Rapport Recherche Opérationelle

Date : 5 janvier 2015

Axandre PATRY, Benjamin BRIAND

Le répertoire **RO** contient deux répertoire **q1** et **q2** qui correspondent aux questions 8 et 9 du projet. Pour compiler les projets, il faut se rendre à la racine d'un des répertoires puis tapper la commande **javac -d ./ -cp . ./*.java** puis exécuter le main en faisant **java projet.main**.

1 Question8 : un algorithme glouton

Objectif : Implémentation de l'algorithme glouton pour obtenir une borne primale au problème.

Code

L'algorithme glouton se trouve dans le fichier **q1/ro.java** :

```
package projet;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.Collections;
import java.util.Iterator;
import java.lang.Math;
class ro {
//sort taks
//int compareTo(Tache);
//construit le tableau des intervalles 0: dispo 1:occupé

public int verif (int dateDebut, int duree, boolean intervalle[]){
    int j = dateDebut;
    while (j < (dateDebut + duree)){
if(intervalle[j]){
    j++;
    return j;
}
j++;
    }
    return dateDebut;
}
```

```
public void insereTache (int dateDebut, int duree, boolean intervalle[]){
    int j = dateDebut;
    while (j < (dateDebut + duree)){
intervalle[j]=true;
j++;
    }
}

    public int cout(ArrayList <Tache> taches){
int cout = 0;
Iterator<Tache> iterator= taches.iterator();
while (iterator.hasNext()){
    Tache tache = iterator.next();
    cout+= Math.max(tache.dateDebut + tache.duree - tache.livraison,0);
}
return cout;
    }


 public void glouton (ArrayList<Tache> taches){
    boolean [] intervalle = new boolean[100000];
    Collections.sort(taches);
    Iterator<Tache> iterator= taches.iterator();
    while (iterator.hasNext()){
Tache tache = iterator.next();
int dateDebut = tache.dispo;
int j = verif(dateDebut,tache.duree, intervalle);
while(dateDebut != j){
    dateDebut = j;
    j = verif(dateDebut,tache.duree, intervalle);
}
tache.dateDebut = j;
insereTache(dateDebut, tache.duree, intervalle);
    }
}

}
```

2 Explications

On a pris un tableau de temps de taille arbitraire suffisemment grande. D'où le 100000. Dans l'algorithme on trie la liste des taches, et on vérifie que la date de début d'une tâche ne s'intersecte pas avec la date de fin d'une autre tâche. On construit un tableau des intervalles dans lequel 0 indique qu'aucune tâche n'est prises, et 1 pour dire que ce créneau est pris par une tâche. La fonction `glouton` se charge d'appeler `verif` et `insereTache` tant qu'il y a des Taches. Ainsi on ordonne les taches de maniere optimale.

3 Resultats

Dans le fichier **q1/main.java**, trois tâches ont été créees :

```
//Tache (int duree, int dispo, int livraison, int indice)

Tache tache1 = new Tache(34, 82, 100, 1);
Tache tache2 = new Tache(23, 146, 108, 2);
Tache tache3 = new Tache(100, 123, 215, 3);
```

Voici le résultat (tapper la commande **java projet.main**) :

```
[tache numero 2  dateDebut : 146
, tache numero 1  dateDebut : 82
, tache numero 3  dateDebut : 169
]
Borne primale :131
```

4 Question9 : un algorithme glouton

Code

L'algorithme glouton se trouve dans le fichier **q2/ro.java** :

```
package projet;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.Collections;
import java.util.Iterator;
import java.lang.Math;
class ro2 {


public Tache2 tache2(int t, ArrayList<Tache2> taches){
int dureeMin = 100000;
int livraisonMin = 100000;
Tache2 result = new Tache2();
Iterator <Tache2> iterator = taches.iterator();
while(iterator.hasNext()){
Tache2 tache = iterator.next();
if(tache.dispo <= t && tache.duree <= dureeMin){
    if (tache.duree == dureeMin){
if(tache.livraison <= livraisonMin){
    result = tache;
    dureeMin = tache.duree;
    livraisonMin = tache.livraison;
}
    }
    else{
```

```
 result = tache;
 dureeMin = tache.duree;
 livraisonMin = tache.livraison;
    }
}
}
return result;
}


  public int [] glouton2 (ArrayList<Tache2> taches){
  int t = 0;
  int []intervalle = new int [100000];
  while (!taches.isEmpty()){
Tache2 tacheEnCours = tache2(t, taches);
intervalle[t] = tacheEnCours.indice;
tacheEnCours.duree --;
if(tacheEnCours.duree <= 0){
taches.remove(tacheEnCours);
}
t++;
}
return intervalle;
}

    public int cout(int[] solution, ArrayList<Tache2> taches){
int []arrets = new int[taches.size()];
Integer coutTotal = 0;
Iterator <Tache2> iterator = taches.iterator();
for(int i = 0; i <100000;i++){
    if(solution[i]>0)
arrets[solution[i] - 1] = i;
}
ArrayList<Integer> d_i = new ArrayList<Integer>();
ArrayList<Integer> C_i= new ArrayList<Integer>();
while(iterator.hasNext()){
    Tache2 tache = iterator.next();
    d_i.add(tache.livraison);
}
for (int i=0;i<taches.size();i++){
    C_i.add(arrets[i]);
}
Collections.sort(d_i);
Collections.sort(C_i);
for (int i = 0; i < taches.size();i++){
    coutTotal += Math.max(C_i.get(i) - d_i.get(i) + 1,0);
}

return coutTotal;
    }
```

```
public void print(int []entier, int l){
for(int i =0; i <  l; i++){

System.out.print("|t =" + i + ", tache numero" + entier[i] + "|");

}
System.out.println("");
}
}
```

5 Resultats

Dans le fichier **q2/main.java**, trois tâches ont été créees :

```
//Tache2 (int duree, int dispo, int livraison, int indice)

Tache2 tache1 = new Tache2(3, 0, 5, 1);
Tache2 tache2 = new Tache2(4,1,6, 2);
Tache2 tache3 = new Tache2(1,3,8,3);
```

Voici le résultat (tapper la commande **java projet.main**) :

```
|t =0, tache numero 1||t =1, tache numero 1||t =2, tache numero 1||t =3, tache numero 3||
t =4, tache numero 2||t =5, tache numero 2||t =6, tache numero 2||t =7, tache numero 2||
t =8, tache numero 0||t =9, tache numero 0||t =10, tache numero 0||t =11, tache numero 0|
Borne duale :0
```

6 Question 10

Comme nous venons de le voir, nous obtenons les résultats suivants :

Problème 1 :

```
[tache numero 2  dateDebut : 146
, tache numero 1  dateDebut : 82
, tache numero 3  dateDebut : 169
]
Borne primale :131
```

Problème 2 :

```
|t =0, tache numero 1||t =1, tache numero 1||t =2, tache numero 1||t =3, tache numero 3||
t =4, tache numero 2||t =5, tache numero 2||t =6, tache numero 2||t =7, tache numero 2||
t =8, tache numero 0||t =9, tache numero 0||t =10, tache numero 0||t =11, tache numero 0|
Borne duale :0
```

7 Conclusion

Ce projet algorithmique est très intéressant car il permet de mettre en parallèle la programmation et la recherche opérationnelle.