

Dans la mesure du possible, on justifiera le code demandé par des invariants.

## 1 Opérations utiles sur les listes

Une liste  $t$  étant donnée :

1.  $\text{len}(t)$  désigne la longueur de  $t$  ;
2. pour tout  $i$  dans  $\text{range}(\text{len}(t))$ ,  $t[i]$  désigne l'élément d'indice  $i$ .
3. On peut construire une liste de  $n$  éléments tous physiquement égaux à  $x$  avec l'expression  $[x] * n$ .
4. On peut ajouter un élément  $x$  à la fin de  $t$  par l'instruction  $t.append(x)$  (la longueur de  $t$  augmente alors de 1).
5. On peut ajouter tous les éléments d'une liste  $u$  à la fin de la liste  $t$  par l'instruction  $t += u$ .

On supposera que ces opérations sur les listes ont respectivement une complexité :

1.  $\Theta(1)$
2.  $\Theta(1)$
3.  $\Theta(n)$
4.  $\Theta(1)$  (ce n'est pas tout à fait vrai mais pour ce TP, c'est raisonnable)
5.  $\Theta(\text{len}(u))$  (ce n'est pas tout à fait vrai mais pour ce devoir, c'est raisonnable)

## 2 Programmation

1. Écrire une fonction `random_list(n, k)` construisant une liste de  $n$  entiers tirés au hasard dans l'intervalle  $\text{range}(k)$ . On pourra utiliser la fonction `randrange` du module `random` à cet effet.
2. Écrire une fonction `counting_sort(k, t)` prenant en argument une liste  $t$  d'entiers appartenant à  $\text{range}(k)$  et retournant une copie triée de ce tableau. On utilisera impérativement l'algorithme suivant :
  - On construit un tableau  $u$  de taille  $k$  initialisé avec des 0.
  - On parcourt  $t$ . Pour chaque valeur  $x$  trouvée, on incrémente  $u[x]$ . À la fin du parcours, pour tout entier  $i$  de  $\text{range}(k)$ ,  $u[i]$  contient donc le nombre d'occurrences de  $i$  dans  $t$ .
  - Il est alors facile de construire un tableau  $r$  trié répondant à la question posée.
3. Écrire suivant le même principe une fonction `bucket_sort(f, k, t)` retournant une copie de  $t$  triée suivant le critère  $f$ . Plus précisément,  $f$  doit être une fonction prenant ses valeurs dans  $\text{range}(k)$  et les éléments de la liste résultat sont triés par ordre croissant de leurs images par  $f$ .  
De plus, on fera en sorte que le tri soit *stable*, c'est-à-dire que pour tout couple de valeurs  $x$  et  $y$  ayant même image par  $f$ ,  $x$  et  $y$  apparaissent dans le même ordre dans  $t$  et dans la liste triée.

Par exemple, si  $f$  est la fonction  $n \mapsto n^2 \% 5$ , et  $t = [0, 1, 5, 6, 4, 12, 10, 9, 17, 2, 6, 7]$ . Nous pouvons calculer  $f(0) = 0$ ,  $f(1) = 1$ ,  $f(2) = 4$ ,  $f(4) = 1$ ,  $f(5) = 0$ ,  $f(6) = 1$ ,  $f(7) = 4$ ,  $f(9) = 4$ ,  $f(10) = 0$ ,  $f(12) = 4$  et  $f(17) = 4$ . Donc `bucket_sort(f, 4, t)` doit renvoyer `[0, 5, 10, 1, 6, 4, 6, 12, 9, 17, 2, 7]`. Le tableau `[5, 0, 10, 1, 6, 4, 6, 12, 9, 17, 2, 7]` est aussi trié suivant le critère  $f$ , mais ne peut être renvoyé par `bucket_sort(f, 4, t)` car les deux premiers éléments ne sont pas dans le même ordre que dans  $t$ , or on veut un tri stable.

4. On se donne la fonction suivante :

```
def radix_sort(k, t):
    """Retourne une copie de t triée par ordre croissant.
    t doit contenir des entiers appartenant à range(k**2)
    """
    def lp(x):
        return x // k
    def rp(x):
        return x % k
    u = bucket_sort(rp, k, t)
    r = bucket_sort(lp, k, u)
    return r
```

Soit  $t = [0, 5, 10, 1, 6, 4, 6, 12, 9, 17, 2, 7]$  et  $k = 5$ . Calculer  $\text{bucket\_sort}(rp, k, t)$ , et en déduire  $\text{radix\_sort}(k, t)$ .

Quelle hypothèse peut-on formuler concernant la fonction  $\text{radix\_sort}$  ?

### 3 Étude de complexité

5. Expliquer (brièvement) et sous quelle hypothèse pourquoi la complexité de  $\text{random\_list}(n, k)$  est un  $\Theta(n)$ .
6. Justifier que la complexité de  $\text{counting\_sort}(n, k)$  est un  $\Theta(\max(n, k))$ .
7. Justifier que la complexité de  $\text{bucket\_sort}(n, k)$  est un  $\Theta(\max(n, k))$ .
8. Quelle est la complexité de  $\text{radix\_sort}$  ?
9. En admettant que la fonction  $\text{bucket\_sort}$  répond bien à l'énoncé, justifier que  $\text{radix\_sort}$  trie le tableau donné en argument. On pourra commencer par regarder le cas où  $k$  vaut 10 pour comprendre ce qu'il se passe et se poser la question : à quelle(s) condition(s) sur  $lp(x), lp(y), rp(x), rp(y)$  a-t-on  $x \leq y$  ?