



数据结构和算法

(Python描述)

郭 炜

微信公众号



微博: <http://weibo.com/guoweiofpku>

学会程序和算法，走遍天下都不怕!

讲义照片均为郭炜拍摄



北京大学
PEKING UNIVERSITY

信息科学技术学院

北京大学信息学院 郭炜

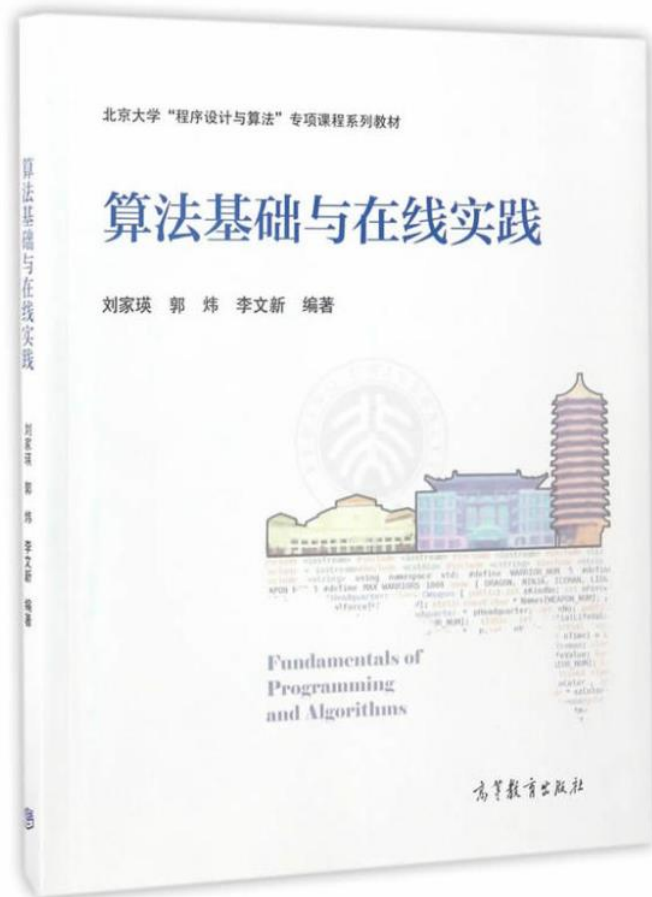
配套教材：

高等教育出版社

《算法基础与在线实践》

刘家瑛 郭炜 李文新 编著

本讲义中所有例题，根据题目名称在
<http://openjudge.cn>
“百练”组进行搜索即可提交





最短路



北京大学
PEKING UNIVERSITY

信息科学技术学院

北京大学信息学院 郭炜

最短路 Dijkstra 算法



美国加州太浩湖

基本思想

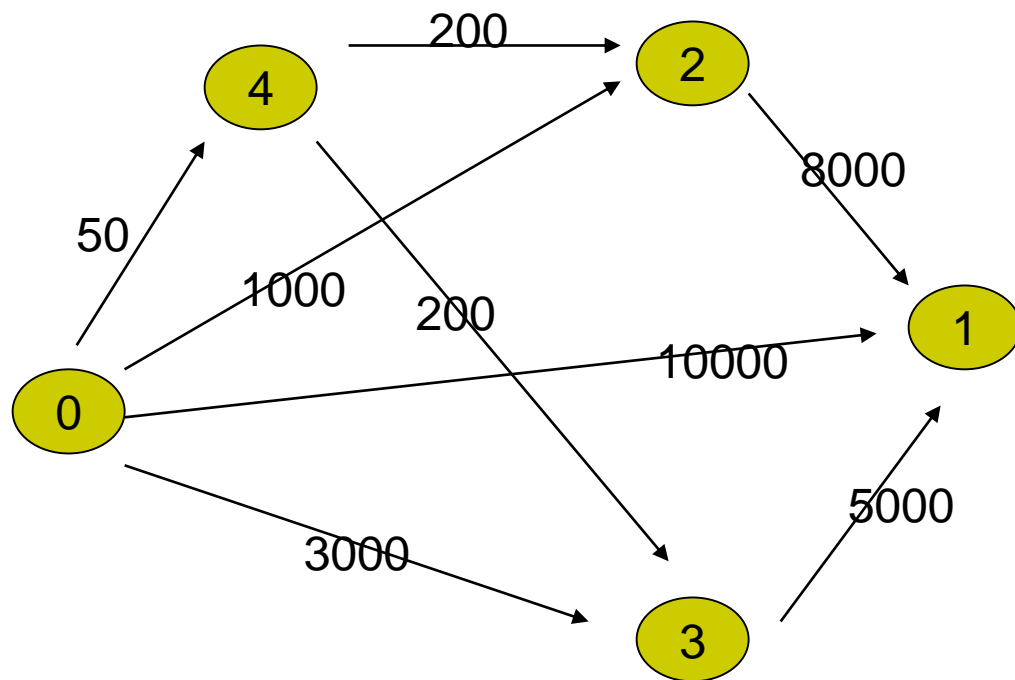
- 解决无负权边的带权有向图或无向图的单源最短路问题
- 贪心思想, 若离源点 s 前 $k-1$ 近的点已经被确定, 构成点集 P , 那么从 s 到离 s 第 k 近的点 t 的最短路径, $\{s, p_1, p_2 \dots p_i, t\}$ 满足 $s, p_1, p_2 \dots p_i \in P$ 。
- 否则假设 $p_i \notin P$, 则因为边权非负, p_i 到 t 的路径 ≥ 0 , 则 $d[p_i] \leq d[t]$, p_i 才是第 k 近。将 p_i 看作 t , 重复上面过程, 最终一定会有找不到 p_i 的情况
- $d[i] = \min(d[p_i] + \text{cost}(p_i, i)), i \notin P, p_i \in P$
 $d[t] = \min(d[i]), i \notin P$

Dijkstra's Algorithm

- $d[i]$ 表示 i 点到起点 s 的距离
- 初始令 $d[s]=0$, $d[i]=+\infty$, $P=\emptyset$
- 找到点 $i \notin P$, 且 $d[i]$ 最小
- 把 i 添入 P , 对于任意 $j \notin P$, 若 $d[i] + \text{cost}(i,j) < d[j]$, 则更新 $d[j] = d[i] + \text{cost}(i,j)$ 。

Dijkstra's Algorithm

- 用邻接表, 不优化, 时间复杂度 $O(V^2 + E)$
- Dijkstra+堆的时间复杂度 $O(E \lg V)$
- 用斐波那契堆可以做到 $O(V \lg V + E)$
- 若要输出路径, 则设置prev数组记录每个节点的前驱点, 在d[i]更新时更新prev[i]



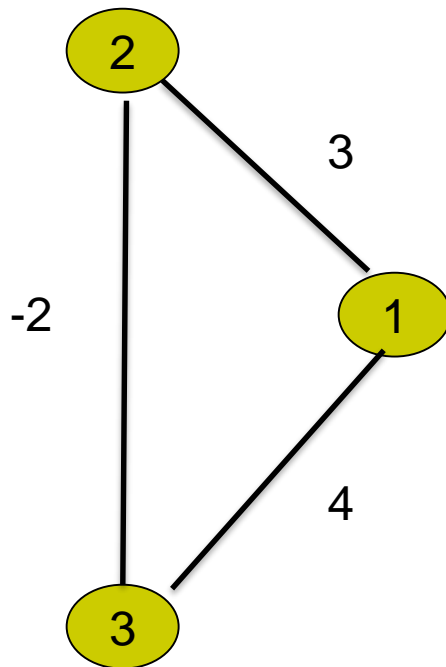
v	Dist[v]
0	0
1	18000 12500
2	12000 12500
3	32000 32500
4	50

Dijkstra's Algorithm

Dijkstra算法也适用于无向图。但不适用于有负权边的图。

$$d[1,2] = 2$$

但用Dijkstra算法求得 $d[1,2] = 3$



- 已经求出到 V_0 点的最短路的点的集合为 T
- 维护Dist数组, $\text{Dist}[i]$ 表示目前 V_i 到 V_0 的“距离”
- 开始 $\text{Dist}[0] = 0$, 其他 $\text{Dist}[i] = \text{无穷大}$, T 为空集
- 1) 若 $|T| = N$, 算法完成, Dist数组就是解。否则取 $\text{Dist}[i]$ 最小的不在 T 中的点 V_i , 将其加入 T , $\text{Dist}[i]$ 就是 V_i 到 V_0 的最短路长度。
- 2) 更新所有与 V_i 有边相连且不在 T 中的点 V_j 的Dist值:
 - $\text{Dist}[j] = \min(\text{Dist}[j], \text{Dist}[i] + W(V_i, V_j))$
- 3) 转到1)

POJ3159 Candies

有N个孩子 ($N \leq 3000$)分糖果。

有M个关系($M \leq 150,000$)。每个关系形如：

A B C (A,B是孩子编号)

表示A比B少的糖果数目，不能超过C

求第N个学生最多比第1个学生能多分几个糖果

POJ3159 Candies

思路：30000点，150000边的稀疏图求单源最短路

读入 “A B C” ，就添加A->B的有向边，权值为C

然后求1到N的最短路

```
import heapq
class Edge:
    def __init__(self,k=0,w=0):
        self.k ,self.w = k,w    #有向边的终点和边权值, 或当前k到源点的距离
    def __lt__(self,other):
        return self.w < other.w

bUsed = [0 for i in range(30010)]# bUsed[i]为1表示源到i的最短路已经求出
INF = 1000000000
N,M = map(int,input().split())
G = [[] for i in range(N+1)]
for i in range(M):
    s,e,w = map(int,input().split())
    G[s].append(Edge(e,w))
pq = []
heapq.heapify(pq)
heapq.heappush(pq,Edge(1,0))    #源点是1号点,1号点到自己的距离是0
```

```
while pq != []:
    p = pq[0]
    heapq.heappop(pq)
    if bUsed[p.k]: #已经求出了最短路
        continue
    bUsed[p.k] = 1
    if p.k == N: #因只要求1-N的最短路，所以要break
        break
    L = len(G[p.k])
    for i in range(L):
        q = Edge()
        q.k = G[p.k][i].k
        if bUsed[q.k]:
            continue
        q.w = p.w + G[p.k][i].w
        heapq.heappush(pq, q) #队列里面已经有q.k点也没关系
print( p.w )
```



北京大学
PEKING UNIVERSITY

信息科学技术学院

北京大学信息学院 郭炜

floyd算法



美国加州1号公路

弗洛伊德算法

- 用于求每一对顶点之间的最短路径。有向图，无向图均可。有向图可以有负权边，但是不能有负权回路。

弗洛伊德算法

- 用于求每一对顶点之间的最短路径。有向图，无向图均可。有向图可以有负权边，但是不能有负权回路。
- 假设求从顶点 v_i 到 v_j 的最短路径。如果从 v_i 到 v_j 有边，则从 v_i 到 v_j 存在一条长度为 $\text{cost}[i,j]$ 的路径，该路径不一定是最短路径，尚需进行 n 次试探。

弗洛伊德算法

- 用于求每一对顶点之间的最短路径。有向图，无向图均可。有向图可以有负权边，但是不能有负权回路。
- 假设求从顶点 v_i 到 v_j 的最短路径。如果从 v_i 到 v_j 有边，则从 v_i 到 v_j 存在一条长度为 $\text{cost}[i,j]$ 的路径，该路径不一定是最短路径，尚需进行 n 次试探。
- 考虑路径 (v_i, v_1, v_j) 是否存在（即判别弧 (v_i, v_1) 和 (v_1, v_j) 是否存在）。如果存在，则比较 $\text{cost}[i,j]$ 和 (v_i, v_1, v_j) 的路径长度，取长度较短者为从 v_i 到 v_j 的中间顶点的序号不大于1的最短路径，记为新的 $\text{cost}[i,j]$ 。

弗洛伊德算法

- 用于求每一对顶点之间的最短路径。有向图，无向图均可。有向图可以有负权边，但是不能有负权回路。
- 假设求从顶点 v_i 到 v_j 的最短路径。如果从 v_i 到 v_j 有边，则从 v_i 到 v_j 存在一条长度为 $\text{cost}[i,j]$ 的路径，该路径不一定是最短路径，尚需进行 n 次试探。
- 考虑路径 (v_i, v_1, v_j) 是否存在（即判别弧 (v_i, v_1) 和 (v_1, v_j) 是否存在）。如果存在，则比较 $\text{cost}[i,j]$ 和 (v_i, v_1, v_j) 的路径长度，取长度较短者为从 v_i 到 v_j 的中间顶点的序号不大于1的最短路径，记为新的 $\text{cost}[i,j]$ 。
- 假如在路径上再增加一个顶点 v_2 ，如果 (v_i, \dots, v_2) 和 (v_2, \dots, v_j) 分别是当前找到的中间顶点的序号不大于2的最短路径，那么 $(v_i, \dots, v_2, \dots, v_j)$ 就有可能是从 v_i 到 v_j 的中间顶点的序号不大于2的最短路径。将它和已经得到的从 v_i 到 v_j 的中间顶点的序号不大于1的最短路径相比较，从中选出中间顶点的序号不大于2的最短路径之后，再增加一个顶点 v_3 ，继续进行试探。依次类推。

弗洛伊德算法

- 在一般情况下, 若 (v_i, \dots, v_k) 和 (v_k, \dots, v_j) 分别是 v_i 到 v_k 和从 v_k 到 v_j 的中间顶点的序号不大于 $k-1$ 的最短路径, 则将 $(v_i, \dots, v_k, \dots, v_j)$ 和已经得到的从 v_i 到 v_j 且中间顶点的序号不大于 $k-1$ 的最短路径相比较, 其长度较短者便是从 v_i 到 v_j 的中间顶点的序号不大于 k 的最短路径。这样, 在经过 n 次比较后, 最后求得的必是从 v_i 到 v_j 的最短路径。按此方法, 可以同时求得各对顶点间的最短路径。
- 复杂度 $O(n^3)$ 。不能处理带负权边的无向图, 和有负权回路的有向图

弗洛伊德算法

- 记 $\text{dist}^k(i,j)$ 为从 V_i 到 V_j 的途经的顶点编号不大于 k 的最短路长度, 则有:

$$\text{dist}^1(i,j) = W_{ij} \text{ (} W_{ij} \text{是边}(i,j) \text{权值, 边不存在则为无穷大)}$$

$$\text{dist}^0(i,j) = \min\{\text{dist}^1(i,j), \text{dist}^1(i,0) + \text{dist}^1(0,j)\}$$

$$\text{dist}^1(i,j) = \min\{\text{dist}^0(i,j), \text{dist}^0(i,1) + \text{dist}^0(1,j)\}$$

.....

$$\text{dist}^k(i,j) = \min\{\text{dist}^{k-1}(i,j), \text{dist}^{k-1}(i,k) + \text{dist}^{k-1}(k,j)\}$$

.....

$$\text{dist}^{n-1}(i,j) = \min\{\text{dist}^{n-2}(i,j), \text{dist}^{n-2}(i,n-1) + \text{dist}^{n-2}(n-1,j)\}$$

其中 $\text{dist}^1(i,j)$ 表示从 V_i 到 V_j 的不途经任何顶点的最短路径长度。

$\text{dist}^{n-1}(i,j)$ 就是 V_i 到 V_j 的最短路的长度

弗洛伊德算法实现

```
def floyd(G): #G是邻接矩阵, 顶点编号从0开始算, 无边则边权值为INF
    n = len(G)
    INF = 10**9
    prev = [[None for i in range(n)] for j in range(n)]
    #prev[i][j]表示到目前为止发现的从i到j的最短路上, j的前驱。
    dist = [[INF for i in range(n)] for j in range(n)]
    for i in range(n):
        for j in range(n):
            if i == j:
                dist[i][j] = 0
            else:
                if G[i][j] != INF: #i到j的边存在
                    dist[i][j] = G[i][j]
                    prev[i][j] = i
```

弗洛伊德算法实现

```
for k in range(n):  
    for i in range(n):  
        for j in range(n):  
            if dist[i][k] + dist[k][j] < dist[i][j]:  
                dist[i][j] = dist[i][k] + dist[k][j]  
                prev[i][j] = prev[k][j]  
  
return dist, prev
```

最终: $\text{dist}[i][j]$ 就是 i 到 j 的最短路

$\text{prev}[i][j]$ 是 i 到 j 的最短路上 j 的前驱, $\text{prev}[i][\text{prev}[i][j]]$ 是 j 的前驱的前驱.....

例题： POJ3660 Cow Contest

N个选手，如果A比B强,B比C强，则A必比C强

告知若干个强弱关系，问有多少人的排名可以确定

- **Sample Input**

5 5

5个人，5个胜负关系

4 3

4 比3强

4 2

4 比2强

3 2

3 比2强

1 2

.....

2 5

- **Sample Output**

2

例题： POJ3660 Cow Contest

如果一个点 u , 有 x 个点能到达此点, 从 u 点出发能到达 y 个点, 若 $x+y=N-1$, 则 u 点的排名是确定的。用floyd算出每两个点之间的距离, 最后统计, 若 $\text{dist}[a][b]$ 无穷大且 $\text{dist}[b][a]$ 无穷大, 则 a 和 b 的排名都不能确定。最后用点个数减去不能确定点的个数即可。

模版例题

POJ1125