

Problemes 3

- 3.1. **(Reconstruint text)** Us donen una cadena de n caràcters $s[1 \dots n]$, que pot ser un text on totes les separacions entre mots ha desaparegut, per exemple *aquestaesunafrasequepodiaserunexemple*. Volem reconstruir el text original amb ajut d'un diccionari $D(\cdot)$, tal que per a tot mot possible w

$$D(w) = \begin{cases} \text{cert} & \text{si } w \text{ és un mot valid} \\ \text{fals} & \text{altrament} \end{cases}$$

- (a) Doneu un algorisme de programació dinàmica que determine si la cadena $s[\cdot]$ es pot reconstruir com a una seqüència de mots vàlids. Si assumim que les crides a D es poden fer en temps $\Theta(1)$. Quina és la complexitat del vostre algorisme?
- (b) Si la cadena és vàlida, fes que el teu algorisme escrigui la frase correcta, amb separacions entre mots.
- 3.2. **(Joc a una graella)** Considerem una graella de 4 files per n columnes, i un conjunt de $2n$ fitxes. Una fitxa es pot col·locar exactament a una casella de la graella. Definim un *patró legal columna* a una columna de la graella com la situació resultant de col·locar fitxes a la columna de manera que no dues fitxes estiguin en caselles adjacents. De la mateixa manera podem definir un *patró legal fila*. Una *configuració legal* és la situació resultant de situar fitxes a la graella, de manera que totes les columnes i files tinguin patrons legals. A cada casella de la graella hi ha escrit un enter. El *valor* de la configuració és la suma dels enters de les caselles ocupades.

Dos patrons a columnes adjacents són *compatibles* si formen una configuració legal a la matriu formada per les dos columnes.

- (a) Determineu el nombre total de patrons legals que pot haver en una columna.
- (b) Dissenyeu un algorisme $O(n)$ per calcular una configuració de valor màxim.
Ajut: Considereu subproblemes amb les primeres k columnes ($k \leq n$) i un patró prefixat a la columna k .
- 3.3. **(Monòlit)** El Rector vol construir el monument a l'alumne sacrificat. Decideix construir al Campus Nord, el monòlit més alt possible de totxo vist. Aconsegueix n tipus diferents de totxos, i de cada tipus una quantitat suficientment gran. Cada totxo de tipus i es pot considerar com un ortoedre amb dimensions $\langle l_i, w_i, h_i \rangle$. Un totxo es pot col·locar en qualsevol de les tres posicions que mantenen les arestes paral·leles als tres eixos fixos. Per a construir el monòlit un totxo es pot col·locar a sobre d'un altre, sols si cada una de les dues arestes de la base del totxo de sobre té longitud estrictament menor que l'aresta que li és paral·lela del totxo de sota. A la base es col·loca un sol totxo. Dissenyeu un algorisme eficient per a determinar el monòlit més alt que el Rector pot construir. Quina és la seva complexitat?.

3.4. **(Server)** Imagina que ets el cap dels serveis informàtics de la FIB, on milers de persones accedeixen cada dia al servidor central. Suposem que tens una estimació (x_1, x_2, \dots, x_n) del nombre d'usuaris que accediran al servidor en els propers n dies. El software que controla el servidor no està ben dissenyat i el nombre d'usuaris per dia que pot gestionar decremента cada dia, a partir del darrer dia en què es va fer *reboot*. Sigui s_i el nombre d'usuaris que el servidor pot gestionar l' i -èsim dia després de la darrera aturada, per tant $s_1 > s_2 > s_3 > \dots > s_n$. Assumim que el dia que es fa el reboot, no es pot donar servei a cap usuari. Donada una seqüència de carrega (x_1, \dots, x_n) i de limitacions (s_1, \dots, s_n) , dissenyeu un algorisme per a la planificació que especifique els dies òptims que s'han de fer els reboots de manera que es maximitze el nombre total de clients als quals el servidor dóna servei. Per exemple, si $n = 5$ i $s_1 = 16, s_2 = 8, s_3 = 4, s_4 = 2, s_5 = 1$. Quan $x_1 = 17, x_2 = 9, x_3 = 5, x_4 = 3, x_5 = 2$, la solució òptima és no rebotar i donar servei a 31 clients. Quan $x_1 = 17, x_2 = 9, x_3 = 17, x_4 = 3, x_5 = 17$ la solució òptima és rebotar el segon i quart dies, donant servei a un màxim de 48 clients.

3.5. **(Partició lineal)** El problema de la *partició lineal* es el següent: Donada una seqüència de n valors positius, (s_1, \dots, s_n) volem obtenir una seqüència de $r + 1$ valors, $i_1, \dots, i_{r+1} \in \{1, \dots, n+1\}$ tal que $i_1 = 1, i_{r+1} = n+1$ i $i_j < i_{j+1}$, per $1 \leq j \leq r$. Aquesta successió divideix la seqüència inicial en r rangs. Per cada rang $j, 1 \leq j \leq r$, definim $S_j = \sum_{i_j \leq k < i_{j+1}} s_k$. A cada seqüència i_1, \dots, i_{r+1} se li assigna el cost $S(i_1, \dots, i_{r+1}) = \max_{1 \leq j < r} S_j$. Volem obtenir seqüència que proporcioni r rangs amb cost mínim.

Per exemple, si els valors són:

$$100, 200, 300, 400, 500, 600, 700, 800, 900$$

si $r = 3$, una solució és 1, 4, 7, 10 que té cost 2400, i una solució òptima és 1, 6, 8, 10 amb cost 1700.

Però si els valors són:

$$1000, 250, 120, 40, 50, 160, 700, 180, 90$$

si $r = 3$, una solució és 1, 4, 7, 10 que té cost 1370, i una solució òptima és 1, 2, 7, 10 amb cost 1000.

Dissenyeu un algorisme basat en programació dinàmica per resoldre el problema. Analitza la complexitat temporal i espacial de l'algorisme proposat.

3.6. **(Partint en palíndroms)** Un palíndrom és un mot $w_1 w_2 \dots w_k$ tal que $w_k w_{k-1} \dots w_1$ és el mateix mot. Per exemple *senentesisnensisetnenes*. Donada una cadena $A = a_1 a_2 a_3 \dots a_n$ direm que A té una partició en forma de palíndroms si cada subcadena de la partició és un palíndrom. Per exemple, si $A = ababbabbababa$ tenen particions $aba|b|bbabb|a|b|aba$ i $aba|bbb|abba|b|a|b|a$ en forma de palíndroms. Dissenyeu un algorisme amb complexitat $O(n^3)$ tal que donada una cadena A de talla n , produeixi la partició en forma de palíndroms amb el **mínim nombre de talls**. Podeu dissenyar un algorisme amb complexitat $O(n^2)$? Noteu que si A es un palíndrom no hi haurà cap partició.

3.7. **(Tallant ADN: freqüència)**. En termes computacionals, podem pensar que una cadena de ADN es un mot sobre l'alfabet $\{A, C, G, T\}$. El cost d'una cadena de ADN el definim com

$\prod_{X \in \{A,C,G,T\}} (\text{freq}[X] + 1)$ on $\text{freq}[X]$ es el nombre d'ocurrències del símbol X . Per exemple, si la cadena es $ACGAC$ el seu cost es $3 \times 3 \times 2 \times 1 = 18$

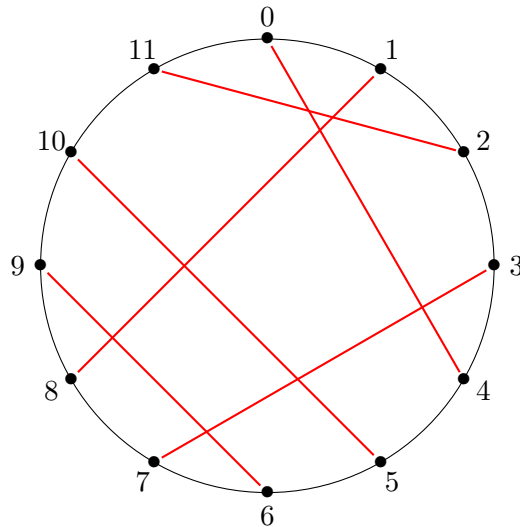
Tenim una cadena de ADN de llargada n i volem dividir-la en k trossos (subcadenaes no buides), de manera que es minimitze el cost màxim d'entre les k portions. Per exemple per $ACGAC$ i $k = 2$ tenim 4 formes de trencar la cadena en dos i una de les que ens dona el millor cost es AC i GAC (amb cost màxim 8). Proporcioneu un algorisme ho més eficient possible per a resoldre aquest problema, donats S i k .

- 3.8. **(Biblioteca)** Considereu el problema d'emmagatzemar n llibres als prestatges de la biblioteca. L'ordre dels llibres és fixat pel sistema de catalogació i, per tant, no es pot canviar. Els llibres han d'aparèixer a les prestatgeries en l'ordre designat. Les prestatgeries d'aquesta biblioteca tenen amplada L i són regulables en alçada. Un llibre b_i , on $1 \leq i \leq n$, té gruix t_i i alçada h_i . Una vegada es decideix quins llibres es fiquen a un prestatge s'ajusta la seva alçada a la del llibre més alt que col·loquem al prestatge. Doneu un algorisme que ens permeti col·locar els n llibres a les prestatgeries de la biblioteca de manera que es minimitzi la suma de les alçades dels prestatges utilitzats.
- 3.9. **(Festa UPC)** La gerenta de la UPC vol donar una festa als PAS de la universitat. Aquest personal té una estructura jeràrquica, en forma d'arbre on la gerenta és l'arrel. L'oficina de personal ha assignat a cada PAS un nombre real que representa el seu grau de *simpatia*. En vista que la festa sigui distesa, la gerenta no vol que cap superior immediat d'una persona convidada, també sigui convidada. Descriviu un algorisme per confeccionar la llista de convidats de manera que es maximitze la suma dels graus de simpatia. Quina és la complexitat del vostre algorisme?. Què hauríeu de fer per assegurar que la gerenta està invitada a la seva pròpia festa?
- 3.10. **(Despertant sensors)** Els professors Maria Serna i Jordi Petit volen resoldre el següent problema: Tenen una xarxa de sensors organitzada en forma d'arbre, on els sensors ocupen els nusos. La major part del temps els sensors estan en un estat letàrgic (de mínim consum d'energia) fins que un sensor que actua com autoritat central, situat a l'arrel de l'arbre, decideix que alguna cosa importat succeeix i *desperta* la resta dels sensors. Aquest procés de despertar els sensors requereix un cert temps ja que en una unitat de temps un sensor pot despertar únicament un dels seus fills (a l'arbre). Òbviament, el temps total per a despertar tots els sensors depèn de l'ordre en què cada sensor desperti els seus fills. Dissenyau un algorisme eficient que determine una ordenació dels fills de cada nus de l'arbre proporcionant el temps mínim per despertar a tots els sensors.
- 3.11. **(El joc dels pots d'or)** Dos jugadors estan jugant al joc dels pots d'or. En aquest joc hi ha n pots d'or disposats en línia, $\{p_1, \dots, p_n\}$, i cada pot p_i conté una determinada quantitat $c_i > 0$ de monedes d'or. Els jugadors juguen de manera alternada i, a cada torn, el jugador que té el torn pot escollir un pot d'un dels extrems de la línia que hi hagi en aquell moment. Per exemple, el jugador que comença el joc pot escollir quedar-se amb el pot p_1 o amb el pot p_n . Si decideix quedar-se amb p_1 , aleshores l'altre jugador podrà escollir entre p_2 i p_n al següent torn; en canvi, si el primer jugador escull p_n , l'altre jugador haurà de triar entre p_1 i p_{n-1} .
- El guanyador és el jugador que, una vegada no quedin pots per agafar, tingui una quantitat més gran de monedes.

Dissenyeu un algorisme eficient per calcular quin és el nombre màxim de monedes que podria obtenir en aquest joc el jugador que comença primer, suposant que els dos jugadors volen guanyar i, per tant, ambdós juguen de manera intel·ligent per tal d'obtenir el màxim nombre possible de monedes. Doneu també la seqüència de pots que hauria d'anar escollint a cada torn per aconseguir aquesta quantitat.

- 3.12. **(Selecció de cordes)** Donat un conjunt de n cordes en el cercle unitat diem que un subconjunt de cordes es *viàble* si no hi han dues cordes que es tallen. Volem trobar un subconjunt viàble amb mida màxima.

Per resoldre el problema assumim que mai dues cordes tenen un extrem en comú. Per això podem enumerar els extrems de les n cordes de 0 a $2n - 1$ seguint el sentit de les agulles del rellotge. Aleshores, l'entrada del problema consisteix en una seqüència de n parelles dels nombres $0, \dots, 2n - 1$ on cada i , $0 \leq i \leq 2n - 1$, apareix exactament en una parella. La parella (i, j) representa la corda amb extrems i i j . A la figura següent teniu un exemple de instància amb 6 cordes:



l'entrada corresponent és $(0, 4), (1, 8), (2, 11), (3, 7), (5, 10), (9, 6)$.

Per $0 \leq i < j \leq 2n - 1$, definim $T(i, j)$ com la mida del subconjunt viàble més gran que es pot formar amb el conjunt de les cordes (a, b) tals que $i \leq a, b \leq j$.

- Per $0 \leq i < j \leq 2n - 1$, proporcioneu una recurrència que permeti calcular $T(i, j)$.
 - Proporcioneu un algorisme que, donat un conjunt de cordes en el cercle unitat, obtingui un conjunt viàble amb mida màxima en temps polinòmic.
- 3.13. **(Observació astronòmica)** Un grup d'amics del departament d'astronomia vol observar el cel aquesta nit (que és un dia amb cel ras). Suposem el següent:

- Hi ha n esdeveniments que ocorren en una seqüència de n minuts, on l'esdeveniment j ocorre al minut j . Si no observem el esdeveniment j al minut j , no l'observem mai.

- Suposem que utilitzem un sistema 1-dimensional per modelitzar el cel (que correspon el grau de l'angle del telescopi); és a dir, l'esdeveniment j té coordenada d_j per $d_j \in \mathbb{Z}$. Al minut 0, la posició del telescopi és 0.
- L'últim esdeveniment n és més important que els altres; per tant estem obligats a observar l'esdeveniment n .

El telescopi del departament és molt gran i només es pot moure un grau cada minut. Per tant, és possible que no es pugui observar tots els esdeveniments. Un conjunt d'esdeveniments S es diu *observable*, si per a cada $j \in S$, es pot observar l'esdeveniment j al minut j i entre dos elements consecutius j, k de S el telescopi té el temps necessari per bellugar-se (amb velocitat com a màxim 1 per minut) de d_j a d_k . Donats n esdeveniments, i una seqüència $\{d_j\}_{j=1}^n$ que correspon a les coordenades dels esdeveniments, volem trobar el conjunt més gran de tots els esdeveniments observables S amb la condició que $n \in S$.

Exemple: Si tenim $n = 9$ i $d_1 = 1, d_2 = -4, d_3 = -1, d_4 = 4, d_5 = 5, d_6 = -4, d_7 = 6, d_8 = 7, d_9 = -2$, llavors la solució òptima és $S = \{1, 3, 6, 9\}$ (Notem que sense la restricció de que l'esdeveniment 9 ha de ser a S la solució seria $\{1, 4, 5, 7, 8\}$).

- (a) Demostreu mitjançant un contraexemple que l'algorisme següent no funciona correctament:

Marca tots els esdeveniments j amb $|d_n - d_j| > n - j$ com a il·legal (no podem observar aquests esdeveniments, perquè hem d'observar l'esdeveniment n) i tots els altres com a legal

inicialització $p(0) := 0, S := \emptyset$

mentre encara no s'om a la fi de la seqüència **fer**

busca el primer esdeveniment legal j a partir del moment al qual es pot arribar bellugant el telescopi amb velocitat $\leq 1/\text{minut}$

$S := S \cup \{j\}$

$p(j) := d_j$

fimentre

tornar S .

- (b) Donada una seqüència d'enters d_1, \dots, d_n , doneu un algorisme correcte i polinòmic en n per calcular el conjunt observable S més gran (amb la condició que $n \in S$).

3.14. **(Grau de simpatia)** Els estudiants de la FIB volen dissenyar una xarxa social (i.e. un graf dirigit $G = (V, E)$) per determinar el grau de simpatia entre tota la comunitat universitària a la UE. El graf es dissenya a partir de relacions personals; si a coneix b , $a, b \in V$ i $(a, b) \in E$. A més, a cada aresta (a, b) se li assigna un pes entre 0 i 10 que indica la simpatia de b en opinió de a (0 molta antipatia, 10 molta simpatia).

Per tal que un estudiant a pugui tenir una idea del grau de simpatia d'un estudiant d que no coneix, simplement ha de trobar el valor del camí amb pes màxim $\mu(a, d)$ i el valor del camí amb pes mínim $\delta(a, b)$. Però hi ha un problema no sabem com trobar el valor del camí amb pes màxim. Per sort hi ha un estudiant de l'assignatura d'Algorísmia de la FIB té una l'idea: negar el valor dels pesos (i.e. si una aresta té pes 7, assignar-li el pes -7) i aplicar Bellman-Ford per a trobar el camí mínim, que serà el màxim sense negar. Penseu que l'algorisme del vostre col·lega és una bona solució?

- 3.15. **(Distàncies en grafs unicíclics)** Un graf *unicíclic* és un graf no dirigit que conté només un cicle. Sigui donat un graf ponderat unicíclic $G = (V, E, w)$, on $w : E \rightarrow \mathbb{R}$, i un vèrtex $u \in V$. Proporcioneu un algorisme (el més eficient que pugueu) per a trobar les distàncies d' u a tots els altres vèrtexs de G en cas que sigui possible definir-les.
- 3.16. **(Arbitratge de divises)** L'arbitratge de divises és una situació en la qual un operador de monedes intel·ligent pot executar una seqüència de canvis de moneda per tal de obtenir una quantitat potencialment il·limitada de diners. Per exemple, suposem que els dòlars nord-americans s'estan comprant en el mercat de divises per 50 rupies i una altra moneda al mercat de divises ven dòlars nord-americans per 40 rupies. En aquesta situació, un operador podria intercanviar 1 milió de dòlars per l'equivalent a 50 milions de rupies i després intercanvien les rupies per $50 \text{ milions} / 40 = 1,25$ milions de dòlars. Les situacions d'arbitratge de divises que ens plantegem són més complexes i impliquen diversos passos de conversió entre moltes monedes. Per exemple, en el següent quadre de conversió teniu un arbitratge de tres passos.

Problem. Given table of exchange rates, is there an arbitrage opportunity?

	USD	EUR	GBP	CHF	CAD
USD	1	0.741	0.657	1.061	1.011
EUR	1.350	1	0.888	1.433	1.366
GBP	1.521	1.126	1	1.614	1.538
CHF	0.943	0.698	0.620	1	0.953
CAD	0.995	0.732	0.650	1.049	1

Ex. \$1,000 \Rightarrow 741 Euros \Rightarrow 1,012.206 Canadian dollars \Rightarrow \$1,007.14497.

$$1000 \times 0.741 \times 1.366 \times 0.995 = 1007.14497$$

Dissenyu un algorisme que, donada una taula de conversió, trobi un arbitratge que ens permeti incrementar, si és possible, la nostra quantitat inicial de diners.

- 3.17. **(MST versus arbres de distàncies)** En aquest problema, estudiem la relació entre *arbres d'expansió mínims* (MST) i *arbres de camins mínims* en un graf no dirigit G . Recordeu que donat un $G = (V, E)$ amb pesos $w : E \rightarrow \mathbb{R}$ i un punt $s \in V$ l'arbre de camins mínims arrelat a s és un subgraf $T' = (V', E')$ de G tal que:
- T' és un arbre, i per tant $|E'| = |V'| - 1$,
 - hi ha un camí de s fins a qualsevol vertex a V' ,
 - per a qualsevol $u \in V'$, la distància de s a u a T' és la mateixa que la distància de s a u a G .

Recordeu que, igual que succeix amb el MST, donat un $s \in V'$, G pot tenir més d'un arbre de camins mínims arrelat a s .

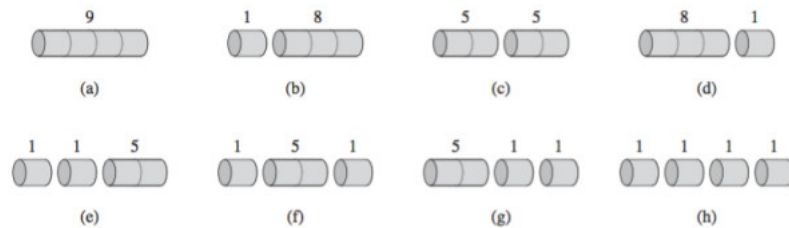
- (a) Demostreu si és cert o no que donat qualsevol graf connex i no dirigit G , amb $w : E \rightarrow \mathbb{R}^+$, sempre hi ha un arbre de camins mínims T' tal que T' també és un arbre d'expansió mínima a G .
- (b) Demostreu si pot haver-hi un graf no dirigit G amb $w : E \rightarrow \mathbb{R}^+$ i connex, tal que G tingui un arbre de camins mínims T' i un MST T que no comparteixen cap aresta.

3.18. **(Camins més llargs en DAGs)** Donat com a entrada un graf dirigit $G = (V, E, w)$ on $w : E \rightarrow \mathbb{Z}^+$, i un vèrtex inicial $s \in V$, volem trobar el camí **simple** de màxima distància entre s i la resta dels vèrtexs a G . Per a grafs generals, no es coneix una solució polinòmica per a aquest problema. Doneu un algorisme polinòmic per al cas particular que G sigui un DAG (graf dirigit sense cicles). Podeu obtenir un algorisme amb cost $O(n + m)$? (Recordeu que un camí simple és aquell que no repeteix cap vèrtex.)

3.19. **(Tallant una barra d'acer)** Els recursos naturals són cada vegada més escassos i hem de reduir-ne i optimitzar-ne l'ús. A la indústria metal·lúrgica això és especialment important perquè treballen principalment amb minerals com a matèria primera. Per aquesta raó hi ha molts processos d'optimització que es desenvolupen i s'apliquen en aquest àmbit. A nivell econòmic, sovint això implica haver de treure el màxim profit econòmic de la quantitat de recursos disponible en un determinat moment.

Tenim una gran barra d'acer de longitud n , i volem tallar-la en trossos per a destinar-los a diferents usos. Cada tros de barra d'acer, en funció de la seva llargària, té un cost al mercat. Un tros de llargària i , amb $i \in \mathbb{Z}^+$ i $1 \leq i \leq n$, val p_i euros. Observeu que les llargàries dels trossos són unitats enteres.

La figura següent en mostra un exemple amb les 8 possibles formes de tallar una barra d'acer de llargària 4. Sobre cadascun dels trossos s'indica el preu que se n'obté ($\{p_1 = 1, p_2 = 5, p_3 = 8, p_4 = 9\}$). La solució òptima és l'opció (c) –tallar la barra en dos trossos de longitud 2– que dona un guany de 10€.



Es demana:

- a) De quantes formes diferents es pot tallar una barra de llargària n ? Raoneu la resposta.
- b) Dissenyeu un algorisme, el més eficient que pugueu, per a decidir com tallar una barra d'acer de longitud n en trossos, de manera que es maximitzi el guany total que se n'obté. El vostre algorisme ha de dir quants talls s'han de fer en total, i a on.

Assumirem que fer un tall a la barra no indueix cap cost afegit. Observeu que no es demana cap requisit sobre el nombre de talls a fer; podeu fer qualsevol nombre de talls entre 0 i $n - 1$.

- 3.20. (**Impressió**) Considerem el problema d'imprimir de manera polida una frase amb una impressora. El text d'entrada és una seqüència de n mots amb longitud l_1, l_2, \dots, l_n , on cada longitud ve donada en caràcters. Cada línia pot contenir com a màxim M caràcters. Realitzem la impressió de manera que si una línia conté els mots de i fins a j , on $i \leq j$, i deixem exactament un espai entre mots, aleshores el nombre de caràcters en blanc al final de cada línia és $M - j + i - \sum_{k=i}^j l_k$, que ha de ser no-negatiu. Volem minimitzar la suma, sobre totes les línies excepte la darrera, dels quadrats d'aquestes magnituds. Dissenyeu un algorisme per imprimir de la manera indicada, un paràgraf amb n mots. Analitzeu les complexitats espacials i temporals del vostre algorisme.
- 3.21. (**Mediana de les distàncies**) Tenim un graph dirigit G , amb n vèrtexs i m arestes, i amb pesos $w : E \rightarrow \mathbb{Z}$, on G no té cicles negatius. De totes les distàncies més curtes entre qualsevol dos vèrtexs, volem trobar quins són els vèrtexs $u, v \in V$ que la seva distància més curta és la mediana de totes les distàncies més curtes. Digueu i expliqueu com utilitzaríeu els algorismes explicats a classe per a solucionar el problema y doneu la complexitat en funció de n , quan
- (a) \vec{G} no té pesos negatius i el graf és dens.
 - (b) \vec{G} pot tenir pesos negatius i el graf és dens.
 - (c) \vec{G} pot tenir pesos negatius i el graf és espars.

- 3.22. (**Copistes**) Abans de la invenció de la impremta, era molt difícil fer una còpia d'un llibre. Tots els continguts havien de ser redactats a mà pels anomenats copistes. A un copista se li donava un llibre i, després de diversos mesos de treball, tornava una còpia del mateix. El temps que trigava era, molt probablement, proporcional al nombre de pàgines del llibre. La feina devia ser prou avorrida i per això podríem assumir que tots els copistes d'un monestir trigaven el mateix temps a copiar una pàgina.

El monestir de Pedralbes va decidir fer una còpia dels llibres de la seva biblioteca i, donat que no tenen copistes propis, han d'enviar els llibres a un altre monestir que sí que en tingui.

Podeu assumir que hi ha un total de n llibres per copiar i que el llibre i -èsim té p_i pàgines. A més, els llibres tenen un ordre predeterminat. Una vegada triat el monestir on es faran les còpies, s'hauran de repartir els llibres entre el seus m copistes. Cada llibre només se li pot assignar a un copista, i a cada copista només se li pot assignar una seqüència contigua de llibres (d'acord amb l'ordre inicial dels llibres). Amb aquesta forma d'assignar llibres a copistes es minimitza el temps de buidar i tornar a omplir la biblioteca. El temps necessari per fer la còpia total de la biblioteca queda determinat pel temps que necessita el darrer copista que finalitza la còpia dels llibres que se li han assignat.

El que no tenen molt clar els encarregats de la biblioteca és com fer l'assignació de llibres a copistes per garantir que el temps de còpia total de la biblioteca sigui el més curt possible. Ajudeu a aquests monjos i dissenyeu un algorisme eficient per a trobar l'assignació òptima dels n llibres als m copistes del monestir. Podeu assumir que coneixeu el temps t_p que necessita cadascun dels copistes per a copiar una pàgina.

3.23. **(Kakia)** A la universitat de Kakia, l'equip de govern està molt preocupat per l'efecte que els exàmens produeixen sobre l'estat anímic dels estudiants, per tant han decidit convertir l'edifici que alberga el Centre de Matemàtica utilitzant Neurons (CMN) en un centre d'esplai per als estudiants. EL Personal docent i investigador (PDI) allotjat a l'edifici del CMN, ha decidit defensar-se del desallotjament institucional i tancar-se a l'edifici. Per aconseguir el desallotjament, l'equip de govern vol construir un eixam (swarm) de microrobots que ataquen al PDI a l'edifici fins que marxen. Els microrobots ataquen de la manera següent:

- (a) Durant n segons, un eixam de robots arriba de manera que a l' i -èsim segon, arriben x_i robots. EL PDI del CMN ha col·locat sensors envoltant l'edifici, de manera que poden preveure la seqüència x_1, x_2, \dots, x_n abans que els primers robots arribin.
- (b) El personal del CMN ha desenvolupat un polsador electromagnètic que pot destruir alguns dels robots quan arriben, el nombre de robots que destrueixen depèn del nivell de càrrega que tingui el polsador. Formalment, existeix una funció f tal que si han transcorregut j segons des de la darrera vegada que es va utilitzar el polsador, es destrueixen $f(j)$ robots. Per tant, si utilitzem el polsador al k -èsim segon, quan feia j segons que s'havia utilitzat, el nombre de robots que destruiran serà $\min(x_k, f(j))$, i s'esgotarà la càrrega del polsador.
- (c) Al començament el polsador està totalment carregat, per tant si el polsador s'utilitza per primer cop al j -èsim segon, pot destruir $f(j)$ robots.

Donada la informació x_1, x_2, \dots, x_n y donada la funció f , volem escollir els moments en què haurem d'utilitzar el polsador per a destruir el màxim nombre possible de robots. Per exemple, si $n = 4$, $x_1 = 1, x_2 = 10, x_3 = 10, x_4 = 1$ i $f(1) = 1, f(2) = 2, f(3) = 4, f(4) = 8$ aleshores la millor solució és activar el polsador al 3er i 4rt segon, al 3er segon destrueix 4 robots i al 4rt segon destrueix 1 robot (per la càrrega). En total es poden destruir 5 robots. Dissenyau un algorisme eficient tal que donats x_1, x_2, \dots, x_n i f , retorni la seqüència de pulsacions que maximitzi el nombre de robots destruïts.

3.24. **(Equipant un avatar)**

En un joc en línia un jugador ha d'equipar un avatar amb peces de certes categories: per exemple, un vestit, un casc, una armilla o cuirassa, una arma blanca curta, un arc de fletxes, una arma de foc curt i una arma de foc llarg. Per a cadascuna de les categories, el jugador ha de seleccionar un ítem de la corresponent categoria. Cada ítem té un preu i proporciona una puntuació.

Hi ha k categories, C_i on, per $1 \leq i \leq k$, $C_i = \{e_{i,0}, e_{i,1}, e_{i,2}, \dots, e_{i,n_i}\}$, amb $n_i \geq 0$ per a tota i , i associat a cada possible ítem $e_{i,j}$ un preu $v_{i,j} > 0$ (excepte l'ítem $e_{i,0}$, que és de franc, és a dir, $v_{i,0} = 0$) i una puntuació $p_{i,j} \geq 0$. Per exemple, C_1 podria ser la categoria dels arcs de fletxes i podria haver-hi $n_1 + 1 = 4$ arcs de fletxes diferents, sent l'arc $e_{1,0}$ l'arc més bàsic, amb cost $v_{1,0} = 0$, i després hi hauria tres arcs més, amb diferents prestacions i costos. El nostre jugador haurà d'equipar el seu avatar amb un dels 4 tipus d'arc, de la mateixa manera que l'haurà d'equipar de vestit, proteccions corporals, etc. Fixeu-vos que pot haver-hi ítems amb puntuació 0 a més dels ítems bàsics $e_{i,0}$ de cada categoria.

Donades totes les dades del problema i la quantitat de diners $0 < Q \leq 1000$ disponibles, volem trobar una solució en la qual s'esculli exactament un ítem de cada categoria de manera que

la suma dels costos és $\leq Q$ i que la puntuació combinada (suma de puntuacions) és màxima. Com que cada categoria conté un ítem $e_{i,0}$ gratis sempre serà possible trobar una solució respectant el pressupost Q . Proporcioneu un algorisme amb cost polinòmic (respecte de la mida de l'entrada) per resoldre aquest problema.

- 3.25. (**Tallant ADN: llargada**) Una operació habitual als laboratoris de biologia molecular és la de trossejar una cadena d'ADN, tallant-la a unes determinades posicions d'interès. En termes computacionals, s'acostuma a representar una cadena d'ADN com un mot sobre l'alfabet $\{A, C, G, T\}$.

Assumim que tallar una cadena d'ADN $c = c_1 \cdots c_n$ de longitud n en dos trossos qualssevol té cost n . Quan parlem de fer un tall a la posició p_i , el símbol d'aquella posició queda com a últim símbol del tros esquerre (és a dir, tallem darrera de p_i).

És fàcil veure que, si es vol fer una seqüència de talls a unes determinades posicions d'interès p_1, \dots, p_k , amb $p_i \in [1, \dots, n]$, l'ordre en què es facin aquests talls afectarà el cost total de l'operació. Preneu com a exemple el cas d'haver de trencar una cadena de longitud $n = 200$ a les posicions 20, 80, 100 i 150. Si fem els talls d'esquerra a dreta, primer tallem una cadena de 200 (a la posició 20) i ens queden dos trossos, un de 20 i un altre de 180; el segon tall el farem a la posició 80 amb cost 180, etc. El cost total serà $600 = 200 + 180 + 120 + 100$. En canvi, podríem fer el primer tall a la posició 100 (amb cost 200) i després tallar cadascun dels dos trossos ($c_1 \cdots c_{100}$ i $c_{101} \cdots c_{200}$) d'esquerra a dreta. Això ens donaria un cost de $480 = 200 + (100 + 80) + (100)$.

Doneu un algorisme de PD tal que, donada una cadena d'ADN i la seqüència de posicions a l'esquerra de les quals volem tallar la cadena, ens proporcioni la forma menys costosa de tallar-la. El vostre algorisme ha de proporcionar el millor cost possible per realitzar els talls i una estructura de dades que ens permeti esbrinar en quin ordre procedir amb els talls per tal d'aconseguir el cost òptim.