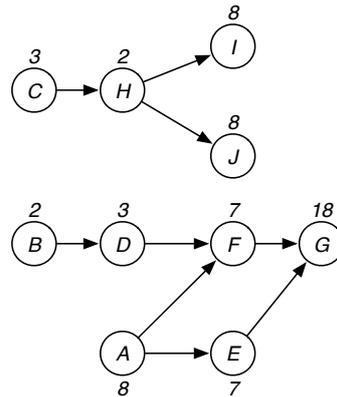


Ordonnancement – 2

1 Anomalies avec les ordonnancements de liste

On considère le graphe de tâches suivant, où les lettres représentent les noms des tâches, et les nombres associés le poids des tâches.



▷ **Question 1** Quel est le makespan obtenu par l'ordonnancement de liste se basant sur le chemin critique, sur 2 processeurs ? Est-ce optimal ?

▷ **Question 2** Supposons maintenant que le poids des tâches est diminué d'une unité (A a maintenant un poids de 7, B un poids de 1, ...). Montrez que le makespan obtenu avec l'ordonnancement basé sur le chemin critique augmente. Montrez que le makespan obtenu avec n'importe quelle heuristique de liste augmente.

▷ **Question 3** On retourne aux poids initiaux, et on suppose maintenant que nous avons 3 processeurs. Montrez que le makespan obtenu avec l'ordonnancement de liste basé sur le chemin critique augmente. Montrez que le makespan obtenu avec n'importe quelle heuristique de liste augmente.

2 Ordonnancement sur un ensemble de processeurs hétérogènes (sans communications)

On dispose de p processeurs et d'un ensemble de n tâches indépendantes T_1, \dots, T_n . On notera p_{ij} le temps de calcul de la tâche T_j sur le processeur P_i . Dans le cas où les processeurs vont simplement à des vitesses différentes (c'est-à-dire quand $p_{ij} = p_j/s_i$ où s_i représente la vitesse du processeur i et p_j la quantité de travail nécessaire au traitement de la tâche T_j), le problème est plus simple et on a le même type de résultat que dans le cas homogène. Dans le cas contraire, la meilleure approximation actuellement connue est une 2-approximation.

▷ **Question 4** Montrer que décider de l'existence d'un ordonnancement dont le temps d'exécution est 3 pour un ensemble de tâches indépendantes $\{T_1, \dots, T_n\}$ sur les processeurs P_1, \dots, P_p est un problème NP-complet. (Indication : on pourra se ramener à 3DM.)

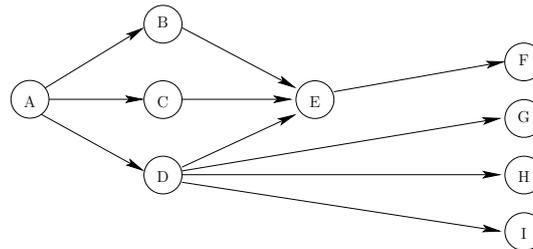
Définition 1 (3-Dimensional-Matching (3DM)). Étant donné trois ensembles $A = \{a_1, \dots, a_n\}$, $B = \{b_1, \dots, b_n\}$ et $C = \{c_1, \dots, c_n\}$ ainsi qu'un ensemble $F = \{T_1, \dots, T_p\}$ de triplets de $A \times B \times C$, trouver un sous-ensemble F' de F tel que tout élément de $A \cup B \cup C$ apparaît dans exactement un triplet de F' .

3 Ordonnement de Coffman et Graham

Soient 2 machines identiques, n tâches $(T_i), i = 1 \dots n$ de même durée et \prec un ordre partiel strict sur les tâches représentant les contraintes de précedence. On note $\sigma = (\mu, \tau)$ un ordonnancement où $\mu(i)$ est la machine effectuant T_i et $\tau(i)$ est la date de début d'exécution de T_i .

Lorsque $T_i \prec T_j$, on dit que T_j est un successeur de T_i . De plus, lorsqu'il n'existe pas de tâche T_k telle que $T_i \prec T_k \prec T_j$, on dit que T_j est successeur direct de T_i . On définit de même les notions de prédécesseur et de prédécesseur direct.

▷ **Question 5** Donner un ordonnancement optimal pour le graphe de tâches suivant.



Étant donnée une fonction de priorité p (supposée injective) sur les tâches, on s'intéresse à l'ordonnement de liste $\sigma_p = (\mu_p, \tau_p)$ défini comme suit : à chaque instant, on choisit parmi toutes les tâches prêtes celle de priorité maximale et on l'exécute sur la machine 1. De même, la seconde tâche prête la plus prioritaire est exécutée sur la machine 2.

▷ **Question 6** Pour commencer, quelle condition doit vérifier p pour que les tâches soient exécutées dans un ordre compatible avec les contraintes de précedence ?

▷ **Question 7** Montrer que la machine 1 n'est jamais inactive, et que si T_i est effectuée sur la machine 1, toutes les tâches T_j effectuées après (ou au même moment) sont de priorité plus petite que T_i .

Pour simplifier, on suppose que dans σ_p , lorsqu'il n'y a pas de tâche prête à être effectuée sur la machine 2, on effectue une tâche "fantôme" sans prédécesseur, de priorité plus faible que les tâches initiales. On définit une suite de couples de tâches (D_k, J_k) , effectuées au même moment, respectivement par la machine 1 et 2. D_0 est la dernière tâche calculée par la machine 1, de même J_0 est la dernière tâche effectuée par P_2 . J_k (si elle existe) est la tâche la plus tardive effectuée avant D_{k-1} sur la machine 2, qui soit de priorité plus petite que D_{k-1} . On note F_k l'ensemble des tâches effectuées strictement après D_{k+1} et strictement avant D_k , plus la tâche D_k , et on note E_k l'ensemble des tâches de F_k sans prédécesseur dans F_k .

▷ **Question 8** Donner pour l'exemple précédent, l'ordonnement σ_p , les tâches D_k et J_k et les ensembles F_k et E_k , en supposant que la priorité p suit l'ordre alphabétique (décroissant) des noms des tâches. Même question pour une priorité compatible avec la précedence (à préciser) conduisant à un ordonnancement optimal.

▷ **Question 9** Montrer que toute tâche de F_k est de priorité plus grande que D_k et que toute tâche T_i de F_{k-1} est successeur de D_k . En déduire que les tâches de E_{k-1} sont les successeurs directs de D_k et que tout autre successeur direct de D_k a une priorité plus faible.

On considère \prec_l l'ordre lexicographique et on suppose que la priorité p vérifie en plus la propriété suivante : $p(T_j) < p(T_i)$ si et seulement si $l(T_j) \prec_l l(T_i)$ où $l(T_j)$ (resp. $l(T_i)$) est la liste des priorités des successeurs directs de T_j (resp. T_i) classé par ordre décroissant.

▷ **Question 10** Montrer alors que toutes les tâches de F_k (notamment celles sans successeur dans F_k) sont prédécesseurs de toutes les tâches de F_{k-1} . Conclure en donnant un algorithme permettant de construire une telle priorité p et un ordonnancement de durée minimale.