😃

模板：动态规划

最长上升子序列

```
#include <iostream>
#include <algorithm>
```

```
using namespace std;

const int N = 1010;

int n;
int a[N], f[N];

int main()
{
    scanf("%d", &n);
    for (int i = 1; i <= n; i ++ ) scanf("%d", &a[i]);

    for (int i = 1; i <= n; i ++ )
    {
        f[i] = 1; // 只有a[i]一个数
        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]);

    printf("%d\n", res);

    return 0;
}
```

最长上升子序列II

```
#include <iostream>
#include <algorithm>

using namespace std;

const int N = 100010;

int n;
int a[N];
int q[N];

int main()
{
    scanf("%d", &n);
    for (int i = 0; i < n; i ++ ) scanf("%d", &a[i]);

    int len = 0;
    for (int i = 0; i < n; i ++ )
    {
        int l = 0, r = len;
        while (l < r)
        {
            int mid = l + r + 1 >> 1;
            if (q[mid] < a[i]) l = mid;
            else r = mid - 1;
        }
        len = max(len, r + 1);
        q[r + 1] = a[i];
    }
    printf("%d\n", len);

    return 0;
}
```

最长公共子序列

```
#include <iostream>
#include <algorithm>

using namespace std;

const int N = 1010;

int n, m;
char a[N], b[N];
int f[N][N];

int main()
{
    scanf("%d%d", &n, &m);
    scanf("%s%s", a + 1, b + 1);

    for (int i = 1; i <= n; i ++ )
        for (int j = 1; j <= m; j ++ )
        {
            f[i][j] = max(f[i - 1][j], f[i][j - 1]);
            if (a[i] == b[j]) f[i][j] = max(f[i][j], f[i - 1][j - 1] + 1);
        }

    printf("%d\n", f[n][m]);

    return 0;
}
```

01背包

```
#include <iostream>
#include <algorithm>

using namespace std;

const int N = 1010;

int n, m;
int v[N], w[N];
int f[N];

int main()
{
    cin >> n >> m;

    for (int i = 1; i <= n; i ++ ) cin >> v[i] >> w[i];
    for (int i = 1; i <= n; i ++ )
        for (int j = m; j >= v[i]; j -- )
            f[j] = max(f[j], f[j - v[i]] + w[i]);

    cout << f[m] << endl;

    return 0;
}
```

完全背包

```
#include <iostream>
#include <algorithm>

using namespace std;

const int N = 1010;

int n, m;
int v[N], w[N];
int f[N];

int main()
{
    cin >> n >> m;
    for (int i = 1; i <= n; i ++ ) cin >> v[i] >> w[i];

    for (int i = 1; i <= n; i ++ )
        for (int j = v[i]; j <= m; j ++ )
            f[j] = max(f[j], f[j - v[i]] + w[i]);

    cout << f[m] << endl;

    return 0;
}
```

多重背包

```
#include <iostream>
#include <algorithm>

using namespace std;

const int N = 110;

int n, m;
int v[N], w[N], s[N];
int f[N][N];

int main()
{
    cin >> n >> m;

    for (int i = 1; i <= n; i ++ ) cin >> v[i] >> w[i] >> s[i];
    for (int i = 1; i <= n; i ++ )
        for (int j = 0; j <= m; j ++ )
            for (int k = 0; k <= s[i] && k * v[i] <= j; k ++ )
                f[i][j] = max(f[i][j], f[i - 1][j - v[i] * k] + w[i] * k);

    cout << f[n][m] << endl;
    return 0;
}
```

多重背包II 二进制优化

```
#include <iostream>
#include <algorithm>

using namespace std;

const int N = 110;

int n, m;
int v[N], w[N], s[N];
int f[N][N];

int main()
{
    cin >> n >> m;

    for (int i = 1; i <= n; i ++ ) cin >> v[i] >> w[i] >> s[i];

    for (int i = 1; i <= n; i ++ )
        for (int j = 0; j <= m; j ++ )
            for (int k = 0; k <= s[i] && k * v[i] <= j; k ++ )
                f[i][j] = max(f[i][j], f[i - 1][j - v[i] * k] + w[i] * k);

    cout << f[n][m] << endl;
    return 0;
}
```

多组背包

```
#include <iostream>
#include <algorithm>

using namespace std;

const int N = 110;

int n, m;
int v[N][N], w[N][N], s[N];
int f[N];

int main()
{
    cin >> n >> m;

    for (int i = 1; i <= n; i ++ )
    {
        cin >> s[i];
        for (int j = 0; j < s[i]; j ++ )
            cin >> v[i][j] >> w[i][j];
    }

    for (int i = 1; i <= n; i ++ )
        for (int j = m; j >= 0; j -- )
            for (int k = 0; k < s[i]; k ++ )
                if (v[i][k] <= j)
                    f[j] = max(f[j], f[j - v[i][k]] + w[i][k]);

    cout << f[m] << endl;

    return 0;
}
```

多重背包III 单调队列优化

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

using namespace std;

const int N = 20010;

int n, m;
int f[N], g[N], q[N];

int main()
{
    cin >> n >> m;
    for (int i = 0; i < n; i ++ )
    {
        int v, w, s;
        cin >> v >> w >> s;
        memcpy(g, f, sizeof f);
        for (int j = 0; j < v; j ++ )
        {
            int hh = 0, tt = -1;
            for (int k = j; k <= m; k += v)
            {
                if (hh <= tt && q[hh] < k - s * v) hh ++ ;
                while (hh <= tt && g[q[tt]] - (q[tt] - j) / v * w <= g[k] - (k - j) / v * w) tt -- ;
                q[ ++ tt] = k;
                f[k] = g[q[hh]] + (k - q[hh]) / v * w;
            }
        }
    }

    cout << f[m] << endl;

    return 0;
}
```

二维费用的背包问题

```cpp
#include <iostream>

using namespace std;

const int N = 110;

int n, V, M;
int f[N][N];

int main()
{
    cin >> n >> V >> M;

    for (int i = 0; i < n; i ++ )
    {
        int v, m, w;
        cin >> v >> m >> w;
        for (int j = V; j >= v; j -- )
            for (int k = M; k >= m; k -- )
                f[j][k] = max(f[j][k], f[j - v][k - m] + w);
    }

    cout << f[V][M] << endl;

    return 0;
}
```

背包问题求方案数

```cpp
#include <cstring>
#include <iostream>

using namespace std;

const int N = 1010, mod = 1e9 + 7;

int n, m;
int f[N], g[N];

int main()
{
    cin >> n >> m;

    memset(f, -0x3f, sizeof f);
    f[0] = 0;
    g[0] = 1;

    for (int i = 0; i < n; i ++ )
    {
        int v, w;
        cin >> v >> w;
        for (int j = m; j >= v; j -- )
        {
            int maxv = max(f[j], f[j - v] + w);
            int s = 0;
            if (f[j] == maxv) s = g[j];
            if (f[j - v] + w == maxv) s = (s + g[j - v]) % mod;
            f[j] = maxv, g[j] = s;
        }
    }

    int res = 0;
    for (int i = 1; i <= m; i ++ )
        if (f[i] > f[res])
            res = i;

    int sum = 0;
    for (int i = 0; i <= m; i ++ )
        if (f[i] == f[res])
            sum = (sum + g[i]) % mod;

    cout << sum << endl;

    return 0;
}
```

背包问题求具体方案

```
#include <iostream>

using namespace std;

const int N = 1010;

int n, m;
int v[N], w[N];
int f[N][N];

int main()
{
    cin >> n >> m;
    for (int i = 1; i <= n; i ++ ) cin >> v[i] >> w[i];

    for (int i = n; i >= 1; i -- )
        for (int j = 0; j <= m; j ++ )
        {
            f[i][j] = f[i + 1][j];
            if (j >= v[i]) f[i][j] = max(f[i][j], f[i + 1][j - v[i]] + w[i]);
        }

    int j = m;
    for (int i = 1; i <= n; i ++ )
        if (j >= v[i] && f[i][j] == f[i + 1][j - v[i]] + w[i])
        {
            cout << i << ' ';
            j -= v[i];
        }

    return 0;
}
```

有依赖的背包问题

```
#include <cstring>
#include <iostream>
```

```
#include <algorithm>

using namespace std;

const int N = 110;

int n, m;
int v[N], w[N];
int h[N], e[N], ne[N], idx;
int f[N][N];

void add(int a, int b)
{
    e[idx] = b, ne[idx] = h[a], h[a] = idx ++ ;
}

void dfs(int u)
{
    for (int i = h[u]; ~i; i = ne[i])   // 循环物品组
    {
        int son = e[i];
        dfs(e[i]);

        // 分组背包
        for (int j = m - v[u]; j >= 0; j -- )  // 循环体积
            for (int k = 0; k <= j; k ++ )  // 循环决策
                f[u][j] = max(f[u][j], f[u][j - k] + f[son][k]);
    }

    // 将物品u加进去
    for (int i = m; i >= v[u]; i -- ) f[u][i] = f[u][i - v[u]] + w[u];
    for (int i = 0; i < v[u]; i ++ ) f[u][i] = 0;
}

int main()
{
    cin >> n >> m;

    memset(h, -1, sizeof h);
    int root;
    for (int i = 1; i <= n; i ++ )
    {
        int p;
        cin >> v[i] >> w[i] >> p;
        if (p == -1) root = i;
        else add(p, i);
    }

    dfs(root);

    cout << f[root][m] << endl;

    return 0;
}
```

混合背包问题

```
#include <iostream>

using namespace std;
```

```
const int N = 1010;

int n, m;
int f[N];

int main()
{
    cin >> n >> m;

    for (int i = 0; i < n; i ++ )
    {
        int v, w, s;
        cin >> v >> w >> s;
        if (!s)
        {
            for (int j = v; j <= m; j ++ )
                f[j] = max(f[j], f[j - v] + w);
        }
        else
        {
            if (s == -1) s = 1;
            for (int k = 1; k <= s; k *= 2)
            {
                for (int j = m; j >= k * v; j -- )
                    f[j] = max(f[j], f[j - k * v] + k * w);
                s -= k;
            }
            if (s)
            {
                for (int j = m; j >= s * v; j -- )
                    f[j] = max(f[j], f[j - s * v] + s * w);
            }
        }
    }

    cout << f[m] << endl;

    return 0;
}
```

状态压缩DP:蒙德里安的梦想

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

using namespace std;

const int N = 12, M = 1 << N;

int n, m;
long long f[N][M];
bool st[M];

int main()
{
    while (cin >> n >> m, n || m)
    {
        for (int i = 0; i < 1 << n; i ++ )
        {
            int cnt = 0;
            st[i] = true;
            for (int j = 0; j < n; j ++ )
                if (i >> j & 1)
                {
                    if (cnt & 1) st[i] = false;
                    cnt = 0;
                }
                else cnt ++ ;
            if (cnt & 1) st[i] = false;
        }

        memset(f, 0, sizeof f);
        f[0][0] = 1;
        for (int i = 1; i <= m; i ++ )
            for (int j = 0; j < 1 << n; j ++ )
                for (int k = 0; k < 1 << n; k ++ )
                    if ((j & k) == 0 && st[j | k])
                        f[i][j] += f[i - 1][k];

        cout << f[m][0] << endl;
    }
    return 0;
}
```

状态压缩DP:炮兵阵地

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

using namespace std;

const int N = 10, M = 1 << 10;

int n, m;
int g[1010];
int f[2][M][M];
vector<int> state;
int cnt[M];

bool check(int state)
{
    for (int i = 0; i < m; i ++ )
        if ((state >> i & 1) && ((state >> i + 1 & 1) || (state >> i + 2 & 1)))
            return false;
    return true;
}

int count(int state)
{
    int res = 0;
    for (int i = 0; i < m; i ++ )
        if (state >> i & 1)
            res ++ ;
    return res;
}
```

```
int main()
{
    cin >> n >> m;
    for (int i = 1; i <= n; i ++ )
        for (int j = 0; j < m; j ++ )
        {
            char c;
            cin >> c;
            g[i] += (c == 'H') << j;
        }

    for (int i = 0; i < 1 << m; i ++ )
        if (check(i))
        {
            state.push_back(i);
            cnt[i] = count(i);
        }

    for (int i = 1; i <= n; i ++ )
        for (int j = 0; j < state.size(); j ++ )
            for (int k = 0; k < state.size(); k ++ )
                for (int u = 0; u < state.size(); u ++ )
                {
                    int a = state[j], b = state[k], c = state[u];
                    if (a & b | a & c | b & c) continue;
                    if (g[i] & b | g[i - 1] & a) continue;
                    f[i & 1][j][k] = max(f[i & 1][j][k], f[i - 1 & 1][u][j] + cnt[b]);
                }

    int res = 0;
    for (int i = 0; i < state.size(); i ++ )
        for (int j = 0; j < state.size(); j ++ )
            res = max(res, f[n & 1][i][j]);

    cout << res << endl;

    return 0;
}
```

区间DP:环形石子合并

```
#include<iostream>
#include<cstring>
using namespace std;
#define endl '\n'

const int N=410;
const int INF=0x3f3f3f3f;
int pile[N];//n堆石子
int dp[N][N];//合并区间[l-r]的最小值
int pd[N][N];//合并区间[l-r]的最大值
int prefix[N];

int main(){
    ios::sync_with_stdio(0);
    cin.tie(0),cout.tie(0);
    int n;
    cin>>n;
    for(int i=1;i<=n;i++) cin>>pile[i],pile[i+n]=pile[i];
    for(int i=1;i<=n*2;i++) prefix[i]=prefix[i-1]+pile[i];
    memset(dp,0x3f,sizeof dp);
    for(int i=1;i<=2*n;i++) dp[i][i]=0;
    for(int len=2;len<=n;len++){
        for(int l=1;l+len-1<=n*2;l++){
            int r=l+len-1;
            int sum=prefix[r]-prefix[l-1];
            for(int i=l;i<r;i++){
                dp[l][r]=min(dp[l][r],dp[l][i]+dp[i+1][r]+sum);
                pd[l][r]=max(pd[l][r],pd[l][i]+pd[i+1][r]+sum);
            }
        }
    }
    int ans1=INF,ans2=0;
    for(int i=1;i<=n+1;i++) ans1=min(ans1,dp[i][i+n-1]),ans2=max(ans2,pd[i][i+n-1]);
    cout<<ans1<<endl<<ans2;
    return 0;
}
```

区间DP:棋盘分割

```
#include<iostream>
#include<cmath>
#include<iomanip>
using namespace std;

const int N=15,M=10;
const int INF=0x3f3f3f3f;
int m=8,n;
int matrix[M][M];
int dp[M][M][M][M][N];//在一个区块内切了k刀的方案集合
int prefix[M][M];//二维前缀和
double x_;//平均数

//求一个区块中数值的和的平方
int chunk_sum(int r1,int c1,int r2,int c2){
    int res=prefix[r2][c2]-prefix[r1-1][c2]-prefix[r2][c1-1]+prefix[r1-1][c1-1];
    return res*res;
}

//记忆化搜索一个区块切k刀的解
int dfs(int r1,int c1,int r2,int c2,int k){
    int &d=dp[r1][c1][r2][c2][k];
    if(d==0) d=INF;
    else return d;

    if(k==0) return d=chunk_sum(r1,c1,r2,c2);

    for(int i=r1+1;i<=r2;i++){
        //向下搜索
        d=min(d,dfs(i,c1,r2,c2,k-1)+chunk_sum(r1,c1,i-1,c2));
        //向上搜索
        d=min(d,dfs(r1,c1,i-1,c2,k-1)+chunk_sum(i,c1,r2,c2));
    }
    for(int i=c1+1;i<=c2;i++){
        //向右搜索
        d=min(d,dfs(r1,i,r2,c2,k-1)+chunk_sum(r1,c1,r2,i-1));
        //向左搜索
        d=min(d,dfs(r1,c1,r2,i-1,k-1)+chunk_sum(r1,i,r2,c2));
    }
    return d;
}

int main(){
    ios::sync_with_stdio(0);
    cin.tie(0),cout.tie(0);
    cin>>n;
    for(int i=1;i<=m;i++) for(int j=1;j<=m;j++){
        cin>>matrix[i][j];
        prefix[i][j]=prefix[i-1][j]+prefix[i][j-1]-prefix[i-1][j-1]+matrix[i][j];
    }
    x_=prefix[m][m]*1./n;
    double ans=dfs(1,1,m,m,n-1);
    ans=sqrt(ans/n-pow(x_,2));
    cout<<fixed<<setprecision(3)<<ans;
    return 0;
}
```

树形DP:皇宫看守

```
#include<iostream>
#include<vector>
using namespace std;

typedef vector<int> VI;
typedef long long LL;

const int N=1510;
const int INF=0x3f3f3f3f;
int n;
VI tree[N];
int cost[N];
LL dp[N][3];//#1选 #2不选但有子树选 #3不选且无子树选

void dfs(int node,int root){
    dp[node][0]+=cost[node];
    LL minn=INF;
    for(int i:tree[node]){
        if(i==root) continue;
        dfs(i,node);
        dp[node][0]+=min(dp[i][0],min(dp[i][1],dp[i][2]));
        minn=min(minn,dp[i][0]-min(dp[i][0],dp[i][1]));
        dp[node][1]+=min(dp[i][0],dp[i][1]);
        dp[node][2]+=dp[i][1];
    }
    dp[node][1]+=minn;
}

int main(){
    ios::sync_with_stdio(0);
    cin.tie(0),cout.tie(0);
    cin>>n;
    for(int i=1;i<=n;i++){
        int id,m,r;
        cin>>id>>cost[id]>>m;
        while(m--){
            cin>>r;
            tree[id].push_back(r);
            tree[r].push_back(id);
        }
    }
    dfs(1,-1);
    cout<<min(dp[1][0],dp[1][1]);
    return 0;
}
```

数位DP:不要62

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

using namespace std;

const int N = 35;

int f[N][10];

void init()
{
    for (int i = 0; i <= 9; i ++ )
        if (i != 4)
            f[1][i] = 1;

    for (int i = 1; i < N; i ++ )
        for (int j = 0; j <= 9; j ++ )
        {
            if (j == 4) continue;
            for (int k = 0; k <= 9; k ++ )
            {
                if (k == 4 || j == 6 && k == 2) continue;
                f[i][j] += f[i - 1][k];
            }
        }
}

int dp(int n)
{
    if (!n) return 1;

    vector<int> nums;
    while (n) nums.push_back(n % 10), n /= 10;

    int res = 0;
    int last = 0;
    for (int i = nums.size() - 1; i >= 0; i -- )
    {
        int x = nums[i];
        for (int j = 0; j < x; j ++ )
        {
            if (j == 4 || last == 6 && j == 2) continue;
            res += f[i + 1][j];
        }

        if (x == 4 || last == 6 && x == 2) break;
        last = x;

        if (!i) res ++ ;
    }
    return res;
}

int main()
{
    init();

    int l, r;
    while (cin >> l >> r, l || r)
    {
        cout << dp(r) - dp(l - 1) << endl;
    }

    return 0;
}
```

单调队列优化DP:绿色通道

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

const int INF=0x3f3f3f3f;
const int N=5e4+10;
int n,t;
int cost[N];
int dp[N];//前i位都合法且选第i位的方案
int que[N];

bool check(int m){//判断当前空段是否可达
    int hh=0,tt=0;
    for(int i=1;i<=n;i++){//单调队列维护dp
        if(i-que[hh]>m+1) hh++;
        dp[i]=dp[que[hh]]+cost[i];
        while(dp[i]<=dp[que[tt]]&&tt>=hh) tt--;
        que[++tt]=i;
    }

    int res=INF;
    for(int i=n;i>=1;i--) if(i+m>=n) res=min(dp[i],res); else break;
    if(res<=t) return true; else return false;
}

int main(){
    ios::sync_with_stdio(0);
    cin.tie(0),cout.tie(0);
    cin>>n>>t;
    for(int i=1;i<=n;i++) cin>>cost[i];

    int l=0,r=n;
    while(l<r){//二分最大空段
        int mid=l+r>>1;
        if(check(mid)) r=mid;
        else l=mid+1;
    }

    cout<<r;
    return 0;
}
```