

David的ACM模板

一、动态规划

背包DP

01背包

```
1 #include <iostream>
2 #include <cstring>
3 #include <algorithm>
4 #include <cmath>
5 using namespace std;
6 const int _ = 1005;
7 int w[_], v[_];
8 long long dp[_];
9 int main() {
10     int n, C;
11     scanf("%d%d", &n, &C);
12     for (int i = 1; i <= n; ++i) scanf("%d", &w[i]);
13     for (int i = 1; i <= n; ++i) scanf("%d", &v[i]);
14     for (int i = 1; i <= n; ++i) {
15         for (int j = C; j >= v[i]; --j)
16             dp[j] = max(dp[j], dp[j - v[i]] + w[i]);
17     }
18     printf("%lld\n", dp[C]);
19 }
```

多重背包

```
1 #include <iostream>
2 #include <algorithm>
3 using namespace std;
4 const int N = 100010;
5 int n, C, dp[N];
6 int w[N], c[N], m[N];
7 int new_n;
8 int new_w[N], new_c[N], new_m[N];
9 int main() {
10     ios::sync_with_stdio(false);
11     cin.tie(nullptr);
```

```

12     cout.tie(nullptr);
13     cin >> n >> C;
14     for (int i = 1; i <= n; ++i) {
15         cin >> w[i] >> c[i] >> m[i];
16     }
17     int new_n = 0;
18     for (int i = 1; i <= n; ++i) {
19         for (int j = 1; j <= m[i]; j <= 1) {
20             m[i] -= j;
21             new_c[++new_n] = j * c[i];
22             new_w[new_n] = j * w[i];
23         }
24         if (m[i]) {
25             new_c[++new_n] = m[i] * c[i];
26             new_w[new_n] = m[i] * w[i];
27         }
28     }
29     for (int i = 1; i <= new_n; ++i) {
30         for (int j = C; j >= new_c[i]; --j) {
31             dp[j] = max(dp[j], dp[j - new_c[i]] + new_w[i]);
32         }
33     }
34     cout << dp[C] << '\n';
35     return 0;
36 }

```

单调队列优化

```

1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4  const int N = 100010;
5  int n, C;
6  int dp[N], q[N], num[N];
7  int w, c, m;
8  int main() {
9      ios::sync_with_stdio(false);
10     cin.tie(nullptr);
11     cout.tie(nullptr);
12     cin >> n >> C;
13     for (int i = 1; i <= n; ++i) {
14         cin >> w >> c >> m;
15         if (m > C / c) m = C / c;
16         for (int b = 0; b < c; ++b) {
17             int head = 1, tail = 1;

```

```

18         for (int y = 0; y <= (C - b) / c; ++y) {
19             int tmp = dp[b + y * c] - y * w;
20             while (head < tail && q[tail - 1] <= tmp) --tail;
21             q[tail] = tmp;
22             num[tail++] = y;
23             while (head < tail && y - num[head] > m) ++head;
24             dp[b + y * c] = max(dp[b + y * c], q[head] + y * w);
25         }
26     }
27 }
28 cout << dp[C];
29 return 0;
30 }

```

完全背包

```

1 #include <iostream>
2 #include <cstring>
3 #include <algorithm>
4 using namespace std;
5 const int N=1e4+5,M=1e7+5;
6 long long v[N],w[N],f[M];
7 int main(){
8     int n, m; scanf("%d%d",&m,&n);
9     for(int i=1; i<=n; i++)
10         scanf("%d%d",&v[i],&w[i]); //费用, 价值
11     for(int i=1; i<=n; i++) //枚举物品
12         for(int j=v[i]; j<=m; j++) //枚举体积
13             f[j]=max(f[j],f[j-v[i]]+w[i]);
14     printf("%lld\n",f[m]);
15 }

```

混合背包

1

二维费用背包

1

分组背包

1

线性DP

最长公共子序列LCS

```
1 #include <iostream>
2 #include <cstring>
3 #include <algorithm>
4 using namespace std;
5 const int N=1010;
6 int n, m;
7 char a[N], b[N];
8 int f[N][N];
9 int main(){
10     cin>>n>>m>>a+1>>b+1;
11     for(int i=1; i<=n; i++)
12         for(int j=1; j<=m; j++)
13             if(a[i]==b[j]) f[i][j]=f[i-1][j-1]+1;
14             else f[i][j]=max(f[i-1][j-1],max(f[i-1][j],f[i][j-1]));
15     cout<<f[n][m];
16 }
```

最长上升子序列LIS

```
1 #include <iostream>
2 #include <cstring>
3 #include <algorithm>
4 using namespace std;
5 const int N=1010;
6 int n, a[N];
7 int f[N];
8 int main(){
9     cin>>n;
10     for(int i=1; i<=n; i++) cin>>a[i];
11     for(int i=1; i<=n; i++) f[i]=1;
12     for(int i=1; i<=n; i++)
13         for(int j=1; j<i; j++)
14             if(a[j]<a[i]) f[i]=max(f[i],f[j]+1);
```

```

15     int res=0;
16     for(int i=1; i<=n; i++) res=max(res,f[i]);
17     cout<<res;
18 }

```

二分

```

1  #include <iostream>
2  #include <cstring>
3  #include <algorithm>
4  using namespace std;
5  const int N=100010;
6  int n, a[N];
7  int len, b[N]; //记录上升子序列
8  int main(){
9     scanf("%d", &n);
10    for(int i=0; i<n; i++) scanf("%d", &a[i]);
11    b[0]=-2e9; //哨兵
12    for(int i=0; i<n; i++)
13        if(b[len]<a[i]) b[++len]=a[i]; //新数大于队尾数，则插入队尾
14        else *lower_bound(b,b+len,a[i])=a[i]; //替换第一个大于a[i]的数(贪心)
15    printf("%d\n", len);
16 }

```

编辑距离

dp[i][j]表示word1到i位置转换成word2到j位置需要的最少步数

$dp[i][j] = \min(dp[i-1][j] + 1, dp[i][j-1] + 1, dp[i-1][j-1] + (word1[i] \neq word2[j]));$

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int _ = 2005;
4  char s[_], t[_];
5  int dp[_][_];
6  int main() {
7     scanf("%s%s", s, t);
8     int slen = strlen(s), tlen = strlen(t);
9     for (int i = 0; i <= slen; ++i) dp[i][0] = i;
10    for (int i = 0; i <= tlen; ++i) dp[0][i] = i;
11    for (int i = 1; i <= slen; ++i) {
12        for (int j = 1; j <= tlen; ++j) {
13            dp[i][j] = std::min(dp[i-1][j], dp[i][j-1]) + 1;
14            dp[i][j] = std::min(dp[i][j], dp[i-1][j-1] + (s[i-1] != t[j-1]));
15        }
16    }
17    printf("%d\n", dp[slen][tlen]);
18 }

```

```

15     }
16 }
17 printf("%d\n", dp[slen][tlen]);
18 }

```

树形DP

没有上司的舞会

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int _ = 6e3 + 5;
4  int n;
5  int a[_][2], b[_], w[_], f[_][2];
6  bool fa[_];
7  void dfs(int u) {
8      f[u][1] = w[u];
9      for (int i = 0; i < b[u]; ++i) {
10         int son = a[u][i];
11         dfs(son);
12         f[u][0] += max(f[son][0], f[son][1]);
13         f[u][1] += f[son][0];
14     }
15 }
16 int main() {
17     ios::sync_with_stdio(false);
18     cin.tie(nullptr);
19     cin >> n;
20     for (int i = 1; i <= n; ++i) cin >> w[i];
21     for (int i = 0; i < n - 1; ++i) {
22         int x, y;
23         cin >> x >> y;
24         a[y][b[y]++] = x;
25         fa[x] = true;
26     }
27     int root = 1;
28     while (fa[root]) root++;
29     dfs(root);
30     cout << max(f[root][0], f[root][1]);
31 }

```

树的重心

```

1 #include <iostream>
2 #include <cstring>
3 #include <algorithm>
4 #include <vector>
5 using namespace std;
6
7 const int N=100010;
8 int n, a, b;
9 vector<int> e[N];
10 int siz[N], pos, ans=1e9;
11
12 void dfs(int x, int fa){
13     siz[x]=1;
14     int mx=0;
15     for(auto y : e[x]){
16         if(y == fa) continue;
17         dfs(y, x);
18         siz[x] += siz[y];
19         mx=max(mx, siz[y]);
20     }
21     mx=max(mx, n-siz[x]);
22     if(mx<ans) ans=mx,pos=x;
23 }
24 int main(){
25     scanf("%d", &n);
26     for(int i=1;i<n;i++){
27         scanf("%d%d",&a,&b);
28         e[a].push_back(b);
29         e[b].push_back(a);
30     }
31     dfs(1, 0);
32     printf("%d\n",ans);
33     return 0;
34 }

```

树的直径

```

1 #include <cstring>
2 #include <iostream>
3 #include <algorithm>
4 using namespace std;
5
6 const int N=10010,M=20010;
7 int n,a,b,c,ans;
8 struct edge{int v,w;};

```

```

9  vector<edge> e[N];
10
11 int dfs(int x,int fa){
12     int d1=0,d2=0;
13     for(auto ed : e[x]){
14         int y=ed.v, z=ed.w;
15         if(y==fa) continue;
16         int d=dfs(y,x)+z;
17         if(d>=d1) d2=d1,d1=d;
18         else if(d>d2) d2=d;
19     }
20     ans=max(ans,d1+d2);
21     return d1;
22 }
23 int main(){
24     cin>>n;
25     for(int i=1; i<n; i++){
26         cin>>a>>b>>c;
27         e[a].push_back({b,c});
28         e[b].push_back({a,c});
29     }
30     dfs(1,-1);
31     cout<<ans;
32 }

```

树的中心

```

1  #include <iostream>
2  #include <cstring>
3  #include <algorithm>
4  #include <vector>
5  using namespace std;
6
7  const int N=20010;
8  int n,a,b,c,ans=2e9;
9  struct edge{int v,w};
10 vector<edge> e[N];
11 int d1[N],d2[N],path[N],up[N];
12
13 void dfs(int x,int fa){
14     for(auto ed : e[x]){
15         int y=ed.v, z=ed.w;
16         if(y==fa) continue;
17         dfs(y, x);
18         if(d1[y]+z>d1[x])

```



```

19     d2[x]=d1[x],d1[x]=d1[y]+z,path[x]=y;
20     else if(d1[y]+z>d2[x]) d2[x]=d1[y]+z;
21 }
22 }
23 void dfs2(int x,int fa){
24     for(auto ed : e[x]){
25         int y=ed.v, z=ed.w;
26         if(y==fa) continue;
27         if(y==path[x])up[y]=max(up[x],d2[x])+z;
28         else up[y]=max(up[x],d1[x])+z;
29         dfs2(y, x);
30     }
31 }
32 int main(){
33     cin>>n;
34     for(int i=1; i<n; i++){
35         cin>>a>>b>>c;
36         e[a].push_back({b,c});
37         e[b].push_back({a,c});
38     }
39     dfs(1, 0);
40     dfs2(1, 0);
41     for(int i=1; i<=n; i++)
42         ans=min(ans,max(d1[i],up[i]));
43     cout<<ans;
44 }

```

树形背包

1

换根dp

1

数位DP

1

区间DP

环形石子合并

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 const int _ = 205;
4 long long a[_], s[_], minn[_][_], maxx[_][_], res_min = LLONG_MAX, res_max =
    -1;
5 int main() {
6     ios::sync_with_stdio(false); cin.tie(nullptr); cout.tie(nullptr);
7     int n; cin >> n;
8     for (int i = 1; i <= n; ++i) {
9         cin >> a[i]; a[i + n] = a[i];
10    }
11    for (int i = 1; i <= 2 * n; ++i) {
12        s[i] = s[i - 1] + a[i];
13    }
14    for (int len = 2; len <= n; ++len) {
15        for (int i = 1; i <= 2 * n - len + 1; ++i) {
16            int j = i + len - 1;
17            minn[i][j] = LLONG_MAX, maxx[i][j] = -1;
18            for (int k = i; k < j; ++k) {
19                minn[i][j] = min(minn[i][j], minn[i][k] + minn[k + 1][j] +
s[j] - s[i - 1]);
20                maxx[i][j] = max(maxx[i][j], maxx[i][k] + maxx[k + 1][j] +
s[j] - s[i - 1]);
21            }
22        }
23    }
24    for (int i = 1; i <= n; ++i) {
25        res_min = min(res_min, minn[i][i + n - 1]);
26        res_max = max(res_max, maxx[i][i + n - 1]);
27    }
28    cout << res_min << '\n' << res_max << '\n';
29 }
```

状压DP

bitset

Demo: 国王问题

行内合法: $!(i \& i \gg 1)$ 为真

行间兼容: $!(a \& b) \&\& !(a \& b \gg 1) \&\& !(a \& b \ll 1)$ 为真

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 int n, k;
4 int cnt;
5 int s[1<<12];
6 int num[1<<12];
7 long long f[12][144][1<<12];
8 int main() {
9     ios::sync_with_stdio(false);
10    cin.tie(nullptr);
11    cout.tie(nullptr);
12    cin >> n >> k;
13    for (int i = 0; i < (1 << n); ++i) {
14        if (!(i & i >> 1)) {
15            s[cnt++] = i;
16            for (int j = 0; j < n; ++j) {
17                num[i] += (i >> j & 1);
18            }
19        }
20    }
21    f[0][0][0] = 1;
22    for (int i = 1; i <= n + 1; ++i) {
23        for (int j = 0; j <= k; ++j) {
24            for (int a = 0; a < cnt; ++a) {
25                for (int b = 0; b < cnt; ++b) {
26                    int c = num[s[a]];
27                    if ((j >= c) && !(s[b] & s[a]) && !(s[b] & (s[a] << 1)) &&
28                        !(s[b] & (s[a] >> 1))) {
29                        f[i][j][a] += f[i - 1][j - c][b];
30                    }
31                }
32            }
33        }
34        cout << f[n + 1][k][0] << '\n';
35        return 0;
36    }

```

Demo:玉米田

```

1 #include <iostream>
2 #include <cstring>
3 #include <algorithm>
4 using namespace std;
5 const int P=1e9;

```

```

6  int n,m;          //行数, 列数
7  int g[14];        //各行的状态值
8  int cnt;          //同一行的合法状态个数
9  int s[1<<14];    //一行的合法状态集
10 int f[14][1<<14];
11 //f[i,a]表示已经种植前i行, 第i行第a个状态时的方案数
12 int main(){
13     cin>>n>>m;
14     for(int i=1; i<=n; i++){
15         for(int j=1; j<=m; j++){
16             int x; cin>>x;
17             g[i]=(g[i]<<1)+x; //各行的状态值
18         }
19         for(int i=0; i<(1<<m); i++) //枚举一行所有状态
20             if(!(i&i>>1)) //如果不存在相邻的1
21                 s[cnt++]=i; //保存一行的合法状态
22         f[0][0]=1;
23         for(int i=1; i<=n+1; i++) //枚举行
24             for(int a=0; a<cnt; a++) //枚举第i行合法状态
25                 for(int b=0; b<cnt; b++) //枚举第i-1行合法状态
26                     if(!(s[a]&s[b]) //不能同列均为1
27                         &&(s[a]&g[i])==s[a]) //种在肥沃土地上
28                         f[i][a]=(f[i][a]+f[i-1][b])%P;
29         printf("%d\n",f[n+1][0]);
30         return 0;
31     }

```

Demo:炮兵阵地

```

1  #include <iostream>
2  #include <cstring>
3  #include <algorithm>
4  using namespace std;
5
6  const int N=110, M=1<<10;
7  int n,m;          //行数, 列数
8  int g[N];         //存储地图各行数值
9  int cnt;          //一行的合法状态个数
10 int s[M];          //一行的合法状态集
11 int num[M];        //每个合法状态包含1的个数
12 int f[N][M][M];    //110*1024*1024*4 = 440MB
13 // f[i][a][b]表示已放好前i行,
14 // 第i行第a个状态, 第i-1行第b个状态时, 能放置的最大数量
15 int main(){
16     cin>>n>>m;

```

```

17  for(int i=1;i<=n;i++)
18      for(int j=0;j<m;j++){
19          char c; cin>>c;
20          if(c=='P') g[i]+=1<<(m-j-1); //地图各行数值
21      }
22  for(int i=0; i<(1<<m); i++) //枚举一行的所有状态
23      if(!(i&i>>1) && !(i&i>>2)){ //如果不存在11和101
24          s[cnt++]=i; //保存一行的合法状态
25          for(int j=0; j<m; j++)
26              num[i]+=(i>>j&1); //每个合法状态包含1的个数
27      }
28  for(int i=1; i<=n+2; i++) //枚举行
29      for(int a=0; a<cnt; a++) //枚举第i行合法状态
30      for(int b=0; b<cnt; b++) //枚举第i-1行合法状态
31      for(int c=0; c<cnt; c++) //枚举第i-2行合法状态
32          if(!(s[a]&s[b])&&!(s[a]&s[c])&&!(s[b]&s[c])
33              &&(g[i]&s[a])==s[a]&&(g[i-1]&s[b])==s[b])
34              f[i][a][b]=max(f[i][a][b],f[i-1][b][c]+num[s[a]]);
35  cout<<f[n+2][0][0]<<endl;
36  return 0;
37  }

```

Demo:Explode 'Em All (炸十字形)

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  char s[30][30];
4  int r[30], dp[(1 << 25) + 100];
5  int has[(1 << 25) + 100];
6  int lowbit(int x) { return x & (-x); }
7  int cal(int x) {
8      int cnt = 0;
9      while (x) {
10         x -= lowbit(x);
11         ++cnt;
12     }
13     return cnt;
14 }
15 int main() {
16     int n, m;
17     scanf("%d %d", &n, &m);
18     for (int i = 0; i < n; ++i) {
19         has[(1 << i)] = i;
20     }
21     memset(r, 0, sizeof(r));

```

```

22     for (int i = 0; i < n; ++i) {
23         scanf("%s", s[i]);
24         for (int j = 0; j < m; ++j) {
25             if (s[i][j] == '*') {
26                 r[i] = r[i] | (1 << j);
27             }
28         }
29     }
30     memset(dp, 0, sizeof(dp));
31     int ans = 0x3f3f3f3f;
32     int rcnt, ccnt;
33     for (int i = 1; i < (1 << n); ++i) {
34         dp[i] = dp[i - lowbit(i)] | r[has[lowbit(i)]];
35         rcnt = n - cal(i);
36         ccnt = cal(dp[i]);
37         ans = min(ans, max(rcnt, ccnt));
38     }
39     printf("%d\n", ans);
40     return 0;
41 }

```

Demo:樱桃炸弹（炸3*3区域）

1

DP优化

单调队列优化DP

Demo:

$$f[i] = \min(f[j]) + w[i], i - m \leq j \leq i - 1$$

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 2e5 + 10;
4  int n, m, w[N], f[N], q[N];
5  int main() {
6      cin >> n >> m;
7      for (int i = 1; i <= n; ++i) cin >> w[i];
8      int ans = 0x3f3f3f3f;
9      int h = 1, t = 0;
10     for (int i = 1; i <= n; ++i) {

```

```

11         while (h <= t && f[q[t]] >= f[i - 1]) t--;
12         q[++t] = i - 1;
13         if (q[h] < i - m) ++h;
14         f[i] = f[q[h]] + w[i];
15         if (i > n - m) ans = min(ans, f[i]);
16     }
17     cout << ans;
18 }

```

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 2e5 + 10;
4  #define int long long
5  int n, m, w[N], f[N], q[N];
6  bool check(int x) {
7      int h = 1, t = 0;
8      for (int i = 1; i <= n; ++i) {
9          while (h <= t && f[q[t]] >= f[i - 1]) t--;
10         q[++t] = i - 1;
11         if (q[h] < i - x) ++h;
12         f[i] = f[q[h]] + w[i];
13         if (i > n - x && f[i] <= m) return true;
14     }
15     return false;
16 }
17 signed main() {
18     cin >> n >> m;
19     for (int i = 1; i <= n; ++i) cin >> w[i];
20     int l = 0, r = n;
21     int ans = n;
22     while (l <= r) {
23         int mid = (l + r) >> 1;
24         if (check(mid)) {
25             ans = mid;
26             r = mid - 1;
27         } else {
28             l = mid + 1;
29         }
30     }
31     cout << ans - 1;
32 }

```

Demo:

$$f[i] = \max(f[j]) + a[i], i - R \leq j \leq i - L$$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 2e5 + 10;
4 int n, L, R, a[N], f[N], q[N];
5 signed main() {
6     cin >> n >> L >> R;
7     for (int i = 0; i <= n; ++i) cin >> a[i];
8     memset(f, -0x3f, sizeof(f));
9     f[0] = 0;
10    int ans = -0x3f3f3f3f;
11    int h = 1, t = 0;
12    for (int i = L; i <= n; ++i) {
13        while (h <= t && f[q[t]] <= f[i - L]) t--;
14        q[++t] = i - L;
15        if (q[h] < i - R) ++h;
16        f[i] = f[q[h]] + a[i];
17        if (i > n - R) ans = max(ans, f[i]);
18    }
19    cout << ans;
20 }

```

斜率优化DP

double与long double

$$f[i] = \min(f[j] + a[i] * b[j]), 0 \leq j \leq i - 1$$

Demo:

$$f[i] = \min(f[j] + (s[i] - s[j])^2 + m), 0 \leq j \leq i - 1$$

$$\Rightarrow f[j] + s[j]^2 = 2s[i]s[j] + f[i] - s[i]^2 - m$$

分析：

(1) 新点 $i-1$ 与队尾点斜率 \leq 队尾邻点直线的斜率，则队尾出队，删除无用点，维护下凸壳

(2) 新点 $i-1$ 入队

(3) 若队头邻点直线斜率 $\leq k_i$ ，则队头出队

(4) 此时队头为最优决策点

```

1 #include <iostream>
2 #include <algorithm>
3 using namespace std;

```



```

4 typedef long long LL;
5 const int N = 500010;
6 int n, m, q[N];
7 LL s[N], f[N];
8 double slope(int i, int j) {
9     return (double)(f[i] + s[i] * s[i] - f[j] - s[j] * s[j]) / (s[i] == s[j] ?
1e-9 : s[i] - s[j]);
10 }
11 int main() {
12     while (~scanf("%d%d", &n, &m)) {
13         for (int i = 1; i <= n; ++i) {
14             scanf("%lld", &s[i]);
15             s[i] += s[i - 1];
16         }
17         int h = 1, t = 0;
18         for (int i = 1; i <= n; ++i) {
19             while (h < t && slope(i - 1, q[t]) <= slope(q[t], q[t - 1])) t--;
20             q[++t] = i - 1;
21             while (h < t && slope(q[h + 1], q[h]) <= 2 * s[i]) h++;
22             int j = q[h];
23             f[i] = f[j] + (s[i] - s[j]) * (s[i] - s[j]) + m;
24         }
25         printf("%lld\n", f[n]);
26     }
27 }

```

```

1 #include <bits/stdc++.h>
2 typedef long long LL;
3 const int N = 500010;
4 int n, m, q[N];
5 LL s[N], f[N];
6 LL dx(int i, int j) { return s[i] - s[j]; }
7 LL dy(int i, int j) { return f[i] + s[i] * s[i] - f[j] - s[j] * s[j]; }
8 int main() {
9     while (~scanf("%d%d", &n, &m)) {
10         for (int i = 1; i <= n; ++i) {
11             scanf("%lld", &s[i]);
12             s[i] += s[i - 1];
13         }
14         int h = 1, t = 0;
15         for (int i = 1; i <= n; ++i) {
16             while (h < t && dy(i - 1, q[t]) * dx(q[t], q[t - 1])
17                 <= dx(i - 1, q[t]) * dy(q[t], q[t - 1])) --t;
18             q[++t] = i - 1;

```

```

19         while (h < t && dy(q[h + 1], q[h]) <= dx(q[h + 1], q[h]) * 2 *
s[i]) ++h;
20         int j = q[h];
21         f[i] = f[j] + (s[i] - s[j]) * (s[i] - s[j]) + m;
22     }
23     printf("%lld\n", f[n]);
24 }
25 }

```

Demo:

$$f[i] = \min(f[j] + (i - (j + 1) + s[i] - s[j] - L)^2)$$

令 $a[i] = s[i] + i$, $b[i] = s[i] + i + L + 1$, 可得:

$$f[j] + b[j]^2 = 2 * a[i] * b[j] + f[i] - a[i]^2$$

```

1 #include <iostream>
2 #include <algorithm>
3 using namespace std;
4 const int _ = 5e4 + 5;
5 typedef long long ll;
6 ll n, L, q[_];
7 ll s[_], f[_];
8 ll a(int i) { return s[i] + i; }
9 ll b(int i) { return a(i) + L + 1; }
10 ll X(int i) { return b(i); }
11 ll Y(int i) { return f[i] + b(i) * b(i); }
12 double slope(int i, int j) {
13     return (double)(Y(i) - Y(j)) / (X(i) == X(j) ? 1e-9 : X(i) - X(j));
14 }
15 int main() {
16     ios::sync_with_stdio(false); cin.tie(nullptr); cout.tie(nullptr);
17     cin >> n >> L;
18     for (ll i = 1; i <= n; ++i) cin >> s[i], s[i] += s[i - 1];
19     ll h = 1, t = 0;
20     for (ll i = 1; i <= n; ++i) {
21         while (h < t && slope(i - 1, q[t]) <= slope(q[t], q[t - 1])) --t;
22         q[++t] = i - 1;
23         while (h < t && slope(q[h + 1], q[h]) <= 2 * a(i)) ++h;
24         ll j = q[h];
25         f[i] = f[j] + (a(i) - b(j)) * (a(i) - b(j));
26     }
27     cout << f[n];
28     return 0;
29 }

```

Demo:

$$f[i] = \max(f[j] + a * (s[i] - s[j])^2 + b * (s[i] - s[j]) + c), 1 \leq j \leq i, a < 0$$
$$\Rightarrow f[j] + a * s[j]^2 - b * s[j] = 2a * s[i] * s[j] + f[i] - a * s[i]^2 - b * s[i] - c$$

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 const int _ = 1e6 + 5;
4 typedef long long ll;
5 ll n, a, b, c, q[_];
6 ll s[_], f[_];
7 ll X(int i) { return s[i]; }
8 ll Y(int i) { return f[i] + a * s[i] * s[i] - b * s[i]; }
9 double slope(int i, int j) {
10     return (double)(Y(i) - Y(j)) / (X(i) == X(j) ? 1e-9 : X(i) - X(j));
11 }
12 int main() {
13     ios::sync_with_stdio(false); cin.tie(0); cout.tie(0);
14     cin >> n >> a >> b >> c;
15     for (ll i = 1; i <= n; ++i) cin >> s[i], s[i] += s[i - 1];
16     ll h = 1, t = 0;
17     for (ll i = 1; i <= n; ++i) {
18         while (h < t && slope(i - 1, q[t]) >= slope(q[t], q[t - 1])) --t;
19         q[++t] = i - 1;
20         while (h < t && slope(q[h + 1], q[h]) >= 2 * a * s[i]) ++h;
21         ll j = q[h];
22         f[i] = f[j] + a * (s[i] - s[j]) * (s[i] - s[j]) + b * (s[i] - s[j]) +
c;
23     }
24     cout << f[n];
25     return 0;
26 }
```

四边形不等式优化DP

石子合并最小值以及四边形不等式优化

石子合并最大值

$dp[i][j] = \max(dp[i][j - 1], dp[i + 1][j]) + w[i][j];$

环形石子合并

```
1 // 开两倍数组
2 #include <bits/stdc++.h>
```

```

3 using namespace std;
4 const int _ = 205;
5 long long a[_], b[_][_], s[_], minn[_][_], maxx[_][_], res_min = 1e9, res_max
  = -1;
6 int main() {
7     ios::sync_with_stdio(false); cin.tie(nullptr); cout.tie(nullptr);
8     int n; cin >> n;
9     for (int i = 1; i <= n; ++i) cin >> a[i], a[i + n] = a[i];
10    for (int i = 1; i <= 2 * n; ++i) b[i][i] = i, s[i] = s[i - 1] + a[i];
11    for (int len = 2; len <= n; ++len) {
12        for (int i = 1; i <= 2 * n - len + 1; ++i) {
13            int j = i + len - 1;
14            minn[i][j] = 1e9, maxx[i][j] = -1;
15            for (int k = i; k < j; ++k) {
16                minn[i][j] = min(minn[i][j], minn[i][k] + minn[k + 1][j] +
s[j] - s[i - 1]);
17                maxx[i][j] = max(maxx[i][j], maxx[i][k] +
maxx[k + 1][j] + s[j] - s[i - 1]);
18            }
19        }
20    }
21    for (int i = 1; i <= n; ++i) res_min = min(res_min, minn[i][i + n - 1]),
res_max = max(res_max, maxx[i][i + n - 1]);
22    cout << res_min << '\n' << res_max << '\n';
23 }

```

```

1 // 开两倍数组
2 #include <bits/stdc++.h>
3 using namespace std;
4 const int _ = 205;
5 long long a[_], b[_][_], s[_], minn[_][_], maxx[_][_], res_min = 1e9, res_max
  = -1;
6 int main() {
7     ios::sync_with_stdio(false); cin.tie(nullptr); cout.tie(nullptr);
8     int n; cin >> n;
9     for (int i = 1; i <= n; ++i) cin >> a[i], a[i + n] = a[i];
10    for (int i = 1; i <= 2 * n; ++i) b[i][i] = i, s[i] = s[i - 1] + a[i];
11    for (int len = 2; len <= n; ++len) {
12        for (int i = 1; i <= 2 * n - len + 1; ++i) {
13            int j = i + len - 1;
14            minn[i][j] = 1e9, maxx[i][j] = -1;
15            for (int k = b[i][j - 1]; k <= b[i + 1][j]; ++k) {
16                if (minn[i][j] > minn[i][k] + minn[k + 1][j] + s[j] - s[i -
17                1]) {

```

```

18         b[i][j] = k;
19     }
20 }
21     maxx[i][j] = max(maxx[i][j - 1], maxx[i + 1][j]) + s[j] - s[i - 1];
22 }
23 }
24     for (int i = 1; i <= n; ++i) res_min = min(res_min, minn[i][i + n - 1]),
    res_max = max(res_max, maxx[i][i + n - 1]);
25     cout << res_min << '\n' << res_max << '\n';
26 }

```

二、字符串

最小表示法

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int _ = 6e5 + 10;
4  int a[_];
5  int main() {
6      ios::sync_with_stdio(false);
7      cin.tie(nullptr);
8      int n;
9      cin >> n;
10     for (int i = 1; i <= n; ++i) cin >> a[i], a[i + n] = a[i];
11     int i = 1, j = 2, k = 0;
12     while (i <= n && j <= n) {
13         for (k = 0; k < n && a[i + k] == a[j + k]; ++k);
14         a[i + k] > a[j + k] ? i = i + k + 1 : j = j + k + 1;
15         if (i == j) ++j;
16     }
17     int t = min(i, j);
18     for (int i = t; i < t + n; ++i) cout << a[i] << " ";
19     return 0;
20 }

```

KMP

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 1e6 + 5;

```

```

4 char s[N], p[N];
5 int nxt[N];
6 int main() {
7     scanf("%s%s", s + 1, p + 1);
8     int slen = strlen(s + 1), plen = strlen(p + 1);
9     nxt[1] = 0;
10    for (int i = 2, j = 0; i <= plen; ++i) {
11        while (j > 0 && p[i] != p[j + 1]) j = nxt[j];
12        if (p[i] == p[j + 1]) ++j;
13        nxt[i] = j;
14    }
15    for (int i = 1, j = 0; i <= slen; ++i) {
16        while (j > 0 && s[i] != p[j + 1]) j = nxt[j];
17        if (s[i] == p[j + 1]) ++j;
18        if (j == plen) {
19            printf("%d\n", i - plen + 1);
20        }
21    }
22    for (int i = 1; i <= plen; ++i) {
23        printf("%d ", nxt[i]);
24    }
25 }

```

统计前缀个数

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 1e6 + 5;
4 char p[N];
5 int nxt[N], cnt[N];
6 int main() {
7     scanf("%s", p + 1);
8     int plen = strlen(p + 1);
9     nxt[1] = 0;
10    for (int i = 2, j = 0; i <= plen; ++i) {
11        while (j > 0 && p[i] != p[j + 1]) j = nxt[j];
12        if (p[i] == p[j + 1]) ++j;
13        nxt[i] = j;
14    }
15    for (int i = 1; i <= plen; ++i) cnt[i]++;
16    for (int i = plen; i >= 1; --i) {
17        cnt[nxt[i]] += cnt[i];
18    }
19    for (int i = 1; i <= plen; ++i) {
20        printf("%d ", cnt[i]);

```

```
21     }
22 }
```

扩展KMP

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  const int _ = 2e7 + 5;
4  #define int long long
5  char s[_], t[_];
6  int z[_], p[_];
7  void get_z(char* s, int n) {
8      z[1] = n;
9      for (int i = 2, l = 0, r = 0; i <= n; ++i) {
10         if (i <= r) z[i] = min(z[i - l + 1], r - i + 1);
11         while (1 + z[i] <= n && i + z[i] <= n && s[1 + z[i]] == s[i + z[i]])
            z[i]++;
12         if (i + z[i] - 1 > r) l = i, r = i + z[i] - 1;
13     }
14 }
15 void get_p(char* s, int n, char* t, int m) {
16     for (int i = 1, l = 0, r = 0; i <= m; ++i) {
17         if (i <= r) p[i] = min(z[i - l + 1], r - i + 1);
18         while (1 + p[i] <= n && i + p[i] <= m && s[1 + p[i]] == t[i + p[i]])
            p[i]++;
19         if (i + p[i] - 1 > r) l = i, r = i + p[i] - 1;
20     }
21 }
22 int xxor(int a[], int n) {
23     if (n == 0) return 0;
24     int res = 0;
25     for (int i = 1; i <= n; ++i) {
26         res ^= i * (a[i] + 1);
27     }
28     return res;
29 }
30 signed main() {
31     scanf("%s%s", t + 1, s + 1);
32     int tlen = strlen(t + 1);
33     int slen = strlen(s + 1);
34     get_z(s, slen);
35     get_p(s, slen, t, tlen);
36     printf("%lld\n", xxor(z, slen));
37     printf("%lld\n", xxor(p, tlen));
38 }
```

Trie树

Trie模板

```
1 #include <iostream>
2 #include <cstring>
3 #include <string>
4 #include <vector>
5 using namespace std;
6 constexpr int N = 3e6 + 5;
7 int getnum(char c) {
8     if (c >= 'A' && c <= 'Z') return c - 'A';
9     else if (c >= 'a' && c <= 'z') return c - 'a' + 26;
10    else return c - '0' + 52;
11 }
12 struct Node {
13     int son[65]; // 48 97
14     int num;
15 }t[N];
16 class Trie {
17 public:
18     int cnt;
19     Trie() {
20         memset(t, 0, sizeof(t));
21         cnt = 0;
22     }
23     void insert(string s) {
24         int now = 0;
25         for (int i = 0; i < s.size(); ++i) {
26             int c = getnum(s[i]);
27             if (!t[now].son[c]) t[now].son[c] = ++cnt;
28             now = t[now].son[c];
29             ++t[now].num;
30         }
31     }
32     int find(string s) {
33         int now = 0;
34         for (int i = 0; i < s.size(); ++i) {
35             int c = getnum(s[i]);
36             if (!t[now].son[c]) return 0;
37             now = t[now].son[c];
38         }
39         return t[now].num;
40     }
```



```

41 };
42 void solve() {
43     int n, q;
44     cin >> n >> q;
45     Trie t;
46     string s;
47     for (int i = 0; i < n; ++i) {
48         cin >> s;
49         t.insert(s);
50     }
51     while (q--) {
52         string s;
53         cin >> s;
54         cout << t.find(s) << '\n';
55     }
56 }
57 int main() {
58     ios::sync_with_stdio(false);
59     cin.tie(nullptr);
60     cout.tie(nullptr);
61     int t;
62     cin >> t;
63     while (t--) {
64         solve();
65     }
66     return 0;
67 }

```

```

1  #include <cstdio>
2  #include <cstring>
3  using namespace std;
4  const int N = 5e5 + 5;
5  struct Node {
6      bool repeat = false;
7      int son[26]{ 0 };
8      int num = 0;
9  }t[N];
10 int cnt = 0;
11 void insert(char* s) {
12     int now = 0;
13     for (int i = 0; s[i]; ++i) {
14         int c = s[i] - 'a';
15         if (!t[now].son[c]) t[now].son[c] = ++cnt;
16         now = t[now].son[c];
17     }

```

```

18     ++t[now].num;
19 }
20 int find(char* s) {
21     int now = 0;
22     for (int i = 0; s[i]; ++i) {
23         int c = s[i] - 'a';
24         if (!t[now].son[c]) return 3;
25         now = t[now].son[c];
26     }
27     if (t[now].num == 0) return 3;
28     if (t[now].repeat == false) {
29         t[now].repeat = true;
30         return 1;
31     }
32     return 2;
33 }
34 char s[55];
35 int main() {
36     int n;
37     scanf("%d", &n);
38     while (n--) {
39         scanf("%s", s);
40         insert(s);
41     }
42     int m;
43     scanf("%d", &m);
44     while (m--) {
45         scanf("%s", s);
46         int r = find(s);
47         if (r == 1) puts("OK");
48         if (r == 2) puts("REPEAT");
49         if (r == 3) puts("WRONG");
50     }
51 }

```

AC自动机

AC自动机(简单版)模板题

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 const int _ = 1e6 + 5;
4 int t[_][26], fail[_], cnt[_];
5 int pos = 0;
6 void insert(char* p) {

```

```

7     int now = 0;
8     for (int i = 0; p[i]; ++i) {
9         int ch = p[i] - 'a';
10        if (t[now][ch] == 0) t[now][ch] = ++pos;
11        now = t[now][ch];
12    }
13    ++cnt[now];
14 }
15 void getFail() {
16     queue<int> q;
17     for (int i = 0; i < 26; ++i) {
18         if (t[0][i]) {
19             fail[t[0][i]] = 0;
20             q.push(t[0][i]);
21         }
22     }
23     while (!q.empty()) {
24         int now = q.front();
25         q.pop();
26         for (int i = 0; i < 26; ++i) {
27             if (t[now][i]) {
28                 fail[t[now][i]] = t[fail[now]][i];
29                 q.push(t[now][i]);
30             } else {
31                 t[now][i] = t[fail[now]][i];
32             }
33         }
34     }
35 }
36 int query(char* s) {
37     int ans = 0;
38     int now = 0;
39     for (int i = 0; s[i]; ++i) {
40         int ch = s[i] - 'a';
41         now = t[now][ch];
42         for (int j = now; j && cnt[j] != -1; j = fail[j]) {
43             ans += cnt[j];
44             cnt[j] = -1;
45         }
46     }
47     return ans;
48 }
49 char s[_];
50 int main() {
51     int T;
52     scanf("%d", &T);
53     while (T--) {

```

```

54     scanf("%s", s);
55     insert(s);
56 }
57 getFail();
58 scanf("%s", s);
59 printf("%d\n", query(s));
60 }

```

AC自动机(加强版)模板

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int _ = 1e6 + 5;
4  int t[_][26], fail[_], cnt[_], tag[_];
5  int pos;
6  char s[_];
7  char p[155][75];
8  void insert(int idx) {
9      int now = 0;
10     for (int i = 0; p[idx][i]; ++i) {
11         int ch = p[idx][i] - 'a';
12         if (!t[now][ch]) t[now][ch] = ++pos;
13         now = t[now][ch];
14     }
15     tag[now] = idx;
16 }
17 void getFail() {
18     queue<int> q;
19     for (int i = 0; i < 26; ++i) {
20         if (t[0][i]) {
21             fail[t[0][i]] = 0;
22             q.push(t[0][i]);
23         }
24     }
25     while (!q.empty()) {
26         int now = q.front();
27         q.pop();
28         for (int i = 0; i < 26; ++i) {
29             if (t[now][i]) {
30                 fail[t[now][i]] = t[fail[now]][i];
31                 q.push(t[now][i]);
32             } else {
33                 t[now][i] = t[fail[now]][i];
34             }
35         }
36     }
37 }

```

```

36     }
37 }
38 void query() {
39     int now = 0;
40     for (int i = 0; s[i]; ++i) {
41         int ch = s[i] - 'a';
42         now = t[now][ch];
43         for (int j = now; j; j = fail[j]) {
44             ++cnt[tag[j]];
45         }
46     }
47 }
48 int main() {
49     int N;
50     while (~scanf("%d", &N) && N) {
51         memset(t, 0, sizeof(t));
52         memset(cnt, 0, sizeof(cnt));
53         memset(tag, 0, sizeof(tag));
54         pos = 0;
55         for (int i = 1; i <= N; ++i) {
56             scanf("%s", p[i]);
57             insert(i);
58         }
59         getFail();
60         scanf("%s", s);
61         query();
62         int maxx = -1;
63         for (int i = 1; i <= N; ++i) {
64             if (cnt[i] > maxx) maxx = cnt[i];
65         }
66         if (maxx != -1) printf("%d\n", maxx);
67         for (int i = 1; i <= N; ++i) {
68             if (cnt[i] == maxx) {
69                 printf("%s\n", p[i]);
70             }
71         }
72     }
73 }

```

回文自动机

```

1 class Solution {
2 public:
3     static constexpr int N = 1e4 + 5;
4     int c[N];

```

```

5     struct node {
6         int len, fail, son[26], sz;
7         void init(int l) {
8             memset(son, 0, sizeof(son));
9             fail = sz = 0;
10            len = l;
11        }
12    }t[N];
13    long long num, last[2], ans, L, R;
14    void init() {
15        last[0] = last[1] = 0;
16        ans = 0, num = 1;
17        L = 5e3 + 5, R = 5e3 + 4;
18        t[0].init(0);
19        memset(c, -1, sizeof(c));
20        t[1].init(-1);
21        t[0].fail = 1;
22    }
23    int getfail(int p) {
24        while (c[R - t[p].len - 1] != c[R]) p = t[p].fail;
25        return p;
26    }
27    void insert(int x) {
28        c[++R] = x;
29        int f = getfail(last[1]);
30        int now = t[f].son[x];
31        if (!now) {
32            now = ++num;
33            t[now].init(t[f].len + 2);
34            t[now].fail = t[getfail(t[f].fail)].son[x];
35            t[now].sz = t[t[now].fail].sz + 1;
36            t[f].son[x] = now;
37        }
38        last[1] = now;
39        if (R - L + 1 == t[now].len) last[0] = now;
40        ans += t[now].sz;
41    }
42    int countSubstrings(string s) {
43        init();
44        for (int i = 0; i < s.size(); ++i) {
45            insert(s[i] - 'a');
46        }
47        return ans;
48    }
49 };

```

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 2e6 + 5;
4  char s[N];
5  int sa[N], cnt[N], x[N], y[N], rk[N], h[N];
6  int n;
7  void calc_sa() {
8      int m = 127;
9      for (int i = 1; i <= m; ++i) cnt[i] = 0;
10     for (int i = 1; i <= n; ++i) ++cnt[x[i] = s[i]];
11     for (int i = 2; i <= m; ++i) cnt[i] += cnt[i - 1];
12     for (int i = n; i >= 1; --i) sa[cnt[x[i]]--] = i;
13     for (int k = 1; k <= n; k <= 1) {
14         int p = 0;
15         for (int i = n - k + 1; i <= n; ++i) y[++p] = i;
16         for (int i = 1; i <= n; ++i) if (sa[i] > k) y[++p] = sa[i] - k;
17         for (int i = 1; i <= m; ++i) cnt[i] = 0;
18         for (int i = 1; i <= n; ++i) ++cnt[x[i]];
19         for (int i = 2; i <= m; ++i) cnt[i] += cnt[i - 1];
20         for (int i = n; i >= 1; --i) sa[cnt[x[y[i]]]--] = y[i], y[i] = 0;
21         swap(x, y);
22         x[sa[1]] = 1;
23         p = 1;
24         for (int i = 2; i <= n; ++i) {
25             x[sa[i]] = (y[sa[i]] == y[sa[i - 1]] && y[sa[i] + k] == y[sa[i - 1]
26         })
27         if (p >= n) break;
28         m = p;
29     }
30 }
31 void getheight() {
32     int k = 0;
33     for (int i = 1; i <= n; ++i) rk[sa[i]] = i;
34     for (int i = 1; i <= n; ++i) {
35         if (rk[1] == 1) continue;
36         if (k) k--;
37         int j = sa[rk[i] - 1];
38         while (i + k <= n && j + k <= n && s[i + k] == s[j + k]) ++k;
39         h[rk[i]] = k;
40     }
41 }
42 int main() {
43     scanf("%d", &n);
44     scanf("%s", s + 1);

```

```

45     n = strlen(s + 1);
46     calc_sa();
47     getheight();
48     long long res = 0LL;
49     for (int i = 1; i <= n; ++i) {
50         res += n + 1 - sa[i] - h[i];
51     }
52     printf("%lld", res);
53     return 0;
54 }

```

三、数据结构

手写栈

```

1 struct MyStack {
2     int a[N];
3     int t = 0;
4     void push(int x) { a[++t] = x; }
5     int top() { return a[t]; }
6     void pop() { t--; }
7     int empty() { return t == 0 ? 1 : 0; }
8 }stk;

```

单调栈

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 3e6 + 5;
4 int h[N];
5 int ans[N];
6 int main() {
7     int n;
8     scanf("%d", &n);
9     for (int i = 1; i <= n; ++i) scanf("%d", &h[i]);
10    stack<int> stk;
11    for (int i = n; i >= 1; --i) {
12        while (!stk.empty() && h[stk.top()] <= h[i]) stk.pop();
13        if (stk.empty()) ans[i] = 0;
14        else ans[i] = stk.top();
15        stk.push(i);

```



```

16     }
17     for (int i = 1; i <= n; ++i) printf("%d ", ans[i]);
18     return 0;
19 }

```

单调队列

```

1  #include <iostream>
2  using namespace std;
3  const int N=1000010;
4  int a[N], q[N];
5  int main(){
6      int n, k; scanf("%d%d", &n, &k);
7      for(int i=1; i<=n; i++) scanf("%d", &a[i]);
8      // 维护窗口最小值
9      int h=1, t=0; //清空队列
10     for(int i=1; i<=n; i++){ //枚举序列
11         while(h<=t&&a[q[t]]>=a[i]) t--; //队尾出队(队列不空且新元素更优)
12         q[++t]=i; //队尾入队(存储下标 方便判断队头出队)
13         if(q[h]<i-k+1) h++; //队头出队(队头元素滑出窗口)
14         if(i>=k) printf("%d ", a[q[h]]); //使用最值
15     }
16     puts("");
17     // 维护窗口最大值
18     h=1, t=0;
19     for(int i=1; i<=n; i++){
20         while(h<=t&&a[q[t]]<=a[i]) t--;
21         q[++t]=i;
22         if(q[h]<i-k+1) h++;
23         if(i>=k) printf("%d ", a[q[h]]);
24     }
25 }

```

二叉堆

```

1  #include <cstdio>
2  #include <algorithm>
3  using namespace std;
4  const int N = 1e6 + 5;
5  int heap[N];
6  int len = 0;
7  void push(int x) {

```

```
8     heap[++len] = x;
9     int i = len;
10    while (i > 1 && heap[i / 2] > heap[i]) {
11        swap(heap[i / 2], heap[i]);
12        i /= 2;
13    }
14 }
15 void pop() {
16     heap[1] = heap[len--];
17     int i = 1;
18     while (2 * i <= len) {
19         int son = 2 * i;
20         if (son < len && heap[son + 1] < heap[son]) {
21             son++;
22         }
23         if (heap[son] < heap[i]) {
24             swap(heap[son], heap[i]);
25             i = son;
26         }
27         else {
28             break;
29         }
30     }
31 }
32 int main() {
33     int n;
34     scanf("%d", &n);
35     while (n-- > 0) {
36         int op;
37         scanf("%d", &op);
38         switch(op) {
39             case 1:
40                 int x;
41                 scanf("%d", &x);
42                 push(x);
43                 break;
44             case 2:
45                 printf("%d\n", heap[1]);
46                 break;
47             case 3:
48                 pop();
49                 break;
50             default:
51                 break;
52         }
53     }
54     return 0;
```

对顶堆

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 priority_queue<int> a; // 大根堆
4 priority_queue<int,vector<int>,greater<int>> b;
5 int main() {
6     int n;
7     cin >> n;
8     for (int i = 1; i <= n; ++i) {
9         int x;
10        cin >> x;
11        if (b.empty() || x >= b.top()) b.push(x);
12        else a.push(x);
13        while (b.size() > k) a.push(b.top()), b.pop();
14        while (b.size() < k) b.push(a.top()), a.pop();
15        cout << b.top();
16        b.pop();
17    }
18 }
```

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 priority_queue<int> a; // 大根堆
4 priority_queue<int,vector<int>,greater<int>> b;
5 int main() {
6     int n, w;
7     cin >> n >> w;
8     for (int i = 1; i <= n; ++i) {
9         int x;
10        cin >> x;
11        if (b.empty() || x >= b.top()) b.push(x);
12        else a.push(x);
13        int k = max(1, i * w / 100);
14        while (b.size() > k) a.push(b.top()), b.pop();
15        while (b.size() < k) b.push(a.top()), a.pop();
16        cout << b.top() << " ";
17        //b.pop();
18    }
19 }
```

并查集

路径压缩

```
1 #include <iostream>
2 #include <cstring>
3 #include <algorithm>
4 using namespace std;
5 const int N=100010;
6 int n,m,x,y,z;
7 int fa[N];
8 int find(int x){
9     if(fa[x]==x) return x;
10    return fa[x]=find(fa[x]);
11 }
12 void unionset(int x,int y){
13     fa[find(x)]=find(y);
14 }
15 int main(){
16     cin>>n>>m;
17     for(int i=1;i<=n;i++) fa[i]=i;
18     while(m --){
19         cin>>z>>x>>y;
20         if(z==1) unionset(x,y);
21         else{
22             x=find(x),y=find(y);
23             if(x==y) puts("Y");
24             else puts("N");
25         }
26     }
27     return 0;
28 }
```

路径压缩+按秩合并

```
1 //路径压缩 + 按秩合并
2 #include <iostream>
3 #include <cstring>
4 #include <algorithm>
5 #include <vector>
6 using namespace std;
7 const int N=100010;
8 int n,m,x,y,z;
```

```

9  int fa[N];
10 int find(int x){
11     if(fa[x]==x) return x;
12     return fa[x]=find(fa[x]);
13 }
14 //记录并初始化子树的大小为1
15 vector<int> siz(N,1);
16 void unionset(int x,int y){
17     x=find(x),y=find(y);
18     if(x==y) return;
19     if(siz[x]>siz[y]) swap(x,y);
20     fa[x]=y;
21     siz[y]+=siz[x];
22 }
23 int main(){
24     cin>>n>>m;
25     for(int i=1;i<=n;i++) fa[i]=i;
26     while(m--){
27         cin>>z>>x>>y;
28         if(z==1) unionset(x,y);
29         else{
30             x=find(x),y=find(y);
31             if(x==y) puts("Y");
32             else puts("N");
33         }
34     }
35     return 0;
36 }

```

最小体力消耗路径

```

1  class Solution {
2  public:
3      struct Point {
4          int x, y, v;
5          Point(int x, int y, int v): x(x), y(y), v(v) {}
6      };
7      int minimumEffortPath(vector<vector<int>>& heights) {
8          int m = heights.size();
9          int n = heights[0].size();
10         vector<Point> edges;
11         for (int i = 0; i < m; ++i) {
12             for (int j = 0; j < n; ++j) {
13                 int id = i * n + j;

```

```

14         if (i > 0) edges.emplace_back(id - n, id, abs(heights[i][j] -
heights[i - 1][j]));
15         if (j > 0) edges.emplace_back(id - 1, id, abs(heights[i][j] -
heights[i][j - 1]));
16     }
17 }
18 sort(edges.begin(), edges.end(), [](const auto& e1, const auto& e2) {
19     return e1.v < e2.v;
20 });
21 vector<int> par(m * n);
22 iota(par.begin(), par.end(), 0);
23 function<int(int)> find = [&](int x) -> int {
24     return par[x] == x ? x : par[x] = find(par[x]);
25 };
26 function<void(int,int)> unite = [&](int x, int y) -> void {
27     int fx = find(x);
28     int fy = find(y);
29     par[fx] = fy;
30 };
31 for (const auto& [x, y, v]: edges) {
32     unite(x, y);
33     if (find(0) == find(m * n - 1)) {
34         return v;
35     }
36 }
37 return 0;
38 }
39 };
40

```

树状数组

树状数组求逆序对

```

1 #include <iostream>
2 #include <cstring>
3 #include <string>
4 #include <algorithm>
5 using namespace std;
6 const int N = 5e5 + 5;
7 #define int long long
8 struct Node {
9     int vis, id;
10     bool operator<(const Node& t) const {
11         return vis < t.vis;

```

```

12     }
13 }a[N];
14 int b[N];
15 int rev[N];
16 int tree[N];
17 struct BIT {
18     int n;
19     BIT(const int& n = 0) : n(n) {
20         memset(tree, 0, sizeof(tree));
21     }
22     int lowbit(int x) {
23         return x & (-x);
24     }
25     void update(int x, int k) {
26         for (; x <= n; x += lowbit(x)) {
27             tree[x] += k;
28         }
29     }
30     int query(int x) {
31         int res = 0;
32         for (; x > 0; x -= lowbit(x)) {
33             res += tree[x];
34         }
35         return res;
36     }
37     int sum(int l, int r) { // [l, r)
38         return query(r) - query(l);
39     }
40 };
41 void solve() {
42     int n;
43     BIT bt(N);
44     cin >> n;
45     for (int i = 1; i <= n; ++i) cin >> a[i].vis, a[i].id = i;
46     stable_sort(a + 1, a + 1 + n);
47     for (int i = 1; i <= n; ++i) {
48         b[a[i].id] = i;
49     }
50     for (int i = n; i > 0; --i) { // 反向计算满足  $a_i > a_j$  逆序对数量, 正向计算  $a_i$ 
51         bt.update(b[i], 1);
52         rev[i] = bt.query(b[i] - 1);
53     }
54     int res = 0;
55     for (int i = 1; i <= n; ++i) {
56         res += rev[i];
57     }
58     cout << res; //  $a_i > a_j$ 

```

```

59 }
60 signed main() {
61     ios::sync_with_stdio(false);
62     cin.tie(nullptr);
63     solve();
64     return 0;
65 }

```

线段树

```

1  class Solution {
2  public:
3      static constexpr long long N = 1e5 + 10;
4      long long a[N];
5      long long tree[N<<2], tag[N<<2];
6      long long ls(long long p) {return p<<1;}
7      long long rs(long long p) {return p<<1|1;}
8      void push_up(long long p) {
9          tree[p] = tree[ls(p)] + tree[rs(p)];
10     }
11     void build(long long p, long long pl, long long pr) {
12         tag[p] = 0;
13         if (pl == pr) {
14             tree[p] = a[pl];
15             return;
16         }
17         long long mid = pl + (pr - pl) / 2;
18         build(ls(p), pl, mid);
19         build(rs(p), mid+1, pr);
20         push_up(p);
21     }
22     void add_tag(long long p, long long pl, long long pr, long long d) {
23         tag[p] += d;
24         tree[p] += d * (pr - pl + 1);
25     }
26     void push_down(long long p, long long pl, long long pr) {
27         if (tag[p]) {
28             long long mid = pl + (pr - pl) / 2;
29             add_tag(ls(p), pl, mid, tag[p]);
30             add_tag(rs(p), mid + 1, pr, tag[p]);
31             tag[p] = 0;
32         }
33     }
34     void update(long long l, long long r, long long p, long long pl, long long pr) {
35         if (l <= pl && pr <= r) {

```



```

36         add_tag(p, pl, pr, d);
37         return;
38     }
39     push_down(p, pl, pr);
40     long long mid = pl + (pr - pl) / 2;
41     if (l <= mid) update(l, r, ls(p), pl, mid, d);
42     if (r > mid) update(l, r, rs(p), mid + 1, pr, d);
43     push_up(p);
44 }
45 long long query(long long l, long long r, long long p, long long pl, long long pr) {
46     if (pl >= l && r >= pr) return tree[p];
47     push_down(p, pl, pr);
48     long long res = 0;
49     long long mid = pl + (pr - pl) / 2;
50     if (l <= mid) res += query(l, r, ls(p), pl, mid);
51     if (r > mid) res += query(l, r, rs(p), mid + 1, pr);
52     return res;
53 }
54 bool checkArray(vector<int>& nums, int k) {
55     long long n = nums.size();
56     for (long long i = 1; i <= n; ++i) a[i] = nums[i - 1];
57     build(1, 1, n);
58     for (long long i = 1; i <= n; ++i) {
59         long long val = query(i, i, 1, 1, n);
60         if (val < 0) {
61             return false;
62         }
63         if (val == 0) continue;
64         if (i + k - 1 <= n) {
65             update(i, i + k - 1, 1, 1, n, -val);
66         } else {
67             return false;
68         }
69     }
70     return true;
71 }
72 };

```

平衡树

splay

```

1 #include <iostream>
2 using namespace std;
3 #define ls(x) tr[x].son[0]

```

```

4 #define rs(x) tr[x].son[1]
5 const int MAX_N = 1e5 + 5, MAX_M = 1e6 + 5;
6 const int MAX_SIZE = MAX_N + MAX_M;
7 const int INF = (1 << 30) + 1;
8 struct Node {
9     int son[2]; //左右儿子
10    int parent; //父亲
11    int val; //节点权值
12    int cnt; //权值出现次数
13    int siz; //子树大小
14    void init(int parent_, int val_) {
15        parent = parent_, val = val_;
16        cnt = siz = 1;
17    }
18 }tr[MAX_SIZE];
19 int root, idx; //根节点编号, 节点个数
20 void pushup(int x) { //x 下标
21     tr[x].siz = tr[ls(x)].siz + tr[rs(x)].siz + tr[x].cnt;
22 }
23 void rotate(int x) { //x 下标
24     int y = tr[x].parent, z = tr[y].parent;
25     int k = tr[y].son[1] == x;
26     tr[z].son[tr[z].son[1] == y] = x;
27     tr[x].parent = z;
28     tr[y].son[k] = tr[x].son[k ^ 1];
29     tr[tr[x].son[k ^ 1]].parent = y;
30     tr[x].son[k ^ 1] = y;
31     tr[y].parent = x;
32     pushup(y), pushup(x);
33 }
34 void splay(int x, int k) {
35     while (tr[x].parent != k) {
36         int y = tr[x].parent, z = tr[y].parent;
37         if (z != k) // 折转底, 直转中
38             (ls(y) == x)^(ls(z) == y) ? rotate(x) : rotate(y);
39         rotate(x);
40     }
41     if (!k) root = x;
42 }
43 void insert(int v) { //插入数值v
44     int x = root, p = 0;
45     while (x && tr[x].val != v)
46         p = x, x = tr[x].son[v > tr[x].val];
47     if (x) tr[x].cnt++;
48     else {
49         x = ++idx;
50         tr[p].son[v > tr[p].val] = x;

```

```

51         tr[x].init(p, v);
52     }
53     splay(x, 0);
54 }
55 void find(int v) { //找到数值v并转到根
56     int x = root;
57     while (tr[x].son[v > tr[x].val] && v != tr[x].val)
58         x = tr[x].son[v > tr[x].val];
59     splay(x, 0);
60 }
61 int get_pre(int v) { //数值v前驱的编号
62     find(v);
63     int x = root;
64     if (tr[x].val < v) return x;
65     x = ls(x);
66     while (rs(x)) x = rs(x);
67     splay(x, 0); // 少一句代码都TLE, #5样例TLE #6样例PASS
68     return x;
69 }
70 int get_suc(int v) { //数值v后继的编号
71     find(v);
72     int x = root;
73     if (tr[x].val > v) return x;
74     x = rs(x);
75     while (ls(x)) x = ls(x);
76     splay(x, 0); // #5样例PASS #6样例TLE
77     return x;
78 }
79 void del(int v) { //数值v删除
80     int pre = get_pre(v);
81     int suc = get_suc(v);
82     splay(pre, 0), splay(suc, pre);
83     int del = tr[suc].son[0];
84     if (tr[del].cnt > 1)
85         tr[del].cnt--, splay(del, 0);
86     else
87         tr[suc].son[0] = 0, splay(suc, 0);
88 }
89
90 int get_rank(int v) { //数值v排名
91     // 这里老师的代码有问题
92     // find(v);
93     // return tr[tr[root].son[0]].siz;
94     insert(v);
95     int res = tr[tr[root].son[0]].siz;
96     del(v);
97     return res;

```

```

98 }
99 int get_val_by_rank(int k) { //数值
100     int x = root;
101     while (1) {
102         int y = ls(x);
103         if (tr[y].siz + tr[x].cnt < k)
104             k -= tr[y].siz + tr[x].cnt, x = rs(x);
105         else if (tr[y].siz >= k) x = y;
106         else break;
107     }
108     splay(x, 0);
109     return tr[x].val;
110 }
111 int main() {
112     ios::sync_with_stdio(false);
113     cin.tie(nullptr);
114     insert(-INF);
115     insert(INF); //哨兵
116     int n, m;
117     scanf("%d%d", &n, &m);
118     int x;
119     while (n--) {
120         scanf("%d", &x);
121         insert(x);
122     }
123     int op, res = 0, last = 0;
124     while (m--) {
125         scanf("%d%d", &op, &x);
126         x ^= last;
127         if (op == 1) insert(x);
128         if (op == 2) del(x);
129         if (op == 3) res ^= (last = get_rank(x));
130         if (op == 4) res ^= (last = get_val_by_rank(x + 1));
131         if (op == 5) res ^= (last = tr[get_pre(x)].val);
132         if (op == 6) res ^= (last = tr[get_suc(x)].val);
133     }
134     cout << res << '\n';
135     return 0;
136 }

```

主席树

查询区间第k大

```
1 #include <bits/stdc++.h>
```

```

2 using namespace std ;
3 const int N = 200010;
4 int cnt = 0;
5 int a[N], b[N], root[N];
6 struct{
7     int L, R, sum;
8 }tree[N<<5];
9 int build(int pl, int pr){
10     int rt = ++ cnt;
11     tree[rt].sum = 0;
12     int mid=(pl+pr)>>1;
13     if (pl < pr){
14         tree[rt].L = build(pl, mid);
15         tree[rt].R = build(mid+1, pr);
16     }
17     return rt;
18 }
19 int update(int pre, int pl, int pr, int x){
20     int rt = ++cnt;
21     tree[rt].L = tree[pre].L;
22     tree[rt].R = tree[pre].R;
23     tree[rt].sum = tree[pre].sum + 1;
24     int mid = (pl+pr)>>1;
25     if (pl < pr){
26         if (x <= mid)
27             tree[rt].L = update(tree[pre].L, pl, mid, x);
28         else
29             tree[rt].R = update(tree[pre].R, mid+1, pr, x);
30     }
31     return rt;
32 }
33 int query(int u, int v, int pl, int pr, int k){
34     if (pl == pr) return pl;
35     int x = tree[tree[v].L].sum - tree[tree[u].L].sum;
36     int mid = (pl+pr)>>1;
37     if (x >= k)
38         return query(tree[u].L, tree[v].L, pl, mid, k);
39     else
40         return query(tree[u].R, tree[v].R, mid+1, pr, k-x);
41 }
42 int main(){
43     int n;     scanf("%d", &n);
44     for (int i=1; i<=n; i++){ scanf("%d", &a[i]); b[i]=a[i]; }
45     sort(b+1, b+1+n);
46     int size = unique(b+1, b+1+n)-b-1;
47     for (int i = 1; i <= n; i ++){
48         int x = lower_bound(b+1, b+1+size, a[i]) - b;

```

```

49     root[i] = update(root[i-1], 1, size, x);
50 }
51 int m;    scanf("%d", &m);
52 while (m--){
53     int x, y, k;    scanf("%d%d%d", &x, &y,&k);
54     int t = query(root[x-1], root[y], 1, size, k);
55     printf("%d\n", b[t]);
56 }
57 return 0;
58 }

```

查询区间小于或等于k的数字个数

```

1  #include<iostream>
2  #include<cstring>
3  #include<algorithm>
4  #include<queue>
5  #include<map>
6  #include<stack>
7  #include<cmath>
8  #include<vector>
9  #include<set>
10 #include<cstdio>
11 #include<string>
12 #include<deque>
13 using namespace std;
14 typedef long long LL;
15 #define eps 1e-8
16 #define INF 0x3f3f3f3f
17 #define maxn 100005
18 struct node{
19     int l,r,sum;
20 }tree[maxn*25];
21 int n,m,k,t,cnt;
22 struct point{
23     int id,w;
24 }a[maxn];
25 int b[maxn],rt[maxn];
26 bool operator <(point s1,point s2){
27     if(s1.w!=s2.w)
28         return s1.w<s2.w;
29     else
30         return s1.id<s2.id;
31 }
32 void update(int root){

```

```

33     tree[root].sum=tree[tree[root].l].sum+tree[tree[root].r].sum;
34 }
35 void build_0(int &root,int l,int r){
36     root=++cnt;
37     tree[root].l=l;
38     tree[root].r=r;
39     tree[root].sum=0;
40     if(l==r)
41         return;
42     int mid=(l+r)/2;
43     build_0(tree[root].l,l,mid);
44     build_0(tree[root].r,mid+1,r);
45     update(root);
46 }
47 void build(int pre,int &root,int l,int r,int index){
48     root=++cnt;
49     tree[root]=tree[pre];
50     if(l==r){
51         tree[root].sum++;
52         return;
53     }
54     int mid=(l+r)/2;
55     if(index<=mid)
56         build(tree[pre].l,tree[root].l,l,mid,index);
57     else
58         build(tree[pre].r,tree[root].r,mid+1,r,index);
59     update(root);
60 }
61 int binary(int l,int r,int k){//二分查找区间里面小于等于k的最后一个数字所在的位置
62     while(l<=r){
63         int mid=(l+r)/2;
64         if(a[mid].w>k)
65             r=mid-1;
66         else
67             l=mid+1;
68     }
69     return r;
70 }
71 int ask(int root1,int root2,int L,int R,int l,int r){
72     if(L>R)
73         return 0;
74     if(l>=L&&r<=R){
75         return tree[root2].sum-tree[root1].sum;
76     }
77     int mid=(l+r)/2;
78     int ans=0;
79     if(mid>=L)

```

```

80     ans+=ask(tree[root1].l,tree[root2].l,L,R,l,mid);
81     if(mid<R)
82     ans+=ask(tree[root1].r,tree[root2].r,L,R,mid+1,r);
83     return ans;
84 }
85 int main()
86 {
87     scanf("%d",&t);
88     int Case=0;
89     while(t--){
90         scanf("%d%d",&n,&m);
91         cnt=0;
92         for(int i=1;i<=n;i++){
93             scanf("%d",&a[i].w);
94             a[i].id=i;
95         }
96         //离散化，我的离散化结果是没有重复的编号，这里其实有点多余
97         sort(a+1,a+n+1);
98         for(int i=1;i<=n;i++){
99             b[a[i].id]=i;
100         }
101         build_0(rt[0],1,n);//建第0颗树
102         for(int i=1;i<=n;i++)
103             build(rt[i-1],rt[i],1,n,b[i]);//建第i颗树
104         int l,r,k;
105         printf("Case %d:\n",++Case);
106         while(m--){
107             scanf("%d%d%d",&l,&r,&k);
108             l++;
109             r++;
110             int index=binary(1,n,k);//查找序列里面最后一个小于等于k的数字的位置，对应：
111             printf("%d\n",ask(rt[l-1],rt[r],1,index,1,n));
112         }
113     }
114     return 0;
115 }

```

ST表

一维RMQ问题

```

1 #include <iostream>
2 #include <algorithm>
3 #include <cstring>
4 #include <cmath>

```



```

5 using namespace std;
6 const int N = 1e6 + 5;
7 int n, k, q;
8 int a[N], dp_max[N][30];
9 void st_init() {
10     for (int i = 1; i <= n; ++i) {
11         dp_max[i][0] = a[i];
12     }
13     int p = (int)(log(double(n)) / log(2.0));
14     for (int k = 1; k <= p; ++k) {
15         for (int s = 1; s + (1 << k) <= n + 1; ++s) {
16             dp_max[s][k] = std::max(dp_max[s][k - 1], dp_max[s + (1 << (k -
17                 1))][k - 1]);
18         }
19     }
20 int st_query_max(int L, int R) {
21     int k = (int)(log(double(R - L + 1)) / log(2.0));
22     return std::max(dp_max[L][k], dp_max[R - (1 << k) + 1][k]);
23 }
24 int main() {
25     ios::sync_with_stdio(false);
26     cin.tie(nullptr);
27     cout.tie(nullptr);
28     cin >> n >> q;
29     for (int i = 1; i <= n; ++i) {
30         cin >> a[i];
31     }
32     st_init();
33     while (q--) {
34         int l, r;
35         cin >> l >> r;
36         cout << st_query_max(l, r) << '\n';
37     }
38 }

```

莫队

普通莫队

HH项链

```

1 #include <cstdio>
2 #include <cstring>
3 #include <cmath>

```

```

4 #include <algorithm>
5 using namespace std;
6 #define maxn 1010000
7 #define maxb 1010
8 int aa[maxn], cnt[maxn], belong[maxn];
9 int n, m, size, bnum, now, ans[maxn];
10 struct query {
11     int l, r, id;
12 } q[maxn];
13 int cmp(query a, query b) {
14     return (belong[a.l] ^ belong[b.l]) ? belong[a.l] < belong[b.l] :
        ((belong[a.l] & 1) ? a.r < b.r : a.r > b.r);
15 }
16 #define isdigit(x) ((x) >= '0' && (x) <= '9')
17 int read() {
18     int res = 0;
19     char c = getchar();
20     while(!isdigit(c)) c = getchar();
21     while(isdigit(c)) res = (res << 1) + (res << 3) + c - 48, c = getchar();
22     return res;
23 }
24 void printi(int x) {
25     if(x >= 10) printi(x / 10);
26     putchar(x % 10 + '0');
27 }
28 int main() {
29     scanf("%d", &n);
30     size = sqrt(n);
31     bnum = ceil((double)n / size);
32     for(int i = 1; i <= bnum; ++i)
33         for(int j = (i - 1) * size + 1; j <= i * size; ++j) {
34             belong[j] = i;
35         }
36     for(int i = 1; i <= n; ++i) aa[i] = read();
37     m = read();
38     for(int i = 1; i <= m; ++i) {
39         q[i].l = read(), q[i].r = read();
40         q[i].id = i;
41     }
42     sort(q + 1, q + m + 1, cmp);
43     int l = 1, r = 0;
44     for(int i = 1; i <= m; ++i) {
45         int ql = q[i].l, qr = q[i].r;
46         while(l < ql) now -= !--cnt[aa[l++]];
47         while(l > ql) now += !cnt[aa[--l]]++;
48         while(r < qr) now += !cnt[aa[++r]]++;
49         while(r > qr) now -= !--cnt[aa[r--]];

```

```

50         ans[q[i].id] = now;
51     }
52     for(int i = 1; i <= m; ++i) printi(ans[i]), putchar('\n');
53     return 0;
54 }

```

带修莫队

数颜色

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define maxn 150500
4  #define maxc 3001000
5  int a[maxn], cnt[maxc], ans[maxn], belong[maxn];
6  struct query {
7      int l, r, time, id;
8  } q[maxn];
9  struct modify {
10     int pos, color, last;
11 } c[maxn];
12 int cntq, cntc, n, m, size, bnum;
13 int cmp(query a, query b) {
14     return (belong[a.l] ^ belong[b.l]) ? belong[a.l] < belong[b.l] :
        ((belong[a.r] ^ belong[b.r]) ? belong[a.r] < belong[b.r] : a.time < b.time);
15 }
16 #define isdigit(x) ((x) >= '0' && (x) <= '9')
17 inline int read() {
18     int res = 0;
19     char c = getchar();
20     while(!isdigit(c)) c = getchar();
21     while(isdigit(c)) res = (res << 1) + (res << 3) + (c ^ 48), c = getchar();
22     return res;
23 }
24 int main() {
25     n = read(), m = read();
26     size = pow(n, 2.0 / 3.0);
27     bnum = ceil((double)n / size);
28     for(int i = 1; i <= bnum; ++i)
29         for(int j = (i - 1) * size + 1; j <= i * size; ++j) belong[j] = i;
30     for(int i = 1; i <= n; ++i)
31         a[i] = read();
32     for(int i = 1; i <= m; ++i) {
33         char opt[100];
34         scanf("%s", opt);

```

```

35     if(opt[0] == 'Q') {
36         q[++cntq].l = read();
37         q[cntq].r = read();
38         q[cntq].time = cntc;
39         q[cntq].id = cntq;
40     }
41     else if(opt[0] == 'R') {
42         c[++cntc].pos = read();
43         c[cntc].color = read();
44     }
45 }
46 sort(q + 1, q + cntq + 1, cmp);
47 int l = 1, r = 0, time = 0, now = 0;
48 for(int i = 1; i <= cntq; ++i) {
49     int ql = q[i].l, qr = q[i].r, qt = q[i].time;
50     while(l < ql) now -= !--cnt[a[l++]];
51     while(l > ql) now += !cnt[a[--l]]++;
52     while(r < qr) now += !cnt[a[++r]]++;
53     while(r > qr) now -= !--cnt[a[r--]];
54     while(time < qt) {
55         ++time;
56         if(ql <= c[time].pos && c[time].pos <= qr) now -= !--
cnt[a[c[time].pos]] - !cnt[c[time].color]++;
57         swap(a[c[time].pos], c[time].color);
58     }
59     while(time > qt) {
60         if(ql <= c[time].pos && c[time].pos <= qr) now -= !--
cnt[a[c[time].pos]] - !cnt[c[time].color]++;
61         swap(a[c[time].pos], c[time].color);
62         --time;
63     }
64     ans[q[i].id] = now;
65 }
66 for(int i = 1; i <= cntq; ++i)
67     printf("%d\n", ans[i]);
68 return 0;
69 }

```

树上莫队

回滚莫队

四、图论

多源BFS

```
1 static constexpr int dirs[4][2] = {{-1,0}, {1, 0}, {0, -1}, {0, 1}};
2 int maximumSafenessFactor(vector<vector<int>>& grid) {
3     int n = grid.size();
4     vector<pair<int,int>> q;
5     vector<vector<int>> dis(n, vector<int>(n, -1));
6     for (int i = 0; i < n; ++i) {
7         for (int j = 0; j < n; ++j) {
8             if (grid[i][j]) q.emplace_back(i,j), dis[i][j] = 0;
9         }
10    }
11    vector<vector<pair<int,int>>> groups = {q};
12    while (!q.empty()) {
13        vector<pair<int,int>> nq;
14        for (auto& [i,j]: q) {
15            for (auto& d: dirs) {
16                int x = i + d[0], y = j + d[1];
17                if (x >= 0 && x < n && y >= 0 && y < n && dis[x][y] < 0) {
18                    nq.emplace_back(x, y);
19                    dis[x][y] = groups.size();
20                }
21            }
22        }
23        groups.push_back(nq);
24        q = move(nq);
25    }
26 }
```

拓扑排序

```
1 #include <iostream>
2 #include <cstring>
3 #include <algorithm>
4 #include <queue>
5 using namespace std;
```

```

6  const int N = 100010;
7  int n,m,a,b;
8  vector<int> e[N], tp;
9  int din[N];
10
11 bool toposort(){
12     queue<int> q;
13     for(int i = 1; i <= n; i++)
14         if(din[i]==0) q.push(i);
15     while(q.size()){
16         int x=q.front(); q.pop();
17         tp.push_back(x);
18         for(auto y : e[x]){
19             if(--din[y]==0) q.push(y);
20         }
21     }
22     return tp.size() == n;
23 }
24 int main(){
25     cin >> n >> m;
26     for(int i=0; i<m; i++){
27         cin >> a >> b;
28         e[a].push_back(b);
29         din[b]++;
30     }
31     if(!toposort()) puts("-1");
32     else for(auto x:tp) printf("%d ",x);
33     return 0;
34 }

```

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  using i64 = long long;
4  void solve() {
5      int n, m;
6      cin >> n >> m;
7      vector<vector<int>> adj(n);
8      vector<int> deg(n);
9      for (int i = 0; i < m; ++i) {
10         int u, v;
11         cin >> u >> v;
12         u--, v--;
13         adj[u].push_back(v);
14         deg[v]++;
15     }

```

```

16     vector<int> q;
17     for (int i = 0; i < n; ++i) {
18         if (!deg[i]) {
19             q.push_back(i);
20         }
21     }
22     for (int i = 0; i < q.size(); ++i) {
23         int x = q[i];
24         for (auto y: adj[x]) {
25             if (!--deg[y]) {
26                 q.push_back(y);
27             }
28         }
29     }
30     if (q.size() == n) {
31         cout << 1 << '\n';
32         for (int i = 0; i < n; ++i) {
33             cout << q[i] + 1 << " \n"[i == n - 1];
34         }
35         return;
36     }
37     cout << 2 << '\n';
38     for (int i = 1; i <= n; ++i) {
39         cout << i << " \n"[i == n];
40     }
41     for (int i = 1; i <= n; ++i) {
42         cout << n + 1 - i << " \n"[i == n];
43     }
44 }
45 int main() {
46     ios::sync_with_stdio(false);
47     cin.tie(nullptr);
48     solve();
49 }

```

```

1  #include <iostream>
2  #include <cstring>
3  #include <algorithm>
4  #include <queue>
5  using namespace std;
6
7  const int N = 100010;
8  int n,m,a,b;
9  vector<int> e[N], tp;
10 int c[N]; //染色数组bool dfs(int x){

```

```

11  c[x] = -1;
12  for(int y : e[x]){
13      if(c[y]<0)return 0; //有环 else if(!c[y])
14          if(!dfs(y))return 0;
15  }
16  c[x] = 1;
17  tp.push_back(x);
18  return 1;
19 }
20 bool toposort(){
21     memset(c, 0, sizeof(c));
22     for(int x = 1; x <= n; x++){
23         if(!c[x])
24             if(!dfs(x))return 0;
25     reverse(tp.begin(),tp.end());
26     return 1;
27 }
28 int main(){
29     cin >> n >> m;
30     for(int i=0; i<m; i++){
31         cin >> a >> b;
32         e[a].push_back(b);
33     }
34     if(!toposort()) puts("-1");
35     else
36         for(int x:tp) printf("%d ",x);
37     return 0;
38 }

```

最短路

Dijkstra算法

```

1  #include <cstdio>
2  #include <vector>
3  #include <cstring>
4  #include <climits>
5  using namespace std;
6  const int N = 1e5 + 5;
7  #define int long long
8  struct edge {
9      int v, w;
10     edge(): v(0), w(0) {}
11     edge(int v, int w): v(v), w(w) {}
12 };

```



```

13 int n, m, s;
14 vector<edge> e[N];
15 int dis[N];
16 int vis[N];
17 void dijkstra() {
18     for (int i = 0; i <= n; ++i) dis[i] = INT_MAX;
19     dis[s] = 0;
20     for (int i = 1; i < n; ++i) {
21         int u = 0;
22         for (int j = 1; j <= n; ++j) {
23             if (!vis[j] && dis[j] < dis[u]) {
24                 u = j;
25             }
26         }
27         vis[u] = 1;
28         for (auto& ed: e[u]) {
29             int v = ed.v;
30             int w = ed.w;
31             if (dis[v] > dis[u] + w) {
32                 dis[v] = dis[u] + w;
33             }
34         }
35     }
36 }
37 signed main() {
38     scanf("%lld%lld%lld",&n,&m,&s);
39     for (int i = 0; i < m; ++i) {
40         int u, v, w;
41         scanf("%lld%lld%lld",&u,&v,&w);
42         e[u].push_back({v, w});
43     }
44     dijkstra();
45     for (int i = 1; i <= n; ++i) {
46         printf("%lld ", dis[i]);
47     }
48     return 0;
49 }

```

```

1 #include <cstdio>
2 #include <cstring>
3 #include <vector>
4 #include <queue>
5 #include <climits>
6 using namespace std;
7 struct edge {

```

```

8     int v, w;
9     edge(): v(0), w(0) {}
10    edge(int v, int w): v(v), w(w) {}
11 };
12 struct node {
13     int dis, u;
14     bool operator>(const node& n) const { return dis > n.dis; }
15 };
16 const int N = 1e5 + 5;
17 vector<edge> e[N];
18 int dis[N];
19 int vis[N];
20 priority_queue<node, vector<node>, greater<node>> q;
21 int n, m, s;
22 void dijkstra() {
23     for (int i = 0; i <= n; ++i) dis[i] = INT_MAX;
24     dis[s] = 0;
25     q.push({0,s});
26     while (!q.empty()) {
27         int u = q.top().u;
28         q.pop();
29         if (vis[u]) continue;
30         vis[u] = 1;
31         for (auto ed: e[u]) {
32             int v = ed.v;
33             int w = ed.w;
34             if (dis[v] > dis[u] + w) {
35                 dis[v] = dis[u] + w;
36                 q.push({dis[v], v});
37             }
38         }
39     }
40 }
41 int main() {
42     scanf("%d%d%d",&n,&m,&s);
43     for (int i = 0; i < m; ++i) {
44         int u, v, w;
45         scanf("%d%d%d",&u,&v,&w);
46         e[u].push_back({v, w});
47     }
48     dijkstra();
49     for (int i = 1; i <= n; ++i) {
50         printf("%d ", dis[i]);
51     }
52     return 0;
53 }

```

负环

Floyd

```
1 //Ford 判负环 740ms
2 #include <cstring>
3 #include <iostream>
4 #include <algorithm>
5 using namespace std;
6 const int inf=0x3f3f3f3f;
7 const int N=2010,M=6010;
8 int n,m;
9 int to[M],ne[M],w[M],h[N],tot;
10 int d[N];
11 void add(int a,int b,int c){
12     to[++tot]=b;w[tot]=c;
13     ne[tot]=h[a];h[a]=tot;
14 }
15 bool ford(){
16     memset(d,inf,sizeof d); d[1]=0;
17     bool flag; //是否松弛
18     for(int i=1;i<=n;i++){ //跑n轮
19         flag=false;
20         for(int u=1;u<=n;u++){ //n个点
21             if(d[u]==inf)continue;
22             for(int j=h[u];j;j=ne[j]){
23                 int v=to[j];
24                 if(d[v]>d[u]+w[j]){
25                     d[v]=d[u]+w[j];
26                     flag=true;
27                 }
28             }
29         }
30         if(!flag)break;
31     }
32     return flag; //第n轮=true,有负环
33 }
34 int main(){
35     int T;
36     scanf("%d",&T);
```

```

37 while(T--){
38     tot=0; memset(h,0,sizeof(h));
39     scanf("%d%d",&n,&m);
40     for(int i=1;i<=m;i++){
41         int u,v,w;
42         scanf("%d%d%d",&u,&v,&w);
43         add(u,v,w);
44         if(w>=0)add(v,u,w);;
45     }
46     puts(ford()?"YES":"NO");
47 }
48 return 0;
49 }

```

```

1 //BFS_spfa 判负环 530ms
2 #include <iostream>
3 #include <cstring>
4 #include <algorithm>
5 #include <queue>
6 using namespace std;
7 const int inf=0x3f3f3f3f;
8 const int N=2010,M=6010;
9 int n,m;
10 int to[M],ne[M],w[M],h[N],tot;
11 int d[N],cnt[N],vis[N];
12 void add(int a,int b,int c){
13     to[++tot]=b;w[tot]=c;
14     ne[tot]=h[a];h[a]=tot;
15 }
16 bool spfa(){ //判负环
17     memset(d,0x3f,sizeof d);
18     memset(vis,0,sizeof vis);
19     memset(cnt,0,sizeof cnt);
20     queue<int>q;
21     q.push(1); vis[1]=1; d[1]=0;
22     while(q.size()){
23         int u=q.front();q.pop();vis[u]=0;
24         for(int i=h[u];i;i=ne[i]){
25             int v=to[i];
26             if(d[v]>d[u]+w[i]){
27                 d[v]=d[u]+w[i];
28                 cnt[v]=cnt[u]+1;
29                 if(cnt[v]>=n)return 1;//判边数
30                 if(!vis[v])q.push(v),vis[v]=1;
31             }

```

```

32     }
33 }
34 return 0;
35 }
36 int main(){
37     int T; scanf("%d",&T);
38     while(T--){
39         tot=0; memset(h,0,sizeof(h));
40         scanf("%d%d",&n,&m);
41         for(int i=1;i<=m;i++){
42             int u,v,w;
43             scanf("%d%d%d",&u,&v,&w);
44             add(u,v,w);
45             if(w>=0)add(v,u,w);;
46         }
47         puts(spfa()?"YES":"NO");
48     }
49     return 0;
50 }

```

```

1 //BFS_spfa 判负环 690ms
2 #include <iostream>
3 #include <cstring>
4 #include <algorithm>
5 #include <queue>
6 using namespace std;
7 const int inf=0x3f3f3f3f;
8 const int N=2010,M=6010;
9 int n,m;
10 int to[M],ne[M],w[M],h[N],tot;
11 int d[N],cnt[N],vis[N];
12 void add(int a,int b,int c){
13     to[++tot]=b;w[tot]=c;
14     ne[tot]=h[a];h[a]=tot;
15 }
16 bool spfa(){ //判负环
17     memset(d,0x3f,sizeof d);
18     memset(vis,0,sizeof vis);
19     memset(cnt,0,sizeof cnt);
20     queue<int>q;
21     q.push(1); vis[1]=1; d[1]=0;
22     while(q.size()){
23         int u=q.front();q.pop();vis[u]=0;
24         for(int i=h[u];i;i=ne[i]){
25             int v=to[i];

```

```

26     if(d[v]>d[u]+w[i]){
27         d[v]=d[u]+w[i];
28         if(++cnt[v]>n) return 1; //判点数
29         if(!vis[v])q.push(v),vis[v]=1;
30     }
31 }
32 }
33 return 0;
34 }
35 int main(){
36     int T; scanf("%d",&T);
37     while(T--){
38         tot=0; memset(h,0,sizeof(h));
39         scanf("%d%d",&n,&m);
40         for(int i=1;i<=m;i++){
41             int u,v,w;
42             scanf("%d%d%d",&u,&v,&w);
43             add(u,v,w);
44             if(w>=0)add(v,u,w);
45         }
46         puts(spfa()?"YES":"NO");
47     }
48     return 0;
49 }

```

```

1 //DFS_spfa 判负环 会卡点 #9
2 #include <iostream>
3 #include <cstring>
4 #include <algorithm>
5 using namespace std;
6 const int inf=0x3f3f3f3f;
7 const int N=2010,M=6010;
8 int n,m;
9 int to[M],ne[M],w[M],h[N],tot;
10 int d[N],vis[N];
11 void add(int a,int b,int c){
12     to[++tot]=b;w[tot]=c;
13     ne[tot]=h[a];h[a]=tot;
14 }
15 bool spfa(int u){ //判负环
16     vis[u]=1;
17     for(int i=h[u];i;i=ne[i]){
18         int v=to[i];
19         if(d[v]>d[u]+w[i]){
20             d[v]=d[u]+w[i];

```

```

21     if(vis[v]||spfa(v))return 1;
22 }
23 }
24 vis[u]=0;
25 return 0;
26 }
27 int main(){
28     int T; scanf("%d",&T);
29     while(T--){
30         tot=0; memset(h,0,sizeof(h));
31         scanf("%d%d",&n,&m);
32         for(int i=1;i<=m;i++){
33             int u,v,w;
34             scanf("%d%d%d",&u,&v,&w);
35             add(u,v,w);
36             if(w>=0)add(v,u,w);;
37         }
38         memset(d,0x3f,sizeof d);d[1]=0;
39         memset(vis,0,sizeof vis);
40         puts(spfa(1)?"YES":"NO");
41     }
42     return 0;
43 }

```

最小环

无向图

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 105;
4  int n, m, a, b, c;
5  typedef long long ll;
6  const ll inf = 1e13;
7  ll res = inf;
8  ll w[N][N];
9  ll d[N][N];
10 int main() {
11     cin >> n >> m;
12     for (int i = 1; i <= n; ++i) {
13         for (int j = 1; j <= n; ++j) {
14             if (i != j) w[i][j] = inf;
15         }
16     }
17     for (int i = 1; i <= m; ++i) {

```

```

18     cin >> a >> b >> c;
19     w[a][b] = w[b][a] = c;
20 }
21 memcpy(d, w, sizeof(d));
22 for (int k = 1; k <= n; ++k) {
23     for (int i = 1; i < k; ++i) {
24         for (int j = i + 1; j < k; ++j) {
25             res = min(res, d[i][j] + w[j][k] + w[k][i]);
26         }
27     }
28     for (int i = 1; i <= n; ++i) {
29         for (int j = 1; j <= n; ++j) {
30             d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
31         }
32     }
33 }
34 if (res == inf) cout << "No solution.";
35 else cout << res;
36 return 0;
37 }

```

最长路

k短路

```

1  #include <cstdio>
2  #include <iostream>
3  #include <cstring>
4  #include <vector>
5  #include <queue>
6  using namespace std;
7  const int N=1010,M=200010;
8  int h[N],rh[N],to[M],w[M],ne[M],tot;
9  void add(int h[],int a,int b,int c){
10     to[++tot]=b;w[tot]=c;
11     ne[tot]=h[a],h[a]=tot;
12 }
13 int n,m,S,T,K;
14 int f[N],vis[N],cnt[N];
15 struct node{
16     int s,v,d; //s排序, v点, d距离
17     bool operator<(const node &x)const
18     {return s>x.s;}

```



```

19 };
20 void dijkstra(){
21     memset(f,0x3f,sizeof f); f[T]=0;
22     priority_queue<pair<int,int>> q;
23     q.push(make_pair(0,T));
24     while(q.size()){
25         pair<int,int> t=q.top(); q.pop();
26         int u=t.second;
27         if(vis[u])continue;
28         vis[u]=true; //第一次出队时最小
29         for(int i=rh[u]; i; i=ne[i]){
30             int v=to[i];
31             if(f[v]>f[u]+w[i]){
32                 f[v]=f[u]+w[i]; //估价函数
33                 q.push(make_pair(-f[v],v));
34             }
35         }
36     }
37 }
38 int aStar(){
39     priority_queue<node> q; //优先队列
40     node a={f[S],S,0}; q.push(a);
41     while(q.size()){
42         node t=q.top(); q.pop();
43         int u=t.v;
44         cnt[u]++; //记录出队次数
45         if(cnt[T]==K) return t.d; //边界
46         for(int i=h[u]; i; i=ne[i]){
47             int v=to[i], d=t.d+w[i];
48             if(cnt[v]<K){
49                 node a={d+f[v],v,d};
50                 q.push(a);
51             }
52         }
53     }
54     return -1;
55 }
56 int main(){
57     scanf("%d%d",&n,&m);
58     for(int i=1; i<=m; i++){
59         int a,b,c;
60         scanf("%d%d%d",&a,&b,&c);
61         add(h,a,b,c); add(rh,b,a,c); //反图
62     }
63     scanf("%d%d%d",&S,&T,&K);
64     if(S==T) K++; //重合点, 0是第一条
65     dijkstra();

```

```

66     printf("%d\n", aStar());
67 }

```

欧拉回路

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int N = 55;
4  int degree[N]; //记录度
5  int G[N][N];   //存图
6  void euler(int u){ //从u开始DFS
7      for(int v = 1; v <= 50; v++) { //v是u的邻居
8          if(G[u][v]) {
9              G[u][v]--;
10             G[v][u]--;
11             euler(v);
12             cout << v << " " << u << endl; //在euler()后打印, 即回溯时打印
13         }
14     }
15 }
16 int main(){
17     int t; cin >> t;
18     int cnt = 0;
19     while (t--){
20         cnt++;
21         if(cnt != 1) cout << endl;
22         cout << "Case #" << cnt << endl;
23         memset(degree, 0, sizeof(degree));
24         memset(G, 0, sizeof(G));
25         int n; cin >> n;
26         int color;
27         for(int i = 0; i < n; i++){ //输入n条边
28             int u, v; cin >> u >> v;
29             color = u; //记录一种颜色。测试的时候可能只出现某些颜色
30             degree[u]++;
31             degree[v]++; //记录点的度
32             G[u][v]++;
33             G[v][u]++; //存图: =0不连接, =1连接, >1有重边
34         }
35         int ok = 1;
36         for(int i = 1; i <= 50; i++){
37             if(degree[i] % 2){ //存在奇点, 无欧拉路
38                 cout << "some beads may be lost" << endl;
39                 ok = 0;
40                 break;

```

```

41     }
42     if(ok) euler(color);           //有欧拉路。随便从某个存在的颜色开始
43 }
44 return 0;
45 }

```

```

1  #include <stdio.h>
2  const int N = 1e5;
3  int num[N];                       //num[v]: 点v后加的数字, num[v]=0~9
4  int st_edge[10*N], top_s;         //栈, 用于存边。top_s指示栈顶
5  char st_ans [10*N]; int top_a;    //栈, 存序列结果。top_a指示栈顶
6  int m;
7  void no_dfs(int v){               //模拟递归, 递归搜点v的10条边, 放进st_edge中
8      int edge;                     //边的值
9      while(num[v]<10){             //在点v(是一个n-1位序列)后加0~9构成10条边
10         edge=10*v + num[v];       //数字edge代表一个边
11         num[v]++;                  //点v添的下一个数字。按字典序递增
12         st_edge[top_s++] = edge;   //把边存入到栈st_edge中, 它是字典序的
13         //printf("%02d -> ",v);   //打印边的起点
14         v = edge%10;               //更新起点为原来的终点, 往下走。点值等于edge的后几
    位
15         //printf("%02d: edge=%03d\n",v,edge); //打印边的终点、边的权值
16     }
17 }
18 int main(){
19     int n, edge;
20     while(scanf("%d",&n)&&n!=0){
21         top_s = top_a = edge = 0;
22         m = 1;
23         for(int i=0;i<n-1;++i) m*=10; //m是点的数量, 共10^(n-1)个点
24         for(int i=0;i<m; i++) num[i]=0;
25         no_dfs(0);                  //从起点0开始, 递归点0的10条边
26         while(top_s){               //继续走
27             edge = st_edge[--top_s];
28             st_ans[top_a++] = edge%10+'0'; //只需要存边值的最后一位
29             no_dfs(edge/10);         //边值的前n-1位, 即上一个点, 作用类似DFS
    的回溯
30         }
31         for(int i=1;i<n;++i) printf("0"); //打印第一组数, 就是n个0
32         while(top_a) printf("%c",st_ans[--top_a]); //打印其他组数, 每组打印1位
33         printf("\n");
34     }
35     return 0;
36 }

```

最小生成树

Kruskal 适合稀疏图

```
1 #include <cstdio>
2 #include <cstring>
3 #include <algorithm>
4 using namespace std;
5 const int N = 5005;
6 const int M = 2e5 + 5;
7 struct Edge {
8     int u, v, w;
9 }e[M];
10 int s[N];
11 int find(int x) {
12     if (x != s[x]) s[x] = find(s[x]);
13     return s[x];
14 }
15 int n, m;
16 void kruskal() {
17     sort(e + 1, e + m + 1, [](const Edge& e1, const Edge& e2) { return e1.w < e2.w; });
18     for (int i = 1; i <= n; ++i) {
19         s[i] = i;
20     }
21     int ans = 0, cnt = 0;
22     for (int i = 1; i <= m; ++i) {
23         if (cnt == n - 1) break;
24         int e1 = find(e[i].u);
25         int e2 = find(e[i].v);
26         if (e1 == e2) continue;
27         else {
28             ans += e[i].w;
29             s[e1] = e2;
30             ++cnt;
31         }
32     }
33     if (cnt == n - 1) printf("%d\n", ans);
34     else printf("orz\n");
35 }
36 int main() {
37     scanf("%d%d", &n, &m);
38     for (int i = 1; i <= m; ++i) {
39         scanf("%d%d%d", &e[i].u, &e[i].v, &e[i].w);
40     }
41     kruskal();
42 }
```

最近公共祖先LCA

第一行包含三个正整数 N, M, S ，分别表示树的结点个数、询问的个数和树根结点的序号。

接下来 $N-1$ 行每行包含两个正整数 x, y ，表示 x 结点和 y 结点之间有一条直接连接的边（数据保证可以构成树）。

接下来 M 行每行包含两个正整数 a, b ，表示询问 a 结点和 b 结点的最近公共祖先。

在线倍增算法

```
1 // 2.0s
2 #include <iostream>
3 #include <cstring>
4 #include <algorithm>
5 using namespace std;
6 const int N=5e5+10,M=2*N;
7 int n,m,s,a,b;
8 int dep[N],fa[N][22];
9 int h[N],to[M],ne[M],tot;
10 void add(int a, int b){
11     to[++tot]=b,ne[tot]=h[a],h[a]=tot;
12 }
13 void dfs(int x, int f){
14     dep[x]=dep[f]+1; fa[x][0]=f;
15     for(int i=0; i<=20; i++)
16         fa[x][i+1]=fa[fa[x][i]][i];
17     for(int i=h[x]; i; i=ne[i])
18         if(to[i]!=f) dfs(to[i], x);
19 }
20 int lca(int x, int y){
21     if(dep[x]<dep[y]) swap(x, y);
22     for(int i=20; ~i; i--)
23         if(dep[fa[x][i]]>=dep[y]) x=fa[x][i];
24     if(x==y) return y;
25
26     for(int i=20; ~i; i--)
27         if(fa[x][i]!=fa[y][i]) x=fa[x][i],y=fa[y][i];
28     return fa[x][0];
29 }
30 int main(){
31     scanf("%d%d%d", &n,&m,&s);
32     for(int i=1; i<n; i++){
33         scanf("%d%d",&a,&b);
34         add(a,b); add(b,a);
35     }
```

```

36     dfs(s, 0);
37     while(m--){
38         scanf("%d%d", &a, &b);
39         printf("%d\n", lca(a, b));
40     }
41     return 0;
42 }

```

离线Tarjan算法

```

1  #include <iostream>
2  #include <algorithm>
3  #include <cstring>
4  #include <vector>
5  using namespace std;
6  const int N=500005,M=2*N;
7  int n,m,s,a,b;
8  vector<int> e[N];
9  vector<pair<int,int>> query[N];
10 int fa[N],vis[N],ans[M];
11 int find(int x){
12     if(x==fa[x]) return x;
13     return fa[x]=find(fa[x]);
14 }
15 void tarjan(int x){
16     vis[x]=true; //标记x已访问
17     for(auto y : e[x]){
18         if(!vis[y]){
19             tarjan(y);
20             fa[y]=x; //回到x时指向x
21         }
22     }
23     //离开x时找LCA
24     for(auto q : query[x]){
25         int y=q.first,i=q.second;
26         if(vis[y]) ans[i]=find(y);
27     }
28 }
29 int main(){
30     scanf("%d%d%d", &n,&m,&s);
31     for(int i=1; i<n; i++){
32         scanf("%d%d",&a,&b);
33         e[a].push_back(b);
34         e[b].push_back(a);
35     }

```

```

36     for(int i=1;i<=m;i++){
37         scanf("%d%d",&a,&b);
38         query[a].push_back({b,i});
39         query[b].push_back({a,i});
40     }
41     for(int i=1;i<=N;i++)fa[i]=i;
42     tarjan(s);
43     for(int i=1; i<=m; i++)
44         printf("%d\n",ans[i]);
45     return 0;
46 }

```

树链剖分

```

1  // 1.6s
2  #include <iostream>
3  #include <cstring>
4  #include <algorithm>
5  #include <vector>
6  using namespace std;
7  const int N=500010;
8  int n,m,s,a,b;
9  vector<int> e[N];
10 int fa[N],son[N],dep[N],siz[N];
11 int top[N];
12 void dfs1(int u,int f){ //搞fa,dep,son
13     fa[u]=f;siz[u]=1;dep[u]=dep[f]+1;
14     for(int v:e[u]){
15         if(v==f) continue;
16         dfs1(v,u);
17         siz[u]+=siz[v];
18         if(siz[son[u]]<siz[v])son[u]=v;
19     }
20 }
21 void dfs2(int u,int t){ //搞top
22     top[u]=t; //记录链头
23     if(!son[u]) return; //无重儿子
24     dfs2(son[u],t); //搜重儿子
25     for(int v:e[u]){
26         if(v==fa[u]||v==son[u])continue;
27         dfs2(v,v); //搜轻儿子
28     }
29 }
30 int lca(int u,int v){
31     while(top[u]!=top[v]){

```

```

32     if(dep[top[u]]<dep[top[v]])swap(u,v);
33     u=fa[top[u]];
34 }
35 return dep[u]<dep[v]?u:v;
36 }
37 int main(){
38     scanf("%d%d%d",&n,&m,&s);
39     for(int i=1; i<n; i++){
40         scanf("%d%d",&a,&b);
41         e[a].push_back(b);
42         e[b].push_back(a);
43     }
44     dfs1(s,0);
45     dfs2(s,s);
46     while(m--){
47         scanf("%d%d",&a,&b);
48         printf("%d\n",lca(a,b));
49     }
50     return 0;
51 }

```

树上分治

静态点分治

动态点分治

树链剖分

树上启发式合并

```

1  #include<bits/stdc++.h>
2  #define LL long long
3  using namespace std;
4  const int MAXN = 1e5 + 10;
5  inline int read() {
6      char c = getchar(); int x = 0, f = 1;
7      while(c < '0' || c > '9') {if(c == '-') f = -1; c = getchar();}
8      while(c >= '0' && c <= '9') x = x * 10 + c - '0', c = getchar();
9      return x * f;
10 }
11 int N, col[MAXN], son[MAXN], siz[MAXN], cnt[MAXN], Mx, Son;
12 LL sum = 0, ans[MAXN];
13 vector<int> v[MAXN];

```



```

14 void dfs(int x, int fa) {
15     siz[x] = 1;
16     for(int i = 0; i < v[x].size(); i++) {
17         int to = v[x][i];
18         if(to == fa) continue;
19         dfs(to, x);
20         siz[x] += siz[to];
21         if(siz[to] > siz[son[x]]) son[x] = to; //轻重链剖分
22     }
23 }
24 void add(int x, int fa, int val) {
25     cnt[col[x]] += val; //这里可能会因题目而异
26     if(cnt[col[x]] > Mx) Mx = cnt[col[x]], sum = col[x];
27     else if(cnt[col[x]] == Mx) sum += (LL)col[x];
28     for(int i = 0; i < v[x].size(); i++) {
29         int to = v[x][i];
30         if(to == fa || to == Son) continue;
31         add(to, x, val);
32     }
33 }
34 void dfs2(int x, int fa, int opt) {
35     for(int i = 0; i < v[x].size(); i++) {
36         int to = v[x][i];
37         if(to == fa) continue;
38         if(to != son[x]) dfs2(to, x, 0); //暴力统计轻边的贡献, opt = 0表示递归完成后
消除对该点的影响
39     }
40     if(son[x]) dfs2(son[x], x, 1), Son = son[x]; //统计重儿子的贡献, 不消除影响
41
42     add(x, fa, 1); Son = 0; //暴力统计所有轻儿子的贡献
43     ans[x] = sum; //更新答案
44     if(!opt) add(x, fa, -1), sum = 0, Mx = 0; //如果需要删除贡献的话就删掉
45 }
46 int main() {
47     N = read();
48     for(int i = 1; i <= N; i++) col[i] = read();
49     for(int i = 1; i <= N - 1; i++) {
50         int x = read(), y = read();
51         v[x].push_back(y); v[y].push_back(x);
52     }
53     dfs(1, 0);
54     dfs2(1, 0, 0);
55     for(int i = 1; i <= N; i++) printf("%I64d ", ans[i]);
56     return 0;
57 }

```

有向图的连通性

Kosaraju算法

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 const int N = 10005;
4 vector<int> G[N], rG[N];
5 vector<int> S; //存第一次dfs1的结果：标记点的先后顺序
6 int vis[N], sccno[N], cnt; // cnt: 强连通分量的个数
7 void dfs1(int u) {
8     if(vis[u]) return;
9     vis[u] = 1;
10    for(int i=0; i<G[u].size(); i++) dfs1(G[u][i]);
11    S.push_back(u); //记录点的先后顺序，标记大的放在S的后面
12 }
13 void dfs2(int u) {
14     if(sccno[u]) return;
15     sccno[u] = cnt;
16     for(int i=0; i < rG[u].size(); i++) dfs2(rG[u][i]);
17 }
18 void Kosaraju(int n) {
19     cnt = 0;
20     S.clear();
21     memset(sccno, 0, sizeof(sccno));
22     memset(vis, 0, sizeof(vis));
23     for(int i = 1; i <= n; i++) dfs1(i); //点的编号: 1~n。递归所有点
24     for(int i = n-1; i >= 0; i--)
25         if(!sccno[S[i]]) { cnt++; dfs2(S[i]); }
26 }
27 int main(){
28     int n, m, u, v;
29     while(scanf("%d%d", &n, &m), n != 0 || m != 0) {
30         for(int i = 0; i < n; i++) { G[i].clear(); rG[i].clear(); }
31         for(int i = 0; i < m; i++){
32             scanf("%d%d", &u, &v);
33             G[u].push_back(v); //原图
34             rG[v].push_back(u); //反图
35         }
36         Kosaraju(n);
37         printf("%s\n", cnt == 1 ? "Yes" : "No");
38     }
39     return 0;
40 }
```

Tarjan算法

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 const int N = 10005;
4 int cnt; // 强连通分量的个数
5 int low[N], num[N], dfn;
6 int sccno[N], stack[N], top; // 用stack[]处理栈, top是栈顶
7 vector<int> G[N];
8 void dfs(int u){
9     stack[top++] = u; //u进栈
10    low[u]= num[u]= ++dfn;
11    for(int i=0; i<G[u].size(); ++i){
12        int v = G[u][i];
13        if(!num[v]){ //未访问过的点, 继续dfs
14            dfs(v); //dfs的最底层, 是最后一个SCC
15            low[u]= min( low[v], low[u] );
16        }
17        else if(!sccno[v]) //处理回退边
18            low[u]= min( low[u], num[v] );
19    }
20    if(low[u] == num[u]){ //栈底的点是SCC的祖先, 它的low = num
21        cnt++;
22        while(1){
23            int v = stack[--top]; //v弹出栈
24            sccno[v]= cnt;
25            if(u==v) break; //栈底的点是SCC的祖先
26        }
27    }
28 }
29 void Tarjan(int n){
30     cnt = top = dfn = 0;
31     memset(sccno,0,sizeof(sccno));
32     memset(num,0,sizeof(num));
33     memset(low,0,sizeof(low));
34     for(int i=1; i<=n; i++){
35         if(!num[i])
36             dfs(i);
37     }
38 }
39 int main(){
40     int n,m,u,v;
41     while(scanf("%d%d", &n, &m), n != 0 || m != 0) {
42         for(int i=1; i<=n; i++){ G[i].clear();}
43         for(int i=0; i<m; i++){
44             scanf("%d%d", &u, &v);
45             G[u].push_back(v);
```

```

45     }
46     Tarjan(n);
47     printf("%s\n", cnt == 1 ? "Yes" : "No" );
48 }
49 return 0;
50 }

```

无向图的连通性

二分图

判定

```

1  #include <iostream>
2  #include <cstring>
3  #include <algorithm>
4  using namespace std;
5
6  const int N=100010,M=2*N;
7  int n,m;
8  struct edge{int v,ne;}e[M];
9  int h[N],idx;
10 int color[N];
11
12 void add(int a,int b){
13     e[++idx]={b,h[a]};
14     h[a]=idx;
15 }
16 bool dfs(int u,int c){
17     color[u]=c;
18     for(int i=h[u];i;i=e[i].ne){
19         int v=e[i].v;
20         if(!color[v]){
21             if(dfs(v,3-c))return 1;
22         }
23         else if(color[v]==c)return 1;//有奇环
24     }
25     return 0;
26 }
27 int main(){
28     cin>>n>>m;
29     for(int i=0;i<m;i++){
30         int a,b;
31         cin>>a>>b;

```

```

32     add(a,b);
33     add(b,a);
34 }
35 bool flag=0;
36 for(int i=1;i<=n;i++)
37     if(!color[i])
38         if(dfs(i,1)){
39             flag=1;//有奇环
40             break;
41         }
42 if(flag) puts("No");
43 else puts("Yes");
44 return 0;
45 }

```

最大匹配

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  int G[510][510];
4  int match[510], reserve_boy[510];
5  int n, m;
6  bool dfs(int x){
7      for(int i=1; i<=m; i++)
8          if(!reserve_boy[i] && G[x][i]){
9              reserve_boy[i] = 1;
10             if(!match[i] || dfs(match[i])){
11                 match[i] = x;
12                 return true;
13             }
14         }
15     return false;
16 }
17 int main(){
18     int e; scanf("%d%d%d",&n,&m,&e);
19     while(e--){int a,b; scanf("%d%d",&a,&b); G[a][b]=1;}
20     int sum=0;
21     for(int i=1; i<=n; i++){
22         memset(reserve_boy,0,sizeof(reserve_boy));
23         if(dfs(i)) sum++;
24     }
25     printf("%d\n",sum);
26     return 0;
27 }

```

竞赛图

竞赛图的性质

1. 兰道定理

把每一个点的出度按从小到大排序形成一个新的序列s, s是合法的比分序列当且仅当

$$\sum_{i=1}^k s_i \geq \binom{k}{2}, (1 \leq k \leq n)$$

并且当 $k = n$ 时必须取等

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 int main(){
4     ios::sync_with_stdio(0);
5     cin.tie(0);cout.tie(0);
6     int n,pre=0;cin>>n;
7     vector<int>d(n,0);
8     for(int i=0;i<n;++i){
9         for(int j=0;j<n;++j){
10             char c;cin>>c;
11             if(c=='1')++d[j];
12         }
13     }
14     sort(d.begin(),d.end());
15     for(int i=0;i<n-1;++i){
16         pre+=d[i];
17         if(pre==(i+1)*i/2){
18             cout<<n-1;
19             return 0;
20         }
21     }
22     cout<<n;
23     return 0;
24 }
```

2. 竞赛图一定有哈密顿路径

3. 竞赛图有哈密顿回路的充要条件是强连通

4. 竞赛图如果有环，最小一定是三元环

求竞赛图任意三元环

```
1 #include <bits/stdc++.h>
```

```

2 using namespace std;
3 const int N = 5007, M = 5000007, INF = 0x3f3f3f3f;
4 int n, m;
5 char s[N][N];
6 bool vis[N];
7 bool dfs(int x , int fa) {
8     vis[x] = 1;
9     for(int i = 1; i <= n; ++i){
10         if(s[x][i] - '0'){
11             if(s[i][fa] - '0'){
12                 printf("%d %d %d\n", fa, x, i);
13                 return true;
14             }
15             if(!vis[i]){
16                 if(dfs(i, x))
17                     return true;
18             }
19         }
20     }
21     return false;
22 }
23 int main() {
24     scanf("%d", &n);
25     for(int i = 1; i <= n; ++ i)
26         scanf("%s", s[i] + 1);
27     for(int i = 1; i <= n; ++ i)
28         if(!vis[i])
29             if(dfs(i, i)) return 0;
30     puts("-1");
31     return 0;
32 }

```

哈密顿路径

网络流

最大流

最小割

费用流

上下界网络流

五、数论与线性代数

快速幂和逆元

```
1 using i64 = long long;
2 const i64 mod = 998244353;
3 i64 qpow(i64 a, i64 b) {
4     i64 res = 1;
5     a %= mod;
6     while (b) {
7         if (b & 1) res = (res * a) % mod;
8         b >>= 1;
9         a = (a * a) % mod;
10    }
11    return res;
12 }
13 i64 inv(i64 a) {
14     return qpow(a, mod - 2);
15 }
```

矩阵快速幂

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define int long long
4 inline int read() {
5     int x = 0, f = 1;
6     char c = getchar();
7     while (c < '0' || c > '9') {
8         if (c == '-') f = -1;
9         c = getchar();
10    }
11    while (c >= '0' && c <= '9') {
12        x = x * 10 + (c - '0');
13        c = getchar();
14    }
15    return x * f;
16 }
17 const int N = 105;
18 struct matrix {
19     int m[N][N];
20 };
21 int n, k;
22 const int mod = 1e9 + 7;
23 matrix operator*(const matrix& a, const matrix& b) {
```



```

24     matrix c;
25     memset(c.m, 0, sizeof(c.m));
26     for (int i = 0; i < n; ++i) {
27         for (int j = 0; j < n; ++j) {
28             for (int k = 0; k < n; ++k) {
29                 c.m[i][j] = (c.m[i][j] + a.m[i][k] * b.m[k][j]) % mod;
30             }
31         }
32     }
33     return c;
34 }
35 signed main() {
36     matrix a;
37     n = read(), k = read();
38     for (int i = 0; i < n; ++i) {
39         for (int j = 0; j < n; ++j) {
40             a.m[i][j] = read();
41         }
42     }
43     matrix ans;
44     memset(ans.m, 0, sizeof(ans.m));
45     for (int i = 0; i < n; ++i) {
46         ans.m[i][i] = 1;
47     }
48     while (k) {
49         if (k & 1) ans = ans * a;
50         a = a * a;
51         k >>= 1;
52     }
53     for (int i = 0; i < n; ++i) {
54         for (int j = 0; j < n; ++j) {
55             printf("%lld ", ans.m[i][j]);
56         }
57         printf("\n");
58     }
59     return 0;
60 }

```

矩阵快速幂加速递推

```

1  #include <cstdio>
2  #include <cstring>
3  using namespace std;
4  #define int long long
5  struct matrix {

```

```

6         int m[2][2];
7     };
8     const int mod = 10000;
9     inline int read() {
10         int x = 0, f = 1;
11         char c = getchar();
12         while (c < '0' || c > '9') {
13             if (c == '-') {
14                 f = -1;
15             }
16             c = getchar();
17         }
18         while (c >= '0' && c <= '9') {
19             x = x * 10 + (c - '0');
20             c = getchar();
21         }
22         return x * f;
23     }
24     matrix operator*(const matrix& a, const matrix& b) {
25         matrix c;
26         memset(c.m, 0, sizeof(c.m));
27         for (int i = 0; i < 2; ++i) {
28             for (int j = 0; j < 2; ++j) {
29                 for (int k = 0; k < 2; ++k) {
30                     c.m[i][j] = (c.m[i][j] + a.m[i][k] * b.m[k][j]) % mod;
31                 }
32             }
33         }
34         return c;
35     }
36     signed main() {
37         matrix a;
38         int n;
39         while (true) {
40             memset(a.m, 0, sizeof(a.m));
41             a.m[0][0] = 1;
42             a.m[0][1] = 1;
43             a.m[1][0] = 1;
44             n = read();
45             if (n == -1) {
46                 break;
47             }
48             if (n == 0) {
49                 puts("0");
50             }
51             else {
52                 matrix res;

```

```

53         memset(res.m, 0, sizeof(res.m));
54         for (int i = 0; i < 2; ++i) {
55             res.m[i][i] = 1;
56         }
57         while (n) {
58             if (n & 1) {
59                 res = res * a;
60             }
61             a = a * a;
62             n >>= 1;
63         }
64         printf("%lld\n", res.m[1][0]);
65     }
66 }
67 return 0;
68 }

```

扩展欧几里得算法

$$(n - m)t + kL = x - y$$

令 $a = n - m$, $b = L$, $c = x - y$

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define ll long long
4  ll extend_gcd(ll a, ll b, ll& x, ll& y) {
5      if (b == 0) {
6          x = 1; y = 0; return a;
7      }
8      ll d = extend_gcd(b, a % b, y, x);
9      y -= a / b * x;
10     return d;
11 }
12 int main() {
13     ll n, m, x, y, L;
14     cin >> x >> y >> m >> n >> L;
15     ll a = n - m, c = x - y;
16     if (a < 0) a = -a, c = -c;
17     ll d = extend_gcd(a, L, x, y);
18     if (c % d != 0) cout << "Impossible";
19     else cout << ((x * (c / d)) % (L / d) + (L / d)) % (L / d);
20 }

```

```

1 long long mod_inverse(long long a, long long m){
2     long long x, y;
3     extend_gcd(a, m, x, y);
4     return (x % m + m) % m;
5 }
6 int main(){
7     long long a, m; cin >> a >> m;
8     cout << mod_inverse(a, m);
9     return 0;
10 }

```

扩展中国剩余定理

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 const int N = 100010;
5 int n;
6 ll ai[N], mi[N];
7 ll mul(ll a,ll b,ll m){ //乘法取模: a*b % m
8     ll res=0;
9     while(b>0){
10         if(b&1) res=(res+a) % m;
11         a=(a+a) % m;
12         b>>=1;
13     }
14     return res;
15 }
16 ll extend_gcd(ll a,ll b,ll &x,ll &y){ //扩展欧几里得
17     if(b == 0){ x=1; y=0; return a;}
18     ll d = extend_gcd(b,a%b,y,x);
19     y -= a/b * x;
20     return d;
21 }
22 ll excrt(){ //求解同余方程组, 返回最小正整数解
23     ll x,y;
24     ll m1 = mi[1], a1 = ai[1]; //第1个等式
25     ll ans = 0;
26     for(int i=2;i<=n;i++){ //合并每2个等式
27         ll a2 = ai[i], m2 = mi[i]; //第2个等式
28         //合并为: aX + bY = c
29         ll a = m1, b = m2, c = (a2 - a1%m2 + m2) % m2;
30         //下面求解 aX + bY = c
31         ll d = extend_gcd(a,b,x,y); //用扩展欧几里得求x0
32         if(c%d != 0) return -1; //无解

```

```

33     x = mul(x,c/d,b/d);           //aX + bY = c 的特解t, 最小值
34     ans = a1 + x* m1;             //代回原第1个等式, 求得特解x'
35     m1 = m2/d*m1;                 //先除再乘, 避免越界。合并后的新m1
36     ans = (ans%m1 + m1) % m1;     //最小正整数解
37     a1 = ans;                     //合并后的新a1
38 }
39 return ans;
40 }
41 int main(){
42     scanf("%d", &n);
43     for(int i=1;i<=n;++i) scanf("%lld%lld",&mi[i],&ai[i]);
44     printf("%lld",exCRT());
45     return 0;
46 }

```

裴蜀定理

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  int main() {
4      int n, a, s;
5      cin >> n;
6      for (int i = 1; i <= n; ++i) {
7          cin >> a;
8          s = __gcd(s, abs(a));
9      }
10     cout << s;
11 }

```

FFT

```

1  #include<iostream>
2  #include<cstdio>
3  #include<cmath>
4  using namespace std;
5  const int MAXN=1e7+10;
6  inline int read() {
7      char c=getchar();int x=0,f=1;
8      while(c<'0' || c>'9'){if(c=='-')f=-1;c=getchar();}
9      while(c>='0'&&c<='9'){x=x*10+c-'0';c=getchar();}
10     return x*f;
11 }

```

```

12 const double Pi=acos(-1.0);
13 struct complex {
14     double x,y;
15     complex (double xx=0,double yy=0): x(xx), y(yy) {}
16 }a[MAXN],b[MAXN];
17 complex operator+(complex a,complex b){ return complex(a.x+b.x, a.y+b.y);}
18 complex operator-(complex a,complex b){ return complex(a.x-b.x, a.y-b.y);}
19 complex operator*(complex a,complex b){ return complex(a.x*b.x-a.y*b.y,
    a.x*b.y+a.y*b.x);}
20 int N,M;
21 int l,r[MAXN];
22 int limit=1;
23 void FFT(complex *A,int type) {
24     for(int i=0;i<limit;i++)
25         if(i<r[i]) swap(A[i],A[r[i]]);
26     for(int mid=1;mid<limit;mid<=<1) {
27         complex Wn(cos(Pi/mid) , type*sin(Pi/mid));
28         for(int R=mid<<1,j=0;j<limit;j+=R) {
29             complex w(1,0);
30             for(int k=0;k<mid;k++,w=w*Wn) {
31                 complex x=A[j+k],y=w*A[j+mid+k];
32                 A[j+k]=x+y;
33                 A[j+mid+k]=x-y;
34             }
35         }
36     }
37 }
38 int main() {
39     int N = read(), M = read();
40     for(int i=0;i<=N;i++) a[i].x = read();
41     for(int i=0;i<=M;i++) b[i].x = read();
42     while(limit<=N+M) limit<=<1,l++;
43     for(int i=0;i<limit;i++) r[i]=(r[i>>1]>>1)|((i&1)<<(l-1));
44     FFT(a,1);
45     FFT(b,1);
46     for(int i=0;i<=limit;i++) a[i]=a[i]*b[i];
47     FFT(a,-1);
48     for(int i=0;i<=N+M;i++) printf("%d ",(int)(a[i].x/limit+0.5));
49     return 0;
50 }

```

NTT

```

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

```

```

3 typedef long long ll;
4 const int NR = 1 << 22, g = 3, gi = 332748118, mod = 998244353;
5 // 998244353的一个原根为3且998244353-1=2^23*119, 3在模998244353意义下的逆元为
   332748118
6 int n, m, rev[NR];
7 ll a[NR], b[NR];
8 ll qpow(ll x, ll y) {
9     ll res = 1;
10    x %= mod;
11    while (y) {
12        if (y & 1) res = (res * x) % mod;
13        x = (x * x) % mod;
14        y >>= 1;
15    }
16    return res;
17 }
18 // NTT, type=1时系数表示法转点值表示法, 否则点值表示法转系数表示法
19 void NTT(ll a[], int n, int type) {
20     for (int i = 0; i < n; ++i) {
21         if (i < rev[i]) swap(a[i], a[rev[i]]);
22     }
23     for (int i = 1; i < n; i <= 1) {
24         ll gn = qpow(type ? g : gi, (mod - 1) / (i << 1));
25         for (int j = 0; j < n; j += (i << 1)) {
26             ll g0 = 1;
27             for (int k = 0; k < i; ++k, g0 = g0 * gn % mod) {
28                 ll x = a[j + k], y = g0 * a[i + j + k] % mod;
29                 a[j + k] = (x + y) % mod;
30                 a[i + j + k] = (x - y + mod) % mod;
31             }
32         }
33     }
34 }
35 int main() {
36     scanf("%d%d", &n, &m);
37     for (int i = 0; i <= n; ++i) scanf("%lld", &a[i]);
38     for (int i = 0; i <= m; ++i) scanf("%lld", &b[i]);
39     int len = 1 << max((int)ceil(log2(n + m)), 1);
40     for (int i = 0; i < len; ++i) rev[i] = (rev[i >> 1] >> 1) | ((i & 1) <<
(max((int)ceil(log2(n + m)), 1) - 1));
41     NTT(a, len, 1);
42     NTT(b, len, 1);
43     for (int i = 0; i <= len; ++i)
44         a[i] = a[i] * b[i] % mod; // O(n)乘法
45     NTT(a, len, 0); // 点值表示法转系数表示法
46     ll inv = qpow(len, mod - 2); // inv为len的逆元 (费马小定理求逆元)
47     for (int i = 0; i <= n + m; ++i) // 输出

```

```

48         printf("%lld ", a[i] * inv % mod); // 除以len在模mod意义下即为乘以inv
49     }

```

素数筛

```

1  const int N = 1e7; //定义空间大小, 1e7约10M
2  int prime[N+1]; //存放素数, 它记录visit[i] = false的项
3  bool visit[N+1]; //true表示被筛掉, 不是素数
4  int E_sieve(int n) { //埃氏筛法, 计算[2, n]内的素数
5      int k=0; //统计素数个数
6      for(int i=0; i<=n; i++) visit[i]= false; //初始化
7      for(int i=2; i<=n; i++) { //从第一个素数2开始。可优化 (1)
8          if(!visit[i]) {
9              prime[k++] = i; //i是素数, 存储到prime[]中
10             for(int j=2*i; j<=n; j+=i) //i的倍数, 都不是素数。可优化 (2)
11                 visit[j] = true; //标记为非素数, 筛掉
12         }
13     }
14     return k; //返回素数个数
15 }

```

```

1  int E_sieve(int n) {
2      for(int i = 0; i <= n; i++) visit[i]= false;
3      for(int i = 2; i*i <= n; i++) //筛掉非素数。改为i<=sqrt(n), 计算更快
4          if(!visit[i])
5              for(int j=i*i; j<=n; j+=i) visit[j] = true; //标记为非素数
6      //下面记录素数
7      int k=0; //统计素数个数
8      for(int i = 2; i <= n; i++)
9          if(!visit[i]) prime[k++] = i; //存储素数
10     return k;
11 }

```

```

1  int prime[N]; //保存质数, 为节约空间, 可以适当减小
2  bool vis[N]; //记录是否被筛
3  int euler_sieve(int n){ //欧拉筛。返回质数的个数。
4      int cnt = 0; //记录质数个数
5      memset(vis,0,sizeof(vis));
6      memset(prime,0,sizeof(prime));
7      for(int i=2;i<=n;i++){ //检查每个数, 筛去其中的合数
8          if(!vis[i]) prime[cnt++]=i; //如果没有筛过, 是质数, 记录。第一个质数是2

```



```

9         for(int j=0; j<cnt; j++){ //用已经得到的质数去筛后面的数
10             if(i*prime[j] >n) break; //只筛小于等于n的数
11             vis[i*prime[j]]=1; //关键1。用x的最小质因数筛去x
12             if(i%prime[j]==0) break; //关键2。如果不是这个数的最小质因子，结束
13         }
14     }
15     return cnt; //返回小于等于n的质数的个数
16 }

```

欧拉函数

```

1 int euler(int n){
2     int ans = n;
3     for(int p = 2; p*p <= n; ++ p){ //试除法：检查从2到sqrt(n)的每个数
4         if(n%p == 0){ //能整除，p是一个因子，而且是质因子，请思考
5             ans = ans/p*(p-1); //求欧拉函数的通式
6             while(n%p == 0) //去掉这个因子的幂，并使得下一个p是质因子
7                 n /= p; //减小了n
8         }
9     }
10    if(n != 1) ans = ans/n*(n-1); //情况(1)：n是一个质数，没有执行上面的分解
11    return ans;
12 }

```

整数分块/数论分块

$$\sum_{i=1}^n \left\lfloor \frac{n}{i} \right\rfloor$$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 int main() {
4     long long n, l, r, ans = 0;
5     cin >> n;
6     for (l = 1; l <= n; l = r + 1) {
7         r = n / (n / l);
8         ans += (r - l + 1) * (n / l);
9     }
10    cout << ans;
11 }

```

$$\sum_{i=1}^n f(i) \lfloor \frac{n}{i} \rfloor$$

前缀和 $s[i] = \sum_{j=1}^i f(j)$

```
1 long long n, l, r, ans = 0;
2 cin >> n;
3 for (l = 1; l <= n; l = r + 1) {
4     r = n / (n / l);
5     ans += (s(r) - s(l - 1)) * (n / l);
6 }
7 cout << ans;
```

$$\sum_{i=1}^n k \bmod i$$

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 int main() {
4     long long n, k;
5     cin >> n >> k;
6     long long L, R, ans = n * k;
7     for (L = 1; L <= n; L = R + 1) {
8         if (k / L == 0) break;
9         R = min(k / (k / L), n);
10        ans -= k / L * (L + R) * (R - L + 1) / 2;
11    }
12    cout << ans;
13 }
```

狄利克雷卷积

1

莫比乌斯函数和莫比乌斯反演

$$[gcd(i, j) = 1] = \sum_{d|gcd(i, j)} \mu(d)$$

Demo:

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

1

Demo:

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

1

高斯消元法

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 double a[105][105];
4 double eps=1e-7;
5 int main(){
6     int n; cin>>n;
7     for(int i=1;i<=n;++i)
8         for(int j=1;j<=n;++j)
9             cin>>a[i][j];
10    for(int i=1;i<=n;++i) {
11        int max=i;
12        for(int j=i+1;j<=n;++j)
13            if(fabs(a[j][i])>fabs(a[max][i])) max=j;
14        for(int j=1;j<=n;++j) swap(a[i][j],a[max][j]);
15        if(fabs(a[i][i])<eps) {
16            if (fabs(a[i][n+1])<eps) cout<<0;
17            else cout<<-1;
18            return 0;
19        }
20        for(int j=n+1;j>=1;j--) a[i][j]=a[i][j]/a[i][i];
21        for(int j=1;j<=n;++j) {
22            if(j!=i) {
23                double temp=a[j][i]/a[i][i];
24                for(int k=1;k<=n;++k) a[j][k]-=a[i][k]*temp;
25            }
26        }
27    }
```

```

26     }
27 }
28 for(int i=1;i<=n;++i) {
29     cout<<"x"<<i<<"=";
30     cout<<fixed<<setprecision(2)<<a[i][n+1]<< '\n';
31 }
32 return 0;
33 }

```

质因数分解

```

1 int prime[N];      //记录质数
2 int vis[N];        //记录最小质因子
3 int euler_sieve(int n){
4     int cnt=0;
5     memset(vis,0,sizeof(vis));
6     memset(prime,0,sizeof(prime));
7     for(int i=2;i<=n;i++){
8         if(!vis[i]){ vis[i]=i; prime[cnt++]=i;} //vis[]记录最小质因子
9         for(int j=0; j<cnt; j++){
10             if(i*prime[j] >n) break;
11             vis[i*prime[j]] = prime[j]; //vis[]记录最小质因子
12             if(i%prime[j]==0) break;
13         }
14     }
15     return cnt;
16 }

```

线性基

质数

约数

同余

反演及数论筛法

六、组合数学

七、计算几何

八、其他Other

常用函数

less<int> 大根堆

greater<int> 小根堆

priority_queue<pair<int,int>>的使用

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 const int _ = 3e5 + 5;
4 pair<int,int> a[_];
5 struct Compare {
6     bool operator()(const pair<int,int>& p2, const pair<int,int>& p1) {
7         if (p2.first == p1.first) {
8             return p1.second < p2.second;
9         }
10        return p1.first > p2.first;
11    }
12 };
13 void solve() {
14     int n, k;
15     cin >> n >> k;
16     priority_queue<pair<int,int>, vector<pair<int,int>>, Compare> pq;
17     for (int i = 0; i < n; ++i) {
18         int x;
19         cin >> x;
20         a[i].first = x % k;
21         a[i].second = i;
22         if (x % k == 0) cout << i + 1 << ' ';
23         else pq.push(a[i]);
24     }
25     while (!pq.empty()) {
26         cout << pq.top().second + 1 << ' ';
27         pq.pop();
28     }
29     cout << '\n';
30 }
31 int main() {
32     ios::sync_with_stdio(false);
33     cin.tie(nullptr);
```

```

34     int t;
35     cin >> t;
36     while (t--) solve();
37 }

```

unordered_map<pair<int,int>>的使用

1

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  void solve() {
4      int n, k;
5      cin >> n >> k;
6      vector<int> a(n);
7      for (int i = 0; i < n; ++i) cin >> a[i];
8      vector<int> p(n);
9      iota(p.begin(), p.end(), 0);
10     stable_sort(p.begin(), p.end(), [&](int i, int j){
11         return (a[i] - 1) % k > (a[j] - 1) % k;
12     });
13     for (int i = 0; i < n; ++i) {
14         cout << p[i] + 1 << " \n"[i == n - 1];
15     }
16 }
17 int main() {
18     ios::sync_with_stdio(false);
19     cin.tie(nullptr);
20     int t;
21     cin >> t;
22     while (t--) {
23         solve();
24     }
25     return 0;
26 }

```

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  void solve() {
4      int n, m;
5      cin >> n >> m;

```

```

6     string s;
7     cin >> s;
8     set<pair<int,int>> st;
9     vector<int> nxt(n + 1, n), lst(n + 1, -1);
10    for (int i = n - 1; i >= 0; --i) {
11        nxt[i] = s[i] == '1' ? i : nxt[i + 1];
12    }
13    for (int i = 0; i < n; ++i) {
14        lst[i + 1] = s[i] == '0' ? i : lst[i];
15    }
16    while (m--) {
17        int l, r;
18        cin >> l >> r;
19        l--;
20        l = nxt[l];
21        r = lst[r];
22        if (l > r) {
23            l = r = -1;
24        }
25        st.emplace(l, r);
26    }
27    cout << st.size() << '\n';
28 }
29 int main() {
30     ios::sync_with_stdio(false);
31     cin.tie(nullptr);
32     int t;
33     cin >> t;
34     while (t--) {
35         solve();
36     }
37     return 0;
38 }

```

set和multiset

bitset

高精度计算

离散化

基础算法

排序

归并排序

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 1e5 + 5;
4 int a[N];
5 int temp[N];
6 void merge_sort(int arr[], int l, int r) {
7     if (l == r) return;
8     int m = (l + r) / 2;
9     merge_sort(arr, l, m);
10    merge_sort(arr, m + 1, r);
11    int t1 = l, t2 = m + 1;
12    for (int i = l; i <= r; ++i) {
13        if ((arr[t1] < arr[t2] && t1 <= m) || t2 > r) {
14            temp[i] = arr[t1++];
15        }
16        else {
17            temp[i] = arr[t2++];
18        }
19    }
20    for (int i = l; i <= r; ++i) {
21        arr[i] = temp[i];
22    }
23 }
24 int main() {
25     int n;
26     scanf("%d", &n);
27     for (int i = 0; i < n; ++i) {
28         scanf("%d", &a[i]);
29     }
30     merge_sort(a, 0, n - 1);
31     for (int i = 0; i < n; ++i) {
32         printf("%d ", a[i]);
33     }
34     printf("\n");
35     return 0;
36 }
```

```
1 // 逆序对
2 #include <bits/stdc++.h>
```



```

3 using namespace std;
4 const int N = 5e5 + 5;
5 long long a[N];
6 long long temp[N];
7 long long res = 0;
8 inline long long read() {
9     long long x = 0, f = 1;
10    char ch = getchar();
11    while (ch < '0' || ch > '9') {
12        if (ch == '-') f = -1;
13        ch = getchar();
14    }
15    while (ch >= '0' && ch <= '9') {
16        x = x * 10 + (ch - '0');
17        ch = getchar();
18    }
19    return x * f;
20 }
21 void merge_sort(long long arr[], long long l, long long r) {
22     if (l == r) return;
23     long long m = l + (r - l) / 2;
24     merge_sort(arr, l, m);
25     merge_sort(arr, m + 1, r);
26     long long t1 = l, t2 = m + 1;
27     for (long long i = l; i <= r; ++i) {
28         if ((arr[t1] > arr[t2] && t2 <= r) || t1 > m) {
29             temp[i] = arr[t2++];
30             res += m - t1 + 1;
31         }
32         else {
33             temp[i] = arr[t1++];
34         }
35     }
36     for (long long i = l; i <= r; ++i) {
37         arr[i] = temp[i];
38     }
39 }
40 int main() {
41     long long n;
42     n = read();
43     for (long long i = 0; i < n; ++i) {
44         a[i] = read();
45     }
46     merge_sort(a, 0, n - 1);
47     printf("%lld\n", res);
48     return 0;
49 }

```

快速排序

```
1 #include <cstdio>
2 #include <algorithm>
3 using namespace std;
4 const int N = 1e5 + 5;
5 int a[N];
6 void quick_sort(int left, int right) {
7     int mid = a[left + (right - left) / 2];
8     int i = left, j = right - 1;
9     while (i <= j) {
10         while (a[i] < mid) ++i;
11         while (a[j] > mid) --j;
12         if (i <= j) {
13             swap(a[i], a[j]);
14             ++i;
15             --j;
16         }
17     }
18     if (left <= j) quick_sort(left, j + 1);
19     if (i < right) quick_sort(i, right);
20 }
21 int main() {
22     int n;
23     scanf("%d", &n);
24     for (int i = 0; i < n; ++i) {
25         scanf("%d", &a[i]);
26     }
27     quick_sort(0, n);
28     for (int i = 0; i < n; ++i) {
29         printf("%d ", a[i]);
30     }
31     return 0;
32 }
```

```
1 #include <cstdio>
2 #include <cstring>
3 #include <algorithm>
4 using namespace std;
5 const int N = 5e6 + 5;
6 int a[N];
7 int quick_sort(int left, int right, int k) {
8     int mid = a[left + (right - left) / 2];
```

```

9      int i = left, j = right - 1;
10     while (i <= j) {
11         while (a[i] < mid) ++i;
12         while (a[j] > mid) --j;
13         if (i <= j) {
14             swap(a[i], a[j]);
15             ++i;
16             --j;
17         }
18     }
19     if (left <= j && k <= j) return quick_sort(left, j + 1, k);
20     if (i < right && k >= i) return quick_sort(i, right, k);
21     return a[k];
22 }
23 int main() {
24     int n, k;
25     scanf("%d%d", &n, &k);
26     for (int i = 0; i < n; ++i) {
27         scanf("%d", &a[i]);
28     }
29     printf("%d\n", quick_sort(0, n, k));
30     return 0;
31 }

```

滑动窗口

```

1 class Solution {
2 public:
3     int countCompleteSubarrays(vector<int>& nums) {
4         int n = nums.size();
5         int cnt1 = 0;
6         array<int, 2001> st{0};
7         for (int n: nums) {
8             if (st[n] == 0) { ++cnt1; st[n] = 1; }
9         }
10        fill(st.begin(), st.end(), 0);
11        int cnt2 = 0;
12        int ans = 0;
13        for (int i = 0, j = 0; i < n; ++i) {
14            while (j < n && cnt2 < cnt1) {
15                if (!st[nums[j]]) {
16                    ++cnt2;
17                }
18                ++st[nums[j]];
19                ++j;

```

```

20         }
21         if (cnt1 == cnt2) ans += n - j + 1;
22         --st[nums[i]];
23         if (!st[nums[i]]) --cnt2;
24     }
25     return ans;
26 }
27 };

```

前缀和与二维前缀和

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  int main() {
4      ios::sync_with_stdio(false);
5      cin.tie(nullptr);
6      int n;
7      cin >> n;
8      vector<int> a(n + 1);
9      for (int i = 1; i <= n; ++i) cin >> a[i];
10     vector<int> sum(n + 1, 0);
11     for (int i = 1; i <= n; ++i) sum[i] = sum[i - 1] + a[i];
12     int m;
13     cin >> m;
14     while (m--) {
15         int l, r;
16         cin >> l >> r;
17         cout << sum[r] - sum[l - 1] << '\n';
18     }
19     return 0;
20 }

```

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int _ = 5005;
4  int s[_][_];
5  int main() {
6      ios::sync_with_stdio(false);
7      cin.tie(nullptr);
8      int n, m;
9      cin >> n >> m;
10     for (int i = 0; i < n; ++i) {
11         int x, y, v;

```

```

12      cin >> x >> y >> v;
13      x++, y++;
14      s[x][y] += v;
15  }
16  for (int i = 1; i <= 5001; ++i) {
17      for (int j = 1; j <= 5001; ++j) {
18          s[i][j] += s[i - 1][j] + s[i][j - 1] - s[i - 1][j - 1];
19      }
20  }
21  int res = 0;
22  for (int i = m; i <= 5001; ++i){
23      for (int j = m; j <= 5001; ++j) {
24          res = max(res, s[i][j] - s[i - m][j] - s[i][j - m] + s[i - m][j - m]
25      }
26  }
27  cout << res;
28  return 0;
29  }

```

树上前缀和

1

差分

1

树上差分

1

表达式求值

递归

二分和三分

二分查找

查找最后一个小于等于target的下标

```
1 int find(int q) {
2     int ans = 0;
3     int l = 1, r = n; // [1, n]
4     while (l <= r) {
5         int mid = (l + r) / 2;
6         if (a[mid] <= q) ans = mid, l = mid + 1;
7         else r = mid - 1;
8     }
9     return ans;
10 }
```

查找第一个大于等于target的下标

```
1 int find(int q) {
2     int ans = 0;
3     int l = 1, r = n; // [1, n]
4     while (l <= r) {
5         int mid = (l + r) / 2;
6         if (a[mid] >= q) ans = mid, r = mid - 1;
7         else l = mid + 1;
8     }
9     return ans;
10 }
```

查找第1个等于x的元素的位置 lower_bound()且pos = x

查找第1个大于x的元素的位置 upper_bound()

查找第1个等于或大于x的元素的位置 lower_bound()

查找最后1个等于x的元素的位置 upper_bound()且pos = x

查找最后1个小于x的元素的位置 lower_bound() - 1

查找最后1个等于或小于x的元素的位置 upper_bound() - 1

计算单调序列中x的个数 upper_bound() - lower_bound()

进制转换

反悔贪心

```
1 class Solution {
2 public:
3     long long findMaximumElegance(vector<vector<int>>& items, int k) {
4         sort(items.begin(), items.end(), [](const auto& i1, const auto& i2){
5             return i1[0] > i2[0];
6         });
7         int n = items.size();
8         set<long long> st;
9         long long res = 0, total_profit = 0;
10        vector<long long> dup;
11        for (int i = 0; i < n; ++i) {
12            long long p = items[i][0], c = items[i][1];
13            if (i < k) {
14                total_profit += p;
15                if (st.find(c) == st.end()) {
16                    st.insert(c);
17                } else {
18                    dup.push_back(p);
19                }
20            } else {
21                if (!dup.empty() && st.find(c) == st.end()) {
22                    st.insert(c);
23                    total_profit -= dup.back();
24                    dup.pop_back();
25                    total_profit += p;
26                }
27            }
28            res = max(res, total_profit + (long long)st.size() * (long long)st.s
29        }
30        return res;
31    }
32 };
```

单阶乘

1

连续单阶乘

1

双阶乘

1

连续双阶乘

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 __int128 read() {
4     __int128 x = 0, f = 1;
5     char c = getchar();
6     while (c < '0' || c > '9') {
7         if (c == '-') f = -1;
8         c = getchar();
9     }
10    while (c >= '0' && c <= '9') {
11        x = (x << 3) + (x << 1) + (c ^ 48);
12        c = getchar();
13    }
14    return x * f;
15 }
16 void write(__int128 x) {
17     if (x < 0) {
18         write(-x);
19         return;
20     }
21     if (x >= 10)
22         write(x / 10);
23     putchar((x % 10) ^ 48);
24 }
25 void solve() {
26     __int128 n;
27     n = read();
28     __int128 res = 0;
29     __int128 u = 5, v, r;
30     while (u <= n) {
31         v = u << 1;
32         r = n / v;
33         res += r * (r - 1) * u;
34         res += r * (u + 1) >> 1;
35         if (n % 2 == 0) res += r;
```



```
36         res += ((n - r * v + 1) >> 1 << 1) * r;
37         if (n >= r * v + u) res += (n - r * v - u + 2) >> 1;
38         u *= 5;
39     }
40     write(res);
41 }
42 int main() {
43     ios::sync_with_stdio(false);
44     cin.tie(nullptr);
45     solve();
46 }
```