模板

目录

数学

二分

```
//区间[l, r]被划分成[l, mid]和[mid + 1, r]时使用：
int bsearch_1(int l, int r)
{
    while(l<r)
    {
        int mid = l + r >> 1;
        if(check(mid)) r = mid;
        else l = mid + 1;
    }
    return l;
}
//区间[l, r]被划分成[l, mid - 1]和[mid, r]时使用：
int bsearch_2(int l, int r)
{
    while(l<r)
    {
        int mid = l + r + 1 >> 1;
        if(check(mid)) l = mid;
        else r = mid - 1;
    }
    return l;
}
```

高精度加法

```
vector<int> add(vector<int> &A, vector<int> &B)
{
    vector<int> C;

    int t = 0;//进位
    for(int i=0;i<A.size()||i<B.size();i++)
    {
        if(i<A.size()) t += A[i];
        if(i<B.size()) t += B[i];
        C.push_back(t%10);
        t /= 10;
    }
    if(t) C.push_back(t);
    return C;
}
```

快速幂

```
using i64 = long long;
i64 quick_pow(i64 a,i64 k,i64 p)
{
    ll res=1%p;
    while(k)
    {
        if(k&1) res=res*a%p;
        k>>=1;
        a=a*a%p;
    }
    return res;
}
vector<vector<int>> mul(vector<vector<int>>& a, vector<vector<int>>& b, int
mod)
{
    int n = a.size();
    vector<vector<int>> res(n, vector<int>(n));
    for (int i = 0; i < a.size(); i++)
        for (int j = 0; j < a[i].size(); j++)
            for (int k = 0; k < a[i].size(); k++)
                res[i][j] = (res[i][j] + a[i][k] * b[k][j]) % mod;
    return res;
}
vector<vector<int>> qp(vector<vector<int>>& a, int b, int mod)
{
    int n = a.size();
    vector<vector<int>> res(n, vector<int>(n));
    for(int i=0;i<a.size();i ++) res[i][i] = 1;
    while(b)
    {
        if(b & 1) res = mul(res, a, mod);
        b >>= 1;
        a = mul(a, a, mod);
    }
    return res;
}
```

欧拉筛

```
//欧拉函数定义：对于整数n，在1~n-1中与n互质的数的个数
#include <iostream>
using namespace std;

#define ll long long
const int N=1000010;

int cnt,primes[N];
int phi[N];
bool st[N];

ll get_eulers(int n)
{
    phi[1]=1;
    for(int i=2;i<=n;i++)
    {
        if(!st[i])
        {
            primes[cnt++]=i;
            phi[i]=i-1;//质数p的欧拉函数为p-1
        }
        for(int j=0;primes[j]<=n/i;j++)
        {
            st[primes[j]*i]=true;
            if(i% primes[j]==0)
            {
                phi[i*primes[j]]=primes[j]*phi[i];
                break;
            }
            phi[i*primes[j]]=phi[i]*(primes[j]-1);
        }
    }
}
```

埃氏筛法

```
const int N=1e9+10;

int cnt,primes[N/10];
bool st[N];

void get_prime(int n)
{
    for(int i=2;i<=n;i++)
    {
        if(!st[i])
        {
            primes[cnt++]=i;
            for(int j=i*i;j > 0 && j<=n;j+=i)
                st[j]=true;
        }
    }
}
```

计算线段与圆的交点数量

```
using i64 = long long;
using i128 = __int128;

i128 dist(i64 x1, i64 y1, i64 x2, i64 y2) {
    return ((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
}

int calc(i64 x1, i64 y1, i64 x2, i64 y2, i64 x, i64 y, i64 r) {
    bool flag1 = false, flag2 = false, flag3 = false, flag4 = false;
    if (dist(x1, y1, x, y) < r * r) flag1 = true;  // 圆内
    if (dist(x2, y2, x, y) < r * r) flag2 = true;
    if (dist(x1, y1, x, y) == r * r) flag3 = true;  // 圆上
    if (dist(x2, y2, x, y) == r * r) flag4 = true;
    if (flag1 and flag2) {
        return 0;  // 两点都在圆内
    } else if (flag1 or flag2) {
        return 1;  // 一点在圆内，一点不在，必有一个交点
    }

    i128 A = dist(x1, y1, x2, y2) * r * r;
    // 叉乘结果的值替代absinθ
    i128 B = (x1 - x) * (y2 - y) - (x2 - x) * (y1 - y);
    B = B * B;
    // 圆心到直线距离大于半径，肯定没有交点
    if (B > A) {
        return 0;
    }

    i128 angle1 = (x - x1) * (x2 - x1) + (y - y1) * (y2 - y1);
    i128 angle2 = (x - x2) * (x1 - x2) + (y - y2) * (y1 - y2);

    // 如果两个角都是锐角，说明此时直线和圆的关系是相切或相交
    if (angle1 > 0 and angle2 > 0) {
        if (A == B) {
            return 1;  // 相切
        } else {
            return 2;  // 相交
        }
    } else {
        // 否则如果存在至少一个圆上的点也可以说明是有一个交点的
        if (flag3 or flag4) {
            return 1;
        } else {
            return 0;  // 相离
        }
    }
    return 0;
}
```

图论

Dijkstra1

```
using pii = pair<int, int>;
void dijkstra()
{
    memset(dist, 0x3f, sizeof(dist));
    dist[s] = 0;
    priority_queue<pii, vector<pii>, greater<pii>> q;
    q.push({0, s});
    while(q.size())
    {
        int ver = q.top().second;
        int dis = q.top().first;
        q.pop();
        if(st[ver]) continue;
        st[ver] = true;
```

```
            for(auto p:g[ver])
            {
                int v = p.first;
                dist[v] = min(dist[v], dist[ver]+p.second);
                if(!st[v]) q.push({dist[v], v});
            }
        }
    }
```

Dijkstra2

```
const int N = 510;
int n, m, g[N][N], dist[N];
bool st[N];

int dijkstra()
{
    memset(dist, 0x3f, sizeof(dist));
    dist[1] = 0;
    for (int i = 0; i < n; i++)
    {
        int t = -1;
        for (int j = 1; j <= n; j ++)
            if(!st[j] && (t == -1 || dist[t] > dist[j]))
                t = j;
        //if(t == n) break;
        st[t] = true;
        for (int j = 1; j<= n; j++)
            dist[j] = min(dist[j], dist[t]+g[t][j]);
    }
    if(dist[n] == 0x3f3f3f3f) return -1;
    return dist[n];
}
```

Bellman_ford

$O(nm) $

```
const int N = 510, M = 10010;
int n, m, k;
int dist[N], backup[N];
struct Edge
{
    int a, b, w;
}edges[N];
int bellman_ford()
{
    memset(dist, 0x3f, sizeof(dist));
    dist[1] = 0;
    for (int i = 0; i < k; i++)
    {
        memcpy(backup, dist, sizeof(dist));
        for (int j = 0; j < m; j++)
        {
            int a = edges[j].a, b = edges[j].b, w = edges[j].w;
            dist[b] = min(dist[b], backup[a] + w);
        }
    }
    if (dist[n] > 0x3f3f3f3f / 2) return -1;
    return dist[n];

}
```

SPFA

```
//O(m)最坏O(nm)
#include <iostream>
#include <cstring>
#include <algorithm>
#include <queue>
using namespace std;
const int N = 1e5+10;
int n, m, h[N], e[N], ne[N], dist[N], w[N], idx;
bool st[N];
typedef pair<int, int> PII;
void add(int a, int b, int c)
{
    e[idx] = b;
    ne[idx] = h[a];
    w[idx] = c;
    h[a] = idx ++;
}

int spfa()
{
    memset(dist, 0x3f, sizeof(dist));
    dist[1] = 0;
    queue<int> q;
    q.push(1);
    st[1] = true;
    while (q.size())
    {
        int t = q.front();
        q.pop();
        st[t] = false;
        for (int i = h[t]; i != -1; i = ne[i])
        {
            int j = e[i];
            if (dist[j] > dist[t] + w[i])
            {
                dist[j] = dist[t] + w[i];
                if (!st[j])
                {
                    q.push(j);
                    st[j] = true;
                }
            }
        }
    }
    if (dist[n] == 0x3f3f3f3f) return -1;
    return dist[n];

}
int main()
{
    scanf("%d %d", &n, &m);
    memset(h, -1, sizeof(h));
    while (m--)
    {
        int a, b, c;
        scanf("%d %d %d", &a, &b, &c);
        add(a, b, c);
    }
    int t = spfa();
    if (t == -1) puts("impossible");
    else printf("%d\n", t);
    return 0;
}
```

Floyd

```
int d[N][N];

void floyd()
{
    for(int k=1;k<=n;k++)
        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]);

}
```

数据结构

KMP

```
const int N = 1e5+10, M = 1e6+10;
int n, m;
char p[N], s[M];
```

```
int ne[N];
//ne[i]表示前i个字符，前后缀相同情况下，最长是多长（ne[i] < i）
int main()
{
    cin >> n >> p + 1 >> m >> s + 1;

    for(int i=2, j=0;i<=n;i++)
    {
        while(j&&p[i]!=p[j+1]) j = ne[j];
        if(p[i] == p[j+1]) j ++;
        ne[i] = j;
    }

    for(int i=1, j=0;i<=m;i++)
    {
        while(j&&s[i]!=p[j+1]) j = ne[j];
        if(s[i]==p[j+1]) j++;
        if(j==n)
        {
            printf("%d ", i - n);
            j = ne[j];
        }
    }
    return 0;
}
```

字符串hash

```


#include <bits/stdc++.h>

using i64 = long long;

class HASH {
   private:
    using ui64 = unsigned long long;
    using i64 = long long;
    std::string s;
    int P1 = 131, P2 = 11311;
    ui64 mod = 1'000'000'007;
    std::vector<ui64> h1, h2, p1;
    std::vector<i64> h3, h4, p2;
    int n;

   public:
    // s的下标从0开始
    HASH(const std::string& s) {
        this->s = s;
        n = s.size();
        h1.resize(n + 1);
        h2.resize(n + 1);
        h3.resize(n + 1);
        h4.resize(n + 1);
        p1.resize(n + 1);
        p2.resize(n + 1);
        p1[0] = p2[0] = 1;
        for (int i = 1; i <= n; i++) {
            h1[i] = h1[i - 1] * P1 + s[i - 1];
            p1[i] = p1[i - 1] * P1;

            h3[i] = h3[i - 1] * P2 % mod + s[i - 1];
            h3[i] %= mod;
            p2[i] = p2[i - 1] * P2;
            p2[i] %= mod;
        }
        for (int i = n; i > 0; i--) {
            h2[n - i + 1] = h2[n - i] * P1 + s[i - 1];

            h4[n - i + 1] = h4[n - i] * P2 % mod + s[i - 1];
            h4[n - i + 1] %= mod;
        }
    }
    // l, r下标从1开始，op为0表示[l, r]的hash值，为1表示将字符串翻转后[l, r]的hash值
    std::pair<ui64, i64> get(int l, int r, int op = 0) {
        ui64 v1 = h1[r] - h1[l - 1] * p1[r - l + 1];
        i64 v3 = (h3[r] - h3[l - 1] * p2[r - l + 1] % mod + mod) % mod;
        int tl = n - r + 1, tr = n - l + 1;
        ui64 v2 = h2[tr] - h2[tl - 1] * p1[tr - tl + 1];
        i64 v4 = (h4[tr] - h4[tl - 1] * p2[tr - tl + 1] % mod + mod) % mod;
        if (!op) {
            return std::make_pair(v1, v3);
        }
        return std::make_pair(v2, v4);
    }
};
```

Trie树

```
const int N = 1e5+10;

int son[N][26], cnt[N], idx;
char str[N];

void insert(char str[])
{
    int p = 0;
    for(int i=0;str[i];i++)
    {
        int u = str[i] - 'a';
        if(!son[p][u]) son[p][u] = ++ idx;
        p = son[p][u];
    }
    cnt[p] ++;
}
int query(char str[])
{
    int p = 0;
    for(int i=0;str[i];i++)
    {
        int u = str[i] - 'a';
        if(!son[p][u]) return 0;
        p = son[p][u];
    }
    return cnt[p];

}
```

匈牙利

```

vector<vector<int>> g;
bool find(int x)
{
    for(auto j:g[x])
        if(!st[j])
        {
            st[j] = true;
            if(match[j] == 0 || find(match[j]))
            {
                match[j] = x;
                return true;
            }
        }
    return false;
}
```

ST表

```

#include <iostream>
#include <cmath>
#define lowbit(x) ((x) & (-x))
using namespace std;
const int N = 1e5+10;
int f[N][25], g[N][25];
int lg[N];
int n, q;
int main()
{
    lg[1] = 0, lg[2] = 1;
    for(int i=3;i<N;i++) if(lowbit(i) == i) lg[i] = lg[i-1] + 1; else lg[i]
= lg[i-1];
```

```
    cin >> n >> q;//n个数，q次询问
    for(int i=1;i<=n;i++)
    {
        int x;
        scanf("%d", &x);
        f[i][0] = g[i][0] = x;
    }
    int ed = 25;
    for(int j=1;j<=ed;j++)
        for(int i=1;i+(1<<j)-1 <= n;i++)
        {
            f[i][j] = max(f[i][j-1], f[i+(1<<(j-1))][j-1]);

            g[i][j] = min(g[i][j-1], g[i+(1<<(j-1))][j-1]);
        }

    while (q--)
    {
        int a, b;
        scanf("%d %d", &a, &b);
        int s = lg[b-a+1];
        //最大值与最小值的差
        printf("%d\n", max(f[a][s], f[b-(1<<s)+1][s]) - min(g[a][s], g[b-(1<<s)+1][s]));
    }

    return 0;
}
```

树状数组

```
const int N = 1e5+10;
int tr[N];
void add(int x, int c)
{
    for(int i=x;i<=n;i+=lowbit(i)) tr[i] += c;
}
char str[20];
int sum(int x)
{
    int res = 0;
    for(int i=x;i;i-=lowbit(i)) res += tr[i];
    return res;
}
```

线段树

```
//AcWing 243
//n, m <= 1e5
```

```
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <iostream>
#define int long long
using namespace std;
const int N = 1e5+10;
int n, m;
int w[N];
struct Node
{
    int l, r;
    int sum, add;
}tr[N * 4];
void pushdown(int u)
{
    auto &root = tr[u], &left = tr[u << 1], &right = tr[u << 1 | 1];
    if(root.add)
    {
        left.add += root.add, left.sum += (left.r - left.l + 1) * root.add;
        right.add += root.add, right.sum += (right.r - right.l + 1) *
root.add;
        root.add = 0;
    }
}
void pushup(int u)
{
    tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;
}
void build(int u, int l, int r)
{
    if(l == r) tr[u] = {l, r, w[r], 0};
    else
    {
        tr[u] = {l, r};
        int mid = l + r >> 1;
        build(u << 1, l, mid), build(u << 1 | 1, mid + 1, r);
        pushup(u);
    }
}
void modify(int u, int l, int r, int d)
{
    if(tr[u].l >= l && tr[u].r <= r)
    {
        tr[u].sum += (tr[u].r - tr[u].l + 1) * d;
        tr[u].add += d;
    }
    else
    {
        pushdown(u);
        int mid = tr[u].l + tr[u].r >> 1;
        if(l <= mid) modify(u << 1, l, r, d);
        if(r > mid) modify(u << 1 | 1, l, r, d);
        pushup(u);
        }
}
int query(int u, int l, int r)
{
    if(tr[u].l >= l && tr[u].r <= r) return tr[u].sum;
    pushdown(u);
    int mid = tr[u].l + tr[u].r >> 1;
    int sum = 0;
    if (l <= mid) sum = query(u << 1, l, r);
    if (r > mid) sum += query(u << 1 | 1, l, r);
    return sum;
}
signed main()
{
    scanf("%lld %lld", &n, &m);//n个数，m条操作
    for(int i=1;i<=n;i++) scanf("%lld", &w[i]);
    build(1, 1, n);
    char op[2];
    int l, r, d;
    while (m--)
    {
        scanf("%s %lld %lld", op, &l, &r);
        if(*op == 'C')
        {
            scanf("%lld", &d);
            modify(1, l, r, d);//操作 ‘C’，将区间[l,r]中的每个数加上d
        }
        else printf("%lld\n", query(1, l, r));//操作 ‘Q’，询问区间[l,r]中所有
数的和
    }

    return 0;
}
```

LCA

```
const int N = 5e5+10;
int fa[N][21];
int depth[N];
void lca(vector<vector<int>>& g, int root)
{
    queue<int> q;
    q.push(root);
    memset(depth, 0x3f, sizeof depth);
    depth[0] = 0, depth[root] = 1;
    while(!q.empty())
    {
        int ver = q.front();
        q.pop();
        for(auto x:g[ver])
        {
            if(depth[x] > depth[ver] + 1)
            {
                q.push(x);
                depth[x] = depth[ver] + 1;
                fa[x][0] = ver;
                for(int k = 1; k <= 20; k++)
                    fa[x][k] = fa[fa[x][k-1]][k-1];
            }
        }
    }
}
int query(int a, int b)//返回a，b的最近公共祖先
{
    if(depth[a] < depth[b]) swap(a, b);
    for(int k = 20; k >= 0; k--)
        if(depth[fa[a][k]] >= depth[b])
            a = fa[a][k];
    if(a == b) return a;
    for(int k = 20; k >= 0; k--)
        if(fa[a][k] != fa[b][k])
        {
            a = fa[a][k];
            b = fa[b][k];
        }
    return fa[a][0];
}
```

PBDS使用实例

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
#include <iostream>
using namespace std;
using namespace __gnu_pbds;
int main()
{
    tree<int, null_type, less<int>, rb_tree_tag,
tree_order_statistics_node_update> tr;
    tr.insert(1);
    tr.insert(2);
    tr.insert(4);
    // 树中元素 1，2，4
    auto it = tr.insert(3); // pair<point_iterator, bool>
    cout << *it.first << endl; // 3
    cout << it.second << endl; // 1

    bool flag = tr.erase(3); // bool
    cout << flag << endl; // 1

    // 树中元素 1，2，4
    // order_of_key(x) 返回小于x的元素个数(int)
    int cnt = tr.order_of_key(3);
    cout << cnt << endl; // 2
    cnt = tr.order_of_key(2);
    cout << cnt << endl; // 1

    // 树中元素 1，2，4
    // find_by_order(k) 返回指向第k小的元素的迭代器 (从0开始)
    auto it2 = tr.find_by_order(1); // point_iterator
    cout << *it2 << endl; // 2

    auto it3 = tr.lower_bound(2); // point_iterator
    cout << *it3 << endl; // 2
    it3 = tr.upper_bound(2); // point_iterator
    cout << *it3 << endl; // 4

    tree<int, null_type, less<int>, rb_tree_tag,
tree_order_statistics_node_update> tr1;
    tr1.insert(8);
    tr1.insert(16);
    tr1.insert(32);

    // tr.join(tr1) 将tr1合并到tr中，tr1中的元素会被删除
    tr.join(tr1);
    for(auto it : tr)
        cout << it << endl; // 1 2 4 8 16 32

    tr.split(4, tr1); // 将tr中小于等于4的元素分离出来，放到tr1中
    for(auto it : tr)
        cout << it << endl; // 1, 2, 4

    return 0;
}
```

背包

01背包

```
//n, v <= 1e3
// O(nv)
#include <iostream>
#define rep(i, a, n) for(int i=(a);i<=(n);i++)
#define dec(i, n, a) for(int i=(n);i>=(a);i--)
using namespace std;
const int N = 1e3+10;
int f[N];
int n, V;
int w[N], v[N];
int main()
{
    cin >> n >> V;//n个物品，背包容量为V
    rep(i, 1, n) cin >> v[i] >> w[i];//v[i]为第i个物品的体积，w[i]为第i个物品的
价值
    rep(i, 1, V) if(v[1] <= i) f[i] = w[1];
    rep(i, 2, n)
        dec(j, V, v[i])
            f[j] = max(f[j], f[j-v[i]] + w[i]);
    cout << f[V];

    return 0;
}
```

完全背包

```
//n,v <= 1e3
//O(nv)
#include <iostream>
#define rep(i, a, n) for(int i=(a);i<=(n);i++)
#define dec(i, n, a) for(int i=(n);i>=(a);i--)
using namespace std;
const int N = 1e3+10;
int f[N];
int n, V;
int w[N], v[N];
int main()
{
    cin >> n >> V;//n个物品，背包容量为V
    rep(i, 1, n) cin >> v[i] >> w[i];//v[i]为第i个物品的体积，w[i]为第i个物品的
价值
    rep(i, 1, V) f[i] = i / v[1] * w[1];
    rep(i, 2, n)
        rep(j, v[i], V)
            f[j] = max(f[j], f[j-v[i]] + w[i]);
    cout << f[V];

    return 0;
}
```

多重背包

```
//n, v <= 1e2
//O(nv^2)
#include <iostream>
#define rep(i, a, n) for(int i=(a);i<=(n);i++)
#define dec(i, n, a) for(int i=(n);i>=(a);i--)
using namespace std;
const int N = 110;
int f[N];
int main()
{
    int n, V;
    cin >> n >> V;//n个物品，背包容量为V
    rep(i, 1, n)
    {
        int v, w, s;
        cin >> v >> w >> s;//v为第i个物品的体积，w为第i个物品的价值，s为第i个物品的数量
        dec(j, V, 1)
            rep(k, 1, min(s, j/v))
                f[j] = max(f[j], f[j-k*v]+k*w);
    }
    cout << f[V];
    return 0;
}
```

多重背包二进制优化

```
//n <= 1e3, v <= 2e3
//O(nvlogv)

#include <iostream>
#define rep(i, a, n) for(int i=(a);i<=(n);i++)
#define dec(i, n, a) for(int i=(n);i>=(a);i--)
using namespace std;
const int N = 1e3+10;
int s[N], v[N], w[N];
int v1[N*11], w1[N*11];
int f[N*2];
int main()
{
    int n, V;
    cin >> n >> V;//n个物品，背包容量为V
    int cnt = 0;
    rep(i, 1, n) cin >> v[i] >> w[i] >> s[i];//体积，价值，数量
    rep(i, 1, n)
    {
        int tmp = s[i];
        rep(j, 0, 12)
        {
            int t = (1 << j);
            if(tmp >= t) tmp -= t;
            else t = tmp, tmp = 0;
            cnt ++;
            v1[cnt] = t * v[i];
            w1[cnt] = t * w[i];
            if(!tmp) break;
        }
    }
    //二进制优化后转为01背包
    rep(i, v1[1], V) f[i] = w1[1];
    rep(i, 2, cnt)
        dec(j, V, v1[i])
```

```
                f[j] = max(f[j], f[j-v1[i]] + w1[i]);
    cout << f[V];

    return 0;
}
```

分组背包

```
//n,v, s <= 100
//O(nvs)
#include <iostream>
#define rep(i, a, n) for(int i=(a);i<=(n);i++)
#define dec(i, n, a) for(int i=(n);i>=(a);i--)
using namespace std;
int n, m;
int f[110];
int v[110], w[110];
int main()
{
    cin >> n >> m;//n组物品，背包容量为m
    rep(i, 1, n)//每组物品只能选一个或者不选（组内01背包）
    {
        int s;
        cin >> s;
        rep(j, 1, s) cin >> v[j] >> w[j];
        dec(j, m, 1)
        {
            rep(k, 1, s)
                if(j >= v[k]) f[j] = max(f[j], f[j-v[k]]+w[k]);
        }
    }
    cout << f[m] << endl;
    return 0;
}
```