TP Algorithmique Avancée : Tri

Serigne A. Gueye

CERI - Licence 2 Informatique - Année Académique 2015-2016

8 novembre 2015

1 Tri Fusion, Quicksort

On souhaite coder une fonction “sort “ permettant de trier des entiers. La fonction aura le prototype suivant :

$$sort(int * \ begin, int * \ end, char * \ type, char \ sens)$$

où *begin* est un pointeur sur le début du tableau d'entiers à trier, *end* un pointeur sur la fin, *type* le type de tri choisi (”Fusion” ou ”Quicksort”), et *sens* permettant de trier en order croissant ('c') ou décroissant ('d').

Implémenter cette fonction. Vous coderez pour cela deux autres fonction permettant de réaliser des tris fusion et quicksort.

2 Tri par Tas (HeapSort)

On souhaite coder une classe **Tas** aux caractéristiques suivantes :

```
class     Tas
{
          int T[ ] ;
          int n ;

public :
          Tas(int max) ;
          ~ Tas() ;
          void ajout(int x) ;
          int suppression() ;
          boolean estvide() ;
          int taille() ;
}
```

où :

- T est le tableau représentant le tas,
- n est le nombre d'éléments du tas,
- Tas() est le constructeur permettant de construire initialement un tableau tas T en le surdimensionnant,
- ~Tas() est le destructeur,
- *ajout* est une méthode permettant de rajouter la valeur x au tas,
- *suppression* est une méthode permettant de retourner et de supprimer l'élément à la racine du tas,
- *estvide*() est une méthode permettant de savoir si le tas est vide, ou non.
- *taille*() est une méthode retournant le nombre d'éléments du Tas.

1/ Coder en C++ cette classe.

2/ Soit R un tableau à trier en ordre décroissant par une variante (donnée ci-dessous) de la méthode du tri par tas. O désigne une instance de la classe Tas et m la taille de R.

```
for i allant de 1 à m do
    O.ajouter(R[i]) ;
end
while (!O.estVide()) do
    tmp = O.taille() ;
    min = O.suppression() ;
    R[tmp] = min ;
end
```

Algorithme 1: Tri par Tas (Version 1)

Ecrire une nouvelle méthode $TriTas(int\ m,\ int\ R[])$, de la classe Tas, permettant de trier l'argument R.