题目讨论

chenguo

问题1：字典序最大的子序列

A 字典序最大的子序列

时间限制：- MS　空间限制：- KB

评测说明：1s 256MB

问题描述

给定字符串s，s只包含小写字母，请求出字典序最大的子序列。

注: 子序列中的字符在原序列中的位置不一定连续

输入格式

一行一个字符串$s(1 <= |s| <= 100,000)$。

输出格式

字典序最大的子序列。

样例输入 1

```
ababba
```

样例输出 1

```
bbba
```

样例输入 2

```
abbcbccacbbcbaaba
```

样例输出 2

```
cccccbba
```

问题1：字典序最大的子序列

思路讨论：

栈的应用。

倒着考虑，如果当前字符大于等于栈顶元素则压入栈中，最后依次弹出栈中元素即为答案。

可以发现答案字符串的字符的字典序是单调不增的。

问题1：字典序最大的子序列

参考代码：

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

char Str[100003];
int main() {
    scanf("%s", Str);
    int Len = strlen(Str);
    char Temp = Str[Len - 1];
    stack<char> S1;
    for(int i = Len - 1; i >= 0; i--) {
        if(Temp <= Str[i]) {
            S1.push(Str[i]);
            Temp = Str[i];
        }
    }
    while(!S1.empty()) {
        printf("%c", S1.top());
        S1.pop();
    }
    printf("\n");
    return 0;
}
```

问题2：漂亮的树

P5613 漂亮的树

时间限制：- MS　空间限制：- KB

评测说明：1s 256MB

问题描述

南渝校园内有n棵树，标号为$1...n$，第i棵树的高度为a_i。如果这n棵树满足以下两个条件，果老师认为这些树是漂亮的:

1. 对于所有的i，$a_i = a_{n-i+1}$；
2. 对于$1 <= i < n/2$(不是整除)，$a_{i+1} = a_i + 1$；

比如说"2 3 4 5 5 4 3 2"和"1 2 3 2 1"是漂亮的而"1 3 3 1"和"1 2 3 1"不是。

现在请问最少修改几棵树的高度（可以变大也可以变小），使得这些树是漂亮的。

输入格式

第一行一个整数n表示树的数量（$1 <= n <= 100,000$）。

第二行n个整数表示树的高度（$1 <= a_i <= 100,000$）。

输出格式

输出一个整数表示最少修改树的高度的数目。

样例输入 1

```
3
2 2 2
```

样例输出 1

```
1
```

样例输入 2

```
4
1 2 2 1
```

样例输出 2

```
0
```

问题2： 漂亮的树

思路讨论：

最少需要修改的树的高度的数目不好求，
我们尝试思考问题的反面，

怎么求最多不需要修改的数目？

问题2： 漂亮的树

思路讨论：

最多不需要修改的数目 =
下标减下标对应的值出现的次数最多的值就是要求的不需要改变的元素
即a[i] – i

但是这个相减出来的值有可能为负，所以统计的时候都加上一个maxn

对于后半段镜像处理即可。

问题2：漂亮的树

参考代码：

```cpp
#include <bits/stdc++.h>
using namespace std;
int n, temp;
const int maxn = 1e5 + 5;
int a[maxn];
int cou[maxn * 2]; //用来存储i-a[i]的个数数组
int main() {
    cin >> n;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }
    memset(cou, 0, sizeof(cou));
    int temp = (n + 1) / 2;
    for (int i = 1; i <= temp; i++) {
        cou[a[i] - i + maxn]++;
    }
    for (int i = temp + 1; i <= n; i++) {
        cou[a[i] - (n - i + 1) + maxn]++;
    }
    int ma = 0;
    for (int i = 1; i <= 2 * maxn; i++) {
        ma = max(ma, cou[i]); //不需要改变的最多的个数
    }
    cout << n - ma << endl;
    return 0;
}
```

问题3：加点

C 加点

时间限制：- MS　空间限制：- KB

评测说明：1s 256MB

问题描述

平面上有若干个点，从每个点出发，你可以往东南西北(右下左上)任意方向走，直到碰到另一个点，然后才可以改变方向。请问至少需要加多少个点，使得任意两点之间都互相可达。

输入格式

第一行一个整数n表示点数（$1 <= n <= 100$）。第二行n行，每行两个整数x_i, y_i表示坐标（$1 <= x_i, y_i <= 1000$）。y轴正方向为北，x轴正方向为东。

输出格式

输出一个整数表示最少需要加的点的数目。

样例输入 1

```
2
2 1
1 2
```

样例输出 1

```
1
```

样例输入 2

```
2
2 1
4 1
```

样例输出 2

```
0
```

问题3： 加点

思路讨论：

方法1: BFS广度优先搜索，从第一个点开始进行搜索，将所有可以连到的点都标记出来，然后求出所有互相不连通的个数，即在n个小连通图之间加上n-1条边使得n个小连通图连通。

方法2: 并查集求最少需要插入的边的数量让所有节点都连通

问题3： 加点

参考代码：

```
const int maxn = 1000 + 100;
int n;
struct Node {
    int x, y;
};
vector<Node> vec;
bool f[maxn];
queue<int> Q;
void bfs(int x) {
    f[x] = 1;
    Q.push(x);
    while (!Q.empty()) {
        int i = Q.front();
        Q.pop();
        for (int j = 0; j < n; j++) {
            if (f[j] == 0) {
                if (vec[i].x == vec[j].x || vec[i].y == vec[j].y) {
                    f[j] = 1;
                    Q.push(j);
                }
            }
        }
    }
    return;
}
```

```
int main() {
    cin >> n;
    for (int i = 0; i < n; i++) {
        Node now;
        cin >> now.x >> now.y;
        vec.push_back(now);
    }
    memset(f, 0, sizeof(f));
    int ans = 0;
    for (int i = 0; i < n; i++) {
        if (f[i] == false) {
            bfs(i);
            ans++;
        }
    }
    cout << ans-1 << endl;
    return 0;
}
```

问题4：讨厌奶牛

D 讨厌奶牛

时间限制：- MS　空间限制：- KB

评测说明：1s 256MB

问题描述

自从果老师做了大量的奶牛题之后，他只要看到奶牛就会恶心。

于是"cow"成为了果老师的禁忌，而长得和"cow"很像的"owc"…凡是同时含有"c","w","o"的都进入了他的禁忌名单。

何老板想给他送一幅幅长为n个字符的长诗，但是又怕触犯他的禁忌。所以他决定要是诗中出现了他的禁忌就宁可不送，可是...他一写起诗来就忘了一切。

何老板想知道他有多少种的诗可能不触犯他的禁忌 注：何老板只会用小写字母写诗

输入格式

一行一个整数n表示诗的长度

对于30%的数据满足 $1 \leq n \leq 5$

对于100%的数据满足$1 \leq n \leq 10^5$

输出格式

一行一个整数表示何老板有多少种可能的诗不触犯果老师的禁忌，由于可能数也许过大，请对$10^9 + 7$取模后输出

样例输入

```
3
```

样例输出

```
17570
```

提示

$n = 3$且包含"c","o","w"的只有6个串，所以答案是26^3-6=17570

问题4：讨厌奶牛

思路讨论：

动态规划

dp[i][j] 前i个字符出现了 j **种** 禁忌字符的方案数

所以
dp[i][0] = (dp[i-1][0] * 23)
dp[i][1] = (dp[i-1][0] * 3 + dp[i-1][1] * 24)
dp[i][2] = (dp[i-1][1] * 2 + dp[i-1][2] * 25)

问题4：讨厌奶牛

参考代码：

```
int main()
{
    cin>>n;
    dp[1][0]=23;dp[1][1]=3;dp[1][2]=0;
    for(int i = 2; i<=n; i++) {
        dp[i][0] = (dp[i-1][0] * 23)%mod;
        dp[i][1] = (dp[i-1][0] * 3 + dp[i-1][1]*24)%mod;
        dp[i][2] = (dp[i-1][1] * 2 + dp[i-1][2]*25)%mod;
    }
    printf("%lld\n",(dp[n][0]+dp[n][1]+dp[n][2])%mod);
    return 0 ;
}
```

问题5：选数

E 选数

时间限制：- MS　空间限制：- KB

评测说明：1s 256MB

问题描述

果老师给定你n个数，要求你从中选出三个数，使得最大的那个减最小的那个的值小于等于d，问有多少种选法。

输入格式

第一行两个整数 n , $d(1 <= n <= 100,000$, $1 <= d <= 1000,000,000)$ ； 第二行n个整数a_i满足$abs(a_i) <= 1,000,000,000$。数据保证$a$单调递增。

输出格式

输出一个整数表示满足条件的选法。

样例输入 1

```
4 3
1 2 3 4
```

样例输出 1

```
4
```

样例输入 2

```
4 2
-3 -2 -1 0
```

样例输出 2

```
2
```

问题5：选数

思路讨论：

升序排序后枚举每一个数字做三个数的最后一个数(最大的那个数)，
然后找出第一个大于等于(枚举数-d) 的数，
从这个数开始到枚举数的前一个数任选两数皆可满足要求，用组合数公式计算即可。

这个题还有线性复杂度的做法
(*^__^*) 嘻嘻……

问题5：选数

参考代码：

```
long long n, d;
long long a[maxn];
long long sum;

int main() {
    scanf("%lld %lld", &n, &d);
    sum = 0;
    for (int i = 1; i <= n; ++i) {
        scanf("%lld", &a[i]);
    }
    for (ll i = 1; i <= n; ++i) {
        int pos = lower_bound(a + 1, a + i, a[i] - d) - a;
        if (i - pos > 1) {
            sum += (i - pos) * (i - pos - 1) / 2;
        }
    }
    printf("%lld", sum);
    return 0;
}
```

问题6：果老师的饮料

问题描述

果老师得到了n瓶饮料。但他不小心把开盖的工具弄丢了,所以他只能利用饮料瓶来开盖。 已知第i个瓶子的品牌为a_i,且其能打开b_i品牌的瓶子。 问有几瓶饮料果老师无法喝到。
注:被用于打开饮料瓶的瓶子不一定需要被打开。一个瓶子不能打开其本身。

输入格式

第一行一个整数n,表示饮料的瓶数.

接下来n行,每行两个整数

$1 \leq n \leq 100$

$1 \leq a_i, b_i \leq 1000$

输出格式

输出一行一个整数,表示果老师无法喝到的饮料瓶数.

样例输入 1

```
4
1 1
2 2
3 3
4 4
```

样例输出 1

```
4
```

问题6：果老师的饮料

思路讨论：

开瓶盖其实只要满足第i瓶的品牌a_i等于第j瓶的品牌a_j能打开的品牌b_j，即$a_i=b_j$且(i!=j)，果老师就能喝到第i瓶饮料，计数器m就加1，最后n-m即为果老师不能喝到的饮料数量。

问题6：果老师的饮料

参考代码：

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef unsigned long long ll;
const int maxn = 2e6 + 5;
int drink[10005], bei[maxn];
bool can[10005];
int main() {
    int n;
    cin >> n;
    int u, v;
    for (int i = 0; i < n; ++i){
        cin >> u >> v;
        drink[i] = u;
        can[v] = 1;
        bei[v] = i;
    }
    int cnt = 0;
    for (int i = 0; i < n; ++i){
        if (can[drink[i]] && bei[drink[i]] != i) ++cnt;
    }
    cout << n - cnt << endl;
    return 0;
}
```

问题7：填数字

问题描述

何老板发现了一种新的游戏--填数字!

每填写一次数字($1 \leq i \leq 9$)需要花费a_i枚金币,何老板总共有n枚金币。

何老板想知道他能得到的最大数字是多少。如果填不了请输出-1。

注:不需要用完所有金币

输入格式

第一行一个数字n,表示金币总数.

第二行9个正整数,第i个数字表示填写一次数字i所需要的金币数.

$0 \leq n \leq 10^6$ $1 \leq a_i \leq 10^5$

输出格式

输出满足条件的最大数字.

样例输入 1

```
5
5 4 3 2 1 2 3 4 5
```

样例输出 1

```
55555
```

问题7：填数字

思路讨论：

先找出花费最小的那个求出最大的位数，然后由高位到低位依次替换为较大的数字。

问题7：填数字

参考代码：

```cpp
#include <bits/stdc++.h>
using namespace std;
int a[101];
int main() {
    int n;
    scanf("%d", &n)
    int minn = 1000010;//表示最小花费
    for(int i = 1; i <= 9; i++) {
        scanf("%d", &a[i]);
        minn = min(minn, a[i]);
    }
    int t = n / minn;//要保证最多的位数不变
    if(t == 0) printf("-1\n");
    else {
        while(t--) {
            for(int i = 9; i >= 1; i--) {
                if(n - a[i] >= minn * t) {
                    cout << i;
                    n = n - a[i];
                    break;
                }
            }
        }
        cout << endl;
    }
    return 0;
}
```

问题8：最大乘积

问题描述

何老板给你一个正整数S，若有若干个正整数的和为S，则这若干的数的乘积最大是多少？请输出答案除以2000000000000000003 (共有17 个零) 的余数。

输入格式

输入的第一行有一个正整数 T，代表该测试数据含有多少组询问。

接下来有T行，每个询问各占 1 行，包含 1 个正整数，代表该询问的 S 值。

$1 \leq T \leq 100$ $1 \leq S \leq 2000$

输出格式

对于每个询问，请输出答案除以 2000000000000000003 (共有17个零) 的余数。

样例输入

```
10
1
2
3
4
5
6
7
8
9
100
```

样例输出

```
1
2
3
4
6
9
12
18
27
7412080755407364
```

问题8：最大乘积

思路讨论：

尽量拆分多的3 不足3的时候 拆分为2 但是 一定不能含有1 如果有1 则就不是最大乘积了
当剩余4 时 拆分为 2 2
举个例子
n = 5 时 拆为 2 3
n = 6时 3 * 3 > 2 * 2 * 2

问题8：最大乘积

参考代码：

```
#include <bits/stdc++.h>
using namespace std;
const long long a = 2000000000000000003;
int t,s;
int main() {
    scanf("%d", &t);
    while(t--) {
        long long sum = 1;
        scanf("%d", &s);
        while(s > 4) {
            sum *= 3;
            sum %= a;
            s -= 3;
        }
        sum *= s;
        sum %= a;
        printf("%lld\n", sum);
    }
    return 0;
}
```

问题9：玩游戏

问题描述

何老板和果老师聚在一起玩游戏，这次他们选择在一个$n * m$的棋盘上玩游戏。

棋盘上的每个方格都有一个非负的分数，游戏从左上角开始右下角结束，双方交替的选择一个方格并获得方格上相应的分数，一方选择的方格必须在上一步另一方选择的方格的右边或者下面，何老板先开始。

现在何老板想知道，如果双方都采取最优策略(最优策略是指双方都希望最终自己的总分数减去对方的总分数最大)，他的总分数减去果老师的总分数会是多少？

输入格式

第一行一个整数T表示数据的组数。$(1 \leq T \leq 20)$

对于每组数据：

第一行两个整数n，m表示棋盘的规格。$(1 \leq n, m \leq 500)$

接下来n行每行m个整数aij表示方格对应的分数。$(0 \leq a_{ij} \leq 10000)$

输出格式

对于每组数据输出一行表示答案。

样例输入

```
1
2 2
1 3
4 5
```

样例输出

```
2
```

问题9： 玩游戏

思路讨论：

动态规划
定义dp[i][j]表示先手从第(i,j) 位置开始取，到右下角结束时先手得分减去后手得分的最大值。
那么后手显然会选择dp[i+1][j]和dp[i+1][j] 中的较大值转移，进而

dp[i][j] = a[i][j] – max(dp[i + 1][j], dp[i][j + 1])

从后往前转移, dp[n][m] = $a_{n,m,}$ dp[1][1]即为答案。 时间复杂度O(nm)

问题9：玩游戏

部分参考代码：

```
if(i==n && j==m)
{
    if((i+j)%2==0)
        dp[i][j] = a[i][j];
    else
        dp[i][j] = -a[i][j];
}
else if(i==n)
{
    if((i+j)%2==0)
        dp[i][j] = dp[i][j+1]+a[i][j];
    else
        dp[i][j] = dp[i][j+1]-a[i][j];
}
else if(j==m)
{
    if((i+j)%2==0)
        dp[i][j] = dp[i+1][j]+a[i][j];
    else
        dp[i][j] = dp[i+1][j]-a[i][j];
}
else
{
    if((i+j)%2==0)
        dp[i][j] = min(dp[i+1][j], dp[i][j+1])+a[i][j];
    else
        dp[i][j] = max(dp[i+1][j], dp[i][j+1])-a[i][j];
}
```

问题10：走格子

评测说明：1s 256MB

问题描述

在平面上有$n*n$大小的正方形，定义正方形左下角坐标是$(1,1)$，右下角坐标是$(n,1)$

现在果老师在左下角，他的初始方向是向右，他要在正方形内走m步 当果老师碰到边界或者已经走过的格子时，他便会逆时针转90°继续走，直到走完m步。

现在给你两个整数n和m，请算出走完m步后果老师的坐标。

输入格式

输入一行两个整数n和m。

$n <= 1000$, $m < n*n$

输出格式

输出一行两个数表示果老师的坐标。

样例输入

```
3 3
```

样例输出

```
3 2
```

问题10： 走格子

思路讨论：

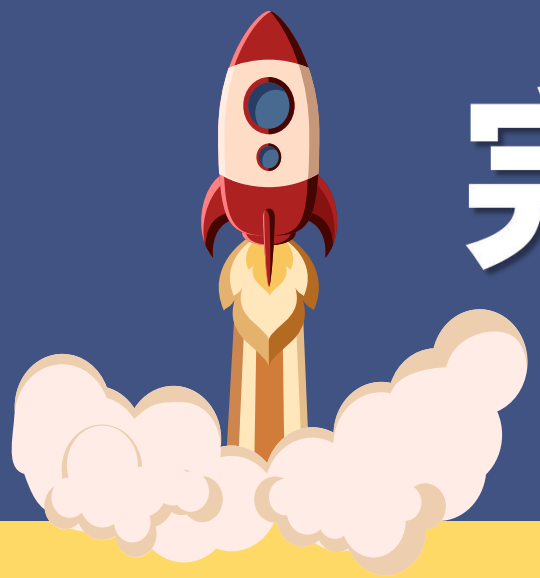
按题意模拟即可。

思考如果实现逆时针转90度

问题10：走格子

参考代码：

略



完！

以梦为码 心之所往