笔记 22-10-13

李肖

2022 年 10 月 13 日

1 贪心算法

1.1 应用场景

优化问题: 最长，最大，最小，最高……

1.2 思路

每步采用局部最优策略，到达全局最优策略

1.3 例题

1.3.1 任务安排问题

博客解析: 博客链接

```
#include<iostream>
#include<algorithm>
#include<vector>
using namespace std;

//任务安排问题
struct task {
    int id;
    int start;
    int finish;
};

bool cmpfinish(task t1,task t2)
```

```
{
    return t1.finish < t2.finish;
}

int main()
{
    int n;
    cin >> n;
    task Task[20];
    vector<task> vtask;
    for (int i = 0; i < n; i++)
    {
        cin >> Task[i].id >> Task[i].start >> Task[i].finish;
    }
    //首先将任务按照结束时间非递减进行排序
    sort(Task, Task + n, cmpfinish);
    //因为第一个任务一定在最优解里面（这个可以反证法证明出来的），所以首先将第一个任务加入
    vtask.push_back(Task[0]);
    //然后看一下后面的任务，其start开始时间，有没有与前一个任务的结束时间fi重合
    int j = 0;//用来记录最新加入的任务，以便确定需要比较的finish时间。
    for (int i = j + 1; i < n; i++)
    {
        if (Task[i].start >= Task[j].finish)//需要注意等于也可以!
        {
            vtask.push_back(Task[i]);
            j = i;
        }
    }
    vector<task>::iterator it;
    for (it = vtask.begin(); it != vtask.end(); it++)
    {
        cout << (*it).id << endl;
    }
    system("pause");
    return 0;
}
```