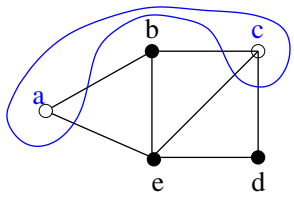
41. El problema del Tall Maxim (MAX-CUT). Donat un graf no dirigit $G = (V, E)$, el problema del maximum cut (MAX-CUT) es trobar la partició $S \cup \bar{S}$ de V tal que maximitze el nombre d'arestes entre S i $\bar{S}$ ($C(S, \bar{S})$,on $S \subseteq V$ i $\bar{S} = V - S$. La maximització entre totes les possibles particiones de V. El problema del MAX-CUT és NP-hard en general. Donat $G = (V, E)$ considereu el següent



$S=\{a,c\}$ $\bar{S}=\{b,e,d\}$

$C(S,\bar{S})=\{(ab),(ae),(bc),(ce),cd)\}$

Figura 1: Exemple de MAX-CUT, amb cost òptim 5

algorisme golafre (greedy) pel problema del MAX-CUT:
Ordeneu els vèrtexs en ordre decreixent respecte al seu grau (nombre veïns). Considereu el resultat de l'ordenació s'escriu com a $(v_1, v_2, \ldots, v_n)$.
Inicialment tenim $S = \emptyset = \bar{S}$. Al primer pas $v_1 \in S$. Al pas i-èsim col.loquem v_i a S o $\bar{S}$ de manera que maximitze $|C(S, \bar{S})|$.

(a) Demostreu la correctessa i doneu la complexitat de l'algorisme golafre?

(b) Demostreu que aquest algorisme golafre és una 2-aproximació al MAX-CUT

42. Donat com a entrada un graf no dirigit $G = (V, E)$, definim el problema de GST (graf sense triangles) com el problema de seleccionar el màxim nombre d'arestes E, de manera que el graf $G' = (V, E')$ no contingui cap triangle (i.e. no hi han 3 vèrtexs x, y, z tal que $(x, y), (x, z)$ i (y, z) siguin arestes a G'). Aquest problema és NP-hard. Dissenyeu un algorisme que done una 2-aproximació al problema. Quina és la complexitat del vostre algorisme?.

43. Un professor té n revisions d'examen. Abans de començar el professor mira la llista d'estudiants que han sol.licitat revisió i pot calcular, per a cada i, el temps t_i que utilitzarà per atendre l'i-èsim estudiant. El temps d'espera de l'estudiant i és el temps que el professor triga a revisar els exàmens dels estudiants que van abans que i. Dissenyeu un algorisme per a computar l'ordre s en que els estudiants $1, \ldots, n$ han de revisar l'examen de manera que es minimitze el temps total d'espera: $T = \sum_{i=1}^{n}$(temps espera estudiant$i$).
Doneu-ne la complexitat i la correctesa

44. Si utilitzem l'algorisme de Huffmann per a comprimir un text format per n, símbols que apareixen amb freqüències $f_1, f_2, f_3, \ldots, f_n$ quina és la màxima longitud de compressió d'un símbol que podem obtenir? Doneu un algorisme per obtenir freqüències on es compleix aquesta condició per un valor de n donat.

45. Considereu el següent problema de planificació. Teniu n treballs. El treball i be determinat pel primer instant de temps al que es pot iniciar, s_i, i el seu temps de processament, p_i. Tenim només un processador i volem una planificació preemptiva a la que l'execució d'un treball es pot aturar a

qualsevol instant del temps per tal de continuar-la mes endavant. L'única restricció es que cap treball es pot iniciar abans del seu primer temps d'inici. Per exemple si $n = 2$ i l'entrada es $s_1 = 2$, $p_1 = 5$ i $s_2 = 0$, $p_2 = 3$, la planificació a la que el treball 2 s'executa del temps 0 al 2; el treball 1 del temps 2 al 7 i finalitzem el treball 2 del temps 7 al 8 es una planificació preemptiva legal. L'objectiu és obtenir una planificació preemptiva legal que minimitzi el temps en el que s'han completat totes les tasques. Doneu-ne la complexitat i la correctesa del vostre algorisme.

46. 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) 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
```

    (b) 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.

Una Solució:

(a) 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) 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.

47. 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.

48. 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 $< l_i, w_i, h_i >$. 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?.

49. 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 $(x_1, x_2, \ldots, x_n)$ 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 decrementa 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 $s_1 > s_2 > s_3 > \cdots s_n$. Assumim que el dia que es fa el reboot, no es pot donar servei a cap usuari. Donada una seqüència de carrega $(x_1, \ldots, x_n)$ i de limitacions $(s_1, \ldots, s_n)$, 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 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.

50. 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?

51. El problema de la *partició lineal* es el següent: Donada una seqüència de n valors positius, $(s_1, \ldots, s_n)$ volem obtenir una seqüència de $r+1$ valores, $i_1, \ldots, i_{r+1} \in \{1, \ldots, n+1\}$ 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 $i_1, \ldots, i_{r+1}$ se li assigna el cost $S(i_1, \ldots, i_{r+1}) = \max_{1 \leq j < r} S_j$. Volem obtenir seqüència que proporcioni r rangs amb cost mínim.

    Per exemple, si els valors són:

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

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

    Però siels valors són:

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

    i $r = 3$, 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.

52. 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 $l_1, l_2, \ldots, l_n$, 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 quadrat 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.

53. 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 ocórren 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 sóm 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 ≤ 1/minut
  $S := S \cup \{j\}$
  $p(j) := d_j$
fimentre
tornar S.

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

54. Us donen una cadena de n caràcters $s[1 \ldots n]$, que pot ser un text on totes les separacions entre mots ha desaparegut, per exemple *aquestaesunafrsequepodiaserunexemple*. Volem reconstruir el text original amb ajut d'un diccionari $D(.)$, 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 valida, fes que el teu algorisme escrigui la frase correcta, amb separacions entre mots.

55. 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 Neurones (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 a 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 $x_1, x_2, \ldots, x_n$ 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 descarregat, per tant si el polsador s'utilitza per primer cop al j-èsim segon, pot destruir $f(j)$ robots.

Donada la informació $x_1, x_2, \ldots, x_n$ 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.
Dissenyeu un algorisme eficient tal que donats $x_1, x_2, \ldots, x_n$ i f, retorni la seqüència de pulsacions que maximitzi el nombre de robots destruïts.

56. 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. Dissenyeu 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.

57. (**L'alineament de seqüències**). Quan es descobreix un nou gen, una manera estàndard de descobrir la seva funció és mirar a una base de dades de gens que ja s'han estudiat molt i per als què es coneix perfectament el què fan, i trobar coincidències el més ajustades que sigui possible entre el nou gen i algun dels gens coneguts. La proximitat de dos gens es mesura pel grau d'alineació. Per formalitzar això en termes computacionals, podem pensar que un gen és una cadena sobre un alfabet $\Sigma = \{A, G, C, T\}$. Considerem dos gens $x = ATGCC$ i $y = TACGCA$, una alineació de x i y és una manera de fer coincidir aquestes dues cadenes escrivint-los en columnes, per exemple:

$$\begin{array}{cccccc} - & A & T & G & C & - \\ T & A & C & G & C & A \end{array}$$

on el caràcter $-$ denota un forat o una no-coincidència. Els caràcters a cada cadena han d'aparèixer en ordre, i cada columna han de contenir un caràcter d'almenys d'una de les cadenes.

La qualitat d'una alineació s'especifica utilitzant una matriu de puntuació δ amb dimensió 5×5. A l'exemple previ, l'alineació té una puntuació resultant de:

$$\delta(-, T) + \delta(A, A) + \delta(T, C) + \delta(G, G) + \delta(C, C) + \delta(-, A).$$

L'elecció dels valors de δ no és senzilla i depèn de l'aplicació concreta. N exemple de matriu de puntuacions és següent:

$$\begin{array}{c} \begin{array}{ccccc} A & T & C & G & - \end{array} \\ \begin{pmatrix} +1 & -1 & -1 & -1 & 0 \\ -1 & +1 & -1 & -1 & 0 \\ -1 & -1 & +1 & -1 & 0 \\ -1 & -1 & -1 & +1 & 0 \\ 0 & 0 & 0 & 0 & \end{pmatrix} \end{array} \begin{array}{c} A \\ T \\ C \\ G \\ - \end{array}$$

Segons aquesta matriu de puntuació l'alineació proposta tindria una puntuació de 2.

Doneu un algorisme que prengui com a entrada dues cadenes X i Y d'ADN, amb longitud n i m respectivament ($X[1, \ldots, n], Y[1, \ldots m]$) i una matriu de puntuació δ, i retorni l'alineació amb puntuació més alta. El temps d'execució del vostre algorisme ha de ser $O(mn)$.

58. Donada una cadena $x \in \{0,1\}^n$, escrivim x^k per a representar x copies de x concatenades (una darrera l'altra) Direm que una cadena x' és una repetició de x si existeix un $k \in \mathbb{N}$ tal que x' és un prefix de x^k (per ex. $x' = 10110110110$ és una repetició de $x = 101$.

Diem que una cadena s és una *trena* de x i y si podem particionar els símbols de s en dues subsequències s' i s'', no necessàriament contigues, de manera que s' és una repetició de x i s'' és una repetició de y. Es a dir, cada símbol de s ha de ser a s' o a s". Per exemple, si $x = 101$, $y = 00$, i $s = 100010101$. s és una trena de x i y, ja que els símbols a les posicions 1,2,5,7,8,9 (=101101) són un repetició de x, i la resta dels símbols forment 000 que és una repetició de y).

Doneu un algorisme eficient tal que donats x, y i s decideixi si s és una trena de x i y.

Una Solució:

Primer de tot establirem notació i definicions auxiliars.

Si $w = w_1 \cdots w_n$, $w[k]$, per $1 \leq k \leq n$, és la subcadena formada per els primers k caràcters de s.

$t[w, i] = w^i[i]$ és el prefixe de longitud i de w^i

Utilizarem V per a representar "cert".

Sigui $x = x_1 x_2 \ldots x_p$, $y = y_1 y_2 \ldots y_q$ i $s = s_1 s_2 \ldots s_n$ una entrada del problema a resoldre.

Per a establir una recurrència que ens permeti resoldre'l considerarem un problema auxiliar que resoldrem recursivament. Aux: determinar si la subcadena $s[k]$, $1 \leq k \leq n$, és una trena de $t[x, i]$ i $t[y, j]$ per a tots els valors possibles de i, j, k on $k = i + j$.

Volem calcular una taula $C(i, j)$, $i + j \leq n$, tal que $C(i, j) = V$ sii $s[i+j]$ és una trena de $t[x, i]$ y $t[y, j]$.

Aquesta condició és equivalent a dir que $s[i + j]$ es pot dividir en s' ($|s'| = i$) una repetició de x i s'' ($|s''| = j$) una repetició de y.

Observem que quan la descomposició és possible l'últim caràcter de $s[i+j]$ ha de coincidir amb l'últim caràcter de $t[x, i]$ o de $t[y, j]$ (o amb els dos). En cas contrari la descomposició no és possible. Al primer cas, $s_{i+j} = x_{i'}$, per $i' = i \mod p$ i tenim, a més, què $[s[i + j - 1]$ ha de ser una trena de $t[x, i - 1]$ i $t[y, j]$. Al segon cas, $s_{i+j} = y_{j'}$, per $j' = j \mod q$ i tenim què $[s[i+j-1]$ ha de ser una trena de $t[x, i]$ i $t[y, j - 1]$.

Aquesta observació sobre suboptimalitat de les solucions ens porta a la recurrència:

$$C(i, j) = [(s_{i+j} = x_{i \mod p}) \vee C(i - 1, j)] \wedge [(s_{i+j} = y_{j \mod q}) \vee C(i, j - 1)],$$

amb $C(0, 0) = V$; per $j \in [n]$, $C(0, j) = V$ sii $s_1 s_2 \ldots s_j$ és una repetició de y; per $i \in [n]$, $C(i, 0) = V$ sii $s_1 s_2 \ldots s_i$ és una repetició de x.

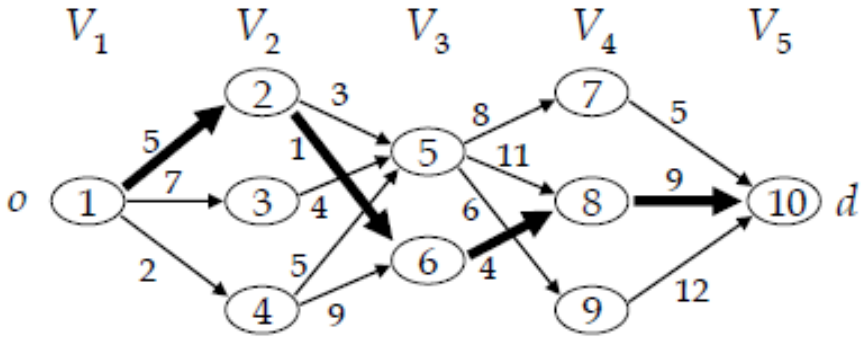
La resposta final de l'algorisme és $\vee_{i+j=n, i \mod p=0, j \mod q=0} C(i, j)$.

El temps total per implementar aquest algorisme recursiu amb un esquema de PD és $O(n^2)$ ja que el cost per element és constant.

Per finalitzar s'ha de fer un recorregut de la diagonal amb suma n de la matriu C acumulant els valors de les posicions que ens interessin. Això ens dona un temps addicional de $O(n)$.

Per tant el cost total és $O(n^2)$ i fem servir espai $O(n^2)$.

59. Camí més curt en DAG multi-etapa. Un graf dirigit acíclic (DAG) $G = (V, E)$ és multi-etapa quan els vèrtexs es poden dividir en k subconjunts $V = V_1 \cup V_2 \cup \cdots \cup V_k$ tal que cada arc va des un subconjunt V_i al següentV_{i+1}. A més, el primer i últim subconjunt contenen exactament un vèrtex cadascun (és a dir, $V_1 = \{o\}$ i $V_k = \{d\}$ als que direm font i embornal, respectivament. A la figura següent teniu un exemple de DAG multi-etapa ponderat.



Dissenyeu un algorisme que calculi el camí més curt a partir de o a d en un DAG multi-etapa ponderat.

60. 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 palìndrom. Per exemple, si $A = ababbbabbababa$ tenen particions $aba|b|bbabb|a|b|aba$ i $aba|bbb|abba|baba$ 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.

61. Considerem una xarxa d'agents mòbils, connectats per radiofreqüència, que tenen una distància curta de comunicació r. Aquestes xarxes es poden representar per un tipus especial de grafs anomenats *random geometric graphs* on els agents venen representats per nodes, i dos agents x i y estan connectats si x cau dintre del cercle amb centre y i radi r. Aquest tipus de xarxes són dinàmiques, per tant en l'instant t pot haver-hi una aresta (x, y), i en l'instant $t + 1$, l'aresta pot deixar d'existir (cada agent pot sortit en direccions contràries). Volem dissenyar un sistema eficient, per mantenir un camí de connectivitat entre dos agents especificats. Volem que el camí sigui curt, però volem que aquest camí no canvii encara que el graf perdi i guanyi arestes. Sigui V el conjunt de nodes que representen els agents, i en l'instant t sigui E_t el conjunt d'arestes. Per tant, als instants $t = 0, 1, \dots, m$ les arestes evolucionen $E_0, E_1, \dots E_m$, cosa que a cada t ens dona un graf $G_t = (V_t, E_t)$. La xarxa d'agents té una representació temporal com una seqüència $\{G_i\}_t$ de grafs, sobre el mateix conjunt V de nodes. Assumim que cada G_i és connex.

    Considerem dos nodes particulars $s, t \in V$. Sigui P_i un camí mínim de s a t a G_i, i sigui $|P_i|$ la seva longitud (nombre d'arestes). Volem tenir una seqüència de camins $P_0, P_1, P_2, \dots, P_m$ un camí mínim per a cada G_i. A més, si definim el nombre de canvis $c(P_1, P_2, \dots, P_m)$ a la seqüència com el nombre de índexs i, $(1 \leq i \leq m - 1)$ per als quals $P_i \neq P_{i+1}$, volem minimitzar aquest nombre de canvis, ja que a cada canvi, tindrem un cost extra k.

    Sigui k una constant donada, definim el *cost* de $P_0, P_1, P_2, \dots, P_m$ com

$$CT(P_0, P_1, P_2, \dots, P_m) = \sum_{i=1}^{m} |P_i| + k \cdot c(P_1, P_2, \dots, P_m).$$

    (a) Suposeu que existeix un únic camí P, que és el mateix camí entre s i t a $G_0, G_1, G_1, \dots, G_m$. Doneu un algorisme polinòmic per a trobar-lo.

    (b) Doneu un algorisme polinòmic per a trobar una seqüència de camins $P_0, \dots, P_m$ amb cost mínim, on P_i és un camí mínim de s a t.

62. Ara que a Catalunya les eleccions s'apropen, hi ha maneres d'agrupar els districtes electorals en circumscripcions electorals de manera molt acurada per tal d'arribar a resultats que afavoreixin a un

partit polític en particular. Aquesta "tècnica"rep el nom de *gerrymandering* als USA i es pot traduïr com *districtació*.

Gràcies al software disponible, la *districtació* ha canviat de ser una activitat portada a terme per un grup de gent amb mapes, llapis i paper a ser un procés automàtic. Actualment, la districtació no és més que un problema computacional (a més de ser molt susceptible al frau electoral). Es tenen bases de dades molt acurades per al seguiment demogràfic dels possibles votants, que arriben a poder conèixer el perfil de votant fins a nivell de carrer i fins i tot de vivendes. Aquestes dades es poden processar amb ordinadors per agrupar els votants en demarcacions "apropiades"als interessos del partit.

Suposem que tenim un conjunt de n districtes $D_1, D_2, \ldots, D_n$, cadascun amb m votants. Se suposa que aquests n districtes es reagrupen en dos circumscripcions, i que cadascuna aglutina $n/2$ districtes. Per a cada districte tenim informació sobre quants electors votaran a cada partit (per a simplificar, suposem que només tenim dos partits R i P). Es diu que $D_1, D_2, \ldots, D_n$ és *susceptible de frau electoral* si és possible realitzar la divisió en dos circumscripcions de tal manera que el mateix partit s'assegura la majoria en les dos circumscripcions.
Doneu un algorisme per determinar si un determinat conjunt de districtes *és susceptible de frau electoral*. Doneu el temps d'execució del vostre algorisme, que ha de ser polinòmic en n i m.
Exemple: Suposem que tenim $m = 400$ votants, $n = 4$ districtes i la informació següent sobre els votants: sigui P_i (R_i, respectivament) el nombre de votants del partit P (R) al districte i. Si $P_1 = 55$, $R_1 = 45$, $P_2 = 43$, $R_2 = 57$, $P_3 = 60$, $R_3 = 40$, $P_4 = 47$, $R_4 = 53$, és possible repartir els districtes de manera que el partit P tingui la majoria en les dues circumscripcions: simplement agrupem el districtes D_1 i D_4 que formen la circumscripció C_1 , i agrupem els districtes D_2 i D_3 en la circumscripció C_2. Per tant, aquest conjunt de districtes és susceptible de fraud. I malgrat que P té en la població total una escassa majoria de 205 contra 195, acaba guanyant en les dues circumscripcions C_1 i C_2.
Noteu, que si $P_1 = 90$, $R_1 = 10$, $P_2 = 45$, $R_2 = 55$, $P_3 = 45$, $R_3 = 55$, $P_4 = 45$, $R_4 = 55$, aquest conjunt de districtes no és susceptible de frau electoral.

63. 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 milions / 40 = 1,25 milions de dòlars. Les situacions d'arbitratge de divises que ens plantegem són més complexes i implican 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 ⇒ 741 Euros ⇒ 1,012.206 Canadian dollars ⇒ \$1,007.14497.

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

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

64. 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 $w : E \to \mathbb{R}$ i un punt $s \in V$ l'arbre de camins mínims arrelat a s és un subgraf $T' = (V', E')$ de G tal que:

    (a) T' és un arbre, i per tant $|E'| = |V'| - 1$,

    (b) hi a un camí de s fins a qualsevol vertex a V',

    (c) per a qualsevol $u \in V'$, la distancia de s a u a T' és la mateixa que la distancia de s a u a G.

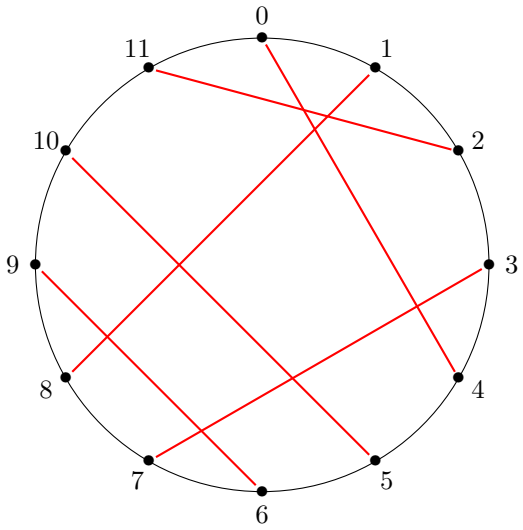
    Recordeu que, igual que succeix amb el MST, donat un $s \in V'$, G pot tenir més d'un arbre de camíns mínims arrelat a s.

    (a) Demostreu si és cert o no que donat qualsevol graf connex i no dirigit G, amb $w : E \to \mathbb{R}^+$, 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 $w : E \to \mathbb{R}^+$ i connex, tal que G tingui un arbre de camins mínims T' i un MST T que no comparteixen cap aresta.

65. Donat un conjunt de n cordes en el cercle unitat diem que un subconjunt de cordes es *viable* si no hi han dues cordes que es tallen. Volem trobar un subconjunt viable 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 $0, \ldots 2n-1$ 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 instancia amb 6 cordes:



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

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

(a) Per $0 \leq i < j \leq 2n-1$, proporcioneu una recurrència que permeti calcular $T(i,j)$.

(b) Proporcioneu un algorisme que, donat un conjunt de cordes en el cercle unitat, obtingui un conjunt viable amb mida màxima en temps polinòmic.

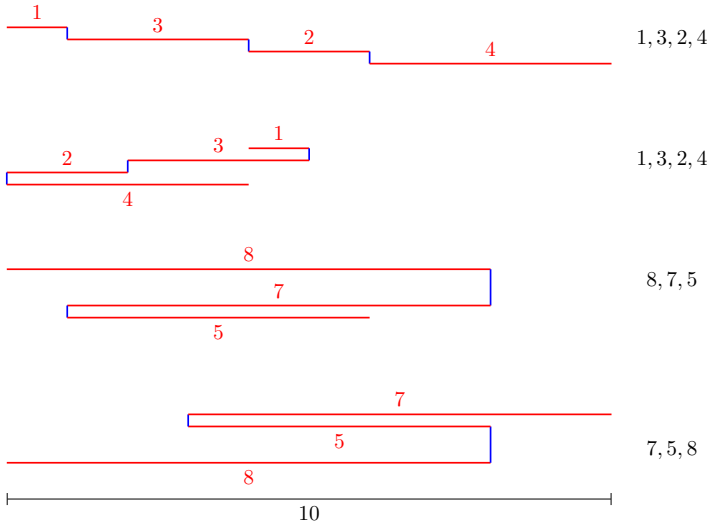
66. El dia de Sant Tomàs, la coral dels alumnes de la facultat organitzen una sortida col·lectiva a esquiar. A l'estació d'esquí on van, tenen, m parells d'esquís, on l'alçada del parell i-èsim és s_i. Sigui n el nombre d'alumnes que no tenen esquís propis i per tant han de llogar-los a l'estació d'esquí. Sigui h_i l'alçada de l'alumne i-ésim (d'entre els alumnes que no tenen esquís). Es ben coneguda la regla que cada esquiador ha d'esquíar amb uns esquís de manera que l'alçada dels esquís sigui el més semblant possible a la seva pròpia alçada. Dissenyeu un algorisme eficient per assignar esquís als n alumnes sense esquís, de manera que es minimitze la diferència total d'alçada entre els esquís i els corresponents alumnes.

67. Un metre de fuster (com el de la figura de sota) està format per uns quants segments de fusta habitualment iguals. Cada segment és rígid i s'uneix al previ i/o al següent pels extrems de manera que es pot rotar completament a les unions.

En aquest problema considerarem una generalització de metre de fuster en el que els segments poden tenir longituds diferents encara que tots tenen la mateixa amplada. A més cada segment té com a

molt 100cm de llargada. Així un metre de fuster està format per n segments de llargades $l_1, \ldots, l_n$ (en aquest ordre) on totes les longituds dels segments son enters al interval $[0, 100]$. Per simplificar la notació considerarem també els extrems $A_0, \ldots A_n$, on A_0 és l'extrem lliure del primer segment, A_1 és l'extrem comú al primer i segon segment, etc, i A_n és l'extrem lliure del segment n-ésim.

Volem analitzar el problema de plegar el metre per tal de ficar-lo a dintre d'una caixa. Per exemple, si els segments són de longitud 1, 3, 2 i 4, el metre es pot guardar en una caixa de longitud 10 (plegar a l'interval [0,10]), però podem fer-ho també en una caixa de longitud 5 (a l'interval [0,5]). Si els segments tenen longitud 8, 7 i 5 en aquest ordre, es pot guardar plegat en una caixa de 8, però si els segments son 7, 5 i 8, llavors la caixa més petita en la que es pot plegar té longitud 10 metres. A la figura de sota teniu una representació estilitzada i bidimensional d'aquests plegaments.



(a) Considereu l'algorisme següent

```
1:  procedure FOLD INSIDE INTERVAL(L(n))
2:      Let m = max L[i]
3:      Place A_0 at position 0
4:      for i = 1, ..., n do
5:          if it is possible to place A_i to the left of A_{i-1} inside [0, 2m] then
6:              place A_i to the left of A_{i-1}
7:          else
8:              place A_i to the right of A_{i-1}
9:          end if
10:     end for
11: end procedure
```

Demostreu que Fold inside interval determina un plegat que ens permet ficar el metre dintre de l'interval $[0, 2m]$ on m es la longitud del segment més llarg i analitzeu-ne el seu cost.

(b) Considereu el problema Min-Fold: Donat un metre de fuster, format per n segments de llargades $l_1, \ldots, l_n \in \mathbb{N}$ (en aquest ordre), $0 < l_i \leq 100$, trobar la llargada ℓ més petita que ens permeti ficar el metre dintre de l'interval $[0, \ell]$.

Dissenyeu un algorisme que ens permeti resoldre Min-Fold i analitzeu-ne el seu cost temporal i espacial.

Ajut: Penseu en com resoldre recursivament el problema de determinar si un metre de fuster es pot ficar a dintre de l'interval $[0, k]$ posant-hi l'extrem A_0 a la posició $j \in [0, k]$, per valors raonables de k.

(c) Analitza el cost de l'algorisme proposat a l'apartat (b) en el cas que els segments del metre de fuster poden tenir qualsevol longitud.

(d) És Fold inside interval una 2-aproximació a Min-Fold?

Una solución.

(a) Solo necesitamos comprobar que cuando colocamos A_i a la izquierda de A_{i-1} la posición de A_i está dentro de $[0, 2m]$. En este caso sabemos que A_i no se puede colocar a la derecha, por lo tanto si $j \in [0, 2m]$ es la posición de A_{i-1}, tenemos que $j + L[i] > 2m$, como $L[i] \leq m$, tenemos que $j > m$. Por lo tanto $j - L[i] > 0$. Por lo tanto el plegado nos permite colocar el metro en $[0, 2m]$.

(b) Voy a utilizar una variación recursiva del algoritmo FOLD INSIDE INTERVAL par resolver Min-Fold. Para un valor de k fijado, el algoritmo resolvera recursivamente el problema de determinar si se puede o no plegar el metro $L_i, \ldots, L_n$ dentro de $[0, k]$ condicionado a que A_i se ubique en la posició $j \in \{0, \ldots, k\}$.

```
1: procedure FOLD INSIDE INTERVAL REC(i, j)
2:     Place A_i at position j
3:     Left = Right = False
4:     if j − L[i] ≥ 0 then
5:         (it is possible to place A_{i+1} to the left of A_i inside [0, k])
6:         Left = FOLD INSIDE INTERVAL REC(i + 1, j − L[i])
7:     end if
8:     if j + L[i] ≤ k then
9:         (it is possible to place A_{i+1} to the right of A_i inside [0, k])
10:        Right = FOLD INSIDE INTERVAL REC(i + 1, j + L[i])
11:    end if
12:    return (Left or Right)
13: end procedure
```

Como una vez hemos ubicado A_i en una posición j, el punto A_{i+1} solo puede ubicarse a la derecha o a la izquierda de j, el algoritmo explora todas las posibilidades y por ello es correcto. El algoritmo solo tiene dos parámetros i, $0 \leq i \leq n$, y j, $0 \leq j \leq k$. Por lo que el número de subproblemas es nk. El costo implementándolo con meomización o con tabla será $O(nk)$.

Para determinar si el metro se puede plegar en $[0, k]$ tendríamos que ver si para algún valor de $j \in [0, k]$ FOLD INSIDE INTERVAL REC$(1, j)$ devuelve cierto. Tendremos un coste adicional $O(k)$.

Finalmente, para resolver Min-Fold, tendriamos que calcular el menor valor k^* para el que el metro se puede plegar en $[0, k^*]$. Por el apartado (a) sabemos que $k^* \leq 2m$. Podemos implementar una búsqueda dicotómica usando el algoritmo previo. El coste total es $O(nm \log m)$. Teniendo en cuenta el enunciado, $m \leq 100$, por lo que el coste del algoritmo es $O(n)$.

(c) Si no tenemos el limite de 100 el coste del algoritmo es $O(nm \log m)$. Como m es un número que es parte de la entrada el coste es pseudopolinómico, y por tanto tiene coste exponencial en el tamaño de la entrada.

(d) Teniendo en cuenta que m es el segmento de longitud máxima, cualquier plegado en $[0, k]$ requiere que $k \geq m$. En particular la solución optima en $[0, k^*]$ tiene que cumplir $k^* \geq m$ y como la que obtenemos en (a) es $2m$, $2m \leq 2k^*$ con lo que podemos concluir que FOLD INSIDE INTERVAL es una 2-aproximación.