

* Compilation *

Le générateur est déjà compilé (pour Linux), mais si vous avez besoin de le recompiler :

```
cd src
make
```

* Exécution *

```
cd bin
./generator *fichier_sortie*
```

Un menu interactif va alors s'ouvrir, vous allez devoir suivre ce chemin **précis** :

Vous devez d'abord choisir le format de l'output, choisissez **ASCII (0)**, puis entrez une *seed* aléatoire.

```
auguste@DESKTOP-V8ESNEK:/mnt/c/Users/augus/Documents/Preparation_cours/TP2/generator/bin$ ./generator output_file
Welcome to Culberson's Quasi-Random Graph Generator.

This program generates quasi-random k-colorable graphs from various
classes and with different controlling parameters. Some settings may
make the coloring problem hard, some easy. You will be asked for the
class of graphs and for various parameters interactively.

      ENJOY!

Enter output format for the graph(0-ASCII, 1-binary): 0
ASCII
Please enter a seed for the random number generator: 5413|
```

Le programme va alors vous proposer de choisir un type de graphe k-colorable. Choisissez **k-colorable (3)**.

```
Which k-colorable graph?
K-coloring schemes:
  1 No hidden coloring
  2 Equi-partitioned
  3 k-colorable
  4 k-colorable(smooth)
  5 k-colorable with delta variation
Alternate graph:
  6 Flatgraph
?|
```

Il faut ensuite choisir l'ordre du graphe, soit son nombre de sommet. C'est **à vous de choisir** la taille d'instance que vous désirez.

```
The order of a graph is the number of vertices in it.
Please input the order of the graph: |
```

Ensuite il faut choisir le nombre de partitions k. C'est le nombre de couleurs utilisées dans la solution optimale. Vous devez choisir **3** peu importe la taille de votre graphe.

```
The partition number k guarantees that there will be k-coloring
because each partition element will be an independent set.
If k is chosen greater than or equal to 45 then no partitioning
will take place.
Please enter the partition number k: |
```

Le générateur vous demande une variabilité. Choisissez **0**.

```
Enter variability (0-2)
```

Il faut ensuite choisir le type de graphe. Choisissez **IID (1)**.

```
Which graph type?
1 IID (independant random edge assignment)
2 Girth and Degree Inhibited
3 Geometric
4 Weight Biased Graph (encourages or inhibits cliques)
5 Clique driven
6 Cycle driven
?
```

Finalement il faut entrer la densité d'arête du graphe. Choisissez **0.3**.

```
Enter edge probability in percent (0.0-1.0) |
```

Enfin le générateur vous demande s'il faut imprimer la solution dans le fichier de l'instance. Cela ne nous intéresse pas, entrez donc **0 pour non**.

```
Do you want to output the cheat with the graph (0-no)? 0
```

* Format des instances *

Les instances générées sont dans ce format :

```
c DESCRIPTION: Quasi-random coloring problem
c CODE SOURCE: Joseph Culberson (joe@cs.ualberta.ca)
c Specifications:
c   Random seed: 5413
c   k-colorable, 3 partitions, 0 variability
c   Probability: 0.300000
c   random IID graph
c   Degree Information:
c   Min:3 Avg:9.244444 Max:14 Std:2.717479
c COLOR VERIFICATION: Using the permuted order
c under simple greedy yields the specified
c coloring number
c Color = 3 specified partitions = 3
c Creation Date: Wed Nov  3 08:56:49 2021
p edge 45 208
c no cheat
e 6 2
e 6 4
e 7 1
e 7 6
e 9 1
e 11 10
e 12 1
e 12 6
e 12 9
e 13 1
e 13 3
e 13 6
e 14 2
e 14 9
e 14 10
e 14 11
e 16 3
e 16 6
e 16 9
e 17 3
e 17 8
e 17 10
```

Les lignes qui nous intéressent sont :

- la ligne commençant par **p** : **p edge 45 208**

le premier nombre correspond au nombre de sommets, le deuxième correspond au nombre d'arêtes

- les lignes commençant par **e** :

```
e 6 2
e 6 4
e 7 1
e 7 6
e 9 1
e 11 10
e 12 1
e 12 6
e 12 9
e 13 1
```

Elles donnent les arêtes du graphe. Il y a une ligne par arête. Les deux nombres correspondent aux index des sommets reliés par l'arête. Ainsi sur la première ligne de l'image, on voit qu'il y a une arête entre les sommets 6 et 2. **Attention ! Les index commencent à 1 et non à 0.**