2022年西北大学校赛题解

难度等级：

我们预测的题目难度等级：

EASY：A → G → D → B

MEDIUM： C → J → F → E

HARD： I → H

实际比赛期间大致与之相符。

Problem A. 第一次相遇

签到题，找到最小的数，他的各位数之和是给定的N。贪心的去想我们肯定要让首位最小同时让数位尽可能少，所以，很容易想到让低位都是9，这样得到的数字最小。

题目数据范围是$1-63$，也可以用很暴力的方法去一个一个数字的去check。

代码：

```
#include<bits/stdc++.h>
#define ll long long
#define endl '\n'
#define pii pair<int,int>
#define rep(i,a,b) for(int i = (int)(a);i <= (int)(b);i ++)
#define IO ios::sync_with_stdio(false);cin.tie(0);
#define all(x) x.begin(),x.end()
using namespace std;
const int maxn = 1e5 + 5;
const int mod = 1e9 + 7;
const int inf = 0x3f3f3f3f;


int main(){
    int n;
    cin >> n;
    string ans = "";
    while(n >= 9){
        ans += '9';
        n -= 9;
    }
    if(n > 0)
        ans += '0' + n;
    reverse(all(ans));
    cout << ans;
    return 0;
}
```

Problem B. 爱丽丝的约定

这道题很明显，我们可以按照题意去模拟。关键在于如何找到每一次的最大能力值的事件，这里我们可以采用优先队列$priority_queue$去维护。

关于优先队列的使用不会的同学可以自行去百度，这是一种很好用的容器。

```
#include<bits/stdc++.h>
#define ll long long
#define endl '\n'
#define pii pair<int,int>
#define rep(i,a,b) for(int i = (int)(a);i <= (int)(b);i ++)
#define IO ios::sync_with_stdio(false);cin.tie(0);
#define all(x) x.begin(),x.end()
using namespace std;
const int maxn = 1e5 + 5;
const int mod = 1e9 + 7;
const int inf = 0x3f3f3f3f;



struct node{
    int num,id;
    bool operator<(const node & other)const{
        if(num != other.num)
            return num < other.num;          //优先队列，先按吸引力值较大的进行排序。
        return id > other.id;                //如果吸引力值大，则按下标较小的排序。
    }
};
int main(){
    int n,k;
    cin >> n >> k;
    priority_queue<node>q;                   //定义有限队列来维护整个过程
    rep(i,1,n){
        int x;  cin >> x;
        q.push({x,i});
    }
    vector<int>ans;                          //保存答案的数组
    while(!q.empty() && k --){               //最大学习k次，同时队列不为空
        node now = q.top();                  //我们每次取出吸引力最大的事情去做
        q.pop();
        if(now.num == 0) break;              //如果当前最大的吸引力是0，那么，就停止这一
切
        ans.push_back(now.id);               //保存事情的下标
        q.push({now.num - (now.num + 1)/2,now.id});          //继续将这个事情的剩余部分
加入到优先队列中。
    }
    cout << ans.size() << endl;
    for(auto v : ans)
        cout << v << ' ';
    return 0;

}
```

Problem C. 这条路从来不简单

简化的题意可以是这样：我们就是要找树上的最大子段和。如果你会一维线性的子段和问题，那么这道题就会很简单。我们只需要在dfs的过程中加入这种类似于线性的子段和的求解方法。

```
#include<bits/stdc++.h>
#define ll long long
```

```
#define endl '\n'
#define pii pair<int,int>
#define rep(i,a,b) for(int i = (int)(a);i <= (int)(b);i ++)
#define IO ios::sync_with_stdio(false);cin.tie(0);
#define all(x) x.begin(),x.end()
using namespace std;
const int maxn = 1e6 + 5;
const int mod = 1e9 + 7;
const int inf = 0x3f3f3f3f;


vector<int>G[maxn];
int valu[maxn];
ll ans = 0;
void dfs(int u,int fa,ll now){
    if(now > 0) now += valu[u] ;   //动态规划的过程
    else now = valu[u];
    ans = max(ans,now);           //维护全局的最大值
    for(auto v : G[u]){
        if(v == fa) continue;
        dfs(v,u,now);
    }
}
int main(){
    int n;
    cin >> n;
    rep(i,1,n) cin >> valu[i];
    for(int i = 1;i < n;i ++){
        int u,v;
        cin >> u >> v;
        G[u].push_back(v);
        G[v].push_back(u);
    }
    dfs(1,-1,0ll);
    cout << ans;
    return 0;
}
```

Problem D. 绝地逢生

我们用 $dp[i][j]$ 表示到达第i行第j列的格子时最少需要铲除的荆棘数目。

那么：

如果$mapp[i][j]$是‘0’那么：$dp[i][j] = min\begin{cases} dp[i][j-1] \\ dp[i-1][j] \end{cases}$

如果$mapp[i][j]$是‘1’那么：$dp[i][j] = min\begin{cases} dp[i][j-1] \\ dp[i-1][j] \end{cases} + 1$

剩下的部分通过的事件很容易确定，给定n, m那么移动的时间是固定的 $n + m - 2$ ，在加上 $T * dp[n][m]$，就是最终需要的最少时间。

```
#include<bits/stdc++.h>
#define ll long long
#define endl '\n'
#define pii pair<int,int>
#define rep(i,a,b) for(int i = (int)(a);i <= (int)(b);i ++)
#define IO ios::sync_with_stdio(false);cin.tie(0);
```

```
#define all(x) x.begin(),x.end()
using namespace std;
const int maxn = 1e5 + 5;
const int mod = 1e9 + 7;
const int inf = 0x3f3f3f3f;



char mapp[1005][1005];
int dp[1005][1005];
int main(){
    int n,m,t;
    cin >> n >> m >> t;
    rep(i,1,n)
        rep(j,1,m)
            cin >> mapp[i][j];
    rep(i,0,n) dp[i][0] = inf;
    rep(j,0,m) dp[0][j] = inf;
    dp[0][1] = dp[1][0] = 0;
    rep(i,1,n)
        rep(j,1,m){
            if(mapp[i][j] == '1')
                dp[i][j] = min(dp[i - 1][j],dp[i][j - 1]) + 1;
            else
                dp[i][j] = min(dp[i - 1][j],dp[i][j - 1]);
        }
    ll ans = 0;
    ans = 1ll * dp[n][m] * t + n + m - 2;
    cout << ans;
    return 0;
}
```

Problem E. 奇妙的数字

预处理+欧拉筛 / 打表

首先对于这么大的数据范围，我们首先考虑预处理出题目要求的范围中的极素数。然后每次去二分一下这个查询区间，通过前缀和就可以得到答案。

那么如何预处理呢？我们首先用欧拉筛筛出所有的质数，然后再对这些素数进行判断。

打表的做法，比较简单就不提供代码了。

```
#include<bits/stdc++.h>
using namespace std;
#define IO ios::sync_with_stdio(0),cin.tie(0)

const int maxn = 1e7+10;
bool issum[maxn];
vector<int> prime;
vector<int> jiPrime;


void init()
{
    prime.resize(0);
    for (int i = 2; i <= 1e7; i ++) {
        if (!issum[i]) prime.emplace_back(i);
```

```
        for (int j = 0; i*prime[j] <= 1e7 && j < int(prime.size()); j ++) { // 防止乘积越界
            issum[i*prime[j]] = true;
            if(!(i%prime[j])) break;
        }
    }
    int len = prime.size();
    jiPrime.clear();
    for(int i = 0;i < len;++i) {
        int k = prime[i]-1;
        bool si = true;
        for(int j = 1;j <= sqrt(k);++j) {
            if(k%j == 0) {
                if(issum[j + k/j]) {
                    si = false;
                    break;
                }
            }
        }
        if(si) {
            jiPrime.push_back(k);
        }
    }
    return ;
}

void solve()
{
    int l,r;
    cin >> l >> r;
    int lid = lower_bound(jiPrime.begin(),jiPrime.end(),l) - jiPrime.begin();
    int rid = upper_bound(jiPrime.begin(),jiPrime.end(),r) - jiPrime.begin();
    cout << rid-lid << endl;
}



int main() {
    IO;
    init();
    int t;
    cin >> t;
    while(t--) {
        solve();
    }
    return 0;
}
```

Problem F .绳往细处断

对于这道题，我们考虑二分答案。具体过程如下：

首先这样想：如果最晚可以N天断，那么$N-1$天也一定可以满足。那么这个答案就满足二分的性质。

我们假设当前若K天断的话，那么如何去check呢？这样想，对于一根绳，它的粗度是A_i，如果A_i大于等于K那么这根绳子我们可以每天都去选。如果A_i小于K，那么说明这根绳子不能天天都选，我们把这样的不够K天的部分$K-A_i$加起来记为SUM。最后我们只需要判断SUM是否小于K即可，如果小于K，我们每天选$N-1$条绳子时，刚好不选这些绳子就行。

```
#include<bits/stdc++.h>
using namespace std;
#define IO ios::sync_with_stdio(false);cin.tie(0);
#define ll long long
#define PII pair<int,int>
#define fi first
#define se second

const int N=1e5+5;
ll a[N];
int n;

bool check(ll x){
    ll cur=0;
    for(int i=1;i<=n;i++)
        if(a[i]-x<0)    cur+=x-a[i];
    if(cur<=x)
        return true;
    else
        return false;
}

void solve(){
    cin>>n;
    ll l = 0,r = 0;
    for(int i = 1;i <= n;i ++){
        cin >> a[i];
        r += a[i];
    }
    while(l<r){
        ll mid = (l+r+1) >> 1;
        if(check(mid))
            l = mid;
        else
            r = mid-1;
    }
    cout << l << "\n";
}

int main(){
    IO
    solve();
    return 0;
}
```

Problem G.爱我有多深

我们可以把每一段连续相同的字符分开处理。我们可以看到，对于每一段连续相同长度为len的的字符串来说，它的每天的贡献是$len, len-1, len-2，\ ......，1$，假设最长的那个子串的长度还没变为1，那么这个字符串接下来的贡献就是1。对于最长的那个字符串来说它的贡献就只有$len，\ len-1, len-2, ...，1$。所以我们首先需要记录出最长的连续相同的字符串，然后对剩下的进行每次加一的处理即可。

```
#include<bits/stdc++.h>
#define ll long long
#define endl '\n'
#define pii pair<int,int>
#define rep(i,a,b) for(int i = (int)(a);i <= (int)(b);i ++)
#define IO ios::sync_with_stdio(false);cin.tie(0);
```

```
#define all(x) x.begin(),x.end()
using namespace std;
const int maxn = 1e5 + 5;
const int mod = 1e9 + 7;
const int inf = 0x3f3f3f3f;

map<int,int>mp;
int main(){
    int n;
    string str;
    cin >> n >> str;
    char lst = str[0];
    int len = 1;
    int mx = 0;
    for(int i = 1;i < n;i ++){
        if(str[i] == lst){
            len ++;
        }
        else{
            mp[len] ++;
            lst = str[i];
            mx = max(mx,len);
            len = 1;
        }
    }
    mp[len] ++;
    mx = max(mx,len);
    ll ans = 0;
    for(auto v : mp){
        ll a =  v.first;
        ll b = v.second;
        ll now = b * (a * (a + 1) / 2 + (mx - a));
        ans += now;
    }
    cout << ans;
}
```

Problem H.littlefish和他的区间

对所给区间进行离线处理，按左端点排序，然后从后往前依次遍历，每次把左端点在当前位置的所有区间全部加入树状数组或者线段树维护[1,r]的最大值。因为从后往前遍历，已经插入的区间左端点大于等于当前查询区间的左端点，那么只需要找到右端点小于等于当前查询区间右端点的区间最大值即可。

```
#include<bits/stdc++.h>
using namespace std;
typedef pair<int, int> PII;
const int maxn = 100010, inf = 1e9;
int tr[maxn];
int a[maxn], ans[maxn];
vector<PII> v[maxn], seg[maxn];
int n, m;
int lowbit(int x) {
    return x & -x;
}
void modify(int x, int y) {
    while (x < maxn) {
        tr[x] = max(tr[x], y);
        x += lowbit(x);
    }
```

```
}
int query(int x) {
    int ans = 0;
    while (x) {
        ans = max(ans, tr[x]);
        x -= lowbit(x);
    }
    if (ans == 0)
        ans = -1;
    return ans;
}
int main() {
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i ++ ) {
        int l, r, w;
        scanf("%d%d%d", &l, &r, &w);
        seg[l].push_back({r, w});
    }
    for (int i = 1; i <= m; i ++ ) {
        int l, r;
        scanf("%d%d", &l, &r);
        v[l].push_back({r, i});
    }
    for (int i = n; i >= 1; i -- ) {
        for (auto ver : seg[i])
            modify(ver.first, ver.second);
        for (auto ver : v[i])
            ans[ver.second] = query(ver.first);
    }
    for (int i = 1; i <= m; i ++ )
        printf("%d\n", ans[i]);
    return 0;
}
```

Problem I. 数字魔法

我们考虑建图去处理，对于每种操作看作一条边，然后在图上跑$Dijkstra$算法去求从A到B的最短路。在具体的实现过程中，我们要开两倍的空间，因为在变化的过程中可能会出现一个较大的数作为中转点，所以空间要开大些。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = (1 << 19) + 5;    //注意这里多开一位空间
ll c[22], dp[maxn], n, m;
int vis[maxn];
int a, b;
struct node{
    int num;
    ll cost;
    bool operator<(const node & other)const{
        return cost > other.cost;
    }
};
int main() {
    cin >> a >> b >> n >> m;
    for (int i = 0; i <= 18; i ++ )
        cin >> c[i];
    memset(dp, 0x3f, sizeof dp);
```

```
    dp[a] = 0;
    priority_queue<node> q;
    q.push(node{a,0});
    while (q.size()) {
        a = q.top().num;
        q.pop();
        if(vis[a]) continue;
        vis[a] = 1;
        if (a + 1 < maxn && dp[a + 1] > dp[a] + n) {
            dp[a + 1] = dp[a] + n;
            q.push({a + 1,dp[a + 1]});
        }
        if (a - 1 >= 0 && dp[a - 1] > dp[a] + m) {
            dp[a - 1] = dp[a] + m;
            q.push({a - 1,dp[a - 1]});
        }
        for (int i = 0; i <= 18; i ++ ) {
            if ((a | (1 << i)) < maxn && dp[a | (1 << i)] > dp[a] + c[i]) {
                dp[a | (1 << i)] = dp[a] + c[i];
                q.push({a | (1 << i),dp[a|(1 << i)]});
            }
        }
    }
    cout << dp[b] << endl;
    return 0;
}
```

Problem J.笔下生花

首先，灰常抱歉，我们的题面出了一些问题，耽误了选手们的时间。最后在公告里发了有问题的地方。

这道题是一道模拟题，按照题意进行模拟即可，具体的实现过程大家还是看代码吧。

```
#include<bits/stdc++.h>
using namespace std;
#define IO ios::sync_with_stdio(false);cin.tie(0);

int cnt[5];
int n,m;

int main(){
    IO
    cin>>n>>m;
    int cur=0,tmp=0;
    bool fi;
    string res;
    for(int i=0;i<n;i++){
        int op;cin>>op;
        if(op==1){
            char ch;cin>>ch;
            if(cur==0){
                if(ch>='A'&&ch<='Z')    ch+=32;
                res+=ch;
                cnt[cur]++;
            }
            else if(cur==1){
                if(ch>='a'&&ch<='z')    ch-=32;
                res+=ch;
                cnt[cur]++;
```

```
            }
            else{
                if(fi){
                    if(ch>='a'&&ch<='z')    ch-=32;
                    res+=ch;
                    cnt[cur]++;
                    fi=false;
                }
                else{
                    if(ch>='A'&&ch<='Z')    ch+=32;
                    res+=ch;
                    cnt[cur]++;
                }
            }
            tmp++;
        }
        else if(op==2){
            res+=" ";
            tmp++;
            if(cur==2)  fi=true;
        }
        else{
            if(tmp<=m){
                cur=(cur+1)%3;
                if(cur==2)  fi=true;
            }
            else{
                int mx=0;
                if(cur==0)  mx=1;
                for(int i=0;i<3;i++){
                    if(i==cur)  continue;
                    if(cnt[i]>cnt[mx])  mx=i;
                }
                cur=mx;
                if(cur==2)  fi=true;
            }
            tmp=0;
        }
    }
    cout<<res<<"\n";
    return 0;
}
```