David的ACM模板

一、动态规划

背包DP

01背包

```
#include <iostream>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;
const int _ = 1005;
int w[_], v[_];
long long dp[_];
int main() {
    int n, C;
    scanf("%d%d", &n, &C);
    for (int i = 1; i <= n; ++i) scanf("%d", &w[i]);
    for (int i = 1; i <= n; ++i) scanf("%d", &v[i]);
    for (int i = 1; i <= n; ++i) {
        for (int j = C; j >= v[i]; --j)
            dp[j] = max(dp[j], dp[j - v[i]] + w[i]);
    printf("%lld\n", dp[C]);
}
```

多重背包

```
#include <iostream>
#include <algorithm>
using namespace std;
const int N = 100010;
int n, C, dp[N];
int w[N], c[N], m[N];
int new_n;
int new_w[N], new_c[N], new_m[N];
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
```

```cpp
    cout.tie(nullptr);
    cin >> n >> C;
    for (int i = 1; i <= n; ++i) {
        cin >> w[i] >> c[i] >> m[i];
    }
    int new_n = 0;
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= m[i]; j <<= 1) {
            m[i] -= j;
            new_c[++new_n] = j * c[i];
            new_w[new_n] = j * w[i];
        }
        if (m[i]) {
            new_c[++new_n] = m[i] * c[i];
            new_w[new_n] = m[i] * w[i];
        }
    }
    for (int i = 1; i <= new_n; ++i) {
        for (int j = C; j >= new_c[i]; --j) {
            dp[j] = max(dp[j], dp[j - new_c[i]] + new_w[i]);
        }
    }
    cout << dp[C] << '\n';
    return 0;
}
```

单调队列优化

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
const int N = 100010;
int n, C;
int dp[N], q[N], num[N];
int w, c, m;
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    cin >> n >> C;
    for (int i = 1; i <= n; ++i) {
        cin >> w >> c >> m;
        if (m > C / c) m = C / c;
        for (int b = 0; b < c; ++b) {
            int head = 1, tail = 1;
            for (int y = 0; y <= (C - b) / c; ++y) {
                int tmp = dp[b + y * c] - y * w;
                while (head < tail && q[tail - 1] <= tmp) --tail;
                q[tail] = tmp;
                num[tail++] = y;
                while (head < tail && y - num[head] > m) ++head;
                dp[b + y * c] = max(dp[b + y * c], q[head] + y * w);
            }
        }
    }
    cout << dp[C];
    return 0;
}
```

完全背包

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

混合背包

```

```

二维费用背包

```

```

分组背包

```

```

线性DP

最长公共子序列LCS

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;
const int N=1010;
int n, m;
char a[N], b[N];
int f[N][N];
int main(){
  cin>>n>>m>>a+1>>b+1;
  for(int i=1; i<=n; i++)
    for(int j=1; j<=m; j++)
      if(a[i]==b[j]) f[i][j]=f[i-1][j-1]+1;
      else f[i][j]=max(f[i-1][j-1],max(f[i-1][j],f[i][j-1]));
  cout<<f[n][m];
}
```

最长上升子序列LIS

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;
const int N=1010;
int n, a[N];
int f[N];
int main(){
  cin>>n;
  for(int i=1; i<=n; i++) cin>>a[i];
  for(int i=1; i<=n; i++) f[i]=1;
  for(int i=1; i<=n; i++)
    for(int j=1; j<i; j++)
      if(a[j]<a[i]) f[i]=max(f[i],f[j]+1);
  int res=0;
  for(int i=1; i<=n; i++) res=max(res,f[i]);
  cout<<res;
}
```

二分

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

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] != word2[j]));

```
#include <bits/stdc++.h>
using namespace std;
const int _ = 2005;
char s[_], t[_];
int dp[_][_];
int main() {
    scanf("%s%s", s, t);
    int slen = strlen(s), tlen = strlen(t);
    for (int i = 0; i <= slen; ++i) dp[i][0] = i;
    for (int i = 0; i <= tlen; ++i) dp[0][i] = i;
    for (int i = 1; i <= slen; ++i) {
        for (int j = 1; j <= tlen; ++j) {
            dp[i][j] = std::min(dp[i - 1][j], dp[i][j - 1]) + 1;
            dp[i][j] = std::min(dp[i][j], dp[i - 1][j - 1] + (s[i - 1] != t[j -
        }
    }
    printf("%d\n", dp[slen][tlen]);
}
```

树形DP

没有上司的舞会

```
#include <bits/stdc++.h>
using namespace std;
const int _ = 6e3 + 5;
int n;
int a[_][2], b[_], w[_], f[_][2];
bool fa[_];
void dfs(int u) {
    f[u][1] = w[u];
    for (int i = 0; i < b[u]; ++i) {
        int son = a[u][i];
        dfs(son);
        f[u][0] += max(f[son][0], f[son][1]);
        f[u][1] += f[son][0];
    }
}
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cin >> n;
    for (int i = 1; i <= n; ++i) cin >> w[i];
    for (int i = 0; i < n - 1; ++i) {
        int x, y;
        cin >> x >> y;
        a[y][b[y]++] = x;
        fa[x] = true;
    }
    int root = 1;
    while (fa[root]) root++;
    dfs(root);
    cout << max(f[root][0], f[root][1]);
}
```

树的重心

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
#include <vector>
using namespace std;

const int N=100010;
int n, a, b;
vector<int> e[N];
int siz[N], pos, ans=1e9;

void dfs(int x, int fa){
  siz[x]=1;
  int mx=0;
  for(auto y : e[x]){
    if(y == fa) continue;
    dfs(y, x);
    siz[x] += siz[y];
    mx=max(mx, siz[y]);
  }
  mx=max(mx, n-siz[x]);
  if(mx<ans) ans=mx,pos=x;
}
int main(){
  scanf("%d", &n);
  for(int i=1;i<n;i++){
    scanf("%d%d",&a,&b);
    e[a].push_back(b);
    e[b].push_back(a);
  }
  dfs(1, 0);
  printf("%d\n",ans);
  return 0;
}
```

树的直径

```cpp
#include <cstring>
#include <iostream>
#include <algorithm>
using namespace std;

const int N=10010,M=20010;
int n,a,b,c,ans;
struct edge{int v,w;};
```

```cpp
vector<edge> e[N];

int dfs(int x,int fa){
  int d1=0,d2=0;
  for(auto ed : e[x]){
    int y=ed.v, z=ed.w;
    if(y==fa) continue;
    int d=dfs(y,x)+z;
    if(d>=d1) d2=d1,d1=d;
    else if(d>d2) d2=d;
  }
  ans=max(ans,d1+d2);
  return d1;
}
int main(){
  cin>>n;
  for(int i=1; i<n; i++){
    cin>>a>>b>>c;
    e[a].push_back({b,c});
    e[b].push_back({a,c});
  }
  dfs(1,-1);
  cout<<ans;
}
```

树的中心

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
#include <vector>
using namespace std;

const int N=20010;
int n,a,b,c,ans=2e9;
struct edge{int v,w;};
vector<edge> e[N];
int d1[N],d2[N],path[N],up[N];

void dfs(int x,int fa){
  for(auto ed : e[x]){
    int y=ed.v, z=ed.w;
    if(y==fa) continue;
    dfs(y, x);
    if(d1[y]+z>d1[x])
```

```
      d2[x]=d1[x],d1[x]=d1[y]+z,path[x]=y;
    else if(d1[y]+z>d2[x]) d2[x]=d1[y]+z;
  }
}
void dfs2(int x,int fa){
  for(auto ed : e[x]){
    int y=ed.v, z=ed.w;
    if(y==fa) continue;
    if(y==path[x])up[y]=max(up[x],d2[x])+z;
    else up[y]=max(up[x],d1[x])+z;
    dfs2(y, x);
  }
}
int main(){
  cin>>n;
  for(int i=1; i<n; i++){
    cin>>a>>b>>c;
    e[a].push_back({b,c});
    e[b].push_back({a,c});
  }
  dfs(1, 0);
  dfs2(1, 0);
  for(int i=1; i<=n; i++)
    ans=min(ans,max(d1[i],up[i]));
  cout<<ans;
}
```

树形背包

```

```

换根dp

```

```

数位DP

```

```

区间DP

环形石子合并

```cpp
#include <bits/stdc++.h>
using namespace std;
const int _ = 205;
long long a[_], s[_], minn[_][_], maxx[_][_], res_min = LLONG_MAX, res_max = -1;
int main() {
    ios::sync_with_stdio(false); cin.tie(nullptr); cout.tie(nullptr);
    int n; cin >> n;
    for (int i = 1; i <= n; ++i) {
        cin >> a[i]; a[i + n] = a[i];
    }
    for (int i = 1; i <= 2 * n; ++i) {
        s[i] = s[i - 1] + a[i];
    }
    for (int len = 2; len <= n; ++len) {
        for (int i = 1; i <= 2 * n - len + 1; ++i) {
            int j = i + len - 1;
            minn[i][j] = LLONG_MAX, maxx[i][j] = -1;
            for (int k = i; k < j; ++k) {
                minn[i][j] = min(minn[i][j], minn[i][k] + minn[k + 1][j] + s[j] - s[i - 1]);
                maxx[i][j] = max(maxx[i][j], maxx[i][k] + maxx[k + 1][j] + s[j] - s[i - 1]);
            }
        }
    }
    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]);
    }
    cout << res_min << '\n' << res_max << '\n';
}
```

状压DP

bitset

Demo: 国王问题

行内合法: !(i & i >> 1)为真

行间兼容: !(a & b) && !(a & b >> 1) && !(a & b << 1)为真

```cpp
#include <bits/stdc++.h>
using namespace std;
int n, k;
int cnt;
int s[1<<12];
int num[1<<12];
long long f[12][144][1<<12];
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    cin >> n >> k;
    for (int i = 0; i < (1 << n); ++i) {
        if (!(i & i >> 1)) {
            s[cnt++] = i;
            for (int j = 0; j < n; ++j) {
                num[i] += (i >> j & 1);
            }
        }
    }
    f[0][0][0] = 1;
    for (int i = 1; i <= n + 1; ++i) {
        for (int j = 0; j <= k; ++j) {
            for (int a = 0; a < cnt; ++a) {
                for (int b = 0; b < cnt; ++b) {
                    int c = num[s[a]];
                    if ((j >= c) && !(s[b] & s[a]) && !(s[b] & (s[a] << 1)) && !(s[b] & (s[a] >> 1))) {
                        f[i][j][a] += f[i - 1][j - c][b];
                    }
                }
            }
        }
    }
    cout << f[n + 1][k][0] << '\n';
    return 0;
}
```

Demo:玉米田

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;
const int P=1e9;
```

```
int n,m;      //行数，列数
int g[14];    //各行的状态值
int cnt;      //同一行的合法状态个数
int s[1<<14]; //一行的合法状态集
int f[14][1<<14];
//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
int main(){
  cin>>n>>m;
  for(int i=1; i<=n; i++)
    for(int j=1; j<=m; j++){
      int x; cin>>x;
      g[i]=(g[i]<<1)+x; //各行的状态值
    }
  for(int i=0;i<(1<<m);i++) //枚举一行所有状态
    if(!(i&i>>1))           //如果不存在相邻的1
      s[cnt++]=i;           //保存一行的合法状态
  f[0][0]=1;
  for(int i=1; i<=n+1; i++) //枚举行
  for(int a=0; a<cnt; a++)  //枚举第i行合法状态
  for(int b=0; b<cnt; b++)  //枚举第i-1行合法状态
    if(!(s[a]&s[b])         //不能同列均为1
      &&(s[a]&g[i])==s[a])  //种在肥沃土地上
        f[i][a]=(f[i][a]+f[i-1][b])%P;
  printf("%d\n",f[n+1][0]);
  return 0;
}
```

Demo:炮兵阵地

```
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N=110, M=1<<10;
int n,m;     //行数,列数
int g[N];    //存储地图各行数值
int cnt;     //一行的合法状态个数
int s[M];    //一行的合法状态集
int num[M]; //每个合法状态包含1的个数
int f[N][M][M]; //110*1024*1024*4 = 440MB
// f[i][a][b]表示已放好前i行,
// 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
int main(){
  cin>>n>>m;
```

```cpp
  for(int i=1;i<=n;i++)
    for(int j=0;j<m;j++){
      char c; cin>>c;
      if(c=='P') g[i]+=1<<(m-j-1); //地图各行数值
    }
  for(int i=0; i<(1<<m); i++)   //枚举一行的所有状态
    if(!(i&i>>1) && !(i&i>>2)){ //如果不存在11和101
      s[cnt++]=i;               //保存一行的合法状态
      for(int j=0; j<m; j++)
      num[i]+=(i>>j&1);         //每个合法状态包含1的个数
    }
  for(int i=1; i<=n+2; i++) //枚举行
  for(int a=0; a<cnt; a++)  //枚举第i行合法状态
  for(int b=0; b<cnt; b++)  //枚举第i-1行合法状态
  for(int c=0; c<cnt; c++)  //枚举第i-2行合法状态
    if(!(s[a]&s[b])&&!(s[a]&s[c])&&!(s[b]&s[c])
      &&(g[i]&s[a])==s[a]&&(g[i-1]&s[b])==s[b])
        f[i][a][b]=max(f[i][a][b],f[i-1][b][c]+num[s[a]]);
  cout<<f[n+2][0][0]<<endl;
  return 0;
}
```

Demo:Explode 'Em All（炸十字形）

```cpp
#include <bits/stdc++.h>
using namespace std;
char s[30][30];
int r[30], dp[(1 << 25) + 100];
int has[(1 << 25) + 100];
int lowbit(int x) { return x & (-x); }
int cal(int x) {
    int cnt = 0;
    while (x) {
        x -= lowbit(x);
        ++cnt;
    }
    return cnt;
}
int main() {
    int n, m;
    scanf("%d %d", &n, &m);
    for (int i = 0; i < n; ++i) {
        has[(1 << i)] = i;
    }
    memset(r, 0, sizeof(r));
    for (int i = 0; i < n; ++i) {
        scanf("%s", s[i]);
        for (int j = 0; j < m; ++j) {
            if (s[i][j] == '*') {
                r[i] = r[i] | (1 << j);
            }
        }
    }
    memset(dp, 0, sizeof(dp));
    int ans = 0x3f3f3f3f;
    int rcnt, ccnt;
    for (int i = 1; i < (1 << n); ++i) {
        dp[i] = dp[i - lowbit(i)] | r[has[lowbit(i)]];
        rcnt = n - cal(i);
        ccnt = cal(dp[i]);
        ans = min(ans, max(rcnt, ccnt));
    }
    printf("%d\n", ans);
    return 0;
}
```

Demo:樱桃炸弹（炸3*3区域）

```

```

DP优化

单调队列优化DP

Demo:

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

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

```cpp
        while (h <= t && f[q[t]] >= f[i - 1]) t--;
        q[++t] = i - 1;
        if (q[h] < i - m) ++h;
        f[i] = f[q[h]] + w[i];
        if (i > n - m) ans = min(ans, f[i]);
    }
    cout << ans;
}
```

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 2e5 + 10;
#define int long long
int n, m, w[N], f[N], q[N];
bool check(int x) {
    int h = 1, t = 0;
    for (int i = 1; i <= n; ++i) {
        while (h <= t && f[q[t]] >= f[i - 1]) t--;
        q[++t] = i - 1;
        if (q[h] < i - x) ++h;
        f[i] = f[q[h]] + w[i];
        if (i > n - x && f[i] <= m) return true;
    }
    return false;
}
signed main() {
    cin >> n >> m;
    for (int i = 1; i <= n; ++i) cin >> w[i];
    int l = 0, r = n;
    int ans = n;
    while (l <= r) {
        int mid = (l + r) >> 1;
        if (check(mid)) {
            ans = mid;
            r = mid - 1;
        } else {
            l = mid + 1;
        }
    }
    cout << ans - 1;
}
```

Demo:

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

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 2e5 + 10;
int n, L, R, a[N], f[N], q[N];
signed main() {
    cin >> n >> L >> R;
    for (int i = 0; i <= n; ++i) cin >> a[i];
    memset(f, -0x3f, sizeof(f));
    f[0] = 0;
    int ans = -0x3f3f3f3f;
    int h = 1, t = 0;
    for (int i = L; i <= n; ++i) {
        while (h <= t && f[q[t]] <= f[i - L]) t--;
        q[++t] = i - L;
        if (q[h] < i - R) ++h;
        f[i] = f[q[h]] + a[i];
        if (i > n - R) ans = max(ans, f[i]);
    }
    cout << ans;
}
```

斜率优化DP

double与long double

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

Demo:

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

分析：

(1) 新点 i - 1 与队尾点斜率 $\le$ 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 $\le$ k_i，则队头出队

(4) 此时队头为最优决策点

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
```

```cpp
typedef long long LL;
const int N = 500010;
int n, m, q[N];
LL s[N], f[N];
double slope(int i, int j) {
    return (double)(f[i] + s[i] * s[i] - f[j] - s[j] * s[j]) / (s[i] == s[j] ?
1e-9 : s[i] - s[j]);
}
int main() {
    while (~scanf("%d%d", &n, &m)) {
        for (int i = 1; i <= n; ++i) {
            scanf("%lld", &s[i]);
            s[i] += s[i - 1];
        }
        int h = 1, t = 0;
        for (int i = 1; i <= n; ++i) {
            while (h < t && slope(i - 1, q[t]) <= slope(q[t], q[t - 1])) t--;
            q[++t] = i - 1;
            while (h < t && slope(q[h + 1], q[h]) <= 2 * s[i]) h++;
            int j = q[h];
            f[i] = f[j] + (s[i] - s[j]) * (s[i] - s[j]) + m;
        }
        printf("%lld\n", f[n]);
    }
}
```

```cpp
#include <bits/stdc++.h>
typedef long long LL;
const int N = 500010;
int n, m, q[N];
LL s[N], f[N];
LL dx(int i, int j) { return s[i] - s[j]; }
LL dy(int i, int j) { return f[i] + s[i] * s[i] - f[j] - s[j] * s[j]; }
int main() {
    while (~scanf("%d%d", &n, &m)) {
        for (int i = 1; i <= n; ++i) {
            scanf("%lld", &s[i]);
            s[i] += s[i - 1];
        }
        int h = 1, t = 0;
        for (int i = 1; i <= n; ++i) {
            while (h < t && dy(i - 1, q[t]) * dx(q[t], q[t - 1])
                           <= dx(i - 1, q[t]) * dy(q[t], q[t - 1])) --t;
            q[++t] = i - 1;
            while (h < t && dy(q[h + 1], q[h]) <= dx(q[h + 1], q[h]) * 2 * s[i]) ++h;
            int j = q[h];
            f[i] = f[j] + (s[i] - s[j]) * (s[i] - s[j]) + m;
        }
        printf("%lld\n", f[n]);
    }
}
```

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:

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

```
#include <iostream>
#include <algorithm>
using namespace std;
const int _ = 5e4 + 5;
typedef long long ll;
ll n, L, q[_];
ll s[_], f[_];
ll a(int i) { return s[i] + i; }
ll b(int i) { return a(i) + L + 1; }
ll X(int i) { return b(i); }
ll Y(int i) { return f[i] + b(i) * b(i); }
double slope(int i, int j) {
    return (double)(Y(i) - Y(j)) / (X(i) == X(j) ? 1e-9 : X(i) - X(j));
}
int main() {
    ios::sync_with_stdio(false); cin.tie(nullptr); cout.tie(nullptr);
    cin >> n >> L;
    for (ll i = 1; i <= n; ++i) cin >> s[i], s[i] += s[i - 1];
    ll h = 1, t = 0;
    for (ll i = 1; i <= n; ++i) {
        while (h < t && slope(i - 1, q[t]) <= slope(q[t], q[t - 1])) --t;
        q[++t] = i - 1;
        while (h < t && slope(q[h + 1], q[h]) <= 2 * a(i)) ++h;
        ll j = q[h];
        f[i] = f[j] + (a(i) - b(j)) * (a(i) - b(j));
    }
    cout << f[n];
    return 0;
}
```

Demo:

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

```cpp
#include <bits/stdc++.h>
using namespace std;
const int _ = 1e6 + 5;
typedef long long ll;
ll n, a, b, c, q[_];
ll s[_], f[_];
ll X(int i) { return s[i]; }
ll Y(int i) { return f[i] + a * s[i] * s[i] - b * s[i]; }
double slope(int i, int j) {
    return (double)(Y(i) - Y(j)) / (X(i) == X(j) ? 1e-9 : X(i) - X(j));
}
int main() {
    ios::sync_with_stdio(false); cin.tie(0); cout.tie(0);
    cin >> n >> a >> b >> c;
    for (ll i = 1; i <= n; ++i) cin >> s[i], s[i] += s[i - 1];
    ll h = 1, t = 0;
    for (ll i = 1; i <= n; ++i) {
        while (h < t && slope(i - 1, q[t]) >= slope(q[t], q[t - 1])) --t;
        q[++t] = i - 1;
        while (h < t && slope(q[h + 1], q[h]) >= 2 * a * s[i]) ++h;
        ll j = q[h];
        f[i] = f[j] + a * (s[i] - s[j]) * (s[i] - s[j]) + b * (s[i] - s[j]) + c;
    }
    cout << f[n];
    return 0;
}
```

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

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

```cpp
using namespace std;
const int _ = 205;
long long a[_], b[_][_], s[_], minn[_][_], maxx[_][_], res_min = 1e9, res_max = -1;
int main() {
    ios::sync_with_stdio(false); cin.tie(nullptr); cout.tie(nullptr);
    int n; cin >> n;
    for (int i = 1; i <= n; ++i) cin >> a[i], a[i + n] = a[i];
    for (int i = 1; i <= 2 * n; ++i) b[i][i] = i, s[i] = s[i - 1] + a[i];
    for (int len = 2; len <= n; ++len) {
        for (int i = 1; i <= 2 * n - len + 1; ++i) {
            int j = i + len - 1;
            minn[i][j] = 1e9, maxx[i][j] = -1;
            for (int k = i; k < j; ++k) {
                minn[i][j] = min(minn[i][j], minn[i][k] + minn[k + 1][j] + s[j] - s[i - 1]);
                                maxx[i][j] = max(maxx[i][j], maxx[i][k] + maxx[k + 1][j] + s[j] - s[i - 1]);
            }
        }
    }
    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]);
    cout << res_min << '\n' << res_max << '\n';
}
```

```cpp
// 开两倍数组
#include <bits/stdc++.h>
using namespace std;
const int _ = 205;
long long a[_], b[_][_], s[_], minn[_][_], maxx[_][_], res_min = 1e9, res_max = -1;
int main() {
    ios::sync_with_stdio(false); cin.tie(nullptr); cout.tie(nullptr);
    int n; cin >> n;
    for (int i = 1; i <= n; ++i) cin >> a[i], a[i + n] = a[i];
    for (int i = 1; i <= 2 * n; ++i) b[i][i] = i, s[i] = s[i - 1] + a[i];
    for (int len = 2; len <= n; ++len) {
        for (int i = 1; i <= 2 * n - len + 1; ++i) {
            int j = i + len - 1;
            minn[i][j] = 1e9, maxx[i][j] = -1;
            for (int k = b[i][j - 1]; k <= b[i + 1][j]; ++k) {
                if (minn[i][j] > minn[i][k] + minn[k + 1][j] + s[j] - s[i - 1]) {
                    minn[i][j] = minn[i][k] + minn[k + 1][j] + s[j] - s[i - 1];
                    b[i][j] = k;
                }
            }
            maxx[i][j] = max(maxx[i][j - 1], maxx[i + 1][j]) + s[j] - s[i - 1];
        }
    }
    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]);
    cout << res_min << '\n' << res_max << '\n';
}
```

二、字符串

最小表示法

```cpp
#include <bits/stdc++.h>
using namespace std;
const int _ = 6e5 + 10;
int a[_];
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int n;
    cin >> n;
    for (int i = 1; i <= n; ++i) cin >> a[i], a[i + n] = a[i];
    int i = 1, j = 2, k = 0;
    while (i <= n && j <= n) {
        for (k = 0; k < n && a[i + k] == a[j + k]; ++k);
        a[i + k] > a[j + k] ? i = i + k + 1 : j = j + k + 1;
        if (i == j) ++j;
    }
    int t = min(i, j);
    for (int i = t; i < t + n; ++i) cout << a[i] << " ";
    return 0;
}
```

KMP

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

```cpp
char s[N], p[N];
int nxt[N];
int main() {
    scanf("%s%s", s + 1, p + 1);
    int slen = strlen(s + 1), plen = strlen(p + 1);
    nxt[1] = 0;
    for (int i = 2, j = 0; i <= plen; ++i) {
        while (j > 0 && p[i] != p[j + 1]) j = nxt[j];
        if (p[i] == p[j + 1]) ++j;
        nxt[i] = j;
    }
    for (int i = 1, j = 0; i <= slen; ++i) {
        while (j > 0 && s[i] != p[j + 1]) j = nxt[j];
        if (s[i] == p[j + 1]) ++j;
        if (j == plen) {
            printf("%d\n", i - plen + 1);
        }
    }
    for (int i = 1; i <= plen; ++i) {
        printf("%d ", nxt[i]);
    }
}
```

统计前缀个数

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1e6 + 5;
char p[N];
int nxt[N], cnt[N];
int main() {
    scanf("%s", p + 1);
    int plen = strlen(p + 1);
    nxt[1] = 0;
    for (int i = 2, j = 0; i <= plen; ++i) {
        while (j > 0 && p[i] != p[j + 1]) j = nxt[j];
        if (p[i] == p[j + 1]) ++j;
        nxt[i] = j;
    }
    for (int i = 1; i <= plen; ++i) cnt[i]++;
    for (int i = plen; i >= 1; --i) {
        cnt[nxt[i]] += cnt[i];
    }
    for (int i = 1; i <= plen; ++i) {
        printf("%d ", cnt[i]);
    }
}
```

扩展KMP

```
#include <bits/stdc++.h>
using namespace std;
const int _ = 2e7 + 5;
#define int long long
char s[_], t[_];
int z[_], p[_];
void get_z(char* s, int n) {
    z[1] = n;
    for (int i = 2, l = 0, r = 0; i <= n; ++i) {
        if (i <= r) z[i] = min(z[i - l + 1], r - i + 1);
        while (1 + z[i] <= n && i + z[i] <= n && s[1 + z[i]] == s[i + z[i]])
z[i]++;
        if (i + z[i] - 1 > r) l = i, r = i + z[i] - 1;
    }
}
void get_p(char* s, int n, char* t, int m) {
    for (int i = 1, l = 0, r = 0; i <= m; ++i) {
        if (i <= r) p[i] = min(z[i - l + 1], r - i + 1);
        while (1 + p[i] <= n && i + p[i] <= m && s[1 + p[i]] == t[i + p[i]])
p[i]++;
        if (i + p[i] - 1 > r) l = i, r = i + p[i] - 1;
    }
}
int xxor(int a[], int n) {
    if (n == 0) return 0;
    int res = 0;
    for (int i = 1; i <= n; ++i) {
            res ^= i * (a[i] + 1);
    }
    return res;
}
signed main() {
    scanf("%s%s", t + 1, s + 1);
    int tlen = strlen(t + 1);
    int slen = strlen(s + 1);
    get_z(s, slen);
    get_p(s, slen, t, tlen);
    printf("%lld\n", xxor(z, slen));
    printf("%lld\n", xxor(p, tlen));
}
```

Trie树

Trie模板

```cpp
#include <iostream>
#include <cstring>
#include <string>
#include <vector>
using namespace std;
constexpr int N = 3e6 + 5;
int getnum(char c) {
    if (c >= 'A' && c <= 'Z') return c - 'A';
    else if (c >= 'a' && c <= 'z') return c - 'a' + 26;
    else return c - '0' + 52;
}
struct Node {
    int son[65]; // 48 97
    int num;
}t[N];
class Trie {
public:
    int cnt;
    Trie() {
        memset(t, 0, sizeof(t));
        cnt = 0;
    }
    void insert(string s) {
        int now = 0;
        for (int i = 0; i < s.size(); ++i) {
            int c = getnum(s[i]);
            if (!t[now].son[c]) t[now].son[c] = ++cnt;
            now = t[now].son[c];
            ++t[now].num;
        }
    }
    int find(string s) {
        int now = 0;
        for (int i = 0; i < s.size(); ++i) {
            int c = getnum(s[i]);
            if (!t[now].son[c]) return 0;
            now = t[now].son[c];
        }
        return t[now].num;
    }
};
void solve() {
    int n, q;
    cin >> n >> q;
    Trie t;
    string s;
    for (int i = 0; i < n; ++i) {
        cin >> s;
        t.insert(s);
    }
    while (q--) {
        string s;
        cin >> s;
        cout << t.find(s) << '\n';
    }
}
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int t;
    cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

```cpp
#include <cstdio>
#include <cstring>
using namespace std;
const int N = 5e5 + 5;
struct Node {
    bool repeat = false;
    int son[26]{ 0 };
    int num = 0;
}t[N];
int cnt = 0;
void insert(char* s) {
    int now = 0;
    for (int i = 0; s[i]; ++i) {
        int c = s[i] - 'a';
        if (!t[now].son[c]) t[now].son[c] = ++cnt;
        now = t[now].son[c];
    }
    ++t[now].num;
}
int find(char* s) {
    int now = 0;
    for (int i = 0; s[i]; ++i) {
        int c = s[i] - 'a';
        if (!t[now].son[c]) return 3;
        now = t[now].son[c];
    }
    if (t[now].num == 0) return 3;
    if (t[now].repeat == false) {
        t[now].repeat = true;
        return 1;
    }
    return 2;
}
char s[55];
int main() {
    int n;
    scanf("%d", &n);
    while (n--) {
        scanf("%s", s);
        insert(s);
    }
    int m;
    scanf("%d", &m);
    while (m--) {
        scanf("%s", s);
        int r = find(s);
        if (r == 1) puts("OK");
        if (r == 2) puts("REPEAT");
        if (r == 3) puts("WRONG");
    }
}
```

AC自动机

AC自动机(简单版)模板题

```
#include <bits/stdc++.h>
using namespace std;
const int _ = 1e6 + 5;
int t[_][26], fail[_], cnt[_];
int pos = 0;
void insert(char* p) {
    int now = 0;
    for (int i = 0; p[i]; ++i) {
        int ch = p[i] - 'a';
        if (t[now][ch] == 0) t[now][ch] = ++pos;
        now = t[now][ch];
    }
    ++cnt[now];
}
void getFail() {
    queue<int> q;
    for (int i = 0; i < 26; ++i) {
        if (t[0][i]) {
            fail[t[0][i]] = 0;
            q.push(t[0][i]);
        }
    }
    while (!q.empty()) {
        int now = q.front();
        q.pop();
        for (int i = 0; i < 26; ++i) {
            if (t[now][i]) {
                fail[t[now][i]] = t[fail[now]][i];
                q.push(t[now][i]);
            } else {
                t[now][i] = t[fail[now]][i];
            }
        }
    }
}
int query(char* s) {
    int ans = 0;
    int now = 0;
    for (int i = 0; s[i]; ++i) {
        int ch = s[i] - 'a';
        now = t[now][ch];
        for (int j = now; j && cnt[j] != -1; j = fail[j]) {
            ans += cnt[j];
            cnt[j] = -1;
        }
    }
    return ans;
}
char s[_];
int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        scanf("%s", s);
        insert(s);
    }
    getFail();
    scanf("%s", s);
    printf("%d\n", query(s));
}
```

AC自动机(加强版)模板

```
#include <bits/stdc++.h>
using namespace std;
const int _ = 1e6 + 5;
int t[_][26], fail[_], cnt[_], tag[_];
int pos;
char s[_];
char p[155][75];
void insert(int idx) {
    int now = 0;
    for (int i = 0; p[idx][i]; ++i) {
        int ch = p[idx][i] - 'a';
        if (!t[now][ch]) t[now][ch] = ++pos;
        now = t[now][ch];
    }
    tag[now] = idx;
}
void getFail() {
    queue<int> q;
    for (int i = 0; i < 26; ++i) {
        if (t[0][i]) {
            fail[t[0][i]] = 0;
            q.push(t[0][i]);
        }
    }
    while (!q.empty()) {
        int now = q.front();
        q.pop();
        for (int i = 0; i < 26; ++i) {
            if (t[now][i]) {
                fail[t[now][i]] = t[fail[now]][i];
                q.push(t[now][i]);
            } else {
                t[now][i] = t[fail[now]][i];
            }
        }
```

```cpp
    }
}
void query() {
    int now = 0;
    for (int i = 0; s[i]; ++i) {
        int ch = s[i] - 'a';
        now = t[now][ch];
        for (int j = now; j; j = fail[j]) {
            ++cnt[tag[j]];
        }
    }
}
int main() {
    int N;
    while (~scanf("%d", &N) && N) {
        memset(t, 0, sizeof(t));
        memset(cnt, 0, sizeof(cnt));
        memset(tag, 0, sizeof(tag));
        pos = 0;
        for (int i = 1; i <= N; ++i) {
            scanf("%s", p[i]);
            insert(i);
        }
        getFail();
        scanf("%s", s);
        query();
        int maxx = -1;
        for (int i = 1; i <= N; ++i) {
            if (cnt[i] > maxx) maxx = cnt[i];
        }
        if (maxx != -1) printf("%d\n", maxx);
        for (int i = 1; i <= N; ++i) {
            if (cnt[i] == maxx) {
                printf("%s\n", p[i]);
            }
        }
    }
}
```

回文自动机

```cpp
class Solution {
public:
    static constexpr int N = 1e4 + 5;
    int c[N];
    struct node {
        int len, fail, son[26], sz;
        void init(int l) {
            memset(son, 0, sizeof(son));
            fail = sz = 0;
            len = l;
        }
    }t[N];
    long long num, last[2], ans, L, R;
    void init() {
        last[0] = last[1] = 0;
        ans = 0, num = 1;
        L = 5e3 + 5, R = 5e3 + 4;
        t[0].init(0);
        memset(c, -1, sizeof(c));
        t[1].init(-1);
        t[0].fail = 1;
    }
    int getfail(int p) {
        while (c[R - t[p].len - 1] != c[R]) p = t[p].fail;
        return p;
    }
    void insert(int x) {
        c[++R] = x;
        int f = getfail(last[1]);
        int now = t[f].son[x];
        if (!now) {
            now = ++num;
            t[now].init(t[f].len + 2);
            t[now].fail = t[getfail(t[f].fail)].son[x];
            t[now].sz = t[t[now].fail].sz + 1;
            t[f].son[x] = now;
        }
        last[1] = now;
        if (R - L + 1 == t[now].len) last[0] = now;
        ans += t[now].sz;
    }
    int countSubstrings(string s) {
        init();
        for (int i = 0; i < s.size(); ++i) {
            insert(s[i] - 'a');
        }
        return ans;
    }
};
```

SA

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 2e6 + 5;
char s[N];
int sa[N], cnt[N], x[N], y[N], rk[N], h[N];
int n;
void calc_sa() {
    int m = 127;
    for (int i = 1; i <= m; ++i) cnt[i] = 0;
    for (int i = 1; i <= n; ++i) ++cnt[x[i] = s[i]];
    for (int i = 2; i <= m; ++i) cnt[i] += cnt[i - 1];
    for (int i = n; i >= 1; --i) sa[cnt[x[i]]--] = i;
    for (int k = 1; k <= n; k <<= 1) {
        int p = 0;
        for (int i = n - k + 1; i <= n; ++i) y[++p] = i;
        for (int i = 1; i <= n; ++i) if (sa[i] > k) y[++p] = sa[i] - k;
        for (int i = 1; i <= m; ++i) cnt[i] = 0;
        for (int i = 1; i <= n; ++i) ++cnt[x[i]];
        for (int i = 2; i <= m; ++i) cnt[i] += cnt[i - 1];
        for (int i = n; i >= 1; --i) sa[cnt[x[y[i]]]--] = y[i], y[i] = 0;
        swap(x, y);
        x[sa[1]] = 1;
        p = 1;
        for (int i = 2; i <= n; ++i) {
            x[sa[i]] = (y[sa[i]] == y[sa[i - 1]] && y[sa[i] + k] == y[sa[i - 1]
        }
        if (p >= n) break;
        m = p;
    }
}
void getheight() {
    int k = 0;
    for (int i = 1; i <= n; ++i) rk[sa[i]] = i;
    for (int i = 1; i <= n; ++i) {
        if (rk[1] == 1) continue;
        if (k) k--;
        int j = sa[rk[i] - 1];
        while (i + k <= n && j + k <= n && s[i + k] == s[j + k]) ++k;
        h[rk[i]] = k;
    }
}
int main() {
    scanf("%d", &n);
    scanf("%s", s + 1);
    n = strlen(s + 1);
    calc_sa();
    getheight();
    long long res = 0LL;
    for (int i = 1; i <= n; ++i) {
        res += n + 1 - sa[i] - h[i];
    }
    printf("%lld", res);
    return 0;
}
```

三、数据结构

手写栈

```
struct MyStack {
    int a[N];
    int t = 0;
    void push(int x) { a[++t] = x; }
    int top() { return a[t]; }
    void pop() { t--; }
    int empty() { return t == 0 ? 1 : 0; }
}stk;
```

单调栈

```
#include <bits/stdc++.h>
using namespace std;
const int N = 3e6 + 5;
int h[N];
int ans[N];
int main() {
    int n;
    scanf("%d", &n);
    for (int i = 1; i <= n; ++i) scanf("%d", &h[i]);
    stack<int> stk;
    for (int i = n; i >= 1; --i) {
        while (!stk.empty() && h[stk.top()] <= h[i]) stk.pop();
        if (stk.empty()) ans[i] = 0;
        else ans[i] = stk.top();
        stk.push(i);
    }
    for (int i = 1; i <= n; ++i) printf("%d ", ans[i]);
    return 0;
}
```

单调队列

```
#include <iostream>
using namespace std;
const int N=1000010;
int a[N], q[N];
int main(){
  int n, k; scanf("%d%d", &n, &k);
  for(int i=1; i<=n; i++) scanf("%d", &a[i]);
  // 维护窗口最小值
  int h=1, t=0;                    //清空队列
  for(int i=1; i<=n; i++){         //枚举序列
    while(h<=t&&a[q[t]]>=a[i]) t--; //队尾出队(队列不空且新元素更优)
    q[++t]=i;                      //队尾入队(存储下标 方便判断队头出队)
    if(q[h]<i-k+1) h++;            //队头出队(队头元素滑出窗口)
    if(i>=k) printf("%d ", a[q[h]]);  //使用最值
  }
  puts("");
  // 维护窗口最大值
  h=1, t=0;
  for(int i=1; i<=n; i++){
    while(h<=t&&a[q[t]]<=a[i]) t--;
    q[++t]=i;
    if(q[h]<i-k+1) h++;
    if(i>=k) printf("%d ", a[q[h]]);
  }
}
```

二叉堆

```
#include <cstdio>
#include <algorithm>
using namespace std;
const int N = 1e6 + 5;
int heap[N];
int len = 0;
void push(int x) {
    heap[++len] = x;
    int i = len;
    while (i > 1 && heap[i / 2] > heap[i]) {
        swap(heap[i / 2], heap[i]);
        i /= 2;
    }
}
void pop() {
    heap[1] = heap[len--];
    int i = 1;
    while (2 * i <= len) {
        int son = 2 * i;
        if (son < len && heap[son + 1] < heap[son]) {
            son++;
        }
        if (heap[son] < heap[i]) {
            swap(heap[son], heap[i]);
            i = son;
        }
        else {
            break;
        }
    }
}
int main() {
    int n;
    scanf("%d", &n);
    while (n--) {
        int op;
        scanf("%d", &op);
        switch(op) {
        case 1:
            int x;
            scanf("%d", &x);
            push(x);
            break;
        case 2:
            printf("%d\n", heap[1]);
            break;
        case 3:
            pop();
            break;
        default:
            break;
        }
    }
    return 0;
```

```
}
```

对顶堆

```cpp
#include <bits/stdc++.h>
using namespace std;
priority_queue<int> a; // 大根堆
priority_queue<int,vector<int>,greater<int>> b;
int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; ++i) {
        int x;
        cin >> x;
        if (b.empty() || x >= b.top()) b.push(x);
        else a.push(x);
        while (b.size() > k) a.push(b.top()), b.pop();
        while (b.size() < k) b.push(a.top()), a.pop();
        cout << b.top();
        b.pop();
    }
}
```

```cpp
#include <bits/stdc++.h>
using namespace std;
priority_queue<int> a; // 大根堆
priority_queue<int,vector<int>,greater<int>> b;
int main() {
    int n, w;
    cin >> n >> w;
    for (int i = 1; i <= n; ++i) {
        int x;
        cin >> x;
        if (b.empty() || x >= b.top()) b.push(x);
        else a.push(x);
        int k = max(1, i * w / 100);
        while (b.size() > k) a.push(b.top()), b.pop();
        while (b.size() < k) b.push(a.top()), a.pop();
        cout << b.top() << " ";
        //b.pop();
    }
}
```

并查集

路径压缩

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;
const int N=100010;
int n,m,x,y,z;
int fa[N];
int find(int x){
  if(fa[x]==x) return x;
  return fa[x]=find(fa[x]);
}
void unionset(int x,int y){
  fa[find(x)]=find(y);
}
int main(){
  cin>>n>>m;
  for(int i=1;i<=n;i++) fa[i]=i;
  while(m --){
    cin>>z>>x>>y;
    if(z==1) unionset(x,y);
    else{
      x=find(x),y=find(y);
      if(x==y) puts("Y");
      else puts("N");
    }
  }
  return 0;
}
```

路径压缩+按秩合并

```cpp
//路径压缩 + 按秩合并
#include <iostream>
#include <cstring>
#include <algorithm>
#include <vector>
using namespace std;
const int N=100010;
int n,m,x,y,z;
int fa[N];
int find(int x){
  if(fa[x]==x) return x;
  return fa[x]=find(fa[x]);
}
//记录并初始化子树的大小为1
vector<int>siz(N,1);
void unionset(int x,int y){
  x=find(x),y=find(y);
  if(x==y)return;
  if(siz[x]>siz[y])swap(x,y);
  fa[x]=y;
  siz[y]+=siz[x];
}
int main(){
  cin>>n>>m;
  for(int i=1;i<=n;i++) fa[i]=i;
  while(m --){
    cin>>z>>x>>y;
    if(z==1) unionset(x,y);
    else{
      x=find(x),y=find(y);
      if(x==y) puts("Y");
      else puts("N");
    }
  }
  return 0;
}
```

最小体力消耗路径

```cpp
class Solution {
public:
    struct Point {
        int x, y, v;
        Point(int x, int y, int v): x(x), y(y), v(v) {}
    };
    int minimumEffortPath(vector<vector<int>>& heights) {
        int m = heights.size();
        int n = heights[0].size();
        vector<Point> edges;
        for (int i = 0; i < m; ++i) {
            for (int j = 0; j < n; ++j) {
                int id = i * n + j;
                if (i > 0) edges.emplace_back(id - n, id, abs(heights[i][j] - heights[i - 1][j]));
                if (j > 0) edges.emplace_back(id - 1, id, abs(heights[i][j] - heights[i][j - 1]));
            }
        }
        sort(edges.begin(), edges.end(), [](const auto& e1, const auto& e2) {
            return e1.v < e2.v;
        });
        vector<int> par(m * n);
        iota(par.begin(), par.end(), 0);
        function<int(int)> find = [&](int x) -> int {
            return par[x] == x ? x : par[x] = find(par[x]);
        };
        function<void(int,int)> unite = [&](int x, int y) -> void {
            int fx = find(x);
            int fy = find(y);
            par[fx] = fy;
        };
        for (const auto& [x, y, v]: edges) {
            unite(x, y);
            if (find(0) == find(m * n - 1)) {
                return v;
            }
        }
        return 0;
    }
};

```

树状数组

树状数组求逆序对

```
#include <iostream>
#include <cstring>
#include <string>
#include <algorithm>
using namespace std;
const int N = 5e5 + 5;
#define int long long
struct Node {
    int vis, id;
    bool operator<(const Node& t) const {
        return vis < t.vis;
```

```cpp
    }
}a[N];
int b[N];
int rev[N];
int tree[N];
struct BIT {
    int n;
    BIT(const int& n = 0) : n(n) {
        memset(tree, 0, sizeof(tree));
    }
    int lowbit(int x) {
        return x & (-x);
    }
    void update(int x, int k) {
        for (; x <= n; x += lowbit(x)) {
            tree[x] += k;
        }
    }
    int query(int x) {
        int res = 0;
        for (; x > 0; x -= lowbit(x)) {
            res += tree[x];
        }
        return res;
    }
    int sum(int l, int r) { // [l, r)
        return query(r) - query(l);
    }
};
void solve() {
    int n;
    BIT bt(N);
    cin >> n;
    for (int i = 1; i <= n; ++i) cin >> a[i].vis, a[i].id = i;
    stable_sort(a + 1, a + 1 + n);
    for (int i = 1; i <= n; ++i) {
        b[a[i].id] = i;
    }
    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
        bt.update(b[i], 1);
        rev[i] = bt.query(b[i] - 1);
    }
    int res = 0;
    for (int i = 1; i <= n; ++i) {
        res += rev[i];
    }
    cout << res; // a_i > a_j
```

```
}
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    solve();
    return 0;
}
```

线段树

```
class Solution {
public:
    static constexpr long long N = 1e5 + 10;
    long long a[N];
    long long tree[N<<2], tag[N<<2];
    long long ls(long long p) {return p<<1;}
    long long rs(long long p) {return p<<1|1;}
    void push_up(long long p) {
        tree[p] = tree[ls(p)] + tree[rs(p)];
    }
    void build(long long p, long long pl, long long pr) {
        tag[p] = 0;
        if (pl == pr) {
            tree[p] = a[pl];
            return;
        }
        long long mid = pl + (pr - pl) / 2;
        build(ls(p), pl, mid);
        build(rs(p), mid+1,pr);
        push_up(p);
    }
    void add_tag(long long p, long long pl, long long pr, long long d) {
        tag[p] += d;
        tree[p] += d * (pr - pl + 1);
    }
    void push_down(long long p, long long pl, long long pr) {
        if (tag[p]) {
            long long mid = pl + (pr - pl) / 2;
            add_tag(ls(p), pl, mid, tag[p]);
            add_tag(rs(p), mid + 1, pr, tag[p]);
            tag[p] = 0;
        }
    }
    void update(long long l, long long r, long long p, long long pl, long long p
        if (l <= pl && pr <= r) {
            add_tag(p, pl, pr, d);
            return;
        }
        push_down(p, pl, pr);
        long long mid = pl + (pr - pl) / 2;
        if (l <= mid) update(l, r, ls(p), pl, mid, d);
        if (r > mid) update(l, r, rs(p), mid + 1, pr,d);
        push_up(p);
    }
    long long query(long long l, long long r, long long p, long long pl, long lo
        if (pl >= l && r >= pr) return tree[p];
        push_down(p,pl,pr);
        long long res = 0;
        long long mid = pl + (pr - pl) /2;
        if(l <= mid) res += query(l,r,ls(p),pl,mid);
        if (r > mid) res += query(l,r,rs(p),mid+1,pr);
        return res;
    }
    bool checkArray(vector<int>& nums, int k) {
        long long n = nums.size();
        for (long long i = 1; i <= n; ++i) a[i] = nums[i - 1];
        build(1, 1, n);
        for (long long i = 1; i <= n; ++i) {
            long long val = query(i,i,1,1,n);
            if (val < 0) {
                return false;
            }
            if (val == 0) continue;
            if (i + k - 1 <= n) {
                update(i, i + k - 1, 1, 1, n, -val);
            } else {
                return false;
            }
        }
        return true;
    }
};
```

平衡树

splay

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

```cpp
#define rs(x) tr[x].son[1]
const int MAX_N = 1e5 + 5, MAX_M = 1e6 + 5;
const int MAX_SIZE = MAX_N + MAX_M;
const int INF = (1 << 30) + 1;
struct Node {
    int son[2]; //左右儿子
    int parent; //父亲
    int val; //节点权值
    int cnt; //权值出现次数
    int siz; //子树大小
    void init(int parent_, int val_) {
        parent = parent_, val = val_;
        cnt = siz = 1;
    }
}tr[MAX_SIZE];
int root, idx; //根节点编号,节点个数
void pushup(int x) {//x 下标
    tr[x].siz = tr[ls(x)].siz + tr[rs(x)].siz + tr[x].cnt;
}
void rotate(int x) {//x 下标
    int y = tr[x].parent, z = tr[y].parent;
    int k = tr[y].son[1] == x;
    tr[z].son[tr[z].son[1] == y] = x;
    tr[x].parent = z;
    tr[y].son[k] = tr[x].son[k ^ 1];
    tr[tr[x].son[k ^ 1]].parent = y;
    tr[x].son[k ^ 1] = y;
    tr[y].parent = x;
    pushup(y), pushup(x);
}
void splay(int x, int k) {
    while (tr[x].parent != k) {
        int y = tr[x].parent, z = tr[y].parent;
        if (z != k) // 折转底, 直转中
            (ls(y) == x)^(ls(z) == y) ? rotate(x) : rotate(y);
        rotate(x);
    }
    if (!k) root = x;
}
 void insert(int v) { //插入数值v
     int x = root, p = 0;
     while (x && tr[x].val != v)
         p = x, x = tr[x].son[v > tr[x].val];
     if (x) tr[x].cnt++;
     else {
         x = ++idx;
         tr[p].son[v > tr[p].val] = x;
         tr[x].init(p, v);
     }
     splay(x, 0);
 }
void find(int v) { //找到数值v并转到根
     int x = root;
     while (tr[x].son[v > tr[x].val] && v != tr[x].val)
         x = tr[x].son[v > tr[x].val];
     splay(x, 0);
}
int get_pre(int v) { //数值v前驱的编号
    find(v);
    int x = root;
    if (tr[x].val < v) return x;
    x = ls(x);
    while (rs(x)) x = rs(x);
    splay(x, 0);// 艹,少一句代码都TLE,#5样例TLE#6样例PASS
    return x;
}
int get_suc(int v) { //数值v后继的编号
    find(v);
    int x = root;
    if (tr[x].val > v) return x;
    x = rs(x);
    while (ls(x)) x = ls(x);
    splay(x, 0);// #5样例PASS#6样例TLE
    return x;
}
void del(int v) { //数值v删除
    int pre = get_pre(v);
    int suc = get_suc(v);
    splay(pre, 0), splay(suc, pre);
    int del = tr[suc].son[0];
    if (tr[del].cnt > 1)
        tr[del].cnt--, splay(del, 0);
    else
        tr[suc].son[0] = 0, splay(suc, 0);
}

int get_rank(int v) { //数值v排名
//    这里老师的代码有问题
//    find(v);
//    return tr[tr[root].son[0]].siz;
    insert(v);
    int res = tr[tr[root].son[0]].siz;
    del(v);
    return res;
```

```cpp
}
int get_val_by_rank(int k) { //数值
    int x = root;
    while (1) {
        int y = ls(x);
        if (tr[y].siz + tr[x].cnt < k)
            k -= tr[y].siz + tr[x].cnt, x = rs(x);
        else if (tr[y].siz >= k) x = y;
        else break;
    }
    splay(x, 0);
    return tr[x].val;
}
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    insert(-INF);
    insert(INF); //哨兵
    int n, m;
    scanf("%d%d", &n, &m);
    int x;
    while (n--) {
        scanf("%d", &x);
        insert(x);
    }
    int op, res = 0, last = 0;
    while (m--) {
        scanf("%d%d", &op, &x);
        x ^= last;
        if (op == 1) insert(x);
        if (op == 2) del(x);
        if (op == 3) res ^= (last = get_rank(x));
        if (op == 4) res ^= (last = get_val_by_rank(x + 1));
        if (op == 5) res ^= (last = tr[get_pre(x)].val);
        if (op == 6) res ^= (last = tr[get_suc(x)].val);
    }
    cout << res << '\n';
    return 0;
 }
```

主席树

查询区间第k大

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

```cpp
using namespace std ;
const int N = 200010;
int cnt = 0;
int a[N], b[N], root[N];
struct{
int L, R, sum;
}tree[N<<5];
int build(int pl, int pr){
    int rt = ++ cnt;
    tree[rt].sum = 0;
    int mid=(pl+pr)>>1;
    if (pl < pr){
        tree[rt].L = build(pl, mid);
        tree[rt].R = build(mid+1, pr);
    }
    return rt;
}
int update(int pre, int pl, int pr, int x){
    int rt = ++cnt;
    tree[rt].L = tree[pre].L;
    tree[rt].R = tree[pre].R;
    tree[rt].sum = tree[pre].sum + 1;
    int mid = (pl+pr)>>1;
    if (pl < pr){
        if (x <= mid)
            tree[rt].L = update(tree[pre].L, pl, mid, x);
        else
            tree[rt].R = update(tree[pre].R, mid+1, pr, x);
    }
    return rt;
}
int query(int u, int v, int pl, int pr, int k){
    if (pl == pr) return pl;
    int x = tree[tree[v].L].sum - tree[tree[u].L].sum;
    int mid = (pl+pr)>>1;
    if (x >= k)
        return query(tree[u].L, tree[v].L, pl, mid, k);
    else
        return query(tree[u].R, tree[v].R, mid+1, pr, k-x);
}
int main(){
    int n;    scanf("%d", &n);
    for (int i=1; i<=n; i++){ scanf("%d", &a[i]);  b[i]=a[i]; }
    sort(b+1, b+1+n);
    int size = unique(b+1, b+1+n)-b-1;
    for (int i = 1; i <= n; i ++){
        int x = lower_bound(b+1, b+1+size, a[i]) - b;
        root[i] = update(root[i-1], 1, size, x);
    }
    int m;    scanf("%d", &m);
    while (m--){
        int x, y, k;      scanf("%d%d%d", &x, &y,&k);
        int t = query(root[x-1], root[y], 1, size, k);
        printf("%d\n", b[t]);
    }
    return 0;
}
```

查询区间小于或等于k的数字个数

```
#include<iostream>
#include<cstring>
#include<algorithm>
#include<queue>
#include<map>
#include<stack>
#include<cmath>
#include<vector>
#include<set>
#include<cstdio>
#include<string>
#include<deque>
using namespace std;
typedef long long LL;
#define eps 1e-8
#define INF 0x3f3f3f3f
#define maxn 100005
struct node{
    int l,r,sum;
}tree[maxn*25];
int n,m,k,t,cnt;
struct point{
    int id,w;
}a[maxn];
int b[maxn],rt[maxn];
bool operator <(point s1,point s2){
    if(s1.w!=s2.w)
    return s1.w<s2.w;
    else
    return s1.id<s2.id;
}
void update(int root){
    tree[root].sum=tree[tree[root].l].sum+tree[tree[root].r].sum;
}
void build_0(int &root,int l,int r){
    root=++cnt;
    tree[root].l=l;
    tree[root].r=r;
    tree[root].sum=0;
    if(l==r)
    return;
    int mid=(l+r)/2;
    build_0(tree[root].l,l,mid);
    build_0(tree[root].r,mid+1,r);
    update(root);
}
void build(int pre,int &root,int l,int r,int index){
    root=++cnt;
    tree[root]=tree[pre];
    if(l==r){
        tree[root].sum++;
        return;
    }
    int mid=(l+r)/2;
    if(index<=mid)
    build(tree[pre].l,tree[root].l,l,mid,index);
    else
    build(tree[pre].r,tree[root].r,mid+1,r,index);
    update(root);
}
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
    while(l<=r){
        int mid=(l+r)/2;
        if(a[mid].w>k)
        r=mid-1;
        else
        l=mid+1;
    }
    return r;
}
int ask(int root1,int root2,int L,int R,int l,int r){
    if(L>R)
    return 0;
    if(l>=L&&r<=R){
        return tree[root2].sum-tree[root1].sum;
    }
    int mid=(l+r)/2;
    int ans=0;
    if(mid>=L)
    ans+=ask(tree[root1].l,tree[root2].l,L,R,l,mid);
    if(mid<R)
    ans+=ask(tree[root1].r,tree[root2].r,L,R,mid+1,r);
    return ans;
}
int main()
{
    scanf("%d",&t);
    int Case=0;
    while(t--){
        scanf("%d%d",&n,&m);
        cnt=0;
        for(int i=1;i<=n;i++){
            scanf("%d",&a[i].w);
            a[i].id=i;
        }
        //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
        sort(a+1,a+n+1);
        for(int i=1;i<=n;i++){
            b[a[i].id]=i;
        }
        build_0(rt[0],1,n);//建第0颗树
        for(int i=1;i<=n;i++)
        build(rt[i-1],rt[i],1,n,b[i]);//建第i颗树
        int l,r,k;
        printf("Case %d:\n",++Case);
        while(m--){
            scanf("%d%d%d",&l,&r,&k);
            l++;
            r++;
            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
            printf("%d\n",ask(rt[l-1],rt[r],1,index,1,n));
        }
    }
    return 0;
}
```

ST表

一维RMQ问题

```
#include <iostream>
#include <algorithm>
#include <cstring>
#include <cmath>
```

```cpp
using namespace std;
const int N = 1e6 + 5;
int n, k, q;
int a[N], dp_max[N][30];
void st_init() {
    for (int i = 1; i <= n; ++i) {
        dp_max[i][0] = a[i];
    }
    int p = (int)(log(double(n)) / log(2.0));
    for (int k = 1; k <= p; ++k) {
        for (int s = 1; s + (1 << k) <= n + 1; ++s) {
            dp_max[s][k] = std::max(dp_max[s][k - 1], dp_max[s + (1 << (k - 1))][k - 1]);
        }
    }
}
int st_query_max(int L, int R) {
    int k = (int)(log(double(R - L + 1)) / log(2.0));
    return std::max(dp_max[L][k], dp_max[R - (1 << k) + 1][k]);
}
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    cin >> n >> q;
    for (int i = 1; i <= n; ++i) {
        cin >> a[i];
    }
    st_init();
    while (q--) {
        int l, r;
        cin >> l >> r;
        cout << st_query_max(l, r) << '\n';
    }
}
```

莫队

普通莫队

HH项链

```cpp
#include <cstdio>
#include <cstring>
#include <cmath>
```

```
#include <algorithm>
using namespace std;
#define maxn 1010000
#define maxb 1010
int aa[maxn], cnt[maxn], belong[maxn];
int n, m, size, bnum, now, ans[maxn];
struct query {
        int l, r, id;
} q[maxn];
int cmp(query a, query b) {
    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);
}
#define isdigit(x) ((x) >= '0' && (x) <= '9')
int read() {
    int res = 0;
    char c = getchar();
    while(!isdigit(c)) c = getchar();
    while(isdigit(c)) res = (res << 1) + (res << 3) + c - 48, c = getchar();
    return res;
}
void printi(int x) {
    if(x >= 10) printi(x / 10);
    putchar(x % 10 + '0');
}
int main() {
    scanf("%d", &n);
    size = sqrt(n);
    bnum = ceil((double)n / size);
    for(int i = 1; i <= bnum; ++i)
        for(int j = (i - 1) * size + 1; j <= i * size; ++j) {
            belong[j] = i;
        }
    for(int i = 1; i <= n; ++i) aa[i] = read();
    m = read();
    for(int i = 1; i <= m; ++i) {
        q[i].l = read(), q[i].r = read();
        q[i].id = i;
    }
    sort(q + 1, q + m + 1, cmp);
    int l = 1, r = 0;
    for(int i = 1; i <= m; ++i) {
        int ql = q[i].l, qr = q[i].r;
        while(l < ql) now -= !--cnt[aa[l++]];
        while(l > ql) now += !cnt[aa[--l]]++;
        while(r < qr) now += !cnt[aa[++r]]++;
        while(r > qr) now -= !--cnt[aa[r--]];
        ans[q[i].id] = now;
    }
    for(int i = 1; i <= m; ++i) printi(ans[i]), putchar('\n');
    return 0;
}
```

带修莫队

数颜色

```
#include <bits/stdc++.h>
using namespace std;
#define maxn 150500
#define maxc 3001000
int a[maxn], cnt[maxc], ans[maxn], belong[maxn];
struct query {
    int l, r, time, id;
} q[maxn];
struct modify {
    int pos, color, last;
} c[maxn];
int cntq, cntc, n, m, size, bnum;
int cmp(query a, query b) {
    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);
}
#define isdigit(x) ((x) >= '0' && (x) <= '9')
inline int read() {
    int res = 0;
    char c = getchar();
    while(!isdigit(c)) c = getchar();
    while(isdigit(c)) res = (res << 1) + (res << 3) + (c ^ 48), c = getchar();
    return res;
}
int main() {
    n = read(), m = read();
    size = pow(n, 2.0 / 3.0);
    bnum = ceil((double)n / size);
    for(int i = 1; i <= bnum; ++i)
        for(int j = (i - 1) * size + 1; j <= i * size; ++j) belong[j] = i;
    for(int i = 1; i <= n; ++i)
        a[i] = read();
    for(int i = 1; i <= m; ++i) {
        char opt[100];
        scanf("%s", opt);
        if(opt[0] == 'Q') {
            q[++cntq].l = read();
            q[cntq].r = read();
            q[cntq].time = cntc;
            q[cntq].id = cntq;
        }
        else if(opt[0] == 'R') {
            c[++cntc].pos = read();
            c[cntc].color = read();
        }
    }
    sort(q + 1, q + cntq + 1, cmp);
    int l = 1, r = 0, time = 0, now = 0;
    for(int i = 1; i <= cntq; ++i) {
        int ql = q[i].l, qr = q[i].r, qt = q[i].time;
        while(l < ql) now -= !--cnt[a[l++]];
        while(l > ql) now += !cnt[a[--l]]++;
        while(r < qr) now += !cnt[a[++r]]++;
        while(r > qr) now -= !--cnt[a[r--]];
        while(time < qt) {
            ++time;
            if(ql <= c[time].pos && c[time].pos <= qr) now -= !--cnt[a[c[time].pos]] - !cnt[c[time].color]++;
            swap(a[c[time].pos], c[time].color);
        }
        while(time > qt) {
            if(ql <= c[time].pos && c[time].pos <= qr) now -= !--cnt[a[c[time].pos]] - !cnt[c[time].color]++;
            swap(a[c[time].pos], c[time].color);
            --time;
        }
        ans[q[i].id] = now;
    }
    for(int i = 1; i <= cntq; ++i)
            printf("%d\n", ans[i]);
    return 0;
}
```

树上莫队

```

```

回滚莫队

```

```

四、图论

多源BFS

```cpp
static constexpr int dirs[4][2] = {{-1,0}, {1, 0}, {0, -1}, {0, 1}};
int maximumSafenessFactor(vector<vector<int>>& grid) {
    int n = grid.size();
    vector<pair<int,int>> q;
    vector<vector<int>> dis(n, vector<int>(n, -1));
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (grid[i][j]) q.emplace_back(i,j), dis[i][j] = 0;
        }
    }
    vector<vector<pair<int,int>>> groups = {q};
    while (!q.empty()) {
        vector<pair<int,int>> nq;
        for (auto& [i,j]: q) {
            for (auto& d: dirs) {
                int x = i + d[0], y = j + d[1];
                if (x >= 0 && x < n && y >= 0 && y < n && dis[x][y] < 0) {
                    nq.emplace_back(x, y);
                    dis[x][y] = groups.size();
                }
            }
        }
        groups.push_back(nq);
        q = move(nq);
    }
}
```

拓扑排序

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
#include <queue>
using namespace std;
```

```cpp
const int N = 100010;
int n,m,a,b;
vector<int> e[N], tp;
int din[N];

bool toposort(){
  queue<int> q;
  for(int i = 1; i <= n; i++)
    if(din[i]==0) q.push(i);
  while(q.size()){
    int x=q.front(); q.pop();
    tp.push_back(x);
    for(auto y : e[x]){
      if(--din[y]==0) q.push(y);
    }
  }
  return tp.size() == n;
}
int main(){
  cin >> n >> m;
  for(int i=0; i<m; i++){
    cin >> a >> b;
    e[a].push_back(b);
    din[b]++;
  }
  if(!toposort()) puts("-1");
  else for(auto x:tp) printf("%d ",x);
  return 0;
}
```

```cpp
#include <bits/stdc++.h>
using namespace std;
using i64 = long long;
void solve() {
    int n, m;
    cin >> n >> m;
    vector<vector<int>> adj(n);
    vector<int> deg(n);
    for (int i = 0; i < m; ++i) {
        int u, v;
        cin >> u >> v;
        u--, v--;
        adj[u].push_back(v);
        deg[v]++;
    }
    vector<int> q;
    for (int i = 0; i < n; ++i) {
        if (!deg[i]) {
            q.push_back(i);
        }
    }
    for (int i = 0; i < q.size(); ++i) {
        int x = q[i];
        for (auto y: adj[x]) {
            if (!--deg[y]) {
                q.push_back(y);
            }
        }
    }
    if (q.size() == n) {
        cout << 1 << '\n';
        for (int i = 0; i < n; ++i) {
            cout << q[i] + 1 << " \n"[i == n - 1];
        }
        return;
    }
    cout << 2 << '\n';
    for (int i = 1; i <= n; ++i) {
        cout << i << " \n"[i == n];
    }
    for (int i = 1; i <= n; ++i) {
        cout << n + 1 - i << " \n"[i == n];
    }
}
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    solve();
}
```

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
#include <queue>
using namespace std;

const int N = 100010;
int n,m,a,b;
vector<int> e[N], tp;
int c[N]; //染色数组bool dfs(int x){
  c[x] = -1;
  for(int y : e[x]){
    if(c[y]<0)return 0; //有环 else if(!c[y])
      if(!dfs(y))return 0;
  }
  c[x] = 1;
  tp.push_back(x);
  return 1;
}
bool toposort(){
  memset(c, 0, sizeof(c));
  for(int x = 1; x <= n; x++)
    if(!c[x])
      if(!dfs(x))return 0;
  reverse(tp.begin(),tp.end());
  return 1;
}
int main(){
  cin >> n >> m;
  for(int i=0; i<m; i++){
    cin >> a >> b;
    e[a].push_back(b);
  }
  if(!toposort()) puts("-1");
  else
    for(int x:tp) printf("%d ",x);
  return 0;
}
```

最短路

Dijkstra算法

```
#include <cstdio>
#include <vector>
#include <cstring>
#include <climits>
using namespace std;
const int N = 1e5 + 5;
#define int long long
struct edge {
    int v, w;
    edge(): v(0), w(0) {}
    edge(int v, int w): v(v), w(w) {}
};
```

```cpp
int n, m, s;
vector<edge> e[N];
int dis[N];
int vis[N];
void dijkstra() {
    for (int i = 0; i <= n; ++i) dis[i] = INT_MAX;
    dis[s] = 0;
    for (int i = 1; i < n; ++i) {
        int u = 0;
        for (int j = 1; j <= n; ++j) {
            if (!vis[j] && dis[j] < dis[u]) {
                u = j;
            }
        }
        vis[u] = 1;
        for (auto& ed: e[u]) {
            int v = ed.v;
            int w = ed.w;
            if (dis[v] > dis[u] + w) {
                dis[v] = dis[u] + w;
            }
        }
    }
}
signed main() {
    scanf("%lld%lld%lld",&n,&m,&s);
    for (int i = 0; i < m; ++i) {
        int u, v, w;
        scanf("%lld%lld%lld",&u,&v,&w);
        e[u].push_back({v, w});
    }
    dijkstra();
    for (int i = 1; i <= n; ++i) {
        printf("%lld ", dis[i]);
    }
    return 0;
}
```

```cpp
#include <cstdio>
#include <cstring>
#include <vector>
#include <queue>
#include <climits>
using namespace std;
struct edge {
    int v, w;
    edge(): v(0), w(0) {}
    edge(int v, int w): v(v), w(w) {}
};
struct node {
    int dis, u;
    bool operator>(const node& n) const { return dis > n.dis; }
};
const int N = 1e5 + 5;
vector<edge> e[N];
int dis[N];
int vis[N];
priority_queue<node, vector<node>, greater<node>> q;
int n, m, s;
void dijkstra() {
    for (int i = 0; i <= n; ++i) dis[i] = INT_MAX;
    dis[s] = 0;
    q.push({0,s});
    while (!q.empty()) {
        int u = q.top().u;
        q.pop();
        if (vis[u]) continue;
        vis[u] = 1;
        for (auto ed: e[u]) {
            int v = ed.v;
            int w = ed.w;
            if (dis[v] > dis[u] + w) {
                dis[v] = dis[u] + w;
                q.push({dis[v], v});
            }
        }
    }
}
int main() {
    scanf("%d%d%d",&n,&m,&s);
    for (int i = 0; i < m; ++i) {
        int u, v, w;
        scanf("%d%d%d",&u,&v,&w);
        e[u].push_back({v, w});
    }
    dijkstra();
    for (int i = 1; i <= n; ++i) {
        printf("%d ", dis[i]);
    }
    return 0;
}
```

Bellman-Ford算法 SPFA算法

```

```

负环

Floyd

```
//Ford 判负环 740ms
#include <cstring>
#include <iostream>
#include <algorithm>
using namespace std;
const int inf=0x3f3f3f3f;
const int N=2010,M=6010;
int n,m;
int to[M],ne[M],w[M],h[N],tot;
int d[N];
void add(int a,int b,int c){
  to[++tot]=b;w[tot]=c;
  ne[tot]=h[a];h[a]=tot;
}
bool ford(){
  memset(d,inf,sizeof d); d[1]=0;
  bool flag; //是否松弛
  for(int i=1;i<=n;i++){ //跑n轮
    flag=false;
    for(int u=1;u<=n;u++){ //n个点
      if(d[u]==inf)continue;
      for(int j=h[u];j;j=ne[j]){
        int v=to[j];
        if(d[v]>d[u]+w[j]){
          d[v]=d[u]+w[j];
          flag=true;
        }
      }
    }
    if(!flag)break;
  }
  return flag; //第n轮=true,有负环
}
int main(){
  int T;
  scanf("%d",&T);
```

```cpp
  while(T--){
    tot=0; memset(h,0,sizeof(h));
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++){
      int u,v,w;
      scanf("%d%d%d",&u,&v,&w);
      add(u,v,w);
      if(w>=0)add(v,u,w);;
    }
    puts(ford()?"YES":"NO");
  }
  return 0;
}
```

```cpp
//BFS_spfa 判负环 530ms
#include <iostream>
#include <cstring>
#include <algorithm>
#include <queue>
using namespace std;
const int inf=0x3f3f3f3f;
const int N=2010,M=6010;
int n,m;
int to[M],ne[M],w[M],h[N],tot;
int d[N],cnt[N],vis[N];
void add(int a,int b,int c){
  to[++tot]=b;w[tot]=c;
  ne[tot]=h[a];h[a]=tot;
}
bool spfa(){ //判负环
  memset(d,0x3f,sizeof d);
  memset(vis,0,sizeof vis);
  memset(cnt,0,sizeof cnt);
  queue<int>q;
  q.push(1); vis[1]=1; d[1]=0;
  while(q.size()){
    int u=q.front();q.pop();vis[u]=0;
    for(int i=h[u];i;i=ne[i]){
      int v=to[i];
      if(d[v]>d[u]+w[i]){
        d[v]=d[u]+w[i];
        cnt[v]=cnt[u]+1;
        if(cnt[v]>=n)return 1;//判边数
        if(!vis[v])q.push(v),vis[v]=1;
      }
    }
  }
  return 0;
}
int main(){
  int T; scanf("%d",&T);
  while(T--){
    tot=0; memset(h,0,sizeof(h));
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++){
      int u,v,w;
      scanf("%d%d%d",&u,&v,&w);
      add(u,v,w);
      if(w>=0)add(v,u,w);;
    }
    puts(spfa()?"YES":"NO");
  }
  return 0;
}
```

```cpp
//BFS_spfa 判负环 690ms
#include <iostream>
#include <cstring>
#include <algorithm>
#include <queue>
using namespace std;
const int inf=0x3f3f3f3f;
const int N=2010,M=6010;
int n,m;
int to[M],ne[M],w[M],h[N],tot;
int d[N],cnt[N],vis[N];
void add(int a,int b,int c){
  to[++tot]=b;w[tot]=c;
  ne[tot]=h[a];h[a]=tot;
}
bool spfa(){ //判负环
  memset(d,0x3f,sizeof d);
  memset(vis,0,sizeof vis);
  memset(cnt,0,sizeof cnt);
  queue<int>q;
  q.push(1); vis[1]=1; d[1]=0;
  while(q.size()){
    int u=q.front();q.pop();vis[u]=0;
    for(int i=h[u];i;i=ne[i]){
      int v=to[i];
      if(d[v]>d[u]+w[i]){
        d[v]=d[u]+w[i];
        if(++cnt[v]>n)return 1;//判点数
        if(!vis[v])q.push(v),vis[v]=1;
      }
    }
  }
  return 0;
}
int main(){
  int T; scanf("%d",&T);
  while(T--){
    tot=0; memset(h,0,sizeof(h));
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++){
      int u,v,w;
      scanf("%d%d%d",&u,&v,&w);
      add(u,v,w);
      if(w>=0)add(v,u,w);;
    }
    puts(spfa()?"YES":"NO");
  }
  return 0;
}
```

```cpp
//DFS_spfa 判负环 会卡点 #9
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;
const int inf=0x3f3f3f3f;
const int N=2010,M=6010;
int n,m;
int to[M],ne[M],w[M],h[N],tot;
int d[N],vis[N];
void add(int a,int b,int c){
  to[++tot]=b;w[tot]=c;
  ne[tot]=h[a];h[a]=tot;
}
bool spfa(int u){ //判负环
  vis[u]=1;
  for(int i=h[u];i;i=ne[i]){
    int v=to[i];
    if(d[v]>d[u]+w[i]){
      d[v]=d[u]+w[i];
      if(vis[v]||spfa(v))return 1;
    }
  }
  vis[u]=0;
  return 0;
}
int main(){
  int T; scanf("%d",&T);
  while(T--){
    tot=0; memset(h,0,sizeof(h));
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++){
      int u,v,w;
      scanf("%d%d%d",&u,&v,&w);
      add(u,v,w);
      if(w>=0)add(v,u,w);;
    }
    memset(d,0x3f,sizeof d);d[1]=0;
    memset(vis,0,sizeof vis);
    puts(spfa(1)?"YES":"NO");
  }
  return 0;
}
```

最小环

无向图

```
#include <bits/stdc++.h>
using namespace std;
const int N = 105;
int n, m, a, b, c;
typedef long long ll;
const ll inf = 1e13;
ll res = inf;
ll w[N][N];
ll d[N][N];
int main() {
    cin >> n >> m;
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= n; ++j) {
            if (i != j) w[i][j] = inf;
        }
    }
    for (int i = 1; i <= m; ++i) {
        cin >> a >> b >> c;
        w[a][b] = w[b][a] = c;
    }
    memcpy(d, w, sizeof(d));
    for (int k = 1; k <= n; ++k) {
        for (int i = 1; i < k; ++i) {
            for (int j = i + 1; j < k; ++j) {
                res = min(res, d[i][j] + w[j][k] + w[k][i]);
            }
        }
        for (int i = 1; i <= n; ++i) {
            for (int j = 1; j <= n; ++j) {
                d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
            }
        }
    }
    if (res == inf) cout << "No solution.";
    else cout << res;
    return 0;
}
```

最长路

k短路

```
#include <cstdio>
#include <iostream>
#include <cstring>
#include <vector>
#include <queue>
using namespace std;
const int N=1010,M=200010;
int h[N],rh[N],to[M],w[M],ne[M],tot;
void add(int h[],int a,int b,int c){
  to[++tot]=b;w[tot]=c;
  ne[tot]=h[a],h[a]=tot;
}
int n,m,S,T,K;
int f[N],vis[N],cnt[N];
struct node{
  int s,v,d; //s排序，v点，d距离
  bool operator<(const node &x)const
    {return s>x.s;}
```

```
};
void dijkstra(){
  memset(f,0x3f,sizeof f); f[T]=0;
  priority_queue<pair<int,int>> q;
  q.push(make_pair(0,T));
  while(q.size()){
    pair<int,int> t=q.top(); q.pop();
    int u=t.second;
    if(vis[u])continue;
    vis[u]=true; //第一次出队时最小
    for(int i=rh[u]; i; i=ne[i]){
      int v=to[i];
      if(f[v]>f[u]+w[i]){
        f[v]=f[u]+w[i]; //估价函数
        q.push(make_pair(-f[v],v));
      }
    }
  }
}
int aStar(){
  priority_queue<node> q; //优先队列
  node a={f[S],S,0}; q.push(a);
  while(q.size()){
    node t=q.top(); q.pop();
    int u=t.v;
    cnt[u]++; //记录出队次数
    if(cnt[T]==K) return t.d; //边界
    for(int i=h[u]; i; i=ne[i]){
      int v=to[i], d=t.d+w[i];
      if(cnt[v]<K){
        node a={d+f[v],v,d};
        q.push(a);
      }
    }
  }
  return -1;
}
int main(){
  scanf("%d%d",&n,&m);
  for(int i=1; i<=m; i++){
    int a,b,c;
    scanf("%d%d%d",&a,&b,&c);
    add(h,a,b,c); add(rh,b,a,c); //反图
  }
  scanf("%d%d%d",&S,&T,&K);
  if(S==T) K++; //重合点，0是第一条
  dijkstra();
  printf("%d\n",aStar());
}
```

欧拉回路

```
#include<bits/stdc++.h>
using namespace std;
const int N = 55;
int degree[N];  //记录度
int G[N][N];    //存图
void euler(int u){                    //从u开始DFS
    for(int v = 1; v <= 50; v++)  {  //v是u的邻居
        if(G[u][v]) {
            G[u][v]--;
            G[v][u]--;
            euler(v);
            cout << v << " " << u << endl;    //在euler()后打印，即回溯时打印
        }
    }
}
int main(){
    int t; cin >> t;
    int cnt = 0;
    while (t--) {
        cnt++;
        if(cnt != 1) cout << endl;
        cout << "Case #" << cnt << endl;
        memset(degree, 0, sizeof(degree));
        memset(G, 0, sizeof(G));
        int n;  cin >> n;
        int color;
        for(int i = 0; i < n; i++) {    //输入n条边
            int u, v;  cin>>u>>v;
            color = u;                //记录一种颜色。测试的时候可能只出现某些颜色
            degree[u]++;
            degree[v]++;              //记录点的度
            G[u][v]++;
            G[v][u]++;                //存图：=0不连接，=1连接，>1有重边
        }
        int ok = 1;
        for(int i = 1; i <= 50; i++)
            if(degree[i] % 2) {       //存在奇点，无欧拉路
                cout<<"some beads may be lost"<<endl;
                ok = 0;
                break;
```

```c
        }
        if(ok)  euler(color);         //有欧拉路。随便从某个存在的颜色开始
    }
    return 0;
}
```

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

最小生成树

Kruskal 适合稀疏图

```cpp
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
const int N = 5005;
const int M = 2e5 + 5;
struct Edge {
    int u, v, w;
}e[M];
int s[N];
int find(int x) {
    if (x != s[x]) s[x] = find(s[x]);
    return s[x];
}
int n, m;
void kruskal() {
    sort(e + 1, e + m + 1, [](const Edge& e1, const Edge& e2) { return e1.w < e2
    for (int i = 1; i <= n; ++i) {
        s[i] = i;
    }
    int ans = 0, cnt = 0;
    for (int i = 1; i <= m; ++i) {
        if (cnt == n - 1) break;
        int e1 = find(e[i].u);
        int e2 = find(e[i].v);
        if (e1 == e2) continue;
        else {
            ans += e[i].w;
            s[e1] = e2;
            ++cnt;
        }
    }
    if (cnt == n - 1) printf("%d\n", ans);
    else printf("orz\n");
}
int main() {
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= m; ++i) {
        scanf("%d%d%d", &e[i].u, &e[i].v, &e[i].w);
    }
    kruskal();
}
```

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

```
// 2.0s
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;
const int N=5e5+10,M=2*N;
int n,m,s,a,b;
int dep[N],fa[N][22];
int h[N],to[M],ne[M],tot;
void add(int a, int b){
  to[++tot]=b,ne[tot]=h[a],h[a]=tot;
}
void dfs(int x, int f){
  dep[x]=dep[f]+1; fa[x][0]=f;
  for(int i=0; i<=20; i++)
    fa[x][i+1]=fa[fa[x][i]][i];
  for(int i=h[x]; i; i=ne[i])
    if(to[i]!=f) dfs(to[i], x);
}
int lca(int x, int y){
  if(dep[x]<dep[y]) swap(x, y);
  for(int i=20; ~i; i--)
    if(dep[fa[x][i]]>=dep[y]) x=fa[x][i];
  if(x==y) return y;

  for(int i=20; ~i; i--)
    if(fa[x][i]!=fa[y][i]) x=fa[x][i],y=fa[y][i];
  return fa[x][0];
}
int main(){
  scanf("%d%d%d", &n,&m,&s);
  for(int i=1; i<n; i++){
    scanf("%d%d",&a,&b);
    add(a,b); add(b,a);
  }
  dfs(s, 0);
  while(m--){
    scanf("%d%d", &a, &b);
    printf("%d\n",lca(a, b));
  }
  return 0;
}
```

离线Tarjan算法

```
#include <iostream>
#include <algorithm>
#include <cstring>
#include <vector>
using namespace std;
const int N=500005,M=2*N;
int n,m,s,a,b;
vector<int> e[N];
vector<pair<int,int>>query[N];
int fa[N],vis[N],ans[M];
int find(int x){
  if(x==fa[x]) return x;
  return fa[x]=find(fa[x]);
}
void tarjan(int x){
  vis[x]=true;//标记x已访问
  for(auto y : e[x]){
    if(!vis[y]){
      tarjan(y);
      fa[y]=x;//回到x时指向x
     }
  }
  //离开x时找LCA
  for(auto q : query[x]){
    int y=q.first,i=q.second;
    if(vis[y])ans[i]=find(y);
  }
}
int main(){
  scanf("%d%d%d", &n,&m,&s);
  for(int i=1; i<n; i++){
    scanf("%d%d",&a,&b);
    e[a].push_back(b);
    e[b].push_back(a);
  }
  for(int i=1;i<=m;i++){
    scanf("%d%d",&a,&b);
    query[a].push_back({b,i});
    query[b].push_back({a,i});
  }
  for(int i=1;i<=N;i++)fa[i]=i;
  tarjan(s);
  for(int i=1; i<=m; i++)
    printf("%d\n",ans[i]);
  return 0;
}
```

树链剖分

```
// 1.6s
#include <iostream>
#include <cstring>
#include <algorithm>
#include <vector>
using namespace std;
const int N=500010;
int n,m,s,a,b;
vector<int> e[N];
int fa[N],son[N],dep[N],siz[N];
int top[N];
void dfs1(int u,int f){ //搞fa,dep,son
  fa[u]=f;siz[u]=1;dep[u]=dep[f]+1;
  for(int v:e[u]){
    if(v==f) continue;
    dfs1(v,u);
    siz[u]+=siz[v];
    if(siz[son[u]]<siz[v])son[u]=v;
  }
}
void dfs2(int u,int t){ //搞top
  top[u]=t;  //记录链头
  if(!son[u]) return; //无重儿子
  dfs2(son[u],t);     //搜重儿子
  for(int v:e[u]){
    if(v==fa[u]||v==son[u])continue;
    dfs2(v,v); //搜轻儿子
  }
}
int lca(int u,int v){
  while(top[u]!=top[v]){
    if(dep[top[u]]<dep[top[v]])swap(u,v);
    u=fa[top[u]];
  }
  return dep[u]<dep[v]?u:v;
}
int main(){
  scanf("%d%d%d",&n,&m,&s);
  for(int i=1; i<n; i++){
    scanf("%d%d",&a,&b);
    e[a].push_back(b);
    e[b].push_back(a);
  }
  dfs1(s,0);
  dfs2(s,s);
  while(m--){
    scanf("%d%d",&a,&b);
    printf("%d\n",lca(a,b));
  }
  return 0;
}
```

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

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

```cpp
void dfs(int x, int fa) {
    siz[x] = 1;
    for(int i = 0; i < v[x].size(); i++) {
        int to = v[x][i];
        if(to == fa) continue;
        dfs(to, x);
        siz[x] += siz[to];
        if(siz[to] > siz[son[x]]) son[x] = to;//轻重链剖分
    }
}
void add(int x, int fa, int val) {
    cnt[col[x]] += val;//这里可能会因题目而异
    if(cnt[col[x]] > Mx) Mx = cnt[col[x]], sum = col[x];
    else if(cnt[col[x]] == Mx) sum += (LL)col[x];
    for(int i = 0; i < v[x].size(); i++) {
        int to = v[x][i];
        if(to == fa || to == Son) continue;
        add(to, x, val);
    }
}
void dfs2(int x, int fa, int opt) {
    for(int i = 0; i < v[x].size(); i++) {
        int to = v[x][i];
        if(to == fa) continue;
        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
    }
    if(son[x]) dfs2(son[x], x, 1), Son = son[x];//统计重儿子的贡献，不消除影响

    add(x, fa, 1); Son = 0;//暴力统计所有轻儿子的贡献
    ans[x] = sum;//更新答案
    if(!opt) add(x, fa, -1), sum = 0, Mx = 0;//如果需要删除贡献的话就删掉
}
int main() {
    N = read();
    for(int i = 1; i <= N; i++) col[i] = read();
    for(int i = 1; i <= N - 1; i++) {
        int x = read(), y = read();
        v[x].push_back(y); v[y].push_back(x);
    }
    dfs(1, 0);
    dfs2(1, 0, 0);
    for(int i = 1; i <= N; i++) printf("%I64d ", ans[i]);
    return 0;
}
```

有向图的连通性

Kosaraju算法

```cpp
#include<bits/stdc++.h>
using namespace std;
const int N = 10005;
vector<int> G[N], rG[N];
vector<int> S;                  //存第一次dfs1的结果：标记点的先后顺序
int vis[N], sccno[N], cnt;      // cnt：强连通分量的个数
void dfs1(int u) {
    if(vis[u]) return;
    vis[u] = 1;
    for(int i=0; i<G[u].size(); i++)   dfs1(G[u][i]);
    S.push_back(u);             //记录点的先后顺序，标记大的放在S的后面
}
void dfs2(int u) {
    if(sccno[u]) return;
    sccno[u] = cnt;
    for(int i=0; i < rG[u].size(); i++)   dfs2(rG[u][i]);
}
void Kosaraju(int n) {
   cnt = 0;
   S.clear();
   memset(sccno, 0, sizeof(sccno));
   memset(vis, 0, sizeof(vis));
   for(int i = 1; i <= n; i++)  dfs1(i);   //点的编号：1~n。递归所有点
   for(int i = n-1; i >= 0; i--)
       if(!sccno[S[i]]) { cnt++; dfs2(S[i]);}
}
int main(){
    int n, m, u, v;
    while(scanf("%d%d", &n, &m), n != 0 || m != 0) {
        for(int i = 0; i < n; i++) { G[i].clear(); rG[i].clear();}
        for(int i = 0; i < m; i++){
            scanf("%d%d", &u, &v);
            G[u].push_back(v);     //原图
            rG[v].push_back(u);    //反图
        }
        Kosaraju(n);
        printf("%s\n", cnt == 1 ? "Yes" : "No");
    }
    return 0;
}
```

Tarjan算法

```cpp
#include<bits/stdc++.h>
using namespace std;
const int N = 10005;
int cnt;                                    // 强连通分量的个数
int low[N], num[N], dfn;
int sccno[N], stack[N], top;               // 用stack[]处理栈，top是栈顶
vector<int> G[N];
void dfs(int u){
    stack[top++] = u;                       //u进栈
    low[u]= num[u]= ++dfn;
    for(int i=0; i<G[u].size(); ++i){
        int v = G[u][i];
        if(!num[v]){                        //未访问过的点，继续dfs
            dfs(v);                         //dfs的最底层，是最后一个SCC
            low[u]= min( low[v], low[u] );
        }
        else if(!sccno[v])                  //处理回退边
            low[u]= min( low[u], num[v] );
    }
    if(low[u] == num[u]){                   //栈底的点是SCC的祖先，它的low = num
        cnt++;
        while(1){
            int v = stack[--top];           //v弹出栈
            sccno[v]= cnt;
            if(u==v) break;                 //栈底的点是SCC的祖先
        }
    }
}
void Tarjan(int n){
        cnt = top = dfn = 0;
        memset(sccno,0,sizeof(sccno));
        memset(num,0,sizeof(num));
        memset(low,0,sizeof(low));
        for(int i=1; i<=n; i++)
            if(!num[i])
                dfs(i);
}
int main(){
    int n,m,u,v;
    while(scanf("%d%d", &n, &m), n != 0 || m != 0) {
        for(int i=1; i<=n; i++){ G[i].clear();}
        for(int i=0; i<m; i++){
            scanf("%d%d", &u, &v);
            G[u].push_back(v);
        }
        Tarjan(n);
        printf("%s\n", cnt == 1 ? "Yes" : "No" );
    }
    return 0;
}
```

无向图的连通性

二分图

判定

```
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N=100010,M=2*N;
int n,m;
struct edge{int v,ne;}e[M];
int h[N],idx;
int color[N];

void add(int a,int b){
  e[++idx]={b,h[a]};
  h[a]=idx;
}
bool dfs(int u,int c){
  color[u]=c;
  for(int i=h[u];i;i=e[i].ne){
    int v=e[i].v;
    if(!color[v]){
      if(dfs(v,3-c))return 1;
    }
    else if(color[v]==c)return 1;//有奇环
  }
  return 0;
}
int main(){
  cin>>n>>m;
  for(int i=0;i<m;i++){
    int a,b;
    cin>>a>>b;
    add(a,b);
    add(b,a);
  }
  bool flag=0;
  for(int i=1;i<=n;i++)
    if(!color[i])
      if(dfs(i,1)){
        flag=1;//有奇环
        break;
      }
  if(flag) puts("No");
  else puts("Yes");
  return 0;
}
```

最大匹配

```
#include<bits/stdc++.h>
using namespace std;
int G[510][510];
int match[510], reserve_boy[510];
int n, m;
bool dfs(int x){
     for(int i=1; i<=m; i++)
         if(!reserve_boy[i] && G[x][i]){
             reserve_boy[i] = 1;
             if(!match[i] || dfs(match[i])){
                match[i] = x;
                return true;
            }
         }
     return false;
}
int main(){
    int e; scanf("%d%d%d",&n,&m,&e);
    while(e--){int a,b; scanf("%d%d",&a,&b); G[a][b]=1;}
    int sum=0;
    for(int i=1; i<=n; i++){
        memset(reserve_boy,0,sizeof(reserve_boy));
        if(dfs(i)) sum++;
    }
    printf("%d\n",sum);
    return 0;
}
```

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

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

并且当 $k = n$ 时必须要取等

```cpp
#include<bits/stdc++.h>
using namespace std;
int main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    int n,pre=0;cin>>n;
    vector<int>d(n,0);
    for(int i=0;i<n;++i){
        for(int j=0;j<n;++j){
            char c;cin>>c;
            if(c=='1')++d[j];
        }
    }
    sort(d.begin(),d.end());
    for(int i=0;i<n-1;++i){
        pre+=d[i];
        if(pre==(i+1)*i/2){
            cout<<n-1;
            return 0;
        }
    }
    cout<<n;
    return 0;
}
```

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

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

```cpp
using namespace std;
const int N = 5007, M = 5000007, INF = 0x3f3f3f3f;
int n, m;
char s[N][N];
bool vis[N];
bool dfs(int x , int fa) {
    vis[x] = 1;
    for(int i = 1; i <= n; ++i){
        if(s[x][i] - '0'){
            if(s[i][fa] - '0'){
                printf("%d %d %d\n", fa, x, i);
                return true;
            }
            if(!vis[i]){
                if(dfs(i, x))
                    return true;
            }
        }
    }
    return false;
}
int main() {
    scanf("%d", &n);
    for(int i = 1; i <= n; ++ i)
    scanf("%s", s[i] + 1);
    for(int i =1; i <= n; ++ i)
        if(!vis[i])
        if(dfs(i, i)) return 0;
    puts("-1");
    return 0;
}
```

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

```cpp
using i64 = long long;
const i64 mod = 998244353;
i64 qpow(i64 a, i64 b) {
    i64 res = 1;
    a %= mod;
    while (b) {
        if (b & 1) res = (res * a) % mod;
        b >>= 1;
        a = (a * a) % mod;
    }
    return res;
}
i64 inv(i64 a) {
    return qpow(a, mod - 2);
}
```

矩阵快速幂

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
inline int read() {
    int x = 0, f = 1;
    char c = getchar();
    while (c < '0' || c > '9') {
        if (c == '-') f = -1;
        c = getchar();
    }
    while (c >= '0' && c <= '9') {
        x = x * 10 + (c - '0');
        c = getchar();
    }
    return x * f;
}
const int N = 105;
struct matrix {
    int m[N][N];
};
int n, k;
const int mod = 1e9 + 7;
matrix operator*(const matrix& a, const matrix& b) {
    matrix c;
    memset(c.m, 0, sizeof(c.m));
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            for (int k = 0; k < n; ++k) {
                c.m[i][j] = (c.m[i][j] + a.m[i][k] * b.m[k][j]) % mod;
            }
        }
    }
    return c;
}
signed main() {
    matrix a;
    n = read(), k = read();
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            a.m[i][j] = read();
        }
    }
    matrix ans;
    memset(ans.m, 0, sizeof(ans.m));
    for (int i = 0; i < n; ++i) {
        ans.m[i][i] = 1;
    }
    while (k) {
        if (k & 1) ans = ans * a;
        a = a * a;
        k >>= 1;
    }
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            printf("%lld ", ans.m[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

矩阵快速幂加速递推

```cpp
#include <cstdio>
#include <cstring>
using namespace std;
#define int long long
struct matrix {
        int m[2][2];
};
const int mod = 10000;
inline int read() {
    int x = 0, f = 1;
    char c = getchar();
    while (c < '0' || c > '9') {
        if (c == '-') {
            f = -1;
        }
        c = getchar();
    }
    while (c >= '0' && c <= '9') {
        x = x * 10 + (c - '0');
        c = getchar();
    }
    return x * f;
}
matrix operator*(const matrix& a, const matrix& b) {
    matrix c;
    memset(c.m, 0, sizeof(c.m));
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < 2; ++j) {
            for (int k = 0; k < 2; ++k) {
                c.m[i][j] = (c.m[i][j] + a.m[i][k] * b.m[k][j]) % mod;
            }
        }
    }
    return c;
}
signed main() {
    matrix a;
    int n;
    while (true) {
        memset(a.m, 0, sizeof(a.m));
        a.m[0][0] = 1;
        a.m[0][1] = 1;
        a.m[1][0] = 1;
        n = read();
        if (n == -1) {
            break;
        }
        if (n == 0) {
            puts("0");
        }
        else {
            matrix res;
            memset(res.m, 0, sizeof(res.m));
            for (int i = 0; i < 2; ++i) {
                res.m[i][i] = 1;
            }
            while (n) {
                if (n & 1) {
                    res = res * a;
                }
                a = a * a;
                n >>= 1;
            }
            printf("%lld\n", res.m[1][0]);
        }
    }
    return 0;
}
```

扩展欧几里得算法

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

令a = n - m, b = L, c = x - y

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
ll extend_gcd(ll a, ll b, ll& x, ll& y) {
    if (b == 0) {
        x = 1; y = 0; return a;
    }
    ll d = extend_gcd(b, a % b, y, x);
    y -= a / b * x;
    return d;
}
int main() {
    ll n, m, x, y, L;
    cin >> x >> y >> m >> n >> L;
    ll a = n - m, c = x - y;
    if (a < 0) a = -a, c = -c;
    ll d = extend_gcd(a, L, x, y);
    if (c % d != 0) cout << "Impossible";
    else cout << ((x * (c / d)) % (L / d) + (L / d)) % (L / d);
}
```

```
long long mod_inverse(long long a, long long m){
    long long x, y;
    extend_gcd(a, m, x, y);
    return (x % m + m) % m;
}
int main(){
    long long a, m;  cin >> a >> m;
    cout << mod_inverse(a, m);
    return 0;
}
```

扩展中国剩余定理

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 100010;
int n;
ll ai[N], mi[N];
ll mul(ll a,ll b,ll m){                     //乘法取模: a*b % m
    ll res=0;
    while(b>0){
        if(b&1) res=(res+a) % m;
        a=(a+a) % m;
        b>>=1;
    }
    return res;
}
ll extend_gcd(ll a,ll b,ll &x,ll &y){   //扩展欧几里得
    if(b == 0){ x=1; y=0; return a;}
    ll d = extend_gcd(b,a%b,y,x);
    y -= a/b * x;
    return d;
}
ll excrt(){                                  //求解同余方程组，返回最小正整数解
    ll x,y;
    ll m1 = mi[1], a1 = ai[1];               //第1个等式
    ll ans = 0;
    for(int i=2;i<=n;i++){                   //合并每2个等式
        ll a2 = ai[i], m2 = mi[i];           // 第2个等式
        //合并为: aX + bY = c
        ll a = m1, b = m2, c = (a2 - a1%m2 + m2) % m2;
        //下面求解 aX + bY = c
        ll d = extend_gcd(a,b,x,y);   //用扩展欧几里得求x0
        if(c%d != 0) return -1;       //无解
```

```cpp
        x = mul(x,c/d,b/d);          //aX + bY = c 的特解t，最小值
        ans = a1 + x* m1;            //代回原第1个等式，求得特解x'
        m1 = m2/d*m1;                //先除再乘，避免越界。合并后的新m1
        ans = (ans%m1 + m1) % m1;    //最小正整数解
        a1 = ans;                    //合并后的新a1
    }
    return ans;
}
int main(){
    scanf("%d", &n);
    for(int i=1;i<=n;++i)  scanf("%lld%lld",&mi[i],&ai[i]);
    printf("%lld",excrt());
    return 0;
}
```

裴蜀定理

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n, a, s;
    cin >> n;
    for (int i = 1; i <= n; ++i) {
        cin >> a;
        s = __gcd(s, abs(a));
    }
    cout << s;
}
```

FFT

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

```cpp
const double Pi=acos(-1.0);
struct complex {
    double x,y;
    complex (double xx=0,double yy=0): x(xx), y(yy) {}
}a[MAXN],b[MAXN];
complex operator+(complex a,complex b){ return complex(a.x+b.x, a.y+b.y);}
complex operator-(complex a,complex b){ return complex(a.x-b.x, a.y-b.y);}
complex operator*(complex a,complex b){ return complex(a.x*b.x-a.y*b.y,
a.x*b.y+a.y*b.x);}
int N,M;
int l,r[MAXN];
int limit=1;
void FFT(complex *A,int type) {
    for(int i=0;i<limit;i++)
        if(i<r[i]) swap(A[i],A[r[i]]);
    for(int mid=1;mid<limit;mid<<=1) {
        complex Wn(cos(Pi/mid) , type*sin(Pi/mid));
        for(int R=mid<<1,j=0;j<limit;j+=R) {
            complex w(1,0);
            for(int k=0;k<mid;k++,w=w*Wn) {
                complex x=A[j+k],y=w*A[j+mid+k];
                A[j+k]=x+y;
                A[j+mid+k]=x-y;
            }
        }
    }
}
int main() {
    int N = read(), M = read();
    for(int i=0;i<=N;i++) a[i].x = read();
    for(int i=0;i<=M;i++) b[i].x = read();
    while(limit<=N+M) limit<<=1,l++;
    for(int i=0;i<limit;i++) r[i]=(r[i>>1]>>1)|((i&1)<<(l-1));
    FFT(a,1);
    FFT(b,1);
    for(int i=0;i<=limit;i++) a[i]=a[i]*b[i];
    FFT(a,-1);
    for(int i=0;i<=N+M;i++) printf("%d ",(int)(a[i].x/limit+0.5));
    return 0;
}
```

NTT

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

```cpp
typedef long long ll;
const int NR = 1 << 22, g = 3, gi = 332748118, mod = 998244353;
// 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
int n, m, rev[NR];
ll a[NR], b[NR];
ll qpow(ll x, ll y) {
    ll res = 1;
    x %= mod;
    while (y) {
        if (y & 1) res = (res * x) % mod;
        x = (x * x) % mod;
        y >>= 1;
    }
    return res;
}
// 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
void NTT(ll a[], int n, int type) {
    for (int i = 0; i < n; ++i) {
        if (i < rev[i]) swap(a[i], a[rev[i]]);
    }
    for (int i = 1; i < n; i <<= 1) {
        ll gn = qpow(type ? g : gi, (mod - 1) / (i << 1));
        for (int j = 0; j < n; j += (i << 1)) {
            ll g0 = 1;
            for (int k = 0; k < i; ++k, g0 = g0 * gn % mod) {
                ll x = a[j + k], y = g0 * a[i + j + k] % mod;
                a[j + k] = (x + y) % mod;
                a[i + j + k] = (x - y + mod) % mod;
            }
        }
    }
}
int main() {
    scanf("%d%d", &n, &m);
    for (int i = 0; i <= n; ++i) scanf("%lld", &a[i]);
    for (int i = 0; i <= m; ++i) scanf("%lld", &b[i]);
    int len = 1 << max((int)ceil(log2(n + m)), 1);
    for (int i = 0; i < len; ++i) rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (max((int)ceil(log2(n + m)), 1) - 1));
    NTT(a, len, 1);
    NTT(b, len, 1);
    for (int i = 0; i <= len; ++i)
        a[i] = a[i] * b[i] % mod;          // O(n)乘法
    NTT(a, len, 0);                        // 点值表示法转系数表示法
    ll inv = qpow(len, mod - 2);           // inv为len的逆元（费马小定理求逆元）
    for (int i = 0; i <= n + m; ++i)       // 输出
```

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

素数筛

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

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

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

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

欧拉函数

```
int euler(int n){
    int ans = n;
    for(int p = 2; p*p <= n; ++ p){ //试除法：检查从2到sqrt(n)的每个数
        if(n%p == 0){               //能整除，p是一个因子，而且是质因子，请思考
            ans = ans/p*(p-1);      //求欧拉函数的通式
            while(n%p == 0)         //去掉这个因子的幂，并使得下一个p是质因子
                n /= p;             //减小了n
        }
    }
    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
    return ans;
}
```

整数分块/数论分块

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

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    long long n, l, r, ans = 0;
    cin >> n;
    for (l = 1; l <= n; l = r + 1) {
        r = n / (n / l);
        ans += (r - l + 1) * (n / l);
    }
    cout << ans;
}
```

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

前缀和 $s[i] = \sum_{j=1}^{i} f(j)$

```cpp
long long n, l, r, ans = 0;
cin >> n;
for (l = 1; l <= n; l = r + 1) {
    r = n / (n / l);
    ans += (s(r) - s(l - 1)) * (n / l);
}
cout << ans;
```

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

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    long long n, k;
    cin >> n >> k;
    long long L, R, ans = n * k;
    for (L = 1; L <= n; L = R + 1) {
        if (k / L == 0) break;
        R = min(k / (k / L), n);
        ans -= k / L * (L + R) * (R - L + 1) / 2;
    }
    cout << ans;
}
```

狄利克雷卷积

```

```

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:

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

```
```

Demo:

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

```
```

高斯消元法

```cpp
#include<bits/stdc++.h>
using namespace std;
double a[105][105];
double eps=1e-7;
int main(){
    int n; cin>>n;
    for(int i=1;i<=n;++i)
        for(int j=1;j<=n+1;++j)
                cin>>a[i][j];
    for(int i=1;i<=n;++i) {
        int max=i;
        for(int j=i+1;j<=n;++j)
            if(fabs(a[j][i])>fabs(a[max][i])) max=j;
        for(int j=1;j<=n+1;++j) swap(a[i][j],a[max][j]);
        if(fabs(a[i][i])<eps) {
            if (fabs(a[i][n+1])<eps) cout<<0;
            else cout<<-1;
            return 0;
        }
        for(int j=n+1;j>=1;j--) a[i][j]=a[i][j]/a[i][i];
        for(int j=1;j<=n;++j) {
            if(j!=i) {
                double temp=a[j][i]/a[i][i];
                for(int k=1;k<=n+1;++k) a[j][k]-=a[i][k]*temp;
            }
        }
    }
    for(int i=1;i<=n;++i) {
        cout<<"x"<<i<<"=";
        cout<<fixed<<setprecision(2)<<a[i][n+1]<< '\n';
    }
    return 0;
}
```

质因数分解

```
int prime[N];       //记录质数
int vis[N];         //记录最小质因子
int euler_sieve(int n){
     int cnt=0;
     memset(vis,0,sizeof(vis));
     memset(prime,0,sizeof(prime));
        for(int i=2;i<=n;i++){
         if(!vis[i]){ vis[i]=i; prime[cnt++]=i;}    //vis[]记录最小质因子
         for(int j=0; j<cnt; j++){
             if(i*prime[j] >n)  break;
             vis[i*prime[j]] = prime[j];           //vis[]记录最小质因子
             if(i%prime[j]==0)  break;
         }
     }
     return cnt;
}
```

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

```cpp
#include <bits/stdc++.h>
using namespace std;
const int _ = 3e5 + 5;
pair<int,int> a[_];
struct Compare {
    bool operator()(const pair<int,int>& p2, const pair<int,int>& p1) {
        if (p2.first == p1.first) {
            return p1.second < p2.second;
        }
        return p1.first > p2.first;
    }
};
void solve() {
    int n, k;
    cin >> n >> k;
    priority_queue<pair<int,int>, vector<pair<int,int>>, Compare> pq;
    for (int i = 0; i < n; ++i) {
        int x;
        cin >> x;
        a[i].first = x % k;
        a[i].second = i;
        if (x % k == 0) cout << i + 1 << ' ';
        else pq.push(a[i]);
    }
    while (!pq.empty()) {
        cout << pq.top().second + 1 << ' ';
        pq.pop();
    }
    cout << '\n';
}
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int t;
    cin >> t;
    while (t--) solve();
}
```

unordered_map<pair<int,int>>的使用

```

```

```
#include <bits/stdc++.h>
using namespace std;
void solve() {
    int n, k;
    cin >> n >> k;
    vector<int> a(n);
    for (int i = 0; i < n; ++i) cin >> a[i];
    vector<int> p(n);
    iota(p.begin(), p.end(), 0);
    stable_sort(p.begin(), p.end(), [&](int i, int j){
        return (a[i] - 1) % k > (a[j] - 1) % k;
    });
    for (int i = 0; i < n; ++i) {
        cout << p[i] + 1 << " \n"[i == n - 1];
    }
}
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int t;
    cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

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

```cpp
    string s;
    cin >> s;
    set<pair<int,int>> st;
    vector<int> nxt(n + 1, n), lst(n + 1, -1);
    for (int i = n - 1; i >= 0; --i) {
        nxt[i] = s[i] == '1' ? i : nxt[i + 1];
    }
    for (int i = 0; i < n; ++i) {
        lst[i + 1] = s[i] == '0' ? i : lst[i];
    }
    while (m--) {
        int l, r;
        cin >> l >> r;
        l--;
        l = nxt[l];
        r = lst[r];
        if (l > r) {
            l = r = -1;
        }
        st.emplace(l, r);
    }
    cout << st.size() << '\n';
}
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int t;
    cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

set和multiset

bitset

高精度计算

离散化

基础算法

排序

归并排序

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
int a[N];
int temp[N];
void merge_sort(int arr[], int l, int r) {
    if (l == r) return;
    int m = (l + r) / 2;
    merge_sort(arr, l, m);
    merge_sort(arr, m + 1, r);
    int t1 = l, t2 = m + 1;
    for (int i = l; i <= r; ++i) {
        if ((arr[t1] < arr[t2] && t1 <= m) || t2 > r) {
            temp[i] = arr[t1++];
        }
        else {
            temp[i] = arr[t2++];
        }
    }
    for (int i = l; i <= r; ++i) {
        arr[i] = temp[i];
    }
}
int main() {
    int n;
    scanf("%d", &n);
    for (int i = 0; i < n; ++i) {
        scanf("%d", &a[i]);
    }
    merge_sort(a, 0, n - 1);
    for (int i = 0; i < n; ++i) {
        printf("%d ", a[i]);
    }
    printf("\n");
    return 0;
}
```

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

```cpp
using namespace std;
const int N = 5e5 + 5;
long long a[N];
long long temp[N];
long long res = 0;
inline long long read() {
    long long x = 0, f = 1;
    char ch = getchar();
    while (ch < '0' || ch > '9') {
            if (ch == '-') f = -1;
            ch = getchar();
    }
    while (ch >= '0' && ch <= '9') {
            x = x * 10 + (ch - '0');
            ch = getchar();
    }
    return x * f;
}
void merge_sort(long long arr[], long long l, long long r) {
    if (l == r) return;
    long long m = l + (r - l) / 2;
    merge_sort(arr, l, m);
    merge_sort(arr, m + 1, r);
    long long t1 = l, t2 = m + 1;
    for (long long i = l; i <= r; ++i) {
        if ((arr[t1] > arr[t2] && t2 <= r) || t1 > m) {
            temp[i] = arr[t2++];
            res += m - t1 + 1;
        }
        else {
            temp[i] = arr[t1++];
        }
    }
    for (long long i = l; i <= r; ++i) {
        arr[i] = temp[i];
    }
}
int main() {
    long long n;
    n = read();
    for (long long i = 0; i < n; ++i) {
            a[i] = read();
    }
    merge_sort(a, 0, n - 1);
    printf("%lld\n", res);
    return 0;
}
```

快速排序

```cpp
#include <cstdio>
#include <algorithm>
using namespace std;
const int N = 1e5 + 5;
int a[N];
void quick_sort(int left, int right) {
    int mid = a[left + (right - left) / 2];
    int i = left, j = right - 1;
    while (i <= j) {
        while (a[i] < mid) ++i;
        while (a[j] > mid) --j;
        if (i <= j) {
            swap(a[i], a[j]);
            ++i;
            --j;
        }
    }
    if (left <= j) quick_sort(left, j + 1);
    if (i < right) quick_sort(i, right);
}
int main() {
    int n;
    scanf("%d", &n);
    for (int i = 0; i < n; ++i) {
        scanf("%d", &a[i]);
    }
    quick_sort(0, n);
    for (int i = 0; i < n; ++i) {
        printf("%d ", a[i]);
    }
    return 0;
}
```

```cpp
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
const int N = 5e6 + 5;
int a[N];
int quick_sort(int left, int right, int k) {
    int mid = a[left + (right - left) / 2];
    int i = left, j = right - 1;
    while (i <= j) {
        while (a[i] < mid) ++i;
        while (a[j] > mid) --j;
        if (i <= j) {
            swap(a[i], a[j]);
            ++i;
            --j;
        }
    }
    if (left <= j && k <= j) return quick_sort(left, j + 1, k);
    if (i < right && k >= i) return quick_sort(i, right, k);
    return a[k];
}
int main() {
    int n, k;
    scanf("%d%d", &n, &k);
    for (int i = 0; i < n; ++i) {
        scanf("%d", &a[i]);
    }
    printf("%d\n", quick_sort(0, n, k));
    return 0;
}
```

滑动窗口

```cpp
class Solution {
public:
    int countCompleteSubarrays(vector<int>& nums) {
        int n = nums.size();
        int cnt1 = 0;
        array<int,2001> st{0};
        for (int n: nums) {
            if (st[n] == 0) { ++cnt1; st[n] = 1; }
        }
        fill(st.begin(), st.end(), 0);
        int cnt2 = 0;
        int ans = 0;
        for (int i = 0, j = 0; i < n; ++i) {
            while (j < n && cnt2 < cnt1) {
                if (!st[nums[j]]) {
                    ++cnt2;
                }
                ++st[nums[j]];
                ++j;
            }
            if (cnt1 == cnt2) ans += n - j + 1;
            --st[nums[i]];
            if (!st[nums[i]]) --cnt2;
        }
        return ans;
    }
};
```

前缀和与二维前缀和

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int n;
    cin >> n;
    vector<int> a(n + 1);
    for (int i = 1; i <= n; ++i) cin >> a[i];
    vector<int> sum(n + 1, 0);
    for (int i = 1; i <= n; ++i) sum[i] = sum[i - 1] + a[i];
    int m;
    cin >> m;
    while (m--) {
        int l, r;
        cin >> l >> r;
        cout << sum[r] - sum[l - 1] << '\n';
    }
    return 0;
}
```

```cpp
#include <bits/stdc++.h>
using namespace std;
const int _ = 5005;
int s[_][_];
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int n, m;
    cin >> n >> m;
    for (int i = 0; i < n; ++i) {
        int x, y, v;
        cin >> x >> y >> v;
        x++, y++;
        s[x][y] += v;
    }
    for (int i = 1; i <= 5001; ++i) {
        for (int j = 1; j <= 5001; ++j) {
            s[i][j] += s[i - 1][j] + s[i][j - 1] - s[i - 1][j - 1];
        }
    }
    int res = 0;
    for (int i = m; i <= 5001; ++i){
        for (int j = m; j <= 5001; ++j) {
            res = max(res, s[i][j] - s[i - m][j] - s[i][j - m] + s[i - m][j - m]
        }
    }
    cout << res;
    return 0;
}
```

树上前缀和

```

```

差分

```

```

树上差分

```

```

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

```
int find(int q) {
    int ans = 0;
    int l = 1, r = n; // [1, n]
    while (l <= r) {
        int mid = (l + r) / 2;
        if (a[mid] <= q) ans = mid, l = mid + 1;
        else r = mid - 1;
    }
    return ans;
}
```

查找第一个大于等于target的下标

```
int find(int q) {
    int ans = 0;
    int l = 1, r = n; // [1, n]
    while (l <= r) {
        int mid = (l + r) / 2;
        if (a[mid] >= q) ans = mid, r = mid - 1;
        else l = mid + 1;
    }
    return ans;
}
```

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

```cpp
class Solution {
public:
    long long findMaximumElegance(vector<vector<int>>& items, int k) {
        sort(items.begin(), items.end(), [](const auto& i1, const auto& i2){
            return i1[0] > i2[0];
        });
        int n = items.size();
        set<long long> st;
        long long res = 0, total_profit = 0;
        vector<long long> dup;
        for (int i = 0; i < n; ++i) {
            long long p = items[i][0], c = items[i][1];
            if (i < k) {
                total_profit += p;
                if (st.find(c) == st.end()) {
                    st.insert(c);
                } else {
                    dup.push_back(p);
                }
            } else {
                if (!dup.empty() && st.find(c) == st.end()) {
                    st.insert(c);
                    total_profit -= dup.back();
                    dup.pop_back();
                    total_profit += p;
                }
            }
            res = max(res, total_profit + (long long)st.size() * (long long)st.s
        }
        return res;
    }
};
```

单阶乘

```

```

连续单阶乘

```

```

双阶乘

```

```

连续双阶乘

```cpp
#include <bits/stdc++.h>
using namespace std;
__int128 read() {
    __int128 x = 0, f = 1;
    char c = getchar();
    while (c < '0' || c > '9') {
        if (c == '-') f = -1;
        c = getchar();
    }
    while (c >= '0' && c <= '9') {
        x = (x << 3) + (x << 1) + (c ^ 48);
        c = getchar();
    }
    return x * f;
}
void write(__int128 x) {
    if (x < 0) {
        write(-x);
        return;
    }
    if (x >= 10)
        write(x / 10);
    putchar((x % 10) ^ 48);
}
void solve() {
    __int128 n;
    n = read();
    __int128 res = 0;
    __int128 u = 5, v, r;
    while (u <= n) {
        v = u << 1;
        r = n / v;
        res += r * (r - 1) * u;
        res += r * (u + 1) >> 1;
        if (n % 2 == 0) res += r;
        res += ((n - r * v + 1) >> 1 << 1) * r;
        if (n >= r * v + u) res += (n - r * v - u + 2) >> 1;
        u *= 5;
    }
    write(res);
}
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    solve();
}
```