Dictionnaires

Mutabilité

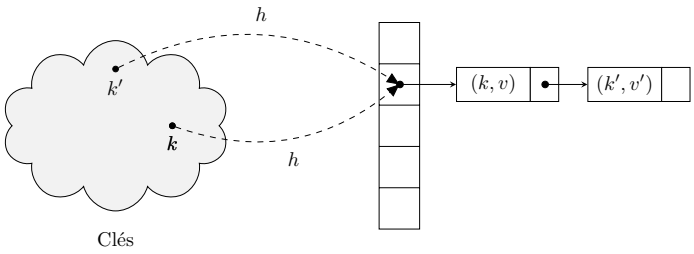
Un objet est dit **mutable** si l'on peut changer sa valeur après sa création sans nouvelle affectation. Il est dit **immutable** dans le cas contraire.

Objets immutables : entiers, flottants, booléens, chaines de caractères, tuples...	Objets mutables : listes, dictionnaires, sets...

Dictionnaires

- Un **dictionnaire** est une structure de données qui à chaque **clé** associe une **valeur**. Il possède les opérations suivantes :
  - Ajouter une association (clé, valeur).
  - Supprimer une association (clé, valeur).
  - Obtenir les valeurs associée à une clé donnée.
- Les dictionnaires de Python sont implémentés par **table de hachage** mais il est aussi possible de les implémenter avec un arbre binaire de recherche (hors-programme).
  Une table de hachage est composée de :
  - Un **tableau** contenant les valeurs.
  - Une **fonction de hachage** h telle que, si k est une clé, h(k) est l'indice du tableau où se trouve la valeur associée à k.

  h(k) n'est définit que si k est immutable, c'est-à-dire non modifiable. Il est en effet fortement déconseillé d'utiliser une clé qui puisse être modifiée puisque cela changerait son image par la fonction de hachage.
- Souvent, il y a beaucoup plus de clés possibles que de cases du tableau, ce qui conduit à des collisions : plusieurs clés ayant la même image par h. On peut résoudre ces collisions soit par **chaînage**, en stockant une liste à chaque position de la table de hachage, soit par **adressage ouvert**.

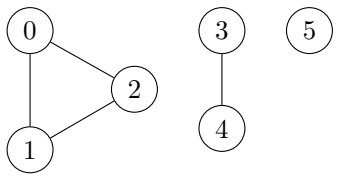


Resolution de collisions par chaînage

- Exemple d'utilisation de dictionnaire :

```
d = {"rouge": (255, 0, 0), "jaune": (255, 255, 0)}
# "rouge" est une clé de  valeur associée (255, 0, 0)
d["rouge"] # donne (255, 0, 0)
d["blabla"] # donne une erreur
"rouge" in d # renvoie True
for k in d:
    print(k) # affiche "rouge" et "jaune"
len(d) # nombre de clés
```

- Au lieu de la représentation par matrice/liste d'un graphe G, on peut représenter G par un dictionnaire `d`, où `d[v]` est la liste (ou l'ensemble) des voisins du sommet `v`.
  Exemple :




```
d = {0 : [1, 2], 1: [0, 2], 2: [0, 1],
     3: [4], 4: [3], 5: []}
```

Manipulation des dictionnaires

Python	Description	Complexité
`d[k] = v`	Ajout (ou modification) d'une association de `k` à `v`	O(1) en moyenne
`d[k]`	Accès à la valeur de clé `k`	O(1) en moyenne
`len(d)`	Nombre de clés de `d`	O(1) en moyenne
`for k in d:`	Parcourir les clés `k` de `d`	O(n) où n est le nombre de clés
`k in d`	Test si `k` est une clé de `d`	O(1) en moyenne
`d.copy()`	Copie de `d`	O(n) où n est le nombre de clés