

Compte rendu SAE 1.02
-
Comparaison d'approches algorithmiques

Choix du projet

Nous avons choisi de garder le projet proposé pour cette SAE car celui-ci nous plaisait et on voulait pouvoir l'aborder pour répondre au mieux à la demande.

Fonctionnalités

Dans notre projet, nous avons mis en place plusieurs fonctionnalités :

- Affichage : Cette fonction nous permet d'afficher les différents affichages nécessaires dans notre projet avec plus ou moins de détail en fonction du besoin.
- Chargement : Cette fonction permet le chargement des tableaux ainsi que la lecture et l'insertion de nouveaux devis.
- Précédences : Permet de regarder, d'ajouter et de charger des tâches pour connaître leurs durées.
- Recherches dichotomiques : Recherche spécifique au sein de notre projet avec une particularité pour les tâches
- Tris : Tris dans les différentes parties de notre projet.
- Free : libère certaine structure pour une meilleure optimisation
- Sauvegarde : enregistrement de certain élément
- File : Créer une file

Comparaison

Au cours de notre projet, nous avons utilisé deux méthodes de tri : le tri par échange et le tri par insertion.

Le choix du tri par insertion pour la liste des tâches a été fait pour sa simplicité d'implémentation et son efficacité pour des petites listes ou des listes partiellement voire non triées. Il peut être suffisamment rapide pour des situations de petite échelle, comme une liste de Devis (qui en théorie ne devrait pas dépasser les 10 à 20 par Offre). Ce choix a été aussi fait pour la clarté du code et la facilité de compréhension. L'usage pour trier par le nom de l'entreprise dès son insertion dans le chargement, faisait de cet algorithme un des plus préférables. Si la liste était significativement plus grande, un autre algorithme de tri plus efficace aurait été envisagé.

Quant au tri par échange pour le tri des devis par leur coût est également justifié par sa simplicité avec son usage assez simple, il s'agit d'un algorithme des plus connu pour le tri. Il est simple à mettre en œuvre et convient bien pour des petites quantités de données comme le tri par insertion. Il est également lisible et léger car il est factorisé et découpé en plusieurs fonctions d'échange et de comparaison.

Dans ce projet, la priorité a été accordée à la compréhension avec des algorithmes assez légers en exécution sans récursivité, tout en maintenant une performance adéquate au vu de la quantité de données.

=====

La **récursivité** a été utilisée pour l'affichage et certains parcours de liste/file dans le projet. Son utilisation a été limitée pour éviter les coûts de performances ou possibles débordements de la pile mémoire. Dans certains cas des boucles itératives s'avéraient préférables en termes de complexité ou que la différence d'optimisation était très faible.