Université de Limoges

Algorithmique et Complexité

TP 3 : Poseur d'affiche

Vous écrirez vos programmes en langage C.

1 Problème du poseur d'affiche

1. Écrivez les fonctions correspondant aux algorithmes `Affiche_Naif` et `Affiche_DpR` vus en TD.
2. Écrivez un programme qui lit une valeur de n sur la ligne de commande, crée un tableau d'entiers de taille n, l'initialise avec des valeurs aléatoires comprises entre 1 et 20, et affiche la surface de la plus grande affiche qu'il est possible de poser sur ce bloc de n immeubles.
3. Introduisez un mécanisme de comptage du nombre d'opérations élémentaires (par exemple les comparaisons) effectuées par chacun des algorithmes.
4. Pour chaque valeur de n spécifiée par l'utilisateur, comptez le nombre d'opérations élémentaires nécessaires en moyenne pour chaque méthode sur r tableaux aléatoires. La valeur r du nombre de runs pourra être prise égale à 100 ou 1000 par exemple.
5. Si votre programme affiche uniquement une ligne de la forme

   ```
   n nb_op_naif nb_op_dpr
   ```

   vous n'aurez aucun mal à créer un fichier texte contenant ces données sur trois colonnes pour des valeurs croissantes de n.
   Remarque : l'avantage de lire la valeur de n sur la ligne de commande est de pouvoir enchaîner facilement diverses exécutions pour différentes valeurs de n. Par exemple :

   ```
   $ for n in $(seq 100 100 10000)
   > do
   > mon_programme $n
   > done
   ```

   que vous pouvez ensuite rediriger dans un fichier texte.

6. Utilisez l'outil de visualisation *gnuplot* pour tracer sur un même graphique les courbes de complexité de chaque méthode :

```
$ gnuplot
gnuplot> plot "data.txt" using 1:2 with lines, "data.txt"
         using 1:3 with lines
gnuplot> quit
$
```

7. Optimisation : trouvez une façon astucieuse de déterminer l'indice du plus petit immeuble, permettant de mieux équilibrer les parties gauche et droite faisant l'objet de la récursivité.

   Comparez avec *gnuplot* également les temps d'exécution moyens avec et sans cette optimisation.