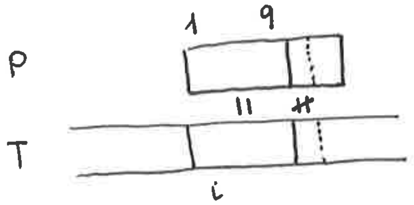
KMP ... ¡ES FACIL! (Una presentación muy gráfica)

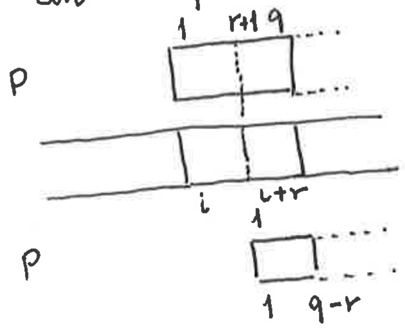
Introducción

- Datos $T[1..n]$, $P[1..m]$
- Estamos tratando de encajar P a partir de una cierta posición i



Tras comprobar que $P[1..q] = T[i..q+i-1]$ vemos que allí P no encaja, pues $P[q+1] \neq T[i+q]$. Ahora el algoritmo ingenuo buscaría el encaje a partir de $i+1$, y además repetiría las comparaciones desde 0. Pero si utilizamos adecuadamente cierta información previamente recopilable sobre P, ¡podemos ir mucho más deprisa!

- Supongamos que P encaja en T a partir de cierta posición $i+r$ con $r<q$. Observamos en el gráfico, que como ya sabíamos que

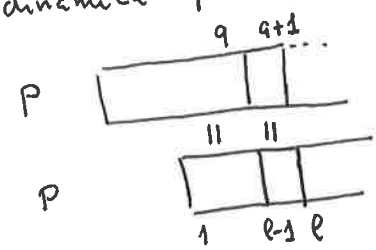


$P[r+1..q] = T[i+r..i+q-1]$
ahora deberíamos tener para empezar
$P[r+1..q] = P[1..q-r]$
lo cual significa que el prefijo P_{q-r} de P (y también de P_q) es el sufijo $_{q-r}(P_q)$ de dicha secuencia P_q.

De manera que si para cada q tuviesemos calculado el mayor prefijo tal y éste fuese P_{q-r}, sabríamos con seguridad dos cosas:

1) P no puede encajar en T en ninguna posición $i+s$ con $s<r$.
2) Para comprobar si P encaja en T a partir de $i+r$ podemos empezar comparando $T[i+q]$ con $P[q-r+1]$

- Pongamonos entonces a calcular la información requerida sobre P. Denotaremos por $\Pi : 1..m \rightarrow 0..m-1$ la función que calcula el correspondiente valor de $q-r$. Veamos que podemos utilizar programación dinámica para ello:



Si P_ℓ es sufijo de P_{q+1} entonces: i) $P[q+1] = P[\ell]$; ii) $P_{\ell-1}$ es sufijo de P_q

Por otra parte, ser prefijo y ser sufijo son relaciones transitivas, por lo que su intersección también lo es.

Al haberse añadido la condición i) no tenemos la garantía de que $P_{\Pi(q)+1}$ la cumpla, pero si no fuera así debido a la transitividad antes indicada, debería tratarse de un prefijo y sufijo estricto de la misma, por lo que deberíamos probar con $P_{\Pi^2(q)+1}$, reiterando el proceso hasta encontrar un j que verifique $P[\Pi^j(q)+1] = P[q+1]$, tomando entonces $\Pi(q+1) = \Pi^j(q)+1$. Naturalmente, si ni al llegar a 0 tenemos $P[1] = P[q+1]$, deberíamos tomar $\Pi(q+1) = 0$.

- El cálculo de Π se hace pues por medio de un doble bucle, el primero variando q de 1 a m y el segundo calculando los correspondientes valores necesarios $\Pi^j(q)$. Un cálculo apresurado de su coste nos facilitaría un coste $O(m^2)$, pero una aplicación más reposada de lo aprendido sobre complejidad amortizada reduce de inmediato el coste a $O(m)$.
- Una vez tenemos computado Π el proceso de búsqueda reiterada de P en T va comparando los correspondientes elementos de P y T:
  - Se prosigue mientras resultan iguales, y de alcanzarse el final de P nos anotamos un exito.
  - Cada vez que se produce un fallo nos mantenemos en la misma posición de T, pero "avanzamos" P según nos indica Π.
  - Cada vez que se produce un éxito es fácil ver que de cara a buscar el siguiente, también debemos "avanzar" el patrón según indica Π.
- El algoritmo de búsqueda reiterada va avanzando entonces en T (cada vez que una comparación de un carácter de P y T tiene éxito y cada vez que Π devuelve 0), pero en los demás casos vuelve a comparar el mismo carácter de T con otros de P.
- De nuevo el cálculo apresurado del coste produciría $O(mn)$ pero la aplicación de las mismas técnicas de coste amortizado que antes reducen el coste a $O(n)$.

Ver algoritmos en Cormen et. al. 3ª ed pgs. 1005 y 1006.