921

NOM : PINAULT **Prénom :** Laureline **Jury :**

Algèbre ← Entourez l'épreuve → **Analyse**

Sujet choisi : 921. Algorithmes de recherche et structures de données associées

Autre sujet :

I Cadre de l'étude

1 Motivations

- Quelques exemples de la vie de tous les jours :
  - recherche dans un annuaire (pages jaunes)
  - recherche d'un mot dans un texte (ctrl-F, google)
  - recherche d'un dossier (explorateur de fichiers)
  - recherche d'une adresse (google maps)
  - recherche de la prochaine tâche à effectuer (processeur)
  - suggestions (recherche dans un graphe biparti)
- Ces exemples sont variés. Ils concernent :
  - des types de données variés
  - des critères de recherche variés
  - des types de structures statiques ou dynamiques

2 Ce qu'on va étudier

- La recherche d'un élément identifié (on teste une égalité entre éléments en $O(1)$) → II
- La recherche d'un élément selon certains critères dans un ensemble ordonné (minimum et maximum) → III
- La recherche d'un mot dans un texte → IV

II Recherche d'un élément identifié

1 Dans un ensemble non ordonné : recherche linéaire / séquentielle

Idée 1. Pour trouver un élément donné il suffit de parcourir tous les éléments séquentiellement.

Exple 2. Dans une liste, comme le temps d'accès à un élément est linéaire, ce sera l'algorithme utilisé.

Exple 3. Pour trouver un sommet dans un graphe, on peut parcourir le graphe. Cela amène à développer des algorithmes de parcours :
- parcours en profondeur
- parcours en largeur

Rq 4. On s'intéresse dorénavant à des structures dont le temps d'accès est moins que linéaire

2 Dans un ensemble ordonné statique : lien avec le tri

Exple 5. C'est ce qu'on va vouloir faire quand on cherche dans un annuaire ou un dictionnaire (peu de changements par rapport au nombre de recherche).

Algo 6 (recherche dichotomique)

ENTRÉE : T (tableau trié) min, max (bornes du tableau) x (élément à chercher)

SORTIE : i tel que $T[i] = x$ (-1 si $x \notin T$)

ALGO : si $min \leq max$, $i := \left\lfloor \frac{min + max}{2} \right\rfloor$

si $T[i] = x$ renvoyer i 2
si $T[i] < x$ ALGO($T, min, i-1, x$) 3
sinon ALGO($T, i+1, max, x$) 4
sinon renvoyer -1

Prop 7 Si n est le nombre de données, l'algo 6 nécessite un prétraitement en $O(n \log n)$ (tri fusion, tri rapide) en complexité pire cas et/ou espérée (distribution uniforme) et se fait en temps $O(\log n)$. (temps)

Rq 8 Si l'entrée est des entiers entre k et $k+r$, le prétraitement peut se faire en $O(r)$ (tri seau ou tri par comptage).

Algo 9 (recherche par interpolation)
IDÉE : si on cherche un mot dans le dictionnaire on va plutôt le chercher au début qu'à la fin ou même au milieu.
ENTRÉE : T (tableau de nombres triés), min, max (bornes du tableau), x (élt cherché)
SORTIE : i tel que $T(i) = x$ (-1 si $x \notin T$)
ALGO : comme la recherche dichotomique mais à la ligne 1 on a $i := min + \left\lfloor \frac{(max-min)(x-T[min])}{T[max]-T[min]} \right\rfloor$

Prop 10 Le pré-traitement a la même complexité que la dichotomie mais sous hypothèse de distribution uniforme de l'entrée la recherche se fait en temps $O(\log \log n)$

Rq 11 On peut encoder des mots par des nombres pour appliquer cet algorithme à la recherche dans un dictionnaire par exemple. Pour ce faire on peut associer à chaque lettre un nombre entre 0 et $k-1$ où k est la taille de l'alphabet puis à un mot $a_0 \dots a_n$ on associe $\sum a_i \times k^i$

3 Dans un ensemble ordonné dynamique : maintien d'une structure triée (Dictionnaire)

Def 12 (Arbre binaire de recherche). Un arbre binaire de recherche est un arbre binaire tel que pour tout noeud, ~~l'ensemble~~ les noeuds du sous arbre droit lui sont supérieurs et ceux du sous arbres gauche lui sont inférieurs ~~(voir exemple en annexe)~~.

Prop 13 Si on note h la hauteur d'un arbre binaire de recherche, la recherche, l'insertion et la suppression d'un élément se font en $O(h)$

Idée 14 On va vouloir des arbres "équilibrés" (dont la hauteur est en $O(\log n)$ où n est le nombre de noeud). Cela impose une structure supplémentaire que l'insertion et la suppression doivent conserver tout en restant pas trop coûteux.

Exple 15 (Arbre rouge noir) Un arbre rouge noir est un arbre ~~étant~~ binaire de recherche dont tous les noeuds sont colorés en rouge ou en noir et respectant les conditions suivantes
- la racine est noire
- les feuilles sont noires et ne contiennent pas d'information
- si un noeud est rouge, ses fils sont noirs.

Dans un tel arbre, les opérations d'insertion, de suppression et de recherche se font en $O(\log n)$ en temps où n est le nombre de noeud.

Exple 16 (Arbre ZigZag) Un arbre ZigZag est un arbre binaire de recherche vérifiant la propriété : pour tout noeud, la hauteur du sous arbre droit est égale à la hauteur du sous arbre gauche ± 1.
Dans un tel arbre, les opération d'insertion, de suppression et de recherche se font en temps $O(\log n)$.

Idée 17 (Arbre de van Emde Boas) Si on a des entiers entre k et $k+r$ alors en s'inspirant du tri par seau on peut construire une structure de donnée qui effectue les opération de recherche, d'insertion et de suppression en $O(\log \log r)$.

DVT 1

III Recherche du minimum ou maximum

1 Dans un ensemble statique

Prop 18 Grâce à un tableau trié (pré-traitement en temps $O(n \log n)$ ou $O(n)$, voir II, 2), on peut trouver le minimum et le maximum en temps $O(1)$.

2 Dans un ensemble dynamique (File de priorité)

Exple 19. (C'est ce qu'on va vouloir faire pour gérer la liste des processus, chacun avec une priorité, qui change tout le temps)

Def 20 (Tas min/max). Un tas min (resp max) est un arbre binaire parfait (tous les étages sont

remplis sauf éventuellement le dernier qui est rempli de la gauche vers la droite) qui vérifie la propriété : tout noeud est inférieur ou égal à chacun de ses fils (resp supérieur ou égal). ~~(voir exemple en annexe)~~

Prop21. Dans un tas min (resp max), on peut trouver le minimum (resp. le maximum) en temps $O(1)$ tandis que l'ajout, la suppression et la mise à jour se font en temps $O(\log n)$.

Rq22. Comme un tas est un arbre binaire parfait, on peut l'implémenter avec un tableau ~~(voir annexe)~~

IV Recherche d'un mot dans un texte

But23. On veut rechercher les occurrences d'un motif (pattern) de taille m dans un texte de taille n.

Prop24. L'algorithme naïf réflechie en temps $O(n \times m)$ ($(n-m) \times m$ opérations).

OVT 2

Algo25. (Knuth-Morris-Pratt). On peut améliorer l'algorithme naïf pour avoir une complexité en temps $O(m+n)$.

Pb26. Parfois on veut chercher un mot dans un texte on veut permettre quelques erreurs.

Exple27. • Lorsqu'on cherche un gène dans une séquences d'ADN on veut permettre les petites modification liées aux mutations.

• Google permet des fautes de frappe durant une recherche.

Def28. La distance d'édition entre deux mots est le nombre minimum d'opérations parmi (changement d'une lettre, suppression d'une lettre, ajout d'une lettre) à appliquer à un mot pour obtenir l'autre.

Algo29. L'algorithme de programmation dynamique suivant permet de calculer la distance d'édition de deux mots x et y en $O(|x||y|)$ en temps :

ENTREE : x, y (les deux mots)
SORTIE : La distance d'édition entre x et y
ALGO :

```
pour i de 0 à |x|   EDIT[i,0] = i
pour j de 0 à |y|   EDIT[0,j] = j
pour i de 0 à |x|
    pour j de 0 à |y|
        EDIT(i,j) = min(EDIT[i-1,j]+1,
            EDIT(i,j-1)+1, EDIT[i-1,j-1]+δ(x[i],y[j]))
renvoyer EDIT[|x|,|y|].
```

Prop30. En adaptant un peu l'algorithme précédent on peut rechercher la présence d'un motif dans un texte en permettant des erreurs en $O(mn)$.

V Ouvertures

1 Des données plus complexes

Pour traiter des données comportant plusieurs champs on utilise des bases de données.

2 Chercher selon des critères complexes

Rechercher selon certains critères des optimum est un vaste domaine et il existe des algorithmes pour répondre à ce besoin.

Par exemple, quand les contraintes sont linéaires on peut utiliser l'algorithme du simplexe.

3 Beaucoup de données

Quand les données sont très grandes et ne tiennent pas en mémoire vive, l'opération prépondérante (en terme de coût) est l'accès mémoire. Il y a alors des structures adaptées (B-trees).

4 Informatique quantique

En informatique quantique on peut résoudre le problème de l'annuaire inversé parmi N éléments en $O(\sqrt{N})$ temps. (Algorithme de Grover).