OI 题目数据生成工具

1. test.cpp 文件使用说明

注意：`test.cpp` 和 `DataGenerator.h` 以及 `DataGenerator.cpp` 必须放在同一目录下

1.1 文件序号

```
// 文件名起始序号
int startIndex = 1;
// 文件名截止序号
int endIndex = 20;
```

1.2 文件流生成

```
/**
    openOutputFileStream( const string& filePath )
     * @brief 生成输入文件流
     * @param dataFilePath 字符串常量，输入文件路径（封装在 DataGenerator.h 头文件中，根据个人需求修改）
     * @param index 输入文件序号
     * @param inFileType 输入文件类型 ".in"
*/
ofstream writeInputFile = openOutputFileStream( dataFilePath + to_string( index ) + inFileType );

// 和生成输入文件流类似，生成输出文件流，注意输出文件类型 ".out"
ofstream writeOutputFile = openOutputFileStream( dataFilePath + to_string( index ) + outFileType );
```

1.3 输入输出数据容器

```
vector < string > INPUT_DATA; // 多组输入数据
vector < string > OUTPUT_DATA; // 多组输出数据

double OFFSET; // 随机树权重
stringstream ss; // 输出字符串流
stringstream ins; // 输入字符串流


while ( _ -- ) { // 循环生成多组数据
    solve();
    INPUT_DATA.emplace_back( ins.str() ); // 将输入数据存入输入数据容器
    ins.str( "" ); // 清空输入字符串流
    OUTPUT_DATA.emplace_back( ss.str() ); // 将输出数据存入输出数据容器
    ss.str( "" ); // 清空输出字符串流
}
```

2. 数据生成常用工具(具体使用方法请自行跳转函数)

2.1 随机数生成器：`random()`

函数说明

```
function
inline long long random(
    long long l,
    long long r,
    double opt = 1)
  in namespace FESDRER_RAND

在 [l,r] 中随机生成一个整数。
opt 在 [0,+infinity]，其越大，生成的整数越可能接近 r。

DataGenerator.h
```

代码示例

```
void solve() {
    int n = random( 1, 1000 , OFFSET ) ; // 生成一个位于 [1,1000] 的随机数
    ins << n << "\n"; // 将随机数存入输入字符串流
    return;
}
```

2.2 随机字符串生成器：`getRandomString()`

函数说明

代码示例

```
void solve() {
    string s = getRandomString( 10 ,"ABCD" ) ; // 生成一个长度为 10 且由 "ABCD" 四种字母组成的字符串
    ss << s << "\n"; // 将字符串存入到输出字符串流
    return;
}
```

2.3 随机图生成类：`RandomGraph`

类说明

2.3.1 类函数生成合法的点 n 和边数 m：`nm`

```
public method
inline std::vector<int> nm(
    int N,
    int M,
    bool connected,
    bool repeated_edges,
    bool self_rings,
    bool directional,
    double opt = 1)
  in class RANDOMGRAPH
```

生成一个点数不超过 N，边数不超过 M 的点数和边数，以 {点数，边数} 的形式返回。注意 N 和 M 应大于等于 3。
opt 在 [0,+infinity]，其越大，生成的点数边数越接近 N 和 M。
其余的变量是图的若干限制条件，取名直观，不做解释。

DataGenerator.h

2.3.2 类函数检查点 n 和 边数 m 的合法性：`check`

```
public method
inline bool check(
    int n,
    int m,
    bool connected,
    bool repeated_edges,
    bool self_rings,
    bool directional)
  in class RANDOMGRAPH
```

检查点数 n 和边数 m 是否合法。其余的变量是图的若干限制条件，取名直观，不做解释。

DataGenerator.h

2.3.3 类函数生成随机图的边：`graph`

```
public method
inline std::vector<std::pair<int, int>> graph(
    int n,
    int m,
    bool connected,
    bool repeated_edges,
    bool self_rings,
    bool directional,
    bool vertices_rand = 1)
  in class RANDOMGRAPH
```

随机生成一个 n 个点 m 条边的图。
vertices_rand 控制是否要打乱点，若不打乱则返回的每个 pair<int,int> 中 first 总是小于等于 second，即使打乱也不改变 1 号点的编号。
图为有向图且连通时，数据保证从 1 号点可以到达任意一个点。
其余的变量是图的若干限制条件，取名直观，不做解释。

DataGenerator.h

2.4 随机树生成类：`RandomTree`

类说明

2.4.1 类函数生成随机树：`random_tree`

```
public method
inline std::vector<std::pair<int, int>> random_tree(
    int n,
    double rho = 2,
    bool vertices_rand = 1)
  in class RANDOMTREE
```

随机生成一个 n 个点的以 1 为根的树。
rho 在 [0,+infinity]，其越大，树的深度越深，点的度数越小。
可以通过 vertives_rand 控制是否打乱节点编号，打乱后根节点仍然是 1，不打乱则父亲编号永远小于儿子。

DataGenerator.h

2.5 高精度数据结构体：`BigIntTiny`

类说明

特殊数据类型：专门用来处理高精度问题，`OI WIKI` 高精度模板。

3. 其他

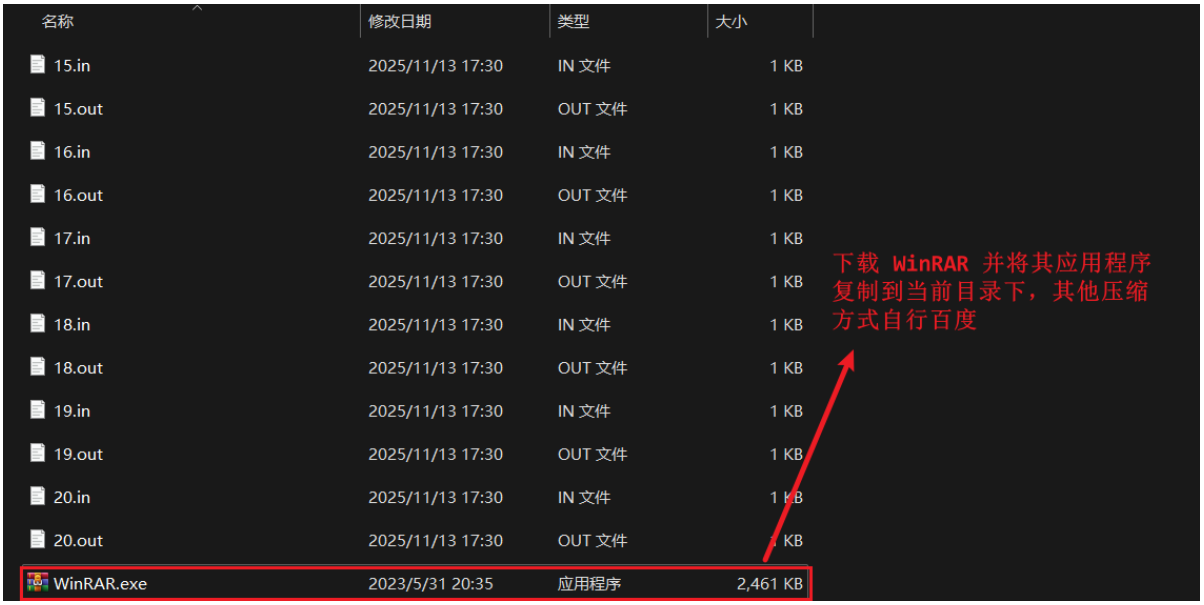
3.1 生成效果图

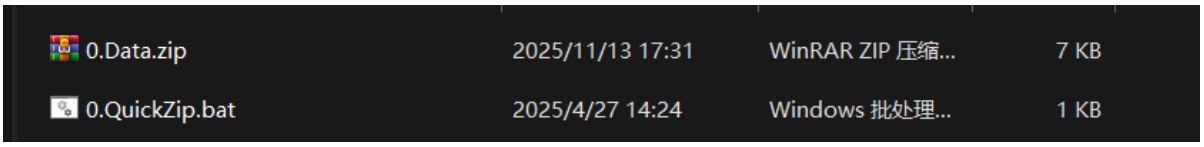
1.in	2025/11/13 17:30	IN 文件	1 KB
1.out	2025/11/13 17:30	OUT 文件	1 KB
2.in	2025/11/13 17:30	IN 文件	1 KB
2.out	2025/11/13 17:30	OUT 文件	1 KB
3.in	2025/11/13 17:30	IN 文件	1 KB
3.out	2025/11/13 17:30	OUT 文件	1 KB
4.in	2025/11/13 17:30	IN 文件	1 KB
4.out	2025/11/13 17:30	OUT 文件	1 KB
5.in	2025/11/13 17:30	IN 文件	1 KB
5.out	2025/11/13 17:30	OUT 文件	1 KB
6.in	2025/11/13 17:30	IN 文件	1 KB
6.out	2025/11/13 17:30	OUT 文件	1 KB

3.2 快捷打包指令

`winrar a 0.Data.zip 1.in 2.in 3.in 4.in 5.in 6.in 7.in 8.in 9.in 10.in 11.in 12.in 13.in 14.in 15.in 16.in 17.in 18.in 19.in 20.in 1.out 2.out 3.out 4.out 5.out 6.out 7.out 8.out 9.out 10.out 11.out 12.out 13.out 14.out 15.out 16.out 17.out 18.out 19.out 20.out`

快捷指令需要在打包目录下添加 `WinRAR.exe` 应用程序。




0.Data.zip
2025/11/13 17:31
WinRAR ZIP 压缩...
7 KB
0.QuickZip.bat
2025/4/27 14:24
Windows 批处理...
1 KB