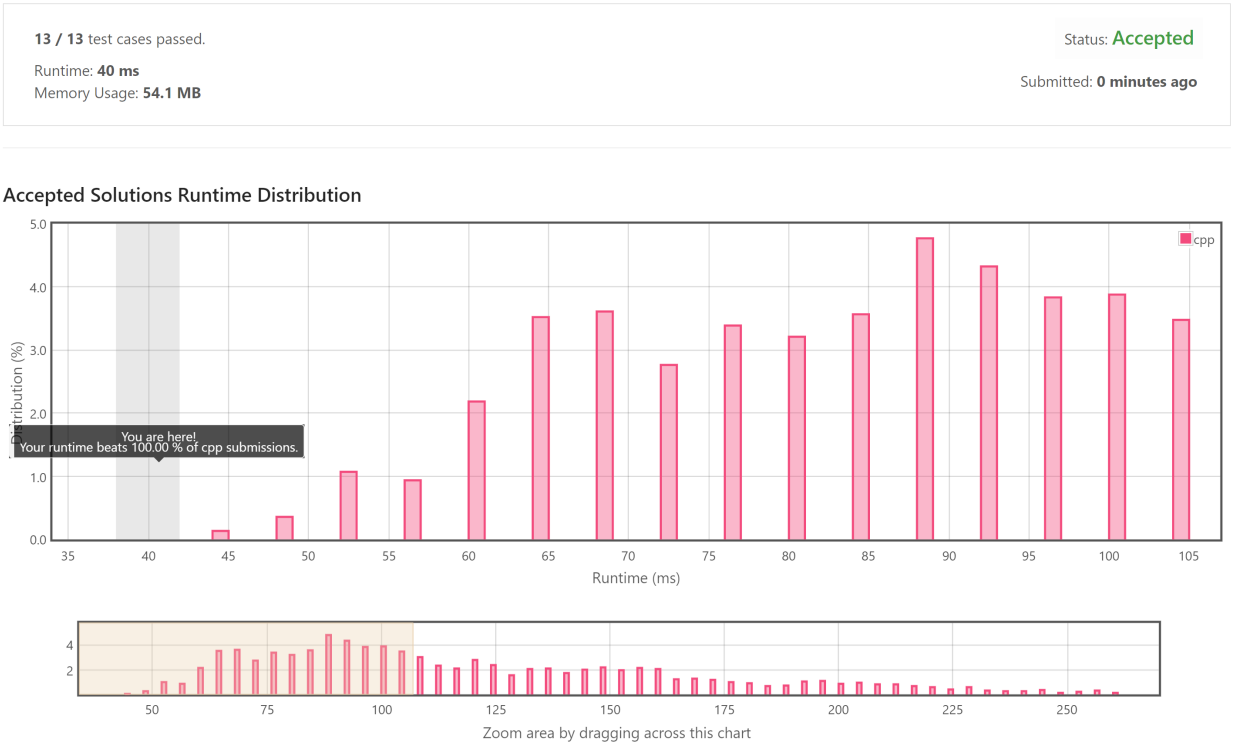
1 Miscellaneous Topics

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 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 runtime distribution bar chart, for example:



And the running time can be so small to disappear on the chart, see Fig. 1.1 for an example.

It is recommended that you put global variables (e.g. large arrays) outside of

```
class Solution {
};
```

, because for each test case the judger will initialize a new Solution instance.

For C++, the displayed running time will include the time spent for reading the inputs. For Java, the running time only counts the time spent within your function (does not count the time for initializing the JVM and reading the inputs). This explains why sometimes Java seems to be faster than C++.

1.2 Memory

Only the actually used memory counts, so we may use

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

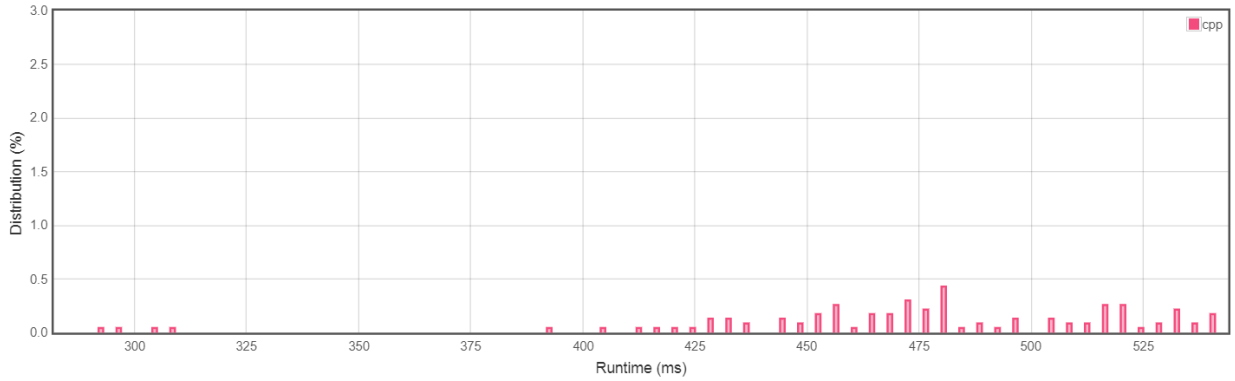
78 / 78 test cases passed.

Status: Accepted

Runtime: **204 ms**
Memory Usage: **106.6 MB**

Submitted: **0 minutes ago**

Accepted Solutions Runtime Distribution



Runtime: 204 ms, faster than 100.00% of C++ online submissions for Avoid Flood in The City.

Memory Usage: 106.6 MB, less than 100.00% of C++ online submissions for Avoid Flood in The City.

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

1.3 IO

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

```
1 //IO
2 int _IO=[](){
3     std::ios::sync_with_stdio(0);
4     cin.tie(0);
5     return 0;
6 }();
```

1.4 Global Variables

Be careful to initialize them manually. Also, there can be multiple test cases.

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.

There are also problems asking for multithreading, e.g. 1195. Fizz Buzz Multithreaded.

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

Compiled with clang 11 using the latest C++ 17 standard.

LeetCode use -O1 optimization, and #pragma GCC optimize(2) does not work.

2021/1/13 update: it uses -O2 optimization now.

1.9 Compile Error

The following compile errors are detected:

- non-void function does not return a value in all control paths.

1.10 Runtime Error

The following runtime errors are detected:

- signed integer overflow.
- left shift of negative value.
- reference binding to null pointer of type 'int'.
- undefined-behavior: passing zero to ctz(), which is not a valid argument.

Also, AddressSanitizer is used to help detect out-of-bounds and use-after-free bugs.

1.11 Code Length

The code size limit is 100KB.

1.12 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.

1.13 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 0ms 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;
}();
```

This also allows you to answer online queries using offline algorithms.

Another way is to use

```
#define main _main
```

and ignore the (hidden) main function. See the 28ms code for 894. All Possible Full Binary Trees.

Useful tools in C++:

```
#include<bin/bash>
#include<semaphore.h>
#include<thread>
#include<unistd.h>
#include<immintrin.h>
#include<xmmintrin.h>
#define main _main
#define Dbg(x) cout<<"debug: "<<__FUNCTION__<<"() @ "<<__TIMESTAMP__<<"\n"\
        <<__FILE__<<" L"<<__LINE__<<"\n"<<#x" = "<<(x)<<endl
__attribute__((target("avx2"))) void f(){}
__attribute__((target("avx512bw")))  //only works for LC-CN
__attribute__((no_sanitize_address,no_sanitize_memory))
typedef int v4si __attribute__ ((vector_size(16)));
getcwd(buffer,255);
system("cd / && ls -al");
thread *t=new thread(func);
fork();
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.

Useful tools in Python:

```
import urllib,urllib2,cookielib
```

Useful tools in Ruby:

```
puts `cat /proc/cpuinfo`
puts `cat /proc/meminfo`
```

1.14 New Test Cases

New test cases will sometimes be added, but may without rejudging the existing Accepted submissions. Therefore, some solution samples on the leaderboard may be incorrect: for example, the fastest code for 1296. Divide Array in Sets of K Consecutive Numbers got Wrong Answer, and the (previous) fastest code for 1488. Avoid Flood in The City got Time Limit Exceeded (78/78 test cases passed, but took too long).

References