贪心算法

什么是贪心？贪心是一种策略，它的本质是选择每一阶段的局部最优，从而实现全局最优。

例如，有一堆钞票，你可以拿走十张，如果想拿走最大数额的钱，那么要怎么做呢？

你的答案一定是每次都拿走面额最大的钞票，最终结果就是拿走最大数额的钱。

在这里，“每次拿面额最大的钞票就是局部最优”，“最后拿走最大数额的钱就是全局最优”

策略选择

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，只与当前的状态有关。

贪心算法的最难点就是通过**局部最优推出全局最优**。因为贪心算法真的没有什么特征。

贪心算法不是对所有问题都能得到整体最优解。所以，当你面对一道题的时候，如果想使用贪心算法，你必须证明它的可行性。最简单的办法就是举反例。

P2240 部分背包问题

题目描述

阿里巴巴走进了装满宝藏的藏宝洞。藏宝洞里面有 N(N≤100)堆金币，第i堆金币的总重量和总价值分别是 mi,vi(1≤mi,vi≤100)。阿里巴巴有一个承重量为 T(T≤1000)的背包，但并不一定有办法将全部的金币都装进去。他想装走尽可能多价值的金币。所有金币都可以随意分割，分割完的金币重量价值比（也就是单位价格）不变。请问阿里巴巴最多可以拿走多少价值的金币？

输入格式

第一行两个整数 N,T。

接下来N行，每行两个整数 mi,vi。

输出格式

一个实数表示答案，输出两位小数

输入样例

```
4 50
10 60
20 100
30 120
15 45
```

输出样例

240.00

解析

因为包的承重量有限，如果能拿走相同重量的金币，当然是有限拿走单位价格最贵的金币。所以，正确的做法就是将金币的单价从高向低排序，然后按照顺序将整堆金币都放入包里。如果整堆放不进背包，就分割这一堆金币直到刚好能装下为止。

当你有了思路之后，别忘了还要证明它。

常见的证明方式有两种：

1、数学归纳法

2、反证法

在这里，我们使用反证法。

假设我们没有在背包中放入单价最高的金币，而是放入了单价更低的金币，却依然能达到总价值最高的目的。

例如我们装入了5枚1元的硬币，那么当前的最大价值就是5。

但是，我们很快就能发现，如果我把其中的一个1元换成2元，那么最大价值就会升高。这个结果与最优解矛盾，所以贪心算法成立。

以上就是使用反证法来证明贪心的过程：假设要选择的方案不是贪心算法的方案，只需要用贪心算法的方案替换掉要选择的方案，结果会更好（至少不会更差）。

编码

```
#include <bits/stdc++.h>

using namespace std;
const int Maxn = 100 + 1;
int w, n;
//最终装入的总价值
double countNum = 0;

//定义金币的结构体，存储其重量和价值
struct Coin {
    int m;
    int v;
    double ratio;
};

Coin Coins[Maxn]; //金币数组
```

```cpp
//比价单个金币的性价比
bool Compare(Coin a, Coin b) {
    return a.ratio > b.ratio;
}

int main() {
    scanf("%d %d", &n, &w);
    //读入金币的重量和价值，并计算它们的性价比
    for (int i = 1; i <= n; ++i) {
        scanf("%d %d", &Coins[i].m, &Coins[i].v);
        Coins[i].ratio = Coins[i].v * 1.0 / Coins[i].m;
    }
    //按照性价比进行排序
    sort(Coins + 1, Coins + n + 1, Compare);

    //遍历所有的金币，开始尝试放入
    for (int i = 1; i <= n; ++i) {
        Coin coin = Coins[i];
        //如果剩余的背包容量大于金币的重量，那么就直接装入
        if (w > coin.m) {
            w -= coin.m;
            countNum += coin.v;
        }
        //否则，则切割金币，进行装入
        else {
            countNum += w * coin.ratio;
            w = 0;
        }
    }
    //输出最终的总价值
    printf("%.2lf", countNum);
    return 0;
}
```

新问题

如果藏宝洞不是一堆堆金币，而是一个个单价不一且无法分割的金块，还能使用类似的策略吗？

假设背包承重量是50，有4个金块，其重量和价值分别是：

10 60
20 100
30 120
15 45

此刻，如果我们继续按照贪心策略，优先将单价最高的装进背包，如果空间不够就跳过，不断的遍历剩下的尽快，直到不能装下为止。我们会得到下图的结果：

重量	价值	单价
10	60	6
20	100	5
15	45	3

总价：205

实际上，我们应该战略性的舍弃单价最高的，这样我们就能得到总价值最高的组合。结果如下：

重量	价值	单价
20	100	5
30	120	4

总价：220

因此，对于本问题，我们只需要一个反例就能证明贪心策略的错误。所以这道题不能使用贪心策略，而应该使用动态规划。