《L4-CSP-J 复赛进阶算法与数据结构》

Day1 贪心策略(上)

主讲人：小武老师

可达信奥_小武老师_keda.ac

贪心策略

贪心算法是指，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。

可达信奥_小武老师_keda.ac

贪心策略

greedy algorithm

adj.贪婪的；贪心的；贪吃的；渴望的；

在算法竞赛中求解某些问题时，只需要做出在当前看来是最好的选择就能获得最好的结果，而不需要考虑整体上的最优。

说简单点就是可大的拿，往死里塞

如何获得好成绩？

如果想在算法竞赛中得奖，就要尽可能多读书、多思考、多练习。一般来说，学习越努力，成绩就越好。

每天 上课0小时 刷题0小时 收益：0	每天 上课2小时 刷题3小时 收益：5	每天 上课10小时 刷题12小时 收益：？

但因为花了太多时间在编程上而极度压缩休息的时间，反而会效率低下，得不偿失。不过，在很多场合，确实是越多越好的。

贪心策略

基本思路

把求解的问题分成若干个子问题

↓

对每个子问题求解，得到子问题的局部最优解

↓

把子问题的最优解合成原问题的全局最优解

适用问题

前提：局部最优策略能导致产生全局最优解

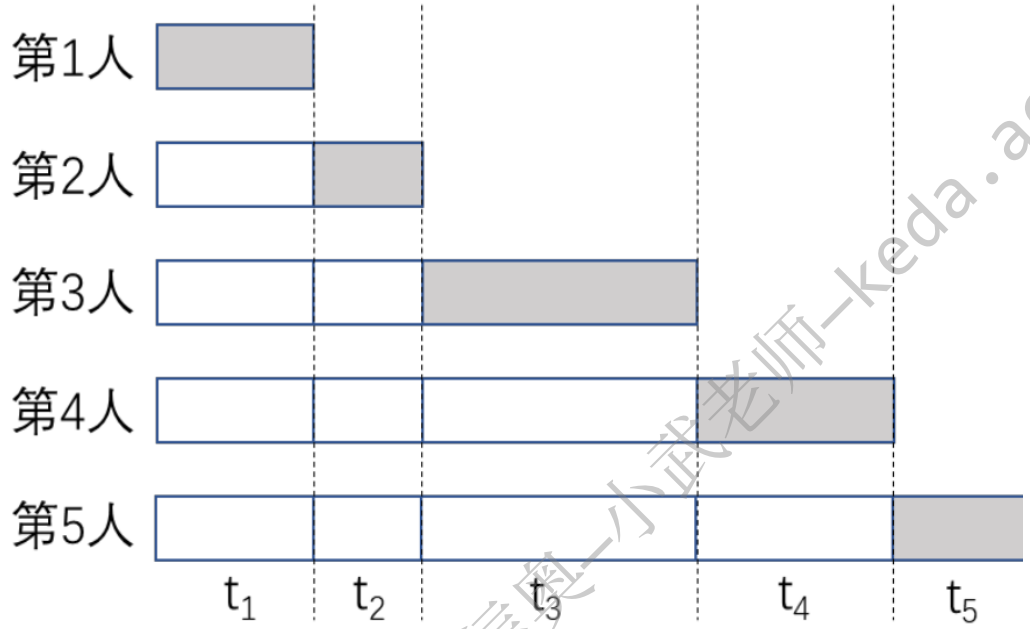
贪心策略

P0380. 排队接水

有 n 个人在一个水龙头前排队接水，每个人接水的时间为 T_i。 找出这 n 个人排队的一种顺序，使这 n 个人的平均等待时间最小。

分析：某人等待时间总和就是前 面每个单人时间的和第一个人不需要等待，第二个人需要等待一个人的时间，第三个 人要等待前两人。假设经安排，第 i个同学的打水时间是 t_i则，同学等待时间总和为：

$$s = (n-1)t_i + (n-2)t_2 \cdots + 1 \cdot t_{n-1} + 0 \cdot t_n$$



猜测：t_1到t_n应该从小到大排序， 可以使时间总和 s 最小. 如何证明？

贪心策略证明

P0380. 排队接水

证明（反证法）

假设最佳方案中，t_1到t_n不是小到大排，当 $i< j$时，$t_i > t_j$。

这两项贡献的总时间是：$S_1 = a \cdot t_i + b \cdot t_j, \quad a > b$

若 t_i 和 t_j调换，那么贡献总时间变为：$S_2 = a \cdot t_j + b \cdot t_i$ 两者相减，得到：

$$S_1 - S_2 = a(t_i - t_j) - b(t_i - t_j) = (a - b)(t_i - t_j) > 0$$

说明，S1大于S2，调换后总时间会缩短，和原来认为S1是“最佳方案”矛盾

所以贪心算法成立

贪心策略

```cpp
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1001;;
struct stu{
    int time;
    int id;
}a[maxn];
bool cmp(stu a, stu b){
    if(a.time != b.time)
        return a.time < b.time;
    return a.id < b.id;
}
```

按照接水时间从小到大排序，时间相同时编号小的同学优先。

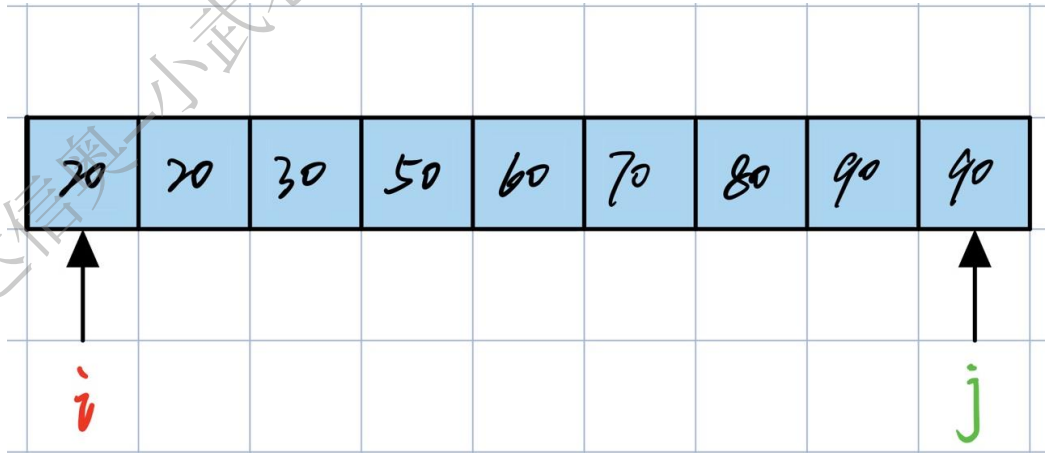
```cpp
int main(){
    int n;
    double sum = 0;
    scanf("%d", &n);
    for(int i = 1; i <= n; i++) {
        a[i].id = i;
        cin >> a[i].time;
    }
    sort(a+1, a+n+1, cmp);
    for(int i = 1; i < n; i++){
        printf("%d ", a[i].id);
        sum += i*a[n-i].time;
    }
    printf("%d\n", a[n].id);
    printf("%.2f", sum/n);
    return 0;
}
```

编程实践 Online Judge

P0041 P0042 P0043 P0044 P0389

可达信奥_小武老师_keda.ac

P0042. 纪念品分组



尝试证明（反证法）

```cpp
#include<bits/stdc++.h>
using namespace std;
const int maxn = 30001;
int a[maxn];
int main(){
    int w, n, cnt = 0;
    cin >> w >> n;
    for(int i = 1; i <= n; i++) cin >> a[i];
    sort(a+1, a+n+1);
    int i = 1; int j = n;
    while(i<=j){
        cnt++;
        if(a[i]+a[j] <= w){
            i++; j--;
        }else{
            j--;
        }
    }
    cout << cnt;
    return 0;
}
```

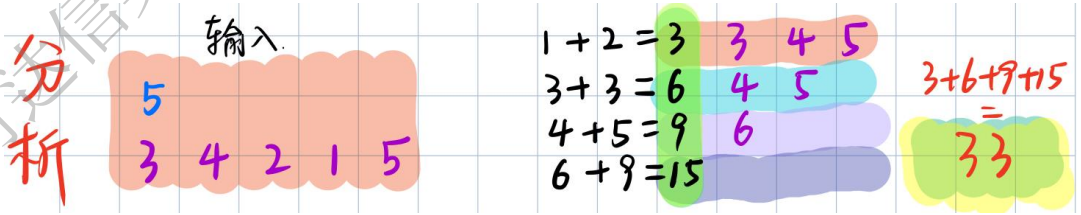
P0043. 最大整数

```
6
321 32 407 135 13 217
```

32132 or 32321 ?

```cpp
#include<bits/stdc++.h>
using namespace std;
const int maxn = 50;
string ss[maxn];
bool cmp(string a, string b){
    if(a+b < b+a) return true;
    else return false;
}
int main(){
    int n; cin >> n;
    for(int i = 1; i <= n; i++) cin >> ss[i];
    sort(ss+1, ss+n+1,cmp);
    for(int i = n; i >= 1; i--)
        cout << ss[i];
    return 0;
}
```

P0044. 合并果子



每次找到最小的两个，动态维护序列

```cpp
#include<bits/stdc++.h>
using namespace std;
int n,x,ans;
priority_queue<int,vector<int>,greater<int> >q;
int main(){
    cin>>n;
    for(int i=1;i<=n;i++) cin>>x,q.push(x);
    while(q.size()>=2){
        int a=q.top(); q.pop();
        int b=q.top(); q.pop();
        ans+=a+b;
        q.push(a+b);
    }
    cout<<ans;
    return 0;
}
```

优先队列（小顶堆来实现）

P0389. 金银岛

```cpp
#include<bits/stdc++.h>
using namespace std;
struct stone{
    int n; // weight
    int v; // value
    double avg;
}a[10001];
bool cmp(stone a, stone b){
    return a.avg > b.avg;
}
```

```cpp
int main(){
    int k;  cin >> k;
    while(k--){
        int w, s;
        cin >> w >> s;
        for(int i = 1; i <= s; i++){
            cin >> a[i].n >> a[i].v;
            a[i].avg = 1.0*a[i].v/a[i].n;
        }
        sort(a+1, a+s+1,cmp);
        double sum = 0;
        for(int i = 1; i <= s; i++){
            if(a[i].n > w){
                sum += a[i].avg*w;
                break;
            }
            sum += a[i].v;
            w -= a[i].n;
        }
        printf("%.2f\n", sum);
    }
    return 0;
}
```

贪心总结

贪心不难 ⟶ **难的是证明**

课后习题与实验

Talk is cheap, show me the code !

下节课见啦！