Algorithmique Avancée 2018 École Centrale-Supélec

Examen écrit final

Christoph Dürr, Nguyễn Kim Thắng

1 Jeu de boules dans un tube

C'est un jeu à deux joueurs, qui se joue avec un tube transparent rempli initialement de n boules. Le diamètre des boules est juste un peu moins que le diamètre du tube, pour que les boules peuvent se déplacer librement dans le tube, mais sans pouvoir changer d'ordre. Les jeux joueurs s'appellent *blanc* et *noir* et c'est blanc qui commence. Quand c'est au tour d'un joueur, il choisit une extrémité du tube et prelève autant de boules qu'il désire par cette extrémité, mais il doit en retirer au moins une. Ensuite on change de tour. Quand le tube est vide, le jeu est terminé et chacun compte les nombres inscrits sur les boules prélevées par lui. Le but pour un joueur est de maximiser ce nombre total.

On vous donne un tableau $t_0, \ldots, t_{n-1} \in \mathbb{Z}$, décrivant les nombres sur les boules dans le tube initial, et on vous demande quel sera le score du joueur blanc, si les deux joueurs jouent parfaitement. Donnez un programme dynamique en $O(n^3)$ pour ce problème.

Exemples

entrée	sortie	explication
4 -10 -20 7	-6	blanc prend 4, noir prend 7, -20, puis blanc prend -10
1 2 3 4	10	blanc prend toutes les boules en un seul coup

2 Problème de réservation de voiture

Vous disposez d'un seule voiture tout terrain, que vous mettez en location pour des touristes. Plusieurs groupes ont fait chacun une demande de réservation chez vous. Chaque réservation i (pour $1 \leq i \leq n$) est de la forme: nous souhaitons disposer de la voiture pour p_i jours consécutifs n'importe quand entre le jour r_i inclus et le jour d_i exclus. Par exemple, si $r_i = 1, p_i = 2, d_i = 5$, le groupe i pourra avoir la voiture pendant les jours $1, 2$ ou les jours $2, 3$, voire les jours $3, 4$. Chaque jour la voiture peut être mise à disposition à au plus un groupe. Est-ce qu'il vous est possible de satisfaire toutes les demandes de réservation? Prouvez que ce problème est NP-complet par réduction à partir du problème de 3-Partition.

Rappel: Une instance de 3-partition consiste en $3m$ entiers $a_1, \ldots, a_{3m}$, qu'on doit partitionner en m parties de 3 entiers tel que la somme de chaque partie soit identique. Ce

problème est NP-complet même si $B/4 < a_i < B/2$ et B est entier pour B étant tel que $a_1 + \ldots + a_{3m} = mB$.

Exemples

entrée $\langle r_1, p_1, d_1\rangle \ldots \langle r_n, p_n, d_n\rangle$	sortie	explication
$\langle 1, 2, 5\rangle, \langle 1, 3, 5\rangle$	Non	dans l'intervalle $[1, 5)$ il n'y a que 4 jours, et le total demandé est de 5 jours
$\langle 0, 2, 4\rangle, \langle 2, 3, 6\rangle, \langle 7, 1, 8\rangle$	Oui	`|-|1|1|2|2|2|-|3|`

3 Problème de réservation de femme de ménage

Cette fois-ci vous êtes une femme de ménage, et une demande de réservation $\langle r_i, p_i, d_i\rangle$ veut dire: est-ce que vous pourriez venir travailler p_i jours chez moi entre le jour r_i inclus et le jour d_i exclus. Les jours que vous consacrez à ce client ne sont pas forcément consécutifs, mais doivent être dans l'intervalle demandé et à chaque jour vous pourriez être chez un client à la fois. Pouvez vous satisfaire toutes les n demandes ? Notons $S = \max_i d_i - \min_j r_j$. Trouvez un algorithme polynomial en S pour ce problème, en le réduisant vers un problème polynomial connu.

4 Clustering

On vous donne un entier k et un ensemble de n points $P = \{p_1, \ldots, p_n\}$ dans le plan Euclidien et vous devez les grouper en k *clusters* tel que le diamètre maximal des clusters soit minimal. Celui-ci est défini comme la plus grande distance entre deux points appartenant à un même cluster.

Ce problème est NP-difficile, mais voici un algorithme polynomial qui produit une 2-approximation, voir Figures sur la page suivante. Soit d la distance Euclidienne dans le plan, et pour un point p et un ensemble de points S notons $d(p, S) = \min_{p' \in S} d(p, p')$.

> Choisir un premier centre $\mu_1 \in P$. Pour pour j de 2 à k, choisir le centre μ_j parmi les points de P de plus grande distance de $\mu_1, \ldots, \mu_{j-1}$, donc μ_j est un maximiseur de $\max_{p \in P} d(p, \{\mu_1, \ldots, \mu_{j-1}\})$. Puis les clusters $C_1, \ldots, C_n \subseteq P$ sont construits en affectant chaque point à un centre le plus proche.

La solution produite est une 2-approximation et la preuve est similaire à celle pour le problème *Vertex Cover* (couverture des arêtes par des sommets). On vous demande d'écrire cette preuve et on vous donne quelques indices. Considerez un point $q \in P$ de distance maximale de $\mu_1, \ldots, \mu_k$, et notez r cette distance.

- Prouvez que le diamètre de chacun des k clusters est au plus $2r$.
- Prouvez que le diamètre optimal est au moins r. Pour cela montrez que les points $\mu_1, \ldots, \mu_k, q$ sont à distance au moins r les uns des autres.

Figure 9.5 Some data points and the optimal $k = 4$ clusters.

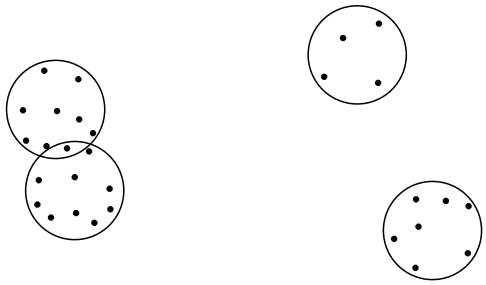

Figure 9.6 (a) Four centers chosen by farthest-first traversal. (b) The resulting clusters.

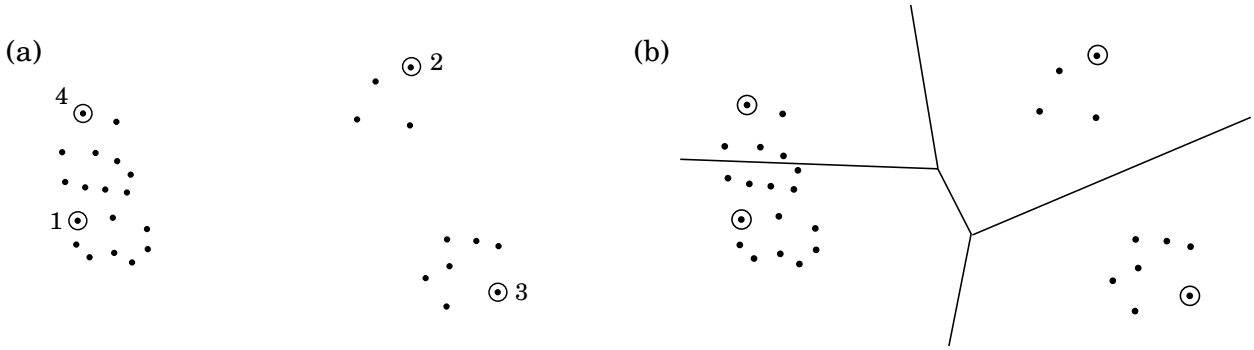

Illustration tirée de *Algorithms* par S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani.

5 Jeu d'arcade

On considère un jeu d'arcade qui se déroule dans une coupe transversale d'un immeuble. Concrètement on vous donne une grille de m lignes et de n colonnes. Les lignes sont numérotées de 1 à m de haut en bas, et les colonnes de 1 à n de gauche à droite. Chaque ligne représente un étage de l'immeuble, et chaque case de la grille une portion de l'étage. On distingue trois différentes cases, la case normale indiquée par un tiret, la case ouverture indiquée par le symbole $\vee$ et la case cœur indiquée, vous l'aurez compris, par un cœur. Dans ce jeu on manipule un personnage, qui initialement se trouve dans la case $(1, 1)$, et le but est de le déplacer vers la case (m, n) en ramassant le plus de cœurs que possible. On appele le nombre de cœurs ramassés le *score*, et on dit que le score est $-\infty$ si le personnage ne peut pas atteindre la case finale (m, n). Pour atteindre le but on peut déplacer le personnage avec les flêches gauche ou droite de la manière suivante.

Si vous actionnez la flèche droite — le cas gauche est similaire — et que le personnage n'est pas encore dans la dernière colonne n, alors il se déplacera une case vers la droite. Si cette case contient un cœur le personnage le ramasse. Si cette case est une ouverture il tombe dans la case en dessous. De nouveau si cette case contient un cœur il est ramassé, mais si cette nouvelle case est également une ouverture le personnage meurt, car la chute est trop importante. Par exemple ceci sera le cas lors d'un pas gauche à partir de la case $(3, 2)$ de la Figure 1.

Écrivez un algorithme de complexité $O(m \cdot n)$ qui pour une grille $M \in \{-, \vee, \heartsuit\}^{m \times n}$ donnée détermine le score maximal que le personage peut atteindre. La grille donnée est telle que $M[1, 1] = M[m, n] = -$ et la ligne m ne contient pas de $\vee$.

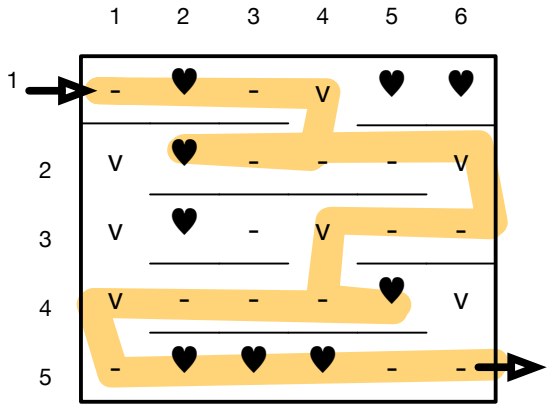

Figure 1: Une instance de ce jeu d'arcade. Son score est de 6.

A Correction

A.1 Jeu de boules dans un tube

Appelons pour $0 \leq i \leq j \leq n-1$, $A[i,j]$ le score maximal que peut obtenir le joueur qui commence sur la configuration avec les boules de i jusqu'à j dans le tube. La réponse au problème est $A[0, n-1]$.

Notons également $T[i,j] = t_i + t_{i+1} + \ldots + t_j$.

Cas de base: $A[i, i-1] = 0$. C'est la configuration vide où aucun joueur ne peut faire de coup.

Cas de récurence: $A[i,j] = T[i,j] - \min_{k=i,\ldots,j} \min\{A[i,k-1], A[k+1,j]\}$.

Explication: Le premier joueur choisit de prélever soit un préfixe soit un suffixe des boules dans le tube, donc il prélève soit les boules de i à k, soit les boules de k à j dans le tube pour un k entre i et j, laissant la configuration $[k+1, j]$ ou $[i, k-1]$ à l'adversaire. Son score sera le total des boules moins le score maximum atteignable par l'advesaire.

Complexité: il y a $O(n^2)$ variables et chacune est calculée par maximisation sur un nombre linéaire d'alternatives. Ceci donne la complexité $O(n^3)$.

A.2 Problème de réservation de voiture

L'encadrement par $B/4$ et $B/2$ des entiers a_i assure qu'une partition avec la bonne somme B pour chaque partie aura exactement 3 entiers par partie.

La réduction consiste en $n = 4m$ réservations. Les $3m$ premières réservations ont $r_j = 0, d_j = m(B+1)$ et $p_j = a_j$. Puis pour chaque $k = 1, \ldots, m$ il y a une réservation *unitaire* $j = 3m + k$ avec $r_j = (k-1)(B+1), d_j = r_j + 1, p_j = 1$.

Montrons que s'il y a une réservation valide, alors il existe une partition valide. Les réservations unitaires sont fixées tous les $B+1$ unités de temps et divisent l'intervalle $[0, m(B+1)]$ en m blocs de taille B chacuns. Chaque réservation non-unitaire doit être entièrement dans un des blocs. Comme le volume total des réservations est mB, l'affectation réservations non-unitaires vers les blocs forme une solution au problème de partition.

Montrons que s'il y a une partition valide, alors il existe une réservation valide. On commence par placer les réservation unitaires dans les seules emplacements possibles. Puis pour la i-ième partie de la partition, on place les réservation non-unitaires à la suite dans le i-ième bloc et dans un ordre arbitraire. Comme la somme de ces réservations par bloc est exactement B, il n'y aura pas de chevauchement entre les réservations.

A.3 Problème de réservation de femme de ménage

Ce problème peut être vu comme un problème de flot ou un problème de couplage. La réduction vers un problème de flot consiste en un graphe orienté, comportant un sommet

source, un sommet par réservation, un sommet pour chacun des S jours et un sommet puits. La source est connectée à tout sommet réservation i avec une capacité de p_i, qui a sont tour est connecté à tous les sommets jours t avec $r_i \leq t < d_i$ avec une capacité de 1. Finalement les sommets jours sont connectés au puits avec capacité 1. Produire ce graphe est clairement polynomial en n et en S, et donc en S seulement car $n \in O(S)$.

Maintenant s'il y a un flot de valeur $\sum p_i$, alors il y a une solution au problème de réservation. En effet, les capacités étant entières, il existe un tel flot entier, et chaque arête entre les sommets réservation et les sommets jours ont une valeur de flot soit 0 soit 1, indiquant ainsi l'empoi de temps de la femme de ménage. Ici c'est important que le graphe soit orienté pour que l'affectation ne se fasse que dans ce sens. Comme l'arc qui mène un jour vers le puit est de capacité 1, on assure qu'une seule femme de ménage ne soit affectée par jour.

La preuve que l'existence d'une solution au problème de réservation implique l'existence d'une solution au problème de flot est simple et omise.

Une réduction au problème de couplage maximum est également possible, mais il faut traiter à part le cas où $\sum_i p_i > S$, car sinon la réduction ne sera plus polynomiale en S.

A.4 Clustering

- Pour deux points p, p' appartenant au même cluster au centre μ_i on a par définition de r, que $d(p, \mu_i) \leq r$ et $d(p', \mu_i) \leq r$. Par l'inégalité triangulaire on a

$$d(p, p') \leq d(p, \mu_i) + d(\mu_i, p') \leq r + r.$$

- On veut montrer que les points $\mu_1, \ldots, \mu_k, q$ soient à distance au moins r les uns des autres. Ceci suffit pour montrer que le diamètre optimal est au moins r. En effet au moins deux des $k+1$ points doivent faire parti d'un même cluster parmi les k clusters, qui a donc diamètre au moins r. Observons à présent que q était candidat à chaque sélection de centre μ_i et donc la distance entre μ_i et les centres $\mu_1, \ldots, \mu_{i-1}$ est au moins

$$d(q, \{\mu_1, \ldots, \mu_{i-1}\}) \geq d(q, \{\mu_1, \ldots, \mu_k\}) = r.$$

  On en déduit que pour tout $1 \leq i < j \leq k$ on a

$$d(\mu_j, \mu_i) \geq d(\mu_j, \{\mu_1, \ldots, \mu_{j-1}\}) \geq d(q, \{\mu_1, \ldots, \mu_{j-1}\}) \geq r.$$

  On a montré que les points $\mu_1, \ldots, \mu_k, q$ sont à distance au moins r les uns des autres.

A.5 Jeu d'arcade

Appelons un segment une suite consécutive et maximale de cases dans une même ligne qui ne sont que type normal ou cœur. Pour chaque segment v, notons $C[v]$ le nombre de ♡

dans v. Un simple parcours des lignes de M permet de déterminer les segments et leurs valeurs C.

Soit V l'ensemble de ces segments. On défini un graphe orienté $G(V, A)$, où il y a un arc (u, v) entre $u = [i_u, j_u]$ et $v = [i_v, j_v]$ si et seulement s'il existe une colonne $j \in \{i_u - 1, j_u + 1\}$ avec $M[i_u, j] = \vee$ et $j \in [i_v, j_v]$, en d'autres termes s'il est possible d'atteindre le segment v à partir du segment u. Ce graphe orienté est sans cycle, et les valeurs C donnent une pondération sur ses sommets.

Soit v_0 le sommet correspondant au segment contenant la case $(1, 1)$, et v_1 le sommet correspondant au segment contenant la case (m, n). Le but est alors de trouver un chemin de v_0 à v_1 qui maximise le poids total des sommets. Pour cela notons pour tout sommet v, $A[v]$ le poids maximum sur les chemins de v à v_1. Ceci peut se faire par programmation dynamique.

Le case de base est $A[v_1] = C[v_1]$. Le cas de récurence est

$$A[v] = C[v] + \max_{u:(v,u)\in A} A[u],$$

où la valeur du maxmum sur un ensemble vide est définie comme $-\infty$.

Le graphe peut être construit en temps $O(n \cdot m)$ en notant pour chaque case sans $\vee$ le segment qui la contient. Le graphe a alors $O(n \cdot m)$ sommets et arêtes. Il y a autant de variables et chacune se calcule par un maximum sur un nombre constant de valeurs. Ceci montre que la complexité de l'algorithme est de $O(n \cdot m)$.

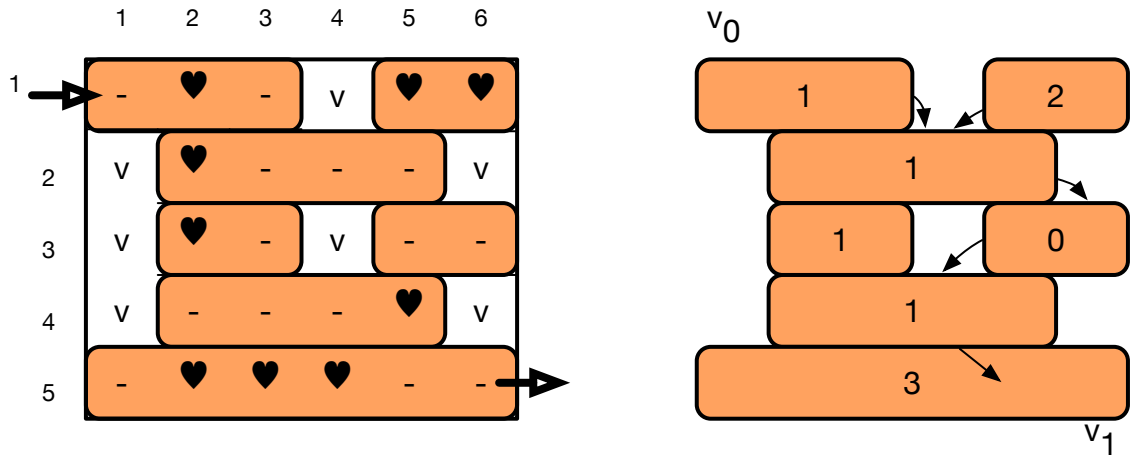

Figure 2: Réduction vers le problème de chemin de poids maximum dans un DAG.