Algorithmique III (I41) - Licence d'Informatique

Contrôle terminal (Session 2 - Juin 2023)

Un hôpital gère les patients qui entrent aux urgences à l'aide d'une *file de priorité* `FP`. C'est un enregistrement constitué de deux champs, un tableau indexé `FP.T` de *patients* de taille prédéfinie codant la file d'attente et le nombre `FP.nb` de patients dans cette file (voir figure 1).

Un *patient* est modélisé par un couple $(a, d) \in \mathbb{N}^* \times [\![0, 7]\!]$ où a est le numéro d'arrivée du patient et d le délai acceptable pour être soigné. Si $d = 0$, le patient doit être pris en charge immédiatement et à l'autre extrême, si $d = 7$, il peut attendre une place en consultation. Le couple (a, d) est codé par un enregistrement, donc chaque cellule `FP.T[i]` du tableau contient les deux champs `(FP.T[i]).a` et `(FP.T[i]).d`.

Le tableau `FP.T` a été dimensionné afin que le nombre de patients n'excède jamais sa taille et a été initialisé avec des couples $(0, 0)$. Le nombre de patients `FP.nb` dans la file a été initialisé à 0. Le tableau `FP.T` doit conserver une structure de *tas* afin que le prochain patient à soigner soit toujours au début du tableau, c'est-à-dire `FP.T[1]`.

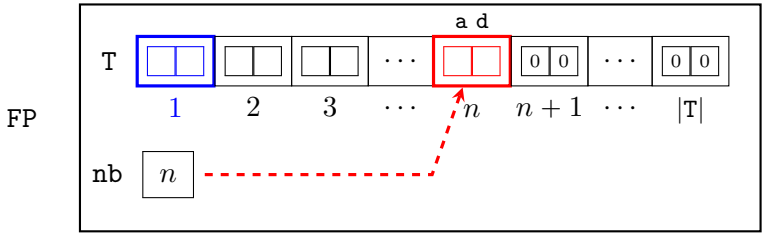

FIGURE 1. Structure de file de priorité.

On dispose d'un algorithme `Tamiser(@T,i,ifin):booléen` qui tamise le nœud d'indice i dans le tableau T avec la valuation $\nu(a, p) := p$ et l'ordre partiel défini sur l'arbre binaire d'ensemble de nœuds X associé à T par

$$\forall x \in X \quad \forall y \in \Gamma(x) \quad \nu(x) \leqslant \nu(y). \tag{1}$$

1) [2.0] La structure d'APO ne sert qu'à la compréhension des mécanismes de gestion des tableaux/tas. Réécrivez la proposition (1) sans faire référence à la valuation ν ou à l'APO, mais directement avec `FP`.

Réponse. Le fils gauche d'un nœud père d'indice i ayant pour indice $2i$ et le fils droit $2i+1$, il suffit de s'assurer que les indices des fils existent pour transposer la propriété et on sait que le plus grand indice de du sous-arbre ayant au moins un fils est $\lfloor \frac{n}{2} \rfloor$:

$$\forall i \in [\![1, \left\lfloor \frac{n}{2} \right\rfloor]\!] \quad \big(T[i] \leqslant T[2i]\big) \wedge \big((2i+1 \leqslant n) \Rightarrow (T[i] \leqslant T[2i+1])\big).$$

où T désigne le tableau `FP.T`. L'implication dans la condition à droite de la conjonction n'était pas exigée, elle formalise le fait que l'on ne fait la comparaison avec le fils droit que s'il existe, l'existence du fils gauche étant assurée par $i \leqslant \lfloor \frac{n}{2} \rfloor$.

2) [2.0] Rappelez la propriété fondamentale de l'algorithme `Tamiser`.

Réponse. Cette propriété est énoncée dans le lemme 8 du cours. C'est le résultat le plus important sur l'algorithme `Tamiser` qui justifie le fonctionnement des algorithmes sur lequel ils reposent :

> **Lemme.** *Soit $A = (X, U)$ un arbre binaire dont les sous-arbres gauche et droit sont des APO. Après application de l'algorithme `Tamiser` à sa racine, cet arbre est un APO.*

Soit `FP` une file de priorité à laquelle on a rajouté un nouveau patient à la position n, le tableau `FP.T` était donc un tas juste avant cet ajout.

3) [2.0] Démontrez qu'en appliquant successivement l'algorithme `Tamiser` aux nœuds d'indices $f^k(n)$ pour k allant de 1 à $\lfloor \log_2(n) \rfloor$, le tableau `FP.T` est de nouveau un tas.

Indication : soit $f : \mathbb{N} \to \mathbb{N}$ l'application définie par $f(n) := \lfloor \frac{n}{2} \rfloor$ et f^k sa k-ème itérée ($f^k(n) := f(f^{k-1}(n))$ si $k > 1$ et $f^1(n) := f(n)$). Considérez le prédicat $P(k)$ défini pour $k \in [\![1, \lfloor \log_2(n) \rfloor]\!]$ par *"l'arbre d'indice $f^k(n)$ est un APO après son tamisage"* et faites une récurrence finie forte.

Réponse. Puisque les deux sous-arbres gauche et droit (si ce dernier existe) de l'arbre dont la racine est d'indice $f^1(n) = \lfloor \frac{n}{2} \rfloor$ sont des APO, le lemme s'applique et $P(1)$ est vrai. Tous les sous-arbres étant des APO avant l'insertion du nouveau patient, seule la racine d'un APO obtenu après application de `Tamiser` et dont la valuation a pu changer, peut violer l'ordre partiel avec son père, son frère reste un APO.

Soit $k < \lfloor \log_2(n) \rfloor$. D'après l'hypothèse de récurrence, tous les sous-arbres d'indices $f^1(n)$ à $f^k(n)$ sont des APO. Comme les frères de ces sous-arbres

étaient APO et le sont restés après application de l'algorithme `Tamiser`, les deux sous-arbres gauche et droit de l'arbre d'indice $f^{k+1}(n) = \lfloor f^k(n)/2 \rfloor$ sont APO et le lemme s'applique ce qui prouve que $P(k+1)$ est vrai.

4) [2.0] Simplifiez `Tamiser` afin qu'il ne fasse d'échange qu'entre la racine de l'arbre d'indice i et son fils de plus petite valuation, si nécessaire. Il doit renvoyer un booléen indiquant s'il y a eu échange ou non.

Réponse. L'algorithme auxiliaire `IdxPatientMin(T,i,j)` renvoie l'indice du patient avec le plus petit délai de prise en charge d, ou l'indice du premier arrivé en cas d'égalité (voir Algo. 1). L'oubli de cette éventualité n'a pas été sanctionné dans l'examen.

```
ALGORITHME IdxPatientMin(T,i,j):entier
DONNEES
 · T: tableau de patients
 · i,j: entiers
DEBUT
 · SI (T[i].d < T[j].d) ALORS
 ·  · RENVOYER i
 · SINON
 ·  · SI (T[j].d < T[i].d) ALORS
 ·  ·  · RENVOYER j
 ·  · SINON
 ·  ·  · SI (T[i].a < T[j].a) ALORS
 ·  ·  ·  · RENVOYER i
 ·  ·  · SINON
 ·  ·  ·  · RENVOYER j
 ·  ·  · FSI
 ·  · FSI
 · FSI
FIN
```

Algo. 1. Calcul de l'indice du patient prioritaire.

Seuls les nœuds qui sont sur le chemin reliant la racine d'indice 1 de l'arbre au sous-arbre d'indice n peuvent (éventuellement) être affectés par l'algorithme Tamiser car l'arbre était un `apo` avant l'insertion du dernier patient. Il suffit donc de passer de père en père si nécessaire. Notons `S-Tamiser` l'algorithme de tamisage simplifié. Il suffit de comparer la valuation du père avec celle(s) de son/ses fils et de faire l'échange des valeurs si nécessaire (cf. Algo. 2).

```
ALGORITHME S-Tamiser(@FP,i,ifin):booléen
DONNEES
 · FP: file de priorité
 · i,ifin: entiers
VARIABLES
 · ifils: entier
DEBUT
 · SI ((2*i < ifin) ALORS      # Il existe un fils droit
 ·  · ifils ← IdxPatientMin(FP.T,2*i,2*i+1)
 · SINON                       # Il n'existe que le fils gauche
 ·  · ifils ← 2*i
 · FSI
 · SI ((ifils < ifin) ET (PatientMin(FP.T,i,ifils) != i)) ALORS
 ·  · Echanger(FP.T,i,ifils)   # Patient prioritaire = fils
 ·  · RENVOYER VRAI
 · SINON
 ·  · RENVOYER FAUX
 · FSI
FIN
```

Algo. 2. Version simplifiée de l'algorithme de tamisage.

5) [1.0] Calculez la complexité de votre algorithme.

Réponse. L'algorithme comporte un nombre constant d'instructions, il n'y a pas de boucles, sa complexité en $\Theta(1)$ si on considère que la structure de file de priorité est passée "par adresse", comme c'est le cas ici.

6) [2.5] En déduire un algorithme `AjouterPatient(@FP,a,d)` qui ajoute un patient dans la file de priorité.

Réponse. Il suffit de rajouter le nouveau patient en fin de tableau et de restaurer la structure de tas en appliquant l'algorithme de tamisage simplifié aux sous-arbres d'indices $f^k(n)$ en partant de $k = 1$ tant qu'il y a eu échange. (cf. Algo. 3).

```
ALGORITHME AjouterPatient(@FP,a,d)
DONNEES
 · FP: file de priorité
 · a,d: entiers
VARIABLES
 · i: entier
DEBUT
 · FP.nb ← FP.nb + 1      # Ajout du nouveau patient
 · (FP.T[FP.nb]).a ← a    # patient à la position
 · (FP.T[FP.nb]).d ← d    # n + 1
 · i ← FP.nb DIV 2        # Indice du père
 · TQ ((i > 0) ET S-Tamiser(FP,i,FP.nb)) FAIRE
 ·  · i ← i DIV 2         # Restauration du TAS
 · FTQ
FIN
```

Algo. 3. Ajout d'un patient et restauration du tas.

7) [1.0] Calculez la complexité dans le meilleur des cas et dans le pire des cas de votre algorithme.

Réponse. Dans le meilleur des cas, le délai d du patient qui vient d'être rajouté dans la file de priorité est supérieur à celui de son père, il n'y a pas d'échange et on n'entre pas dans la boucle `TQ`, la complexité est donc constante en $\Theta(1)$. Dans le pire des cas, la propriété APO est systématiquement violée, ce qui impose de continuer la boucle `TQ` jusqu'à atteindre le nœud racine d'indice 1. On entre donc dans la boucle autant de fois que la profondeur de l'arbre c'est-à-dire $\lfloor \log_2(n) \rfloor$ et la complexité est donc $\Theta(\log(n)$ si n désigne le nombre de patients dans la file de priorité.

8) [2.5] Écrivez un algorithme `RetirerPatient(@FP):tpatient` qui retire le patient (a, d) de plus petit délai de la file de priorité `FP` et le renvoie. La file de priorité doit rester un tas après cette opération.

Réponse. Le patient avec le délai de prise en charge le plus court est au sommet de l'APO, c'est-à-dire en position 1 dans le tas `FP.T`. Il suffit donc de le sauvegarder et de le remplacer par celui en dernière position, puis de tamiser la racine pour retrouver la propriété `apo` potentiellement violée par cet échange.

```
ALGORITHME RetirerPatient(@FP):tpatient
DONNEES
 · FP: file de priorité
 · p: tpatient
VARIABLES
 · i: entier
DEBUT
 · p ← FP.T[1]             # Sauvegarde du patient prioritaire
 · FP.T[1] ← FP.T[FP.nb]   # Copie du dernier patient
 · (FP.T[FP.nb]).a ← 0     # Initialisation de la
 · (FP.T[FP.nb]).d ← 0     # cellule libérée
 · FP.nb ← FP.nb - 1       # Décrément du nombre de patients
 · Tamiser(FP,1,FP.nb)     # Restauration de la propriété APO
 · RENVOYER(p)
FIN
```

Algo. 4. Retrait du patient prioritaire et restauration du tas.

NB. Il faut remarquer, mais ce n'était pas exigé, qu'il aurait également fallu modifier l'algorithme `Tamiser` afin qu'en cas d'égalité des délais entre un père et son fils de plus petit délai, l'échange se fasse tout de même si l'ordre d'arrivée du fils est antérieur à celui du père.

9) [2.5] Montrez que votre algorithme s'arrête et donnez les éléments de preuve de correction partielle de votre algorithme.

Réponse. Toutes les instructions, les 4 affectations et le renvoi du patient prioritaire, sont de complexité en temps constante, exceptée l'algorithme `Tamiser` qui contient une boucle. L'arrêt de l'algorithme `RetirerPatient` est donc assuré par celui de l'algorithme `Tamiser`. Le patient prioritaire étant en tête du tableau par construction, la variable `p` contient bien l'information requise et, puisque l'algorithme de tamisage s'arrête, sera bien renvoyée. Par hypothèse, la validité de l'algorithme `Tamiser` est assurée, le tableau retrouvera donc bien son statut de tas (voir Algo. 4).

10) [2.5] Calculez la complexité dans le meilleur des cas et dans le pire des cas de votre algorithme.

Réponse. Dans le pire des cas, le délai de prise en charge du dernier patient p est strictement supérieur à tous ceux des patients sur le chemin menant de la racine à l'avant-dernier patient, auquel cas l'algorithme de tamisage fera redescendre p jusqu'en bas de l'arbre soit en $\Theta(\log(n))$.

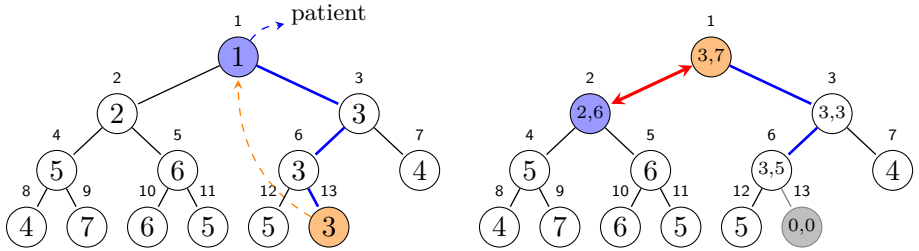

Figure 2. Illustration du meilleur des cas lors du retrait d'un patient.

L'évaluation du meilleur des cas était la seule question difficile. La figure permet d'illustrer une telle situation, les valeurs de certains nœud sont détaillées en précisant l'ordre d'arrivée. Par construction de la file de priorité, le patient en position n a le délai de prise en charge le plus long de tous les patients sur le chemin qui le relie à la racine de l'arbre (patients d'indices 6,3 et 1), ou est arrivé après eux en cas d'égalité comme c'est le cas dans cette figure. Une fois le patient au sommet de l'arbre retiré et remplacé par le patient en position n, il est nécessaire que la valuation de ce dernier soit inférieure à celles de ses fils pour qu'il reste à la racine, ce qui est impossible. En effet, sa valuation est nécessairement supérieure (ou égale, mais dans ce cas, il est arrivé après les autres) à celles des patients sur le chemin qui le reliait à la racine avant sa remontée au sommet. Autrement dit, il faut que son autre fils aient une valuation *inférieure* pour que l'algorithme `Tamiser` ne le fasse pas redescendre le long de ce chemin. En revanche, il est possible qu'un seul échange avec le fils de valuation inférieure suffise à rétablir la propriété APO, comme l'illustre la figure. Dans ce cas la complexité est évidemment $\Theta(1)$.