TP4

Dans ce tp, on cherche directement le temps de délais les plus petits. On mit un tableau de temps **(int[] table = new int[100000000])** ,maintenant le temps est divisé par un seconde. Pour chaque tache, on fait deux tests, s'il y a assez de machines libres au début **if (table[taches.get(i).debut] + taches.get(i).size <= num_machine)** et s'il y a assez de machines toute la durée de cette tâche **if (table[m] + taches.get(i).size > num_machine)** . Si cette tache satisfait les deux conditions, il peut utiliser les machines (C'est à dire pendant le temps de cette tache entre jusqu'à qu'il finit, Il va occuper des machines selon la taille de cette tache). Sinon il a besoin de faire le délai jusqu'à satisfaire les deux conditions.

On peut promettre qu'il n'excite pas la possibilité qu'il y a un temps les tailles de taches qui sont plus la nombre de la machine, on commence de consulte d'améliorer les ordres de tâches pour minimiser la somme de délais. On mit l'ordre de toutes les taches par la fin des toutes les taches au grande à petit. Mais on peut aussi veut faire les taches qui sont plus petits qui sont mis plus tôt, c'est pourquoi on utilise le temps de fin qui divise la size de cette tache.

```
    return   (duree + debut) / size-(((Tache) o).duree + ((Tache) o).debut) / ((Tache) o).size;
   Collections.sort(taches);
   Collections.reverse(taches);
```

Après avoir mis toutes les taches, on arrange les ordres des tâches selon leur index.

```
Arrays.sort(arr_tache, new compare_taches());
```