Binôme Philippot Grégoire Pelage François-Xavier

A savoir :
a_i = date d'arrivée
size_i = nombreed machine nécessaire pour exectuer ai
t_i = temps d'exect de ai

Di date de démarage de la tâcheà calculer
W_i : attente avant démarage
W_i = D_i - a_i

objectif trouver un W_i minimal

C_i date de complétion de la tache =
C_i = a_i + W_i + t_i

Données :
m //nombre de machines du cluster
n //nombre de tâches
l (a_i, size_i, t_i)// liste des tâches ordre croissant des a_i

arrivée	taille	durée
0	50	1200
0	20	3600
0	20	3600
0	20	3600
2400	75	120
3000	15	7200
3400	42	1337

mettre les petites tâches en priorité

python3.6 dunno.py (production ordonnancement)
python3.6 gantt.py donnees.txt ordo.txt >> diagramme.svg (production diagramme)

algo ordonnancement :

ajout indice au tuple des tâches pour le repositionnement

```
fonction ordonnancement (liste_tache,nb_tache,nb_machine)
	l= tri(liste_tache) // on trie les taches d'une certaine façon
	l_ordo[nb_tache]
	taches_courantes=[]
	prochain_depart = 0
	machines_actuelles = nb_machine
	pour i de 1 à n //pas de 1
		Si l[i][1] <= machines_actuelles //si nombre de machines ok
			taches_courantes += l[i]
			machines_actuelles -= l[i][1] // retrait des machines utilisées
			Si l[i][0] <= prochain_depart // Si inférieur la tache démarre au
prochain départ sinon à sa date d'arrivée
				l_ordo[l[i][3]] = prochain_depart
			Sinon
				l_ordo[l[i][3]] = l[i][0]
			Fin Si
		Sinon
			Tant que(machines_actuelles < l[i][1]):
				tache_finie = plus_petite_tache() // fonction cherchant la tâche
en court qui se finira le plus tôt
				machines_actuelle += tache_finie[1] // on libère les taches
finies
				Si tache_finie[0] + tache_finie[2] < prochain_depart
					prochaind depart = tache_finie[0] + tache_finie[2]
				Fin Si
				tache_courante.remove(tache_finie) // on enlève la tache de la
liste des tâche qui sont encore "executée"
				Si tache_finie[0] + tache_finie[2] < prochain_depart  // ajouter
la durée des éléments
					prochaind depart += tache_finie[2]
				Fin Si
			Fin Tantque
			Si l[i][0] > prochain_depart
				prochain_depart = l[i][0]
			Fin Si
			machines_actuelles -= l[i][1]
			taches_courantes += l[i]
			l_ordo[l[i][3]] = prochain_depart
		fin Si
```

Ci_dessous test fait avec le fichier donnees_test.txt:

comparaison : comp_arrivee_duree
m=128 machines and n=31 tasks
Score = 305020

comp : durée
m=128 machines and n=31 tasks
Score = 1573780

comp : arrivée / durée
m=128 machines and n=31 tasks
Score = 3322600

comp arrivee + duree:
m=128 machines and n=31 tasks
Score = 305020

comp arrive et duree:
m=128 machines and n=31 tasks
Score = 300220

comp arrivee et taille:
m=128 machines and n=31 tasks
Score = 773111

comp arrivée + taille et comp size inf
m=128 machines and n=31 tasks
Score = 1189154

comp arrivée + taille et comp size sup
m=128 machines and n=31 tasks
Score = 494441

comp durée :
m=128 machines and n=31 tasks
Score = 473068

comp taille :
m=128 machines and n=31 tasks
Score = 773111