1 Miscellaneous Topics

Let's forget about the theoretical solutions for a while and get hands dirty. In the following, the default language we use is C++.

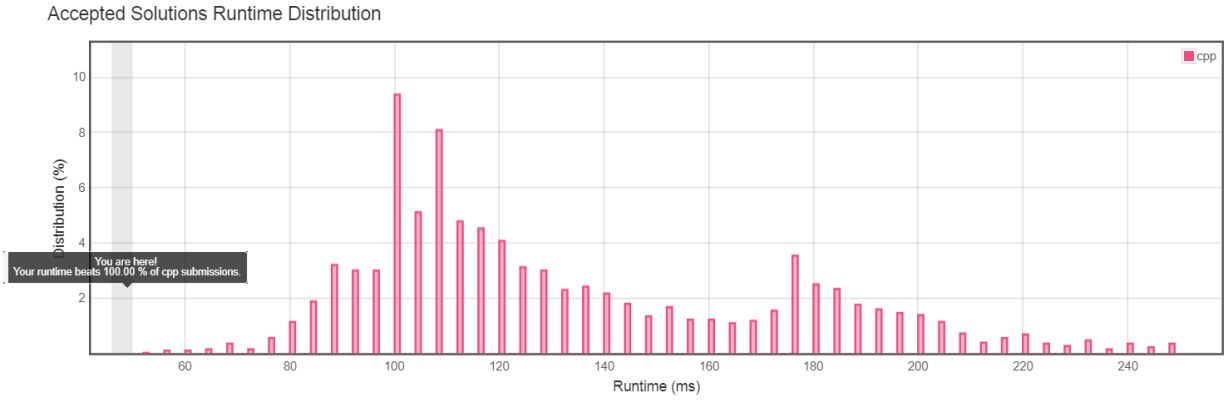
1.1 IO

LeetCode implicitly uses cin for IO when testing your code. To speedup the IO, you can use the following code globally:

```cpp
//IO
int _IO=[](){
    std::ios::sync_with_stdio(0);
    cin.tie(0);
    return 0;
}();
```

1.2 Running Time

The final running time is the total running time over all test cases. If your running time strictly beats 100%, your code will not immediately appear in the bar chart, for example:



1.3 Memory

Only the actually used memory counts, so we may use

```cpp
int cnt[1<<26];
```

at a small cost. Unless specified, the memory limit is 800MB.

1.4 Global Variables

Be careful to initialize them manually.

1.5 Multithreading

LeetCode supports multithreading. For example, see the 292ms code for 318. Maximum Product of Word Lengths. However, there are many (small) test cases, so multithreading usually makes the code slower.

1.6 Template Metaprogramming

LeetCode supports template metaprogramming. However, if the compile time is too long, you will get Compile Error: Compile time limit exceeded. see the 60ms code for 338. Counting Bits.

1.7 Assembly Language

LeetCode supports inline assembly language. see the 20ms code for 307. Range Sum Query - Mutable.

1.8 Compiler Options

LeetCode does not use O2 optimization, and #pragma GCC optimize(2) does not work.

1.9 Runtime Error

The following runtime errors are detected:

- signed integer overflow.
- reference binding to null pointer of type 'int'.

1.10 Random Samples

There are some problems asking you to generate random samples, e.g. see 398. Random Pick Index. It is possible that LeetCode wrote special judges for these problems. If your algorithm behaves too deterministic, usually you'll get Wrong Answer, but due to the nature of these problems, some incorrect (but somewhat random) algorithms may still get Accepted.

1.11 Hack the Online Judge System

When you submit your code, LeetCode enables you to use many powerful functions, so we can do interesting things. For example, we can replace the LeetCode cin IO by our hand-written faster IO, and significantly improve the total running time. For example, see the 4ms code for 307. Range Sum Query - Mutable.

Use the following code to use your own main function and overide the (hidden) main function provided by the judge system:

```
//main
int _main=[](){
    FILE *fout=fopen("./user.out","w");
    //bla bla bla
    exit(0);
    return 0;
}();
```

useful tools in C++:

```
#include<bin/bash>
#define Dbg(x) cout<<"debug: "<<__FUNCTION__<<"() @ "<<__TIMESTAMP__<<"\n"\
        <<__FILE__<<" L"<<__LINE__<<"\n"<<#x" = "<<(x)<<endl
typedef int v4si __attribute__ ((vector_size(16)));
getcwd(buffer,255)
system("cd / && ls -al")
pthread_create(&p[i],NULL,func,(void*)&(ids[i]))
FILE *fout=fopen("./user.stdout","w")
__asm__ __volatile__("test %%eax,%%eax\n")
```

for more information, see template.cpp.

References