20230117题解

这场比赛本来是按照水题大赛的标准来出的，但是好像不是所有人都做的很好。。。

copy

T1大水，真没啥好说的。循环 n 次，每隔 m 次额外输出一句话，单独设一个计数器。

做得很快，一血只用了两分钟。

```
#include <bits/stdc++.h>
using namespace std;
const string S1 = "Plagiarism and duplication of solutions are actions of
the incompetent, and you will be despised.";
const string S2 = "stO carry5307 Orz";

int N, M, x;

int main() {
    ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
    cin >> N >> M;
    for (int i = 1; i <= N; i++) {
        x++;
        cout << S1 << endl;
        if (x == M) {
            cout << S2 << endl;
            x = 0;
        }
    }
    return 0;
}
```

还有赛时才发现 Karry 的名字打错了。。。向卡老师道歉。

file

这个题确实不怎么适合 T2 。反倒是 T4 会更简单。

但是字符串处理确实是很重要的技巧，

两个文件必然可以用如下的格式来表示：

```
/...相同部分.../...不同部分...
/...相同部分.../...不同部分...
```

答案就是相同部分的最后一个文件夹。（题面虽然不清楚但是样例说的很明白）

所以我们要首先找出相同部分，这个并不难，用一个循环扫过去，在第一个不同的位置 `break` 就好。

但是重点是细节：

首先要注意相同部分不一定是以 `/` 为结尾的。比如以下两个文件：

```
/etc/abc/a
/etc/abcc/a
```

我们的相同部分会一直保留到 `/etc/abc` ，但是 `abc` 只是第二个路径中的一部分。所以要减小相同范围，直到相同部分以 `/` 为结尾为止。

然后就简单了。先判一下，如果相同部分只有一个字符，那就输出 `/` ，否则找到末尾两个 `/` 之间的字符串并输出就行。

```
#include <bits/stdc++.h>
using namespace std;

char f1[10010], f2[10010];
int pos;

int main() {
//      freopen("file.in", "r", stdin);
//      freopen("file.out", "w", stdout);
        scanf("%s", f1);
        scanf("%s", f2);
        while (true) {
                if (f1[pos] == f2[pos]) pos++;
                else {
                        pos--;
                        while (f1[pos] != '/') pos--;
                        if (pos == 0) {
                                printf("/\n");
                                break;
                        }
                        pos--;
                        stack<char> s;
                        while (f1[pos] != '/') s.push(f1[pos--]);
                        while (!s.empty()) {
                                putchar(s.top());
                                s.pop();
                        }
                        putchar('\n');
                        break;
                }
```

```
        }
        return 0;
}
```

biteasy

我是没想到一个半小时都没人做这道题。。。看来位运算还是太难。

位运算模板题。我们首先要搞清楚怎么只对某一位做修改。

对于变量 `a` ，如果要取出它从低到高第 i 位，我们可以用 `a & (1 << (i - 1))` 。

体会一下什么意思。

比如 `a` 如果是这样的：

```
00010100110(从低位到高位)
```

`1 << (i - 1)` 得到的是一个第 i 位是1,其他都是0的二进制数。比如 $i = 4$ 时，就是：

```
00010000000
```

这两个数做按位与运算，除了第四位之外肯定都是0,而第四位则取决于 `a` 的第四位。

注意哦，这样取出来的位是保留位值的，我们取出的是 `0001` 也就是8, 而并非1。

在知道如何读取之后，我们还要将答案写入到一个新的变量里。由于每次修改的都是不同的位，我们直接让新变量加上本次的结果就好。当然更好的办法是使用按位或。

细节看代码：

```
#include <bits/stdc++.h>
using namespace std;

unsigned long long a, b, ans;

int main() {
//      freopen("bit.in", "r", stdin);
//      freopen("bit.out", "w", stdout);
        scanf("%llu %llu", &a, &b);
        for (int i = 0; i < 32; i++) {
                //注意这里的i从0开始，奇偶性有变
                if (i % 2 == 0) ans |= (a & (1 << i)) & (b & (1 << i));
                /*
                (a & (1 << i)) 和 (b & (1 << i))分别取出当前位，然后根据奇偶性
进行运算
                最后用 |= 写入答案
                */
```

```
                else ans |= (a & (1 << i)) | (b & (1 << i));
        }
        printf("%llu\n", ans);
        return 0;
}
```

fire

改编自 P1789 【Mc生存】插火把 - 洛谷 | 计算机科学教育新生态 。

这一题和它唯一的区别就是加大了数据范围，并且扩大了火把的照明范围，不得不说其实非常简单，不配当 T4 。

由于数据范围扩大了，所以对于每一格都判断一次是否被照亮就难以通过，我们应该对火把下手：对于每个火把，将它能照亮的格子用一个 `bool` 数组做上标记，最后统计没有做标记的格子有多少就行了。

那么如何快速找到火把能够照亮的格子呢？观察可以发现，火把只能照到距离它的曼哈顿距离小于等于3的格子。所谓曼哈顿距离，就是水平差距与竖直差距的和，或者说平面迷宫中的最小步数。

你可以试试看，在题面给的示例里，确实是到火把的水平距离与竖直距离和小于等于3的点能被照亮。

那么就容易了。对于每个火把，枚举它周围 7x7 范围的所有格子，如果这个格子到火把的曼哈顿距离小于等于3,就说明这个格子能被照亮。显然， 7x7 范围外的格子不可能被照亮。

最后在枚举的时候要注意边界，不要出界。

```
#include <bits/stdc++.h>
using namespace std;

const int maxn = 1e3 + 5;

bool ans[maxn][maxn];
int n, m;

int main() {
//      freopen(".in", "r", stdin);
//      freopen(".out", "w", stdout);
        scanf("%d %d", &n, &m);
        for (int i = 1; i <= m; i++) {
                static int x, y;
                scanf("%d %d", &x, &y);
                for (int ii = max(1, x - 3); ii <= min(n, x + 3); ii++) {
                        for (int jj = max(1, y - 3); jj <= min(n, y + 3);
jj++) {
                                if (abs(ii - x) + abs(jj - y) <= 3) { //
```

曼哈顿距离

```
                                        ans[ii][jj] = true;
                                }
                        }
                }
        }
        int sum = 0;
        for (int i = 1; i <= n; i++) {
                for (int j = 1; j <= n; j++) {
                        if (!ans[i][j]) sum++;
                }
        }
        printf("%d\n", sum);
        return 0;
}
```

sort

排序模板题。排序压根就没学过的就歇菜吧。但是这题我给 $O(n^2)$ 的算法放了70分，所以哪怕用冒泡或者选排也应该有很好的分数。

使用 `std::sort` 就可以通过。但是貌似有人被卡了，看了一下代码，应该是 cin 的原因。五十万的数据量，还是字符串读写，而且每个字符串都有10个字符，cin 确实容易被卡。所以要养成好习惯，不到万不得已的时候不用 cin 。（用 `getchar()` 是可以解决 `string` 读取的问题的）

或者你觉得 cin 太香，那就用 `ios::sync_with_stdio(false);`

排序的实现不是我们的重点，重点是优先级定义，怎么样才能优雅地处理优先级呢?

有的同学在 `cmp` 函数里实现的大量的 `if` ,确实不怎么优雅。我们可以用数组存下三个关键字， `cmp` 就定义为先按照下标 0，再是下标 1，最后是下标 2。但是在读入的时候，我们要按照题目要求的顺序，把三个值存到对应的下标里。

也就是说，首先我们要得到对应关键字所对应的下标，如 `BF:2,DS:1,DM:3` ，然后读入的时候就按照对应的下标读入。使用 `map<string, int>` 可以帮我们优雅地实现这一操作。

如果你看不懂 `map` ，那么用 `if` 也未尝不可。

看代码：

```
#include <bits/stdc++.h>
using namespace std;

const int maxn = 5e5 + 5;

int n;
```

```
struct node {
        char name[12];
        int v[3];
} stu[maxn];

bool compare(node a, node b) {
        if (a.v[0] != b.v[0]) return a.v[0] > b.v[0];
        else if (a.v[1] != b.v[1]) return a.v[1] > b.v[1];
        else return a.v[2] > b.v[2];
}

map<string, int> mp;

int main() {
//      freopen(".in", "r", stdin);
//      freopen(".out", "w", stdout);
        for (int i = 0; i < 3; i++) {
                string s;
                cin >> s;
                mp[s] = i;
        }
        scanf("%d", &n);
        for (int i = 1; i <= n; i++) {
                scanf("%s %d %d %d", stu[i].name, &stu[i].v[mp["BF"]],
&stu[i].v[mp["DS"]], &stu[i].v[mp["DM"]]); //在mp中找到对应的下标
        }
        sort(stu + 1, stu + n + 1, compare);
        for (int i = 1; i <= n; i++) {
                printf("%s\n", stu[i].name);
        }
        return 0;
}
```

顺便贴上可怜的被卡了 cin 的杨叶忆轩的 `if` 代码：

```
#include<bits/stdc++.h>
using namespace std;
struct node{
        string name;
        int a,b,c;
}t[500005];
int n;
int f[5];
bool cmp(node x,node y){
        if(f[1]==1){
                if(x.a!=y.a) return x.a>y.a;
                else{
                        if(f[2]==2){
                                if(x.b!=y.b) return x.b>y.b;
```

```
                                else return x.c>y.c;
                        }
                        else{
                                if(x.c!=y.c) return x.c>y.c;
                                else return x.b>y.b;
                        }
                }
        }
        if(f[2]==1){
                if(x.b!=y.b) return x.b>y.b;
                else{
                        if(f[1]==2){
                                if(x.a!=y.a) return x.a>y.a;
                                else return x.c>y.c;
                        }
                        else{
                                if(x.c!=y.c) return x.c>y.c;
                                else return x.a>y.a;
                        }
                }
        }
        else{
                if(x.c!=y.c) return x.c>y.c;
                else{
                        if(f[1]==2){
                                if(x.a!=y.a) return x.a>y.a;
                                else return x.b>y.b;
                        }
                        else{
                                if(x.b!=y.b) return x.b>y.b;
                                else return x.a>y.a;
                        }
                }
        }
}
int main(){
        char input[5];
        for(int i=1;i<=3;i++){
                cin>>input;
                if(input[0]=='B') f[1]=i;
                else if(input[1]=='S') f[2]=i;
                else f[3]=i;
        }
        scanf("%d",&n);
        for(int i=1;i<=n;i++){
                cin>>t[i].name;
                scanf("%d%d%d",&t[i].a,&t[i].b,&t[i].c);
        }
        sort(t+1,t+n+1,cmp);
```

```
        for(int i=1;i<=n;i++) cout<<t[i].name<<endl;

        return 0;
}
```