**Answer Author: Lei Wang (yiak.wy@gmail.clom)**

**Lucky draw**

You'll need to implement a Lucky Draw program using C/C++. Actually, you're not +only required to write a program. Instead, you need to compose a shared library, which exposes only 1 function.

The exported function's declaration looks like below:

*int lucky\_draw (int persons[]);*

'persons' array is a fixed length integer array represent IDs of persons. The length of the array is 100.

The requirements of the lucky\_draw function is below:

A. If calling lucky\_draw() 100 times, it should return a unique ID for each call. Saying, a person cannot be luck twice.

B. Choose a person randomly!

There're 2 extra requirements for this programming task.

1. The shared library should have a unit test using google test. i.e. you should write test for *lucky\_draw* function

For more information, you can check the following link

<https://github.com/google/googletest>

1. The entire project (shared library + unitest) should be organized using CMake

[https://cmake.org](https://cmake.org/)

In summary, I expect to receive the following files in a compressed format.

1. draw.cc and draw.h

2. unitest file

3. CMakeLists.txt, which organizes everything

本次解答，可以参考记录 <https://github.com/yiakwy/Onsite-Blackboard-Code-Interview/blob/master/apollo/cpp/src/modules/solutions/StoneTech.md>

解答：

1 draw.h, draw.cc ：

如无必要，不推荐.h, .cc分开写。项目根地址：<https://github.com/yiakwy/Onsite-Blackboard-Code-Interview/tree/master/apollo/cpp>

安装依赖，可以通过以下方式完成：

> wget https://raw.githubusercontent.com/yiakwy/grpc-gateway/master/scripts/install.sh

会自动执行安装和构建。

执行

》sh scripts/init\\_xcode\\_proj.sh

或者

》 sh scripts/build.sh

会进行编译

解答源文件地址：<https://github.com/yiakwy/Onsite-Blackboard-Code-Interview/blob/master/apollo/cpp/src/modules/solutions/lucky_draw_stone_tech.hpp>

2. unit test

<https://github.com/yiakwy/Onsite-Blackboard-Code-Interview/blob/master/apollo/cpp/tests/gtest/test_lucky_draw_benchmark.cpp>

执行编译，会自动生成gtest测试用例，测试用例结果如下

[==========] Running 1 test from 1 test case.

[----------] Global test environment set-up.

[----------] 1 test from LuckyDrawTest

[ RUN ] LuckyDrawTest.BENCH\_100\_CALLS

[ OK ] LuckyDrawTest.BENCH\_100\_CALLS (0 ms)

[----------] 1 test from LuckyDrawTest (0 ms total)

[----------] Global test environment tear-down

[==========] 1 test from 1 test case ran. (1 ms total)

[ PASSED ] 1 test.

Program ended with exit code: 0

3 CmakeLists.txt

见根目录CMakeLIsts， 以及src/modules/solutions/CmakeLists.txt

**Find a given value in an integral map.**

Integral map is defined as a matrix *I(x,y)* with monotonic increasing row and column, i.e.





Fig. Illustration of an integral map

Prob.1 Develop a less complex algorithm compared with global search, to find a designated value *v* in the map. (Codes) (10 points)

解答：

题目相当于在二维空间检索，并且提供了某种排序。一般我们采用QuadTree来进行二维检索，但是普通QuadTree主要是利用aabb 碰撞，相交来缩小范围。从排序出发，容易得出，左上方，为最小值，右下方为最大值，满足题设不等式。因此，可以通过此不等式来完成4岔树筛选。

主程序先排除不在范围的，对于每个可在范围的节点，都进行检索，返回第一次成功检索结果。

<https://github.com/yiakwy/Onsite-Blackboard-Code-Interview/blob/master/apollo/cpp/src/modules/solutions/integral_map.cpp>

Prob.2 Compare the complexity of the above routine with global search. (5 points)

记矩阵的最大维度为n, 最坏渐进复杂度满足 T（n）= 4\*T(floor(n/4)) + O(1)，其中常数操作，是指，算起每个小格子，最大值（右下角），最小值（左上角）。算法实际平均运行复杂的会比T(n)要好，是因为我们排除了，显然没有落在最小值，最大值之间的格子。

根据主定理方法，假定，T(n) <=c\* nlog(n) 对k <= n成立, 根据xlogx单调性以及 floor(x) <= x，则 T（n）<= c\*4 \* n/4 \* log(n/4) + O(1) = c\*n\*log(n) – c\*n\*llog4+O(1) <= cnlog(n)，因此，可以递归出，T（n）= O(nlog(n)) 要比 一个一个像素搜要快O(n\*n)

**Quicksort and more**

Given that an array of *M* length is constructed employing a random number generator of (0~65535) unsigned type.

Prob.1 Implement a quicksort routine for ascending sorting of the linked list. (Codes) (10 points)

解答

非随机化QuickSort，传统方法，主要是通过交换的方式来将问题，划分为子规模问题。对于链表来说，插入，删除最为容易，本方案，就用两个链表投分别指向以传入表0号位置分界两个子链表。子问题结束后，执行链表合并策略。

<https://github.com/yiakwy/Onsite-Blackboard-Code-Interview/blob/master/apollo/cpp/src/modules/solutions/quick_linked_sort.cpp>

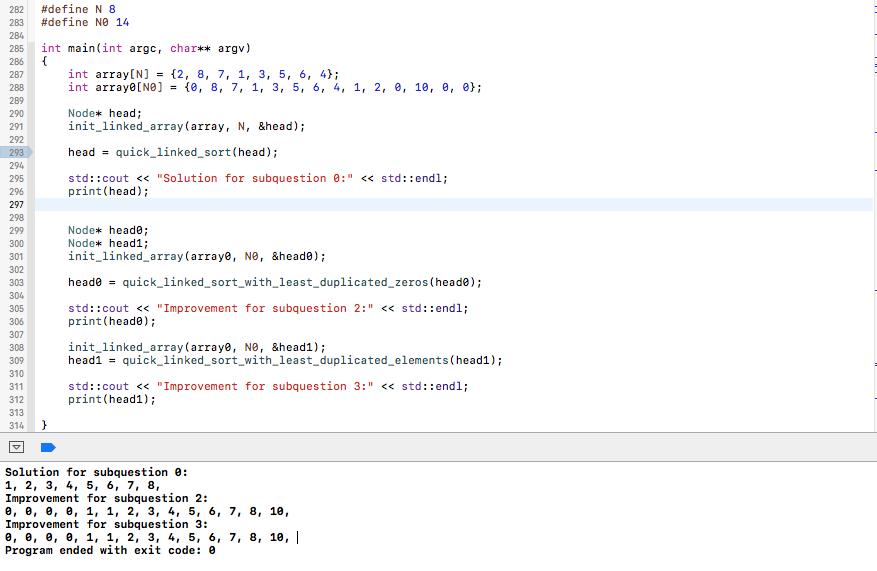
由于随机化技术需要额外遍历，故本方案为加入随机化技术。测试与程序在同一个文件。

Prob.2a Suppose we have the priori knowledge that a decent percentage of number generated is 0, say 25%. Improve the above routine. (2 points)

条件意思是，有大量数重复，且，重复数最小，因此只需要在第一阶段排除大量重复数即可提高排序效率，

请阅读函数： Node\* quick\_linked\_sort\_with\_least\_duplicated\_elements(Node\* head)

输出结果示例：



Prob.2b Suppose we have the priori knowledge of the random number’s PDF, which is not necessarily uniform. Improve the above routine. (3 points)

非均匀分布，意味着，有的值取得概率高，容易重复，因此本方案只需要排除重复数，参与子问题规模即可。

将原表达式，if （curr->val <=）再按等于，和小于区分，等于，smaller指向队列的队尾部；小于插入头部。这样，就将原问题分解成三个区间，

small head -> duplicated （1）

duplicated -> pivot （2）

pivot -> bigger （3）

进行子问题迭代，只需要处理（1）， （3）

末尾，将三个排好序的连接即可。

输出示例同上。