TD 26 : Allocation de salle de cours

Le principe des algorithmes gloutons a déjà été rencontré avec le problème du rendu de monnaie et le problème du sac à dos. On s'intéresse à deux nouveaux algorithmes gloutons : la sélection d'activités et l'allocation de salles de cours. On rappelle ici qu'un algorithme glouton réalise un choix optimal localement (pour un sous problème plus simple), afin de résoudre un problème d'optimisation, en espérant trouver une solution globale optimale.

Sélection d'activités : Soit E un ensemble de n activités, chacune pouvant être réalisée sur une certaine plage horaire $[d_i, f_i[$ avec $i \in [[0, n-1]]$. On cherche un sous ensemble I d'activités **disjointes** de cardinal maximal.
Exemple : Des produits à réaliser par une chaîne de production.

Méthode naïve : une recherche naïve de la solution optimale consiste à parcourir les 2^n parties de l'ensemble E des activités. La fonction à maximiser est le cardinal du sous ensemble à retenir.
Un tel algorithme effectue une recherche exhaustive mais est de complexité exponentielle (2^n cas à traiter).

Algorithme proposé : On trie les intervalles par date de fin croissantes, puis pour i allant de 0 à $n-1$, si $[d_i, f_i[$ n'intersecte aucun élément déjà choisit, on l'ajoute à I l'ensemble renvoyé à la fin.

Exercice 1 (Sélection d'activité)

1. Écrire une fonction Python `selection_optimiale(E)` qui reçoit le tableau des activités où chaque activité est un couple (d, f) et qui retourne une sélection d'activités par la méthode ci-dessus. On veillera à ne pas modifier E, et on pourra utiliser la méthode `sort` applicable à une liste.
   Aide pour `sort` : la méthode accepte une fonction de tri $L.sort(key = f)$ qui trie par valeurs croissantes des images par la fonction f. On pourra utiliser $L.sort(key = lambdax : x[0])$ qui trie tous les éléments `x` de `L` par ordre croissant de la première valeur de `x` (`x[0]`).
2. Estimer la complexité temporelle de cet algorithme (attention au tri)
3. Montrer par récurrence forte sur n que l'algorithme précédent renvoie un sous ensemble d'intervalles disjoints de cardinale maximal (à faire proprement sur feuille avant de passer à la suite).

Allocation de salles de cours : Soit C un ensemble de n cours, chacun devant se dérouler dans une salle, sur une certaine plage horaire $[d_i, f_i[$ pour $i \in [[0, n-1]]$. On cherche le nombre minimal de salles nécessaires et la répartition des cours dans ces salle.

Algorithme proposé : On trie les cours par date de fin croissante, pour i allant de 0 à $n-1$, s'il existe une salle inoccupée à l'instant d_i, on ajoute le cours à cette salle, sinon, on crée une nouvelle salle pour ce cours.

Exercice 2 (Allocation de salle de cours)

1. Montrer que l'algorithme précédent renvoie un ensemble de salles de cardinal minimal en procédant par récurrence sur le nombre d'éléments de C.
2. Écrire une fonction `est_libre(L,h)` qui prend en argument une liste L des instant où mes salles déjà utilisés sont libres, et l'heure h de début du prochain cours à affecter et renvoie l'indice i d'une salle pouvant accueillir le cours ou -1 si une nouvelle salle est nécessaire.
3. Écrire une fonction `allocation(C)` qui prend en entrée un tableau de cours sous forme de couples (d, f) et qui renvoie une liste de listes de couples, représentant l'occupation des salles de cours.
4. Déterminer la complexité temporelle de cet algorithme.