

```
tasser_ascendant(T, i) :      tasser_descendant(T, i):  
    Si  $i \neq 0$  et  $T[i] > T[i//2]$ :    $k = i$   
         $T[i], T[i//2] =$                 si  $2i+1 < T.taille$  et  $T[k] <$   
 $T[i//2], T[i]$                          $T[2i+1]:$   
            tasser_ascendant( T,  $i//2$ )    $k = 2i+1$   
                                            si  $2i+2 < T.taille$  et  $T[k] <$ 
```

```

T[2i+1]:
    k = 2i+2
    si k != i:
        T[i], T[k] = T[k], T[i]
        tasser_descendant(T, k)

```

Implémentation 13 Une file de priorité peut être implémenté en utilisant les opérations de tamisage sur les tas. On peut alors implémenter les fontions inserer et extraire_max.

```

inserer(T, e):
    T.taille = T.taille + 1
    T[T.taille - 1] = e
    tasser_ascendant(T, T.taille - 1)

extraire_max(T):
    m = T[0]
    T[0] = T[T.taille - 1]
    T.taille = T.taille - 1
    tasser_descendant(T, 0)
    renvoyer m

```

Si on veut des strutures de taille variables il faut utiliser d'autres méthodes d'implémentation.

B. Implémentation à l'aide de tableaux dynamiques

Implémentation 14 Une pile peut être implémenté à l'aide d'un tableau dynamique et d'un indice de fin de pile.

Implémentation 15 Une file peut être implémenté à l'aide d'un tableau dynamique, un indice de début et de fin de file. On veillera, lors du changement de la taille, à corriger l'éventuel cycle dans la structure.

Implémentation 16 Une file de priorité peut être implémenté à l'aide d'un tableau dynamique, soit en conservant le tableau trié à l'insertion, soit en sélectionnant l'élément de plus grande priorité au défilement.

C. Implémentation à l'aide de listes chaînées

Implémentation 17 Une pile peut être implémenté à l'aide d'une liste chaînée. Le sommet de la pile correspond à la tête de la liste.

Remarque 18 Pour une implémentation immutable, cette pile correspond au type list d'Ocaml.

Implémentation 19 Une file peut être implémentée à l'aide d'une liste chaînée cyclique, et d'un pointeur vers le dernier élément

de la liste. Les éléments sont défilés depuis la tête de la liste, et enfilé à la fin de celle-ci.

Complexité 20 Voici l'ensemble des complexités des implémentations des piles, files et files de priorité.

	Pile		File		File de priorité	
	empiler	défiler	enfiler	défiler	enfiler	défiler max
Tableau	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(\log(n))$	$\mathcal{O}(\log(n))$
Tableau Dynamique	$\mathcal{O}(1)^*$	$\mathcal{O}(1)^*$	$\mathcal{O}(1)^*$	$\mathcal{O}(1)^*$	$\mathcal{O}(\log(n))^*$	$\mathcal{O}(\log(n))^*$
Liste Chaînée	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$

D. Implémentations croisées

Implémentation 21 Deux files permettent d'implémenter une pile. Cependant l'opération défiler est de complexité *amortie* constante.

Implémentation 22 Il est possible, mais inintéressant, d'implémenter une pile avec deux files. La complexité de l'opération défiler est alors linéaire.

III. Application [COR3] [TOR]

A. Algorithmique

Application 23 Un parcours de graphe s'implémentent en utilisant une pile, une file ou une file de priorité.

Exemple 24 Le parcours en largeur d'un graphe utilise une file.

Exemple 25 Le parcours en profondeur d'un graphe utilise une pile.

Application 26 Les algorithmes gloutons , c'est à dire les algorithmes qui à chaque étape choisissent la solution localement optimale, s'appuient souvent sur une file de priorité pour choisir l'option la plus avantageuse.

Exemple 27 **Problème du sac à dos** Pour remplir un sac à dos avec des objets de valeur, il est possible de choisir successivement les objets les plus valeureux pouvant être rajoutés. C'est un algorithme glouton qui ne donne pas toujours la solution optimale.

Application 28 Les algorithmes de plus court chemin comme Dijkstra et A* utilisent des piles.

Application 29 Les algorithmes de prim et Kruskal pour le calcul d'un arbre couvrant minimal utilisent une pile.

Exemple 30 L'algorithme de Graham calcule l'enveloppe convexe d'un ensemble de points non tous alignés. **TODO** rapport avec pile file ?

Application 31

Application 32 Le tri par file de priorité est une méthode de tri en assimilant les priorités et les valeurs des éléments. Selon l'implémentation de la file de priorité, la complexité de ce tri peut être optimale.

Définition 33 La notation polonaise inverse permet d'écrire des expressions arithmétiques sans parenthèses et sans ambiguïté sur l'ordre des opérations.

Application 34 L'évaluation d'une formule en notation polonaise inversée se faire à l'aide d'une pile où les nombres et opérations sont dépilés et le résultat empilé.

Application 35 L'algorithme de la gare de triage permet, à l'aide d'une pile, de transformer des expressions arithmétiques infixes parenthésées en expressions en notation polonaise inverse.

B. Application au fonctionnement de l'ordinateur

Application 36 La pile d'appel est une partie de l'adressage virtuel d'un processus. Elle est utilisée pour stocker les variables locales et le cadre de pile à chaque appel de fonction.

Remarque 37 Un parcours de graphe par appel récursif utilise implicitement la pile d'appel comme pile comme que le parcours en largeur.

Application 38 L'historique des modifications dans un éditeur de texte est aussi un exemple de l'utilisation d'une file.

Application 39 Des politiques d'ordonnancement sont utilisées faisant intervenir des piles et files :

- ▶ PEPS consiste en le maintien d'une file de processus.
- ▶ Shortest Job First consiste en le maintien d'une file de priorité de processus en prenant la durée d'exécution comme priorité.
- ▶ Round Robin est un algorithme d'ordonnancement préemptif qui, comme PEPS, maintient une file de processus, mais qui suspend un processus s'il prend trop de temps et le met au bout de la file.
- ▶ MultiLevel Feedback Queue fonctionne de manière similaire à Round Robin mais avec plusieurs files qui ont toutes des « priorités » différentes. Lorsqu'un processus est interrompu, il est enfilé dans une file de « priorité » inférieure. Le prochain processus à charger est défilé d'abord dans les files les plus prioritaires. Régulièrement, on remet tous les processus dans la file la plus prioritaire.

Application 40 Différentes politiques de remplacement de cache existent :

- ▶ PEPS consiste en le maintien d'une file de processus.
- ▶ Shortest Job First consiste en le maintien d'une file de priorité de processus, en prenant la durée d'exécution comme priorité.
- ▶ LRU (Least Recently Used), on retire l'élément accédé il y a le plus longtemps. Pour ça, on maintient une file de priorité min, avec comme priorité le temps d'accès.
- ▶ L'implémentation du LRU étant trop coûteuse pour des applications bas niveau, on peut l'approximer en ajoutant à une file un « bit d'accès », qu'on met à 1 lorsqu'on accède à l'élément dans le cache. Quand on défile l'élément à retirer, si le bit d'accès est à 1, on le remet dans la file avec un bit d'accès à 0, et on reessaie en défilant l'élément d'après.

Implémentations et application des piles, files et files de priorité	
I. Définitions [TOR] - [NSIT 7]	
1	Def Une pile
2	Def Une file
3	Rem DEPS et PEPS
4	Def Une file de priorité
5	Rem D'autres opérations sur les files de priorités
6	Rem
13	Implem Une file de priorité
B. Implémentation à l'aide de tableaux dynamiques	
14	Implem Une pile
15	Implem Une file
16	Implem Une file de priorité
C. Implémentation à l'aide de listes chaînées	
17	Implem Une pile
18	Rem
19	Implem Une file
27	Ex Problème du sac à dos
28	App Les algorithmes de plus court chemin
29	App Les algorithmes de prim et Kruskal
30	Ex L'algorithme de Graham
31	App
32	App Le tri par file de priorité
33	Def La notation polonaire inverse
34	App L'évaluation d'une formule en notation polonaise inversée
35	App L'algorithme de la gare de triage
B. Application au fonctionnement de l'ordinateur	
36	App La pile d'appel
37	Rem Parcours Graphe Appel récursif

II. Implémentations	
A. Implémentation à l'aide des tableaux	
7	Implem Une pile
8	Implem Une file
9	Implem Une file de priorité
10	Def Un tas binaire max
11	Rem La notation de Sosa Stradonitz
12	Def Le tamisage
20	Complex
D. Implémentations croisées	
21	Implem Deux files
22	Implem Une pile avec deux files
III. Application [COR3] - [TOR]	
A. Algorithmique	
23	App Un parcours de graphe
24	Ex Le parcours en largeur
25	Ex Le parcours en profondeur
26	App Les algorithmes gloutons
38	App L'historique des modifications
39	App Des politiques d'ordonnancement
40	App Différentes politiques de remplacement de cache

Remarque

Potentiellement moins séparer application et implémentation.

1. Ce n'est pas pédagogique,
 2. Celà crée une énumération ennuyant à l'oral
- Potentiellement s'inspirer plus du plan de Pablo et Alexandre.

Bibliographie

[TOR] T. Balabonski & S. Conchon & J. Filliâtre & K. Nguyen & L. Sartre, *MP2I MPI, Informatique Cours et exercices corrigés*.

[NSIT] T. Balabonski & S. Conchon & J. Filliâtre & K. Nguyen, *Numériques et Sciences Informatiques Terminale*.

[COR3] T. H. Cormen, *Introduction à l'algorithmique (3rd édition)*.