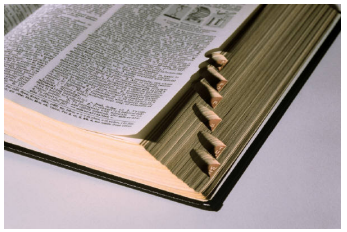
M2103 - Structures de Données

Dictionnaires

Plan du Cours

- Introduction
- Classe `Dictionnaire`
- Implémentation
  - Tableau
  - Liste chaînée

Introduction



- Ensemble de couples (Clé,Valeur)
  - Clé : mot
  - Valeur : définition
- Ordre total sur les clés.
- Opérations définies :
  - init() : dictionnaire initialisé et vide
  - contient(maClé) : permet de tester si le dictionnaire contient une valeur associée à **maClé**
  - valeur(maClé) : retourne la valeur associée à **maClé**
  - associe(maClé,maVal) : si le dictionnaire contient **maClé**, remplace par **maVal** la valeur associée, sinon ajoute le couple (**maClé**,**maVal**) au dictionnaire
  - supprime(maClé) : supprime le couple contenant **maClé** si le dictionnaire la contient

Classe `Dictionnaire` : Spécification

```
def __init__():
  """
  :sortie self:
  :post-condition: dictionnaire initialisé et vide
  """

def contient(self,maClé):
  """
  :entrée self:
  :entrée maClé: object
  :sortie b: bool
  :post-condition: b est True ssi. Le dictionnaire
    contient maClé
  """

def valeur (self,maClé):
  """
  :entrée self:
  :entrée maClé: object
  :sortie val: object
  :pré-condition: le dictionnaire contient maClé
  :post-condition: val est la valeur associée à maClé
   """
```

```
def associe(self,maClé,maVal):
   """
   :entrée-sortie self:
   :entrée maClé: object
   :entrée maVal: object
   :post-condition: si maClé est dans le dictionnaire,
     remplace la valeur associée par maVal, sinon
     ajoute le couple (maClé,maVal) au dictionnaire
   """

 def supprime(self,maClé):
   """
   :entrée-sortie self:
   :entrée maClé: object
   :post-condition: si maClé est dans le dictionnaire,
     supprime le couple contenant maClé, sinon ne
     fait rien
    """
```

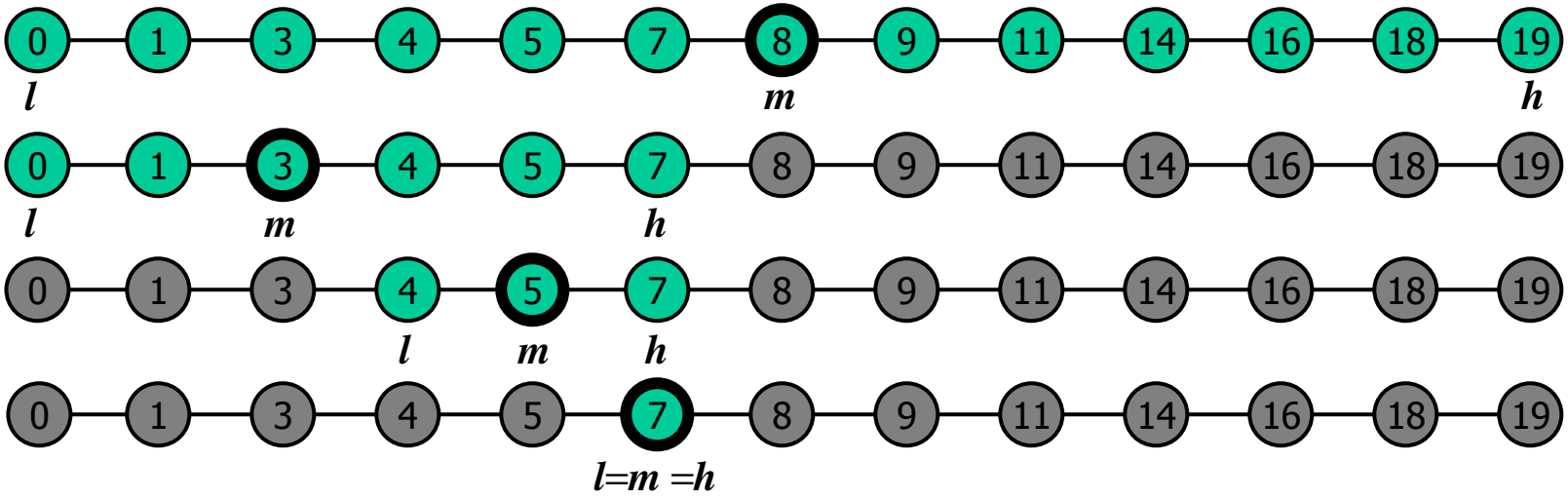
Implémentation avec Tableau

- Dictionnaire implémenté au moyen d'une séquence ordonnée
  - Les éléments sont stockés dans un tableau ordonné
- Performance :
  - La recherche a une complexité $O(\log n)$
  - L'insertion a une complexité $O(n)$
  - La suppression a une complexité $O(n)$
- Implémentation efficiente pour des petits dictionnaires ou dictionnaires pour lesquels l'opération de recherche est la plus usitée (e.g., autorisations de cartes de crédit)

Rappel : Recherche dans le Dictionnaire

- Recherche dichotomique permet d'implémenter cherche(clé)
  - À chaque étape, le nombre d'éléments candidats est divisé par 2
  - Se termine après O(log n) étapes

- Exemple : cherche(7)



Classe `Dictionnaire` : Implémentation avec liste chaînée simple

```
class MaillonDico:
  def __init__(self, clé, val, suiv):
      self._clé = clé
      self._val = val
      self._suiv = suiv

#méthode privée
  def cherche_maillon (self, maClé):
     """
    :entrée self:
    :entrée clé: object
    :sortie tmp, prec_tmp: maillonDico
    :post-condition: s'il existe un maillon avec maClé,
      tmp référence ce maillon, sinon tmp référence le
      premier maillon avec une clé supérieure; s'il n'en
      existe pas, tmp est None. prec_tmp référence le
      dernier maillon portant une clé inférieure à
      maClé; s'il n'en existe pas, prec_tmp est none.
     """
     prec_tmp = None
     tmp = self._premier
     while tmp != None and tmp.clé < maClé:
       prec_tmp = tmp
       tmp = tmp.suiv
     return tmp, prec_tmp

  def __init__():
    self._premier = None
```

```
def valeur (self,maClé):
  (cour, prec) = self._cherche_maillon(maClé)
  val = cour._val
  return val

 def associe(self,maClé,maVal):
    (cour, prec) = self._cherche_maillon(maClé)
    if (cour != None) and (cour._clé == maClé):
      cour._val = maVal
    else:
      nouv=Dictionnaire.MaillonDico(maClé, maVal, cour)
      if prec == None:
        self._premier = nouv
      else:
        prec._suiv = nouv

def supprime(self, maClé):
  (cour, prec) = self._cherche_maillon(maClé)
  if (cour != None) and (cour._clé == maClé):
    if prec == None:
      self._premier = cour._suiv
    else:
      prec._suiv = cour._suiv

def contient(self,maClé):
  (cour, prec) = self._cherche_maillon(maClé)
  b = (cour != None) and (cour._clé == maClé)
  return b
```