贪心

贪心算法是对完成一件事情的方法的描述，贪心算法每一次都做出**当前看起来最好的选择，而不用考虑其它可能的选择。**

分发问题

C2-搬题

分发饼干

- 先排序，然后直接贪心，先满足小的然后注意往后满足大的

```
int findContentChildren(vector<int>& g, vector<int>& s) {
        int gn = g.size();
        int sn = s.size();
        int ans = 0;
        sort(g.begin(),g.end());
        sort(s.begin(),s.end());
        for(int i = 0,j = 0; i < gn && j < sn ;i++,j++)
        {
            while(i < gn && j < sn && s[j]<g[i])
                j++;
            if(j>=sn)
                break;
            else
                ans++;
        }
```

```
        return ans;
    }
```

分发糖果

我们可以将「相邻的孩子中，评分高的孩子必须获得更多的糖果」这句话拆分为两个规则，分别处理。

左规则：当`ratings[i-1]<ratings[i]`时，`i`号学生的糖果数量将比`i-1`号孩子的糖果数量多。

右规则：当`ratings[i]>ratings[i+1]`时，`i`号学生的糖果数量将比`i+1`号孩子的糖果数量多。

我们遍历该数组两次，处理出每一个学生分别满足左规则或右规则时，最少需要被分得的糖果数量。每个人最终分得的糖果数量即为这两个数量的最大值。

具体地，以左规则为例：我们从左到右遍历该数组，假设当前遍历到位置`i`，如果有`ratings[i-1]<ratings[i]`那么`i`号学生的糖果数量将比`i-1`号孩子的糖果数量多，我们令`left[i]=left[i-1]+1`即可，否则我们令`left[i]=1`。

在实际代码中，我们先计算出左规则`left`数组，在计算右规则的时候只需要用单个变量记录当前位置的右规则，同时计算答案即可。

```
int candy(int* ratings, int ratingsSize) {
    int left[ratingsSize];
    for (int i = 0; i < ratingsSize; i++) {
        if (i > 0 && ratings[i] > ratings[i - 1]) {
            left[i] = left[i - 1] + 1;
        } else {
            left[i] = 1;
        }
    }
    int right = 0, ret = 0;
    for (int i = ratingsSize - 1; i >= 0; i--) {
        if (i < ratingsSize - 1 && ratings[i] > ratings[i + 1]) {
            right++;
        } else {
            right = 1;
        }
        ret += fmax(left[i], right);
    }
    return ret;
}
```

找零问题

柠檬水找零

- 由于**面值较小的零钱可以匹配更多的情况**，所以优先选择面值较大的零钱进行找零

```
bool lemonadeChange(int* bills, int billsSize){
    int f = 0, t = 0;
    for(int i = 0; i < billsSize; i++)
    {
        if(bills[i]==5)
```

```
            f++;
        else if(bills[i]==10)
        {
            if(f==0)
                return false;
            else
            {
                f--,t++;
            }
        }
        else
        {
            if( t > 0 && f > 0)
            {
                t--,f--;
            }
            else if(t==0 && f >= 3)
            {
                f-=3;
            }
            else
                return false;
        }
    }
```

[2014-C3-大家一起数钢镚]

[2015-C3-怠惰的王木木II]

区域选择问题

无重叠区间

2014-C3-忙碌的Nova君

2015-C3-Magry的朋友很多 - Wonderland的邀请篇

- 其实就是**总的区间个数**减去**最多无重叠区间的个数**（经典贪心问题），得到的结果就是**需要移除最小区间的个数。**
- 求最多无重叠区间的个数——对**区间结束时间**进行排序即可

```
#include<cstdio>
#include<cstdlib>
#include<cstring>
#define ms(a,b) memset(a,b,sizeof(a))
using namespace std;
inline int read()
{
    int X=0,w=1; char ch=0;
    while(ch<'0' || ch>'9') {if(ch=='-') w=-1;ch=getchar();}
    while(ch>='0' && ch<='9') X=(X<<3)+(X<<1)+ch-'0',ch=getchar();
```

```
    return X*w;
}
inline void write(int x) {
    if(x < 0)putchar('-'),x=-x;
    if (x > 9)write(x / 10);
    putchar(x % 10 + 48);
}
int n,cnt,i,j;
struct task{
    int s;
    int f;
}t[1000007];
int cmp(const void*p1,const void*p2)
{
    struct task *a=(struct task*)p1;
    struct task *b=(struct task*)p2;
    if(a->f!=b->f)  return a->f-b->f;
    else return a->s-b->s;
}
int main(){
    while(~scanf("%d",&n))
    {
        cnt = 1;
        ms(t,0);
        for(i=0;i<n;i++)
        {
            t[i].s = read();
            t[i].f = read();
        }
        qsort(t,n,sizeof(struct task),cmp);
        i = 0;
        for(j = 1;j<n;j++)
        {
            if(t[i].f<=t[j].s)
            {
                cnt++;
                i = j;
            }
        }
        write(cnt);
        putchar('\n');
    }
    return 0;
}
```

用最少数量的箭引爆气球

```
#include<cstdio>
#include<cstdlib>
#include<cstring>
#define ms(a,b) memset(a,b,sizeof(a))
using namespace std;
inline int read()
{
    int X=0,w=1; char ch=0;
    while(ch<'0' || ch>'9') {if(ch=='-') w=-1;ch=getchar();}
    while(ch>='0' && ch<='9') X=(X<<3)+(X<<1)+ch-'0',ch=getchar();
```

```
    return x*w;
}
inline void write(int x) {
    if(x < 0)putchar('-'),x=-x;
    if (x > 9)write(x / 10);
    putchar(x % 10 + 48);
}
int n,cnt;
struct task{
    int s;
    int f;
}t[100007];
int cmp(const void*p1,const void*p2)
{
    struct task *a=(struct task*)p1;
    struct task *b=(struct task*)p2;
    if(a->f!=b->f)  return a->f-b->f;
    else return a->s-b->s;
}
int main(){
    while(~scanf("%d",&n))
    {
        cnt = 1;
        ms(t,0);
        for(int i=0;i<n;i++)
        {
            t[i].s = read();
            t[i].f = read();
        }
        qsort(t,n,sizeof(struct task),cmp);
        int pos = t[0].f;
        for (int i = 0; i < n; ++i) {
            if (t[i].s > pos) {
                pos = t[i].f;
                ++cnt;
            }
        }
        write(cnt);
        putchar('\n');
    }
    return 0;
}
```

跳跃问题

跳跃游戏

- 维护最远距离，判断最远距离能否到达终点

```
bool canJump(int* nums, int numsSize){
    if(numsSize == 1)   return true;
    int rightmost = 0;
    for(int i = 0;i<numsSize;i++)
    {
        if(i<=rightmost)
```

```
        {
            rightmost = fmax(rightmost,i+nums[i]);
            if(rightmost>=numsSize-1)
                return true;
        }
    }
    return false;
}
```

跳跃游戏 II

```
int jump(int* nums, int numsSize){
    int cnt=0,maxfar=0,end=0;
    for(int i=0;i<numsSize-1;i++)
    {
       maxfar = fmax(maxfar,i+nums[i]);
       if(i == end)
       {
           end = maxfar;
           cnt++;
       }
    }
    return cnt;
}
```

分数背包

2014-C4-机智零崎不会没梗 I

- 贪心就完事了，但是要注意`add = min(g , Node[cnt].w);`

```
#include<cstdio>
#include<algorithm>
using namespace std;
int i,j,cnt = 0,k;
double g,w,v,n,res,add;
struct node{
    double w;
    double rate;
    inline bool operator < (const node& o)const {
        return o.rate < rate;
    }
}Node[10007];
int main(){
    while(~scanf("%lf %d %lf",&g,&k,&n))
    {
        for(int i = 0;i<k;i++)
        {
            scanf("%lf %lf",&w,&v);
            Node[i].w = w;
            Node[i].rate =  v/w;
        }
        sort(Node,Node+k);
        cnt = 0;
```

```
        res = 0.0;
        for(int i = 0;i<k;i++)
        {
            while(g>0.0 && cnt < k && Node[cnt].rate>0)
            {
                add = min(g , Node[cnt].w);
                res += add*Node[cnt++].rate;
                g-=add;
            }
        }
        if(res>=n)  puts("summon!");
        else    printf("%.3lf\n",n-res);
    }
    return 0;
}
```

【深基12.例1】部分背包问题

- 考虑了精度问题，值得借鉴

```
#include<cstdio>
#include<algorithm>//用到sort
using namespace std;
struct Node{//金币结构体
    int w,v;//w表示重量，v表示价值
}a[110];
int read(){//普通的快读，不解释
    int x=0,f=1;
    char c=getchar();
    while(c<'0'||c>'9'){
        if(c=='-') f=-1;
        c=getchar();
    }
    while(c>='0'&&c<='9'){
        x=x*10+c-'0';
        c=getchar();
    }
    return x*f;
}
bool cmp(Node aa,Node bb){//定义排序方法
    return aa.v*bb.w>aa.w*bb.v;//按性价比从高到低排序，为防止精度问题直接交叉相乘
}
int main(){//主函数
    int n=read(),m=read();
    double ans=0;//记录答案
    for(int i=1;i<=n;i++) a[i].w=read(),a[i].v=read();
    sort(a+1,a+n+1,cmp);//排序
    for(int i=1;i<=n;i++){//一次遍历
        if(a[i].w<=m) ans+=a[i].v,m-=a[i].w;//够就全拿
        else{//不够
            ans+=a[i].v*m*1.0/(a[i].w*1.0);//拿上能拿的部分，注意强转double
            break;//直接退出循环
        }
    }
    printf("%.2lf",ans);//保留2位小数
    return 0;//华丽结束
}
```

哈夫曼编码

2014-C4-机智零崎不会没梗II

```
#include<iostream>
#include<cstring>
#include<queue>
#define ms(a,b) memset(a,b,sizeof(a))
typedef long long LL;
using namespace std;
inline LL read()
{
    LL X=0,w=1; char ch=0;
    while(ch<'0' || ch>'9') {if(ch=='-') w=-1;ch=getchar();}
    while(ch>='0' && ch<='9') X=(X<<3)+(X<<1)+ch-'0',ch=getchar();
    return X*w;
}
inline void write(LL x) {
    if(x < 0)putchar('-'),x=-x;
    if (x > 9)write(x / 10);
    putchar(x % 10 + 48);
}
char s[2021];
LL fre[100],ans,t,i,sum,a,b,l;
priority_queue <LL,vector<LL>,greater<LL> > q;
int main(){
    t = read();
    while(t--)
    {
        ms(fre,0);
        ans = sum = 0;
        while(!q.empty())
            q.pop();
        scanf("%s",s);
        l = strlen(s);
        for(i=0;i<l;i++)
        {
            fre[s[i]-'A']++;
        }
        for(i=0;i<100;i++)
        {
            if(fre[i]!=0)
            {
                q.push(fre[i]);
                sum++;
            }
        }
        while(q.size()>1)
        {
            a = q.top();
            q.pop();
            b = q.top();
            q.pop();
            ans+=(a+b);
            q.push(a+b);
```

```
        }
        write(ans);
        putchar('\n');
    }
    return 0;
}
```

排序重写cmp问题

- 这类问题往往先列出只有两个人的情况，然后大胆贪心写为sort中的cmp，然后进行排序后操作。

两地调度

公司计划面试 `2n` 人。给你一个数组 `costs`，其中 `costs[i] = [aCosti, bCosti]`。第 `i` 人飞往 `a` 市的费用为 `aCosti` ，飞往 `b` 市的费用为 `bCosti`。

返回将每个人都飞到 `a` 、`b` 中某座城市的最低费用，要求每个城市都有 `n` 人抵达。

```
#include <cstdio>
#include <cstdlib>
#include <cstring>
#define ms(a, b) memset(a, b, sizeof(a))
using namespace std;
inline int read()
{
    int X = 0, w = 1;
    char ch = 0;
    while (ch < '0' || ch > '9')
    {
        if (ch == '-')
            w = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9')
        X = (X << 3) + (X << 1) + ch - '0', ch = getchar();
    return X * w;
}
inline void write(int x)
{
    if (x < 0)
        putchar('-'), x = -x;
    if (x > 9)
        write(x / 10);
    putchar(x % 10 + 48);
}
int n, cnt, i, j;
struct task
{
    int s;
    int f;
} t[1000007];
int cmp(const void *p1, const void *p2)
{
    struct task *a = (struct task *)p1;
    struct task *b = (struct task *)p2;
```

```
    return a->s + b->f < a->f + b->s;
}
int main()
{
    scanf("%d", &n);

    ms(t, 0);
    for (i = 0; i < n; i++)
    {
        t[i].s = read();
        t[i].f = read();
    }
    qsort(t, n, sizeof(struct task), cmp);
    int a = 0, b = 0;
    for (int i = 0; i < n / 2; i++)
    {
        b += t[i].f;
    }
    for (int i = n / 2; i < 2 * n; i++)
    {
        a += t[i].s;
    }
    write(a + b);

    return 0;
}
```

2018-C4-商人卖鱼

- 同样以两个为例作为cmp的书写条件，莽一手排序之后直接操作

```
#include <bits/stdc++.h>
#define mp make_pair
#define ms(a, b) memset(a, b, sizeof(a))
#define maxn 1000007
typedef long long LL;
using namespace std;
inline LL read()
{
    LL X = 0, w = 1;
    char ch = 0;
    while (ch < '0' || ch > '9')
    {
        if (ch == '-')
            w = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9')
        X = (X << 3) + (X << 1) + ch - '0', ch = getchar();
    return X * w;
}
inline void write(LL x)
{
    if (x < 0)
        putchar('-'), x = -x;
    if (x > 9)
        write(x / 10);
```

```
    putchar(x % 10 + 48);
}
struct Node
{
    LL t, d;
//    bool operator < (const Node& o) const {
//        //这题如何设计这里是最核心的问题
//        //先拿两个人A和B来看 先走A的话等待B.s*A.return_time 先走B的话等待
A.s*B.return_time
//        return o.d * t < d * o.t;
//    };
} node[maxn];
bool cmp(const struct Node &a, const struct Node &b)
{
    return a.t * b.d < a.d * b.t;
}
LL n, ans;
int main()
{
    while (~scanf("%lld", &n))
    {
        ms(node, 0);
        LL time = 0;
        LL ans = 0;
        for (int i = 0; i < n; i++)
        {
            node[i].t = read(), node[i].d = read();
        }
        sort(node, node + n,cmp);
        for (int i = 0; i < n; i++)
        {
            ans += node[i].d * time;
            time += node[i].t;
        }
        write(ans);
        putchar('\n');
    }
}
```