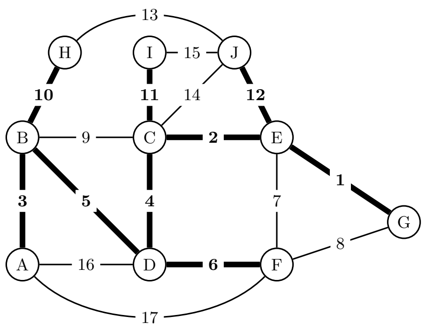
Final 11/1/2005

Observacions:

- Les solucions proposades no tenen perquè ser úniques.
- Els textos entre claudàtors proporcionen aclariments o informació addicional que no es demanava a l'examen.

1. [Com que el graf no té pesos repetits a les arestes, hi ha un únic arbre d'expansió mínim.]



2. a) Un contraexemple possible amb dues tasques es dóna quan $t[1] = 2$ i $\ell[1] = 5$ i quan $t[2] = 3$ i $\ell[2] = 1$. En aquest cas, l'algorisme proposat col·loca la tasca 1 abans de la 2 i, per tant, produeix una seqüència infactible. En canvi, col·locant la tasca 2 abans de la 1 s'obté una seqüència factible.

   b) L'algorisme voraç ordena les tasques segons el seu temps d'acabament $t[i] + \ell[i]$.

   Si la seqüència de tasques no és factible, l'algorisme retorna una seqüència infactible, tal com cal. Altrament, sigui t la solució trobada per l'algorisme proposat i sigui s qualsevol altra solució factible. Llavors s i t tenen, almenys, una inversió. És fàcil veure que quan s'intercanvien dues tasques consecutives que contradiuen l'ordre proposat inversió, s'obté una altra solució factible. Així, desfent l'una rera l'altra aquestes inversions consecutives de s, s'obté t. Per tant, t és una solució factible, tal com cal.

   El cost de l'algorisme és dominat pel cost de l'ordenació i, per tant, és $\Theta(n \log n)$ en el cas pitjor.

3. El darrer espai ha d'iniciar el procés de tornada enrera amb una solució parcial buida, deixant el resultat al camp `iso`. Com que el primer vèrtex és el zero, cal omplir aquest espai amb `iso = backtracking(0);`.

   El primer espai es correspon a la condició de final amb èxit de la tornada enrera. Aquesta es produeix quan la solució parcial és una solució total, per tant cal omplir aquest espai amb `u == n`.

   En el segon espai, es prepara l'iteració següent, la qual generarà el proper fill a provar. Aquí cal restablirel valor de cert a `usat[w]`, perquè `w` ja no és usat. Cal doncs omplir el espai amb `usat[w] = false;`.

   El tercer espai és a la funció que comprova que l'addició del vèrtex `u` a `f` formi un isomorfisme parcial, sabent que ja el formaven tots els demés vèrtexs usats. Per tant, aquí cal comprovar que per a tot `v`, si hi ha una aresta de `v` a `u` en `G1`, aquesta també hi sigui en `G2`, i del revés. Cal doncs omplir el espai amb `G1[u][v]==G2[f[u]][f[v]]`.

4. a) Aquí va la reducció del problema d'en Jonny al problema del geni: Agafem $m := n+1$ i per a cada $1 \leqslant i \leqslant n$, agafem $b_i := p_i$. A més, agafem $b_{n+1} := -T$.

   $\Rightarrow$ Si alguns dels p_i sumen exactament T, llavors aquells b_i amb $-T$ sumen zero.

   $\Leftarrow$ Si algun conjunt no buit de b_i suma zero, és perquè entre ells hi ha el $-T$ (altrament no sumarien zero perquè tots els b_i amb $i \neq n+1$ són estrictament positius). Llavors els p_i corresponents sumen T.

   Aquesta reducció es pot portar a terme en temps lineal (i, per tant, en temps polinòmic).

   [Com que la Steffy ens ha dit que el problema d'en Jonny és **NP**-complet, ara sabem que el problema del geni és **NP**-difícil. Això justifica que el venedor d'Istambul digués que la llàntia resol eficientment qualsevol problema **NP** (el venedor, però, es va callar que calia trobar les reduccions adients per utilitzar-la!).]

   b) La Steffy ens ha fet saber que el problema d'en Jonny és **NP**-complet, i a l'apartat anterior hem reduït el problema d'en Jonny al del geni. Per tant, per acabar de demostrar que el problema del geni és **NP**-complet, només cal demostrar que aquest pertany a **NP**.

   Un testimoni possible per al problema del geni és una taula de m bits: El bit i-èsim indica si cal agafar o no el número b_i. El testimoni té llargada polinòmica respecte de l'entrada (aquesta ha d'ocupar almenys m bits) i es pot comprovar en temps lineal si els números seleccionats sumen o no zero.

5. Trobar el nombre mínim de talls que calen per aconseguir un retall en forma de palíndroms de s correspon a calcular $t[1,n]$. Per fer-ho, utilitzem la recurrència següent:

$$t[i,j] = \begin{cases} 0 & \text{si } s[i,j] \text{ és un palíndrom,} \\ \min\{1 + t[i,k] + t[k+1,j] \ : \ i \leqslant k < j\} & \text{altrament.} \end{cases}$$

   En efecte, si $s[i,j]$ no és un palíndrom, cal trobar un punt k amb $i \leqslant k < j$ per fer-li un tall, tallar de forma òptima la part esquerra $s[i,k]$, i tallar de forma òptima la part dreta $s[k+1,j]$. Logicament, cal triar el valor de k que minimitzi aquesta suma de talls.

   La recurrència anterior es pot implementar directament de dalt cap a baix de forma recursiva, tot utilitzant memorització per tal de no repetir càlculs.

   Com que, gràcies a la memorització, cada $t[i,j]$ es calcula com a molt un cop, el cost d'aquesta algorisme és:

$$T(n) = \sum_{i=1..n} \sum_{j=i..n} \left(\mathrm{O}(j-i+1) + \sum_{k=i..j} \Theta(1) \right) = \Theta(n^3).$$

   (Per a cada $t[i,j]$ amb $1 \leqslant i \leqslant j \leqslant n$ calen $\mathrm{O}(j-i+1)$ passos per calcular si $s[i,j]$ és un palíndrom i $\sum_{k=i..j} \Theta(1)$ passos per calcular el mínim de $j-i$ valors.)

   També es pot implementar la programació dinàmica de baix cap a dalt omplint la taula iterativament. Com que l'estructura d'aquesta recurrència és idèntica a la del problema dels productes encadenats de matrius o a la del problema del tall de les barres d'acer, un algorisme semblant farà el fet. El seu cost també serà cúbic.

   [Amb un algorisme de programació dinàmica més astut es pot aconseguir una solució quadràtica.]