

《算法设计与分析》 第2.2讲穷举法

山东师范大学信息科学与工程学院 段会川 2014年9月

目录

- □ 穷举法的定义
- □ 穷举法的通用算法
- □ 百元买白鸡问题的穷举法
- □ 素数测试问题的穷举法
- □ 0-1背包问题的穷举法
- □ TSP问题的穷举法

AND A SHOULD SELVEN

2

穷举法定义

□ 在计算机科学中,穷举搜索(exhaustive search)或蛮力搜索(brute-force search),也称为生成+测试法(generate and test),是一种非常通用的问题求解方法,该方法由两部分组成,一是系统化地枚举问题各种可能的候选解,二是检查每一个解是否满足问题的求解要求。

维基百科

第2.2讲穷举法

穷举法定义

- □ 进行蛮力搜索必须实现4个步骤,即首选(first),再选(next),验证(valid)和输出(output),它们必须以问题的实例为输入参数,实现下面具体的功能:
 - 1.first (P): 产生问题P的第一个候选解.
 - 2.next (P, c): 从当前候选解c顺次产生下一个候选解.
 - 3.valid (P, c): 检查候选解c是否为问题P的解.
 - 4.output (P, c): 如果c为P的解则将其输出.
- □ 再选(next)步骤必须也能判断是否还有下一个候选解,如果没有通常返回一个"空候选"("null candidate"),常以Λ表示。同样地,首选(first)步骤在实例P没有候选时也应该返回Λ.

维基百科

第2.2讲穷举法

4

穷举法的通用算法

- □ 算法名称: 通用穷举法(ExhaustiveSearch)
- □ 输入: 问题实例P
- □ 输出:问题的解
- \square 1: c \leftarrow first(P)
- \square 2: while $c \neq \Lambda$
- \square 3: if valid(P,c) then output(P, c)
- \Box 4: $c \leftarrow next(P)$
- ☐ 5: end while

第2.2讲穷举法

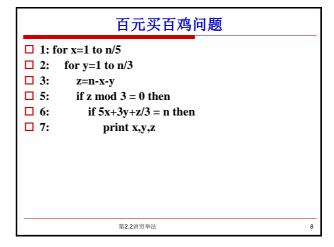
百元买百鸡问题

- □ 鸡翁一值钱5,鸡母一值钱3,鸡雏三值钱1。百钱买 百鸡,问鸡翁、母、雏各几何?"
- □ 算法问题: n元买n鸡问题
- □ 数学模型
 - x+y+z=n

第2.2讲穷举法

讲穷举法

百元买百鸡问题□ 1: for x=1 to n □ 2: for y=1 to n □ 3: for z=1 to n □ 4: if x+y+z=n then □ 5: if z mod 3 = 0 then □ 6: if 5x+3y+z/3 = n then □ 7: print x,y,z



素数测试—试除法(trial division) □ 试除法是测试—个数N是否为素数的蛮力方法 ■ 由于如果N有大于√N的因子p,则一定有一个小于√N因子q,因此只要用小于√N每个素数去试除N,如果找到一个数能够除尽N,则N就不是素数,如果所有的素数都除不尽N,则N必是素数 ■ 上述方法未考虑获得所有小于√N的素数的代价 ■ 也未考虑计算√N的代价



素数测试—朴素(na ive)试除法伪代码 | 1: ret = true | 2: i = 2 | 3: do | 4: if N MOD i = 0 | 5: ret = false | 6: break | 7: end if | 8: i = i+1 | 9: while i*i<=N

```
0-1背包问题的穷举法
□ vector<int> KSv;
□ void Knapsack(int n) {
if (n==0) {
      for (auto x:KSv)
cout << x:
cout << endl;
return; }
KSv.push_back(0);
Knapsack(n-1);
KSv.pop_back();
KSv.push_back(1);
Knapsack(n-1);
KSv.pop_back();
□ }
```

```
# Ctcygwinhome\HPDuankctbinDebug\ccexe argc: 1 argv[1]: /home/HPDuankctbinDebug\cc xe argv[1]:
```

```
TSP问题的穷举法
vector<int> TSPv;
                                void TSPt(int n)
void TSP(int i, int n) {
  if (i==n-1) {
                                  for (int i=0; i<n; ++i)
    for (auto x:TSPv)
                                    TSPv.push_back(i);
                                  TSP(0,n);
      cout << x+1;
    cout << endl:
    return;
  for (int j=i; j<n; ++j) \{
    swap(TSPv[i],TSPv[j]);\\
    TSP(i+1, n);
    swap(TSPv[i],TSPv[j]);
                   第2.2讲穷举法
```



