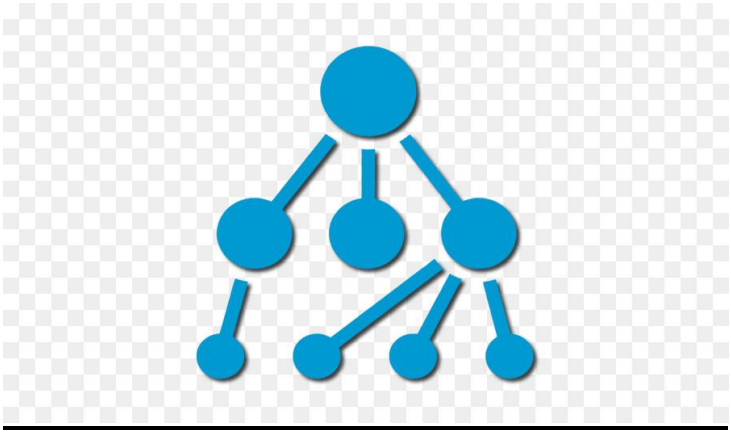
RAPPORT PROJET :

« Algorithme de recherche »

ALGORITHME AVANCÉ



ENCADRÉ PAR:

- Monsieur Imad HAFIDI

REALISÉ PAR:

- Yasser SOKRI
- Abdellah OUBADOU

Table de matière :

Introduction : Présentation du problématique

- En informatique, un algorithme de recherche est un type d'algorithme qui, pour un domaine, un problème de ce domaine et des critères donnés, retourne en résultat un ensemble de solutions répondant au problème. Supposons que l'ensemble de ses entrées soit divisible en sous-ensemble, par rapport à un critère donné, qui peut être, par exemple, une relation d'ordre. De façon générale, un tel algorithme vérifie un certain nombre de ces entrées et retourne en sortie une ou plusieurs des entrées visées.
- D'autre part, a popularité croissante des applications web qui stockent et analysent de grandes quantités de données telles que Facebook, Twitter et Google, a posé de nouvelles exigences qui défient énormément les systèmes de gestion de bases de données. Les données web sont de plus en plus stockées sous formats de fichier Json. Un fichier Json est un fichier texte qui permis de stocker des données textuelles. Un fichier Json contient plusieurs objets.

Présentation Json.

- Le format .Json provenant du monde JavaScript et représentant un objet, s'apparente au .XML, dans sa fonction du moins : il permet de stocker des données textuelles. Les signes de ponctuations qui permettent de structurer les données dans un fichier .json :
  - ✓ les accolades définissent un objet.
  - ✓ Les guillemets (double-quotes) et les double-points définissent un couple clé/valeur.
  - ✓ Les crochets définissent un tableau.
  - ✓ Les virgules permettent de séparer les membres d'un tableau ou d'un objet .
- Le principal avantage de JSON est qu'il est simple à mettre en œuvre par un développeur tout en étant complet. Au rang des avantages, on peut également citer :
  - ✓ peu verbeux, ce qui le rend lisible aussi bien par un humain que par une machine ;
  - ✓ facile à apprendre, car sa syntaxe est réduite et non extensible (bien qu'il souffre de quelques limitations) ;
  - ✓ ses types de données sont connus et simples à décrire.
- Le JSON ne peut représenter que quelques types généraux, et il n'est pas possible d'en ajouter d'autres. Pour les dates ou les couleurs par exemple, il faut trouver des représentations sous forme de chaînes de caractères. C'est sa principale différence avec un langage comme le XML, où les données sont typées et extensibles, au prix d'une plus grande complexité.
- Ce typage faible affaiblit la sécurité et la fiabilité du langage ; par exemple, il n'y a pas de limite fixe pour les valeurs des entiers, celle-ci dépend de l'interpréteur.
- D'un point de vue plus pratique, il ne peut pas y avoir de commentaires, ce qui est gênant quand on utilise des fichiers JSON dans une configuration. Certaines bibliothèques acceptent les commentaires au format JavaScript.

Présentation de la distance Levenshtein

- La distance de Levenshtein est une distance, au sens mathématique du terme, donnant une mesure de la différence entre deux chaînes de caractères. Elle est égale au nombre minimal de caractères qu'il faut supprimer, insérer ou remplacer pour passer d'une chaîne à l'autre.
- Elle a été proposée par Vladimir Levenshtein en 1965. Elle est également connue sous les noms de distance d'édition ou de déformation dynamique temporelle, notamment en reconnaissance de formes et particulièrement en reconnaissance vocale.
- Cette distance est d'autant plus grande que le nombre de différences entre les deux chaînes est grand. La distance de Levenshtein peut être considérée comme une généralisation de la distance de Hamming. On peut montrer en particulier que la distance de Hamming est un majorant de la distance de Levenshtein.

- ***Algorithme méthode de levenshtein :***

```
int levenshtein(char s[], char t[])
{
        int ls = strlen(s);
        int lt = strlen(t);
        int d[ls+1][lt+1];
        int i ,j;
        for (i= 0; i <= ls; i++){
            for (j=0; j <= lt; j++) {
            d[i][j] = -1;
            }
        }
        int dist(int i, int j) {
            if(d[i][j]>=0){
                return d[i][j];
            }
            int x;
            if (i == ls)
            x = lt - j;
            else if (j == lt)
            x = ls - i;
            else if (s[i] == t[j])
            x = dist(i + 1, j + 1);
            else {
                x=dist(i+1,j+1);
                int y;
                if ((y = dist(i, j + 1)) < x)
                        x = y;
                if ((y = dist(i + 1, j)) < x)
                        x = y;
                x++;
            }
            return d[i][j] = x;
        }
return dist(0, 0);
}
```

Les fonctions principales :

- *comparaison(char s[], char t[]) :*
  - C'est une fonction de type entier qui prend en entré deux chaines de caractères et il fait la comparaison entre ces deux chaines.

- *struct Personne :*
  - c'est une structure de qui contient cinq champs (nom, prenom, adresse_email, ville, pays) .

- *Personne initPersonne(char json_objet[]) :*
  - C'est une fonction dans laquelle on sépare les champs d'un objet JSON, c'est-à-dire la séparation entre la clé et la valeur, ensuite on met ces clés dans une structure de type personne.

- *long taille(char *adress) :*
  - il prend en entré le nom de fichier et il calcule sa taille.

- *lower_string(char s[]) :*
  - Cette fonction transforme les lettres majuscules en minuscule (exemple : A->a).

- *char **lire (char file_name[]) :*
  - Il prend en entré le nom de fichier saisi par l'utilisateur et renvoie une liste de chaine de caractère (chaque chaine représente un objet Json).

- nbr_obj(char file name[]) :
  - Il calcule le nombre d'objet Json en entrant le nom du fichier.

- researche_function :
  - Cette fonction fait la recherche dans la base Json.

La complexité des fonctions principales :

- La distance de livenshtein **:**

  Sa complexité est quadratique O(ls*lt) ,car on deux boucle for imbriqué.

- int comparaison(char s[], char t[]) :

  Sa complexité est d'ordre 1 O(1).

- Personne initPersonne(char json_objet[]) :

  Sa complexité est linéaire O(nombres d'objets) .

- lower_string(char s[]) :

  Sa complexité est linéaire O(n).

- researche_function(int nb_obj,char **ourlist ,char nom[],,char prenom[],char ville[],char pays[]) :

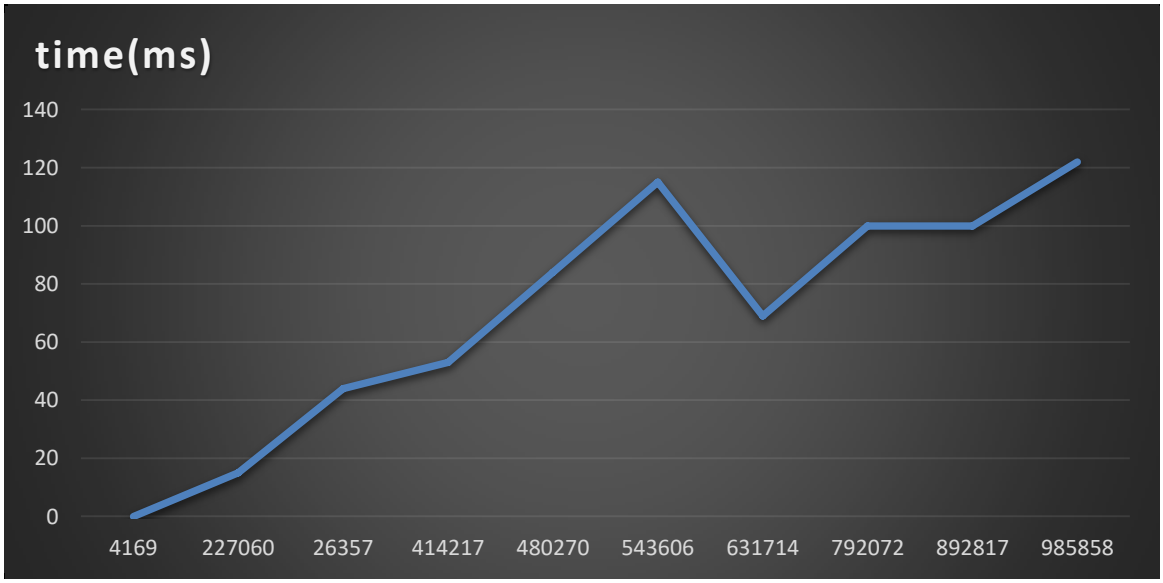
  Sa complexité est d'ordre n O(n) car il contient une seule boucle for qui parcours le fichier.

Graphes d'évolution du temps d'exécution :

❖ Nous avons tracé les graphes qui représentent l'évolution du temps en (milliseconde) du chargement des données JSON en fonction de leurs tailles en (bytes) ainsi que l'évolution du temps de l'exécution (milliseconde) de la requête en fonction de la taille du fichier JSON (bytes) en utilisant le logiciel Microsoft Excel:

- **Recherche : le temps en fonction de la taille :**

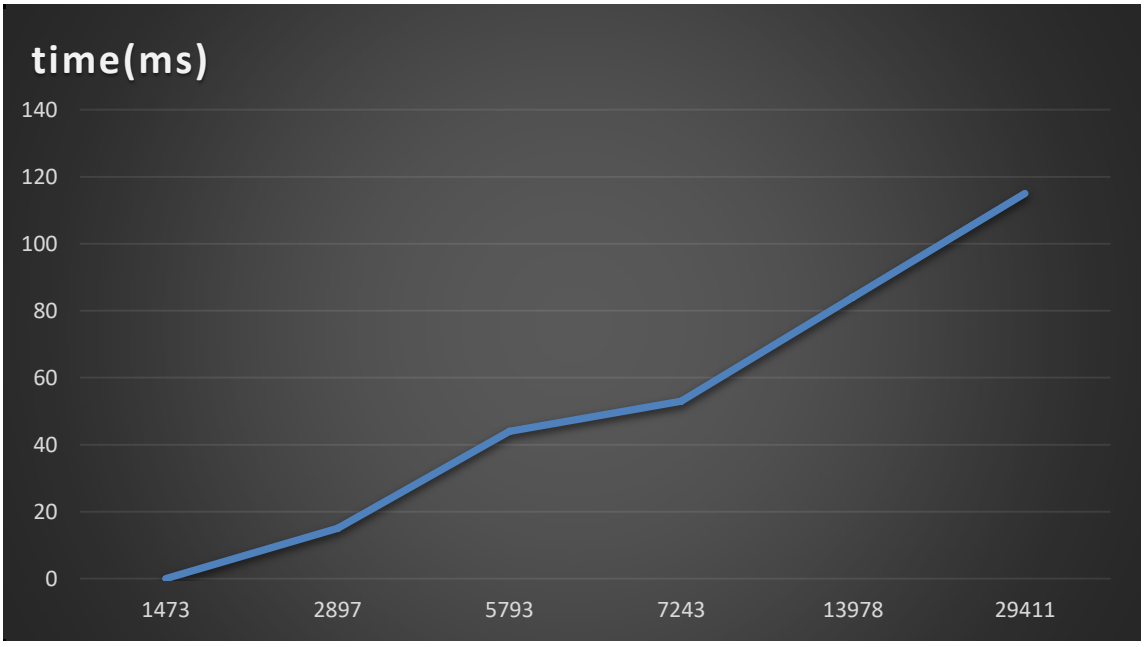
Time (ms)	Size (byte)	Nombre d'objets
0	4169	27
16	227060	1474
22	26357	1711
37	414217	2689
47	480270	2988
53	543606	3529
69	631714	4101
100	792072	5142
100	892817	5796
122	985858	6193



Graphe d'évolution du temps en fonction de la taille

Chargement des données : le temps en fonction de la taille :

Time (ms)	Size (byte)
0	1473
15	2897
44	5793
53	7243
84	13978
115	29411



Graphe d'évolution du temps en fonction de la taille

Conclusion :

Grâce à ce projet nous avons acquérir plusieurs méthodes et techniques pour la construction d'un algorithme de recherche, du fait qu'il représente un outil très pertinent au niveau de plusieurs projets informatiques.

De surcroit, nous pouvons dire que le fichier JSON se caractérise par la vitesse de traitement des données, également, le fichier JSON est un outil efficace pour la recherche grâce à sa structure, il est donc devenu très populaire pour le stockage, la lecture…

Finalement nous vous remercions que vous avez nous donné la chance d'améliorer nos compétence à travers ce projet.