SAÉ S1.02 : Comparaison d'approches algorithmiques – Année 23-24

Compétence : Optimiser des applications informatiques (Compétence 2)

- Apprentissage Critique (AC1) : Analyser un problème avec méthode (découpage en éléments algorithmiques simples, structures de données...)
- Apprentissage Critique (AC2) : Comparer des algorithmes pour des problèmes classiques (tris simples, recherche...)

Thème : Gestion de Projets (Méthode PERT)

Sujet : L'objet de la méthode PERT (Programm Evaluation and Review Technique) est la planification dans le temps d'un certain nombre de tâches liées par des contraintes de précédences. Grâce à la chronologie et l'interdépendance de chacune des tâches, on structure l'ensemble du projet et on peut alors planifier la réalisation de chacune des tâches les unes par rapport aux autres, afin de minimiser les délais, ainsi que réduire l'impact des retards lors de l'exécution des différentes tâches.

Dans cette SAE, nous allons illustrer les structures avec lesquelles vous devez travailler à travers un projet de construction immobilière (libre choix à vous de prendre d'autres projets dans votre SAE). Ainsi un constructeur (ou maître d'œuvre) doit délivrer un planning détaillé (en fonction des sous-traitants et entrepreneurs qui agissent l'un après l'autre et parfois simultanément sur le chantier) afin que le propriétaire puisse suivre l'évolution de la construction étape par étape.

Afin de finaliser la mise en œuvre de la méthode PERT, un certain nombre d'activités doivent être menées à bien :

Les données

- définir de manière précise le projet : dans notre cas la construction immobilière ;

- définir un responsable de projet auquel on rendra compte et qui prendra les décisions importantes : un des membres de l'équipe (l'architecte);

- analyser le projet et définir très précisément l'ensemble {Ti} des tâches auxquelles on s'intéresse (Structure du Bâtiment, Gros-Œuvres, Plomberie, Electricité, Menuiserie, Peinture, Finition, …);

- attribuer une durée à chaque tâche : dans notre cas, pour chaque tâche un appel d'offre est lancé afin de sélectionner l'entreprise qui va réaliser la tâche. Dans cette phase la durée de la tâche sera connue ;

- définir un ensemble de couples (Ti,Tj) (qui veut dire la tâche Ti doit être terminée pour que la tâche Tj puisse commencer) définissant ainsi la relation de précédence entre les différentes tâches.

Le résultat :

- chercher un arrangement séquentiel des tâches qui respecte les contraintes de précédence ;
- chercher, pour chaque tâche, une date de commencement au plus tôt, découlant de la durée des tâches qui doivent nécessairement être achevées avant que la tâche en question puisse commencer.
- détecter des éventuels bugs dans les données : par exemple, présence d'une boucle ou d'un circuit dans les relations de précédence.

Partie I : Contexte et codage des informations

Pour un projet immobilier, un cabinet d'architecture identifie les tâches à réaliser (Gros-Œuvre, Electricité, Plomberie, Peinture, Carrelage, Menuiserie, Finition, …) et édite dans un fichier « précédences.txt » les contraintes de précédence entre ces tâches.
Exemple :

Gros-Œuvre Electricité
Plomberie Peinture
Electricité Peinture
Plomberie Carrelage
…

précedences.txt

Par exemple : la première ligne de ce fichier se lit : « la tâche électricité ne peut commencer qu'après que la tâche Gros-Œuvre soit terminée ».
Ensuite, le cabinet d'architecture lance plusieurs appels d'offre (un appel d'offre par tâche identifiée) afin de réaliser son projet. Pour chaque tâche, le cabinet reçoit plusieurs offres d'entreprises différentes (devis). Chaque devis, présente le nom de l'entreprise, son adresse, son capital, la durée en jours pour réaliser la tâche ainsi que le coût total pour réaliser la tâche.
Toutes les informations concernant toutes les entreprises doivent être stockées dans un fichier (binaire ou texte) :
Exemple :

Plomberie //Nom de la tâche
Pros De L'eau // nom de l'entreprise
12, Av. la république – 63000 – Clermont-Ferrand // adresse de l'entreprise
500 000 // capital en euros
35 // durée en jours pour réaliser la tâche
25000 // coût de la tâche
} champs du type Devis
Electricité
Electro Maître
50, zone industrielle sud – 15000 – Aurillac
450000
60
42000
Plomberie
DolcéAqua
33, zone Industriel Aubière -63170 - Aubière
450 000
40
27000
Carrelage
…

Ce fichier doit être chargé dans des structures comme suit.

Tous les travaux à réaliser seront enregistrés au moyen d'un tableau *tabTravaux* de pointeurs vers des structures de type *Offre* (cf Figure 1).

Une structure de type *Offre* est composée d'une chaîne de caractères *travaux* (30 caractères) et d'une liste de devis *ldevis* (du type *ListeDevis).*

Les devis des entreprises ayant répondu à une offre sont enregistrés dans une liste chaînée *ldevis.* Ils se présenteront dans l'ordre croissant des noms des entreprises.

Une variable de type *ListeDevis* est un pointeur vers un maillon de type *MaillonDevis*. Ce pointeur a la valeur NULL si la liste est vide
Une structure de type *MaillonDevis* est composée d'une structure de type Devis (voir fichier des devis ci-dessus) et d'un pointeur *suiv* vers le maillon suivant. Le pointeur *suiv* du dernier maillon de la liste contient l'adresse fictive NULL.

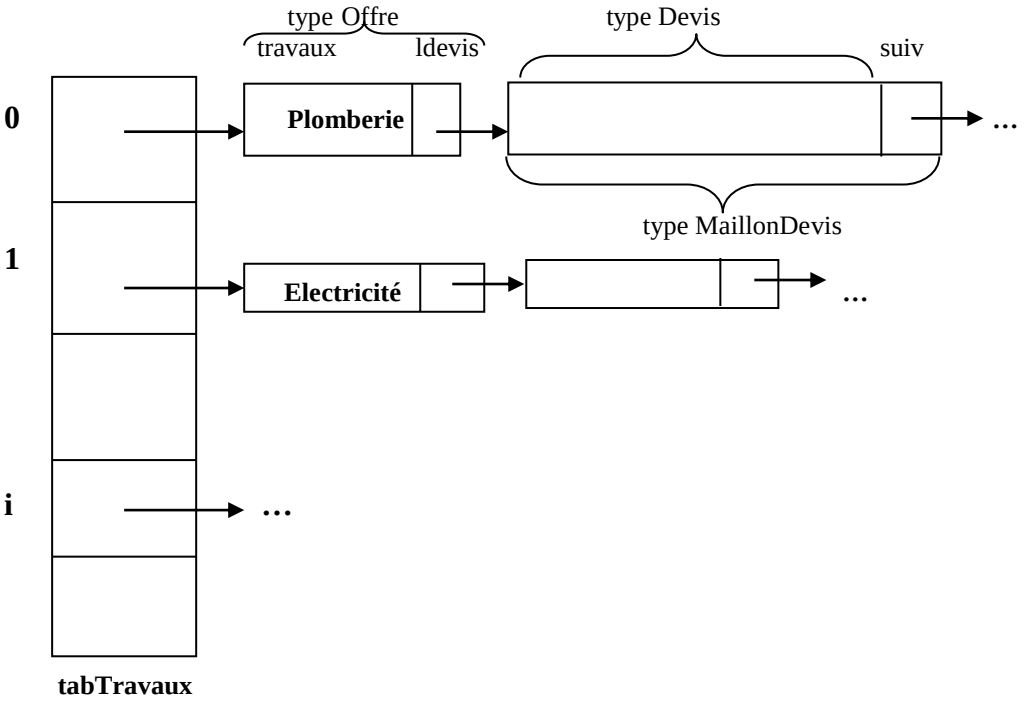

Figure 1

A l'issue de cette phase, votre application devrait permettre de :

- Afficher l'ensemble des devis pour un type de travaux ;
- Afficher le devis d'une entreprise donnée pour un type de travaux donnée
- …

Partie II : Sélection des entreprises retenues

A partir de la structure (Figure 1), une seule entreprise par tâche est retenue : celle ayant le coût de réalisation le moins cher ; et en cas d'égalité c'est l'entreprise ayant un gros capital qui est retenue. Toutes les autres entreprises seront supprimées de la structure.

Dans cette partie, l'affichage des entreprises retenues est demandé.

Partie III : Nouvelle structure

Soit la structure suivante :

```
typedef struct
{
        char tache[20] ; // nom de la tâche ex : Plomberie
        int duree ; // durée de la tâche
        int nbPred ; // nombre de prédécesseurs de la tâche
        Liste succ ; // liste des successeurs de la tâche
        int dateDebut ; // date de début au plutôt de la tâche
        bool traite ; // booléen : la tâche est-elle traitée
}Tache ;
```

Vous devez à partir de la structure correspondante à la figure obtenue dans la partie II, et du fichier pécédences.txt, charger un tableau de pointeurs vers des structures de type Tache.
Le nom de la tâche et la durée sont à récupérer de la figure 1, le nombre de prédécesseurs ainsi que la liste des noms de successeurs sont à calculer à partir du fichier précédences.txt.
La date de début de chaque tâche ainsi que le booléen traite sont initialisés à 0.

Partie IV : Calcul de la date de début de chaque tâche et affichage des résultats

Pour calculer la date de début de chaque tâche voici l'algorithme imposé à implémenter.
La méthode proposée utilise une file d'attente représentant les tâches qui peuvent être commencées mais ne sont pas encore traitées. Remarque : d'autres structures peuvent bien convenir, mais ici la structure de file d'attente est imposée.

Algorithme :

- Initialisation : placer dans la file d'attente toutes les tâches sans prédécesseur. Leur date de commencement au plus tôt est l'instant 0. Si la file d'attente est vide après cette initialisation c'est qu'il y a présence d'une anomalie dans les contraintes de précédences (boucle et/ou circuit).

- Etape courante (utilisation d'une boucle)
  - ➔ sortir une tâche Ti de la file d'attente. Sa date de commencement au plus tôt n'est autre que sa date actuelle.
  - ➔ Pour chaque tâche Tj de l'ensemble des successeurs de Ti, faire :
    date de Tj = max(date actuelle de Tj , date actuelle de Tj + durée de Ti) ;
    nombre de prédécesseurs de Tj = (nombre de prédécesseurs de Tj) – 1 ;
    si (nombre de prédécesseurs de Tj) = 0, entrer Tj dans la file d'attente.

- Quand la file d'attente est vide :
  - ➔ **Si** toutes les tâches ont été exécutées, arrêter l'algorithme (fin).
  - ➔ **Sinon**, c'est qu'il y a une boucle dans les relations de précédence. Annoncer cette anomalie, stopper le programme et revoir vos contraintes.

Après l'application de cet algorithme, vous devez pouvoir :

- afficher par ordre d'exécution chaque tâche Ti et sa date de commencement au plus tôt (donc tri du tableau selon le critère date de début).
- calculer la durée de réalisation du projet.
- Lister les tâches qui restent à réaliser à une date donnée.

Livrables :

À l'issue de cette SAÉ les différents livrables que vous devez fournir sont :

- le code de l'application avec sa documentation
- un compte-rendu dans lequel partie par partie vous :
    - expliquez votre projet si vous choisissez un autre projet autre qu'une construction immobilière ;
    - explicitez les fonctionnalités réalisées dans votre logiciel en deux lignes maximum chacune;
    - comparez les différents algorithmes de tris ainsi que les différents algorithmes de recherche que vous avez mis en œuvre pour coder chaque fonction de tri ou de recherche nécessaire à votre logiciel, de différentes façons.

Évaluation :

Vous devez développer d'une façon collaborative.
Seront évalués vos livrables : le code, la documentation et le compte-rendu. Concernant le code, seront évaluées sa qualité et sa complexité, à savoir si vous avez ou pas utilisé les différentes notions vues en structures de données :

- fichier texte et ou binaire (chargement/sauvegarde)
- algorithmes de tri
- algorithmes de recherche
- saisie contrôlée
- récursivité
- menu

Cette évaluation sera complétée par un oral.

Date de rendu : le vendredi 12/01/2024