最优解包含贪心选择的证明可用反证法，证明如下：

假设最优解不含贪心选择，即最优解A的某个活动a所选择的会场r'不是结束时间最早的会场r，

由于r的结束时间小于r'，a 能在r中进行，说明a必定能在r中进行，

因此用 r代替 r'的结果$(A-\{r'\})\cup\{r\}$仍是一个最优解，

即最优解包含贪心选择，与假设矛盾。

FatMouse' Trade

```
struct node{
    double j,f,val;
    bool operator <(const node &T)const{
        return  val>T.val;
    }
}a[N];
// bool cmp(node a,node b){
//     return a.val>b.val;
// }
int n,m;
void solve(){
    while(cin>>m>>n){
        if(m==-1&&n==-1){
             break;
        }
        for(int i=1;i<=n;++i){
            cin>>a[i].j>>a[i].f;
            a[i].val=(1.0*a[i].j)/a[i].f;
        }
        sort(a+1,a+1+n);
        double ans=0, now=m;
        for(int i=1;i<=n;++i){
            if(now>a[i].f){
                ans+=a[i].j;
                now-=a[i].f;
            }else{
                ans+=a[i].val*now;
                break;
            }
        }
        // cout<<fixed<<setprecision(3)<<ans<<endl;
        printf("%.3f\n",ans);
    }
}
```

独木舟

```
int a[N];
void solve(){
    int n,w;
    cin>>n>>w;
    for(int i=1;i<=n;++i){
        cin>>a[i];
    }
    sort(a+1,a+1+n);
    int l=1,r=n;
    int ans=0;
    while(l<r){
        if(a[l]+a[r]>w){
            r--;
        }else{
            r--,l++;
        }
        ans++;
    }
    if(l==r){
        ans++;
    }
    cout<<ans<<endl;
}
```

拼数

```
string s[N];
bool cmp(string a,string b){
    return (a+b>b+a);
}
void solve(){
    int n;
    cin>>n;
    for(int i=1;i<=n;++i){
        cin>>s[i];
    }
    sort(s+1,s+1+n,cmp);
    for(int i=1;i<=n;++i){
        cout<<s[i];
    }
    cout<<endl;
}
```

今年暑假不AC

```
struct node{
    int st,ed;
    bool operator <  (const node T )const{
        return ed<T.ed;
    }
}a[N];
void solve(){
    int n;
    while(cin>>n){
        if(n==0){
            break;
        }
        for(int i=1;i<=n;++i){
            cin>>a[i].st>>a[i].ed;
        }
        sort(a+1,a+1+n);
        int now=a[1].ed,cnt=1;
        for(int i=2;i<=n;++i){
            if(a[i].st>=now){
                now=a[i].ed;
                ++cnt;
            }
        }
        cout<<cnt<<endl;
    }
}
```

区间选点&&最大不相交区间数量

```
struct node{
    int l,r;
    bool operator <(const node T){
        return r < T.r;
    }
}a[N];
void solve(){
    int n;
    cin>>n;
    for(int i=1;i<=n;++i){
        cin>>a[i].l>>a[i].r;
    }
    sort(a+1,a+1+n);
    int now=a[1].r,cnt=1;
    for(int i=2;i<=n;++i){
        if(a[i].l>now){
            cnt++;
            now=a[i].r;
        }
    }
    cout<<cnt<<endl;
```

```
}
```

区间分组

```
struct node{
    int l,r;
    bool operator <(const node T){
        return l < T.l;
    }
}a[N];
void solve(){
    int n;
    cin>>n;
    for(int i=1;i<=n;++i){
        cin>>a[i].l>>a[i].r;
    }
    sort(a+1,a+1+n);
    int cnt=0;
    priority_queue<int,vector<int>,greater<int> > q;
    for(int i=1;i<=n;++i){
        if(q.size()==0||q.top()>=a[i].l){
            cnt++;
        }else{
            q.pop();
        }
        q.push(a[i].r);

    }
    cout<<cnt<<endl;


}
```

区间覆盖

```
struct node{
    int l,r;
    bool operator <(const node T){
        return l < T.l;
    }
}a[N];
void solve() {
    int st, ed, n;
    cin >> st >> ed >> n;
    for (int i = 1; i <= n; ++i) {
        cin >> a[i].l >> a[i].r;
```

```
    }
    sort(a + 1, a + 1 + n);
    int ok = 0, ans = 0;
    for (int i = 1; i <= n; i++) {
        int pos = i, r = -2e9;
        while (pos <= n && a[pos].l <= st) {
            r = max(r, a[pos].r);
            pos++;
        }
        if (r < st) {
            ans = -1;
            break;
        }
        ans++;
        if (r >= ed) {
            ok=1;
            break;
        }
        st = r;
        i = pos - 1;
    }
    if (!ok) {
        ans = -1;
    }
    cout << ans << endl;
}
```

Rock and Lever

```
#define int ll
int a[N],cnt[N];
void solve(){
    int n;
    cin>>n;
    for(int i=1,x;i<=n;++i){
        cin>>x;
        cnt[(int)log2(x)]++;
    }
    int ans=0;
    for(int i=0;i<=40;++i){
        ans+=((cnt[i]*(cnt[i]-1)/2));
        cnt[i]=0;
    }
    cout<<ans<<endl;

}
```

[合并果子 / USACO06NOV] Fence Repair G

```
void solve(){
    int n;
    cin>>n;
    priority_queue<int, vector<int>, greater<int> > q;
    for(int i=1,x;i<=n;++i){
        cin>>x;
        q.push(x);
    }
    int ans=0;
    while(q.size()!=1){
        int num1=q.top();q.pop();
        int num2=q.top();q.pop();
        q.push(num1+num2);
        ans+=(num1+num2);
    }
    cout<<ans<<endl;
}
```

Work Scheduling G

```
struct node {
    int ddl, val;
    bool operator<(const node &T) const {
        return val>T.val;
    }
} a[N];

bool cmp(node x, node y) {
    return x.ddl < y.ddl;
}

void solve() {
    int n, ans = 0;
    cin >> n;
    for (int i = 1; i <= n; ++i) {
        cin >> a[i].ddl >> a[i].val;
    }
    sort(a + 1, a + 1 + n, cmp);
    priority_queue<node> q;
    for (int i = 1; i <= n; ++i) {
        if (a[i].ddl > q.size()) {
            ans += a[i].val;
            q.push(a[i]);
        } else {
            if (a[i].val > q.top().val) {
                ans -= q.top().val, q.pop();
```

```
                q.push(a[i]), ans += a[i].val;
            }
        }
    }
    cout << ans << endl;
}
```