Devoir surveillé n°6 – NSI
25/02/2025

Les coûts possibles dans ce contrôle sont uniquement parmi les suivants :
linéaire $O(n)$, quadratique $O(n^2)$ ou logarithmique $O(\log_2(n))$.

**Rappel** $\log_2(n)$ peut être vu comme le nombre de fois que l'on peut diviser $n$ par 2.
C'est le coût d'une recherche dichotomique

## Exercice 1 (Coût – 6 points)

Les deux questions de cet exercice sont indépendantes.

1. On considère l'algorithme écrit en langage Python ci-dessous, où `n` est un entier.

```
x=1
for i in range(2,6):
  for j in range(n):
    print(x+j)
  x=x+i
```

   (a) Quel est le nombre de sommes effectuées, en fonction de `n`, dans l'algorithme ci-contre ?
   Expliquer.

   (b) En déduire le coût de l'algorithme en fonction de `n`.

2. Pour chacun des algorithmes suivants,
   - Évaluer, en expliquant, le nombre de fois que la boucle se répète en fonction de `n` ;
   - En déduire le coût en fonction de `n`.

(a)
```
x = 0
while x<n:
    x = x+1/n
```

(b)
```
x = 1
while x<n:
    x = x*2
```

(c)
```
x = 0
while x<n:
    x = x+2
```

## Exercice 2 (8 points)

On considère ci-contre une fonction écrite en langage Python.
Le paramètre `liste` est une liste de nombres flottants non vide (autrement dit, `liste` a une longueur `n`$\geqslant 1$).

```
def fonction(liste):
  maxi = liste[0]
  n = len(liste)
  i = 1
  while i < n:
    if liste[i]>maxi:
      maxi = liste[i]
    i+=1
  return maxi
```

1. On exécute `fonction([5,2.3,7,6])`.
   Détailler l'exécution de l'algorithme à l'aide du tableau suivant, à reproduire et compléter (lire les indications plus bas) :

| `i` (itération de la boucle) | avant la boucle | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| `i<n` | | | | | |
| `liste[i]` | | | | | |
| `liste[i]>maxi` | | | | | |
| `maxi` | | | | | |

   La ligne `i<n` est celle de la condition de la boucle `while`, il s'agit donc de remplir avec V (vrai) ou F (faux). Il en est de même pour la ligne `liste[i]>maxi`, condition du `if`.
   Pour la première colonne (avant la boucle), ces conditions ne sont pas testées, d'où les cellules grisées et barrées.
2. En déduire ce que retourne `fonction([5,2.3,7,6])`.
3. Démontrer que la boucle de l'algorithme se termine dans tous les cas.
4. Démontrer que la propriété « `maxi` est le maximum de `liste[0:i]` et `i`$\leqslant$`n` » est un invariant de la boucle.
   On rappelle que `liste[a:b]` est la sous-liste des éléments de `liste` d'indices `a` à `b-1` inclus si `b>a`, sinon c'est la liste vide.
5. Déduire des questions précédentes que la fonction retourne l'élément maximal de `liste`.

Devoir surveillé n°6 – NSI
25/02/2025

Les coûts possibles dans ce contrôle sont uniquement parmi les suivants :
linéaire $O(n)$, quadratique $O(n^2)$ ou logarithmique $O(\log_2(n))$.

**Rappel** $\log_2(n)$ peut être vu comme le nombre de fois que l'on peut diviser $n$ par 2.
C'est le coût d'une recherche dichotomique

## Exercice 1 (Coût – 6 points)

Les deux questions de cet exercice sont indépendantes.

1. On considère l'algorithme écrit en langage Python ci-dessous, où `n` est un entier.

```
x=1
for i in range(n):
  for j in range(2,5):
    print(x+j)
  x=x+i
```

   (a) Quel est le nombre de sommes effectuées, en fonction de `n`, dans l'algorithme ci-contre ?
   Expliquer.

   (b) En déduire le coût de l'algorithme en fonction de `n`.

2. Pour chacun des algorithmes suivants,
   - Évaluer, en expliquant, le nombre de fois que la boucle se répète en fonction de `n` ;
   - En déduire le coût en fonction de `n`.

(a)
```
x = 1
while x<n:
    x = x*2
```

(b)
```
x = 0
while x<n:
    x = x+1/n
```

(c)
```
x = 0
while x<n:
    x = x+2
```

## Exercice 2 (8 points)

On considère ci-contre une fonction écrite en langage Python.
Le paramètre `liste` est une liste de nombres flottants.

```
def fonction(liste):
  res = 0
  n = len(liste)
  i = 0
  while i < n:
    res = res+liste[i]
    i+=1
  return res
```

1. On exécute `fonction([5,2.3,7,6])`.
   Détailler l'exécution de l'algorithme à l'aide du tableau suivant, à reproduire et compléter (lire les indications plus bas) :

| étape de la boucle | avant | 1 | 2 | 3 | ... |
|---|---|---|---|---|---|
| `i<n` | | | | | |
| `liste[i]` | | | | | |
| `res` | | | | | |
| `i` (fin d'itération) | 0 | | | | |

   La ligne `i<n` est celle de la condition de la boucle **`while`**, il s'agit donc de remplir avec V (vrai) ou F (faux).
   Pour la première colonne (avant la boucle), la condition n'est pas testée, d'où les cellules grisées et barrées.

2. En déduire ce que retourne `fonction([5,2.3,7,6])`.
3. Démontrer que la boucle de l'algorithme se termine dans tous les cas.
4. Démontrer que la propriété « `res = liste[0]+...+liste[i-1]` et `i`$\leqslant$`n` » est un invariant de la boucle.
   On considérera que la somme `liste[0]+...+liste[i-1]`, qui est vide si `i-1<0`, est nulle dans ce cas.
5. Déduire des questions précédentes que la fonction retourne la somme des termes de la liste `liste`.