INF2010 - Structures de données et algorithmes

Été 2020

Travail Pratique 4

Monceaux

Ce travail pratique sera assez différent des précédents. On vous demandait de rouler des tests localement et vous aviez accès au code du test lui-même pour vérifier si vous aviez la bonne logique. Ici on utlisera la plateforme HackerRank, c'est un site extrêmement utilisé pour se pratiquer pour des entrevues ou simplement pour se pratiquer à résoudre des problèmes algorithmiques.

Directives pour commencer le laboratoire :

- Rendez-vous sur le lien : https://www.hackerrank.com/poly-inf2010-tp4
- Faites-vous un compte HackerRank, rappelez-vous de votre nom d'utilisateur, il sera utile pour la remise
- Il est fortement recommandé de faire les énoncés dans cet ordre :
    - https://www.hackerrank.com/contests/poly-inf2010-tp4/challenges/construction-dun-monceau
    - https://www.hackerrank.com/contests/poly-inf2010-tp4/challenges/mauvais-amis-communs

Directives de remise :

- Télécharger vos deux fichiers Java (vous pouvez simplement faire copier-coller de l'éditeur en ligne dans un fichier sur votre ordinateur local). Vous pouvez leur donner comme noms :
    - Construction-dun-monceau.java
    - Mauvais-amis-communs.java
- Mettez clairement vos matricules ainsi que le nom d'utilisateur que nous devons vérifier le score sur HackerRank dans les commentaires de votre code (dès le début de préférence)
- Remettez-les comme à l'habitude sur moodle

Barème de correction :

Construction d'un monceau	/9
Mauvais amis communs	/9
Explication de complexité dans mauvais amis communs	/1
Style	/1

Le point sur le style est comme à l'habitude, il faut que le code soit clair et respecte les bonnes pratiques, le copier-coller (la duplication de code) est généralement une mauvaise pratique.

Votre explication doit être claire et concise, il faut expliquer d'où vient la complexité de chaque terme. Par exemple, si nous avons 'n' éléments auquel on applique un tri par bulle, la complexité est O(n^2). Pas besoin d'expliquer le tri par bulle.

Les deux énoncés ce retrouvent en annexes, mais il est recommandé de les visionner sur HackerRank.

Construction d'un monceau

Compléter les TODOs à l'intérieur du code. Il faut que vous soyez en mesure de respecter la complexite de chacune des methodes pour obtenir tous vos points. Pour prouver que votre structure de données fonctionne réellement, vous devrez appliquer le tri par monceau. Ce tri est extrêmement utile parce que, comparé au tri rapide (quick sort), son pire temps est toujours de O(nlog(n)), tandis que quick sort pourait être de O(n^2).

Lecture recommendée: complexité de build/heapify

Input Format

Les données en entrées vont comme suit:

- La première ligne est 1 ou 0, soit ascendant ou descendant étant l'ordre dans lequel les données doivent être triés en sortie
- La deuxième ligne est une liste d'entiers aléatoire

Constraints

Il est interdit d'utiliser les librairies de java comme PriorityQueue ou sort

- La première ligne: 0 ou 1
- La deuxième ligne: positif, negatif et aucune limite sur la taille de la liste

Output Format

Une liste contenant les entiers triés dans l'ordre demandé.

Sample Input 0

```
1
2 4 7 0 6 3 1 8 5 9
```

Sample Output 0

```
0 1 2 3 4 5 6 7 8 9
```

Sample Input 1

```
0
4 1 6 5 0 9 7 2 3 8
```

Sample Output 1

```
9 8 7 6 5 4 3 2 1 0
```

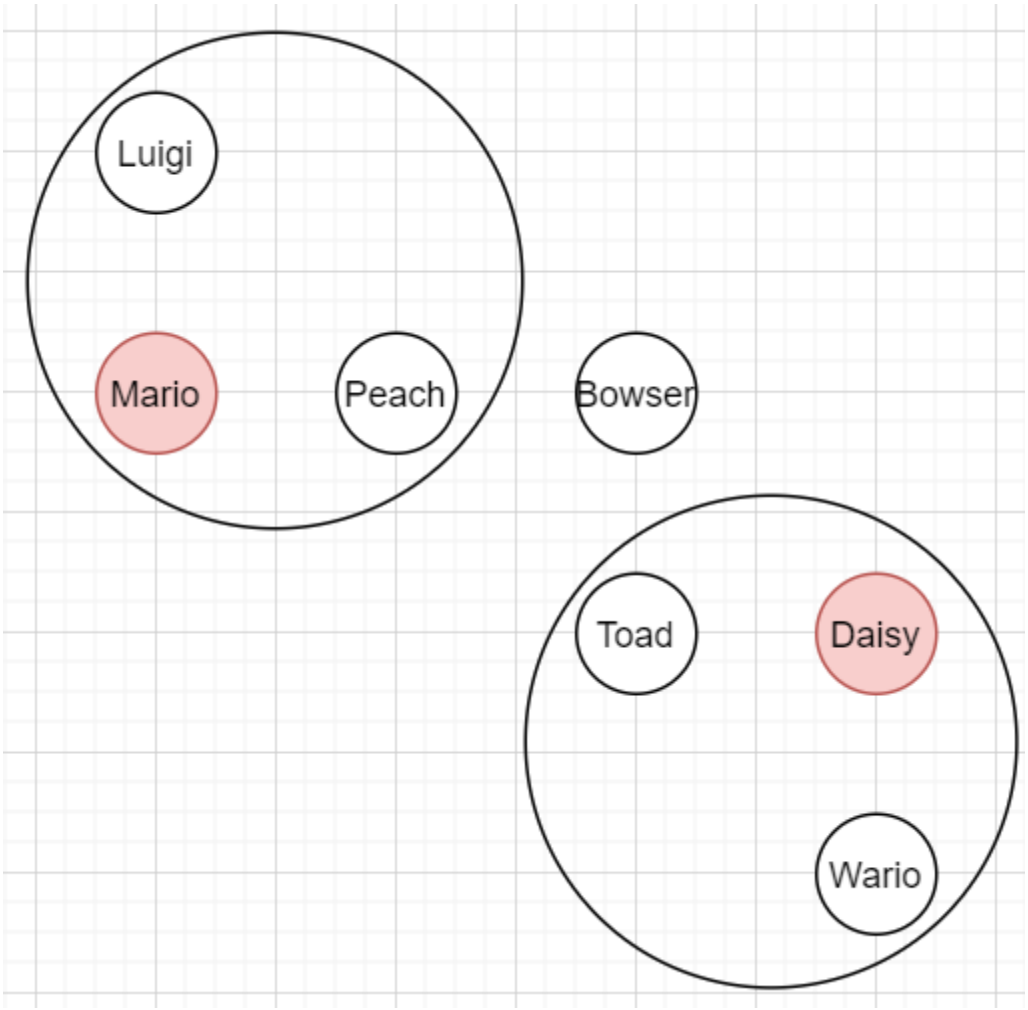
Mauvais amis communs

En ce temps de COVID, il est assez difficile de voir tous ses amis. Avec les nouvelles mesures du gouvernement provincial, il est maintenant rendu légal de voir un maximum de 9 amis (rassemblement de 10) autour d'un BBQ. Dans un monde idéal, chaque personne fait partie d'un seul cercle d'amis. Mais tout le monde connait quelqu'un qui bascule entre plusieurs groupes!

Le problème ici est de trouver toutes ces personnes qui lient des cercles d'amis. Un cercle d'amis est considéré comme les 'a' personnes qui sont le plus près de certains individus dans un plan cartésien. Dans l'exemple du gouvernement, 'a' serait de 9.

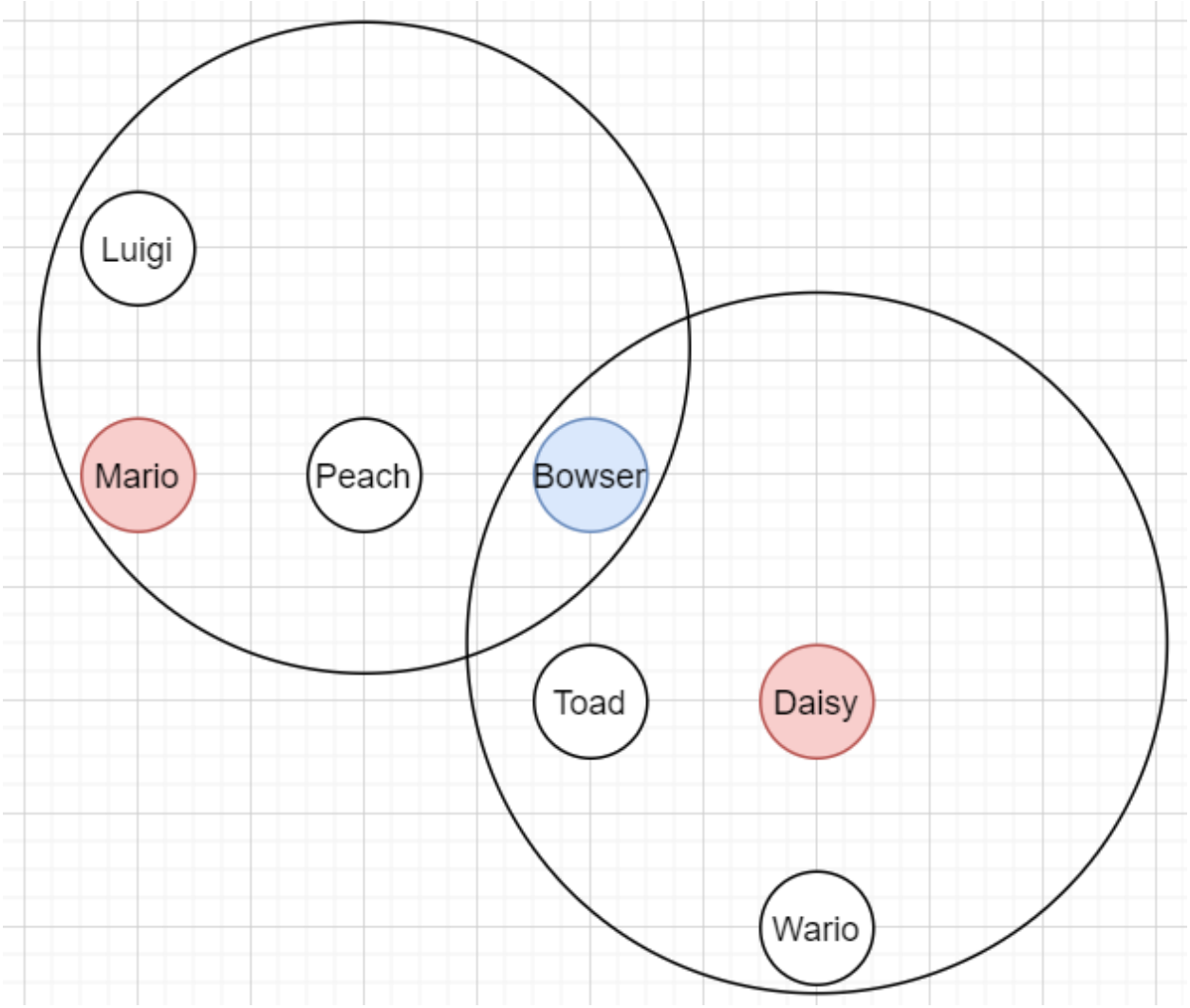
Exemples:

- on vous donne Mario et Daisy comme les centres des cercles d'amis et un 'a' de 2.



Il n'y a aucun amis problématique

- on vous donne Mario et Daisy comme les centres des cercles d'amis et un 'a' de 3.



Bowser devrait être évité!

Evalué votre complexité en fonction de: - 'a' (nombre d'amis) - 'c' (nombre d'individus cibles) - 'n' (nombre d'individus dans la population)

Input Format

- La première ligne contient le 'a'
- La deuxième ligne contient les indices des individus cibles
- La troisième ligne contient les coordonées en x et en y séparé par des virgules.

Constraints

Vous pouvez utiliser toutes les librairies standard de Java. On utilise la distance de Manhattan

- La première ligne: un entier positif
- La deuxième ligne: une liste de 'c' entiers
- La troisième ligne: 'n' paires d'entiers

Output Format

Les indices des amis dangereux séparés par des espaces et en ordre croissant. S'il n'y a aucun ami dangereu, répondre par -1.

Sample Input 0

```
2
1 5
0 0,0 1,1 1,2 1,2 2,3 2,3 3
```

Sample Output 0

```
-1
```

Explanation 0

Ceci représente la première image. A = 2, Mario = 1, Daisy = 5, 0 0 = Coordonnées de Luigi, 0 1 = Coordonnées de Mario, 1 1 = Coordonnées de Peach, etc. Il n'y a aucun lien entre les cercles d'amis, donc -1.

Sample Input 1

```
3
1 5
0 0,0 1,1 1,2 1,2 2,3 2,3 3
```

Sample Output 1

```
3
```

Explanation 1

Ceci représente la deuxième image. A = 3, Bowser = 3. Il lie deux cercles d'amis, donc on le retourne