Gestion d'un entrepôt

Épreuve pratique d'algorithmique et de programmation
Concours commun des écoles normales supérieures

Durée de l'épreuve: 3 heures 30 minutes

Juin/Juillet 2010

ATTENTION !

N'oubliez en aucun cas de recopier votre u_0
à l'emplacement prévu sur votre fiche réponse

Important.

Sur votre table est indiqué un numéro u_0 qui servira d'entrée à vos programmes. Les réponses attendues sont généralement courtes et doivent être données sur la fiche réponse fournie à la fin du sujet. À la fin du sujet, vous trouverez en fait deux fiches réponses. La première est un exemple des réponses attendues pour un $\widetilde{u_0}$ particulier (précisé sur cette même fiche et que nous notons avec un tilde pour éviter toute confusion!). Cette fiche est destinée à vous aider à vérifier le résultat de vos programmes en les testant avec $\widetilde{u_0}$ au lieu de u_0. Vous indiquerez vos réponses (correspondant à votre u_0) sur la seconde et vous la remettrez à l'examinateur à la fin de l'épreuve.

En ce qui concerne la partie orale de l'examen, lorsque la description d'un algorithme est demandée, vous devez présenter son fonctionnement de façon schématique, courte et précise. Vous ne devez en aucun cas recopier le code de vos procédures!
Quand on demande la complexité en temps ou en mémoire d'un algorithme en fonction d'un paramètre n, on demande l'ordre de grandeur en fonction du paramètre, par exemple: $O(n^2)$, $O(n \log n)$,...
Il est recommandé de commencer par lancer vos programmes sur de petites valeurs des paramètres et de ***tester vos programmes sur des petits exemples que vous aurez résolus préalablement à la main ou bien à l'aide de la fiche réponse type fournie en annexe.*** Enfin, il est recommandé de lire l'intégralité du sujet avant de commencer afin d'effectuer les bons choix de structures de données dès le début.

1 Introduction

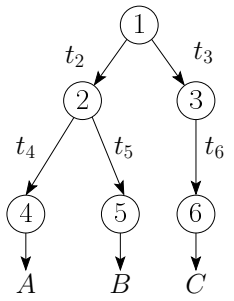
On s'intéresse dans ce sujet à l'organisation d'un atelier. Cette atelier fabrique à partir de plaques d'acier toute une gamme de pièces. La plaque initiale subit un traitement (découpage, emboutissage, etc.) qui produit une ou plusieurs pièces temporaires, qui subissent à leurs tours différents traitements jusqu'à produire les pièces finales. Il est naturel de modéliser l'ensemble de traitements à effectuer par un arbre, dont les nœuds représentent les différents traitements.

Définition 1 (Arbre) *Un arbre est un couple $\mathcal{T} = (V, E)$ où $V = \{1, \ldots, n\}$ est un ensemble de nœuds et $E \subset V \times V$ est une relation (aussi appelée ensemble d'arêtes) sur l'ensemble des nœuds vérifiant la propriété suivante :*

Pour tout $j \in \{2, \ldots, n\}$, il existe un unique $i \in \{1, \ldots, j-1\}$ tel que $(i, j) \in E$.

Un tel i est appelé père de j et noté $p(j)$. Pour tout $i \in V$, on appelle fils de i l'ensemble, noté $f(i)$, des j tels que $(i, j) \in E$. Seul le nœud 1 n'a pas de père : ce nœud particulier est appelé racine de l'arbre. Enfin, les nœuds n'ayant pas de fils sont appelés feuilles.
On appelle degré $d(i)$ d'un nœud i son nombre de fils : $d(i) = card(f(i))$.
On appelle descendants de i les fils de i, les fils des fils de i et ainsi de suite. On appelle sous-arbre enraciné en i l'arbre dont les nœuds sont i et ses descendants, et dont les arêtes sont les arêtes de l'arbre initial qui ont pour sommets i et ses descendants.

Par la suite, on note t_i la pièce temporaire qui sera traité par le nœud i, qui correspond à l'arête $(p(i), i)$.
La figure ci-contre représente par exemple les traitements nécessaires pour obtenir les pièces A, B et C. La plaque initiale est d'abord transformée en deux pièces temporaire t_2 et t_3 par le traitement 1, puis t_2 est à son tour transformée en t_4 et t_5 par le traitement 2, tandis que t_6 est produit à partir de t_3. Chacune des pièces t_4, t_5 et t_6 donnent ensuite les pièces finales A, B et C.



L'atelier dispose d'un entrepôt dans lequel sont stockées les pièces temporaires, avant qu'elles ne soient transformées à leur tour. Cet entrepôt est de taille limitée. Pour simplifier, on supposera que l'entrepôt, ainsi que les pièces, sont de dimension un. On pourra par exemple assimiler l'entrepôt à une étagère linéaire sur laquelle sont posées les pièces temporaires. La taille de l'entrepôt est la longueur de l'étagère L, et chaque pièce temporaire t_i a une taille s_i (c'est la longueur qu'elle occupe sur l'étagère). On suppose qu'il n'y a pas de problème de fragmentation (on peut déplacer les pièces sur l'étagère), de telle sorte que l'entrepôt peut stocker tout ensemble de pièces S tel que $\sum_{t_i \in S} s_i \leqslant L$.
Les pièces finales sont immédiatement acheminées vers leur lieu de stockage finale, elle ne sont donc pas stockées dans l'entrepôt. On modélise le problème sous la forme d'un arbre étiqueté, où le poids d'une arête représente la taille de la pièce temporaire associée.

Définition 2 (Arbre étiqueté) *On appellera arbre étiqueté un triplet $\mathcal{T} = (V, E, w)$ où (V, E) est un arbre et $w : E \mapsto \mathbb{N}^*$ indique la poids de chaque arête.*
Pour un arbre étiqueté, on peut définir le degré pondéré de chaque nœud i de la façon suivante :

$$\delta(i) = \begin{cases} 0 & si\ f(i) = \emptyset \\ \sum_{j \in f(i)} w(i,j) & sinon \end{cases}$$

On note Δ le degré pondéré maximum de tout nœud de l'arbre :

$$\Delta = \max_{i \in V} \delta(i)$$

De même, on définit le poids total W de l'arbre par la relation suivante :

$$W = \sum_{(i,j) \in E} w_{i,j}$$

Le but de cette étude est de déterminer s'il est possible d'effectuer un arbre de traitements avec un entrepôt de taille donnée, ainsi que de déterminer la plus petite taille d'entrepôt nécessaire pour effectuer un arbre de traitements.

2 Génération aléatoire d'arbres

Considérons la suite d'entiers (u_k) définie pour $k \geqslant 0$ par :

$$u_k = \begin{cases} \textit{votre } \mathtt{u_0} \textit{ (à reporter sur votre fiche)} & \text{si } k = 0 \\ 15\,091 \times u_{k-1} \mod 64\,007 & \text{si } k \geqslant 1 \end{cases}$$

Question 1 *Que valent :* ***a)*** u_{10} ***b)*** u_{100} ***c)*** u_{1000}

On s'assurera de pré-calculer et de stocker suffisamment de valeurs de u_n de manière à pouvoir y accéder en temps constant par la suite.

Définition 3 *Pour $n \in \mathbb{N}^*$ on note $\mathcal{T}_n = (V, E, w)$ l'arbre étiqueté défini par :*
- $V = \{1, \ldots, n\}$,
- $E = \left\{ \left(1 + \left\lfloor \frac{(j-1) \times u_{2j}}{64007} \right\rfloor, j\right) | j \in \{2, \ldots, n\} \right\}$,
- $w(p(j), j) = 2.\left(1 + \left\lfloor \frac{400 \times u_{2j+1}}{64007} \right\rfloor\right)$

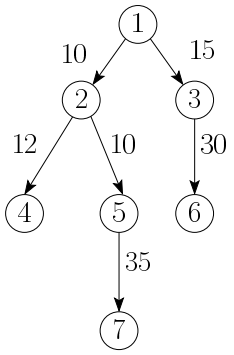
Question 2 *Calculer le plus grand degré des nœuds ainsi que le nombre de feuilles des arbres suivants :* ***a)*** $\mathcal{T}_{10}$ ***b)*** $\mathcal{T}_{100}$ ***c)*** $\mathcal{T}_{200}$

Question 3 *Calculer Δ et W pour les arbres suivants :* ***a)*** $\mathcal{T}_{10}$ ***b)*** $\mathcal{T}_{100}$ ***c)*** $\mathcal{T}_{200}$

3 Test d'heuristiques simples

Dans cette partie, nous allons évaluer quelle est la taille minimale de l'entrepôt nécessaire pour des stratégies simples de parcours de l'arbre. On considère d'abord les stratégies de parcours en profondeur d'abord : lorsqu'on traite un nœud i, on traite ensuite tout le sous-arbre enraciné en i avant de traiter une autre partie de l'arbre.

Lorsqu'un nœud possède plusieurs fils, il faut choisir dans quel ordre les traiter : on décide de traiter d'abord le fils qui a le degré pondéré le plus faible (en cas d'égalité, on choisit le fils de numéro le plus petit). On rappelle qu'une feuille a un degré pondéré nul. Pour l'arbre ci-contre, cette stratégie parcourt les nœuds dans l'ordre suivant : 1, 2, 4, 5, 7, 3 puis 6. La taille de l'entrepôt nécessaire pour cette stratégie est 50, qui correspond au stockage des pièces temporaires t_3 et t_7 après le traitement 5.



Question 4 *Quelle est la taille minimale de l'entrepôt pour traiter avec cette stratégie les arbres suivants :* ***a)*** $\mathcal{T}_{10}$ ***b)*** $\mathcal{T}_{100}$ ***c)*** $\mathcal{T}_{200}$

Nous nous affranchissons maintenant de la contrainte de parcours en profondeur d'abord : à chaque étape, nous choisissons de traiter parmi les nœuds actifs (c'est-à-dire ceux dont le père a déjà été traité), celui qui a le degré pondéré le plus faible (encore une fois, en cas d'égalité, on choisit le nœud de numéro le plus petit). Avec cette stratégie, on traite les nœuds de l'exemple précédent dans l'ordre suivant : 1, 2, 4, 3, 6, 5 et 7. La taille minimale de l'entrepôt est 40, qui correspond au stockage des pièces t_5 et t_6, après le traitement du nœud 3.

Question 5 *Quelle est la taille minimale de l'entrepôt nécessaire pour traiter avec cette stratégie les arbres suivants :* ***a)*** $\mathcal{T}_{10}$ ***b)*** $\mathcal{T}_{100}$ ***c)*** $\mathcal{T}_{200}$

Question à développer pendant l'oral :
- Quelle est la complexité de vos algorithmes pour les questions 4 et 5 ?
- Que pensez-vous du critère de tri des nœuds (degré pondéré croissant) ? Pouvez-vous proposez d'autres critères et les justifier ?

4 Recherche de la taille minimale de l'entrepôt

Dans cette partie, nous allons chercher à trouver la façon de traiter l'arbre qui minimise la taille de l'entrepôt nécessaire. Nous nous intéressons tout d'abord à certains types d'arbres.

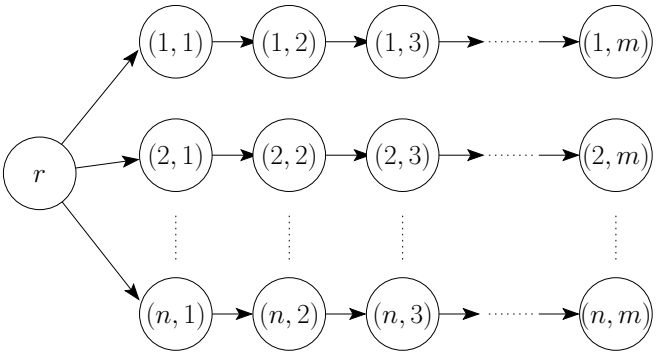
4.1 Cas particulier des arbres homogènes

On s'intéresse ici aux cas particuliers où toutes les pièces sont de même taille ; on supposera que cette taille vaut une unité de stockage pour toutes les arêtes. Pour tester les algorithmes, on utilisera dans cette partie les arbres définis précédemment sans prendre en compte la pondération w des arêtes.

Question à développer pendant l'oral : Soit r la racine d'un arbre, et $f_1, \ldots, f_k$ ses fils. En supposant qu'on connaît la taille minimale de l'entrepôt nécessaire pour traiter chacun des sous-arbres enracinés en $f_1, \ldots, f_k$, comment peut-on en déduire la taille minimale de l'entrepôt nécessaire pour traiter l'arbre enraciné en r, ainsi que l'ordre de traitement ?

Question 6 *En déduire un algorithme calculant la taille minimale de l'entrepôt nécessaire pour effectuer les traitements décrits par les arbres suivants, lorsque toutes les pièces sont de taille unitaire :* ***a)*** $\mathcal{T}_{10}$ ***b)*** $\mathcal{T}_{100}$ ***c)*** $\mathcal{T}_{200}$

4.2 Cas particulier des harpons



On appelle harpon un arbre dont tous les nœuds sont de degré un, sauf la racine. Un harpon est formée de plusieurs branches, qui sont chacunes de chaînes de nœuds. On considère plus particulièrement les harpons $\mathcal{H}_{n,m}$ définis comme suit, et illustré ci-contre : $\mathcal{H}_{n,m}$ est un harpon à n branches, où chaque branche possède m nœuds. On désigne par (i,j) le $j^{\text{ème}}$ nœud de la $i^{\text{ème}}$ branche. Chaque nœud $(i,1)$ est connecté à la racine r. Par commodité, on considère que le nœud $(i,0)$ dénote la racine, pour tout i. Le poids de l'arête $((i,j-1),(i,j))$, correspondant à la taille de la pièce temporaire produite par le nœud $(i,j-1)$ et utilisée par (i,j) est donnée par la formule suivante :

$$w((i,j-1),(i,j)) = 1 + \left\lfloor 100 \left(1 + \frac{j}{m} \times \left(\frac{n \times u_{i\times m+j}}{64007} - 1\right)\right)\right\rfloor$$

Question 7 *Calculer Δ et W pour les harpons suivants :* ***a)*** $\mathcal{H}_{3,5}$, ***b)*** $\mathcal{H}_{10,100}$, ***c)*** $\mathcal{H}_{50,1000}$

Question 8 *Écrire un algorithme permettant de tester si on peut traiter un harpon avec un entrepôt de taille L donné. On pourra par exemple se souvenir de l'arête de poids minimal atteinte sur chaque branche. Tester cet algorithme sur les couples (harpon, taille d'entrepôt) suivants :*

a) *$(\mathcal{H}_{3,5},\ 1,2 \times \Delta(\mathcal{H}_{3,5}))$,*
b) *$(\mathcal{H}_{10,100},\ 1,2 \times \Delta(\mathcal{H}_{10,100}))$,*
c) *$(\mathcal{H}_{50,1000},\ 1,2 \times \Delta(\mathcal{H}_{50,1000}))$.*

Question à développer pendant l'oral :
– Quelle est la complexité de votre algorithme ? Peut-on l'améliorer ?

Question 9 *En vous appuyant sur l'algorithme de la question précédente, calculer la taille minimum de l'entrepôt nécessaire pour les harpons suivants :* ***a)*** $\mathcal{H}_{3,5}$, ***b)*** $\mathcal{H}_{10,100}$, ***c)*** $\mathcal{H}_{50,1000}$

Question à développer pendant l'oral :
– Quelle est la complexité de cet algorithme ?
– Lorsque l'algorithme qui vérifie si le traitement d'un harpon est faisable indique qu'il n'y a pas de solution, comment peut-on en déduire une borne inférieure sur la taille de l'entrepôt nécessaire ? Expliquer comment on pourrait modifier cet algorithme pour calculer la taille minimum de l'entrepôt. Comparez la complexité des deux approches, en fonction du nombre de nœuds dans l'arbre et de la taille maximale S des pièces.

4.3 Cas général

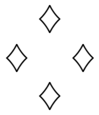
Attention : Les questions de cette partie sont particulièrement difficiles. On conseille aux candidats de ne l'aborder qu'après avoir traité les questions précédentes.

On retrouve ici le cas général de l'arbre. On cherche tout d'abord à vérifier s'il est possible de traiter un arbre avec une taille d'entrepôt donné.

Question 10 *On cherche ici à élargir la notion d'arête minimale atteignable vue pour les harpons. Sur un graphe, on cherchera un état temporaire de taille de stockage minimale, atteignable depuis un nœud i avec un entrepôt de taille L donnée. Par exemple, sur l'arbre représenté à la question 4, en partant du nœud 1 et avec $L = 37$, l'état atteignable de taille minimal consiste à stocker t_3 et t_5, après avoir traiter les nœuds 1, 2 et 4. On ne peut plus effectuer de traitement puisque le traitement du nœud 5 demanderait un stockage de 35 (en plus de $t_3 = 15$), et celui du nœud 6 demanderait un stockage de 30 (en plus de $t_5 = 10$). On notera que lorsqu'on traite une feuille, il n'y a plus de pièce temporaire à stocker, la taille de stockage est donc nulle.*
Écrivez une fonction qui, pour un espace de taille L disponible dans l'entrepôt, renvoie la taille minimal de stockage de tout état atteignable à partir d'un nœud i. Tester votre algorithme sur les arbres suivants, en prenant comme taille d'entrepôt le degré pondéré maximal Δ de l'arbre : ***a)*** *$\mathcal{T}_{10}$,* ***b)*** *$\mathcal{T}_{100}$,* ***c)*** *$\mathcal{T}_{200}$.*

Question à développer pendant l'oral : Expliquez comment on peut utiliser l'algorithme précédent pour vérifier s'il est possible de traiter un arbre avec une taille d'entrepôt donnée.

Question 11 *En déduire un algorithme pour calculer la taille minimale de l'entrepôt nécessaire pour traiter un arbre. Tester votre algorithme sur les arbres :* ***a)*** *$\mathcal{T}_{10}$* ***b)*** *$\mathcal{T}_{100}$* ***c)*** *$\mathcal{T}_{200}$*



Fiche réponse type: Gestion d'un entrepôt

$\widetilde{u_0}$: 30

Question 1

a) 11015

b) 18003

c) 31540

Question 2

a) (3,6)

b) (6,52)

c) (7,96)

Question 3

a) (700,2009)

b) (1137,21017)

c) (1175,38694)

Question 4

a) 700

b) 1600

c) 1753

Question 5

a) 700

b) 1276

c) 1317

Question 6

a) 3

b) 7

c) 7

Question 7

a) 333,2039

b) 1052,307087

c) 5130,65195519

Question 8

a) oui

b) oui

c) non

Question 9

a) 397

b) 1248

c) 6168

Question 10

a) 0

b) 123

c) 0

Question 11

a) 700

b) 1167

c) 1175