Objectif
Manipuler les listes linéaires chainées et tableaux.
S'approfondir en programmation C.

Enoncé
On veut exécuter un ensemble de tâches séquentielles de manière automatique (une après l'autre). En effet, chaque tâche peut être réalisée automatiquement par plusieurs programmes ou logiciels (Prg_1,….,Prg_m). La qualité d'un programme est mesurée par 3 critères soient : temps d'exécution (T), taux de Consommation des ressources (C), taux de Fiabilité (F). Donc un programme donné Prg_i est représenté par le triplet V =(T_i, C_i, F_i)
Le but étant de choisir pour chaque tâche le meilleur programme Prgi de telle manière que la qualité globale du processus (l'ensemble des tâches) soit la meilleure, càd ayant la meilleure combinaison des 3 critères.

Donc, on commence par chercher toutes les solutions potentielles (les meilleures solutions possibles) à ce problème. Pour ce faire, on utilisera la relation de dominance entre deux vecteurs V_i (T_i,C_i,F_i) et V_j (T_j,C_j,F_j), on dit que V_i domine V_j (ou V_j est dominé par V_i) ssi:

- V_i est meilleur ou égal à V_j dans tous les paramètres {T, C, F}.
- V_i es meilleur par rapport à V_j dans au moins un paramètre{T, C, E}

i.e. :$\forall$ q $\in$ {T, C, E} : $q(V_i) \geq q(V_j)$ et $\exists$ q' $\in$ {T, C, F} : $q'(V_i) > q'(V_j)$. (Si le plus grand est meilleur sinon on utilisera $\leq$)

Exemple : (3, 2, 4) domine (3, 1, 4) et (2, 2, 4) suivant la relation $\geq$

Si on choisit Prg_{ki} (T_{ki}, C_{ki} ,F_{ki}) pour la $TÂCHE_i$ (i=1…N), la qualité globale de $S_{i=1..N}$ ($TACHE_1$, $TACHE_2$, $TACHE_3$, …..$TACHE_N$) peut être exprimée de la manière suivante (par récursion) :

$$\begin{cases} (T_{k1}, C_{k1} F_{k1}) & si\ \ N = 1 \\ \left(T_{k(i-1)} + T_{ki},\ \ \max(C_{k(i-1)},\ C_{ki}),\ \ \text{moyenne}(F_{k(i-1)}\ , F_{ki}\right) & sinon \end{cases}$$ .. **Formule2**

Une solution sera composée donc de la liste des programmes (Prg_{k1} Prg_{k2} Prg_{k3}……..Prg_{kN}) choisis pour chaque tâche.

Questions :

1. Créer la structure de donnée et la bibliothèque Bibio_Prog.h
2. Ecrire le module d'ajout d'un programme à une tâche, et celui qui permet d'ajouter une tâche à un processus.
3. Utiliser la relation de dominance pour supprimer pour chaque tâche tous les programmes candidats ***dominés*** par d'autres programmes.
4. Utiliser la relation de dominance et la formule 2 pour calculer toutes les solutions potentielles (les meilleures solutions possibles), afficher les résultats à chaque étape. Voir un exemple détaillé en annexe.
5. Trier la liste obtenue des solutions possibles selon la valeur suivante : w1*T+w2*C+w3*F tel que w1, w2, w3 sont donnés par l'utilisateur tel que w1+w2+w3=100 (des pourcentages)

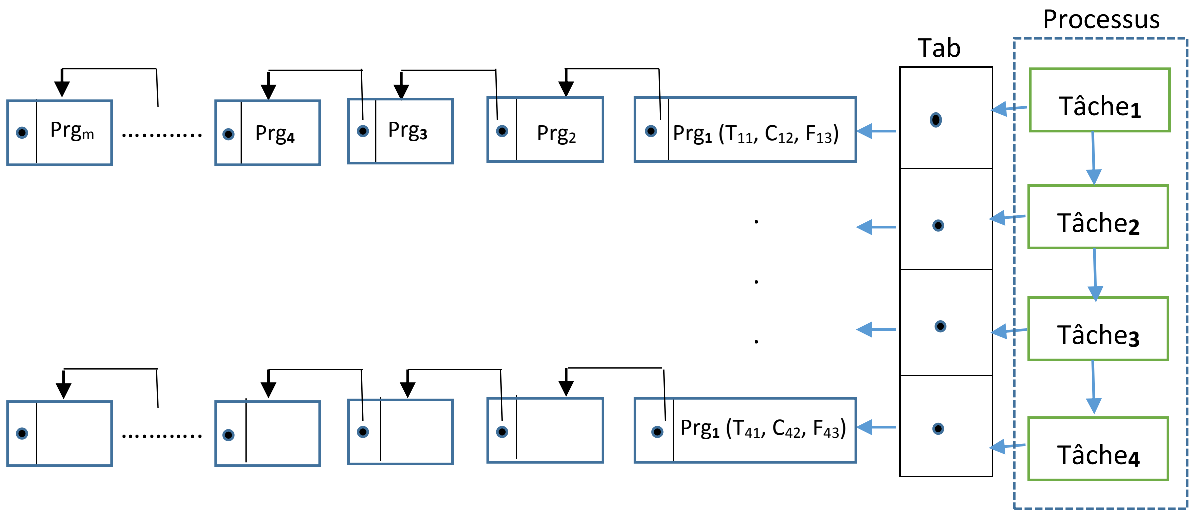

Schéma illustratif de la structure

Chaque maillon de la liste contient : le nom du programme Prg_k, les valeurs de T, C, F (4 champs) + pointeurs vers le maillon suivant.

Les solutions possibles seront représentés par une liste où chaque maillon contiendra les noms des programmes (seront concaténé), les valeurs obtenus de T, C , F.

Notes :
- Tous les programmes doivent être fonctionnels et écrits en langage C.
- Tous les programmes écrits doivent être lisibles, claires et commentés.
- TP est réalisé en binôme.
- Le code source doit être envoyé par mail ou déposé sur la plateforme TICE en fichier compressé et nommé ainsi : Nom1_prénom1_Nom2_prénom2_Groupe_TP1
- Date de remise **03 avril 2016**. Tout retard sera pénalisé.

Exemple :

Soit un processus S composé de 3 tâches : T1, T2, T3.

T1→{Prg11(2, 0.3, 0.7), prg12(4, 0.3, 0,7), prg13(1.5, 0.2, 0.7), prg14(2, 0.5, 0.8)}

T2→{Prg21(4, 0.5, 0.9), prg22(4, 0.3, 0,6), prg23(2.5, 0.3, 0.7), prg24(3, 0.3, 0.8)}

T3→{Prg31(4, 0.5, 0.9), prg32(4, 0.3, 0,6), prg33(2.5, 0.3, 0.7)}

Etape 1 : éliminer tous les candidats dominés (relation de dominance)

Prg12 est dominé par Prg11 et Prg11 est dominé par Prg3 donc supprimé de la liste Prg12 et Prg11. Pour T_1 on aura {prg13, Prg14}

En procédant de la même manière : on aura dans T2 {prg23, prg24} et dans T3{Prg31,Prg33}. Pour:

T_1 on aura {prg13, Prg14}

T_2 {prg23, prg24}

T_3{Prg31,Prg33}

Etape 2 : On combine T1, T2 et T3 (T1⊗T2) ⊗ T3, en commence par T1⊗T2=

{(Prg13⊗ prg23), (prg13⊗ prg24), (Prg14⊗ prg23), (prg14⊗ prg24)}

{(4,0.3, 0.7) , (4.5, 0.3, 0.75) , (3.5, 0.5, 0.75) , (5, 0.5, 0.8)}

⊗ est l'opérateur de combinaison

A ce niveau, il n y a aucun programme dominé, donc la liste reste telle qu'elle.

Ensuite on combine ((T1⊗T2) ⊗T3) = (T1⊗T2) ⊗ (Prg31, Prg33)

i.e. ((4,0.3, 0.7) , (4.5, 0.3, 0.75) , (3.5, 0.5, 0.75) , (5, 0.5, 0.8)) ⊗ ((4, 0. 5, 0.9) , (2.5, 0.3, 0.7)}

= **{**(4,0.3,0.7) ⊗ (4,0.5,0,9), → Prg13 ⊗ Prg23⊗ Prg31

(4,0.3,0.7) ⊗(2.5,0.3,0.7), → Prg13 ⊗ Prg23⊗ Prg33

(4.5,0.3,0.75)⊗(4,0.5,0,9), → Prg13 ⊗ Prg24⊗ Prg31

(4.5,0.3,0.75)⊗(2.5,0.3,0.7), → Prg13 ⊗ Prg24⊗ Prg33

(3.5,0.5,0.75)⊗(4,0.5,0,9), → Prg14 ⊗ Prg23⊗ Prg31

(3.5,0.5,0.75)⊗(2.5,0.3,0.7), → Prg14 ⊗ Prg23⊗ Prg33

(5,0.5,0.8)⊗(4,0.5,0,9) , → Prg14 ⊗ Prg24⊗ Prg31

(5,0.5,0.8)⊗(2.5,0.3,0.7)**}**= → Prg14 ⊗ Prg23⊗ Prg33

{(8,0.5,0.8) , (6.5,0.3,0.7), (8.5,0.5,0.82) , (7,0.3,0.72) , (7.5,0.5,0.82) , (6,0.5,0.72) , (9,0.5,0.85), (7.5,0.5,0.75)}

Après application de la relation de dominance : on aura (8, 0.5, 0.8), (7.5, 0.5, 0.75) , (8.5, 0.5, 0.82) des solutions dominées (par d'autres solutions).

Ce qui reste donc dans la liste {(6.5, 0.3, 0.7), (7, 0.3, 0.72) , (7.5, 0.5, 0.82) , (6, 0.5, 0.72) , (9, 0.5, 0.85)} qui correspondra respectivement aux solutions

{(Prg_{13}, Prg_{23}, Prg_{33}),

(Prg_{13}, Prg_{24}, Pg_{31}),

(Prg_{13}, Prg_{24}, Prg_{33}),

(Prg_{14}, Prg_{23}, Prg_{33}),

(Prg_{14}, Prg_{24}, Prg_{31})}

5 solutions possibles