Algorísmia QT 2016–2017

Una solució al Examen final **9 de Gener de 2017**

Durada: 2h 50m

Exercici 1 (3.5 punts)

(a) (0.5 punts) Siguin A i B dos conjunts d'enters. Doneu un algorisme que, amb temps esperat $O(n)$, ens digui si els dos conjunts contenen els mateixos elements.

Sol. Utilitzem una taula de hash T amb grandària $2n$ d'un bit. Inicialitzem T a 0's. Agafem una funció de hash que computable en temps constant i fem $T[h(a)] = 1$ per $a \in A$. Després apliquem la funció de hash als elements de B fins a trobar $b \in B$ amb $T[h(b)] = 0$ o tractar tots es elements. Al primer cas $A \neq B$, altrament $h(A) = h(B)$. L'algorisme the whp temps esperat $O(n)$.

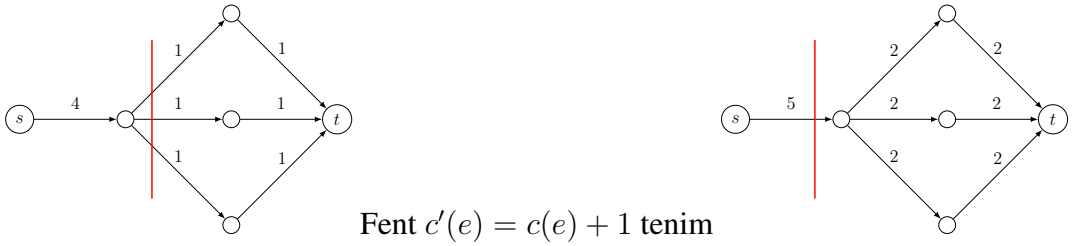
Digueu i justifiqueu si cadascuna de les afirmacions següents són certes o falses.

(b) (0.5 punts) Donat un graf $G = (V, A)$ amb $w : A \to \mathbb{Z}^+$ i un conjunt $S \subseteq V$, si $e = (u, v)$ és una aresta tal que e té el cost mínim entre S i $V - S$, digueu si l'arbre d'expansió miním (Minimum Spanning Tree) a G conté l'aresta e.

Sol. Cert, per la regla blava.

(c) (0.5 punts) Tenim una xarxa $(G = (V, E), s, t, \{c(e)\})$ amb $\forall e \in E$, $c(e) \in \mathbb{Z}^+$. Sigui (A, B) un tall mínim $s - t$. Si afegim 1 a totes les capacitats, aleshores (A, B) encara és un tall mínim $s - t$ respecte a les noves capacitats $c(e) + 1$.

Sol. Fals. Al dibuix hi ha una xarxa de fluxe que ens dona un contraexemple. Els min-cuts corresponen a la linea vermella.



Fent $c'(e) = c(e) + 1$ tenim

(d) (0.5 punts) Donat un graf dirigit $\vec{G} = (V, \vec{E})$, amb pesos a les arestes $w(e) \in \mathbb{R}$, la manera més ràpida de trobar les distàncies mínimes entre tots els parells de vèrtexs és aplicar n cops l'algorisme de Dijkstra.

Sol. Fals. Dijkstra no pot tractar arestes amb pes negatiu, en aquest cas no es resol el problema. Si els pesos son positius l'algorisme de Floyd-Warshall es mes eficient.

Doneu una explicació breu i concisa

(e) (0.5 punts) Tenim n caixes $(c_1, \ldots, c_n)$, una d'elles conté un bitllet de 500 euros i les altres estan buides. Volem trobar el bitllet obrint el mínim nombre de caixes. Considereu l'algorisme següent:

- Seleccionem $x \in \{0, 1\}$ amb distribució uniforme.
  - Si $x = 0$ obrim caixes en l'ordre $\{c_1, \ldots, c_n\}$ fins trobar el bitllet.
  - Si $x = 1$ obrim caixes en l'ordre $\{c_n, \ldots, c_1\}$ fins trobar el bitllet.

Quin es el nombre esperat de caixes que s'obren en aquest algorisme?

Sol.

Suposant que el bitllet està a la caixa k. Si $x = 0$ hemd'obrir k caixes i quan $x = 1$ $n - k + 1$. Sigui Z la v.a. = nombre de caixes obertes:

$$E[Z] = \frac{1}{2}k + \frac{1}{2}(n - k + 1) = \frac{n+1}{2}.$$

(f) (0.5 punts) Que és un *blockchain*?

Sol. Una *estructura de dades* amb llistes encadenades, on cada registre apunta al previ, però a més del punter tenim una copia dels continguts del registre previ via un hash criptogràfic.

(g) (0.5 punts) Donat un graf no-dirigit amb dos vèrtexs distingits s i t, explica com podríeu utilitzar teoria de fluxos per a trobar tots els camins disjunts entre s i t.

Sol. Construïm un graf dirigit amb les arcs en les dues direccions, si $\{u, v\} \in E$ afegim els arcs (u, v) i (v, u) excepte en el cas que $s = u$, només afegim (s, v), o $t = u$, només afegim (v, t). La capacitat de tots els arcs serà 1. Per calcular els camins disjunts es suficient aplicar l'algorisme de Ford-Fulkerson tal i com hem vist a classe.

Exercici 2 (1 punt) La suma **o-exclusiva** $\oplus$ és un operador lògic on $0 \oplus 1 = 1 \oplus 0 = 1$ i $0 \oplus 0 = 1 \oplus 1 = 0$, per tant donades cadenes de bits B_1 i B_2, $B_1 \oplus B_2$ consisteix en l'aplicació bit a bit de l'operador $\oplus$ (per exemple $10110 \oplus 00111 = 10001$). Donat un missatge M (com a cadena de bits) definim la funció h_c de la manera següent:

- Fem una partició de M en k blocs $M = B_1||B_2|| \cdots ||B_k$, (on $||$ és l'operador concatenació) i on cada bloc B_i conté exactament de 5 bits. Si la grandària de M no és múltiple de 5, afegim 0s la dreta, fins a assolir la grandària desitjada.
- $h_c(M) = B_1 \oplus B_2 \oplus \cdots \oplus B_k$.

(a) (0.25 punts) Quina és la sortida de $h_c(10101110)$?.

Sol. Els dos blocs son 10101 i 11000, per tant $h_c(10101110) = 01101$.

(b) (0.75 punts) És h_c una bona funció criptogràfica? (és a dir, té les propietats següents: (1) Per a una entrada M amb grandària qualsevol, sempre dóna una sortida amb la mateixa grandària; (2) $h_c(M)$ es pot calcular fàcilment; (3); h_c^{-1} és difícil de calcular; (4) coneixent h_c i M, és difícil calcular una M' tal que $h_c(M') = h_c(M)$; (5) si canviem un bit a M aleshores $h_c(M)$ canvia molt.)

(1) Cert. Sempre dona una sortida de grandària 5.

(2) Cert. Tenim $k = \theta(n/5)$ blocs i farem $k - 1$ $\oplus$ operacions sobre cadenes de longitud 5, $O(n)$.

(3) Cert. En general tenim 2^5 sortides mentre que hi han 2^n entrades, llavors hem de tenir una sortida a la que van a parar $2^n/2^5$ entrades.

(4) Fals. Es suficient amb intercanviar dos blocs.

(5) Fals. Si canviem 1 bit de l'entrada,només canviarà un bit de la sortida.

Exercici 3 (2,5 punts)

Tenim un programa que permet la simulació d'un sistema físic de temps discret. Volem simular tants passos del sistema com sigui possible. El nostre laboratori te accés a dos supercomputadors (A i B) capaços de processar el treball. No obstant això, són màquines compartides i no sempre poden executar els nostres treballs amb la prioritat més alta. Tant A com B poden processar el nostre programa.

Suposem que sabem, pels següents n minuts, la potència de processament disponible a cada màquina. Al minut i, podem executar a_i passos de la simulació a A o bé b_i a B. La simulació es pot transferir d'una màquina a un altra però, per fer-ho, s'ha de salvar i restaurar l'estat i això té un cost d'un minut de temps en el què no es pot fer cap progrés en la simulació. Volem un pla d'execució pels n minuts següents. Aquest pla ha de indicar, per a cada minut, A o B o "mou", i ha de ser consistent amb les restriccions donades. A més, volem que maximitzi el nombre total de passos de simulació executats.

(a) (0,5 punts) Demostreu que el següent algorisme no resol correctament el problema proposat.

```
1:  procedure PLA D'EXEC(a, b)
2:      if a[1] ≥ b[1] then
3:          s[1] = 'A'
4:      else
5:          s[1] = 'B'
6:      end if
7:      i = 2
8:      while i ≤ n do
9:          if s[i − 1] == 'A' then
10:             if b[i + 1] > a[i] + a[i + 1] then
11:                 s[i] = 'mou'; s[i + 1] = 'B'; i = i + 2
12:             else
13:                 s[i] = 'A'; i = i + 1
14:             end if
15:         else
16:             Com al cas previ canviant A/a per B/b
17:         end if
18:     end while
19: end procedure
```

Sol. Per als vectors $a =< 2, 1 >$ i $b =< 2, 20 >$, suposant que l'accés fora de rang no falla, el programa retornaria la solució $< A, A >$ que és incorrecta.

(b) (2 punts) Doneu un algorisme eficient que, donats $a_1, \ldots, a_n$ i $b_1, \ldots, b_n$, proporcioni un pla d'execució que permeti executar el màxim nombre de passos de simulació.

Sol. Para encontrar una solución analizamos la estructura de suboptimalidad de una solución óptima. Observemos que una solución óptima ejecutará pasos de simulación en el instante n, si no no seria óptima. Puede hacerlo en A on en B. Suponiendo que sea en A, el paso previo puede ser `A` o `mou`. En el primer caso la solución debe ser una solución óptima para $n-1$ pasos ejecutando en A en el paso $n-1$ y en el segundo una solución óptima para $n-2$ pasos ejecutando en B en el paso $n-2$.

Para establecer la recurrencia utilizaremos notación adicional, para $0 \leq k \leq n$:

$A(k) =$ el número máximo de pasos de simulación que podemos ejecutar en k pasos ejecutando la simulación en A en el paso k.

$B(k) =$ el número máximo de pasos de simulación que podemos ejecutar en k pasos ejecutando la simulación en B en el paso k.

Tenemos la recurrencia:

$$A(k) = a(k) + \max(A(k-1), B(k-2))$$
$$A(k) = a(m) + \max(B(m-1), A(k-2))$$

para $k \geq 2$, y los casos base $A(0) = B(0) = 0$, $A(1) = a[1]$ y $B(1) = b[1]$.

El coste de la solución óptima que buscamos es $\max(A(n), B(n))$.

Como el número total de subproblemas es $O(n)$ podemos utilizar PD. Para ello basta con un recorrido en orden creciente de las dos tablas guardando un puntero con la información de la opción de dónde proviene el valor máximo.

El coste de calcular uno de los valores de la tabla A o B es constante y por ello el algoritmo necesita tiempo $O(n)$ incluido el paso de recuperación de la solución siguiendo la información de los punteros.

Exercici 4 (3 punts) SuperFast és una empresa de transport que està intentant decidir si li interessa participar en una oferta de treball de Amazon a Barcelona. Amazon voldria subcontractar el transport de productes de proximitat a SuperFast. El transport ha de garantir uns compromisos de puntualitat molt estrictes i SuperFast vol una estimació de la quantitat de nous vehicles que hauria d'incorporar a la seva flota i una estimació del cost corresponent al seu ús. Amazon li ha proporcionat els resultats de les seves simulacions de comportament dels clients i, en particular, de les necessitats de transport diàries per uns quants dies. Per un dia es disposa d'una llista de sol·licituds de transport. Cada sol·licitud de transport especifica les coordenades GPS de l'origen i del destí d'un enviament juntament amb l'hora de recollida i d'entrega. SuperFast disposa d'un programa que li proporciona el temps necessari de desplaçament d'un vehicle entre qualsevol parell de posicions de la ciutat coneixent el temps d'inici del trasllat.

En aquest primer estudi SuperFast assumeix el cas pitjor en el què els vehicles no poden portar més d'un enviament. A més, el vehicle ha de ser a la posició d'origen al temps estipulat i no pot deixar el punt de destí fins el temps estipulat de trasllat. Així, un vehicle que transporti un enviament pot encarregar-se d'un altre sempre que el temps de desplaçament entre el destí i el nou origen li permeti arribar l'hora estipulada. També assumeix que l'estimació del temps necessari de desplaçament és acurada.

Disenyeu algorismes amb cost polinòmic que:

(a) (1,5 punts) Donats un nombre k i les sol·licituds de transport d'un dia, determini si es poden servir amb k vehicles.

Sol. Utilizando la ayuda identificaremos los vehiculos con unidades de flujo y las solicitudes con arcos con capacidad limitada con cota superior e inferior.

De las restricciones del problema sabemos que un vehículo puede servir otro envío siempre que pueda desplazarse con tiempo suficiente desde el destino al nuevo origen, asumiendo que permanece en el destino hasta la hora estipulada. Esta restricción nos indica cual es la red a considerar.

Tendremos, para cada solicitud i, dos nodos o_i y d_i y un arco (o_i, d_i) con capacidad $[1, 1]$, reflejando el hecho de que todas las solicitudes tienen que ser servidas. Añadiremos un arco (d_i, o_j) cuando saliendo del destino de la solicitud i en el tiempo estipulado podamos llegar al origen de la solicitud j en el tiempo estipulado, reflejando la observación previa.

Añadiremos tres vertices adicionales s, x, t y los arcos (s, x) con capacidad k, para forzar que el flujo máximo sea $\leq k$, i, para cada solicitud i, los arcos (x, o_i) i (d_i, t) con capacidad 1.

Notemos que la existencia de un flujo que satisface las restricciones de capacidad nos garantiza la existencia de $\leq k$ caminos que no comparten aristas de s a t que, además, cubren todos los arcos correspondientes a solicitudes. Esto es equivalente a poder servir las peticiones con k o menos vehículos.

Una vez obtenida la red de flujo, podemos determinar si hay un flujo que satisface las

restricciones en tiempo polinómico utilizando el algoritmo visto en clase para decidir la existencia de flujos con demandas y cotas inferiores.

(b) (0,5 punt) Donades les sol·licituds de transports d'un dia, determini el nombre mínim de vehicles que les poden servir ($k_{\min}$)

Sol. Siempre hay una solución con n vehículos, por ello $1 \leq k_{\min} \leq n$ podemos implementar una búsqueda binaria utilizando el algoritmo del apartado (a) que nos permite decidir si las solicitudes se pueden servir con $\leq k$. En total tendermos que hacer $O(\log n)$ ejecuciones del algoritmo del apartado (a).

(c) (1 punt) Donades les sol·licituds de transports de un dia, proporcioni una planificació per a cadascun dels $k_{\min}$ vehicles indicant, per a cada vehicle, la seqüència de sol·licituds de transport que ha de servir i el temps total de desplaçament del vehicle.

Sol. Una vez encontrado el valor de $k_{\min}$ utilizando el algoritmo del apartado (a) para este valor de k tendremos una flujo de s a t con valor del flujo $k_{\min}$ que satisface la restricción de que todas las solicitudes están servidas. Como hemos visto en clase lo único que tenemos que hacer es aplicar el algoritmo de extracción de caminos disjuntos como vimos en clae. Esto nos da $k_{\min}$ ejecuciones de BFS sobre el grafo formado por los arcos con flujo positivo.

Podeu suposar que el càlcul del temps de desplaçament entre dos posicions té un temps constant i que entre l'hora de recollida i la d'entrega hi ha temps suficient per fer-hi el desplaçament amb puntualitat.

Ajut: Podeu pensar un vehicle com una unitat de flux i una sol·licitud de transport com un arc amb fites inferiors i superiors a la seva capacitat.