第 5 章 贪心

5.1 贪心基础

一、什么是贪心

贪心算法（又称贪婪算法）是指，在对问题求解时，总是做出在当前看来是最好的选择。也就是说，不从整体最优上加以考虑，他所做出的是在某种意义上的局部最优解。

贪心算法不是对所有问题都能得到整体最优解，关键是贪心策略的选择，选择的贪心策略必须具备无后效性，即某个状态以前的过程不会影响以后的状态，只与当前状态有关。

贪心算法的使用前提：局部最优解一定能导致全局最优解。

学过的贪心问题：数塔、摘花生、过河卒！

贪心的解决策略：

过程

1、建立数学模型来描述问题；

2、把求解的问题分成若干个子问题；

3、对每一子问题求解，得到子问题的局部最优解；

4、把子问题的解局部最优解合成原来解问题的一个解。

5.2 贪心习题训练

一、课堂案例

1326：【入门】需要安排几位师傅加工零件？

思路：由于题目要求的是最少需要多少师傅来加工零件，因此我们优先挑选加工能力强的师傅来加工零件。

```
#include <bits/stdc++.h>
using namespace std;

//辅助降序排序的函数
bool cmp(int a,int b){
    if(a > b){
        return true;
    }else{
        return false;
    }
}

int a[110],i,s,n,m;
int main(){
    cin>>m>>n;
    //读入 n 个帅傅的加工能力
    for(i = 1;i <= n;i++){
        cin>>a[i];
    }

    //对 n 个师傅的加工能力进行排序
    sort(a+1,a+n+1,cmp);

    //逐个求和
    for(i = 1;i <= n;i++){
        s = s + a[i];
        //人数是否足够
        if(s >= m){
            cout<<i;
            break;
```

```
        }
    }

    //如果所有师傅都来加工人数也不够
    if(s < m){
        cout<<"NO";
    }

}
```

1228：【基础】排队打水问题

	方案一				方案二				方案三			
	龙头1	龙头2			龙头1	龙头2			龙头1	龙头2		
	2	5			5	6			2	4		
	4	6			2	4			5	6		
总计	8	16	24	总计	12	16	28	总计	9	14	23	

每个人的总打水时间 = 该用户打水时间 + 排队等待时间，因此只有打水快的人先打，总的排队时间才是最少的。

假设有 6 个人，打水时间分别为：2 5 8 6 4 9，最少总打水时间是多少？

打水时间

a	2	4	5	6	8	9
	1	2	3	4	5	6

每个人的总打水时间

a	2	4	7	10	15	19
	1	2	3	4	5	6

前 r 个人不需要排队，从第 r+1 个人开始，a[i]=a[i]+a[i-r]

a[3]=a[3]+a[1]

a[4]=a[4]+a[2]

a[5]=a[5]+a[3]

a[6]=a[6]+a[4]

```
#include <bits/stdc++.h>
using namespace std;

int a[510],r,i,n,s;
int main(){
    cin>>n>>r;
    //读入 n 个人的打水时间
    for(i = 1;i <= n;i++){
        cin>>a[i];
    }

    //打得快的人先打，对打水时间升序排序
    sort(a+1,a+n+1);

    //计算每个人的总打水时间
    for(i=1;i<=n;i++){
        //从第 r+1 个人开始要重新计算每个人的总打水时间
        if(i >= r + 1){
            a[i] = a[i] + a[i - r];
        }

        s = s + a[i];
    }
```

```
    cout<<s;
}
```

1229：【提高】拦截导弹的系统数量求解

思路：找当前拦截系统中，高度最矮的系统拦截当前的导弹！

导弹	389	207	175	300	299	182	160	165

拦截策略：如果没有系统能够拦截，则开一个系统，修改系统高度
如果有系统能够拦截，则找当前拦截高度最矮(第一个能拦截的)的系统进行拦截，修改系统高度为当前导弹的高度
特点：由于采用上述策略，导致前面的拦截系统的高度<后面的系统

拦截系统储存每套系统拦截的最高高度	160	165						
	k=1	k=2						

```
#include <bits/stdc++.h>
using namespace std;

//x：代表每个导弹的高度
//p：找到的能拦截导弹的系统的下标
//k：a 数组中已经有的能够拦截导弹的系统数量
int a[1010],i,n,x,p,k,j;
int main(){
    cin>>n;
    for(i = 1;i <= n;i++){
        cin>>x;
        p = -1;
        //循环 a 数组，找到第一个能够拦截的系统
        for(j = 1;j <= k;j++){
            if(a[j] >= x){
                p = j;
                break;
            }
        }

        //如果没找到系统拦截
        if(p == -1){
            k++;
            a[k] = x;//设定系统能拦截的最高高度
        }else{
            //用第 p 个系统拦截，修改系统的最高高度
            a[p] = x;
        }
    }

    cout<<k;
}
```

思考：如果要记录第几个系统拦截了哪些飞弹，应该如何解？

1372：【基础】活动选择

我们使用的贪心策略如下。即每一步总是选择这样的活动来占用资源：使得余下的未调度时间最大化，使得兼容的活动尽可能多。为了达到这个目的，我们将 n 个待选活动按结束时间递增的顺序排序：e1’<=e2’<=…<=en’。

开始时间	结束时间

1	4	end=4
3	5	
0	6	
5	7	end=7
3	8	
5	9	
6	10	
8	11	end=11
8	12	
2	13	
12	14	end=14

```
#include <bits/stdc++.h>
using namespace std;

/*
  要求安排的活动尽量多。请问最多可以安排多少活动
  安排活动的策略：尽可能选择结束时间早的活动
  第一步：将结束时间，按照升序排序，开始时间要做联动排序
  第二步：选择第一个活动的结束 end 时间为依据，向下找到第一个开始时间 >=end
          的活动为第 2 个活动，以此类推
*/
int b[110],e[110];
//c:计数器，计算最多能够排多少个活动
int n,i,j,c;
int main(){
    cin>>n;
    //读入 n 个活动
    for(i = 1;i <= n;i++){
        cin>>b[i]>>e[i];
    }

    //对 n 个活动的起止时间，按照结束时间升序排序
    for(i = 1;i <= n - 1;i++){
        //第 i 轮，从下标为 1 的数，排到下标为 n-i
        for(j = 1;j <= n - i;j++){
            if(e[j] > e[j+1]){
                swap(e[j],e[j+1]);
                swap(b[j],b[j+1]);
            }
        }
    }

//  for(i = 1;i <= n;i++){
//      cout<<b[i]<<" "<<e[i]<<endl;
//  }

    int end = e[1];//第一个活动一定会选到
    c = 1;
    for(i = 2;i <= n;i++){
        //找到 b[i]>=end 的活动，选择它
        if(b[i] >= end){
            c++;
            end = e[i];
        }
    }

    cout<<c;
}
```

二、作业

1375：【提高】拦截导弹方案求解

1373：【基础】删数问题

1371：【基础】均分纸牌

1485：【基础】接水问题

1235：【基础】过河的最短时间

提示：需递推出如果只有一个人（t1）两个方案下的过河总时间，如果有 2 个人（t1、t2）两个方案下的过河总时间，如果有 3 个人（t1、t2、t3）两个方案下的过河总时间……5 个人（t1、t2、t3、t4、t5）方案下的过河总时间，对比两个方案的差异，找出规律。

1730：【入门】购买贺年卡