Premier rapport - Semaine 2

Groupe 2.2

Vendredi 25 septembre 2014

Question 1

Question 2

Question 3

Question 4

Question 5

Question 6 : Arnaud Dethise

Soient deux files, q_1 et q_2.

push(A) : ajouter A dans la file non-vide (q_1).

pop() : transférer tous les éléments de la file non-vide (q_1) dans la file vide (q_2), à l'exception du dernier élément. Retourner le dernier élément. q_2 est maintenant la file non-vide, q_1 est la file vide.

Ainsi, les opérations suivantes :

```
push(1)
push(2)
push(3)
a = pop()
b = pop()
```

se traduisent en utilisant des files par :

```
q1.enqueue(1)                  %   q1: [1]          q2: []
q1.enqueue(2)                  %   q1: [2,1]        q2: []
q1.enqueue(3)                  %   q1: [3,2,1]      q2: []
q2.enqueue(p1.dequeue())       %   q1: [3,2]        q2: [1]
q2.enqueue(p1.dequeue())       %   q1: [3]          q2: [2,1]
a = q1.dequeue()               %   q1: []           q2: [2,1]
q1.enqueue(p2.dequeue())       %   q1: [1]          q2: [2]
b = q2.dequeue()               %   q1: [1]          q2: []
```

La complexité de la fonction push est alors de $\Theta(1)$ et celle de pop est de $\Theta(N)$ où N est le nombre d'élément dans la pile. Il est possible d'implémenter la pile au moyen de files de telle sorte que les complexités de push et pop soient inversées.

Cette solution est donc moins efficace que les implémentations normales d'une pile, dont la complexité est en temps constant pour push() et pop().

Nous avons également discuté en séance de la possibilité d'implémenter une file à partir de deux piles. La solution, pour laquelle la fonction push a une complexité constante $\Theta(1)$ et la fonction pop a une complexité moyenne $\Theta(1)$, est indiquée ci-dessous.

```
pile A
pile B

function enqueue(n) :
  A.push(n)

function dequeue() :
  if B.isEmpty() :
    while not A.isEmpty() :
      B.push(A.pop())
  return B.pop()
```

Question 7 : Gil De Grove, Romain Henneton

Fonction d'écriture :

```
public static int WriteInFile(String filePath, String toWrite)
  {

    Path file = Paths.get(filePath);
    if (! file.toFile().exists())
    {
      try {
        Files.createFile(file);
      } catch (IOException e) {
        e.printStackTrace();
      }

    }
    if (! file.toFile().canWrite()){
      return -1;
    }

    try {
      Files.write(file,toWrite.getBytes("US-ASCII"));
    } catch (UnsupportedEncodingException e) {
      e.printStackTrace();
    } catch (IOException e) {
      e.printStackTrace();
    }
    return 0;

  }
```

Fonction de lecture :

```
public static List<String> ReadFromFile(String filePath)
  {
    List<String> text = new ArrayList<String>();
    Path file = Paths.get(filePath);
    if(! file.toFile().exists())
    {
      return text;
    }

    try {
      text = Files.readAllLines(file, Charset.forName("US_ASCII"));
    } catch (IOException e) {
      e.printStackTrace();
    }
    return text;
  }

```