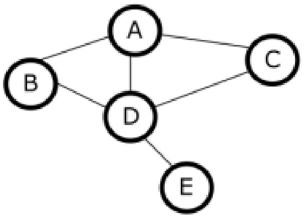
Evaluation théorique N°4_Corrigé

Exercice 1 :

1.
   a. 192.168.128.131
   b. nombre d'adresses possible 256 (nombre de machines réellement adressables : 254, car il faut retirer le 0 (adresse réseau) et le 255 (adresse de broadcast)).
2.
   a. liste des routeurs directement reliés à A (métrique = 1) : B, C et D
   b.



3.

Débit	100 kbps	500 kbps	10 Mbps	100 Mbps
Métrique associée	1000	200	10	1

4.
   a. le chemin emprunté par un message pour aller de F à I est :
   F -> H -> J -> K -> I (coût de 13, tous les autres chemins possibles ont un coût supérieur à 13).

   b.

Destination	Métrique
F	0
G	8
H	5
I	13
J	6
K	8
L	11

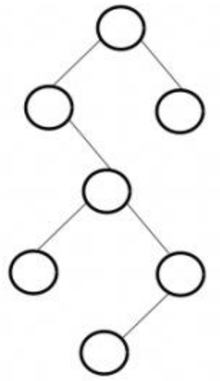
   c.
   Trois routeurs permettent d'accéder à F directement I, H, et G. Si H tombe en panne, il resterait donc uniquement G et I. Or le coût de la liaison I -> F est de 20, ce qui rend préférable l'option de passer par G dans tous les cas de figure (par exemple, même pour aller de I à F, le chemin retenu serait I -> K -> J -> G -> F avec un coût de 19).

Exercice 2 :

1.

   a. La hauteur de l'arbre est de 2

   b.



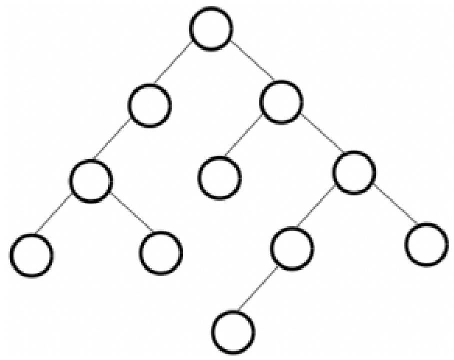
2.

Algorithme hauteur(A):
 test d'assertion : A est supposé non vide
 si sous_arbre_gauche(A) vide et sous_arbre_droit(A) vide:
  renvoyer 0
 sinon, si sous_arbre_gauche(A) vide:
  renvoyer 1 + hauteur(sous_arbre_droit(A))
 sinon, si sous_arbre_droit(A) vide :
  renvoyer 1 + hauteur(sous_arbre_gauche(A))
 sinon:
  renvoyer 1 + max(hauteur(sous_arbre_gauche(A)),
      hauteur(sous_arbre_droit(A)))

3.

   a. Si D est l'arbre vide, la hauteur de R est égale à 1 + hauteur(G) soit 1+2=3, comme la hauteur de R est 4, D ne peut pas être l'arbre vide .
   Comme ni D et ni G sont des arbres vides, hauteur(R) = 1+max(hauteur(G), hauteur(D)), d'où hauteur(R) = 1+max(2,hauteur(D)) avec hauteur(R) = 4, hauteur(D) est obligatoirement égale à 3.

   b.



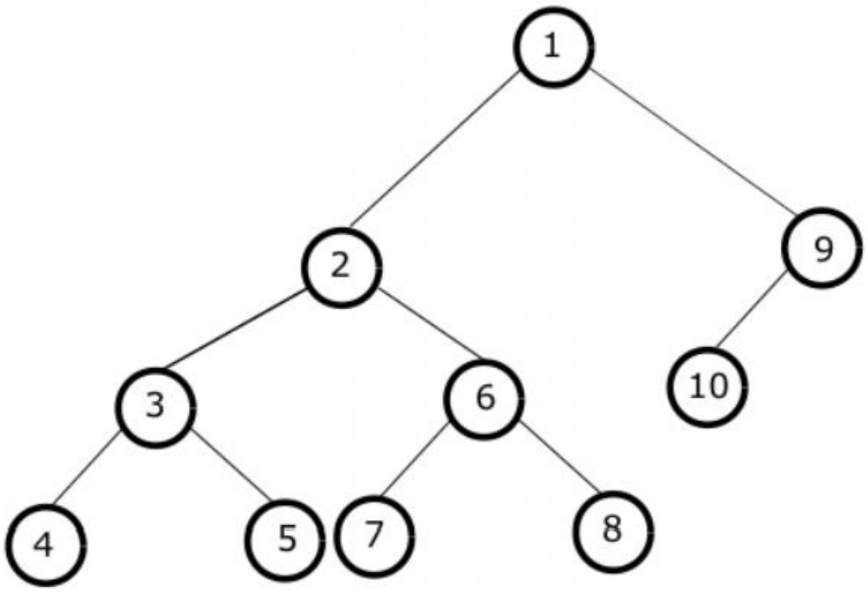
4.

a. L'arbre de la question 1.a. possède 4 nœuds d'où n = 4
Ce même arbre a une hauteur de 2 d'où h = 2
D'où h+1 = 3 et $2^{h+1} - 1 = 2^3 - 1 = 7$
Nous avons bien 4 qui est compris entre 3 et 7, les inégalités sont donc vérifiées.

b. Pour créer un arbre binaire de hauteur h quelconque ayant h+1 nœuds, il suffit de créer un arbre où chaque nœud, a au maximum un enfant.

c. Pour créer un arbre binaire de hauteur h quelconque ayant $2^{h+1} - 1$ nœuds, il suffit de créer un arbre où chaque nœud, qui n'est pas une feuille, a 2 enfants.

5.



6.

```
def fabrique(h, n):
  def annexe(hauteur_max):
    if n == 0 :
      return arbre_vide()
    elif hauteur_max == 0:
      n=n-1
      return arbre(arbre_vide(), arbre_vide())
    else:
      n=n-1
      gauche = annexe(hauteur_max - 1)
      droite = annexe(hauteur_max - 1)
      return arbre(gauche, droite)
  return annexe(h)
```