Algorithmique procédurale

Maps et tables de hachage

Équipe pédagogique

CY Tech

Bibliographie - Sitographie

- Rémy Vernay
- Cormen T.H., Leiserson C.E., Rivest R.L. et Stein C. Algorithmique. Dunod. 2010, 1188 pages.

Maps

- Introduction
- Définition
- Méthode
- Fonctions
- Préconditions
- Postconditions
- Exemple

Introduction

- Le type de donnée abstrait : les tables de symboles.

Introduction

- Le type de donnée abstrait : les tables de symboles.
- But : stocker des paires "clé-valeur" et pouvoir y accéder efficacement, en utilisant seulement la clé.

Introduction

- Le type de donnée abstrait : les tables de symboles.
- But : stocker des paires "clé-valeur" et pouvoir y accéder efficacement, en utilisant seulement la clé.
- Exemple : répertoire dans un téléphone portable

Introduction

- Le type de donnée abstrait : les tables de symboles.
- But : stocker des paires "clé-valeur" et pouvoir y accéder efficacement, en utilisant seulement la clé.
- Exemple : répertoire dans un téléphone portable
  - la clé = le nom du contact
  - la valeur = numéro de téléphone.

Introduction

- Le type de donnée abstrait : les tables de symboles.
- But : stocker des paires "clé-valeur" et pouvoir y accéder efficacement, en utilisant seulement la clé.
- Exemple : répertoire dans un téléphone portable
  - la clé = le nom du contact
  - la valeur = numéro de téléphone.
- Exemple : un dictionnaire, où est associé un mot à une définition.

Introduction

- Le type de donnée abstrait : les tables de symboles.
- But : stocker des paires "clé-valeur" et pouvoir y accéder efficacement, en utilisant seulement la clé.
- Exemple : répertoire dans un téléphone portable
  - la clé = le nom du contact
  - la valeur = numéro de téléphone.
- Exemple : un dictionnaire, où est associé un mot à une définition.
  - la clé = nom
  - la valeur = la définition.

Introduction

- Le type de donnée abstrait : les tables de symboles.
- But : stocker des paires "clé-valeur" et pouvoir y accéder efficacement, en utilisant seulement la clé.
- Exemple : répertoire dans un téléphone portable
  - la clé = le nom du contact
  - la valeur = numéro de téléphone.
- Exemple : un dictionnaire, où est associé un mot à une définition.
  - la clé = nom
  - la valeur = la définition.
- Dans les 2 cas, on a un nom ou un mot pour retrouver une information le numéro de téléphone ou la définition.

Introduction

- Autre exemple : les news d'un site internet. Chaque news possède un numéro d'identification unique qui permet la distinguer,
  - la clé : id
  - les valeurs : le reste de la news (titre, contenu, etc).

Introduction

- Il existe deux type de tables de symboles :
  - les tables associatives (Maps en anglais)
  - les dictionnaires.

Introduction

- Il existe deux type de tables de symboles :
  - les tables associatives (Maps en anglais)
  - les dictionnaires.
- La différence : les clés des maps sont uniques tandis que celles d'un dictionnaire ne le sont pas nécessairement.

Introduction

- Il existe deux type de tables de symboles :
  - les tables associatives (Maps en anglais)
  - les dictionnaires.
- La différence : les clés des maps sont uniques tandis que celles d'un dictionnaire ne le sont pas nécessairement.
- Dans nos exemples, le répertoire téléphonique est un dictionnaire : possibilité de contacts portant le même nom avec des numéros différents.

Introduction

- Il existe deux type de tables de symboles :
  - les tables associatives (Maps en anglais)
  - les dictionnaires.
- La différence : les clés des maps sont uniques tandis que celles d'un dictionnaire ne le sont pas nécessairement.
- Dans nos exemples, le répertoire téléphonique est un dictionnaire : possibilité de contacts portant le même nom avec des numéros différents.
- Les news ont un seul id avec éventuellement plusieurs valeurs - c'est un Maps.

Introduction

- Un Map représente donc une relation binaire surjective : chaque élément d'un Map est une paire qui met en relation une clé à une valeur : chaque clé est unique, mais on peut avoir des doublons pour les valeurs.

Introduction

- Un Map représente donc une relation binaire surjective : chaque élément d'un Map est une paire qui met en relation une clé à une valeur : chaque clé est unique, mais on peut avoir des doublons pour les valeurs.
- Une relation binaire est surjective si tout élément admet un antécédent.

Introduction

- Un Map représente donc une relation binaire surjective : chaque élément d'un Map est une paire qui met en relation une clé à une valeur : chaque clé est unique, mais on peut avoir des doublons pour les valeurs.
- Une relation binaire est surjective si tout élément admet un antécédent.
- Il peut être représenté par un tableau à deux colonnes.

Définition

- Association $\ll$ clef,valeur $\gg$

Définition

- Association $\ll$ clef,valeur $\gg$
- Une clé donne une seule valeur

Définition

- Association $\ll$ clef,valeur $\gg$
- Une clé donne une seule valeur
- Une valeur peut être obtenue par plusieurs clés

Définition

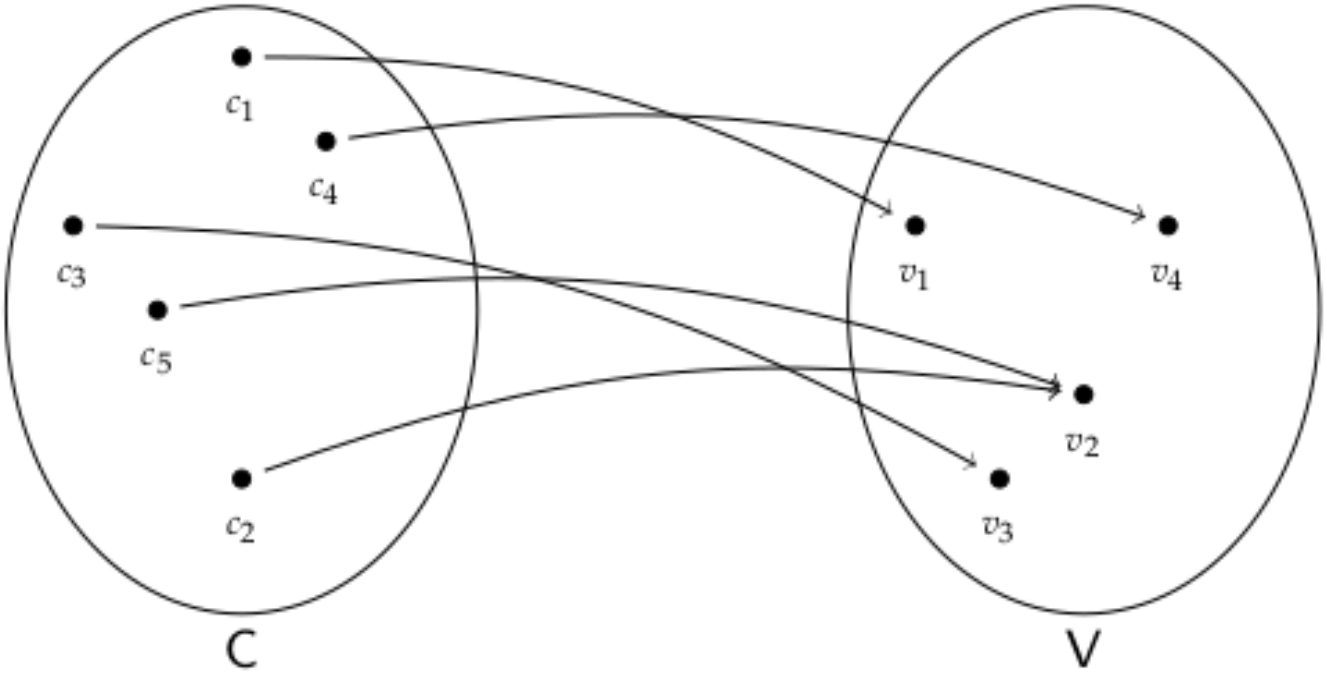
- Association $\ll$ clef,valeur $\gg$
- Une clé donne une seule valeur
- Une valeur peut être obtenue par plusieurs clés
- Aucune valeur n'existe sans clé

Définition

- Association $\ll$ clef,valeur $\gg$
- Une clé donne une seule valeur
- Une valeur peut être obtenue par plusieurs clés
- Aucune valeur n'existe sans clé
- Aucune clé n'existe sans valeur

Définition

- Soient C l'ensemble des clés et V l'ensemble des valeurs



$$f : \begin{array}{ccc} C & \longrightarrow & V \\ c_i & \longmapsto & f(c_i) \end{array}$$

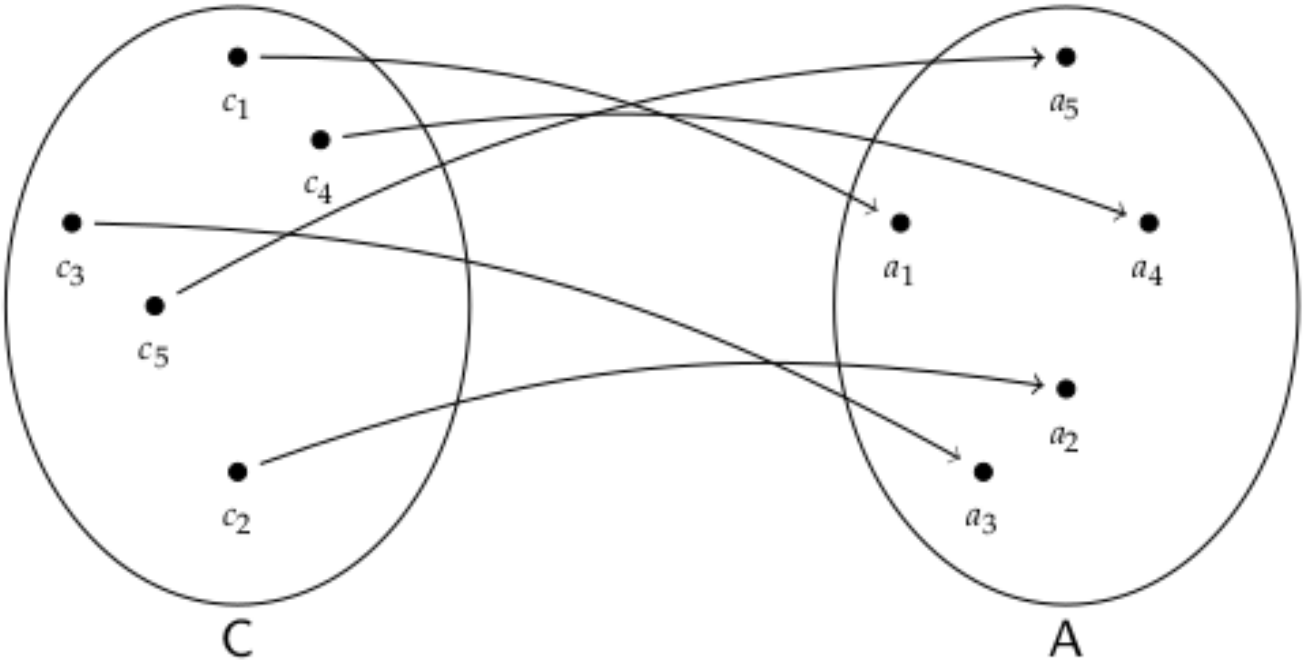
Il s'agit donc d'une application surjective

Définition

- Application surjective
  - Problème
    - ★ $f(c_j) = f(c_k) \Rightarrow$ affectation de $f(c_j)$ écrase $f(c_k)$
  - Solution
    - ★ utilisation d'alvéoles

Définition

- Soient C l'ensemble des clés et A l'ensemble des alvéoles



$$g : \begin{array}{ccc} C & \longrightarrow & A \\ c_i & \longmapsto & g(c_i) = a_i \end{array}$$

Il s'agit donc d'une application bijective

Définition

- Utilisation d'alvéoles
  - les clés sont liées aux alvéoles
  - la donnée est stockée dans l'alvéole
  - modifier le contenu d'une alvéole n'affecte aucune autre
  - concept théorique identique

Définition

- Propriétés
  - connaissance de la méthode de stockage des clés inutile
  - connaissance de la méthode de stockage des valeurs inutile
  - accès direct à chaque valeur en fonction de sa clé

Méthode

- Déclaration du conteneur Map
  - un type pour les clés
  - un type pour les valeurs

map : Map<type_clé> de type_valeur

Exemple :
map : Map<chaîne> de réel

Fonctions

procédure créerMap(map: Map<Élément> de Élément (S))

procédure stocker(map: Map<Élément> de Élément (E/S) ; clef: Élément ; valeur: Élément)
procédure supprimer(map: Map<Élément> de Élément (E/S) ; clef: Élément)

fonction estVide(map: Map<Élément> de Élément): Booléen
fonction cardinalite(map: Map<Élément> de Élément): Entier

fonction clefExiste(map: Map<Élément> de Élément ; clef: Élément): Booléen
fonction valeurExiste(map: Map<Élément> de Élément ; valeur: Élément): Booléen

fonction valeurDe(map: Map<Élément> de Élément ; clef: Élément): Élément
procédure listeClefs(map: Map<Élément> de Élément ; clefs: Liste de Élément (S))

procédure copier(map1: Map<Élément> de Élément ; map2: Map<Élément> de Élément (S))

Préconditions

créerMap(map) $\Leftrightarrow$ aucune

stocker(map,clef,valeur) $\Leftrightarrow$ map initialisée

supprimer(map,clef) $\Leftrightarrow$ clef existante

estVide(map) $\Leftrightarrow$ map initialisée

cardinalite(map) $\Leftrightarrow$ map initialisée

clefExiste(map,clef) $\Leftrightarrow$ map initialisée

valeurExiste(map,valeur) $\Leftrightarrow$ map initialisée

valeurDe(map,clef) $\Leftrightarrow$ clef existante

listeClefs(map,clefs) $\Leftrightarrow$ map initialisée

copier(map1,map2) $\Leftrightarrow$ map1 initialisée

Postconditions

créerMap(map) $\Rightarrow$ map initialisée

stocker(map,clef,valeur) $\Rightarrow$ cardinalite(map)$>$0

supprimer(map,clef) $\Rightarrow$ map initialisée

estVide(map) $\Rightarrow$ aucune

cardinalite(map) $\Rightarrow$ aucune

clefExiste(map,clef) $\Rightarrow$ aucune

valeurExiste(map,valeur) $\Rightarrow$ aucune

valeurDe(map,clef) $\Rightarrow$ aucune

listeClefs(map,clefs) $\Rightarrow$ clefs initialisée avec des clefs existantes

copier(map1,map2) $\Rightarrow$ map2 initialisée

Exemple

Saisie de noms d'étudiants avec leur moyenne.

```
nb, i: Entier
valeur: réel
clef: Chaîne
mapMoyennes: Map<chaîne> de réel
créerMap(mapMoyennes)
répéter
    écrire("Combien d'étudiants ?")
    lire(nb)
jusqu'à nb≥0
pour i←1 à nb faire
    répéter
        écrire("Nom de l'étudiant ?")
        lire(clef)
    jusqu'à non(clefExiste(mapMoyennes,clef))
    répéter
        écrire("Sa moyenne ?")
        lire(valeur)
    jusqu'à 0≤valeur et valeur≤20
    stocker(mapMoyennes,clef,valeur)
fin pour
```

Tables de hachage

- Introduction
- Principe
- Collisions
- Fonctions de hachage
- Exemple

Introduction

- Les tables de hachage diminue la complexité des méthodes courantes :
  - de recherche
  - d'insertion
  - de suppression

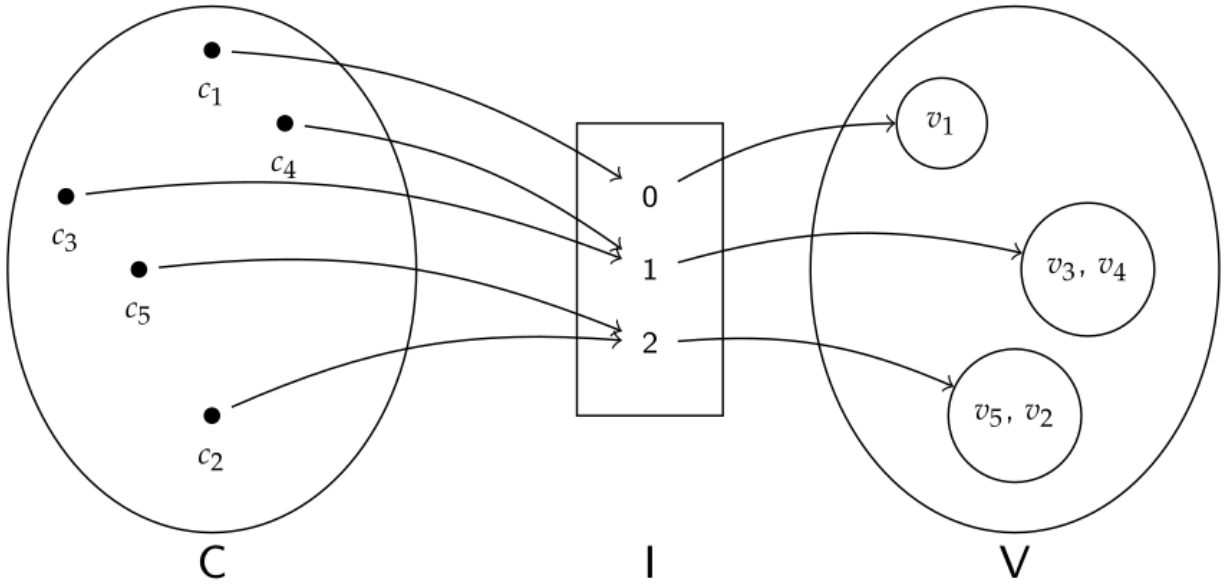
  Les implémentations possibles des maps peuvent être :
  - par adressage direct
  - par des méthodes de hachage

Introduction

- La méthode par adressage direct : on utilise un tableau dans lequel chaque alvéole correspond à une clé
  - cette méthode est trop coûteuse pour beaucoup de données
    - ★ car on a un index contenant autant de clés que de valeurs
    - ★ des types de clés volumineux (chaînes, etc.)
  - on va tenter de
    - ★ ne stocker que des entiers comme index
    - ★ limiter la taille de l'index

Principe

- Les tables de hachage requiert moins de place qu'une table à adressage direct :
  - Soient C l'ensemble des clés, I celui des index et V celui des valeurs



$$v \circ h : \begin{array}{ccc} C & \longrightarrow & V \\ c_i & \longmapsto & v \circ h(c_i) \end{array}$$

Attention : Problème de collisions ! ! !

Principe

- Fonctionnement
  - on utilise une fonction de hachage $h(c)$
    - ★ qui renvoie une valeur de hachage avec $i \in 0, 1, 2, \ldots, n-1$
    - ★ avec des valeurs de hachage de $h(c)$ équiprobables
  - un stockage en fonction de $h(c)$ pour gérer les collisions

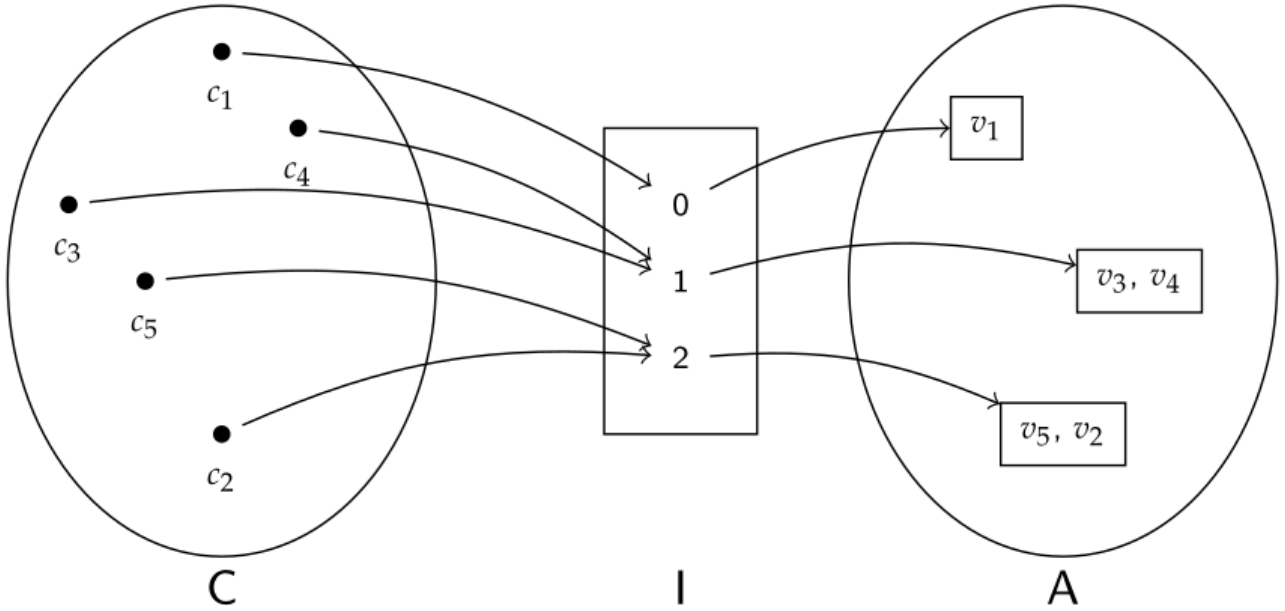
Collisions

- Plusieurs valeurs à stocker par valeur de hachage $h(c)$
  - utilisation d'alvéoles contenant des listes
  - utilisation de structures pour les éléments de la liste

Type StructValeur = **structure**
    clef: Élément
    valeur: Élément
fin type

Collisions

- Utilisation de listes



Collisions

- On stocke la structure comprenant la clé et la valeur
- Le nombre de valeurs stockées par liste reste faible
- La recherche est rapide
- Les valeurs de hachage doivent être bien réparties

Fonctions de hachage

- Exemple de fonction de hachage : clé du numéro de sécurité sociale

$$h(c) = 97 - (c - \left\lfloor \frac{c}{97} \right\rfloor \times 97)$$
$$h(c) \in \{0, 1, 2, \ldots, 96\}$$

$$1\ 95\ 74\ 10\ 023\ 068 \Rightarrow c = 1957410023068$$
$$h(1957410023068) = 10$$

Fonctions de hachage

- Différentes méthodes
  - créées sur mesure
  - méthode dite de la division
  - méthode dite de la multiplication
  - hachage universel
- On doit avoir la même probabilité d'obtenir toutes les valeurs de hachage

  et il faut tendre au maximum vers cet absolu

Remarque : Collisions

- Utilisation de l'adressage ouvert
  - On stocke les valeurs de hachage dans des cases contigües
  - La position de ces cases est déterminée par une méthode de sondage
  - L'idée simple est d'ajouter 1 jusqu'à trouver une position libre

Exemple

Assurés

préconditions : `numero` est un numéro de sécurité sociale valide
postconditions : $0 \leqslant$ h(`numero`) < 97
fonction h(numero: Entier): Entier
    retourner 97-(numero mod 97)
fin fonction

Type Assure $=$ **structure**
    numero: Entier
    nom: Chaîne
    prenom: Chaîne
fin type

Exemple

Assurés

```
programme Exemple
    alveoles: Tableau de Liste de Assure
    i, numero: Entier
    nom, prenom: Chaîne
    créerTableau(alveoles,97)
    pour i←0 à 96 faire
        créerListe(alveoles[i])
    fin pour
    ...
    lire(nom)
    lire(prenom)
    lire(numero)
    enregistrer(alveoles,nom,prenom,numero)
    ...
fin programme
```

Exemple

Assurés

```
procedure enregistrer(alveoles: Tableau de Liste de Assure (E/S) ; nom, prenom: Chaîne ;
numero: Entier)
    assure: Assure
    nouveau: Booléen
    l: Liste de Assure
    nouveau←VRAI
    l←alveoles[h(numero)]
    tant que non(estVide(l) et nouveau faire
        si tête(l).numero=numero alors
            nouveau←FAUX
            tête(l).nom←nom
            tête(l).prenom←prenom
        fin si
        l←reste(l)
    fin tant que
    si nouveau alors
        assure.numero←numero
        assure.nom←nom
        assure.prenom←prenom
        ajouter(alveoles[h(numero)],assure)
    fin si
fin programme
```