Min Makespan

Discussion sur le programme

# Complexité en temps

## Generation d’instances

### Random

#### Pseudo-code :

Def doRandom(m : entier, n : entier, k : entier, min : décimal, max : décimal) :

Initialiser datas tableau 2-D de décimaux

Pour i allant de 1 à k :

Initialiser data tableau de décimaux

data[1] = m

data[2] = n

data[3 à 3 + n] = generation de n nombres décimaux aléatoires entre min et max

Ajouter data à datas

Fin Pour

Fin doRandom

#### Complexité :

On réalise une boucle k fois, dans laquelle on génère n nombre décimaux, on a donc une complexité de O(n\*k)

### Im

#### Pseudo-code :

Def doIm(m : entier) :

Initialiser data tableau de 2m+3 décimaux

data[1] = m

data[2] = n

durée = m

Pour i allant de 1 à 2n + 1 :

Si i > 3 et i est pair : ajouter 1 à durée

Ajouter durée à data

Fin Pour

Fin doIm

#### Complexité :

On a simplement une boucle de 2n+1 itérations, donc une complexité de O(n)

## Copie en mémoire des instances

### If et Ic

#### Pseudo-code :

Def lectureFichier(fichier : String)

Récupérer dans ligne la première ligne du fichier fichier

Découper ligne à chaque « : »

m = ligne[1]

n = ligne[2]

initialiser un tableau de décimaux de taille n + 2, y ajouter m et n

Pour i allant de 1 à n :

Tableau[i + 2] = ligne[i + 2]

Fin Pour

Fin lectureFichier

Le code pour l’entrée de l’utilisateur est le même, mais au lieu de lire la première ligne du fichier, lire l’entrée utilisateur.

#### Complexité :

Une unique boucle allant de 1 à n, donc la complexité est de n.

## Algorithmes :

### LSA :

#### Pseudo-code :

Def LSA(m : entier, n : entier, taches : tableau de décimaux, machines : tableau de décimaux)

Pour i allant de 1 à n :

Indice = indice de min(machines)

Machines[indice] += taches[i]

Fin Pour

Retourner machines

Fin LSA

Le paramètre machines permettra de réaliser une LSA après traitement dans MyAlgo.

#### Complexité :

On a une boucle de 1 à n, dans laquelle on cherche le min de machines. On a une complexité de O(n \* c(min)) soit O(n \* m).

### LPT :

#### Pseudo-code :

Def LPT(m : entier, n : entier, taches : tableau de décimaux, machines : tableau de décimaux)

trier taches dans l’ordre décroissant

Réaliser LSA(m, n, taches, machines)

Fin LPT

#### Complexité :

On a une complexité de O(c(tri) + c(LSA)), soit O(n2 + n\*m). On peut supposer m ≤ n car le cas contraire signifie qu’il suffit de retourner le plus grand élément de taches (voir rapport des exercices, question 4). On a alors, pour conclure, une complexité O(n2)

### MyAlgo

#### Pseudo-code :

Def MyAlgo(m : entier, n : entier, taches : tableau de décimaux, machines : tableau de décimaux)

Moyenne = somme(taches)/m

TachesRestantes = tableau vide de décimaux

Pour iM allant de 1 à m :

Pour chaque tache T :

Si machines[iM] + T < moyenne :

Machines[iM] += T

Sinon :

Ajouter T à TachesRestantes

Fin Si

Retirer T de taches

Fin Pour

Fin Pour

Réaliser LPT(m, n, TachesRestantes, machines)

Fin MyAlgo

#### Complexité :

On a une complexité O(c(premièresAffectations) + c(LPT)). On sait que c(premièresAffectations) contient 2 boucles imbriquées de m itérations puis n itérations et que c(LPT) vaut O(n2), donc MyAlgo a une complexité O(n\*m + n2). Comme vu dans la partie Complexité de LPT, on peut considérer cette complexité égale à O(n2). En appliquant la LSA plutôt que la LPT en fin d’algorithme, on peut réduire notre complexité à O(n\*m), sans beaucoup changer les résultats.

## Comparaison d’algorithmes

### Sur Im

On a vu dans le rapport d’exercices (question 11 et 13) que LSA a un ratio d’approximation de 3/2 et LPT a un ratio de 4/3. MyAlgo a relativement les mêmes résultats que LPT, et tend à un ratio de 4/3 quand m est grand.

### Sur Random

Pour étudier le degré d’approximation sur des listes de taches aléatoires, cela se complexifie car beaucoup de paramètres rentrent en jeu : m, n, min, max.

Pour m = 1000, n = 5000, min = 10, max = 100 :

LSA : 1.23 -- LPT : 1.03 – MyAlgo : 1.06

Pour m = 1000, n = 1500, min = 10, max = 100 :

LSA : 1.47 – LPT : 1.0 – MyAlgo = 1.12

Pour m = 1000, n = 5000, min = 1, max = 2

LSA : 1.15 – LPT : 1.01 – MyAlgo : 1.01

Pour m = 1000, n = 5000, min = 1, max = 100

LSA : 1.25 – LPT : 1.04 – MyAlgo : 1.07

On peut donc voir que rapprocher n de m dégrade le ratio de LSA et MyAlgo tandis que cela améliore celui de LPT.

De plus, augmenter le rapport max/min dégrade tous les ratios, mais MyAlgo en souffre plus que LPT.

## Classes d’instances « fétiches »

### LSA

### LPT

### MyAlgo