实验一

动态规划

```
#include <bits/stdc++.h>
using namespace std;
void solve() {
    int V,n; cin>>V>>n; // 体积 个数
    vector<array<int,2>> a(n); // 体积 价值
    for(auto &t: a) cin>>t[0]>>t[1];
    vector<int> f(V + 1);
    for(int i = 0; i < n; ++i) {
        int v = a[i][0]; // 体积
        int w = a[i][1]; // 价值
        for(int j = V; j >= v; --j) f[j] = max(f[j], f[j - v] + w);
    }
    cout<<f[V];
}
int main()
{
    solve();
    return 0;
}
```

回溯法

```
#include <bits/stdc++.h>
using namespace std;
void solve() {
    int V,n; cin>>V>>n; // 体积 个数
    vector<array<int,2>> a(n); // 体积 价值
    int sum = 0;
    for(auto &t: a) cin>>t[0]>>t[1], sum += t[1];
    int ans = 0;
    vector<int> vis(n);
    auto dfs = [&](auto &&self, int now, int tmp) -> void {
        ans = max(ans, tmp);
        for(int i = 0; i < n; ++i) {
            int v = a[i][0], w = a[i][1];
            if(v > now || vis[i]) continue;
            vis[i] = 1;
            self(self, now - v, tmp + w);
            vis[i] = 0;
        }
    };
    dfs(dfs, V, 0);
    cout<<ans<<endl;
}
int main()
{
    solve();
```

```
    return 0;
}
```

分支界限法

```
#include <bits/stdc++.h>
using namespace std;
struct Good {
    int w, v;
    double vpw;
    Good() {};
    Good(int _w, int _v): w(_w), v(_v) {
        vpw = v * 1.0 / w;
    }
    bool operator< (const Good& rhs) {
        return rhs.vpw < this->vpw;
    }
};
vector<Good> goods;
int n,V; // 物品数量及背包体积容量
struct Node {
    int level, v,w;
    double ub;
};
double Bound(Node a) {
    if(a.w > V) return 0.0;

    double bd = a.v;
    int j = a.level + 1;
    int now = a.w;
    int last = V - now;
    bd += last * 1.0 / goods[j].w * goods[j].v;
    return bd;
}

void solve() {
    cin>>V>>n;
    goods.resize(n);
    for(auto &t: goods) {
        int w,v; cin>>w>>v;
        t = Good(w,v);
    }
    sort(goods.begin(), goods.end());
    Node st;
    st.level = -1; st.v = st.w = 0; st.ub = Bound(st);
    int ma = -3;
    queue<Node> q;
    q.push(st);
    while(q.size()) {
        auto y = q.front(); q.pop();
        Node x;
        if(y.level != n - 1) {
            x.level = y.level + 1;
        }
        // 选
        x.w = y.w + goods[x.level].w;
```

```
        x.v = y.v + goods[x.level].v;
        if(x.w <= V && x.v > ma) {
            ma = x.v;
        }
        x.ub = Bound(x);
        if(x.ub > ma) q.push(x);

        // 不选
        x.w = y.w; x.v = y.v; x.ub = Bound(x);
        if(x.ub > ma) {
            q.push(x);
        }
    }
    cout<<ma<<endl;
}
int main()
{
    solve();
    return 0;
}
```

贪心

```
#include <bits/stdc++.h>
using namespace std;
void solve() {
    int V,n; cin>>V>>n; // 体积 个数
    vector<array<int,2>> a(n); // 体积 价值
    for(auto &t: a) cin>>t[0]>>t[1];
    sort(a.begin(), a.end(), [&](auto pre, auto suf) {
        double p = pre[1] * 1.0 / pre[0], s = suf[1] * 1.0 / suf[0];
        return p > s;
    });
    int ans = 0;
    for(int i = 0; i < n; ++i) {
        if(a[i][0] > V) continue;
        V -= a[i][0]; ans += a[i][1];
    }
    cout<<ans<<endl;
}
int main()
{
    solve();
    return 0;
}
```

实验二

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

```cpp
using namespace std;
// dijkstra 堆优化
const int N = 1e5 + 21;
int dist[N];
int vis[N], h[N], e[N], ne[N],idx,w[N]; // w -- 权重
void add(int a, int b, int c) {
    e[idx] = b, ne[idx] = h[a], w[idx] = c, h[a] = idx++;
}
void solve() {
    // 建图
    memset(h, -1, sizeof(h));
    add(0, 5, 100);
    add(0, 2, 10);
    add(0, 4, 30);
    add(4, 5, 60);
    add(4, 3, 20);
    add(3, 5, 10);
    add(1, 2, 5);
    add(2, 3, 50);
    // 最短路
    memset(dist, 0x3f, sizeof(dist));
    dist[0] = 0;
    priority_queue<pair<int,int>, vector<pair<int,int>>, greater<pair<int,int>>>
heap;
    heap.push({0,0}); // 权值 节点
    while(heap.size()) {
        auto t = heap.top(); heap.pop();
        int ver = t.second, distance = t.first;
        if(vis[ver]) continue;
        vis[ver] = 1;
        for(int i = h[ver]; ~i; i = ne[i]) {
            int y = e[i];
            if(dist[y] > distance + w[i]) {
                dist[y] = distance + w[i];
                heap.push({dist[y], y});
            }
        }
    }
    for(int i = 0; i <= 5; ++i) {
        cout<<"0 -> "<<i<<" : "<<dist[i]<<(dist[i] == 0x3f3f3f3f ? " (inf)" :
"")<<endl;
    }
}
int main() {
    solve();
    return 0;
}
```

输出结果：

```
0 -> 0 : 0
0 -> 1 : 1061109567 (inf)
0 -> 2 : 10
0 -> 3 : 50
0 -> 4 : 30
0 -> 5 : 60
```

实验三

```cpp
#include <bits/stdc++.h>
using namespace std;
const int INF = 0x3f3f3f3f;
void solve() {
    int n = 1000;
    vector<int> a(n);
    srand(time(0));
    for(auto &t: a) {
        t = rand();
    }
    auto merge = [&](auto &&self, int l, int r) -> pair<int,int> { // mi, ma
        if(l == r) {
            return {a[l], a[l]};
        }
        if(l + 1 == r) {
            int mi = min(a[l], a[r]), ma = max(a[l], a[r]);
            return {mi, ma};
        }
        int mid = l + r >> 1;
        auto lv = self(self, l, mid), rv = self(self, mid + 1, r);
        int mi = min(lv.first, rv.first), ma = max(lv.second, rv.second);
        return {mi, ma};
    };
    auto ans = merge(merge, 0, n - 1);
    cout<<"min : "<<ans.first<<endl;
    cout<<"max : "<<ans.second<<endl;

    // 测试
    int ans_mi = *min_element(a.begin(), a.end()), ans_ma =
*max_element(a.begin(), a.end());
    if(ans_mi == ans.first && ans_ma == ans.second) {
        cout<<"OK"<<endl;
    }
}
int main()
{
    solve();
    return 0;
}
```