新生杯题解

A A×B problem

时间限制：1000ms，空间限制：256mb

Description

对两个整数A和B，令整数$S=A\times B$，令$f(x)$为十进制数x的各个位数字的和，比如$f(125)=1+2+5=8$，求$f(f(f(f(f(S)))))$。

Input

一行一个整数$A(0\leq A<10^{100000})$

一行一个整数$B(0\leq B<10^{100000})$

题解

法1：**数学**

注意到这个A和B都很大，long long 都装不下，所有我们肯定不能直接输入然后乘。

对于C++选手，显然只有字符串才能装下一个整数A和B。

对于数字各位和，有这样一个性质：

令n为数字本身，m为n的各位和，有

$$n\equiv m\quad (mod 9)$$

证明：设a为一个整数，可以表示为$\sum a_i\times 10^i$

那么在$(mod 9)$下，有$\sum a_i\times 10^i=\sum a_i\times(10^i-1)+\sum a_i\equiv\sum a_i$。就证完了

接下来证明题目中$f(f(f(f(f(S)))))$一定是个位数

由A和B的范围得到$S(0 \le S < 10^{200000})$

放缩$S \le 999\ldots999$（200000个9），则$f(S) \le 1800000$

放缩$f(S) \le 9999999$，则$f(f(S)) \le 63$

放缩$f(f(S)) \le 99$，则$f(f(f(S))) \le 18$，则$f(f(f(S))) \le 9$

则$f(f(f(f(f(f(S))))))$一定是个位数

再根据整数的取模四则运算可得，令$T = (sum(A)\%9) * (sum(B)\%9)\%9$

其中$sum(x)$表示x的各位和

小细节：只要S不是0，$f(f(f(f(f(f(S))))))$一定不会是0

所以$f(f(f(f(f(f(S)))))) = (T == 0?9 : T)$

小细节2：如果A是0或者B是0，那么结果就是0

复杂度：$O(|A| + |B|)$

参考代码：

```
    string a,b;
    cin>>a>>b;
    if(a=="0"||b=="0"){
        cout<<0;return 0;
    }
    int aans=0,bans=0;
    for(char&c:a)
        aans+=c-'0';
    for(char&c:b)
        bans+=c-'0';
    int ans=(aans%9*bans%9)%9;
    if(ans==0)cout<<9;
    else cout<<ans;
```

法2：FFT或者NTT模拟，其中FFT或NTT可以在O(nlogn)的时间内做多项式乘法

法3：python直接模拟

法4：压位高精度暴力乘法（这个解法存在纯属是因为出题人数据出弱了）

cp供题

B AC !

时间限制：1000ms，空间限制：256mb

Description

定义“好串”： 对于一个字符串，选择其中连续的子串“ac”或者“cp”并删除之，删除后左右子串拼接起来（比如字符串“helloacm”删除中间的“ac”之后就是”hellom“），重复上述操作，如果能删完，我们称原字符串为“好串”。

给定一个符文串s，由于能量消耗的限制，你可以执行操作k次：

- 在s的任意位置插入任意字符。

如果能够通过k次操作把s变为好串，输出最小的k，否则输出“No”

Input

第一行一个正整数 n（$1 \leq n \leq 300$)

第二行一个长度为n的字符串s，其中s只包含小写字母

题解

法1： **区间dp**

设dp状态：

$dp[i][j]$表示下标在$[i,j]$的子串的答案(最少操作次数使之变成好串)

初状态：

$dp[i][j]=0,if(j<i)$

$dp[i][j]=inf,if(j\geq i)$

状态转移：

$dp[i][j]=dp[i+1][j-1],if(s[i]+s[j]==“cp”或“ac”)$

$dp[i][j]=min(dp[i][k],dp[k+1][j]),k\in[i,j)$

答案：

$dp[1][n]$即为所求

复杂度：$O(n^3)$

注意如果字符串里不是只含有a、c、p三种字符的，可以直接输出“No”

参考代码：

```
bool check(char a,char b){
    if(a=='a' && b=='c') return true;
    if(a=='c' && b=='p') return true;
    return false;
}
int n;char s[310];
int main(){
    cin>>n>>(s+1);
    for(int i=1;i<=n;i++)
        if(s[i]!='a' && s[i]!='c' && s[i]!='p') {
            cout<<"No"; return 0;
        } //特判字符集
    for(int i=1;i<=n+1;i++)
        f[i][i-1]=0, f[i][i]=1;
    for(int len=2;len<=n;len++){
        for(int i=1,j=i+len-1; j<=n; i++,j++){
        // if (s[i]+s[j]=="ac"||"cp") f[i][j]=f[i+1][j-1];
        if(check(s[i],s[j])) f[i][j]=f[i+1][j-1];
        else f[i][j]=min(f[i+1][j],f[i][j-1])+1;

        for(int k=i; k<j; k++)
            f[i][j]=min(f[i][j], f[i][k]+f[k+1][j]);
        }
    }
    cout<<f[1][n];
}
```

法2：**反悔贪心**

```
因为ac和cp都能匹配，所以我们可以看成
a = (
c = )或者[
p = ]
设tmp表示已经匹配好的()数量（用于反悔），
用cnt1,cnt2表示 多余“(” 和 多余“[” 的数量，ans表示答案。

1.出现一个a，cnt1++
2.出现一个c,if(cnt1>0) {cnt1--; tmp++;}（视为“)”） else cnt2++;（视为“[”）
3.出现一个p，if(cnt2>0) cnt2--; else if(tmp>0) {tmp--; cnt1++;} else ans++;
4.O(n)扫完一遍之后，ans+=(cnt1+cnt2)

```

```
int n; char s[310];
int cnt1=0,cnt2=0,tmp=0,ans=0;
int main(){
    cin>>n>>(s+1);
```

```
    for(int i=1;i<=n;i++)
        if(s[i]!='a' && s[i]!='c' && s[i]!='p') {
            cout<<"No"; return 0;
        } //特判字符集
    for(int i=1;i<=n;i++){
        if(s[i]=='a') cnt1++;
        else if(s[i]=='c'){
            if(cnt1>0) cnt1--,tmp++;
            else cnt2++;
        }
        else if(s[i]=='p'){
            if(cnt2>0) cnt2--;
            else if(tmp>0) tmp--, cnt1++;
            else ans++;
        }
    }
    ans+=(cnt1+cnt2);
    cout<<ans;
}
```

复杂度：$O(n)$

cp供题

C M1917美国制史密斯维森

时间限制：1000ms，空间限制：256mb

Description

ACMon的故事发生在一个自由的国度，这一次，小◯来到了一个神秘的道馆，里面正在进行这么一个游戏——游戏中的n个人都配备一把M1917美国制史密斯维森（一款左轮）：1到n号玩家围成一个圈，轮到第i号玩家时，该玩家可以考虑是否向下一位玩家使用M1917美国制史密斯维森，若使用，该玩家获得一枚金币，且下一位玩家淘汰，然后轮到下一位玩家进行游戏。

规定一开始第i号玩家的下一位玩家是第$i+1$号玩家，第n号玩家的下一位玩家是第1号玩家。若某玩家下一位玩家淘汰了，则该玩家的下一位玩家改为淘汰的玩家的下一位玩家。游戏从1号玩家开始，向下一位依次进行。

玩家们首先希望保证自己不被淘汰，其次希望得到更多的金币。

玩家们都采用最佳策略。

请问最后还剩多少玩家没有被淘汰，以此输出这些玩家的序号。

题解

法1：**队列模拟**

不难发现，3个人是平衡态，到3个人的时候，谁都不敢开枪

设置一个队列 1 1 1 表示三个人都存活

新来了一个人，如果他开枪，接下来他就会成为3人队列中的最后一位，是存活的，所以他会开枪， 所以四人就是 1 0 1 1

新来了一个人，如果他开枪，接下来他就会成为4人队列中的最后一位，是存活的，所以他会开枪， 所以五人就是 1 0 1 0 1

新来了一个人，如果他开枪，接下来他就会成为5人队列中的最后一位，是存活的，所以他会开枪， 所以六人就是 1 0 1 0 1 0

新来了一个人，如果他开枪，接下来他就会成为6人队列中的最后一位，是会去世的，所以他不会开枪， 所以七人就是 1 1 1 1 1 1 1

于是，开一个双端队列，每加一个人，考察队列右端点的状态

1，如果右端点是1，就右边弹出一个，然后左边push一个0，再push一个1

2，如果右端点是0，就全部弹出，然后push n个1

复杂度：$O(n)$

法2：**找规律**

找规律可以发现 平衡局面场上一定会是2^k-1个人，所以开局一路开枪，直到剩2^k-1个人即可

复杂度：$O(n)$

参考代码：

```
int n;
cin >> n;
int ans = 1;
while (2*ans+1<=n) ans=2*ans+1;
cout << ans;
for (int i = 1; i <= n; i++) {
  cout << i;
  if (ans < n) {
     ans++; i++;
  }
}
```

cp供题

D 学号

时间限制：1000ms，空间限制：256mb

Description

学号的格式为“CPxxxxyyyyzzzzzzzzzzzzzz”，“xxxx”是入学年份，

输入学号，输出入学年份

Input

一个长度为24的字符串，表示小◯的学号

题解

签到题 （题解略）

cp供题

E 最小半圆覆盖

时间限制：1000ms，空间限制：256mb

Description

在xOy二维平面中有n个点，请你判断是否存在一个以原点为圆心的半圆能够覆盖所有点，如果存在，请输出最小整数半径，如果不存在，请输出-1。

其中如果点在半圆的边界，也算覆盖。

Input

一行一个整数$n(1 \leq n \leq 2 \times 10^3)$

接下来n行，每行两个以空格隔开的两个整数$x, y(-10^6 \leq x, y \leq 10^6)$

题解

法1：**暴力**

枚举每一个点和原点连线，然后遍历所有点，看看是不是在这条线的一侧

对于(0,0)点，需要特殊处理，包括一个(0,0)点还是很多(0,0)点，还是全是(0,0)点

复杂度：$O(n^2)$

法2：**极角排序**

按照极坐标的角度排序，如果存在下一个点到上一个点的角度差$\geq$ 180°，那就可以，否则不行。

也要特殊考虑(0,0)点问题

复杂度：$O(nlogn)$

cp供题

F 山移公愚

时间限制：1000ms，空间限制：256mb

Description

有一个长度为n数组 a，如果存在一个$i(1 < i < n)$满足对于任意$1 \leq j < i$,有$a_j < a_{j+1}$，同时 对于任意$i \leq j < n$,有$a_j > a_{j+1}$，我们称数组a为凸数组。

cp家族掌握一种魔法：可以任意选择一个$i(1 < i < n)$，使得$a_i := a_{i+1} - a_i + a_{i-1}$。

用最少次数的魔法使得数组a变为凸数组。如果做不到，输出-1

Input

第一行一个正整数$n(3 < n < 10^5)$

第二行n个整数a_i表示数组a，$(-10^8 \leq a_i \leq 10^8)$

题解

差分

求出a数组的差分数组b，可以发现对i使用魔法其实就是交换b_i和b_{i+1}，而我们的目标就是使得b数组前面全是正数，后面全是负数，输出最小操作次数。

对于b数组，如果b恒正或者恒负，或者存在0，都是不可能达到目的的

复杂度：$O(n)$

参考代码：

```
const int N = 2e5 + 20;
int t_, n, a[N];

signed main() {
    std::cin >> t_;
    while (t_--) {
        std::cin >> n;
        for (int i = 1; i <= n; i++)
            std::cin >> a[i];
        for (int i = n; i; i--)
            a[i] -= a[i - 1];
        int cnt0 = 0, cnt1 = 0, cnt2 = 0;
```

```
        for (int i = 2; i <= n; i++) {
            cnt0 += (a[i] == 0);
            cnt1 += (a[i] > 0);
            cnt2 += (a[i] < 0);
        }
        if (cnt0 > 0 or cnt1 == 0 or cnt2 == 0) {
            std::cout << "-1\n";
            continue;
        }
        i64 ans = 0;
        for (int i = 2, j = 1; i <= n; i++)
            if (a[i] > 0)
                ans += (i - (++j));
        std::cout << ans << "\n";
    }
}
```

cp供题

G zyx的快乐水

时间限制：1000ms，空间限制：256mb

Description

对于一个数x，最多$k(k \geq \log_{10} x)$次机会去交换x的十进制数位，请输出可能的最大的x

Input

第一行一个正整数，表示x ： $1 \leq x \leq 10^{1000}$

第二行一个正整数，表示k ： $1 \leq k \leq 10^8$且$k \geq \log_{10} x$

题解

排序

x很大，只能用字符串存

可以发现因为$k \geq \log_{10} x$

那么我们一定可以在k次操作以内做到x按十进制位降序排列

于是我们直接排序即可。

本题数据范围很小，支持$O(n^2)$的冒泡排序或者插入排序，或者$sort()$

zyx供题

H 石子小游戏

时间限制：1000ms，空间限制： 256MB

Description

石子摆放为数组a，表示第i堆有a_i个石子

对于a，选取到 n 堆石子中的第 i 堆$(1 < i < n,\ a_i \geq 2)$，在该堆石子上取出两个，然后随机选取一个数字 $k(k > 0,\ i - k \geq 1,\ i + k \leq n)$并把这两个石子分别放在第 $i - k$ 和 $i + k$ 两堆上面，然后重复这种方式随机操作了多次，注意每次操作选取的 i 不一定是相同的，得到新的石堆排列 $\{b\}$。

对于a，随机选取 n 堆中的第 i 堆$(3 \leq i \leq n,\ a_i \geq 2)$，在该堆石子上取出两个，然后把两个石子随机的放在排在第 i 堆之前的不同的两堆上，然后重复这种方式随机操作了多次，注意每次操作选取的 i 不一定是相同的，并且这里的多次不一定等同于上面的多次，又得到一个新的石堆排列 $\{c\}$。

对于a，随机选取 n 堆石子中的第 i 堆$(1 \leq i \leq n-2,\ a_i \geq 2)$，在该堆石子上取出两个，然后把两个石子随机的放在排在第 i 堆之后的不同的两堆上，然后重复这种方式随机操作了多次，注意每次操作选取的 i 不一定是相同的，并且这里的多次不一定等同于上面的多次，又得到一个新的石堆排列 $\{d\}$。

保证操作后的各个排列中每堆石头数量都不会超过 10^5，现在 $\{b\}$、$\{c\}$、$\{d\}$ 三个石堆排列分别给出。请问哪一堆对应的 $\{b\}$，哪一堆对应的 $\{c\}$，哪一堆对应的是 $\{d\}$

题解

对于每个不同的摆法，考虑函数

$$f(n) = \sum_{i=1}^{n} i \cdot w_i$$

对于第一种操作

就是把中间的某个堆 a_i 数量减 2，然后再把根据这一堆左右对称的两个堆 a_{i-k} 和 a_{i+k} 分别加 1，即修改一次后的函数为
$f^{'}(n) = f(n) - i \times 2 + (i-k) + (i+k) = f(n)$，函数值不变。

对于第二种操作

修改一次后则是 $f^{''}(n) = f(n) - i \times 2 + j + k$，其中 $j, k < i$，则
$f^{''}(n) < f(n)$;

对于第三种操作

修改一次后是 $f'''(n) > f(n)$。

由于三种操作改变的方向各不相同，并且至少操作一次，所以通过计算输入的每个排列对应的函数值，做一次排序便可确定分别是由什么操作修改得到的。

复杂度：$O(n)$

参考代码：

```
struct node {
    int id;
    long long val=0;
    bool operator<(const node &a)const {
        return val<a.val;
    }
} arr[3];
int main() {
    char ans[3],mp[3]= {'c','b','d'};
    int n; cin>>n;
    long long tmp;
    for(int i=0; i<3; i++) {
        arr[i].id=i;
        for(int j=0; j<n; j++) {
            cin>>tmp;
            arr[i].val+=tmp*j;
        }
    }
    sort(arr,arr+3);
    for(int i=0; i<3; i++)ans[arr[i].id]=mp[i];
    for(int i=0; i<3; i++)cout<<ans[i]<<" ";
    return 0;
}
```

njk供题

I 最小公倍数

时间限制：4000ms 空间限制：256MB

Description

对于 整数n （$1 \leq n \leq 5000$），计算

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} lcm(i,j,k)$$

显然答案会很大，你需要将总能量对 2^{32}取模，再输出

- 其中$lcm(i,j,k)$表示i,j,k三个数的最小公倍数

Input

一行，一个正整数 n （$1 \leq n \leq 5000$）

题解

数学

新生杯题目之最小公倍数的题解

zdq供题

J 拆图

时间限制：2000ms 空间限制：128MB

Description

给定一个N个节点，M条边的无向图，可以进行一种操作：

- 每次操作选取一条没有被删掉的边(u,v,w)，然后该次操作的开销是：$w\times$删掉该边之后联通块的数量

  其中，u,v: 代表边的两个端点、w：代表该边的权重。

  GuaGua队需要删掉m条边，使得删完整个图的开销最小。

Input

第一行两个整数：N,M 保证 $N\leq 5\times 10^5$且$M\leq 5\times 10^5$

之后M行，每行三个整数：$u_i,v_i,w_i\ (1\leq u_i,v_i\leq n,1\leq w_i\leq 10^6)$表示一条边的两个点和删除该边时的$w$权重。可能有重边，可以保证答案不会大于$10^{18}$。

题解

最小生成树、并查集、倒序贪心

这道题结合了相对模板的并查集使用方法和相对套路的倒序贪心方式，简单考察选手思维和数据结构能力。

题解如下：

拆图的过程不好分析，我们考虑倒着来做这道题，即把边重新加上去的过程。

（1）我们先考虑一棵树的解法：

因为一棵树**每加一条边都是使得连通块数量减1(性质1)**，所以我们容易想到把边权从小到大排序，再按顺序加进去。这样倒着贪心就可以求出拆掉一棵树的开销。

（2）我们再来考虑一个图的解法：

图相对于树来说，**性质1**不再成立了。但是我们仍然可以解决这个问题。我们仍然是先把所有的边按边权从小到大排序，每次加一条边会出现以下两种情况：

1. 加入这条边，使得连通块数量减1：
   - 那么按照贪心的思路，加入这条边是合理的。因为在当前状态下，能够使得连通块数量减1且边权最小的就是当前这条边。

2. 加入这条边，但是连通块数量不减1：
    - 此时我们没必要加入这条边。我们把这条边留到最后加，就能够使得这条边的开销最小。

综合上面的分析过程来看，这个其实就是最小生成树的一个模板题？使用并查集来判断联通性就可以了。

时间复杂度是: $O(n\alpha + nlogn)$, 空间复杂度: $O(n)$ 。其中 α是一个很小的常数，是并查集Find过程的常数。

参考代码：

```
struct Edge {
  int u, v, w;
  bool operator<(const Edge& rhs) const { return w < rhs.w; }
};
int n, m, fa[maxn];
vector<Edge> edge;
int Find(int x) { return x == fa[x] ? x : fa[x] = Find(fa[x]); }
int main() {
  // input
  cin >> n >> m;
  edge.resize(m);
  for (auto& e : edge) cin >> e.u >> e.v >> e.w;
  // pre work
  for (int i = 1; i <= n; i++) fa[i] = i;
  sort(edge.begin(), edge.end());
  // main work
  long long ans = 0, cnt = n;
  for (const auto& e : edge) {
    int x = Find(e.u), y = Find(e.v);
    if (x == y) {
      ans += e.w;
    }
    else {
      fa[x] = y;
      ans += 1ll * e.w * cnt;
      cnt -= 1;
    }
  }
  // output
  cout << ans << endl;
}
```

gjq供题

K 好多好多 cp

时间限制：1000ms 空间限制： 256MB

Description

对于一个字符串，按以下方式计算 `cp` 的出现次数：

- 只要某个字母 `p` 出现在了某个字母 `c` 之后，无论这两个字母相隔多远，都视为出现 1 次。

对于一个字符串s，有n次询问，询问字符串s的$[l,r]$区间中`cp`出现的次数

题解

前缀和+dp预处理

首先考虑一个简单的版本：询问子串中 `c` 的数量。

使用前缀和。记 $cnt_c\left(i\right)$ 为前 i 个字符中 `c` 的数量。询问 $[l,r]$ 中 `c` 的数量，$cnt_c\left(r\right)-cnt_c\left(l-1\right)$ 即为所求。

然后考虑另外一个简单的版本：询问 $[1,r]$ 中 `cp` 这一子序列的数量。使用类似前缀和的思想。记 $cnt_{cp}\left(i\right)$ 为前 i 个字符中 `cp` 这个子序列的数量，有 $cnt_{cp}\left(i\right)=cnt_{cp}\left(i-1\right)+cnt_c\left(i-1\right)*[s_i=\mathrm{p}]$。

$[s_i=\mathrm{p}]$ 的意思是：第 i 个字符是 `p` 的话，这一块的值为 1，否则为 0。

题目问的是 $[l,r]$ 中 `cp` 子序列的数量。可以用 $[1,r]$ 中 `cp` 的数量减去多计算的部分。

多计算的部分可以分为两类：

一类是 `cp` 都在 $[1,l-1]$ 中，可以直接从前面的预处理结果中得到；

另一类是 `c` 在 $[1,l-1]$ 中，`p` 在 $[l,r]$ 中，可以从前面前缀和的预处理结果得到数量。

$ans=cnt_{cp}\left(r\right)-cnt_{cp}\left(l-1\right)-cnt_c\left(l-1\right)\times\left(cnt_p\left(r\right)-cnt_p\left(l-1\right)\right)$

总时间复杂度为 $\mathcal{O}\left(n+|s|\right)$

注意答案会爆 `int`，需要使用 `long long`。

参考代码：

```
const int N = 2e5 + 10;
```

```
char s[N];
int c[N], p[N], cp[N];

int main() {
  int q;
  cin >> q >> &s[1];
  int n = strlen(&s[1]);
  for (int i = 1; i <= n; i++) c[i] = c[i - 1] + (s[i] == 'c');
  for (int i = 1; i <= n; i++) p[i] = p[i - 1] + (s[i] == 'p');
  for (int i = 1; i <= n; i++) cp[i] = cp[i - 1] + c[i] * (s[i]
== 'p');
  while (q--) {
    int l, r;
    cin >> l >> r;
    l--;
    cout << cp[r] - cp[l] - c[l] * (p[r] - p[l]) << '\n';
  }
}
```

lyz供题

L 世界...遗忘我

时间限制：3000ms 空间限制： 256MB

Description

有n个点在一条直线上，找出m个点使得这m个点中距离最近的2个点之间的距离最大。

题解

二分答案。

设定左端点为0,右端点为最大值。二分出一个答案之后判定答案是否可行。

如果最短距离可以满足在n个点中找出m个符合要求的点的集合，则该距离可行，更新答案，左端点变为答案值继续二分；

如果最短距离不满足，则该距离不可行，右端点变为答案值继续二分。

复杂度：$O(nlog(10^8)+nlogn)$

参考代码：

```
const int N = 1E5+7;
int T,n,m,l,r,mid,a[N];
bool check(int x){
    int cnt=1; int last=1;
    for(int i=2;i<=n;i++){
        if(a[i]-a[last]>=x){
            cnt++;
            last=i;
        }
    }
    if(cnt>=m) return 1;
    return 0;
}
int main(){
    cin>>T;
    while(T--){
        cin>>n>>m;
        for(int i=1;i<=n;i++) cin>>a[i];
        sort(a+1,a+n+1);
        l=0;r=a[n];
        int ans;
```

```
        while(l<=r){
            mid=(l+r)/2;
            if(check(mid)){
                ans=mid;
                l=mid+1;
            }
            else r=mid-1;
        }
        cout<<ans<<endl;

    }
    return 0;
}
```

gzy供题