1. Implémenter une file avec priorités avec une liste positionnelle :
   - non triée
   - triée
2. Rappeler les méthodes pour trier utilisant une file avec priorités
3. Tri “en place”

File d'attente avec priorités implémentée avec une liste positionnelle

non triée

Performance :

insert prend un temps dans ***O(1)*** puisque nous pouvons insérer une entrée au début ou à la fin de la liste en $O(1)$

removeMin et **min** prennent un temps dans ***O(n)*** puisque nous devons parcourir la liste pour trouver l'entrée avec la plus petite clé

triée

Performance :

insert prend un temps dans ***O(n)*** puisque nous devons trouver où insérer l'entrée en fonction de sa clé avec une recherche séquentielle, suivie d'une insertion en $O(1)$

PS. si on utilise une liste basée sur un tableau, on peut chercher en $O(\log n)$ avec une recherche dichotomique mais l'insertion dans un tableau est en $O(n)$

removeMin et **min** prennent un temps dans ***O(1)*** puisque la plus petite clé est au début de la liste (ou si on l'inverse à la fin) et la retirer se fait en $O(1)$ (dans les 2 cas)

Allons voir le code de

`UnsortedPriorityQueue.java`

`SortedPriorityQueue.java`

Trier avec une file d'attente avec priorités

Nous pouvons utiliser une file d'attente avec priorités pour **trier** un ensemble d'éléments

On a qu'à insérer les éléments un par un avec une série d'opérations `insert`

Ensuite, on supprime les éléments dans l'ordre avec une série d'opérations `removeMin`

Le temps d'exécution de cette méthode de tri dépend de l'implémentation de la file d'attente avec priorités !

Algorithme *Trier-FileDAttenteAvecPriorites(S, C)*
  Input séquence *S*, comparateur *C*
  Output séquence *S* triée en ordre croissant selon *C*
  P = File d'attente avec priorités et comparateur C
  Tantque non *S.est_vide*()
    e = *S.retirerSuivant*()
    P.ajouter(e, e)
  Tantque non *P.est_vide*()
    e = *P.retirerMin*().*key*()
    S.append(e)

Tri par sélection

Le **tri par sélection** est une variation du tri par file d'attente avec priorités, où la file d'attente est implémentée avec une **séquence non triée**

Son temps de fonctionnement en 2 étapes :

1. L'insertion des éléments dans la file avec n opérations `insert` prend un temps dans $O(n)$
2. La suppression des éléments de la file se fait avec n opérations `removeMin` qui prend un temps

proportionnel à $n + (n-1) + \ldots + 1 = \sum_{i=1}^{n} i = n(n+1)/2$

Donc, le tri par sélection s'exécute en temps dans $O(n^2)$

Tri par sélection

		Séquence S	File d'attente prioritaire P (non triée)
	Entrée:	(7,4,8,2,5,3,9)	()
	Phase 1		
	(a)	(4,8,2,5,3,9)	(7)
O(n)	(b)	(8,2,5,3,9)	(7,4)
	..	.. ..	
	(g)	()	(7,4,8,2,5,3,9)
	Phase 2		
	(a)	(2)	(7,4,8,5,3,9)
	(b)	(2,3)	(7,4,8,5,9)
	(c)	(2,3,4)	(7,8,5,9)
O(n^2)	(d)	(2,3,4,5)	(7,8,9)
	(e)	(2,3,4,5,7)	(8,9)
	(f)	(2,3,4,5,7,8)	(9)
	(g)	(2,3,4,5,7,8,9)	()

Tri par insertion

Le **tri par insertion** est une variation du tri par file d'attente avec priorités, où la file est implémentée avec une **séquence triée**

Son temps de fonctionnement en 2 étapes :

1. L'insertion des n éléments dans la file d'attente avec priorités avec n opérations `insert` prend un temps proportionnel à $1 + 2 + ... + n$, donc $O(n^2)$
2. La suppression des éléments se fait avec n opérations `removeMin` prend un temps dans $O(n)$

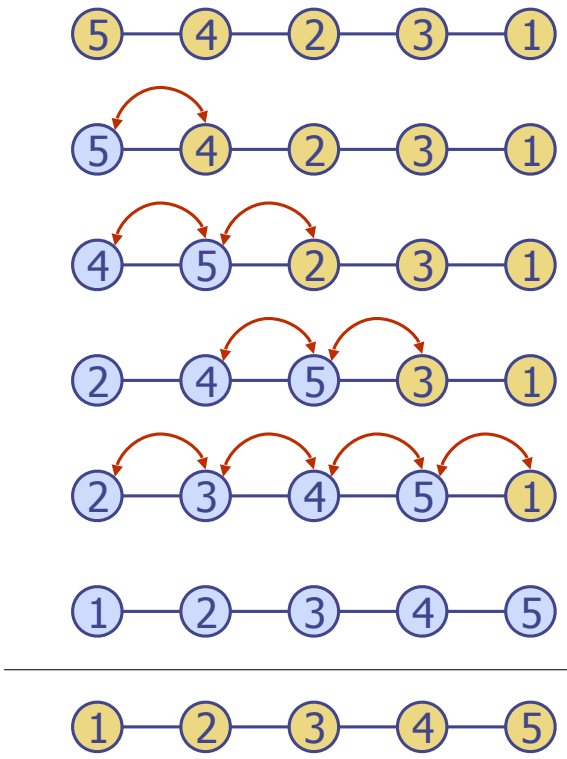
Donc, le tri par insertion s'exécute en temps dans $O(n^2)$

Exemple de tri par insertion

		Séquence S	File d'attente prioritaire P (triée)
	Input:	(7,4,8,2,5,3,9)	()
	Phase 1		
	(a)	(4,8,2,5,3,9)	(7)
	(b)	(8,2,5,3,9)	(4,7)
$O(n^2)$	(c)	(2,5,3,9)	(4,7,8)
	(d)	(5,3,9)	(2,4,7,8)
	(e)	(3,9)	(2,4,5,7,8)
	(f)	(9)	(2,3,4,5,7,8)
	(g)	()	(2,3,4,5,7,8,9)
	Phase 2		
	(a)	(2)	(3,4,5,7,8,9)
O(n)	(b)	(2,3)	(4,5,7,8,9)
	..	..	..
	(g)	(2,3,4,5,7,8,9)	()

Tri “en-place”

- Pas besoin d'une structure de données externe, nous pouvons implémenter ces tris **en-place** (dans une seule liste)
- On se sert d’une partie de la liste comme file d'attente avec priorités.
- Pour le tri par insertion en-place
  - Nous envoyons et gardons triée la partie gauche de la séquence (Q)
  - Nous pouvons faire des échanges au lieu de modifier la séquence



Séquence S	Q
(5,4,2,3,1)	()
(4,2,3,1)	(5)
(2,3,1)	(4,5)
(3,1)	(2,4,5)
(1)	(2,3,4,5)
()	(1,2,3,4,5)
(1)	(2,3,4,5)
(1,2)	(3,4,5)
..	..
(1,2,3,4,5)	()