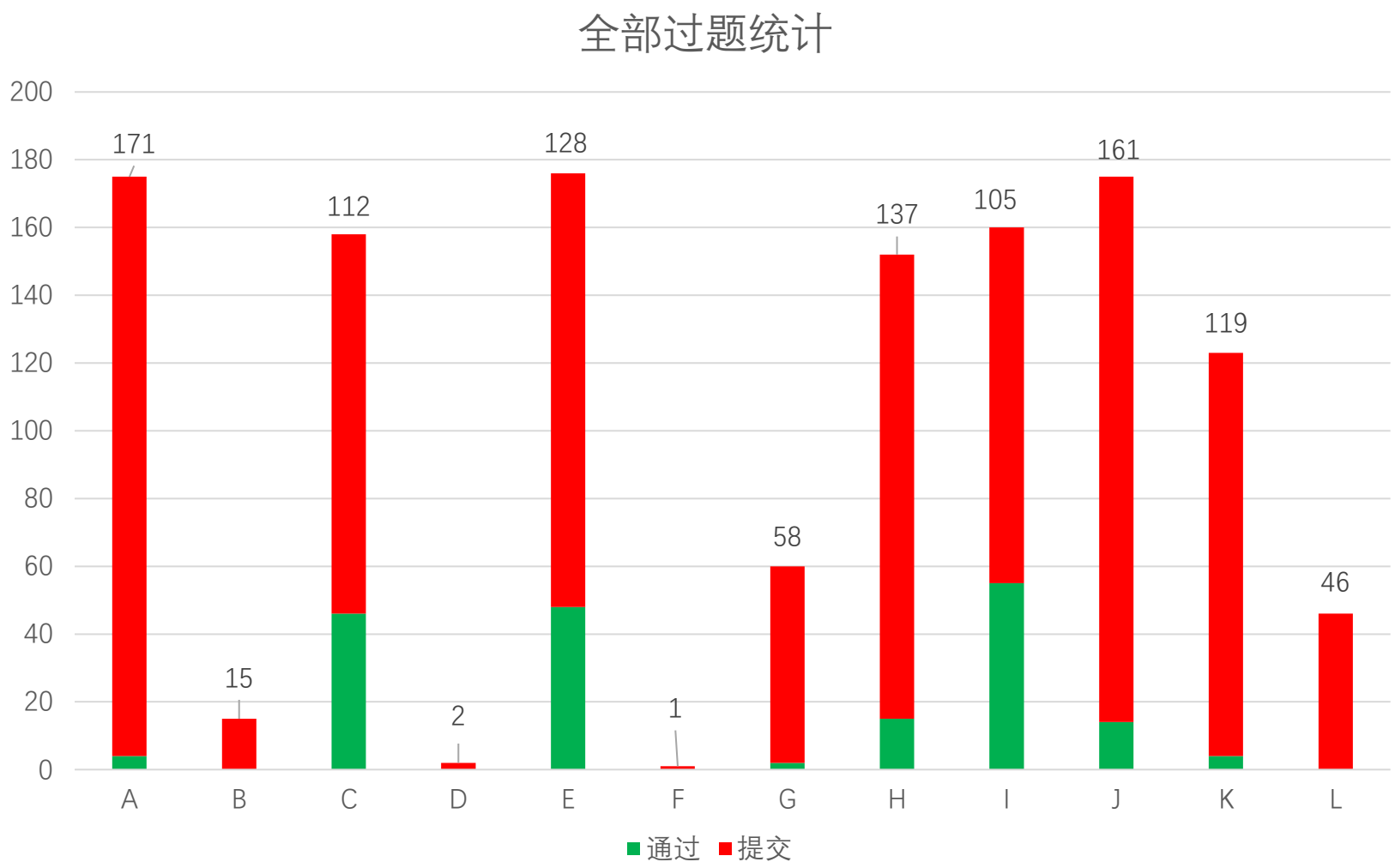
第四届『小白杯』ACM程序设计竞赛新生赛题解

HGNUACM集训队 & HBUE IT协会

2024/12/1

过题统计


全部过题统计
200
180
160
140
120
100
80
60
40
20
0
171
15
112
2
128
1
58
137
105
161
119
46
A
B
C
D
E
F
G
H
I
J
K
L
通过
提交

过题统计

	通过/提交
A	4/171
B	0/15
C	46/112
D	0/2
E	48/128
F	0/1
G	2/58
H	15/137
I	55/105
J	14/161
K	4/119
L	0/46

难度预测

- Easy：E，C，H，I
- Mid：A，G，B，K，J，D
- Hard：L，F

还有机会吗

题意

- 直接输出黄冈师范学院ACM实验室的简称

- 直接按样例复制粘贴输出即可
- 注意最后的感叹号是中文字符的感叹号

还有机会吗

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    cout<<"HGNUACM! "<<'\n';
    return 0;
}
```

防AK

题意

- 将序列倒序输出

- for循环正序读入到数组中，倒序输出

防AK

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

int main()
{
    int n;
    cin>>n;
    int a[n+10];
    for(int i=1;i<=n;i++){
        cin>>a[i];
    }
    for(int i=n;i>=1;i--){
        cout<<a[i]<<" ";
    }
    return 0;
}
```

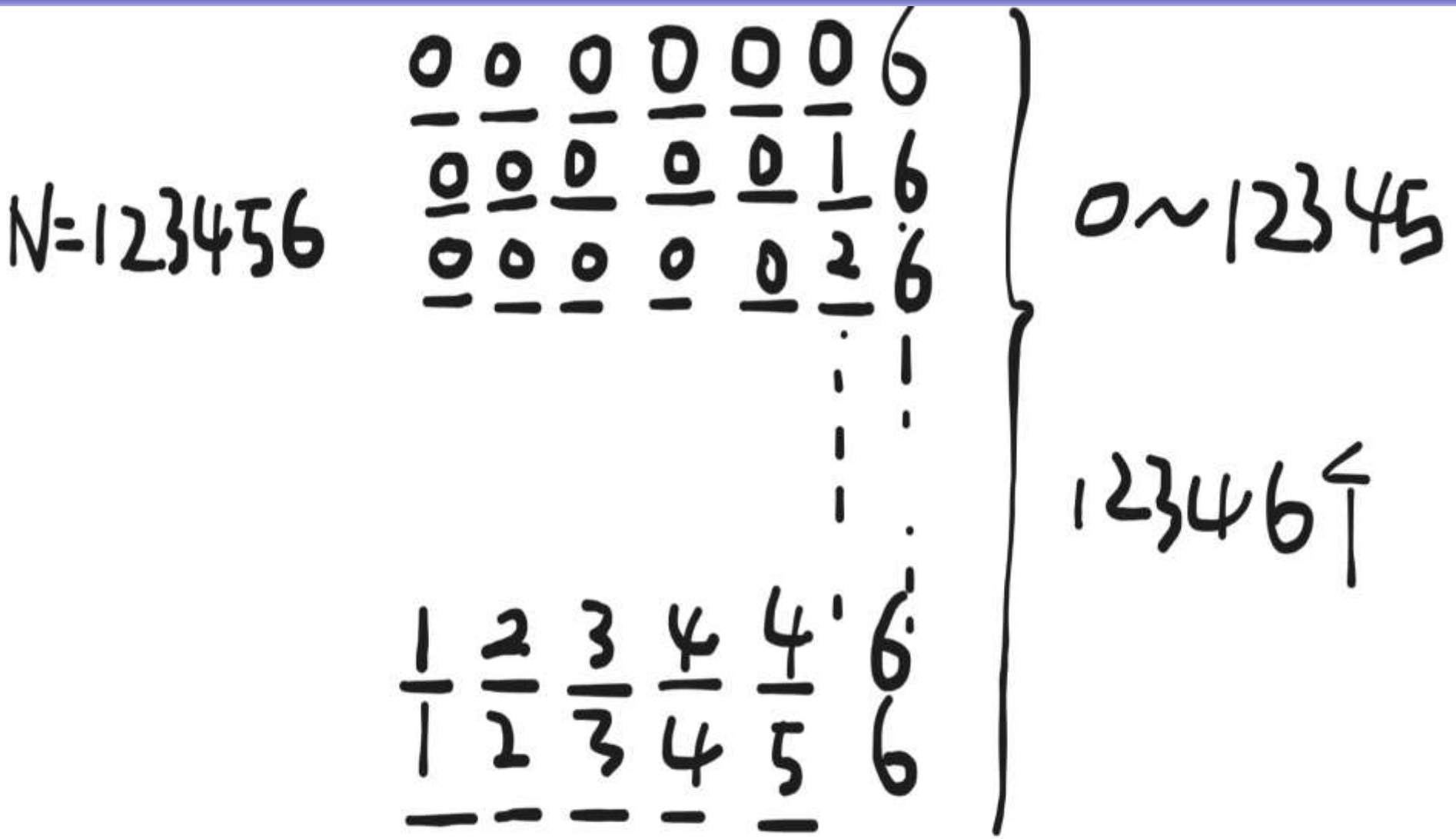
谁出的这题，出题人给我过来

题意

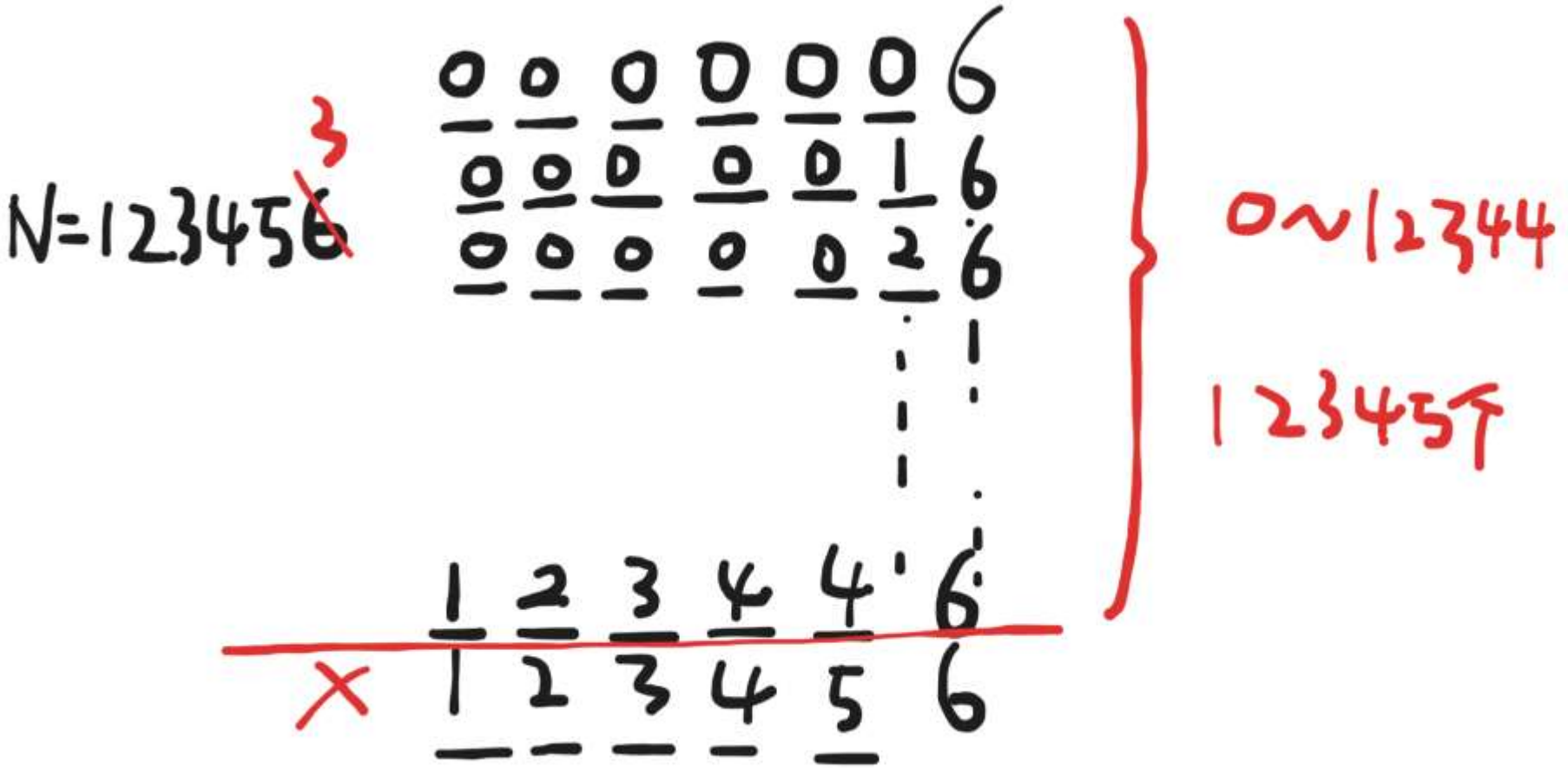
- **输出$1\sim N$中个位是6的数字个数**
- $T(1 \le T \le 10^6)$
- $N(1 \le N \le 10^9)$

- **先让个位为6，然后再考虑1~N的所有数字除去个位之后有多少个不同的数字**

谁出的这题，出题人给我过来



谁出的这题，出题人给我过来



谁出的这题，出题人给我过来

- 先让个位为6，然后再考虑1~N的所有数字除去个位之后有多少个不同的数字
- 如果先不考虑N的个位数字，就是有 $\left\lfloor \frac{N}{10} \right\rfloor$ 种情况
- 如果个位大于等于6，就会多一个

谁出的这题，出题人给我过来

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

int main()
{
    int t;
    cin>>t;
    while(t--)
    {
        int n;
        cin>>n;
        if(n%10>=6)cout<<n/10+1<<endl;
        else cout<<n/10<<endl;
    }
    return 0;
}
```

我才不是签到题

题意

- 将数字的千位分隔表示改成万位间隔表示

- 首先这个题目很容易想到的就是先把数字全部提取出来，然后再加逗号
- 有一点需要注意的就是需要从后往前加，否则会出现"_ _ _ _ , _ _ " 的情况， 然后有一个坑就是末尾的逗号需要去掉，比如说出现这样的情况 " _ _ _ , _ _ _ _ ,"
- 时间复杂度$O(n)$

我才不是签到题

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


int main()
{
    string s;
    cin >> s;
    string s1;
    for (int i = s.size() - 1; i >= 0; i--) {
        if (s[i] >= '0' && s[i] <= '9') s1 = s1 + s[i];
    }
    string ans;
    for (int i = 1; i <= s1.size(); i++) {
        ans = ans + s1[i - 1];
        if (i % 4 == 0) ans = ans + ',';
    }
    if (ans.back() == ',') ans.pop_back();
    reverse(ans.begin(), ans.end());
    cout << ans << '\n';
}
```

再来一发就AC

题意

- 将a变成b的最少操作数
- 每次操作可以给a加一个正奇数x或者减去一个正偶数y

- 如果$a = b$，共操作0次
- 如果$a > b$，如果$(a - b)$为偶数，减去一个偶数，共操作1次，

  如果$(a - b)$为奇数，就让a加上一个1，那么$(a - b + 1)$就为偶数，

  再减去一个偶数，共操作2次

再来一发就AC

题意

- 将a变成b的最少操作数
- 每次操作可以给a加一个正奇数x或者减去一个正偶数y

- 如果$a < b$，如果$(b-a)$为奇数，共操作1次，否则$(b-a)$为偶数，我们可以加两个奇数就是偶数了，共操作2次，但是很明显有的情况错了，因为$\frac{(b-a)}{2}$可能为偶数，根据操作我们是不能加两个偶数的，比如$a=2$，$b=4$，所有我们需要先加两个奇数再减去一个偶数就可以了，$x=\frac{b-a}{2}+1$，$y=2$，共操作3次

再来一发就AC

```cpp
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int t;
    cin>>t;
    while(t--){
        int x,y;
        cin >> x >> y;
        if (x == y) {
            cout << 0 << '\n';
        } else if (x < y && (x % 2 == y % 2) && (y - x) / 2 % 2 == 1) {
            cout << 2 << '\n';
        }  else if (x < y && (x % 2 == y % 2)) {
            cout << 3 << '\n';
        } else if (y > x && (y - x) % 2 == 1) {
            cout << 1 << '\n';
        } else if (x > y && (x - y) % 2 == 0) {
            cout << 1 << '\n';
        } else {
            cout << 2 << '\n';
        }
    }

}
```

学姐给你们出的签到题

题意

- 求出每次询问一个前缀内有多少对相同的数

```
#include<iostream>
using namespace std;
int main(){
    int n;
    cin>>n;
    long long int x,sum;
    sum=0;
    for(int i=1;i<=n;i++){
        cin>>x;
        a[x]=a[x]+1;//记录一个数出现了多少遍
        if(a[x]>1)
        sum=sum+a[x]-1;//出现了2遍及以上则会产生a[x]-1个数对
        cout<<sum<<" ";
    }
    return 0;
}
```

三国杀？启动！

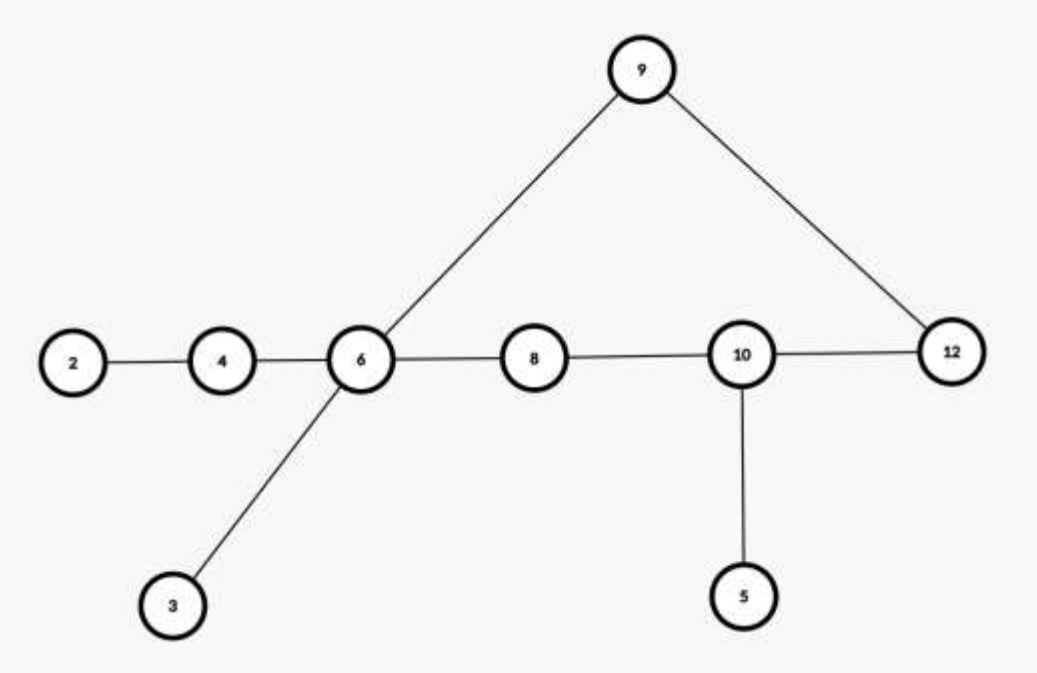
题意

- 将a序列任意顺序排列，使得尽可能多的满足$\gcd(a_{i-1}, a_i) \geq 2$，$(2 \leq i \leq n)$，并输出满足的个数
- $(1 \leq n \leq 10^6)$，$(1 \leq a_i \leq 13)$

- 考虑构造出一个最优结构，使得a序列按次排序满足的个数最多
- 如果 a_i 为1，那么在 a_i 前后的都不会满足条件，所有考虑将所有的1都要放在一起
- 如果 a_i 为7，11，13，那么只有他们各自为连续的时候才会满足 7 7 7 7 7 或者 11 11 11 11 或者 13 13 13 13

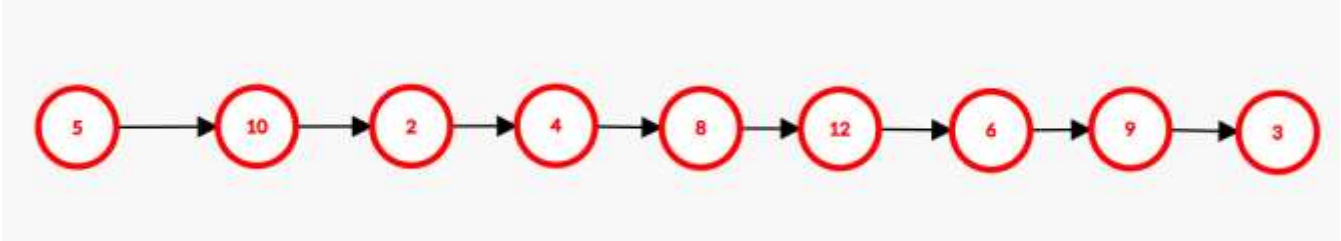
三国杀？启动！

- 再考虑2 4 6 8 10 12 还有 3 6 9 12 以及 5 10 的情况
- 如果这些在序列中连续出现，那么会有更多满足题意的个数
- 我们发现



三国杀？启动！

- 然后再调换一下顺序5 10 2 4 8 12 6 9 3



三国杀？启动！

- 最后综合起来

$$1,1,1,...5,5,...10,10,...2,2,..,4,4,..8,8..12,12,..6,6,..9,9..3,3..,7,7,..11,11,...13,13$$

- 还有其他贪心方法也类似相同

我们终于见面了

题意

- 从A序列跟B序列中各选$\frac{k}{2}$个数字能否凑成$1\sim k$的所有数字
- $(1 \le n,m \le 2 \cdot 10^5)$，$(1 \le A_i,B_i \le 10^6)$

- **如果要满足条件，那么$1\sim k$中的数必须要在两个序列中至少出现一次**
- **而对于只在某一个序列中出现的数的个数都要小于等于$\frac{k}{2}$**

我们终于见面了

```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int _;
    cin>>_;
    while(_--){
        int n,m,k,h=1;
        cin>>n>>m>>k;
        int a[n],b[m],x[k+1],y[k+1];
        for(int i=1;i<=k;i++)x[i]=0,y[i]=0;
        for(int i=0;i<n;i++){
            cin>>a[i];
            if(a[i]<=k)x[a[i]]++;
        }
        for(int i=0;i<m;i++){
            cin>>b[i];
            if(b[i]<=k)y[b[i]]++;
        }
        int p=0,q=0;
        for(int i=1;i<=k;i++){
            if(!x[i]&&y[i]){
                p++;
            }
            if(!y[i]&&x[i]){
                q++;
            }
            if(x[i]==0&&y[i]==0){
                cout<<"NO"<<endl;
                h=0;
                break;
            }
        }
        if(h){
            if(p<=k/2&&q<=k/2){
                cout<<"YES"<<endl;
            }
            else{
                cout<<"NO"<<endl;
            }
        }
```

我们是好朋友

题意

- 要求判断是否所有人属于同一个朋友圈
- 现在，给定每个人最想成为朋友的人的列表，请判断他们是否可以互为朋友

- 本题的意思就是判断它们是否可以形成一个闭环，而形成闭环的条件就是每个人有且仅有一个人指向他，所以我们输入的时候记录一下有多少人指向他，如果大于一则$return$;

我们是好朋友

- 需要注意的是他们之间可以形成多个闭环，此时他们也满足有且仅有一个人指向他的条件，所以我们遍历任何一个闭环，看这个闭环的元素个数是不是n个
- 时间复杂度$O(n)$。

我们是好朋友

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n; cin >> n;
    vector<int> a(n + 1);
    vector<bool> st(n + 1);
     for (int i = 1; i <= n; i++) {
         cin >> a[i];
        if (st[a[i]]) {
            cout << "NO\n";
            return 0;
        }
        st[a[i]] = 1;
    }
    int h = 1, cnt = 1, now = a[1];
    while (now != h) {
        cnt++;
        now = a[now];
    }
    if (cnt != n) {
        cout << "NO\n";
    } else {
        cout << "YES\n";
    }
    return 0;
}
```

我是奶龙，我才是奶龙

题意

- 先把字符矩阵转化成T还有S
- 然后输出最长的“奶龙串”长度
- 字符串长度最多$20 \cdot 20 = 400$

- 直接按题意模拟出T和S字符串
- 然后暴力匹配出最长的“奶龙串”长度即可
- L为字符串长度，时间复杂度$0(L^3)$即可通过此题

我是奶龙，我才是奶龙

```
    int n,m;
    cin>>n>>m;
    vector<string> a(n);
    for(int i=0;i<n;i++){
        cin>>a[i];
    }
    string T,S;
    for(int i=0;i<n;i++){
        if(i%2==0){
            for(int j=0;j<m;j++){
                T+=a[i][j];
            }
        }
        else{
            for(int j=m-1;j>=0;j--){
                T+=a[i][j];
            }
        }
    }
    for(int j=m-1;j>=0;j--){
        if((j%2)^(m%2)){
            for(int i=n-1;i>=0;i--){
                S+=a[i][j];
            }
        }
        else{
            for(int i=0;i<n;i++){
                S+=a[i][j];
            }
        }
    }
    cout<<f(T,n*m)<<" "<<f(S,n*m)<<'\n';
    return 0;
}
```

```
int f(string a,int n){
    int ans=0;
    for(int L=2;L<=n;L+=2){
        for(int x=0;x+L-1<n;x++){
            int y=x+L/2;
            int ok=1;
            for(int i=0;i<L/2;i++){
                if(a[x+i]!=a[y+i]){
                    ok=0;
                    break;
                }
            }
            if(ok)ans=max(ans,L);
        }
    }
    return ans;
}
```

不可视境界线

题意

- 求 $\sum_{i=1}^{n}\sum_{j=1}^{n}(|a_i - a_j| \cdot \max(a_i, a_j))$
- 将数组排序之后等价于

$$\sum_{i=1}^{n}\sum_{j=1}^{i-1}(a_i - a_j) \cdot a_i + \sum_{i=1}^{n}\sum_{j=i+1}^{n}(a_j - a_i) \cdot a_j$$

不可视境界线

- $= \sum_{i=1}^{n}\sum_{j=1}^{i-1}(a_i - a_j)\cdot a_i + \sum_{i=1}^{n}\sum_{j=i+1}^{n}(a_j - a_i)\cdot a_j$
- $= \sum_{i=1}^{n}\sum_{j=1}^{i-1}(a_i - a_j)\cdot a_i + \sum_{j=1}^{n}\sum_{i=1}^{j-1}(a_j - a_i)\cdot a_j$
- $= \sum_{i=1}^{n}\sum_{j=1}^{i-1}2\cdot(a_i - a_j)\cdot a_i$

不可视境界线

- 令 $sum_x = \sum_{i=1}^{x} a_i$
- 最后化简得 $\sum_{i=1}^{n} [(i \cdot a_i - sum_i) \cdot 2 \cdot a_i]$
- 只需预处理前缀和sum_i 计算取模即可

不可视境界线

```cpp
#include <bits/stdc++.h>
using namespace std;
using ll = long long int;
int main() {
    ll n;
    cin>>n;
    ll mod=998244353;
    vector<ll> a(n+1,0),sum(n+1,0);
    for(int i=1;i<=n;i++){
        cin>>a[i];
    }
    sort(a.begin(),a.end());
    for(int i=1;i<=n;i++){
        sum[i]+=sum[i-1]+a[i];
    }
    ll ans=0;
    for(int i=1;i<=n;i++){
        ans=(ans+((a[i]*i-sum[i])*2%mod*a[i])%mod)%mod;
    }
    cout<<ans%mod<<'\n';
    return 0;
}
```

不可视境界线

1

软工2203 张泰林 LV 6 ACM @ 19 小时前

这个题最关键的就是怎么去掉绝对值。 不难发现 $|a_i - a_j|$等价于$max(a_i, a_j) - min(a_i, a_j)$

那么原式就等价于

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(max(a_i, a_j)^2 - a_i * a_j)$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} max(a_i, a_j)^2 - \sum_{i=1}^{n}\sum_{j=1}^{n} a_i * a_j$$

后一项很容易发现为$(a_1 + a_2 + ... + a_n) * (a_1 + a_2 + ... + a_n)$

观察前一项，不难发现$max(a_i, a_j)$只有当$a_j \leq a_i$才会取到a_i，那么我们将数组从小到大排序，对于a_i来看，只有

$(a_1, a_i), (a_i, a_1), (a_2, a_i), (a_i, a_2)...(a_{i-1}, a_i)(a_i, a_{i-1})(a_i, a_i)$一共$2 * i - 1$对会取到$a_i$那么原式就等价于

$$\sum_{i=1}^{n} a_i^2 * (2 * i - 1) - (a_1 + a_2 + ... + a_n) * (a_1 + a_2 + ... + a_n)$$

不可视境界线

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

#define int long long
const int mod = 998244353;

signed main()
{
    int n;
    cin>>n;
    int sum=0;
    vector<int> a(n+10);
    for(int i=1;i<=n;i++){
        cin>>a[i];
        sum+=a[i];
    }
    sort(a.begin()+1,a.begin()+1+n);
    int ans=0;
    for(int i=1;i<=n;i++){
        ans=(ans+a[i]*a[i]*(i*2-1))%mod;
    }
    ans=((ans-sum%mod*sum%mod)%mod+mod)%mod;
    cout<<ans<<endl;
}
```

盗书

0

软工2302 张荣臻 LV 5 ACM @ 1 天前

设计$dp[i][j][k][l]$表示当前选到四种花色数量分别为i,j,k,l个时的概率，那么可以发现转移有

$$dp_{i,j,k,l}=\begin{cases}dp_{i-1,j,k,l}\times p_{\text{当前剩余手牌选到黑桃的概率}} & i\geq 1\\ dp_{i,j-1,k,l}\times p_{\text{当前剩余手牌选到红桃的概率}} & j\geq 1\\ dp_{i,j,k-1,l}\times p_{\text{当前剩余手牌选到梅花的概率}} & k\geq 1\\ dp_{i,j,k,l-1}\times p_{\text{当前剩余手牌选到方片的概率}} & k\geq 1\end{cases}$$

那么每次抽到哪个花色的牌的概率可以用当前已知的每种花色手牌数量和总牌数表示出来。 因为技能发动成功必须一直发动除非全部手牌已经拿完，所以最后的概率应该在乘上一个**失败概率**， 可以计算， 假设已经有每种花色i,j,k,l个， 初始a,b,c,d, 那么$p_{\text{失败概率}}=\begin{cases}\frac{1}{4}\times\frac{b-j+c-k+d-l}{a-i+b-j+c-k+d-l}\\ \frac{1}{4}\times\frac{a-i+c-k+d-l}{a-i+b-j+c-k+d-l}\\ \frac{1}{4}\times\frac{a-i+b-j+d-l}{a-i+b-j+c-k+d-l}\\ \frac{1}{4}\times\frac{a-i+b-j+c-k}{a-i+b-j+c-k+d-l}\end{cases}$

可以发现失败概率就是$\frac{3}{4}$， 初始化的时候$dp_{0,0,0,0}$为1。

盗书

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

int dp[51][51][51][51];
const int mod = 998244353;

int qmi(int a, int b) {
    int res = 1;
    while (b) {
        if (b & 1) res = res * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return res;
}

void update(int &x, int y) {
    x = (x + y) % mod;
}
```

盗书

```
void solve () {
    int a, b, c, d; cin >> a >> b >> c >> d;
    int n = a + b + c + d;
    dp[0][0][0][0] = 1;
    for (int i = 0; i <= a; i ++) {
        for (int j = 0; j <= b; j ++) {
            for (int k = 0; k <= c; k ++) {
                for (int l = 0; l <= d; l ++) {
                    if (i) update(dp[i][j][k][l], dp[i - 1][j][k][l] * (a - i + 1) % mod * qmi(n - i - j - k - l + 1, mod - 2) % mod * qmi(4, mod - 2) % mod);
                    if (j) update(dp[i][j][k][l], dp[i][j - 1][k][l] * (b - j + 1) % mod * qmi(n - i - j - k - l + 1, mod - 2) % mod * qmi(4, mod - 2) % mod);
                    if (k) update(dp[i][j][k][l], dp[i][j][k - 1][l] * (c - k + 1) % mod * qmi(n - i - j - k - l + 1, mod - 2) % mod * qmi(4, mod - 2) % mod);
                    if (l) update(dp[i][j][k][l], dp[i][j][k][l - 1] * (d - l + 1) % mod * qmi(n - i - j - k - l + 1, mod - 2) % mod * qmi(4, mod - 2) % mod);
                }
            }
        }
    }
    for (int i = 0; i <= a; i ++) {
        for (int j = 0; j <= b; j ++) {
            for (int k = 0; k <= c; k ++) {
                for (int l = 0; l <= d; l ++) {
                    if (i != a || j != b || k != c || l != d) dp[i][j][k][l] = (dp[i][j][k][l] * 3  % mod * qmi(4, mod - 2) % mod);
                }
            }
        }
    }
    int q; cin >> q;
    while (q --) {
        int p1, p2, p3, p4;
        cin >> p1 >> p2 >> p3 >> p4;
        cout << dp[p1][p2][p3][p4] << '\n';
    }
}
signed main() {
    int t = 1;
    while (t --) solve();
}
```

end

感谢观看！！！