SAE S1.01-02

Initiation au développement

Bilal EL AYOUBI
Lucas BENOIT

Sommaire

1. **Description et justification des algorithmes**
    1.1 Fonction de recherche
    1.2 Fonction de tri
    1.3 Fonction de filtre

2. **Complexité théorique des algorithmes**
    2.1 Fonction de recherche
    2.2 Fonction de tri
    2.3 Fonction de filtre

3. **Évaluation expérimentale de la complexité**

Bilal EL AYOUBI
Lucas BENOIT

Description et justification des algorithmes

1/ Fonction de recherche

```
void afficher_selon_num() {
   int num;
   printf("Saisir numero de bus souhaite: ");
   scanf("%d", &num);

   for (int i = 0; i < MAX_BUS; i++) {
      if (b[i].numBus == num) {
         ...
         return;
      }
   }
   printf("Pas de bus trouvé avec ce numéro\n");
}
```

La fonction sert à rechercher un trajet selon le numéro de bus saisi par l'utilisateur.
L'algorithme utilisé est une **recherche linéaire**, qui parcourt tous les éléments du tableau jusqu'à avoir trouvé l'élément souhaité.

Ce choix a été fait car les données ne sont pas triées par numéro de bus. Mettre en place un tri pour utiliser une recherche dichotomique aurait ajouté plus d'opération qui serait utile seulement pour cette fonction.

Complexité théorique : O(n)
Dans le pire des cas, la fonction parcourt l'ensemble des trajets.

---

2) Fonction de tri

```
void trier_par_ville_et_date() {
```

```
   int cmp;
   for (int i = 0; i < MAX_BUS - 1; i++) {
      for (int j = 0; j < MAX_BUS - i - 1; j++) {
         cmp = strcmp(b[j].villeDepart, b[j + 1].villeDepart);
         if (cmp > 0
            || b[j].d.a > b[j+1].d.a
            || b[j].d.a == b[j+1].d.a && b[j].d.m > b[j+1].d.m
            || b[j].d.a == b[j+1].d.a && b[j].d.m == b[j+1].d.m && b[j].d.j >
b[j+1].d.j
         ) {
            Bus temp = b[j];
            b[j] = b[j + 1];
            b[j + 1] = temp;
         }
      }
   }
}
```

Cette fonction trie les trajets par **ville de départ**, puis par **date** (année, mois, jour).
L'algorithme utilisé est le **tri à bulles**, qui change élément par élément si nécessaire.

Ce choix a été fait car il s'agit d'un algorithme simple, vu en cours, facile à implémenter et suffisant.

Complexité théorique : O(n²)
Deux boucles imbriquées parcourent l'ensemble du tableau de bus.

---

3) Fonction de filtre

```
void filtre_ville_date_lendemain(){
   char villedep[MAX_CARAC];
```

```
    int j,m,a;
    // demande utilisateur
    for(int i = 0; i < MAX_BUS; i++){
        if (strcmp(villedep, b[i].villeDepart) == 0
            && j == b[i].d.j && m == b[i].d.m && a == b[i].d.a
            && b[i].horaireArrivee < b[i].horaireDepart
            && b[i].horaireDepart >= 0 && b[i].horaireArrivee >= 0)
        {
            // affichage
        }
    }
}
```

Cette fonction filtre les trajets selon trois critères :

- ville de départ,
- date de départ,
- arrivée le lendemain (*horaire d'arrivée inférieur à l'horaire de départ*).

L'algorithme parcourt l'ensemble des trajets et teste chaque condition avant d'afficher les trajets que l'utilisateur a demandé.

Complexité théorique : O(n)
Chaque trajet est testé une seule fois. Le nombre de comparaisons dans la condition est constant et ne modifie pas l'ordre de complexité.

Évaluation expérimentale de la complexité

Fichier de 10 trajets

- Recherche : **10 opérations**
- Tri : **417 opérations**
- Filtre : **80 opérations**

Fichier de 50 trajets

- Recherche : **50 opérations**
- Tri : **11 807 opérations**
- Filtre : **400 opérations**

Fichier de 100 trajets

- Recherche : **100 opérations**
- Tri : **47 529 opérations**
- Filtre : **800 opérations**

Fichier de 500 trajets

- Recherche : **500 opérations**
- Tri : **1 228 277 opérations**
- Filtre : **4 000 opérations**

Fichier de 1000 trajets

- Recherche : **1000 opérations**
- Tri : **4 933 284 opérations**
- Filtre : **8 000 opérations**

Fichier de 5000 trajets

- Recherche : **5000 opérations**
- Tri : **123 969 242 opérations**
- Filtre : **40 000 opérations**

Analyse des résultats

Les mesures expérimentales confirment les complexités théoriques attendues :

- La **recherche** et le **filtre** présentent une évolution linéaire du nombre d'opérations en fonction du nombre de trajets : **Complexité O(n)**.

- Le **tri** présente une croissance quadratique : lorsque le nombre de trajets est multiplié par 10, le nombre d'opérations est approximativement multiplié par 100.
  Complexité $O(n^2)$, caractéristique du tri à bulles.

Ces résultats montrent la cohérence entre la théorie et l'expérimentation mais n'empêche que les résultats restent énormes et qu' une optimisation serait possible.