61. Considereu el problema d'emmagatzemar n llibres als prestatges de la biblioteca. L'ordre dels llibres és fixat pel sistema de catalogació i, per tant, no es pot canviar. Els llibres han d'aparèixer a les prestatgeries en l'ordre designat. Les prestatgeries d'aquesta biblioteca tenen amplada L i són regulables en alçada. Un llibre b_i, on $1 \leq i \leq n$, té gruix t_i i alçada h_i. Una vegada es decideix quins llibres es fiquen a un prestatge s'ajusta la seva alçada a la del llibre més alt que col·loquem al prestatge. Doneu un algorisme que ens permeti col·locar els n llibres a les prestatgeries de la biblioteca de manera que es minimitzi la suma de les alçades dels prestatges utilitzats.

Una solució

No se puede modificar el orden de los libros, lo único que podemos hacer es poner mas o menos libros en un estante para poder ajustar la altura del estante al libro más alto en él.

Para encontrar una solución recursiva observemos que, en una solución óptima tenemos los i primeros libros en el primer estante y, en el resto, una solución de altura óptima para el resto de libros, $i{+}1, \ldots, n$. Como subproblema tenemos el mismo enunciado, pero con los libros todavía no colocados.

Sea $H[i]$ = la altura mínima que necesitamos para colocar en estantes los libros i a n. Queremos calcular $H[1]$.

El único requisito que tiene la asignación es que los libros que se asignen a un estante quepan en él. Anchura total $\leq L$. Utilizo un valor lógico $cabe(i, j)$, $i \leq j$ que indica si los libros i a j caben en un estante.

Para calcular $H[i]$, $1 \leq i \leq n$, $k \leq i \leq n$ tenemos que decidir los elementos que van en el primer estante, y ajustar el coste de la posible solución. Tenemos la recurrencia

$$H[i] = \begin{cases} \max_{i \leq j \leq n} h_j & cabe(i, n) \\ \min_{cabe(i,j), i \leq j < n} \{\max_{i \leq k \leq j} h_k + H[j+1]\} & \text{si no} \end{cases}$$

Implementando esta expresión, tenemos $O(n)$ elementos con coste $O(n^3)$ por elemento, este último coste es debido a los rangos del mínimo y máximo y al cálculo del predicado. El coste total $O(n^4)$ con espacio adicional $O(n)$.

Podemos mejorar el coste del algoritmo si precalculamos los valores que aparecen en la recurrencia y las sumas prefijadas de las anchuras:

- $P[[i] = \sum_{1 \leq \ell \leq i} t_\ell$
- $M[i, k] \max_{i \leq j \leq k} h_j$

de manera que solo accedemos a la posición necesaria. $cabe(i, j)$ es equivakente a $P[j] - P[i-1] \leq L$.

Así la recurrencia queda

$$H[i] = \begin{cases} S[i, n] & cabe(i, n) \\ \min_{cabe(i,j), i \leq j < n} \{M[i, j] + H[j+1]\} & \text{si no} \end{cases}$$

Utilizamos $P[i] = \sum_{1 \leq \ell \leq i} s_\ell$, con $P[0] = 0$, P se puede calcular recursivamente como

$$P[i] = \begin{cases} 0 & i = 0 \\ P[i-1] + s_i & i > 0 \end{cases}$$

De forma similar podemos calcular los valores de M

$$M[i,j] = \begin{cases} h_i & j = i \\ \max\{M[i,j-1], h_j\} & j > i \end{cases}$$

El coste de la precomputación es $O(n^2)$ y esto nos permite calcular la recurrencia con memoización en tiempo $O(n)$ por elemento (el rango del min). Lo que nos da un coste total de $O(n^2)$. Podemos implementar iterativamente la recurrencia realizando un recorrido desde la posición n hasta el inicio, con el mismo coste.

En un caso real es de esperar que el número de libros de la bibiloteca sea grande, pero que el ancho de los estantes sea un valor constante. Si asumimos que L es constante, podemos también asumir que el número máximo de libros que podemos colocar en un estante es $O(L)$, dependiendo de la unidad de medida.

En este caso sabemos que $cabe(i,j)$ solo puede ser cierto para valores de en los que el número de libros es constante. Bajo este punto de vista el número total de llamadas por elemento es $O(L)$ y el coste total $O(n)$. Además los únicos valores significativos de $M[i,j]$ o $cabe(i,j)$ están en el rango $j \in \{i, \ldots, i + cL\}$, en total $O(L)$ por cada i, precalcuando solo esos valores el coste total es $O(n)$.

Si queremos encontrar la forma óptima de hacer la división, como siempre, almacenaremos además la matriz de punteros que nos permita reconstruir la solución.