

TD N° 7

Autour de la structure de tas binaire

1 Structure de tas binaire

Rappel : Un tableau t est structuré en tas binaire **décroissant** entre les indices g et d s'il vérifie la propriété :

$$\forall i \in [g .. d/2], t[i] \geq t[2i] \text{ et } t[i] \geq t[2i+1]$$

Les valeurs $t[2i]$ et $t[2i+1]$ correspondent aux successeurs de $t[i]$ dans l'arbre binaire représentant le tas. Le tas binaire permet une modélisation des files d'attente à priorité et cette modélisation est intéressante car elle est associée à des algorithmes performants d'insertion et de retrait. Cette structure est également à la base d'un algorithme de tri sur place en $n \log_2(n)$.

1.1 Insertion dans une file d'attente à priorité

Soit f une file d'attente à priorité qui est vide.

- Faites la trace de l'évolution de cette file, lorsqu'on y insère successivement des composants de priorités respectives 6, 8, 3, 4, 2, 7 et 9. On utilise pour cela la procédure `Ajouter` vue en cours (page 3 du mémento 7).
- On donne les types suivants :

```
types
  T_arbre= pointeur sur T_noeud
  T_noeud= article
            contenu:composant
            gauche,droit:T_arbre
  finarticle
```

Écrivez la fonction `Construire_Arbre` qui construit l'arbre binaire associé au tas. Dessinez l'arbre associé au tas construit dans la question précédente. Quel est le résultat obtenu lorsque l'on effectue un parcours en largeur sur l'arbre ?

1.2 Suppression de l'élément de tête dans une file à priorité

La procédure `Enlever` vue en cours (page 4 du mémento) utilise en fait la procédure *primitive* `Glisser` vue un peu plus loin à l'occasion du tri en tas.

- Écrivez la fonction `Aîné` (utilisée dans cet algorithme) qui détermine celui des fils de i qui a la plus forte priorité. Où placeriez-vous sa déclaration ?
- Récrivez la procédure `Enlever` du cours en utilisant un appel à `Glisser`.
- Faites la trace des suppressions successives des deux premiers éléments dans la file construite à l'exercice 1.1.

2 Tri en tas

D'une façon très générale, le fait d'avoir une structure de stockage permettant l'ajout d'éléments et le retrait du plus grand d'entre eux peut conduire à une technique de tri. Examinons d'abord celle-ci dans le cas le plus simple où il s'agit de trier une liste. On suppose avoir les objets suivants :

```

type
  T_Liste = ?? {type d'une liste d'Entrée}

fonction estVide (L : T_Liste) : booléen
  {Antécédent : L initialisée}
  {Rôle : renvoie vrai si la liste L est vide, faux sinon}

procédure Prendre (donnée-résultat L : T_Liste
                   résultat élém : T_Entrée)
  {Antécédent : L n'est pas vide}
  {Rôle : élém contient le premier élément de la liste L-
   L+ est L- privée de son premier élément}

procédure Mettre (donnée-résultat L : T_Liste
                   donnée élém : T_Entrée)
  {Antécédent : L est initialisée}
  {Rôle : L+ = élém • L-}

fonction Longueur (L : T_Liste) : entier
  {Antécédent : L initialisée}
  {Rôle : renvoie la longueur de la liste L}

```

- f. Écrivez une procédure de tri qui trie une liste de façon croissante en utilisant une file à priorité.
- g. Quelle est la complexité en temps du tri obtenu par cette voie ?
- h. Comment peut-on adapter cette technique pour faire le tri *sur place* d'un tableau ?

3 Files à priorité et simulation

L'informatique est souvent utilisée pour *simuler* des phénomènes complexes qu'il serait difficile, voire impossible d'expérimenter dans la réalité (circulation automobile, évolution de populations, invasion des États-Unis par les troupes irakiennes, etc.). Une façon simple de faire est de considérer que le phénomène est constitué d'événements discrets et datés et que, à chaque fois qu'un événement se produit (quand on arrive à sa date), d'autres événements deviennent prévisibles dans le futur : quand une automobile franchit un carrefour, un certain temps plus tard elle débouchera dans une nouvelle rue ; quand un individu naît, dans un avenir aléatoire, il aura des enfants et il mourra.

Une file à priorité dont les clés sont des dates (tas *croissant*) peut être utilisée pour gérer la simulation. On suppose les déclarations suivantes :

```

type
  NatureÉvénement = (...) {type énuméré}
  Événement = article
    date : naturel
    nature : NatureÉvénement
    contenu : ... {autres données liées à l'événement}
  finarticle
  Agenda = article
    taille : 0 .. Taille Max
    t : tableau [1 .. Taille Max] de Événement

```

finarticle**variables**

agenda : Agenda
situationInitiale : **ensemble de** Événements
maintenant : 0 .. dateMax
e, conséquence : Événement

procédure Initialiser (**donnée-résultat** simulation : Agenda)
{Rôle : met la simulation à vide}

fonction Conséquences (**donnée** e : Événement) : **ensemble de** Événement
{Rôle : renvoie l'ensemble des événements qui deviennent
prévisibles quand l'événement e se produit}

procédure Traiter (**donnée** e : Événement)
{Rôle : met à jour les compteurs qui serviront à faire des statistiques}

- i.** Écrivez l'algorithme qui réalise la simulation à partir de la situation initiale et jusqu'à la date dateMax.