Structures de Données Simples 2018 (CSV'18)

TD 2 : Compteurs and multimaps

Pour chaque exercice programmer une ou plusieurs fonctions et plusieurs tests unitaires.

Exercice 1 Utiliser les tables associatives pour implémenter des compteurs (principalement, des multiensembles). Un compteur permet de compter le nombre d'occurrences des éléments. Par exemple, le dictionnaire `d = {'a' : 1, 'b' : 3}` représente une occurrence de `a` et 3 occurrences de `b`. Implémenter des fonctionnalités suivantes :

1. La procédure `add` ajoute une occurrence d'un élément au compteur : `add(d,'a')` fait que `d == {'a' : 2, 'b': 3}`, et ensuite `add(d,'c')` fait que `d == {'a' : 2, 'b': 3, c : '1'}`.
2. La procédure `rem` supprime tous les occurrences d'un élément du compteur. Par exemple, pour le compteur `d = {'a' : 2, 'b': 3, c : '1'}` l'execution de `rem(d,'b')` fait que `d == {'a' : 2, c : '1'}`.
3. La fonction `size` calcule le nombre total d'occurrences de tous les éléments : si `c = {'d' : 1, 'f' : 3}`, alors `size(c) = 4`.
4. La fonction `merge` calcule la fusion de deux compteurs :

   ```
   merge({'a' : 1, 'b' : 3}, {'b' : 2, 'c' : 4}) == {'a' : 1, 'b' : 5, 'c' : 4}
   ```

5. La fonction `max_key` trouve l'élément avec le plus grand nombre d'occurrences : pour `d = {'a' : 2, 'b' : 4}` ça donne `max_key(c) == 'b'`.
6. La fonction `elements` qui retourne l'ensemble d'éléments (sans nombre d'occurrences). Par exemple,

   ```
   elements({'a':2, 'b':4, 'c':2}) == set(['a', 'b', 'c'])
   ```

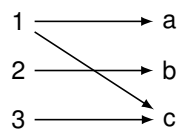
Estimer la complexité de ces procédures et fonctions.

Exercice 2 Utiliser les compteurs pour trouver les 10 mots les plus fréquents dans la Bible. Pareil pour « Les Aventures d'Alice au pays des merveilles ». Ce dernier est décomposé en 67 fichiers `alice.xx.txt` avec les pages consécutives.

Exercice 3 En considérant les tables associatives (`dict`) comme les fonctions avec domaine fini, implémenter les fonctions de composition des fonctions. Par exemple,

```
f = {1:'a', 2:'b', 3:'a', 4:'c'}
g = {'a':1000, 'b':2000}
compose(f, g) == {1:1000, 2:2000, 3:1000}
```

Exercice 4 Une relation binaire $R \subseteq A \times B$ peut être représentée en deux manières différentes : 1) avec un ensemble des paires ou 2) avec un multimap, un dictionnaire qui envoie un élément A à un ensemble d'éléments B. Par exemple, la relation $R = \{(1, a), (1, c), (2, b), (3, a)\}$, en graphe



est représentée avec l'ensemble suivant

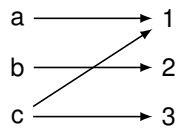
```
R = set([(1,'a'), (2,'b'), (1,'c'), (3,'a')])
```

et avec le multimap suivant

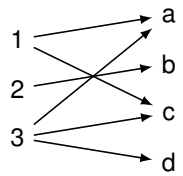
```
R = dict()
R[1] = set(['a'])
R[2] = set(['b', 'c'])
R[3] = set(['a'])
```

Pour les deux représentations implémenter des fonctions qui réalisent des opérations suivantes et caractériser leur complexité.

1. `empty_rel()` qui crée une relation vide,
2. `invert(R)` qui retourne l'inverse de la relation $R^{-1} = \{(y, x) \mid (x, y) \in R\}$. Par exemple, `invert(R)` donne



3. `add(R,x,Y)` qui ajoute à R des liens entre un lien entre x et tout élément de Y : $R \cup \{(x, y) \mid y \in Y\}$. Par exemple, `add(R,3,set(['a','c','d']))` donne



4. `map_element(R,x)` qui renvoie un élément : $R(x) = \{y \mid (x, y) \in R\}$. Par exemple, `map_element(R,1)` donne `set(['a','c'])`.

Implémenter également des fonction de conversion entre les deux représentations.

Exercice 5 Créer un index de recherche pour le livre « Alice's Adventures in Wonderland » c.à.d. une relation qui associe à chaque mot l'ensemble de numéros de pages du livre sur lesquels le mot est présent. Comment utiliser cet indexe pour trouver des pages qui contiennent une suite donnée de mots recherchés (comme avec Google).

Exercice 6 Créer un moteur de recherche de Bible qui pour une suite de mots trouve tous les verses qui contient les mots.