

模型-最优化方法-确定性算法-贪心算法【hxy】

1. 模型名称
2. 模型评价
 - 2.1 适用范围
 - 2.2 模型局限
3. 基本算法
4. 实例
 - 4.1 问题描述
 - 4.2 数学解法
 - 4.3 代码实现
5. 参考资料

模型-最优化方法-确定性算法-贪心算法【hxy】

1. 模型名称

贪心算法 (Greedy Algorithm)

2. 模型评价

2.1 适用范围

- 原问题复杂度过高
- 求全局最优解的数学模型难以建立
- 求全局最优解的计算量过大
- 没有太大必要一定要求出全局最优解，“比较优”就可以

2.2 模型局限

- 不能保证求得的最优解是最佳的
- 不能用来求最大或最小解问题
- 只能求满足某些约束条件的可行解的范围

3. 基本算法

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. 对每个子问题求解，得到子问题的局部最优解
4. 把子问题的局部最优解合成原来问题的一个解
5. 用替换法证明贪心算法得到的解即为最优解

4. 实例

4.1 问题描述

现在有多箱不同的糖果，每箱糖果有自己的价值和重量，每箱糖果都可以拆分成任意散装组合带走，圣诞老人的驯鹿雪橇最多只能装下重量 w 的糖果，请问圣诞老人最多能带走多大价值的糖果

输入：

```
4 15
100 4
412 8
266 7
591 2
```

输出：

```
1193.0
```

4.2 数学解法

解法：按礼物的价值/重量比从大到小依次选取礼物，对选取的礼物尽可能的多装，直到达到总重量 w

复杂度： $O(n\log n)$

证明：替换法

对于用非此法选取的最大价值糖果箱序列，可以按其价值/重量比从大到小排序后得到：

序列1 a_1, a_2, \dots

用贪心算法按其价值/重量比从大到小排序后得到：

序列2 b_1, b_2, \dots

由于价值/重量比相同的若干箱糖果，可以合并成一箱，所以两个序列中元素都不重复

对于发现的第一个 $a_i \neq b_i$ ，则必有 $a_i < b_i$ ，

则在序列1中，用 b_i 这种糖果替代若干重量的 a_i 这种糖果，则会使序列1的总价值增加，

这和序列1是价值最大的取法矛盾，所以序列1 = 序列2（序列2不可能是序列1的一个前缀且比序列1短）

4.3 代码实现

[main.cpp](#)

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

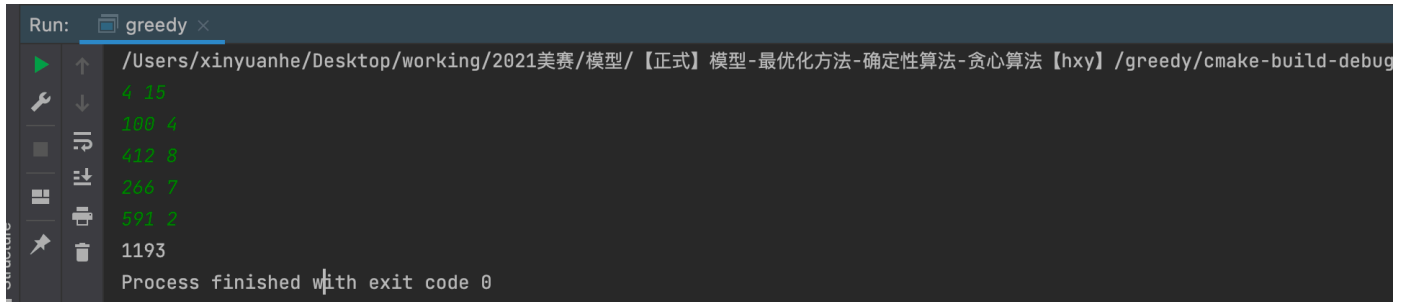
const double eps = 1e-6;
struct Candy {
    int v; int w;
    bool operator < (const Candy & c) const {
        return double(v)/w - double(c.v)/c.w > eps;
    }
} candies[110];
```

```

int main() {
    int n,w;
    cin>>n>>w;
    for (int i=0; i<n; ++i) {
        cin>>candies[i].v>>candies[i].w;
    }
    sort(candies,candies+n);
    int totalW = 0;
    double totalV = 0;
    for (int i=0; i<n; ++i) {
        if (totalW + candies[i].w <= w) {
            totalW += candies[i].w;
            totalV += candies[i].v;
        }
        else {
            totalV += candies[i].v * double(w-totalW) / candies[i].w;
            break;
        }
    }
    cout << totalV;
    return 0;
}

```

结果：



```

Run: greedy x
/Users/xinyuanhe/Desktop/working/2021美赛/模型/【正式】模型-最优化方法-确定性算法-贪心算法【hxy】/greedy/cmake-build-debug
4 15
100 4
412 8
266 7
591 2
1193
Process finished with exit code 0

```

5. 参考资料

1. [MOOC-程序设计与算法-贪心算法](#)
2. [从零开始学贪心算法-案例](#)
3. [贪心算法详解及经典例子](#)
4. [百度百科-贪心算法](#)
5. [漫画：五分钟学会贪心算法](#)