Classe 16/10: notes d'aclariment

1. *Exercici 28 (b)*

   Solució

   Hi ha diverses formes correctes de resoldre el problema, p.ex.:

VERSIÓ 1. Ordeneu per rati d'stream, on el rati d'stream del paquet i-èsim és $r_i = b_i/t_i$. Envieu els paquets en ordre creixent d'aquests ratis. Per simplificar la notació, assumiu que els paquets ens venen donats en aquest ordre. Comproveu si es cumpleix que

$$\sum_{i=1}^{n} b_i \leq r \sum_{i=1}^{n} t_i.$$

Si aquesta desigualtat no es cumpleix, aleshores no hi ha cap altra planificació dels paquets que sigui vàlida. En canvi, si es cumpleix, aleshores la planificació proposada és una planificació vàlida que resol el problema.[1]

El temps d'execució de l'algorisme és $O(n)$. L'enunciat només demana decidir si existeix una planificació vàlida, i això es respon només mirant si es cumpleix la desigualtat $\sum_{i=1}^{n} b_i \leq r \sum_{i=1}^{n} t_i$. Si, a més, també volem donar com a resposta la planificació, aleshores necessitem un temps $O(n \log n)$ per ordenar els ratis.

VERSIÓ 2. De fet, no necessitem ordenar-ho tot per a produir una planificació vàlida. Per a cada paquet i, computem el seu dèficit $s_i = rt_i - b_i$. Assegurem que, si existeix una planificació vàlida, aleshores qualsevol ordenació que comenci amb tots els paquets que tenen dèficit positiu produeix una planificació vàlida. Per veure per què, observeu que una planificació és válida si la suma dels dèficits no és negativa per a cap segment inicial de l'ordenació. Començar amb tots els paquets amb dèficit positiu es crea el dèficit més gran possible abans que es comencin a afegir paquets amb dèficit negatiu. Aquesta observació ens permet crear la planificació vàlida en temps $O(n)$ sense necessitat d'ordenar, senzillament computant el signe del dèficit per a cada paquet.

Noteu que aquest argument també demostra que ordenar els paquets en ordre decreixent de dèficits també funciona, perquè així també es planifiquen els paquets amb dèficit positiu abans dels de dèficit negatiu.

---

[1] DEMO: si la desigualtat no es cumpleix, aleshores no hi ha cap altra ordenació de paquets que produeixi una planificació vàlida, perquè en un temps total de $\sum_{i=1}^{n} t_i$ necessitem enviar $\sum_{i=1}^{n} b_i$ bits, independentment de la forma com els planifiquem.

Si la planificació proposada envia massa bits per a un període de temps $[0,t]$ qualsevol, aleshores la desigualtat tampoc serà certa. Per veure això, considereu un temps t, i sigui i el paquet enviat durant el darrer període de temps. Si el rati del paquet i és com a molt r (és a dir, $r_i \leq r$) aleshores tots els paquets enviats fins aleshores tenen un rati també de, com a molt, r; de manera que el total enviat al període $[0,t]$ és com a molt rt, la qualcosa contradiu la suposició feta que havíem enviat massa bits. Aleshores, és segur que tenim que $r_i > r$. Com que resulta que els paquets estan ordenats creixentment per rati i, a qualsevol instant de temps posterior a t també enviarem com a mínim r bits, doncs. En fer això, el rati total després d'enviar tots els paquets tampoc cumplirà el requeriment de mantenir un rati promig de com a molt r.

Hi ha altres formes de demostrar que la planificació resultant de l'ordenació per ratis és vàlida. Per exemple, fent servir un argument *"greedy stays ahead"* o un argument *"gredy exchange"* (consulteu els documents adjunts).

2. *Exercici 29 (a.2)*

Solució

a.2) *Un arbre d'expansió mínim és també un bottleneck spanning tree.*

Aquest sentit de la implicació és cert. Ho podem demostrar de diverses formes, per contradicció:

1. Sigui $v(T)$ el valor d'un arbre d'expansió, definit com el pes de l'aresta de pes màxim de T, és a dir,
$$v(T) = \max\{w(e) \mid e \in E(T)\}.$$
Suposem que existeix un arbre T que és minimum ST però no és bottleneck ST.
   - $\Rightarrow$ podem substituir una de les seves arestes $e \in E(T)$ amb una altra $e' \in E(T)$ de manera que encara formi un ST (anomenem $T' = (V, E')'$ a aquest nou arbre) i que cumpleixi que $v(T') < v(T)$.
   - $\Rightarrow$ aleshores el pes total de T' és :
$$w(T') = \sum_{e' \in E'} w(e') = \sum_{e \in E} w(e) - v(T) + v(T') < \sum_{e \in E} w(e)$$
   - $\Rightarrow$ per tant T no era minimum ST

2. Suposem que T és un minimum ST de G, i que T' és un ST amb una aresta més lleugera. Aleshores, T conté una aresta e que té més pes que tota aresta de T'. Aleshores, si afegim e a T', formarà un cicle C en què serà l'aresta de més pes (donat que totes les altres arestes de C pertanyen a T'). Per la propietat del tall (*Cut Property*), aleshores e no pertany a cap minimum ST, contradient el fet que és a T i T és un minimum ST.

3. *Exercici 32 (c)*

Solució amb Independent Sets

c) Una alternativa a la solució vista a classe per a dissenyar un algorisme que trobi el recobriment de vèrtexs minimal d'un graf G, és fer servir la relació que existeix amb el problema del conjunt independent de vèrtexos. La relació és la següent:*Un recobriment de vèrtex minimal d'un graf G es correspon amb el complement del conjunt de vèrtexos independent màxim; en conseqüència, el nombre de recobriments de vèrtexos mínims i el nombre de conjunts de vèrtexos independents màxim a G són idèntics.*

funció MAXIMAL_INDEPENDENT_SET ($G = (V, E)$)
    $I \leftarrow \emptyset$
    mentre $V \neq \emptyset$ **fer**
        escollir un node $v \in V$ qualsevol
        $I \leftarrow I \cup \{v\}$
        $V \leftarrow V - \{v\}$
        $V \leftarrow V \setminus \mathcal{N}(v)$ ▷ $\mathcal{N}(v)$ són els veïns del node v al graf induït $G(V)$
    fi mentre
    retornar I
fi funció

funció MINIMAL_VERTEX_COVER ($G = (V, E)$)
    $I \leftarrow$ MAXIMA_INDEPENDENT_SET(G)
    retornar $V \setminus I$
fi funció