NOM : ROUHLING Prénom : Damien Jury :

Algèbre ← Entourez l'épreuve → Analyse

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

Autre sujet : Réf : Froidevaux, Cormen, Crochemore, DNR

I - Recherche d'un élément dans une structure linéaire

1 - Structure non triée

Déf 1 : Une liste est la donnée d'une fonction Suivant, et d'une tête

Ex 1 : $\boxed{1|4|3}$ est donnée par tête = 1 Suivant = $\begin{cases} 1 \mapsto 4 \\ 3 \mapsto Nil \\ 4 \mapsto 3 \end{cases}$

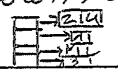
Prop 1 : La recherche d'un élément dans une liste de taille n est réalisée en temps $O(n)$ comme suit :
Entrée : t (tête), Suivant, x (à trouver)
Sortie : Sous-liste de tête x si elle existe, Nil sinon
Algo : Si t = Nil alors Nil
Sinon Si $t = x$ alors $(t$, Suivant$)$
Sinon Rechercher(Suivant(t), Suivant, x)

Rq 1 : En notant q_i la probabilité que x soit le iè élément et p celle que x n'y soit pas, l'algorithme a pour complexité moyenne $\sum_{i=1}^{n} iq_i + (n+1)p$, minimale quand $q_1 \geq q_2 \geq \dots \geq q_n \geq p$.
Cela suggère des méthodes adaptatives de recherche :
- après chaque recherche, placer l'élément cherché en tête
- après chaque recherche, échanger l'élément cherché avec son antécédent.

Déf 2 : Un tableau de taille n est une fonction de domaine $[\![1, n]\!]$.

Prop 2 : La recherche dans un tableau est en $O(n)$.

Déf 3 : Une table de hachage est un tableau de listes associé à une fonction clef donnant l'indice de la case où insérer/rechercher l'élément

Ex 2 :  clef = $\begin{cases} 1 \mapsto 2 \\ 2 \mapsto 1 \\ 3 \mapsto 4 \\ 4 \mapsto 1 \end{cases}$

Thm 1 : Dans le pire cas, la recherche dans une table de hachage est en $O(\max\{\text{tailles des listes}\})$
Si clef est uniforme, la complexité moyenne pour une table de m listes contenant n éléments :
- d'une recherche négative est $\frac{n}{m}$
- d'une recherche positive est $\frac{n-1}{2m} + 1$

2 - Structure triée

On se place dans un tableau trié

Thm 2 : L'algorithme de dichotomie suivant réalise la recherche en temps $O(\log(n))$:
Entrée : x, t (tableau), g et d (bornes)
Sortie : $i \in [\![g, d]\!]$ tel que $t(i) = x$, O sinon
Algo : Si $g \leq d$ alors
$m \leftarrow \lfloor \frac{g+d}{2} \rfloor$
Si $t(x) = m$ alors retourner m
Sinon Si $x < t(m)$ alors dicho$(x, t, g, m-1)$
Sinon dicho$(x, t, m+1, d)$
Sinon retourner O

Ex 3 : Figure 1

Thm 3 : En terme de complexité dans le pire cas, dicho est optimal parmi les algorithmes de recherche par comparaison

Thm 4 : L'algorithme de recherche par interpolation suivant réalise la recherche avec complexité moyenne $O(\log \log n)$:
Entrée : x, t, g, d
Sortie : $i \in [\![g, d]\!]$ tel que $t(i) = x$, O sinon
Algo : comme dicho mais avec $m \leftarrow g + \frac{(d-g)(x - t(g))}{t(d) - t(g)}$

Ex 4 : Figure 2

II_ Recherche d'un élément dans une structure non linéaire

1_ Graphes

Déf 4: Un graphe est la donnée d'un ensemble S de sommets et d'une fonction Voisin : $S \to$ Liste(S)

Ex 5:  est donné par $S = [1,3]$, Voisin $= \begin{cases} 1 \mapsto \boxed{2\,|\,3} \\ 2 \mapsto \boxed{1\,|\,3} \\ 3 \mapsto \boxed{1\,|\,2} \end{cases}$

Rq 2: Sans plus d'hypothèses sur la structure du graphe, vérifier la présence d'un élément dans un graphe se fait par parcours :

Entrée : x, S, Voisin, s (sommet de départ)

Sortie : vrai si x appartient au graphe

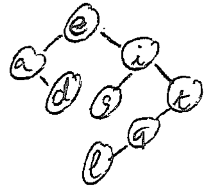
```
Algo : pour tout u ∈ S\{s}
           couleur(u) ← blanc
       couleur(s) ← gris
       E ← Ø
       Insérer(s, E)
       b ← faux
       Tant que E ≠ Ø
           u ← Extraire(E)
           Si u = x alors b ← vrai
           pour tout v ∈ Voisin(u)
               Si couleur(v) = blanc alors
                   couleur(v) ← gris
                   Insérer(v, E)
           couleur(u) ← noir
       Retourner b
```

Thm 5: L'algorithme est correct et linéaire

Rq 3: Si E est une file, on réalise un parcours en largeur
Si E est une pile, c'est un parcours en profondeur

2_ Arbres binaires de recherche

Déf 5: Un arbre binaire de recherche est la donnée d'un ensemble S de noeuds, d'une racine $r \in S$ et de trois fonctions Valeur, Gauche et Droite à valeurs dans $\mathbb{N}$, S et S vérifiant $\forall s \in S$
$\forall n$ noeud du sous-arbre de racine Gauche(s), Valeur$(n) <$ Valeur(s)
$\forall n$ noeud du sous-arbre de racine Droite(s), Valeur$(n) >$ Valeur(s)

Ex 6: 

Développement 1

Thm 6: La recherche dans un arbre binaire de recherche de hauteur h est réalisable en temps $O(h)$.
L'espérance de la hauteur d'un arbre binaire de recherche aléatoire à n noeuds est $O(\log(n))$.

3_ Union - Find

Déf 6: Une structure d'Union-Find est une structure de données représentant une partition d'un ensemble munie des opérations :
- Union(x, y) : réalise l'union des parties contenant x et y
- Find(x) : retourne le représentant de la partie contenant x

Rq 4: Une partition peut être vue comme un ensemble de classes d'équivalence, d'où le terme représentant.

Ex 7: les listes munies de la concaténation et de la fonction "dernier élément" forment une structure d'Union-Find

Thm 7: En utilisant des arbres avec union par rang (Figure 3) et compression de chemin (Figure 4), une série de m opérations pour n éléments est réalisable en temps $O(m\alpha(n))$ où α est une fonction qui croît très lentement ($\alpha(n) \leq 4$ pour toute application réalisable)

III Recherche d'un objet plus complexe

1_ Recherche de motif dans un texte

But : Trouver une ou toutes les occurrences d'un motif pat de taille m dans un texte text de taille $n \geq m$

Rq 5 : Un texte est vu comme un tableau de caractères

Thm 8 (Morris-Pratt) : L'algorithme suivant réalise la recherche de motif en temps linéaire :

Entrée : pat, text, n, m
Sortie : indices de départ des occurrences de pat dans text

```
Algo : i ← 0 , j ← 0
       Tant que i ≤ n-m
         Tant que j < m et pat(j+1) = text(i+j+1)
             j ← j+1
         Si j = m alors afficher i
         i ← i+j - Bord(j)
         j ← max(0, Bord(j))
```

où Bord est calculé par :

```
  i ← 1 , j ← 0
  Tant que i ≤ m-1
     Tant que i+j < m et pat(j+1) = pat(i+j+1)
        j ← j+1
        Si Bord(i+j) = -1 alors Bord(i+j) ← j
     i ← i+j - Bord(j)
     j ← max(0, Bord(j))
```

Ex 8 : Figure 5

Développement 2

Thm 9 : La recherche des facteurs de text à distance d'édition de pat au plus k est réalisable en temps $O(kn)$

2_ Recherche de preuves en logique du premier ordre

But : Étant donné une formule de la logique du premier ordre, trouver de façon algorithmique une preuve de cette formule

Déf 7 : Un littéral est une formule atomique ou sa négation
Une clause est un ensemble fini de littéraux

Rq 6 : Une clause représente la clôture universelle de la disjonction de ses éléments

Ex 9 : $\{R(a,x), R(y,b)\}$ représente $\forall x \forall y\ (R(a,x) \vee R(y,b))$

Déf 8 : La méthode de résolution part d'un ensemble de clauses et cherche à aboutir à la clause vide (ie une contradiction) en utilisant les deux schémas de règles :

$$\frac{C_1 \cup \{L_1\} \quad C_2 \cup \{L_2\} \quad \sigma = mgu(L_1, \overline{L_2})}{C_1[\sigma] \cup C_2[\sigma]}\ res \quad (\text{résolution})$$

$$\frac{C_1 \cup \{L_1, L_2\} \quad \sigma = mgu(L_1, L_2)}{C_1[\sigma] \cup \{L_1[\sigma]\}}\ contr \quad (\text{contraction})$$

où $\sigma = mgu(L_1, L_2)$ signifie "σ est un unificateur le plus général de L_1 et L_2"

Ex 10 : Figure 6

Thm 10 : La résolution fournit un algorithme de décision pour le calcul propositionnel

Rq 7 : La logique du premier ordre étant indécidable, on n'obtient qu'un semi-algorithme en général.

Thm 11 : Un ensemble de clauses est contradictoire ssi on peut dériver la clause vide à partir de lui.

Figure 1: Recherche de 5

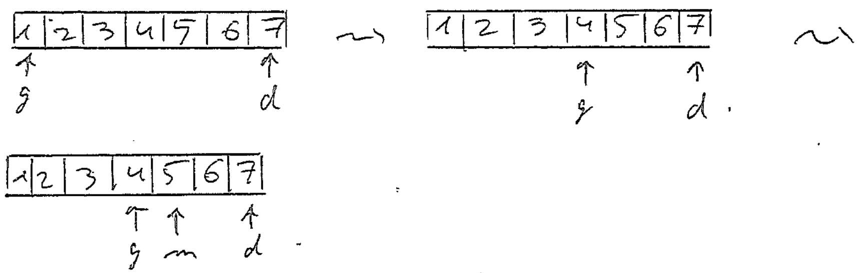


Figure 2 : Recherche de 5

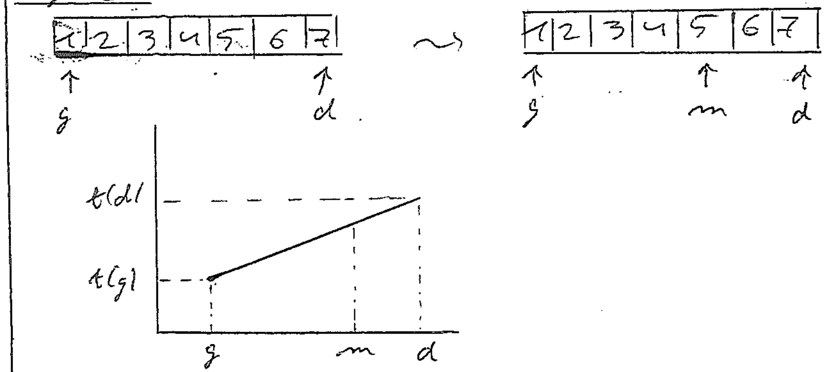


Figure 3 : le rang est une approximation de la hauteur

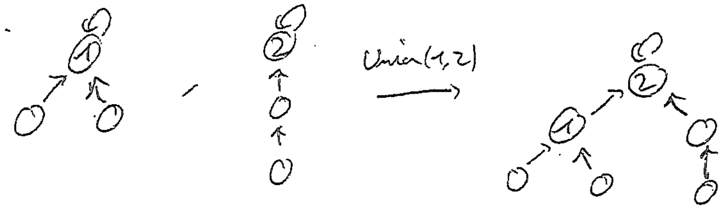


Figure 4 : On compresse à chaque appel de Find

Figure 5 : $Bord(j) = \min_k$ pat[1..k] est un suffixe propre de pat[1..j]

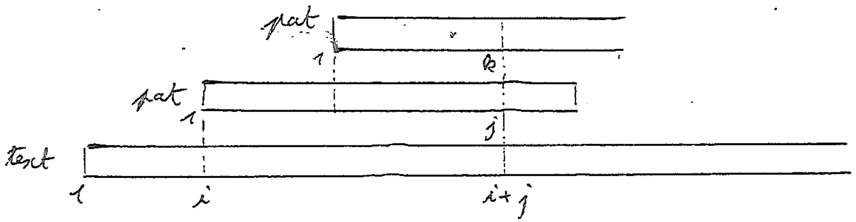


Figure 6 : Pour $C_1 = \{\neg R(x,a), R(x,x)\}$, $C_2 = \{\neg R(v,f(y)), \neg R(v,y)\}$,
$C_3 = \{\neg R(z,z), R(z,u)\}$ $C_4 = \{R(u,f(y)), R(u,y)\}$

$$\dfrac{\dfrac{\dfrac{C_3 \quad C_4}{\{R(f(y),a), R(f(y),y)\}}\,res}{\{R(f(a),a)\}}\,contr \qquad \dfrac{\dfrac{C_1 \quad C_2}{\{\neg R(f(y),a), \neg R(f(y),y)\}}\,res}{\{\neg R(f(a),a)\}}\,contr}{\emptyset}\,res$$