Algorísmia QP 2019–2020

Examen final online **10 de Junio de 2020**

Duración: 2h 30mn

Instruccions generals:

- Heu de lliurar les respostes en un únic arxiu pdf a través del Racó. No oblideu posar el vostre nom. Indiqueu clarament la pregunta que esteu responent.
- Cal que argumenteu la correctesa i l'eficiència dels algorismes que proposeu. Per això podeu donar una descripció d'alt nivell de l'algorisme amb les explicacions i aclariments oportuns que permetin concloure que l'algorisme és correcte i té el cost indicat. La puntuació dependrà fortament d'aquesta argumentació.
- Podeu fer crides a algorismes que s'han vist a classe, però si la solució és una variació n'haureu de donar els detalls.
- Es valorarà especialment la claredat i concisió de la presentació.
- La puntuació total d'aquest examen és de **10 punts**.
- Assumim que us comprometeu a realitzar aquest control vosaltres mateixos sense consultar cap tipus de material ni els companys.

Ejercicio 1 (3 puntos) (SOS Rural). Considerad un camino rural en el Pirineo con casas muy dispersas a lo largo de él. Por motivos de seguridad se quieren colocar estaciones SOS en algunos puntos de la carretera. Los expertos indican que, teniendo en cuenta las condiciones climáticas de la zona, se debería garantizar que cada casa se encuentre como máximo a una distancia de 15km, siguiendo la carretera, de una de las estaciones SOS.

Proporcionad un algoritmo eficiente que consiga este objetivo utilizando el mínimo número de estaciones SOS posibles. Justificad la corrección y el coste de vuestro algoritmo e indicad la complejidad en tiempo de la solución propuesta.

Ejercicio 2 (3.5 puntos) (Emparejando runs) La programación dinámica puede utilizarse para resolver un problema en animación gráfica denominado "morphing": convertir una imagen en otra pasando a traves de una secuencia de imágenes con transiciones suaves.

Para simplificar supongamos que tenemos dos vectores $A[1:n]$ y $B[1:n]$ donde cada $A[i]$ o $B[i]$ es 0 (blanco) o 1 (negro). A y B representan lineas de pixeles en una imagen B/W y queremos ver como transformar una en otra.

Los 0s y 1s definen una serie de *runs*, subcadenas maximales de 1's contiguos que no se pueden extender con más unos. Para identificar un run utilizaremos dos índices $[a;b]$ dónde a es la posición del vector en la que se inicia el run y b su longitud, el número total de unos en el run. Los runs están ordenados de izquierda a derecha por posición de inicio. Utilizaremos R para referirnos a los runs de A y S para referirnos a los runs de B.

Por ejemplo si tenemos los siguientes vectores

$$A = 0111100011101101$$
$$B = 1010100111001110$$

A tiene 4 runs, $R_1 = [2;4]$,$R_2 = [9;3]$, $R_3 = [13;2]$ y $R_4 = [16;1]$. B tiene 5 runs $S_1 = [1;1]$,$S_2 = [3;1]$,$S_3 = [5;1]$,$S_4 = [8;3]$ y $S_5[13;3]$.

Nuestro objetivo es buscar un "emparejamiento"optimo entre todos los runs de los dos vectores. Hay tres posibilidades a la hora de emparejar runs:

Match: Emparejar un run $[i;k]$ de A con un run en $[j;\ell]$ de B: el coste de ese emparejamiento viene dado en función de la diferencia de longitudes y la diferencia en los puntos de inicio, y se calcula como
$c_{match}([i;k];[j;\ell]) = |k-l| + |i-j|$.

Fusión: Emparejar p runs consecutivos de A, $[i_1;k_1]$, $[i_2;k_2]$, $\ldots$, $[i_p;k_p]$, ($i_q + k_q - 1 < i_{q+1}$ para todo $1 \leq q < p$), con un run $[j;\ell]$ de B. El coste en este caso depende del número de runs, la diferencia entre longitudes ℓ (la del run en B) y $i_p + k_p - i_1 + 1$

(la que incluye los runs en A) y la diferencia entre los inicios y se calcula como $c_{fusion}([i_1; k_1], \ldots, [i_p; k_p]; [j; \ell]) = p + |\ell - (i_p + k_p - i_1)| + |i_1 - j|$

Fisión: Emparejar un run $[i; k]$ de A con p runs consecutivos de B. Es la misma operación que Fusion, invirtiendo los papeles de A y B. El coste asociado viene determinado por la misma función intercambiando los roles. El coste es $c_{fision}([i; k]; [j_1; \ell_1], \ldots, [j_p; \ell_p]) = p + |k - (j_p + \ell_p - j_1)| + |i - j_1|$

En ningún caso, si un run en las posiciones $[i, k]$ de A se ha emparejado con un run $[j, \ell]$ de B, puede haber un run $[i', k']$ de A con $i' > i$ emparejado con un run $[j', \ell']$, $j' < j$, en B. Además, todos los runs tienen que quedar emparejados en alguna de las operaciones seleccionadas.

Un posible emparejamiento de los dos vectores del ejemplo es

- Fisión de R_1 con S_1, S_2, S_3.
- Match de R_2 con S_4
- Fusión de S_5 con R_3, R_4.

Este emparejamiento tiene coste

$$\begin{aligned} &c_{fision}([2; 4]; [1; 1], [3; 1], [5; 1]) + c_{match}([9; 3]; [8; 3]) + c_{fusion}([13; 3], [13; 2]; [16; 1]) \\ &\quad = 3 + |4 - (5 + 1 - 1)| + |2 - 1| \\ &\qquad + |3 - 3| + |9 - 8| \\ &\qquad + 2 + |3 - (16 + 1 - 13)| + |13 - 13| \\ &\quad = 5 + 1 + 3 = 9 \end{aligned}$$

Proporcionad un algoritmo de PD para encontrar un emparejamiento de runs con coste mínimo, dados A y B. Justificad la corrección y el coste de vuestro algoritmo e indicad la complejidad en tiempo y en espacio de la solución propuesta.

Exercici 3 (3.5 punts) (SafeGym). ExtremeGym tiene que replanificar los circuitos de entrenamiento extremo incorporando las nuevas restricciones de seguridad sanitaria.

Para ello ha dividido los espacios en salas con capacidades limitadas. Algunas de estas salas se han conectado entre ellas a través de pasillos unidireccionales, debidamente aislados, en los que también se debe limitar el número de personas que los transitan.

Los técnicos de la empresa han diseñado un grafo en 5 niveles que define los posibles circuitos de entrenamiento. Un entrenamiento se inicia y finaliza en la zona de acceso al recinto (con capacidad ilimitada) que tiene acceso directo a todas las salas del primer nivel y desde todas las salas del último nivel. El entrenamiento finaliza tras realizar, en el tiempo estipulado, las rutinas asociadas a 5 salas, una en cada nivel. Los pasillos entre salas permiten acceder de una sala de entrenamiento en un nivel a otra (u otras) en el nivel siguiente.

ExtremeGym quiere un algoritmo eficiente para determinar el nivel de restricción que puede aplicar a la capacidad real de salas y pasillos que permita mantener un flujo mínimo de M personas realizando un entrenamiento en el gimnasio.

Diseñad un algoritmo eficiente para resolver el siguiente problema:

Dados el grafo de niveles y la capacidad de salas y pasillos, determinar si para algún valor entero no negativo $k \geq 0$, se puede limitar la capacidad de cada sala y cada pasillo en un factor 2^k manteniendo un flujo de M entrenamientos. En el caso de que este valor exista proporcionar el valor de k más grande que permita el número de personas entrenando deseado.