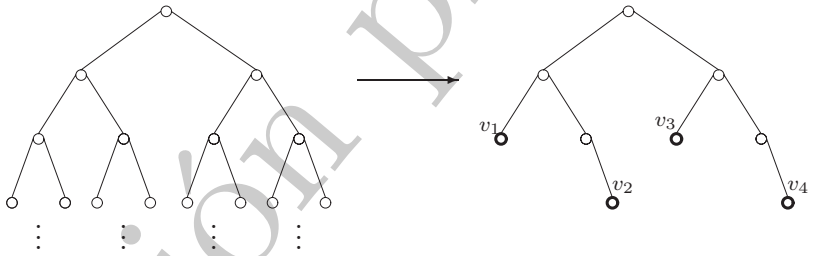
Pero en realidad podemos calcular explícitamente el número de hojas que hay. Para ello, fijémonos en que un árbol de Nim con n monedas se compone de un árbol de Nim con $n-1$ monedas (el árbol que encontramos a partir del descendiente de la raíz que está a la izquierda) y otro con $n-2$ monedas (el de la derecha). De acuerdo, los papeles de I y II están intercambiados, pero eso no afecta al número de hojas que tenga cada árbol. Por supuesto, el número de hojas del árbol total es la suma del número de hojas que tenga cada subárbol, es decir,

$$P_n = P_{n-1} + P_{n-2} \quad n \geq 2\,,$$

nuestra conocida ecuación de Fibonacci. Los valores iniciales son $P_1 = 1$ y $P_2 = 2$; así que el número de hojas de un juego de Nim con n monedas es justamente el número de Fibonacci F_{n+1}. ♣

9.3.2. La poda de un árbol binario

Tomemos el árbol binario (con raíz) infinito. En él vamos a seleccionar k vértices de manera que *ninguno de ellos sea antecesor de ningún otro*. Llamemos $v_1, \ldots, v_k$ a estos vértices señalados, que estarán a alturas (generaciones) $h_1, \ldots, h_k$. Observemos que esta elección de vértices se corresponde con una *poda* del árbol en la que las hojas superivientes están a alturas $h_1, \ldots, h_k$. En la siguiente figura hemos escogido cuatro vértices, $v_1, \ldots, v_4$:

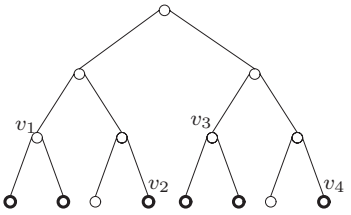


Exigir esta condición hace pensar que los vértices seleccionados no pueden estar todos muy "arriba" en el árbol, es decir, que los h_j no pueden ser arbitrariamente pequeños. De manera más precisa, y esto es lo que nos interesa comprobar, en una poda cualquiera, los h_j correspondientes han de cumplir que

$$\sum_{j=1}^{k} 2^{-h_j} \leq 1\,.$$

Veámoslo: hemos seleccionado unos vértices $v_1, \ldots, v_k$ que están a alturas $h_1, \ldots, h_k$. Llamemos $H = \max\{h_1, \ldots, h_k\}$ a la mayor altura a la que encontramos alguno de los vértices señalados.

Vamos ahora, a partir de esta elección inicial, a seleccionar un conjunto de vértices en el árbol infinito con el siguiente procedimiento: tomamos el vértice v_1 y escogemos sus 2^{H-h_1} descendientes que están en la generación H (quizás solo el propio v_1, si es que es uno de los que vive en esa máxima generación). Esto lo hacemos, sucesivamente, con el resto de los vértices $v_2, \ldots, v_k$. Nótese que los vértices que seleccionamos de esta manera son todos distintos.



En el dibujo que mostramos a la izquierda tenemos un ejemplo de este procedimiento. Los vértices escogidos en primer lugar son v_1, v_2, v_3 y v_4: dos de la segunda generación y otros dos de la tercera. La máxima altura es $H = 3$. En trazo más grueso hemos señalado los vértices de esta tercera generación que seleccionamos en segundo lugar. Si sumamos los números 2^{-h_j} para cada vértice v_j obtenemos $1/4+1/4+1/8+1/8 = 3/4$, un número no mayor que 1. El cálculo análogo, pero ahora para los vértices seleccionados en segundo lugar, nos lleva al mismo resultado.

Llamemos A a la elección inicial de vértices y A' a la nueva, y sea $D(v_j)$ el conjunto de vértices descendientes de v_j que escogemos. Entonces

$$\sum_{v\in A'} 2^{-h(v)} = \sum_{j=1}^{k} \sum_{v\in D(v_j)} 2^{-h(v)} = \sum_{j=1}^{k} 2^{-H} \underbrace{|D(v_j)|}_{=2^{H-h_j}} = \sum_{j=1}^{k} 2^{-h_j} = \sum_{v\in A} 2^{-h(v)}\,.$$

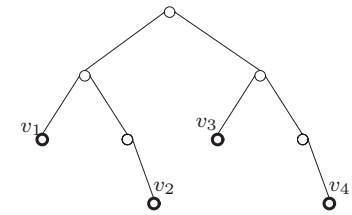
El valor de la suma es el mismo para A que para A'. Pero claro, en A' hay, a lo sumo, 2^H vértices, así que

$$\sum_{j=1}^{k} 2^{-h_j} = \sum_{v\in A'} 2^{-h(v)} = |A'|\,2^{-H} \leq 2^H\,2^{-H} = 1\,.$$

Lo interesante del asunto es que el resultado es cierto en el otro sentido: si damos unos números $h_1, \ldots, h_k$ tales que $\sum 2^{-h_j} \leq 1$, entonces en el árbol binario (infinito) podemos señalar k vértices $v_1, \ldots, v_k$ de alturas respectivas $h_1, \ldots, h_k$ y de manera que ninguno sea antecesor de ningún otro. Llamamos, como antes, $H = \max\{h_1, \ldots, h_k\}$. Y observamos que

$$1 \geq \sum_{j=1}^{k} 2^{-h_j} = 2^{-H} \sum_{j=1}^{k} 2^{H-h_j} \quad \Longrightarrow \quad 2^H \geq \sum_{j=1}^{k} 2^{H-h_j}\,.$$

Ahora, en el árbol binario infinito, en la generación H vamos señalando, de izquierda a derecha y sucesivamente, 2^{H-h_1} vértices, luego 2^{H-h_2}, etc. La desigualdad anterior nos dice que en esa generación encontramos suficientes vértices como para completar el proceso. El paso final consiste en señalar a v_1 como el vértice de la generación h_1 que tiene a los primeros 2^{H-h_1} vértices seleccionados como descendientes en la generación H, a v_2 como el que tiene a los 2^{H-h_2} siguientes, etc. El lector podrá comprobar que el argumento funciona de manera análoga para el caso de un árbol q-ario general (sustituyendo el 2 por q en la desigualdad).



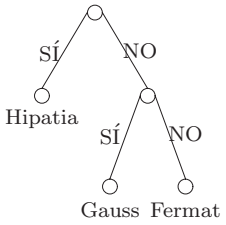
Vamos ahora a interpretar este resultado etiquetando con listas de ceros y unos los vértices seleccionados. Digamos que elegir, en el árbol binario infinito, la rama de la izquierda es un 1, y elegir la de la derecha un 0. Seleccionar un vértice de altura $h \geq 1$ no es, entonces, sino dar una lista de ceros y unos de longitud h. En el ejemplo que hemos estado considerando, al vértice v_1 le correspondería la lista $(1, 1)$, al vértice v_2 la lista $(1, 0, 0)$, y a los vértices v_3 y v_4, las listas $(0, 1)$ y $(0, 0, 0)$, respectivamente. Elegir, como en

el ejemplo, vértices con la restricción de que unos no sean antecesores de otros se traduce en que ninguna de las listas correspondientes puede coincidir con el comienzo de ninguna otra.

Más adelante (en la sección 11.2) llamaremos a esta restricción *condición de prefijo* y veremos que será fundamental a la hora de construir ciertos códigos. Veamos ahora otro ejemplo en el que esta condición de prefijo es relevante.

EJEMPLO 9.3.7 *El problema de los cuestionarios.*

Tenemos k objetos que se pueden identificar con una sucesión de preguntas cuya respuesta es un SÍ o un NO.



Por ejemplo, podríamos tener tres personajes, tres ilustres matemáticos: Hipatia de Alejandría, Gauss y Fermat. Preguntamos primero si el personaje es mujer, y posteriormente si es alemán. El resultado es el árbol que aparece a la izquierda. Con la identificación habitual de 1 para el SÍ y 0 para el NO, a Hipatia le corresponde la lista (1), a Gauss la $(0,1)$ y a Fermat la $(0,0)$. Las tres listas cumplen la condición de prefijo. Esto es algo general: si queremos que el cuestionario realmente identifique a los k objetos, las listas de ceros y unos que codifiquen las posibles respuestas han de cumplir esta condición. ♣

EJERCICIOS.

9.3.1 *Probar la proposición 9.9.*