Cognoms, Nom Data

Algorísmia QP 2018–2019

Examen parcial **10 d'Abril de 2019**

Durada: 1h 45m

Instruccions generals:

- L'exercici 1 s'ha de resoldre fent servir l'espai reservat per a cada resposta.
- Heu d'argumentar 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.
- Heu de justificar totes les vostres afirmacions, en cas contrari la nota de la pregunta serà 0.
- 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.
- Entregueu per separat les vostres solucions de cada bloc d'exercicis (Ex 1, Ex 2 i Ex 3).
- La puntuació total d'aquest examen és de **10 punts**.

Exercici 1 (4 punts)

(a) (0.5 punts) Es poden ordenar n enters amb valor entre 0 i $n^3 - 1$ amb temps lineal.

Solució: Cert, Considereu els enters com nombres amb 2 dígits en radix n. Cada digit tindrà un rang entre 0 a $n-1$. Ordeneu aquests enters amb 2 dígits utilitzant counting i RADIX. Temin el nombre de passos és $T(n) = \Theta((n+b)d) = \Theta(2(n + \log_n(n^3 - 1)) = \Theta(n)$.

(b) (0.75 punts) Com sabeu, l'algorisme de Quicksort pot trigar $O(n^2)$ en ordenar un vector $A[1, \ldots, n]$ (si els elements ja estan ordenats). Podríeu fer una petita modificació de Quicksort, de manera que el temps pitjor sigui com a màxim $O(n \lg n)$?

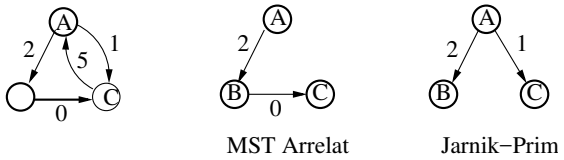
Solució: Podem modificar Quicksort fent que a cada crida de Partició, el lloc d'agafar com a pivot la clau mes a la dreta, utilitzem l'algorisme lineal explicat a classe i agafem com a pivot la mediana dels elements sota consideració. De manera que a cada crida recursiva, dividim l'entrada un dues meitats gaire be iguals i la recurrència sempre es $T(n) = 2T(n/2) + \Theta(n) = O(n \lg n)$.

(c) (0.5 punts) Suposem que implementem una seqüència d'n operacions sobre una estructura de dades on la i-èsima operació té un cost $c_i = i$, si i és una potència de 2, altrament el cost és 1. Utilitzeu el mètode de l'estalviador (accounting) per calcular el cost amortitzat per operació. Per exemple, el si $1 \leq i \leq 5$ els valors de c_i: $1, 2, 1, 4, 1$.

Solució: Mireu el problema 50 de la llista 2.

(d) (0.5 punts) Donat un digraf fortament connex $\vec{G} = (V, \vec{E}), w : \vec{E} \to \mathbb{R}^+$, volem trobar el MST a $\vec{G}$ (i.e. un arbre d' expansió, arrelat a un vèrtex $v \in V$ amb camins dirigits de v a tots els altres nodes), on la suma dels pesos de les arestes de l'arbre sigui minimal respecte a qualsevol altre arbre d'expansió a $\vec{G}$. És cert que es pot adaptar l'algorisme de Jarnik-Prim per a trobar el MST, al problema de trobar un MST arrelat a un graf dirigit?. Si la resposta és afirmativa, doneu un argument informal per a justificar-ne la certesa, altrament doneu un contraexemple.

Solució Jarnik-Prim: FALS, l'algorisme no sempre produeix una solució correcta com es pot veure a la figura.



(e) (0.75 punts) Tenim un text S que utilitza un alfabet $\Sigma = \{a, b, c\}$ on les freqüències amb què apareix cada símbol són respectivament f_a, f_b, f_c. Volem comprimir S codificant Σ amb l'algorisme de Huffman. Per a cadascun dels casos a sota, digueu quins serien els valors de f_a, f_b, f_c que ens donarien la codificació proposada, alternativament expliqueu per què el codi proposat no es pot obtenir.

i. $\{0, 10, 11\}$
ii. $\{0, 1, 11\}$
iii. $\{10, 01, 00\}$

Solució

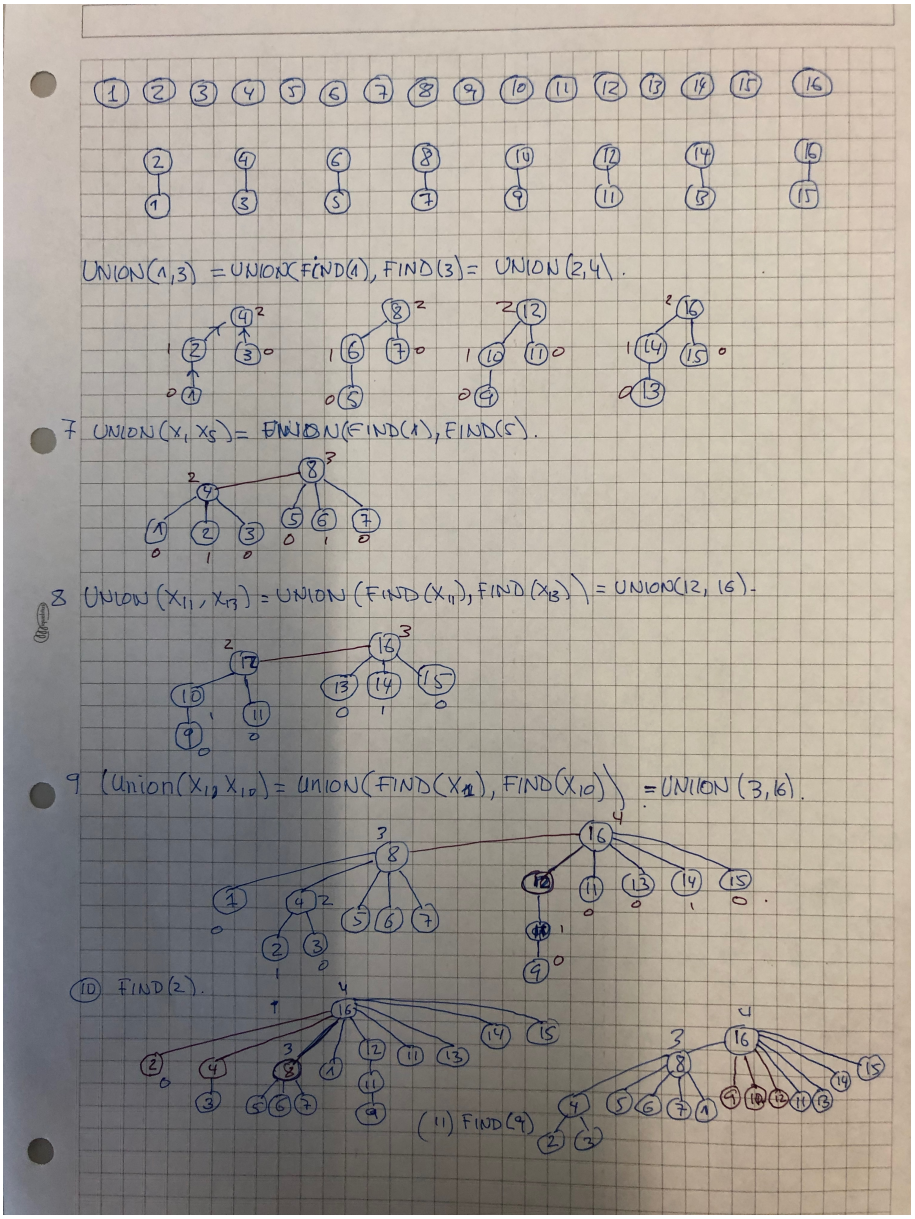
i. $f_a = 1/2$, $f_b = 1/4$ i $f_c = 1/4$
ii. no és prefixat
iii. No es pot, un d'ells hauria de ser codificat amb un bit (no és òptim)

(f) (1 punt) Sigui donat un conjunt $S = \{1, 2, \ldots, 16\}$. Doneu l'estructura de dades després dels passos 1, 3, 5, 7, 8, 9, 10 i l'arbre final resultant del pas 11, quan apliquem union-find al conjunt S implementat amb les heurístiques d'unió per rang i compressió de camí:

(1) **de** $i = 1$ **fins** 16
MAKESET (x_i)
(3) **de** $i = 1$ **fins** 15 **by 2**
UNION (x_i, x_{i+1})
(5) **de** $i = 1$ **fins** 13 **by 4**
UNION (x_i, x_{i+2})
(7) **UNION** (x_1, x_5)
(8) **UNION** (x_{11}, x_{13})
(9) **UNION** (x_1, x_{10})
(10) **FIND** (x_2)
(11) **FIND** (x_9)

Digueu quin es el rang de l'arrel i de cada vèrtex, a l'arbre final. (Quan feu UNION de dos arbres que tenen l'arrel amb el mateix rang, pengeu l'arbre esquerra de l'arbre de la dreta).

Solució


UNION(1,3) = UNION(FIND(1), FIND(3)) = UNION(2,4).
7 UNION(X1, X5) = UNION(FIND(1), FIND(5)).
8 UNION(X11, X13) = UNION(FIND(X11), FIND(X13)) = UNION(12, 16).
9 (Union(X12, X10) = UNION(FIND(X12), FIND(X10)) = UNION(3,16).
10 FIND(2).
(11) FIND(9).

Exercici 2 (3 punts) (Agenda)

A la vostra agenda teniu una llista L de totes les tasques que heu de completar en el dia de avui. Per a cada tasca $i \in L$ s'especifica la durada $d_i \in \mathbb{N}$ que indica el temps necessari per a completar-la i un factor de penalització $p_i \in \mathbb{Z}^+$ que n'agreuja el retard. Heu de determinar en quin ordre realitzar totes les tasques per obtenir el resultat que menys penalització total acumuli.

Tingueu en compte que:

- en un instant de temps només podeu realitzar una única tasca,
- una vegada comenceu a fer una tasca, heu de continuar-la fins a finalitzar-la, i
- s'han de completar totes les tasques.

El criteri d'optimització és la penalització total que s'acumula. La penalització real associada a una tasca $i \in L$ és el temps de finalització t_i de la seva realització, multiplicat per la seva penalització p_i. El temps de finalització t_i es correspon al temps transcorregut des de l'inici de la jornada laboral (és a dir, des de l'instant de temps 0) fins al moment en que s'ha finalitzat la tasca.[1]

Considereu l'algorisme voraç que programa les tasques en ordre decreixent de factor de penalització p_i. Determineu si aquest algoritme resol el problema. En cas que no ho faci, proporcioneu un algorisme (tant eficient com pogueu) per resoldre'l.

Solució

El algoritmo propuesto no es correcto. Para ello consideramos la siguiente entrada con dos tareas: $d_1 = 4, p_1 = 2, d_2 = 1, p_2 = 1$.

Si las realizamos en orden decreciente de penalización el coste es $d_1p_1 + (d_1 + d_2)p_2 = 8+5 = 13$. Si las realizamos en el orden inverso el coste es $d_2p_2+(d_2+d_1)p_1 = 1+10 = 11$. Por tanto el algoritmo propuesto no da la mejor solución posible.

Para resolver el problema ordenaremos las tareas en orden decreciente de p_i/d_i. El coste del algoritmo corresponde al de obtener la ordenación y es $O(n \log n)$.

Para demostrar que es correcto utilizaré un algoritmo de intercambio. Supongamos que en la ordenación óptima dos tareas que se realizan consecutivamente i y $i+1$ no cumplen el criterio propuesto. Es decir $p_i/d_i < p_{i+1}/d_{i+1}$.

Sea T el tiempo en el que se inicia la tarea i y M el coste debido a las tareas que no son ni i ni $i + 1$. El coste de la solución óptima que estamos analizando es

$$M + (T + d_i)p_i + (T + d_i + d_{i+1})p_{i+1}.$$

Si intercambiamos la posiciones de i e $i + 1$ el coste debido a las otras tarea y el tiempo de inicio de t_{i+1} no cambia. Así en este nuevo orden de realización el coste es

$$M + (T + d_{i+1})p_{i+1} + (T + d_{i+1} + d_i)p_i.$$

[1] Observeu que, segons les restriccions del problema, la realització d'una tasca $i \in L$ amb temps de finalització t_i haurà començat a l'instant $t_i - d_i$.

Si calculamos la diferencia entre los dos tenemos:

$$\begin{aligned} M + (T + d_i)p_i + (T + d_i + d_{i+1})p_{i+1} - [M + (T + d_{i+1})p_{i+1} + (T + d_{i+1} + d_i)p_i] \\ = & d_i p_i + (d_i + d_{i+1})p_{i+1} - d_{i+1}p_{i+1} - (d_{i+1} + d_i)p_i \\ = & d_i p_{i+1} - d_{i+1}p_i \\ < & 0 \end{aligned}$$

ya que $p_i/d_i < p_{i+1}/d_{i+1}$. Y llegamos a una contradicción con la hipótesis de que la ordenación era óptima.

Exercici 3 (3 punts) (Estiu als jardins de la UPC).

Per tal de incrementar els ingressos, el Gerent de la UPC ha decidit llogar els jardins de Torre Girona (i part del Campus Nord) per realitzar diferents activitats d'esbarjo durant la temporada d'estiu. Després d'una crida a propostes d'activitats, la UPC ha rebut una gran quantitat de sol·licituds. Cada sol·licitud indica la data i l'hora, d'inici i de fi de l'activitat, i el tipus de recursos que es necessiten per fer-la. Per a cadascuna de les sol·licituds, el servei d'Economia de la UPC ha calculat el benefici que s'obtindria si es fés l'activitat. Per tal d'evitar problemes, en aquesta primera temporada, la UPC ha decidit llogar els espais en exclusiva a les activitats. Això implica que no es poden dur a terme més d'una activitat al mateix temps. UPCNet necessita un programa que permeti obtenir una selecció d'activitats que es puguen planificar en exclusiva i que proporcionin el màxim benefici possible. Proporcioneu un algorisme, tant eficient com us sigui possible, per a resoldre el problema de UPCNet.

Solució El problema a resoldre és el problema de selecció d'activitats compatibles maximitzant el pes de les activitats seleccionades. El pes d'una activitat es el benefici que obtindria la UPC. Aquest problema està resolt amb programació dinàmica a les transparències de l'assignatura i l'algorisme té cost $O(n)$ i fa servir espai addicional $O(n)$.