SD

1. Pile :

* Comme une pile d’assiette : si on veut ajouter ou retirer une assiette on le fait au sommet de la pile.

🡪pr ajouter : il faut utiliser la taille logique(nbr d’éléments présents dans la table) : table[taille\_logique]= nvl élément qu’on veut ajouter. (Pas oublier d’incrémenter la taille)

🡪pr retirer il suffit de diminuer la taille logique de 1.

1. Vecteur :

* On ajoute un élément à un indice(rang) précis, si tt les cases de la table sont pleines, on fait un agrandissement (taille de base \*2).
* Pr agrandir : on créer une table temporaire de taille (table.lenght\*2) ensuite on copie (à l’aide d’une boucle for) tt les élément de table dans la table temporaire

taille++;  
if (taille >= *CAPACITE\_MIN*){  
 char[] temp = new char[table.length\*2];  
 for (int i = 0; i <table.length ; i++) {  
 temp[i]= table[i];  
 }  
 table= temp;  
}

* Ensuite pour insérer un élément à un rang précis, on fait le décalage, puis on insère (car l’ordre doit être conservé).

for (int i = taille-1; i > rang; i--) {  
 table[i]= table[i-1];  
}  
table[rang]= caractere;

* Pour supprimer, pareil, on décale mais de l’autre sens.

for (int i = rang ; i < taille-1; i++) {  
 table[i]=table[i+1];  
}  
taille--;

1. File:

* Comme une vraie file, on rajoute à l’arrière et on retire à l’avant.
* Defile : Pr les suppressions, au lieu de retirer et faire tt les décalages, on utilise un indiceTete qui nous indique où la file commence dans la table.

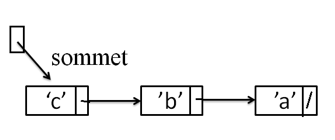
if(indiceTete==table.length-1)  
 indiceTete=0;  
else  
 indiceTete++;

* Enfile : si il n’ya plus assez de place, on aggrandi la table comme on a fait avec les vecteurs, sauf que là, on a besoin de deux boucles for, un qui va de indiceTete-->fin et une qui va de 0-->indiceTete.

table[(taille+indiceTete)% table.length] = element ???????????? ask Barbara why

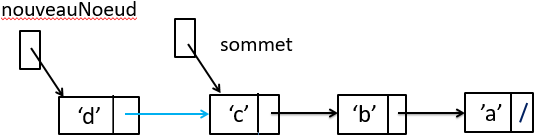
public void enfile(E element){  
 if(taille<table.length){  
 table[(taille+indiceTete)% table.length] = element;  
 }else {  
 if(taille>=table.length) {  
 Object[] temp = new Object[table.length\*2];  
 int j =0;  
 for (int i = indiceTete; i < table.length; i++) {  
 temp[j]=table[i];  
 j++;  
 }  
 for (int i = 0; i < indiceTete; i++) {  
 temp[j]=table[i];  
 j++;  
 }  
 table=temp;  
 indiceTete=0;  
 }  
 table[taille-indiceTete] = element;  
 }  
 taille++;  
}

1. Pile et file via pointeurs:

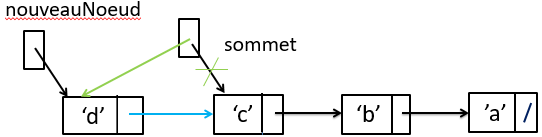


* On utilise une structure chainée. tt les cases sont liées par des noeuds. Sans lien tt est perdu
* Pr ajouter qqch , **nouveauNoeud.suivant = sommet;** puis **sommet = nouveauNoeud ;**

Donc d’abord,



Ensuite,



* pour retirer, (faut pas oublier que dans pile et file les suppressions se font au début, à la tête) il faut juste tete= tete.suivant;
* dans certains ex, pour faciliter la manipulation de la table, on peut avoir une queue qui pointe vers la dernier élément de la table (comme la tête point vers le premier), et on peut aussi avoir des sentinelles (nœuds nulls) qui facilitent les ajouts et suppressions en début et fin de table.
* Il ne faut surtt pas oublier que tt est lié, si on brise un lien tt ce qui était après est perdu.

1. Liste via pointeurs :

* Comme avant, on utilise une structure chainée, donc ici on ajoute un baladeur
* Trop dur à expliquer …
* Go voir théorie
* Pour parcourir :

**Noeud baladeur=tete;**

**while(baladeur!=null) {**

**baladeur=baladeur.suivant; }**

1. Liste récursive ( semaine 5)