二分与贪心算法

郭炜 刘家瑛

那些朴素的算法思想1

- 例如 二分查找
- [前提]: 已经排序好的序列
- 在查找元素时, 首先与序列 中间的元素 进行比较
  - 如果 大于 这个元素, 就在当前序列的 后半 部分继续查找
  - 如果 小于 这个元素, 就在当前序列的 前半 部分继续查找
  - 直到找到相同的元素, 或者所查找的序列范围为空为止

二分的基本思想

- 伪码描述

```
left = 0, right = n -1
while (left <= right)
    mid = (left + right) / 2
    case
        x[mid] < t:    left = mid + 1;
        x[mid] = t:    p = mid; break;
        x[mid] > t:    right = mid -1;
return -1;
```

那些朴素的算法思想2

- 例如 贪心算法
- 问题求解时，总是做出在当前看来是最好的选择
  - 即 不保证全局最优，仅是在某种意义上的局部最优解
- 贪心算法设计的关键是贪心策略的选择
- 自顶向下计算
  - 通过贪心选择，将原问题规约为子问题

贪心的基本思想

- 建立数学模型来描述问题
- 把求解的问题分成若干个子问题
- 对每一子问题求解，得到子问题的局部最优解
- 把子问题的解局部最优解合成原来解问题的一个解

Note:

对所采用的贪心策略一定要仔细分析其是否满足 “无后效性”

POJ 4110
圣诞老人的礼物

北京大学

问题描述

- 圣诞节来临了，在城市A中圣诞老人准备分发糖果，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
- 圣诞老人的驯鹿最多只能承受一定重量的糖果
- 请问圣诞老人最多能带走 多大价值 的糖果

程序输入

- 第一行由两个部分组成, 分别为
  - 糖果箱数正整数 n (1 <= n <= 100)
  - 驯鹿能承受的最大重量正整数 w (0 < w < 10000)

  两个数用空格隔开
- 其余n行每行对应一箱糖果, 由两部分组成
  - 一箱糖果的价值正整数 v
  - 一箱糖果的重量正整数 w

  中间用空格隔开

程序输出

- 输出圣诞老人能带走的糖果的最大总价值，保留1位小数
- 输出为一行，以换行符结束

样例输入

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

样例输出

```
1193.0
```

解题思路

- 装尽可能多的糖果 → 贪心!
  - 先放总重量大的糖果?

  → 重量大的不一定价值高
  - 先放总价值高的糖果?

  → 总价值高的重量可能很大
  - 如何贪心地选择最佳的糖果?
    - 选择“单位价值最大”

解题思路

- 由于每箱糖果可以任意组合
- 所以一定可以 放入所有糖果

  或者 驯鹿能承受最大容量被放满

  - 因此放入的总重量是确定的

  →只用选择放入单位重量价值高的糖果

  - 将糖果按单位重量价值从高到低排序，依次放入

  → 直到放满或者放完为止

参考程序

```
#include <stdio.h>
#include <iostream>
#include <algorithm>
using namespace std;

#define MAXN (100+10)

//每箱糖果
struct Box {
   int v, w; //价值和重量
   double density; //单位重量的价值
};
```

参考程序

```
int n, w;
Box boxes[MAXN];
double totw, totv;

//箱子按单位重量价值排序
bool operator < (const Box &a, const Box &b) {
   return a.density < b.density;
}
```

参考程序

```
int main(){

    scanf("%d%d", &n, &w);
    for ( int i = 0; i < n; ++i ){
        scanf("%d%d", &boxes[i].v, &boxes[i].w);
        boxes[i].density = 1.0*boxes[i].v / boxes[i].w ;
        //计算单位重量的价值
    }
```

参考程序

```
    sort(boxes,  boxes+n);  //按单位重量的价值排序
    totw = totv = 0;
    for (int i = n-1;i >= 0;--i) //按单位重量的价值从大到小依次放入
        if (totw + boxes[i].w <= w) { //未放满
            totw += boxes[i].w;
            totv += boxes[i].v;
        }else { //已放满
            totv += boxes[i].density * (w - totw);
            totw = w;
            break;
     }
     printf("%.1lf\n", totv);
     return 0;
}
```