山东师范大学 SHANDONG NORMAL UNIVERSITY

《算法设计与分析》
第4.2讲 贪心法(2)

山东师范大学信息科学与工程学院
段会川
2014年11月

目录

- 部分背包问题及其贪心算法
- 0-1背包问题及其贪心算法
- TSP问题的贪心算法
- TSP问题的贪心有可能获得最差的解

部分背包问题 可分割背包问题

- 定义 分数 discrete
  - 部分背包问题(fractional knapsack problem)又称为连续背包问题(continuous knapsack problem)。
  - 给定n个重量为$w_1, w_2, \cdots, w_n$价值为$v_1, v_2, \cdots, v_n$的物品和容量为W的背包，在允许物品可以部分地装入背包的情况下，装入哪些物品可以获得最大的价值？

编号	1	2	3	4	5
重量	2	2	6	5	4
价值	6	3	5	4	6

背包容量：10　　王秋芬，P101

部分背包问题的贪心算法

- 将物品按照价值重量比倒序排列
- 按价值重量比由高到低依次向背包装入物品
- 当遇到不能完全装下的物品时，取其正好装满背包的部分装入背包

i	1	2	3	4	5
w	2	2	6	5	4
v	6	3	5	4	6
v/w	3	1.5	0.83	0.8	1.5

$W = 10$
解：1,2,5
重量：2+2+4=8
价值：6+3+6=15

i	1	2	5	3	4
w	2	2	4	6	5
v	6	3	6	5	4
v/w	3	1.5	1.5	0.83	0.8

i	1	2	5	3	tot
w^*	2	2	4	2	10
v^*	6	3	6	1.67	16.67

王秋芬，P101

部分背包问题贪心算法—伪代码

- 算法名称：部分背包问题贪心算法(FracKnapsackG)
- 输入：物品w[n], v[n],背包容量W　$x_i \in \{0,1\}$
- 输出：装入的物品比例x[n]和总价值V　$0 \le x_i \le 1$
- 1: 计算价值重量比数组p[n]并倒序排列
- 2: x[i]=0, i=1,…,n; V=0; j=1
- 3: while W>0
- 4: 　if w[j] <= W
- 5: 　　x[j] = 1; V += v[j]; W -= w[j]
- 6: 　else
- 7: 　　x[j] = W/w[j]; V += p[j]*x[j]; W=0
- 8: 　　break
- 5: end while

算法复杂度为排序的复杂度，即$O(nlogn)$

0-1背包问题的贪心算法1

- 重量价值比优先的贪心策略
  - 将物品按照价值重量比排列
  - 按价值重量比由高到低依次向背包装入物品
  - 当遇到不能完全装下的物品时，算法结束
- 算法复杂度为排序的复杂度，即$O(nlogn)$
- 该贪心算法不能保证获得最优解

i	1	2	3
w	5	20	10
v	50	140	60
v/w	10	7	6

$W = 30$
$X^* = (0, 1, 1)$
$W^* = 20 + 10 = 30$
$V^* = 140 + 60 = 200$

http://www.radford.edu/~nokie/classes/360/greedy.html

0-1背包问题的贪心算法2

- 价值优先的贪心策略
  - 将物品按照价值倒序排列
  - 按价值由高到低依次向背包装入物品
  - 当遇到不能完全装下的物品时，算法结束
- 算法复杂度为排序的复杂度，即$O(nlogn)$
- 该贪心算法不能保证获得最优解

i	1	2	3	4
w	24	10	10	7
v	12	9	9	5

$W = 25$
$X^* = (0, 1, 1, 0)$
$W^* = 10 + 10 = 20$
$V^* = 9 + 9 = 18$

http://www.radford.edu/~nokie/classes/360/greedy.html

0-1背包问题的贪心算法3

不具最优子结构性质

- 重量优先的贪心策略
  - 将物品按照重量排序
  - 按重量由低到高依次向背包装入物品
  - 当遇到不能完全装下的物品时，算法结束
- 算法复杂度为排序的复杂度，即$O(nlogn)$
- 该贪心算法不能保证获得最优解

i	1	2	3	4
w	24	10	10	7
v	12	9	9	5

$W = 25$
$X^* = (0, 1, 1, 0)$
$W^* = 10 + 10 = 20$
$V^* = 9 + 9 = 18$

http://www.radford.edu/~nokie/classes/360/greedy.html

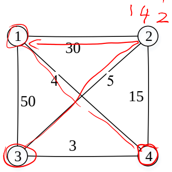
TSP问题的贪心算法—NN

n点排列 n! (n-1)!/2 Dijkstra n! (n-1)! 3!/2 = 3

- 最近邻(NN, Nearest Neighbor)优先的贪心策略
  1. 给定图G(V,E),计已经搜索的顶点集合为S,S初始化为出发顶点{s},则未搜索的顶点集合为V-S。
  2. 从V-S中搜索距离S中新加入顶点最近的顶点，并将该顶点加入到S中。
  3. 重复步骤2，直至不再有顶点可加入。
- 本算法也可看成是基于Prim算法思想的方法

王秋芬，P179，TSP
最优：1432, 42
NN结果相同

12341 42
12431 98
14231 74



TSP问题的贪心算法—NN伪代码

```
1: S = {s}, u = s; L = 0
2: while V-S is not empty
3:     x = ∞, v = nil
4:     for each (u,w) in E and w in V-S
5:         if d(u,w) < x then
6:             v = p
7:     if v = nil then stop
8:     else L += d(u,v); S += {v}
```

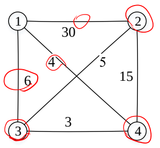
|E| |E|

- 算法复杂度：检查所有的边，即$O(|V|)$

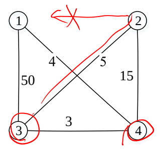
TSP问题的贪心算法—NN

- NN可能找不到最优。
- 在完全图的情况下一定可以找到回路，但一般图甚至还可能找不到回路。

王秋芬，P179，TSP
最优：1324, 30
NN：1432, 42



王秋芬，P179，TSP
最优：1423, 74
NN结果：无



TSP问题的贪心算法—Kruskal思想1

- 对图G(V,E)的所有边按权值排序
- 依次取最小边长的边放入F集合(结果边的集合)，但要满足下面两个条件
  - 新加入的边不会导致F集合中边对应的顶点的度超过2
  - 新加入的边不会产生环，除非它是第|V|条边
- 算法在无向完全图上可以正确执行，其它图不能保证获得一条路径
- 此算法的复杂度为：$O(N^2 \log N)$

8.4.1 A Greedy Algorithm for TSP

Johnson, David S., and Lyle A. McGeoch. "The traveling salesman problem: A case study in local optimization." Local search in combinatorial optimization 1 (1997): 215-310.
Haque, Albert, Jay Shah, Faisal Ejaz, and Jia Xing Xu. "An Empirical Evaluation of Approximation Algorithms for the Metric Traveling Salesman Problem.", 2013

TSP问题的贪心算法—Kruskal思想2

- 对图G(V,E)的所有边按权值排序
- 依次取最小边长的边放入F集合(结果边的集合)
- 检查新加入边的两个顶点在F集合决定的图中的度，如果出现度为2的顶点，则从E集合中删除与该顶点有关的边
- 算法在无向完全图上可以正确执行，其它图不能保证获得一条路径

Sándor Zoltán Németh, Heuristic Optimisation, Lecture 5: Greedy algorithms. Divide and conquer, P12, http://web.mat.bham.ac.uk/S.Z.Nemeth

TSP问题的贪心算法—Kruskal思想2

```
对边E按边上的权建立优先队列Q
F = {}, S = {} //F,S-结果边和顶点的线性表集合
while Q is not empty
    e_ij = ExtractMin(Q)
    F = F∪{e_ij}
    S = S∪{i,j}, 计算i,j的度d(i),d(j)
    if d(i)=2 then
        for each i 相关的边 f_ik
            Delete(Q, f_ik)
    if d(j)=2 then
        for each j 相关的边 f_jk
            Delete(Q, f_jk)
end while
```

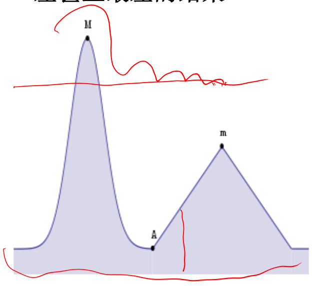
A B C

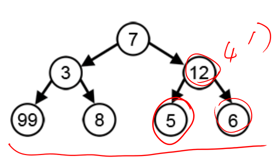
TSP问题的贪心算法—Kruskal思想2

- 算法复杂度 $O(|E|)$ $\log N^2 = O(\log N)$
  - 设顶点数$N = |V|$, 则边的数量为$|E| = \frac{1}{2}N(N-1) = O(N^2)$
  - 算法建立二叉堆优先队列的复杂度为$O(N^2)$ N^2
  - ExtractMin(Q)要执行N次，因而复杂度为$O(N\log N)$
  - 两个Delete操作共执行$|E| - N = O(N^2)$次，因而复杂度为$O(N^2\log N)$
  - 综上所述，算法复杂度为$O(N^2\log N)$

贪心算法可能会获得最差的结果

- 对非最优子结构问题施行贪心算法，有可能会得到较差甚至最差的结果





TSP问题的贪心算法可能会获得最长的路径

- **Gutin, Gregory, Anders Yeo, and Alexey Zverovich. "Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP." Discrete Applied Mathematics 117.1 (2002): 81-86.**

目录

- 部分背包问题及其贪心算法
- 0-1背包问题及其贪心算法
- TSP问题的贪心算法
- TSP问题的贪心算法有可能获得最差的解