Mohamed Aziz TAIEB 11927449

Devoir d'algorithmique

3 -Etude préliminaire et approche heuristique

3.1)En utilisant les arbres , on peut trouver tout les choix possibles . Le gain maximum sera le max entre le score de toutes les feuilles . Les arbres des tableaux 1,2 sont dans le fichier tableau.pdf et servent d'exemple . Chaque noeud dans l'arbre représente le cout de ce choix .
Le gain maximal du tableau 1 est 150.
Le gain maximal du tableau 2 est 160.
Le gain maximal du tableau 3 est 180.
Le gain maximal du tableau 4 est 290.

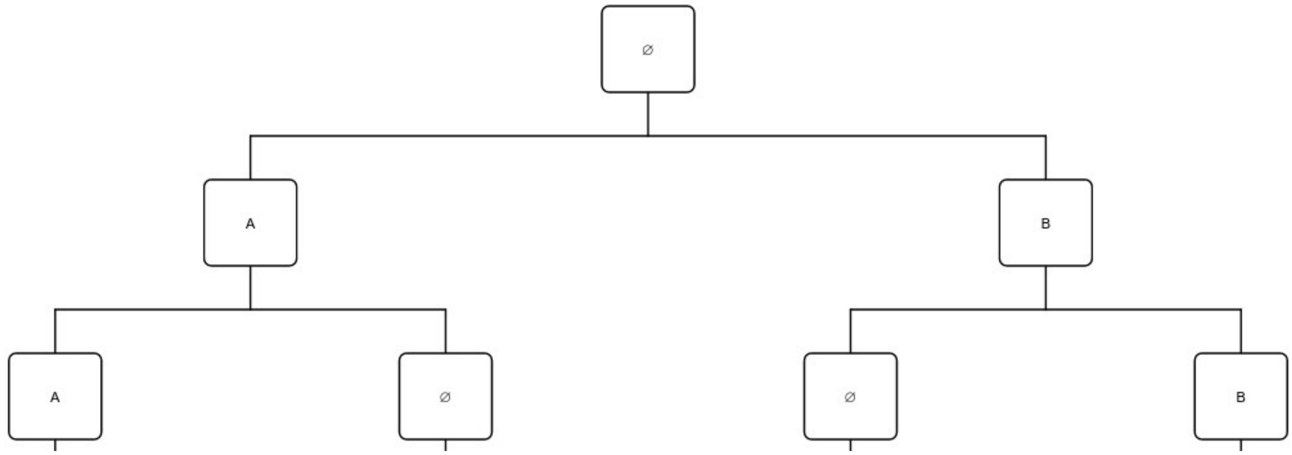
3.2)La complexité de cette approche est linéaire , en O(n) . On effectue une seule boucle allant de 1 à n .
Cet approche est incorrecte en prenant le premier tableau comme contre exemple on obtient pour Planif (n, A, B, 1):
t=1, Temps=40
t=3, Temps=40+60=100
t=4, Temps=100+20=120
t=5, Temps=120+20=140

Pour Planif (n, A, B, 0):
t=1, Temps=20
t=2, Temps=20+20=40
t=3, Temps=40+60=100
t=4, Temps=100+20=120
t=5, Temps=120+20=140

Le max est 140 mais dans l'exercice précédent on avait montré que le score maximum est 150.

4 -Algorithmes de résolution exacte :

4.1) On suppose que le nombre de solution est représenté par une suite U_n qui est égale au nombre de choix d’éxecution possibles pour n minutes .
On sait que pour n=1 , $U_n = 2$ et pour n=2 , $U_n = 4$.



On définit donc $U_n = U_{n-1} + U_{n-2}$, pour n>=3 .

4.2) Pour t=1 , f(1,A)=A[1]
f(1,B)=B[1]

Pour t=2 , si on se trouve en A alors forcément le travail au temps t-1 a été effectué sur le calculateur A sinon au temps t=2 on sera en attente . IDEM pour B. Alors :
Pour t=2 , f(2,A)=A[1]+A[2]=f(1,A)+A[2]
f(2,B)=B[1]+B[2]=f(1,B)+B[2]

4.3) Si pour t-1 , on a effectué le calcul sur la machine A alors au temps t
f(t,A)=f(t-1,A)+A[t]

4.4) Si pour t-2 , on a effectué le calcul sur la machine B alors au temps t
f(t,A)=f(t-2,B)+A[t]

4.5) Pour tout instant t>=3 , si on est sur la machine A on a 2 cas possibles:
Soit f(t,A)=f(t-1,A)+A[t] donc au moment t-1 on utilise la machine A.
Soit f(t,A)=f(t-2,B)+A[t] donc au moment t-1 on a migré et donc forcément en t-2 on a utilisé B.

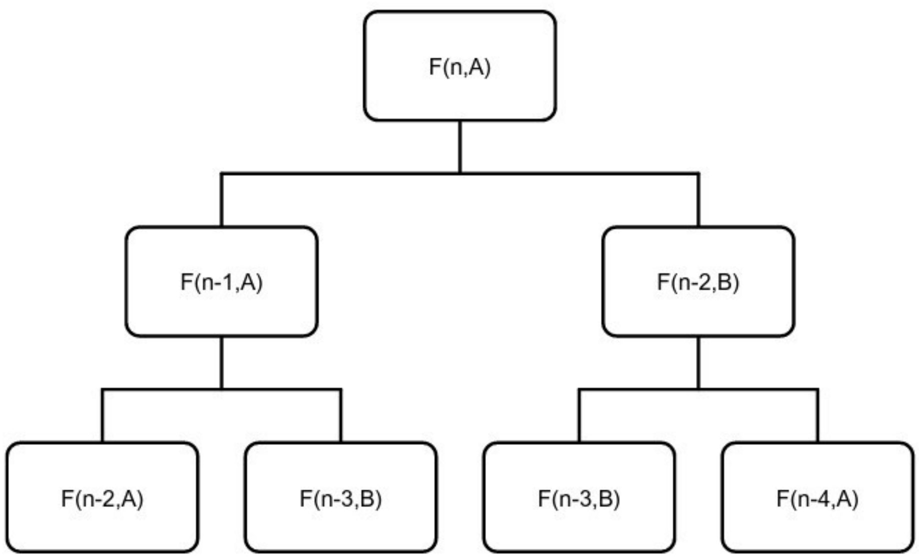
Alors pour t>=3, $f(t,A)=\text{Max}\{f(t-1,A)+A[t],f(t-2,B)+A[t])\}$
$=\text{Max}\{f(t-1,A),f(t-2,B)\}+A[t]$

Idem pour B , $f(t,B)=\text{Max}\{f(t-1,B)+B[t],f(t-2,A)+B[t])\}$
$=\text{Max}\{f(t-1,B),f(t-2,A)\}+B[t]$

4.6)Dans le cas général , $f(n,X)=X[1]$ SI n=1
$X[1]+X[2]$ SI n=2
$\text{Max}\{f(n-1,X),f(n-2,\bar{X})\}+X[t]$ SINON

4.7) Codé en C dans le fichier f_recherche.c

En utilisant les arbres , on se rend compte qu'on doit à chaque instance n , appeler la fonction $f(n-1,X)$ et $f(n-2,\bar{X})$ et prendre le max entre les deux .
Dans chaque appelle de fonction on effectue le même appel jusqu'à obtenir n - nbitérations =1 .
Comme on fait deux appels de fonction par appel de fonction , la complexité est en $O(2^n)$.
Voici un exemple d'appel de fonction f(n,A):



4.8)Codé en C fichier: algo_dynamique.c

```
for(i=0;i<n;i++){
    if(i==0){//t=1
      m[0][0]=tab[0][0];
      m[1][0]=tab[1][0];
    }else if(i==1){//t=2
      m[0][1]=m[0][0]+tab[0][1];
      m[1][1]=m[1][0]+tab[1][1];
    }else{
      m[0][i]=max(m[0][i-1],m[1][i-2])+tab[0][i];
      m[1][i]=max(m[1][i-1],m[0][i-2])+tab[1][i];
    }
  }
```

Soit c=2 le nombre de calculateur et n le nombre de minutes total .
La complexité est en O(cxn) .
On suppose que c=n alors la compléxité est quadratique , $O(n^2)$.

4.9)Codé en C fichier: algo_dynamique.c

5.1)On effectue le même algorithme dynamique mais en vérifiant pour tout les calculateurs .
Pour tout i ,

$f(n,A_i)=A_i[1]$	SI n=1
$A_i[1]+A_i[2]$	SI n=2
$\max\{f(n-\|i-1\|-1, A_1),f(n-\|i-2\|-1, A_2).....f(n-\|i-m\|-1,A_m)\}$	SINON

On supposera que si $n-|i-j|-1<1$ alors $f(n-|i-j|-1, A_j)=0$, pour éviter le débordement du tableau .

5.2)Codé en C fichier: jesuiscourageux.c