?????
```
int pre[maxn], iscur[maxn], bccno[maxn], dfs_clock, bcc_cnt;
vector<int> G[maxn], bcc[maxn];

stack<edge> s;
int dfs(int u, int fa)
{
    int lowu = pre[u] = ++dfs_clock;
    int child = 0;
    for(int i = 0 ; i < G[u].size();i++)
    {
        int v= G[u][i];
        edge e = (edge) {u, v};
        if(!pre[v])
        {
            s.push(e);
            child++;
            int lowv = dfs(v, u);
            lowu = min(lowu, lowv);
            if(lowv >= pre[u])
            {
                iscut[u] = true;
                bcc_cnt++;
                bcc[bcc_cnt].clear();
                for(;;)
                {
                    edge x = s.top(); s.pop();
                    if(bccno[x.u] != bcc_cnt)
                    {
                        bcc[bcc_cnt].push_back(x.u);
                        bccno[x.u] = bcc_cnt;
                    }
                    if(bccno[x.v] != bcc_cnt)
                    {
                        bcc[bcc_cnt].push_back(x.v);
                        bccno[x.v] = bcc_cnt;
                    }
                    if(x.u == u && x.v == v)
                    {
                        break;
                    }
                }
            }
            else if(pre[v] < pre[u] && v != fa)
            {
                S.push(e);
                lowu = min(lowu, pre[v]);
            }
        }
        if(fa < 0 && child == 1)
        {
            iscut[u] = 0;
        }
    }
    return lowu;
}
void find_bcc(int u)
{
    memset(pre, 0, sizeof(pre));
    memset(iscut, 0, sizeof(iscut));
    memset(bccno, 0, sizeof(bccno));
    dfs_clock = bcc_cnt = 0;
    for(int i = 0; i < n; i++)
    {
        if(!pre[i]) dfs(i, -1);
    }
}
```

????BIT

```
#include<cstdio>
using namespace std;
const int maxn = 10000;
int n;
int C[maxn];
int lsb(int x)
{
    return x&(-x);
}
int sum(int x)
{
    int ret = 0;
    while(x > 0)
    {
        ret += C[x];
        x-=lsb(x);
    }
    return ret;
}
void add(int x, int d)
{
    while(x <= n)
    {
        C[x] += d;
        x += lsb(x);
    }
}
/* **************************************************
Author        :kuangbin
??????1??
????dance(0)
ansd?ans???size
ans??????

************************************************** */
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <iostream>
#include <algorithm>
#include <vector>
#include <queue>
#include <set>
#include <map>
#include <string>
#include <math.h>
#include <stdlib.h>
#include <time.h>
using namespace std;
const int maxnode = 100010;
const int MaxM = 1010;
const int MaxN = 1010;
//????
struct DLX
{
    int n,m,size;
    int U[maxnode],D[maxnode],R[maxnode],L[maxnode],Row[maxnode],Col[maxnode];
    int H[MaxN], S[MaxM];
    int ansd, ans[MaxN];
    void init(int _n,int _m)
    {
        n = _n;
        m = _m;
        for(int i = 0;i <= m;i++)
        {
            S[i] = 0;
            U[i] = D[i] = i;
            L[i] = i-1;
            R[i] = i+1;
        }
        R[m] = 0; L[0] = m;
        size = m;
        for(int i = 1;i <= n;i++)
            H[i] = -1;
    }
    void Link(int r,int c)
    {
        ++S[Col[++size]=c];
        Row[size] = r;
        D[size] = D[c];
        U[D[c]] = size;
        U[size] = c;
        D[c] = size;
        if(H[r] < 0)H[r] = L[size] = R[size] = size;
        else
        {
            R[size] = R[H[r]];
            L[R[H[r]]] = size;
            L[size] = H[r];
            R[H[r]] = size;
        }
    }
    void remove(int c)
    {
        L[R[c]] = L[c]; R[L[c]] = R[c];
        for(int i = D[c];i != c;i = D[i])
            for(int j = R[i];j != i;j = R[j])
            {
                U[D[j]] = U[j];
                D[U[j]] = D[j];
                --S[Col[j]];
            }
    }
    void resume(int c)
    {
        for(int i = U[c];i != c;i = U[i])
            for(int j = L[i];j != i;j = L[j])
                ++S[Col[U[D[j]]=D[U[j]]=j]];
        L[R[c]] = R[L[c]] = c;
    }
    //d?????
    bool Dance(int d)
    {
        if(R[0] == 0)
        {
            ansd = d;
            return true;
        }
        int c = R[0];
        for(int i = R[0];i != 0;i = R[i])
            if(S[i] < S[c])
                c = i;
        remove(c);
        for(int i = D[c];i != c;i = D[i])
        {
            ans[d] = Row[i];
            for(int j = R[i]; j != i;j = R[j])remove(Col[j]);
            if(Dance(d+1))return true;
            for(int j = L[i]; j != i;j = L[j])resume(Col[j]);
        }
        resume(c);
        return false;
    }
};

DLX g;
```

```
//????
//K???????
/* ***********************************************
Author        :kuangbin
??????1??
????dance(0)
ansd?ans???size
ans??????
************************************************ */
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include<functional>
#include <string.h>
#include <iostream>
#include <algorithm>
#include <vector>
#include <queue>
#include <set>
#include <map>
#include <string>
#include <math.h>
#include <stdlib.h>
#include <time.h>
using namespace std;
const int inf = 0x3f3f3f3f;
const int maxnode = 100010;
const int maxn = 1010;
const int MaxN = 1010;

struct DLX
{
    int n, m, size;
    int U[maxnode], D[maxnode], R[maxnode], L[maxnode];
    int Row[maxnode], Col[maxnode];
    int H[MaxN], S[MaxN];
    int ansd, ans[MaxN];
    int K;
    void init(int _n, int _m)
```

```
{
    n = _n;
    m = _m;
    for (int i = 0; i <= m; i++)
    {
        U[i] = D[i] = i;
        L[i] = i - 1;
        R[i] = i + 1;
        S[i] = 0;
    }
    R[m] = 0;
    L[0] = m;
    size = m;
    for (int i = 1; i <= n; i++)
    {
        H[i] = -1;
    }

}
void Link(int r, int c)
{
    S[Col[++size] = c]++;
    Row[size] = r;
    U[size] = U[c];
    D[size] = c;
    D[U[c]] = size;
    U[c] = size;
    if (H[r] == -1)
    {
        H[r] = R[size] = L[size] = size;
    }
    else
    {
        R[size] = H[r];
        L[size] = L[H[r]];
        R[L[size]] = size;
        L[H[r]] = size;
    }
}
void remove(int c)
{
    for (int i = D[c]; i != c; i = D[i])
    {
        L[R[i]] = L[i];
        R[L[i]] = R[i];
    }
}
void resume(int c)
{
    for (int i = U[c]; i != c; i = U[i])
    {
        R[L[i]] = L[R[i]] = i;
    }
}
bool v[maxnode];
int h()
{
    int ret = 0;
    for (int c = R[0]; c != 0; c = R[c]) v[c] = true;
    for (int c = R[0]; c != 0; c = R[c])
    {
        if (v[c])
        {
            ret++;
            for (int i = D[c]; i != c; i = D[i])
            {
```

```cpp
                    for (int j = R[i]; j != i; j = R[j])
                    {
                        v[Col[j]] = false;
                    }
                }
            }
        }
        return ret;
    }
    bool dance(int d)
    {
        if (d + h() > K)
        {
            return false;
        }
        if (R[0] == 0)
        {
            return d <= K;
        }
        int c = R[0];
        for (int i = R[0]; i != 0; i = R[i])
        {
            if (S[i] < S[c])
            {
                c = i;
            }
        }
        for (int i = D[c]; i != c; i = D[i])
        {
            for (int j = R[i]; j != i; j = R[j])
            {
                remove(j);
            }
            remove(i);
            if (dance(d + 1)) return true;
            resume(i);
            for (int j = L[i]; j != i; j = L[j])
            {
                resume(j);
            }
        }
        return false;
    }
};

DLX gg;
```

```
//????
//K???????
/* ***********************************************
Author        :kuangbin
??????1??
????dance(0)
ansd?ans???size
ans??????
************************************************/
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include<functional>
#include <string.h>
#include <iostream>
#include <algorithm>
#include <vector>
#include <queue>
#include <set>
#include <map>
#include <string>
#include <math.h>
#include <stdlib.h>
#include <time.h>
using namespace std;
const int inf = 0x3f3f3f3f;
const int maxnode = 100010;
const int maxn = 1010;
const int MaxN = 1010;

struct DLX
{
    int n, m, size;
    int U[maxnode], D[maxnode], R[maxnode], L[maxnode];
    int Row[maxnode], Col[maxnode];
    int H[MaxN], S[MaxN];
    int ansd;
    int K;
    void init(int _n, int _m)
    {
        n = _n;
        ansd = inf;
        m = _m;
        for (int i = 0; i <= m; i++)
        {
            U[i] = D[i] = i;
            L[i] = i - 1;
            R[i] = i + 1;
            S[i] = 0;
        }
        R[m] = 0;
        L[0] = m;
        size = m;
        for (int i = 1; i <= n; i++)
        {
            H[i] = -1;
        }

    }
    void Link(int r, int c)
    {
        S[Col[++size] = c]++;
        Row[size] = r;
        U[size] = U[c];
        D[size] = c;
        D[U[c]] = size;
        U[c] = size;
        if (H[r] == -1)
        {
            H[r] = R[size] = L[size] = size;
        }
        else
        {
            R[size] = H[r];
            L[size] = L[H[r]];
            R[L[size]] = size;
            L[H[r]] = size;
        }
    }
    void remove(int c)
    {
        for (int i = D[c]; i != c; i = D[i])
        {
            L[R[i]] = L[i];
            R[L[i]] = R[i];
        }
    }
    void resume(int c)
    {
        for (int i = U[c]; i != c; i = U[i])
        {
            R[L[i]] = L[R[i]] = i;
        }
    }
    bool v[maxnode];
    int h()
    {
        int ret = 0;
        for (int c = R[0]; c != 0; c = R[c]) v[c] = true;
        for (int c = R[0]; c != 0; c = R[c])
        {
            if (v[c])
            {
                ret++;
                for (int i = D[c]; i != c; i = D[i])
                {
                    for (int j = R[i]; j != i; j = R[j])
                    {
                        v[Col[j]] = false;
                    }
                }
            }
        }
        return ret;
    }
    bool dance(int d)
    {
        if (d + h() > ansd)
        {
            return false;
        }
        if (R[0] == 0)
        {
            if (ansd > d) ansd = d;
            return true;
        }
        bool ret = false;
        int c = R[0];
        for (int i = R[0]; i != 0; i = R[i])
        {
            if (S[i] < S[c])
            {
                c = i;
            }
        }
        for (int i = D[c]; i != c; i = D[i])
        {
            remove(i);
            for (int j = R[i]; j != i; j = R[j])
            {
                remove(j);
            }
            if (dance(d + 1)) ret = true;
            for (int j = L[i]; j != i; j = L[j])
            {
                resume(j);
            }
            resume(i);
        }
        return ret;
    }
    void Dance(int d)
    {
        if (d + h() >= ansd)return;
        if (R[0] == 0)
        {
            if (d < ansd)ansd = d;
            return;
        }
        int c = R[0];
        for (int i = R[0]; i != 0; i = R[i])
            if (S[i] < S[c])
                c = i;
        for (int i = D[c]; i != c; i = D[i])
        {
            remove(i);
            for (int j = R[i]; j != i; j = R[j])remove(j);
            Dance(d + 1);
            for (int j = L[i]; j != i; j = L[j])resume(j);
            resume(i);
        }
    }
};

DLX gg;
```

dijkstra???

```
#include<iostream>
#include<vector>
#include<cstring>
#include<functional>
#include<algorithm>
#include<string>
#include<queue>
using namespace std;
typedef pair<int, int> pii;
const int maxn = 10005;
int dis[maxn];
vector<int> G[maxn];
const int inf = 0x3f3f3f3f;
int ecnt;
struct edge
{
    int from, to, cost;
};
edge es[maxn * maxn];

void dijkstra(int s)
```

```
{
    priority_queue<pii, vector<pii>, greater<pii> > q;
    q.push({0, s});
    dis[s] = 0;
    while (!q.empty())
    {
        pii c = q.top();
        q.pop();
        int cur = c.second;

        for (int i = 0; i < G[cur].size(); i++)
        {
            edge & e = es[G[cur][i]];
            if (dis[e.to] > dis[e.from] + e.cost)
            {
                dis[e.to] = dis[e.from] + e.cost;
                q.push({dis[e.to],e.to});
            }
        }

    }
}
void init(int n)
{
    ecnt = 0;
    for(int i = 0; i < n; i++)
    {
        G[i].clear();
    }
    memset(dis, inf, sizeof(dis));
}
void addedge(int u, int v, int w)
{
    es[ecnt] = {u, v, w};
    G[u].push_back(ecnt++);
}

???
#include<cstdio>
const int maxn = 100000 + 10;
int pa[maxn];
int rank[maxn];
int findset(int x)
{
    return pa[x]!=x ? pa[x] = findset(pa[x]) : x;
}
void uni(int a, int b)
{
    if((a = findset(a)) == (b = findset(b)))
    {
        return;
    }
    if(rank[a] > rank[b])
    {
        pa[b] = a;
    }
    else
    {
        if(rank[a] == rank[b])
            rank[b] ++;
        pa[a] = b;
    }
}
void initps(int n)
{
    for(int i = 0; i <= n; i++)
    {
        pa[i] = i;
        rank[i] = 0;
    }
}
??phi?????n??n???????

int euler_phi(int n)
{
    int m = (int) sqrt(n +0.5);
    int ans = n;
    for(int i = 2; i <= m; i++)
    {
        if(n % i == 0)
        {
            ans = ans / i * (i - 1);
            while(n % i == 0)
            {
                n /= i;
            }
        }
    }
    if(n > 1)
    {
        ans = ans / n * (n - 1);
    }
}
int phi[maxn];
void phi_table(int n)
{
    for(int i = 2; i <= n; i++)
    {
        phi[i] = 0;
    }
    phi[1] = 1;
    for(int i = 2; i <= n; i++)
    {
        if(!phi[i])
        {
            for(int j = i; j <= n; j+= i)
            {
                if(!phi[j])
                {
                    phi[j] = j;
                }
                phi[j] = phi[j] / i * (i - 1);
            }
        }
    }
}


iscut???

int dfs_clock;
int dfs(int u, int fa)
{
    int lowu = pre[u] = ++dfs_clock;//
    int child = 0;
    for(int i = 0; i < G[u].size(); i++)
    {
        int v = G[u][i];
        if(!pre[v])//have not been visited
        {
            child++;
            int lowv = dfs(v, u);
            lowu = min(lowu, lowv);
            if(lowv >= pre[u])
            {
                iscut[u] = true;
            }
        }
        else if(pre[v] < pre[u] && v != fa)
        {
            lowu = min(lowu, pre[u]);
        }
    }
    if(fa < 0 && child == 1)
    {
        iscut[u] = 0;
    }
    low[u] = lowu;
    return lowu;
}


/*
* Kruskal???MST
*/
#include<iostream>
#include<cstring>
#include<algorithm>
using namespace std;
const int MAXN = 110;//????
const int MAXM = 10000;//????
int F[MAXN];//?????
struct Edge
{
    int u, v, w;
}edge[MAXM];//???????????/??/??
int tol;//?????????0
void addedge(int u, int v, int w)
{

    edge[tol].u = u;
    edge[tol].v = v;
    edge[tol++].w = w;
}
bool cmp(Edge a, Edge b)
{//?????????????????
    return a.w<b.w;
}
int find(int x)
{
    if (F[x] == -1)return x;
    else return F[x] = find(F[x]);
}
int Kruskal(int n)//???????????????????????-1
{
    memset(F, -1, sizeof(F));
    sort(edge, edge + tol, cmp);
    int cnt = 0;//???????
    int ans = 0;
    for (int i = 0; i<tol; i++)
    {
        int u = edge[i].u;
        int v = edge[i].v;
        int w = edge[i].w;
        int t1 = find(u);
        int t2 = find(v);
        if (t1 != t2)
        {
            ans += w;
            F[t1] = t2;
            cnt++;
        }
        if (cnt == n - 1)break;
    }
    if (cnt<n - 1)return -1;//???
    else return ans;
}


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


/*
* Prim?MST
* ????cost[][]????0???0~n-1
* ?????????????-1???????
*/
const int INF = 0x3f3f3f3f;
const int MAXN = 110;
bool vis[MAXN];
int lowc[MAXN];
int cost[MAXN][MAXN];
int Prim(int n)//??0~n-1
{
    int ans = 0;
    memset(vis, false, sizeof(vis));
    vis[0] = true;
    for (int i = 1; i<n; i++)lowc[i] = cost[0][i];
    for (int i = 1; i<n; i++)
    {
        int minc = INF;
        int p = -1;
        for (int j = 0; j<n; j++)
            if (!vis[j] && minc>lowc[j])
            {
                minc = lowc[j];
                p = j;
            }
        if (minc == INF)return -1;//?????
        ans += minc;
        vis[p] = true;
        for (int j = 0; j<n; j++)
            if (!vis[j] && lowc[j]>cost[p][j])
                lowc[j] = cost[p][j];
    }
    return ans;
}


???isap??

#include<iostream>
#include<algorithm>
#include<vector>
#include<deque>
#include<cstring>

using namespace std;
const int inf = 0x3f3f3f3f;
const int maxp = 10;
const int maxn = 202 * 2;
int S[maxn][maxp];
int Q[maxn];
int D[maxn][maxp];
struct edge
{
    int from, to, flow, cap;
};
struct isap
{
    int n, m, s, t;
    vector<int> G[maxn];
    vector<edge> es;
    bool vis[maxn];
    int num[maxn];
    int cur[maxn];
    int dis[maxn];
    int p[maxn];
    isap(int n_) :n(n_)
    {

    }
    void add_edge(int u, int v, int cap)
    {
        es.push_back({ u, v, 0, cap });
        es.push_back({ v, u, 0, 0 });
        m = es.size();
        G[u].push_back(m - 2);
        G[v].push_back(m - 1);

    }
    void bfs()
    {
        deque<int> q;
        for (int i = 0; i < n; i++)
        {
            dis[i] = n;
        }
        memset(vis, 0, sizeof(vis));
        q.push_back(t);
        dis[t] = 0;
        vis[t] = true;
        while (!q.empty())
        {
            int pos = q.front();
            q.pop_front();

            for (int i = 0; i < G[pos].size(); i++)
            {
                edge & e = es[G[pos][i] ^ 1];
                if (e.cap > e.flow && !vis[e.from])
                {
                    dis[e.from] = dis[pos] + 1;
                    vis[e.from] = true;
                    q.push_back(e.from);
                }
            }

        }

    }
    int aug()
    {
        int x = t;
        int a = inf;
        while (x != s)
        {
            edge & e = es[p[x]];
            a = min(a, e.cap - e.flow);
            x = e.from;
        }
        x = t;
        while (x != s)
        {
            es[p[x]].flow += a;
            es[p[x] ^ 1].flow -= a;
            x = es[p[x]].from;
        }
        return a;
    }
    int maxflow(int s, int t)
    {
        int flow = 0;
        this->s = s;
        this->t = t;
        bfs();
        memset(num, 0, sizeof(num));
        for (int i = 0; i < n; i++)
        {
            num[dis[i]]++;
        }
        int x = s;
        memset(cur, 0, sizeof(cur));
        while (dis[s] < n)
        {
            if (x == t)
            {
                flow += aug();
                x = s;
            }
            int ok = 0;
            for (int i = cur[x]; i < G[x].size(); i++)
            {
                edge & e = es[G[x][i]];
                if (e.cap > e.flow && dis[x] == dis[e.to] + 1)//Advance
                {
                    ok = 1;
                    p[e.to] = G[x][i];
                    cur[x] = i;
                    x = e.to;
                    break;
                }
            }
            if (!ok)
            {
                int m = n - 1;
                for (int i = 0; i < G[x].size(); i++)
                {
                    edge &e = es[G[x][i]];
                    if (e.cap > e.flow)
                    {
                        m = min(m, dis[e.to]);
                    }
                }
                if (--num[dis[x]] == 0)
                {
                    break;
                }
                num[dis[x] = m + 1]++;
                cur[x] = 0;
                if (x != s)
                {
                    x = es[p[x]].from;
                }
            }
        }
        return flow;
    }
    vector<vector<int> > path(int flow)
    {
        vector<vector<int> > ans;
        int x = s;
        memset(cur, 0, sizeof(cur));
        while (flow)
        {
            if (x == t)
            {
                vector<int> pt;
                int a = inf;
                while (x != s)
                {
                    if(!(x & 1))
                    pt.push_back((x >> 1)+1);
                    a = min(a, es[p[x]].flow);
                    x = es[p[x]].from;
                }
                x = t;
                while (x != s)
                {
                    es[p[x]].flow -= a;
                    x = es[p[x]].from;
                }
                flow -= a;
                pt.push_back(a);
                ans.push_back(pt);

            }
            int ok = 0;
            for (int i = cur[x]; i < G[x].size(); i++)
            {
                edge & e = es[G[x][i]];
                if (e.flow > 0)
                {
                    p[e.to] = G[x][i];
                    ok = 1;
                    cur[x] = i;
                    x = e.to;
                    break;
                }
            }
            if (!ok)
            {
                cur[x] = 0;
                if(x != s)
                x = es[p[x]].from;
            }
        }
        return ans;
    }
};
```

????

```
#include<stdio.h>        //?????????
#include<string.h>
#include<vector>
#include<queue>
#include<algorithm>
using namespace std;
```

```
const int maxm=10000+100;    //????
const int INF=0x3f3f3f3f;

struct edge{        //??????????????????
    int from,to,c,f,cost;
    edge(int a,int b,int m,int n,int p):from(a),to(b),c(m),f(n),cost(p){}
};

int aabs(int a){
    return a>=0?a:-a;
}

struct MCMF{
    int m,s,t;
    vector<edge>e;
    vector<int>g[maxm];
    int dis[maxm],a[maxm],p[maxm];
    bool vis[maxm];

    void init(int n){        //?????
        for(int i=0;i<=n;i++)g[i].clear();
        e.clear();
    }

    void add(int a,int b,int c,int v){    //????
        e.push_back(edge(a,b,c,0,v));
        e.push_back(edge(b,a,0,0,-v));
        m=e.size();
        g[a].push_back(m-2);
        g[b].push_back(m-1);
    }

    bool spfa(int& flow,int& cost){
        memset(dis,0x3f,sizeof(dis));
        memset(vis,0,sizeof(vis));
        queue<int>q;
        q.push(s);
        vis[s]=1;
        dis[s]=0;
        p[s]=0;
        a[s]=INF;
        while(!q.empty()){
            int u=q.front();q.pop();
            vis[u]=0;
            for(int i=0;i<g[u].size();i++){
                edge tmp=e[g[u][i]];
                if(dis[tmp.to]>dis[u]+tmp.cost&&tmp.c>tmp.f){
                    dis[tmp.to]=dis[u]+tmp.cost;
                    p[tmp.to]=g[u][i];
                    a[tmp.to]=min(a[u],tmp.c-tmp.f);
                    if(!vis[tmp.to]){
                        q.push(tmp.to);
                        vis[tmp.to]=1;
                    }
                }
            }
        }
        if(dis[t]==INF)return 0;
        flow+=a[t];
        cost+=dis[t]*a[t];
        int u=t;
        while(u!=s){
            e[p[u]].f+=a[t];
            e[p[u]^1].f-=a[t];
            u=e[p[u]].from;
        }
        return 1;
    }

    int MF(int s,int t){
        this->s=s;this->t=t;
        int flow=0,cost=0;
        while(spfa(flow,cost));
        return cost;
    }

};
```

rmq

```
//sparse table version

#include<algorithm>
#include<cstdio>
using namespace std;

const int MAXN = 50010;
int dp[MAXN][20];
int dp2[MAXN][20];
int mm[MAXN];
//???RMQ, b?????1????0??????
void initRMQ(int n,int b[])
{
    mm[0] = -1;
    for(int i = 1; i <= n; i++)
    {
        mm[i] = ((i&(i-1)) == 0)?mm[i-1]+1:mm[i-1];
        dp2[i][0] = dp[i][0] = b[i];

    }
    for(int j = 1; j <= mm[n]; j++)
        for(int i = 1; i + (1<<j) -1 <= n; i++)
            {
                dp[i][j] = max(dp[i][j-1],dp[i+(1<<(j-1))][j-1]);
                dp2[i][j] = min(dp2[i][j-1],dp2[i+(1<<(j-1))][j-1]);
            }
}
//?????,x y inclusive>?
int rmq(int x,int y)
{
    int k = mm[y-x+1];
    return max(dp[x][k],dp[y-(1<<k)+1][k]) - min(dp2[x][k],dp2[y-(1<<k)+1][k]);
}

int a[MAXN];
int main()
{
    int n,m,x,y;
    while(scanf("%d%d",&n,&m)!=EOF)
    {
        for(int i = 0; i < n; i++)
        {
            scanf("%d",a +i);
        }
        initRMQ(n, a - 1);
        for(int i = 0; i < m; i++)
        {
            scanf("%d%d", &x,&y);
            printf("%d\n",rmq(x,y));
        }
    }
}
```

```
???????/???????????
const int inf = 0x3f3f3f3f;
const int maxn = 100000 + 10;
int minv[maxn];
int v0;
int * pv;
void init(int n)
{
    v0 = 1;
    while(v0 < n)
    {
        v0 <<= 1;
    }
    pv = minv + v0;
}

//l, r, ql, qr ,all inclusive
int query(int o, int l, int r, int ql, int qr)
{
    int mid = l + (r - l) / 2;
    int ans = inf;
    if(ql <= l && r <= qr)
    {
        return minv[o];
    }
    if(ql <= mid)
    {
        ans = min(ans, query(o << 1, l, mid, ql, qr));
    }
    if(M < qr)
    {
        ans = min(ans, query(o * 2 + 1, mid + 1, r, ql, qr));
    }
    return ans;
}
//Point update; l, r inclusive
void update(int o, int l, int r, int p, int v);
{
    int mid = l + (r - l)/2;
    if(l == r)
    {
        minv[o] = v;
    }
    else
    {
        if(p <= mid)
        {
            update(o * 2, l, mid, p, v);
        }
        else
        {
            update(o * 2 + 1, mid + 1, r, p, v);
        }
        minv[o] = min(minv[o * 2 + 1], minv[o * 2]);
    }
}

void maintain(int o, int l, int r)
{
    int lc = o * 2;
    int rc = o * 2 + 1;
   // sumv[o] = minv[lc] = maxv[o] = 0;
    if(r > l)
    {
        sumv[o] = sumv[lc] + sumv[rc];
        minv[o] = min(minv[lc], minv[rc]);
        maxv[o] = max(maxv[lc], maxv[rc]);
    }
    minv[o] += addv[o];
    maxv[o] += addv[o];
    sumv[o] += addv[o] * (R - L + 1);
}

void update(int o, int l, int r, int v)
{
    int lc = o * 2;
    int rc = o * 2 + 1;
    if(y1 <= l && y2 >= r)
    {
        add[o] += v;
    }
    else
    {
        int mid = l +(r - l) / 2;
        if( y1 <= mid)
        {
            update(lc, l, mid);
        }
        if(y2 > mid)
        {
            update(rc, mid + 1, r);
        }
        maintain(o, l, r);
    }
}
int _min, _max, _sum;
void query(int o, int l, int r, int add, int y1, int y2)
{
    if(y1 <= l && y2 >= r)
    {
        _sum += sumv[o] + add * (r - l + 1);
        _min = min(_min, minv[o] + add);
        _max = max(_max, minv[o] + add);
    }
    else
    {
        int mid = l + (r - l) / 2;
        if(y1 <= mid)
        {
            query(o * 2, l, mid, add + addv[o], y1, y2);
        }
        if(y2 > mid)
        {
            query(o * 2 + 1, mid + 1, r, add + addv[o], y1, y2);
        }
    }
}

void update(int o, int l, int r)
{
    int lc = o * 2, rc = o * 2 + 1;
    if()
}
```