Exercici 4 (3 punts) CinemaVis ha de programar l'aparició d'un seguit d'anuncis a una pantalla gegant a la Plaça del Mig la diada de Sant Jordi. La tirada d'anuncis es pot iniciar a temps 0 (l'inici programat) però mai abans. A més, CinemaVis disposa d'un conjunt de n anuncis per fer la selecció. L'anunci i té una durada de 1 minut i té associat dos valors reals no negatius t_i i b_i. L'anunciat pagarà b_i euros a CinemaVis si l'anunci i s'emet a l'interval $[0, t_i]$ i 0 euros si s'emet després. Cap dels n anuncis es pot mostrar més d'una vegada. CinemaVis vol projectar la selecció d'anuncis que li proporcioni màxim benefici. Dissenyeu un algorisme, el més eficient que podeu, per a resoldre aquest problema.

Solució: Tal y como dice el enunciado cada anuncio dura exactamente un minuto. Teniendo esta restricción en cuenta, en cualquier solución podemos asumir que los tiempos de inicio son enteros, si no es así podemos retrasar los tiempos de inicio al entero anterior sin perder beneficios. Además puesto que los anuncios solo se pueden proyectar una vez, es suficente con considerar los tiempos de inicio de proyección de anuncios en los valores 0 a $n-1$. Si en una solución óptima hay algún anuncio emitido a tiempo $t \geq n$ por fuerza queda algún hueco en la planificación en alguno de los instantes 0 a $n-1$. En ese caso podemos retrasar el tiempo de emisión de los anuncios después de hueco. Observemos que el beneficio solo puede aumentar.

El algoritmo primero preprocesará los tiempos de entrada, asignado al anuncio i el entero $\lfloor t_i \rfloor$ ($O(n)$). Luego ordenará en orden decreciente ($O(n \log n)$). Utilizando el orden podemos asignar a cada anuncio el tiempo $y_i \in \{0, \ldots, n-1\}$ mayor en el intervalo $[0, \lfloor t_i \rfloor]$ con coste $O(n)$, recorrido de mayor a menor.

Notemos que a efectos de nuestro algoritmo tenemos construida una entrada donde cada anuncio tiene asignado b_i y y_i y se obtiene beneficio b_i si el anuncio se emite en $\{0, 1, \ldots, y_i\}$. Además sabemos que $0 \leq y_i < n$.

Ahora buscamos la planificación en $\{0, \ldots, n-1\}$ que nos proporcione mayor beneficio. Para ello primero ordenamos los anuncios en orden decreciente de beneficio b_i ($O(n \log n)$). Tratamos los anuncios en este orden. Inicialmente no hay anuncio planificado para ningún instante de tiempo. Asigno el anuncio i a el instante $y_i - 1$, si no hay ningún anuncio ya planificado para este tiempo. En caso contrario busco el mayor tiempo anterior j que esté libre, si $j \leq 0$ planifico anuncio i a tiempo j. En caso contrario el anuncio no se mostrará.

La segunda parte del algoritmo tiene coste $O(n^2)$ ya que para cada anuncio tengo que mirar en caso peor n posibles tiempos. Así, el coste total del algoritmo es $O(n^2)$.

Porqué es correcto? Supongamos que una asignación óptima s no sigue este criterio y sea s' la solución del algoritmo voraz propuesto. Tratando los elementos en orden decreciente de b_i's.

Sea j el primer anuncio en el que las dos planificaciones difieren.

Si s' no muestra j, de acuerdo con el algoritmo, en todos los valores $\{0, \ldots, y_j$ se ha planificado un anuncio anterior. Como para valores mayores de j las dos planificaciones coinciden, s tampoco muestra j. Nos quedan dos casos por considerar:

- s no muestra j y s' muestra i a tiempo $t = s'(j)$. A tiempo t, s puede no emitir un anuncio o emitir un anuncio k ($k > j$). En el primer caso podemos emitir j a tiempo t en s y aumentar el beneficio (s no sería óptimo). En el segundo caso emitiendo j en vez de k a tiempo t no podemos disminuir el beneficio ya que $b_j \geq b_k$.
- s no muestra j a tiempo $t = s(j)$ y s' muestra i a tiempo $t' = s'(j)$. Como las planificaciones coinciden hasta j, y el algoritmo lo emite en el instante mas tardío en el que es posible emitirlo, $t' < t$. De nuevo a tiempo t', s puede no emitir un anuncio o emitir un anuncio k ($k > j$). En el primer caso podemos emitir j a tiempo t' en s y aumentar el beneficio. En el segundo caso emitiendo j a tiempo t' y k a tiempo t, seguimos dentro de los intervalos con beneficio y obtenemos exactamente el mismo beneficio.

Aplicando reiteradamente los cambios mencionados o bien llegamos a lo conclusión que s no es solución óptima o, manteniendo el beneficio, transformamos s en s'. Por lo que concluimos que el algoritmo es correcto.

Comentarios sobre soluciones entregadas

- No se puede usar t_i como índice de una vector, es un valor real.
- El algoritmo propuesto aplicado sobre tiempos t_i no enteros puede eliminar soluciones óptimas. Supongamos que tenemos la entrada [pares $a_i = (t_i, b_i)$]: $a_1 = (1.5, 10)$, $a_2 = (2, 3)$. El algoritmo propuesto planifica a_1 a tiempo 0.5. Esta decisión no permite emitir a_2 con beneficio no nulo, ni antes ni después de a_1.
- Ordenar por orden decreciente de b_i/t_i y asignarlo a tiempo $t_i - 1$ si no está ocupado no resuelve el problema. Considera la entrada $a_1 = \cdots = a_{20} = (20.3, 2)$, $a_{21} = \cdots = a_{82} = (59.5, 5)$ que está ordenada por b_i/t_i. El algoritmo emitiría 20 del primer tipo (tiempos 0 a 19) y 38 del segundo tipo (tiempos 20 a 58) beneficio menor que si emitiésemos 59 del segundo tipo.
- Considerar el tiempo creciendo ($0 \rightarrow$) y emitir el anuncio posible que proporciona mayor beneficio no es correcto. Si la entrada es $a_1 = (1, 3)$, $a_2 = (2, 6)$, $a_3 = (2, 2)$. Se emitiría a_2 a tiempo 0 y a_3 a tiempo 1 que no es óptimo.
- Considerar el tiempo decreciendo ($\max t_i \rightarrow 0$) y emitir el anuncio posible que proporciona mayor beneficio proporciona el criterio correcto. Dependiendo de como se implemente el iterador sobre el tiempo, el coste del algoritmo puede ser o no polinómico. Si se itera desde $t = \max t_i$ hasta 0 con $t--$ el número de iteraciones es exponencial.