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 $D(.)$, 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.

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

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 $(x_1, x_2, \ldots, x_n)$ 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.

3.5. **(Partició lineal)** 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 si els 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.

3.6. **(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.

3.7. **(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. 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.

3.8. **(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.

3.9. **(Parèntesis)** Tenim un conjunt de símbols $A = \{a, b, c\}$. Els elements d'A tenen la taula de multiplicació següent, on les files mostren el símbol del primer operand i les columnes el del segon,

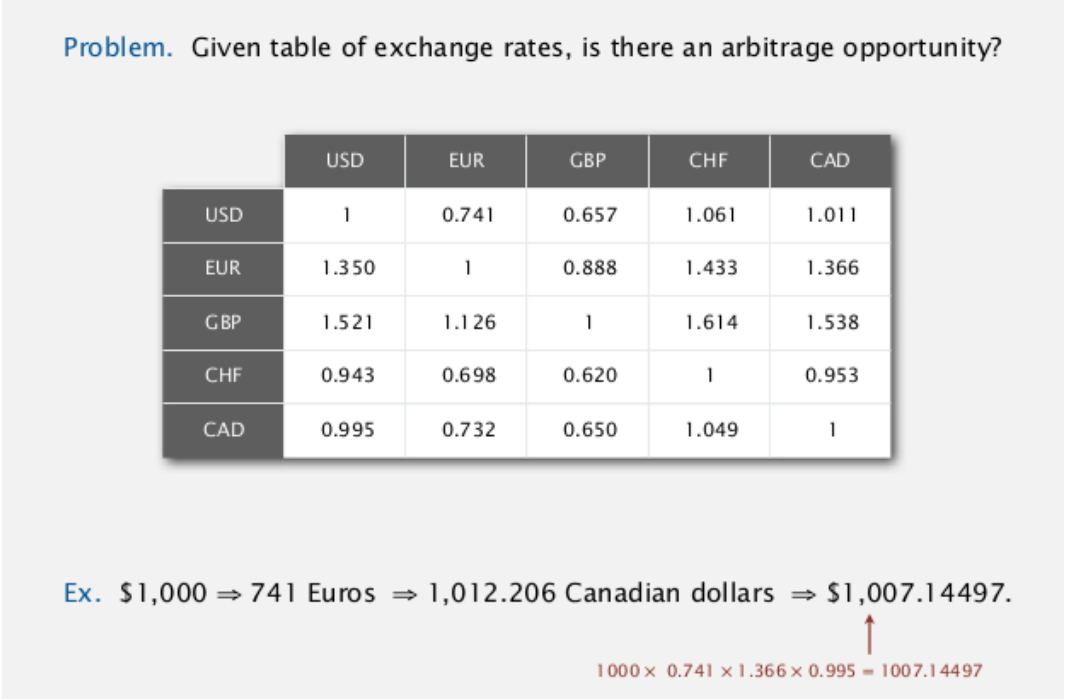
	a	b	c
a	b	b	a
b	c	b	a
c	a	c	c

D'aquesta manera $ab = b$, $ba = c$, etc. Es pot veure que l'operació definida no és commutativa ni associativa i, per tant, el resultat de la multiplicació d'una cadena d'elements d'A depèn de com es posin els parèntesis. Per exemple, si $x = bbbba$, llavors $(b(bb))(ba) = a$ i $(b(b(b(ba))) = a$ però $((bb)(bb))a) = c$.

Dissenyeu un algorisme eficient que, donada una cadena $x = x_1 \dots x_n$ formada per n símbols d'A, determini si és possible o no posar parèntesis a x de manera que el valor de l'expressió resultant sigui a. En cas que sigui possible obtenir a, l'algorisme ha de retornar també com s'han de posar els parèntesis. Analitzeu el cost de l'algorisme i justifiqueu-ne la correctesa.

3.10. **(Distancies 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 $w : E \to \mathbb{R}$, 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.

3.11. **(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 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.



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.

3.12. **(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 $w : E \to \mathbb{R}$ 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.

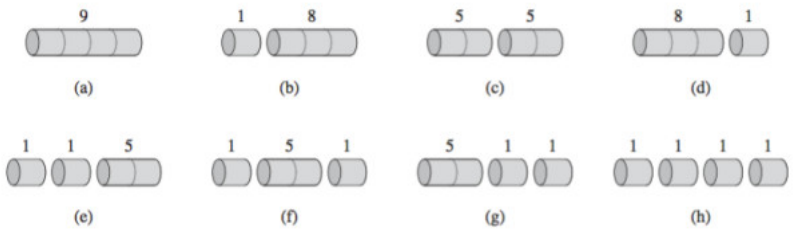
(b) Demostreu si pot haver-hi un graf no dirigit G amb $w : E \to \mathbb{R}^+$ i connex, tal que G tingui un arbre de camins mínims T' i un MST T que no comparteixen cap aresta.

3.13. **(Camins més llargs en DAGs)** Donat com a entrada un graf dirigit $G = (V, E, w)$ on $w : E \to \mathbb{Z}^+$, 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.)

3.14. **(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 dóna 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 afegeix. Observeu que no es demana cap requisit sobre el nombre de talls a fer; podeu fer qualsevol nombre de talls entre 0 i $n - 1$.

3.15. **(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 $l_1, l_2, \ldots, l_n$, 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.

3.16. (**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-ésim 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 assgnar una seqüència contigua de llibres (d'acord amb l'ordre inicial del 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.

3.17. **(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 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 carregat, per tant si el polsador s'utilitza per primer cop al j-èsim segon, pot destruir $f(j)$ robots.

Donada la informació $x_1, x_2, \ldots, x_n$ 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.

3.18. **(Valors calculables)**

Donada una seqüència $A = (a_1, a_2, a_3, ..., a_n)$ de n nombres enters positius, si s'escriuen els enters en l'ordre en què apareixen a A i s'insereixen els símbols + o − davant de cada enter, obtindríem una expressió aritmètica. Si l'avaluació d'una expressió aritmètica d'aquest estil dona valor v, diem que v està *calculat* o que el valor v és *calculable*) per la seqüència A.

Per exemple, si es considera la seqüència $(7, 12, 1, 9, 5)$ de cinc nombres enters positius, tenim:

$$+7 - 12 - 1 + 9 + 5 = 8,$$

$$-7 - 12 + 1 - 9 + 5 = -22,$$

$$-7 + 12 - 1 - 9 + 5 = 0,$$

és a dir, els enters 8, −22, 0 són calculables per aquesta seqüència.

Donada una seqüència $A = (a_1, a_2, a_3, ..., a_n)$, sigui $S = a_1 + a_2 + a_3 + ... + a_n$, i sigui $T \in \mathbb{Z}$. Llavors, perquè T sigui calculable per A, ha de passar que $-S \leq T \leq S$. No obstant això, aquesta condició només és necessària (però no suficient). En l'exemple de cinc elements anterior, $7+12+1+9+5 = 34$, però un enter senar com 23 o un enter parell com 30, malgrat estar dins l'interval $[-34, 34]$, no poden ser calculats per aquesta seqüència.

Donats una seqüència $A = (a_1, a_2, a_3, ..., a_n)$ i un enter T, dissenyeu un algorisme de programació dinàmica (tan eficient com pugueu) per a **determinar si T és *calculable* per la seqüència A donada**, i si ho és, **com**. Podeu assumir que $-S \leq T \leq S$.

Observació: Noteu que pot haver-hi múltiples maneres de calcular el mateix valor T des d'una seqüència donada A. Només cal que en doneu una.

3.19. **(Banquet)**

En Ferran Tallacaps està organitzant un banquet per a la reina de Borbònia i els seus convidats, i té l'encàrrec de col·locar-los asseguts en una sola banda d'una llarga taula de banquet. Disposa de la llista dels $2n$ convidats, on cada convidat i té un enter positiu i distint f_i que indica el **favor** que té amb la reina.

La reina demana a en Ferran que col·loqui els convidats de manera **respectuosa**: ella vol seure al **centre** i vol tenir n **convidats a cada costat**, de manera que el **favor dels convidats decreixi monòtonament** a mesura que s'allunyen d'ella; és a dir, qualsevol convidat assegut entre un altre convidat i i la reina ha de tenir un favor més gran que f_i.

A més, en Ferran sap que tots els convidats s'odien entre ells. Per a cada parell de convidats i, j, en Ferran ha quantificat aquest odi amb un enter positiu $h(i, j) = h(j, i)$, que representa l'**odi mutu** entre ells, on $1 \leq i, j \leq 2n$.

Amb tota aquesta informació sobre els $2n$ convidats del banquet, en Ferran us demana que li dissenyeu un algorisme, tan eficient com pugueu, que determini una disposició respectuosa dels convidats i minimitzi la suma total d'odi mutu entre els parells de convidats asseguts un al costat de l'altre. Tingueu en compte que en Ferran necessitarà saber quina és exactament la posició que ha d'ocupar cada convidat. Justifiqueu la correcció i analitzeu el cost del vostre algorisme.

3.20. **(Profunditat quadrada)** Imagina que comences amb un enter positiu donat n, i vas eliminant repetidament una xifra per un extrem o per l'altre (a la teva elecció), fins que només te'n queda una. La **profunditat quadrada** $S(n)$ de n es defineix com el nombre màxim de nombres quadrats que pots arribar a veure durant aquest procés. Per exemple, $S(32492) = 3$ mitjançant la seqüència:

$$32492 \to 3249 \to 324 \to 24 \to 4$$

ja que 3249, 324 i 4 són quadrats, i no hi ha cap altra seqüència de supressions en què es vegin més quadrats (encara que sí que n'hi pot haver d'altres igual de bones).

Descriviu un algorisme polinòmic per calcular la profunditat quadrada $S(n)$ d'un nombre $(n)_{10} \in \mathbb{Z}^+$ donat. Podeu fer servir que comprovar si un enter és un quadrat perfecte es pot resoldre en temps proporcional al seu nombre de dígits.