IFT 1227 – Architecture des ordinateurs

Devoir 3

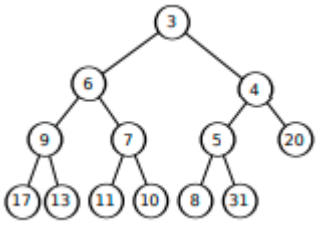
- Remise : le 2.04.2023
- Il y aura une pénalité de **10%** par jour de retard.

Programmation en assembleur MIPS

Un monceau (heap)est une structure de données linéaire et partiellement ordonnée qui permet d'accéder efficacement au plus petit (ou au plus grand) élément d'une collection d'éléments. La structure linéaire d'un monceau, qui ne nécessite aucun pointeur pour lier les éléments entre eux, permet une représentation compacte en mémoire. La principale application du monceau est l'implémentation de files prioritaires. Il est possible d'illustrer graphiquement un monceau (heap) à l'aide d'un arbre binaire complet équivalent. Un arbre complet est un arbre où seul le dernier niveau peut être incomplet, et où les éléments du dernier niveau sont à gauche. Un arbre binaire équivalent à un monceau est structuré de la façon suivante :

- L'élément le plus petit (ou plus grand) se trouve dans le nœud racine ;
- Les enfants d'un nœud sont plus grands (ou plus petits) que ce dernier ;
- Il n'y a pas de contraintes d'ordonnancement entre les enfants de gauche et de droite.

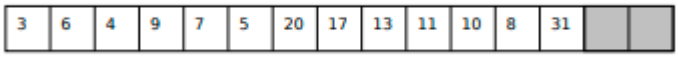
La Figure plus bas illustre un monceau sous forme d'arbre binaire.



Un monceau est représenté à l'aide d'un tableau linéaire où :

- La racine se trouve à l'indice zéro (0) ;
- Les enfants d'un nœud à l'indice i se trouvent aux indices 2i+1 et 2i+2 ;
- Le parent d'un nœud à l'indice i > 0 se trouve à l'indice fonction plancher de l'expression (i−1)/2.

La Figure suivante montre la représentation linéaire d'un monceau équivalent à l'arbre binaire de la Figure plus haut.



Dans ce devoir vous allez implémenter un tri «Heap sort » qui permet de trier un tableau d'éléments en ordre croissant.

1) Écrire un code MIPS qui initialise un tableau de **n** valeurs. Ces valeurs doivent être placées à partir de l'adresse 0x10040000. D'abord, vous devez chercher le paramètre **n** saisi par un utilisateur qui représente le nombre d'éléments du tableau à traiter. Ensuite, cherchez les éléments du tableau et stockez-les à partir de l'adresse spécifiée.

   Utilisez l'appel système correspondant pour pouvoir saisir toutes les valeurs nécessaires.

   Concevoir les fonctions **init**, **swap**, **getLeftChildIndex** et **getRightChildIndex** ainsi que la fonction **main** qui fera les appels à ces 4 fonctions en les testant.
   - La fonction **init** prend aucun paramètre, initialise un tableau de n éléments à partir de l'adresse 0x10040000 et retourne 2 valeurs : adresse du tableau créé et sa taille.
   - La fonction **swap** échange le contenu de deux éléments du tableau avec les indices **i** et **j** respectivement. Les fonctions **getLeftChildIndex** et **getRightChildIndex** retournent les indices des enfants gauche et droite dans la structure de données MONCEAU.

```
void swap(int i, int j)
{
      int temp = a[i];
      a[i] = a[j];
      a[j] = temp;
}
int getLeftChildIndex(int index)
{
    return 2 * index + 1;
}
int getRightChildIndex(int index)
{
      return 2 * index + 2;
}
```

2. Rajouter les implémentations de deux fonctions suivantes au code du numéro 1.
   void fixHeap(int rootIndex, int lastIndex);
   void heapSort(int [], int taille);
   qui compléteront l'implémentation de l'algorithme **HeapSort**.

   Écrire la fonction **main** qui fait l'appel aux **init**, **fixHeap** et **HeapSort** pour trier le tableau créé.

Le code Java des fonctions demandées ainsi que les notes de cours Programmation II, qui traitent ce sujet sont fournis sur le site du cours.

Voici l'ébauche du programme #2 :

```
#segment de la mémoire contenant les données globales
# chaînes ASCII peuvent être déclarées ici
.data
…
#segment de la mémoire contenant le code
.text
main:
     …
     li   $v0,10    #terminer le programme
     syscall
#fonctions
init:
     …
swap:
     …
getLeftChildIndex :
     …
getRightChildIndex:
     …
fixHeap:
     …
heapSort :
     …
```

Directives

Dans toutes les questions, il faut obligatoirement respecter toutes les conventions MIPS et utiliser les bonnes pratiques de la programmation.

L'information concernant les appels système se trouve dans le simulateur Mars, bouton Help => onglet MIPS => onglet Syscalls. Voici l'extrait de l'information concernant des appels système :

Saisir une valeur à l'entrée standard et la faire stockée dans le registre `$t0` :

```
li   $v0, 5    # charger le numéro de service
syscall        # faire appel de ce service
# récupérer le résultat de $v0 dans $t0
add  $t0, $v0, $0
```

Imprimer une valeur stockée dans le registre `$t0` :

```
li  $v0, 1   # service 1 imprime un entier
# charger la valeur à imprimer dans le registre $a0.
add $a0, $t0, $zero # Notre valeur se trouve dans $t0
syscall             # faire appel du service
```

Imprimer une chaîne se trouvant à l'adresse msg

```
la   $a0, msg
li   $v0, 4    # Print string (Code 4 de syscall)
syscall        # faire appel du service
```

Remise

Remettre le 2 fichiers source: **`q1.asm`** et **`heapsortTab.asm`**

Évaluation

Bon fonctionnement du programme :		
	Q1	30
	Q2	30
Programme adéquat (algorithme, déroulement du programme, conventions)		20
Commentaires		5
Contient les descriptifs: but, date, auteur(s), adresse(s) de courriel, code(s) permanent(s)		5
Respect des directives		10
Votre note:		/100