

Diseño de Algoritmos

Grado en Matemáticas - UCM - curso 2020-2021

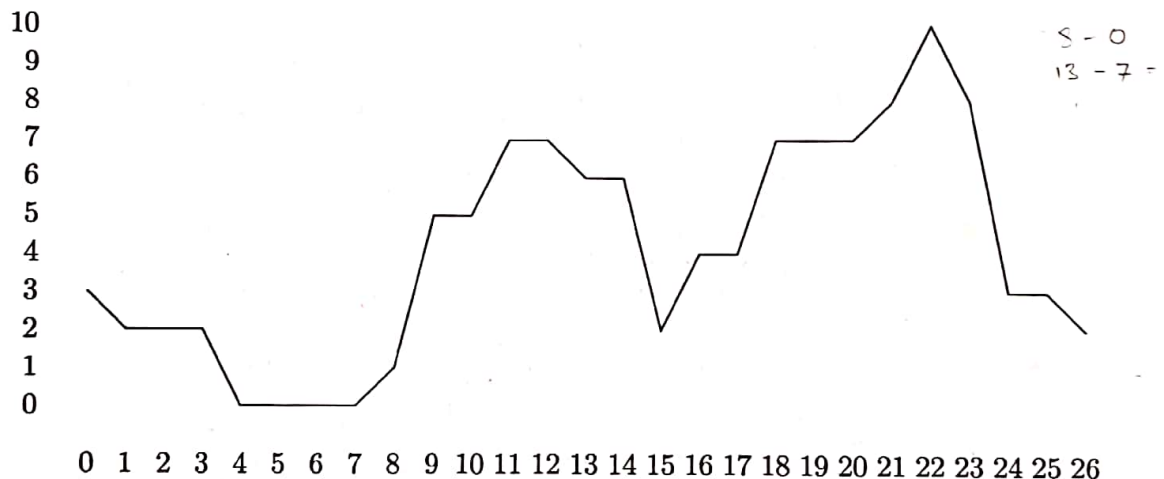
Examen 8 de julio de 2021

Especificación y diseño de programas iterativos (3 pt)

Kilian Korretonet está preparando su gran travesía por los Alpes. Ha representado su plan de travesía mediante una secuencia de valores naturales que indican las alturas de los puntos por los que tiene que pasar. De esta forma, la travesía está formada por *fases ascendentes* (secuencia no vacía de valores no decrecientes) alternando con *fases descendentes* (secuencia no vacía de valores no crecientes).

En una fase ascendente se pueden distinguir varios *tramos* (secuencia estrictamente creciente con al menos dos valores y maximal, es decir, no se puede extender por ningún extremo) separados por *llanos* (secuencia de valores iguales con al menos dos valores y maximal).

Por ejemplo la secuencia: $[3, 2, 2, 2, 0, 0, 0, 0, 1, 5, 5, 7, 7, 6, 6, 2, 4, 4, 7, 7, 7, 8, 10, 8, 3, 3, 2]$ representa una travesía con una fase descendente de longitud 8 que comprende los valores desde el primer 3 hasta el cuarto 0 ambos inclusive, seguido de una fase ascendente de longitud 9 que comprende los valores del primer 0 hasta el último 7, ambos inclusive. Sigue una fase descendente de longitud 5, otra fase ascendente de longitud 8, y una última fase descendente de longitud 5. Por lo tanto, la fase ascendente más larga tiene longitud 9 y tiene dos tramos: $[0, 1, 5]$ y $[5, 7]$ (y tres llanos: $[0, 0, 0, 0]$, $[5, 5]$ y $[7, 7]$).



Dado un vector v no vacío de enteros no negativos:

- Define un predicado $ascenso(v, p, q)$ que devuelva cierto si y solo si el segmento $v[p..q]$ es una fase ascendente.
- Define un predicado $tramo(v, p, q)$ que devuelva cierto si y solo si el segmento $v[p..q]$ es un tramo (recuerda que los tramos son maximales).
- Utilizando los predicados anteriores, especifica una función que dado un vector no vacío de enteros no negativos devuelva la longitud de la fase ascendente más larga, el comienzo y el final de una fase ascendente de longitud máxima, y el número de tramos en dicha fase.
- Diseña e implementa un algoritmo iterativo que resuelva el problema especificado.
- Escribe el *invariante* del bucle que permite demostrar la corrección del mismo y proporciona una *función de cota*.
- Indica el *coste asintótico* del algoritmo en el caso peor y justifica adecuadamente tu respuesta.

Métodos de programación

1. [2 pt] En cualquier travesía por el desierto un buen suministro de agua es fundamental para poder llegar al destino. La compañía Solyarena, especialista en viajes en grupo a lugares desérticos ha desarrollado una aplicación que le permite detectar cuándo debe racionar el agua que proporciona a los excursionistas. Para ello cuenta con una estimación de la cantidad de agua disponible en el depósito al comenzar cada día de travesía. El objetivo es encontrar el primer día en el que debe aplicarse un racionamiento del agua porque la cantidad de agua estimada es insuficiente para dar a cada miembro del grupo un litro por cada día que resta de viaje. También debe indicarse el caso en que no haya que aplicar racionamiento.

Por ejemplo, si la travesía dura 7 días y solo hay un excursionista, para la secuencia de estimación: [20, 15, 10, 5, 2, 1, 0], el primer día de racionamiento será el quinto (cuando solo quedan 2 litros para tres días). En cambio, para 5 días y 3 excursionistas, si la estimación es [20, 15, 12, 9, 5] no habrá racionamiento.

- a) Asumiendo que los datos de entrada son: el número de días que el grupo permanecerá en el desierto, el número de integrantes del grupo (estos dos valores son mayores que cero) y un vector del tamaño del número de días con la estimación de los litros de agua que quedarán cada día en el depósito (son valores enteros y la estimación cada día decrece al menos tanto como el número de integrantes del grupo), escribe un algoritmo recursivo *eficiente* que permita resolver el problema.
 - b) Escribe la *recurrencia* que corresponde al coste de la función recursiva indicando claramente cuál es el *tamaño del problema*. Indica a qué *orden de complejidad asintótica* pertenece dicho coste.
2. [3 pt] Un grupo de M amigos se ha propuesto descender en canoa el río Cares/Deva. La agencia de alquiler dispone de N canoas, todas ellas de las mismas características. Cada canoa puede transportar una, dos o tres personas, siempre y cuando no se superen k kilos. Se conocen los pesos (en kilos) de cada miembro del grupo de amigos. Para ahorrar costes, se quiere alquilar el número mínimo de canoas, pero además del peso limitado hay otro problema: las rencillas establecidas entre los amigos. Dos amigos enemistados (las enemistades son mutuas) no pueden compartir canoa. Estas enemistades vienen dadas mediante una matriz (simétrica) de valores lógicos (cierto si hay enemistad, falso si no la hay).
 - a) Define el *espacio de soluciones* e indica cómo es el *árbol de exploración*.
 - b) Diseña un algoritmo de *vuelta atrás* que resuelva el problema. Explica claramente los *marcadores* que has utilizado.
 - c) Establece una *estimación* del valor alcanzable desde una solución parcial, para poder hacer podas adicionales e incorpórala a tu algoritmo.
 3. [2 pt] Supongamos ahora que las canoas son biplazas y que los amigos han resuelto todas sus rencillas.
 - a) Encuentra una estrategia *voraz* para resolver el problema del alquiler de canoas. Demuestra su *corrección*.
 - b) Implementa la estrategia con un algoritmo e indica su *coste*.

Tiempo: 3 horas.

NOTAS:

- Para expresar los algoritmos, utiliza el pseudo-código utilizado en clase o notación similar.
- Procura explicar lo que haces.
- Es altamente recomendable cuidar la presentación. Una mala presentación puede motivar que la profesora no entienda el contenido del examen.