Introduction à l'informatique - Algorithmes et structures linéaires et arithmétique

Table des matières

Abstraction et structure de données

Abstraction d'une structure de données linéaire : paquets de cartes -> structure : le paquet
pioche : structure de données en pile (premier entrer dernier sortie, LIFO).
Autre type de structure : file d'attente : (premier entrer premier sortie, FIFO)

Autre exemple : jeu d'échec, configuration a instant t : possibilités de coup possibles. -> structure d'arbre ; communication online : ordinateur représenter par un sommet de structure de données (graphe) avec flèches qui correspondent aux communications.

Informations :

- Natures distinctes
- Relations entre elles spécifiques
- Structuration/organisation

➢ Structures de données linéaires (Tableaux, listes chaînées, Piles, Files, Tables de hachage (recherche dans une structures de données linéaires))
➢ Structures de données non-linéaires (Arbres, Graphes (représentation de réseaux))

Briques de bases des algorithmes

- Types de bases :
  booléen (0, 1)
  entier (-131,…)
  réel (76.68,…)
  char ‘a’,….
  string ‘hello’,…
- Appels d’entrées-sorties : exemple
  écrire, écrire(‘Quel est votre nom ?’) //print
  lecture, lire(nom) //input
  ….

Procédures et variables

- Procédure : suite d’instructions nommées, que l’on peut appeler pour l’utiliser
- Variable : espace de stockage (emplacement mémoire) pour conserver des informations d’un type donné
- Exemple : comment demander à une personne son nom, son année de naissance, et lui répondre l’anniversaire qu’elle a fêté ou va fêter cette année ?
- Paramètre donnée (lire) : informations pas toujours demandée directement à l’utilisateur, déjà connue (issue d’une saisie ou d’un calcul préliminaire)
  bonjour() -> bonjour(date = entier) {date déjà existante dans le code}
  paramètre dans le call de fonctions

Affectations (en programmation = ; en pseudo code := (une définition, maths))

Conditions (expression booléenne, permet l’exécution d’une séquence d’instructions dans des cas précis)

Itération (répétition d’une séquence d’instructions, boucle)

Solution optimale : si on connaît une formule pour calculer il faut l’utiliser au lieu de faire une méthode naïve.

Fonctions (procédure particulière qui renvoie une valeur d’un type déterminé)

Paramètres résultat (quand on veut rendre visible le changement de valeur d’une ou plusieurs paramètres donnée)

Premiers algorithmes sur le tableau

Un tableau t de taille n est un vecteur de dimension n composé de n éléments de même type, en général représenté comme suit : (feuille)

Un tableau t de taille n contenant des éléments de E est une application :
t : {1,…, n} -> E

Terminaison : pour prouver qu'une itération se termine, il suffit de trouver une quantité (un variant) qui :

- Est un entier positif avant le début de l'itération
- Est un entier positif à la fin de chaque étape de l'itération
- Décroit strictement à chaque étape de l'itération

Correction : pour prouver qu'une itération produit un résultat, il suffit de trouver une propriété P (un invariant) telle que :

- (entrée) P est vrai à la 1ère étape de l'itération
- (Récurrence) pour tout entier i, si P est vraie à la ième étape, alors P est vraie à la i+1ème étape
- (sortie) après la dernière étape, P et la condition d'itération devenue fausse doivent permettre de prouver que le résultat est celui attendu.

Complexité : f(n) est en O(g(n)) s'il existe un entier n(0) et une constance c > 0 tels que ∀n >= n(0), f(n) <= c*g(n)

Parcours d'un tableau

Recherche dans un tableau

Objectif : réaliser un traitement sur l'ensemble des éléments d'un tableau (parcours séquentiel)

Recherche dans un tableau non trié : recherche séquentielle : répondre vrai ou faux selon qu'un élément donné appartient ou non à un tableau

Recherche dans un tableau trié : dichotomie -> complexité en log_2(n)

Mise à jour d'un tableau

Insertion dans un tableau non trié : append

Insertion dans un tableau trié : « insertion à la kème position » :
- trouver la position de l'élément a insérer
- réaliser l'insertion de l'élément (, ajouter une case, decaler les autres valeurs a n+1 en partant de la fin au debut, puis inserer la valeur)

Supression d'un tableau non trié :
- rechercher sequentiellement l'élement : parcourir a partir de l'index -1
-taser le tableau sur l'element s'il a été trouvé (symétrique de l'ajout) + suppression d'une case

Supression d'un tableau triée :
- position par dichotomie
- on tasse (symétrique de l'ajout) + suppression d'une case

Tri d'un tableau

Pourquoi ?
- reception d'informations par un algorithme -> organisation de celles-ci dans une structure de données
-ensemble muni d'un ordre total –> intéressant de les classer/ordonner

Une relation binaire R sur un ensemble E est une relation d'ordre ssi elle est
- réflexive $\forall i \in E$, iRi
- anti-symétrique $\forall i,j \in E$, iRj => j !Rj
- transitive $\forall i, j, k \in E$, iRj et jRk => iRk

Trier un tableau, c'est classer ses éléments selon un ordre total
plusieurs algorithmes, plus ou moins complexes, existent :

Tri par remplacement (construire un tableau t' trié composé des mêmes éléments que t, complexité n^2 - quadratique)

Tri par insertion (position, décalage jusqu'à la bonne position, ne crer pas de nouveau tableau, complexité quadratique, marche bien sur des petits tableaux)

Tri par récursivité (un algorithme/fonctions qui s'appelle lui-meme)

Tri par fusion (tri par comparaison stable, technique du « diviser pour régner » et n'operant pas en place. (complexité quadratique max n*log_2(n))
Idée générale :

- Si le tableau n'a qu'un élémént il est déjà trié
- Sinon, séparer le tableau en 2 parties égales
- Trier récursivement les 2 parties avec l'algorithme
- Fusionner les deux tableaux triés en un seul tableau trié

(mettre screen de l'algo) :: = concatenation

Algorithmes sur les entiers et les réels

Comment une machine fait-elle pour compter ?

- N := n + 1
  n stocké sur un octet (tableau)
  incrementation : decalage (td) tant qu'il y a un on mets un zero et si 1 mettre 0 (principe de retenue, puis réécrire le reste du nombre)
  dépassement de capacité ? return error code -> besoin de coder sur des espaces mémoires plus grands
- N = 4 mod[27]
  n = m x q+r (q ∈ N, r ∈ N_26)
- A et B premiers entre eux ?
  deux nombres premiers sont premiers entre eux ou quand pgcd(a,b) = 1
  a = x mod[b] si a>b
  recursive q modulo[r]
  jusqu'à trouver diviseur commun (ou 1)
  UTILE ? nombres pseudo-aléatoires -> générateurs congruentiels linéaires (voir slide 14)
  utiliser dans des communication sécurisé (code vigenère si len(clef) = len(message) ;
  cryptosystème RSA (look wikipedia)
  - choisir p != q deux grads nombres premiers
  calculer n = pq
  calculer _phi(n) = (p-1)(q-1)
  choiisir un entier e premoer avec _phi(n) -> clé d'alice
  calculer l'entier d inverse de e mod(_phi(n)) -> clé de bob
  D x e = 1 mod(_phi(n))
  Et graçe a la relation de bézout :
  d*e + k*_phi(n) = 1
  force du RSA : la décomposition d'un entier en produit de facteurs premiers est un problème difficile (np completude)
  (voir déchifrement et chifrement slide 19)
- Calcul de logarithme
  log_2(152) ⇔ 2^x = 152 ; 2^x – 152 = 0 ; metonde de la descente