# IF111 - Algorithmes et structures de données
# EI6 - Examen Blanc

Rohan Fossé

rfosse@labri.fr

## 1 Multiplication Russe

On considère l'algorithme suivant qui calcule et retourne la multiplication de deux entiers positifs $m$ et $n$ :

```
1: function mult(m, n)
2:     r ← 0
3:     while n > 0 do
4:         if n%2 == 1 then
5:             r ← r + m
6:         m ← m * 2
7:         n ← n//2
8:     return r
```

1. Compter le nombre d'opérations primitives exécutées à chaque ligne.
2. En majorant ce nombre, donner une complexité en temps dans le pire cas par rapport à n. O(log n)

## 2 Fonction d'Ackermann

Ecrire un algorithme récursif qui calcule $A(m, n)$ définit comme ceci:

$$\begin{cases} A(0,n) = n+1 \\ A(m,0) = A(m-1,1), \text{pour m} > 0 \\ A(m,n) = A(m-1, A(m,n-1)), \text{pour m} > 0 \text{ et n} > 0 \end{cases}$$

**Exercice 5 : Ackermann**

**Algorithme** Ackermann($m$, $n$ :entier) : entier
**début**
  **si** $m = 0$ **alors**
    **retourner** $n + 1$
  **sinon**
    **si** $n = 0$ **alors**
      **retourner** Ackermann($m - 1$,1)
    **sinon**
      **retourner** Ackermann($m - 1$, Ackermann($m$, $n - 1$))
    **fin si**
  **fin si**
**fin**

## 3 Recherche dichotomique

Écrire un algorithme récursif de recherche dichotomique d'un élément dans un tableau ordonné.

Dans les deux exercices suivants, vous pourrez utiliser les fonctions suivantes :

- $vide(l)$ : Renvoie $True$ si la liste $l$ est vide et $False$ sinon;
- $cons(a, l)$: Renvoie la liste obtenue en ajoutant l'élément $a$ devant la liste $l$;
- $tete(l)$: Renvoie le premier élément de la liste $l$;
- $queue(l)$ Renvoie la queue de la liste, ie $l$ sans le premier élément.

```
Algorithme  recherche(n:entier, t:tableau d'entiers, a, b: entier) :  : boolen
variable  c : entier
début
  si a > b alors
    retourner Faux
  sinon
    c ← (a + b)/2
    si t[c] = n alors
      retourner Vrai
    sinon
      si t[c] < n alors
        retourner recherche(n, t, c+1, b)
      sinon
        retourner recherche(n, t, a, c-1)
      fin si
    fin si
  fin si
fin
```

## 4 Nombre d'occurrence

Définir un algorithme qui compte le nombre d'occurrence d'un élément $x$ dans une liste donnée.

**Exercice 3 : Nombre d'occurence**

```
Algorithme  nbOccurence(L : liste d'entier, x : entier) : entier
début
  si listeEstVide?(L) alors
    retourner 0
  sinon
    si listeTête(L) = x alors
      retourner 1 + nbOccurence(listeQueue(L), x)
    sinon
      retourner nbOccurence(listeQueue(L), x)
```

## 5 Retournement

Correction sur la page suivante

Définir un algorithme qui inverse l'ordre des éléments d'une liste $l$.

## 6 Tas min

Écrire une fonction qui vérifie si un arbre binaire est un tas min. Quelle est sa complexité ?

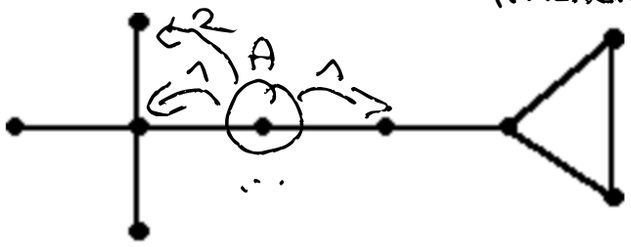
## 7 Le robot

Un robot se promène sur le graphe 1. Partant d'un sommet quelconque s, appelé sommet de stockage, il doit déposer un cube sur chacun des autres sommets. Il possède suffisamment de cubes sur le sommet de stockage, mais ne peut transporter qu'un cube à la fois (il doit donc repasser par le sommet de stockage avant de livrer un autre cube).

**Exercice 4 : Retournement**

```
Algorithme  retournementTerminal(l : liste d'entiers, m : liste d'entiers) :
  rien
début
  si listeEstVide?(l) alors
    retourner m
  sinon
    retourner retournement(listeQueue(l), listeCons(listeTête(l), m))
  fin si
fin
Algorithme  retournement(l : liste d'entiers, m : liste d'entiers) : rien
début
  retourner retournementTerminal(l, listeVide())
fin
```

1. Calculer, pour chacun des sommets du graphe, le trajet minimum que doit parcourir le robot si ce sommet est sommet de stockage.
2. Quel est le "meilleur" sommet de stockage ? Faire la somme des distances minimales.

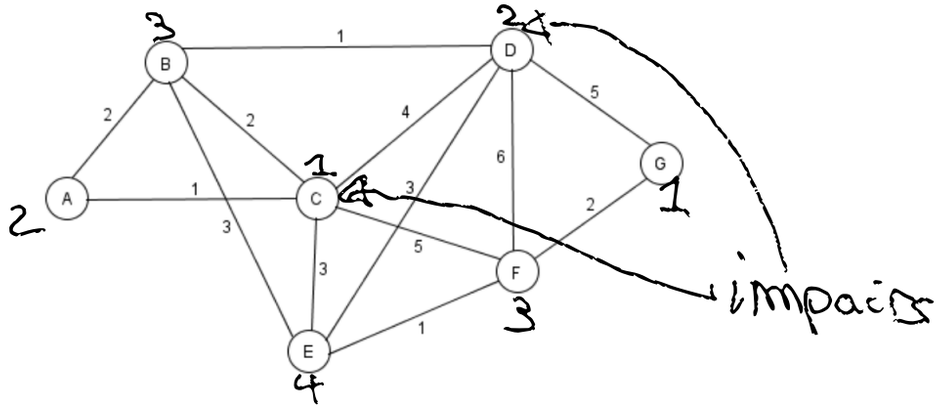



ex A = 1+2+2+2+1+2+3+3 = 16 (=32 si on compte le retour)

Figure 1: Graphe du robot

## 8 Le facteur

Le graphe ci-dessous représente le plan d'une agglomération. Les sommets B, C, D, E, F et G désignent des villes. Une arête représente le trajet entre deux villes et est pondérée par le nombre de feux tricolores situés sur ce trajet.




On s'intéresse au graphe non pondéré. Répondre aux questions suivantes :

1. Ce graphe est-il connexe ? Oui
2. Ce graphe est-il complet ? non
3. Ce graphe admet-il une chaîne eulérienne ? oui car 2 impairs (voir définition)

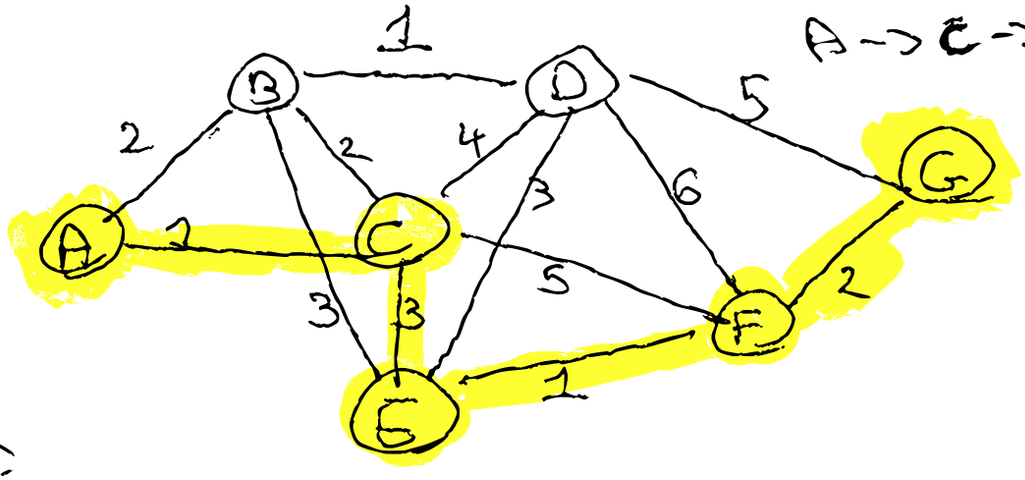
4. Ce graphe admet-il un cycle eulérien ? non car deux impairs

5. Déterminer, en justifiant, le nombre chromatique de ce graphe.

6° On s'intéresse dorénavant au graphe pondéré. Proposer un trajet comportant un minimum de feux tricolores reliant A à G.

5. On sait que le nombre chromatique $\geq 4$ car $K_4$ est un sous-graphe de ce graphe. On a trouvé une coloration propre à 4 couleurs donc le nombre chromatique est de 4.

6°




Donc meilleur chemin
A -> C -> E -> F -> G
de longueur 7

Noeud visité

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| C | 1 | A | A | 3 | C | 6 | |
| B | | | | | | | |
| D | | | | | | | |
| E | | | | | | E | F |
| F | | | | | | | |

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| C | 0 | 2 | 1 | 3 | 4 | 6 | |
| B | | | | | | | |
| D | | | | | | | |
| E | | | | | | 5 | |
| F | | | | | | | 7 |