Université de Bordeaux
Année 2018-2019

Licence Informatique, 3e année

Techniques Algorithmiques et Programmation

TP noté – 2h40

Consignes

Vous avez le droit de consulter les notes de cours [1] sur Internet. L'algorithme du cours est rappelé dans un pdf joint à l'archive. C'est la seule ressource documentaire que vous êtes autorisé à consulter sur Internet. Vous pouvez utiliser vos notes personnelles (TDs, vos programmes, vos notes de cours). C'est une épreuve individuelle, vous n'avez pas le droit de communiquer avec vos voisins (proches ou lointains).

Commencer par télécharger et décompresser l'archive `tp.tgz` depuis Moodle. Puis éditer les premières lignes du fichier `tp.c` pour y indiquer votre NOM/Prénom/Groupe. En fin d'épreuve, vous devrez déposer sous Moodle votre fichier `tp.c`.

La notation prendra en compte :

- la lisibilité de votre code (commentaires)
- l'absence de fuite mémoire
- les performances, testables avec la commande `time tp ...`

Sujet : la paire de points les plus proches

Dans ce TP noté il s'agit de coder des algorithmes calculant la paire de points les plus proches. On utilisera le type prédéfini `point` qui correspond à une paire de `double`, les coordonnées x et y du point. Toutes les coordonnées seront supposées être dans l'intervalle $[0, 1[$. On utilisera aussi le type `paire` qui permettra de renvoyer la paire de points les plus proches (voir le `.h`).

Vous avez à programmer deux algorithmes. Le premier est l'algorithme naïf en $O(n^2)$, où n est le nombre de points, qui cherche la paire de points les plus proches. Le second est l'algorithme récursif vu en cours de complexité $O(n \log n)$. La fonction `dist()` donnant la distance euclidienne entre deux points est déjà écrite. Pour trier les tableaux de points, pensez à utiliser la fonction standard `qsort()` (cf. `man qsort`) avec les fonctions de comparaisons que vous devrez compléter.

Le seul fichier que vous avez à éditer et remettre à la fin du TP est `tp.c` (sans archive donc) avec Nom/Prénom/Groupe complété. Toutes remarques, si vous voulez les partager avec le correcteur, devrait figurer sous forme de commentaires dans le source `tp.c`. Ce programme doit, s'il est exécuté sur la ligne de commande avec un nom de fichier test, permettre d'appliquer un des deux algorithmes.

Le format des fichiers de tests ressemble à ceci :

```
5
0.39507662 0.54847439
0.61524573 0.35726384
0.60721158 0.42669045
0.53734603 0.41907062
0.93016756 0.41819264
```

1. http://dept-info.labri.fr/~gavoille/UE-TAP/cours.pdf

Le `5` indique le nombre de points, et vient ensuite les coordonnées x y des points. Normalement, le résultat de votre programme devrait être (le second argument `naif` ou `rec` spécifie lequel des algorithmes est à utiliser) :

```
> tp p5.txt naif
Point A: 0.61524573 0.35726384
Point B: 0.60721158 0.42669045
Distance: 0.06988993

> tp p5.txt rec
Point A: 0.61524573 0.35726384
Point B: 0.60721158 0.42669045
Distance: 0.06988993
```

La fonction permettant de lire un tel fichier de test (`read()`) est déjà écrite. De même, le `main()` qui vous est fourni permet déjà de lire le fichier, d'exécuter l'un des deux algorithmes et d'afficher le résultat. Vous n'avez donc à programmer que les deux algorithmes demandés, plus d'éventuelles fonctions intermédiaires si bon vous semble. Dans tous les cas, le correcteur doit pouvoir tester vos deux algorithmes en jouant ses propres fichiers de tests.

Il est possible de générer des fichiers de tests aléatoires avec une commande du type :

```
> tp 30000 > p.txt
> cat p.txt
30000
0.90828388 0.38573233
0.65618210 0.31150986
0.74354959 0.97967375
...
```

qui a pour effet de générer `30000` paires de points aléatoires de coordonnées dans $[0, 1[$. Cela vous permettra de tester chacune de vos fonctions, en particulier sur des grands ensembles de points. Vous pouvez bien sûr toujours éditer à la main des fichiers tests particuliers.

À partir d'un certain nombre de points aléatoires, il devient probable que plusieurs points aient la même abscisse. Malheureusement, l'algorithme simplifié présenté dans le cours suppose des abscisses différentes. Il y aura des points bonus pour ceux qui arriveront à étendre l'algorithme pour fonctionne même dans ce cas.