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

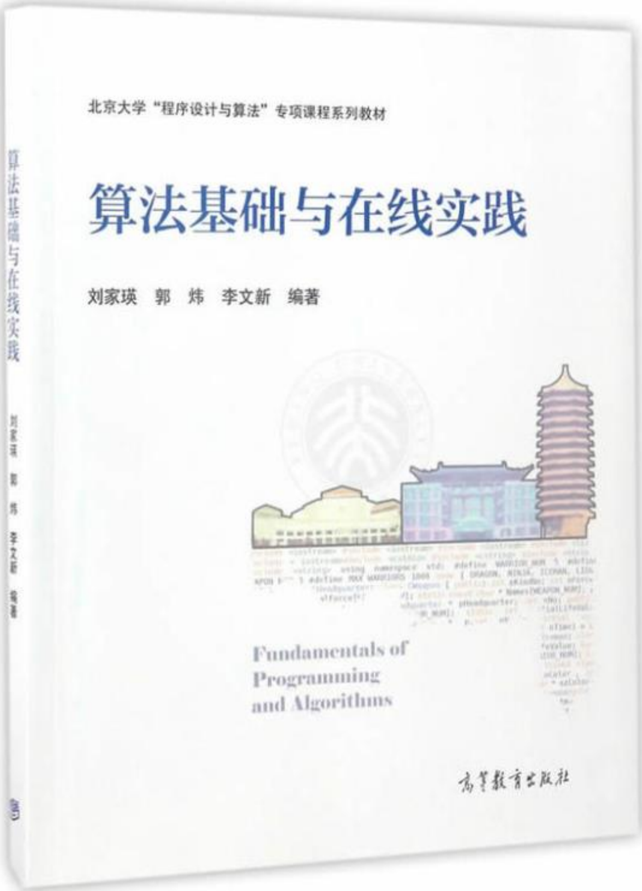
配套教材：

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



最短路和强连通分量

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，$\{s,p_1,p_2...p_i,t\}$满足$s,p_1,p_2...p_i \in P$。
- 否则假设$pi \notin P$，则因为边权非负，pi到t的路径≥0，则$d[pi] \le d[t]$,pi才是第k近。将pi看作t，重复上面过程，最终一定会有找不到pi的情况
- $d[i]=\min(d[p_i]+cost(p_i,i)), i \notin P, p_i \in P$

  $d[t]=\min(d[i]) , i \notin P$

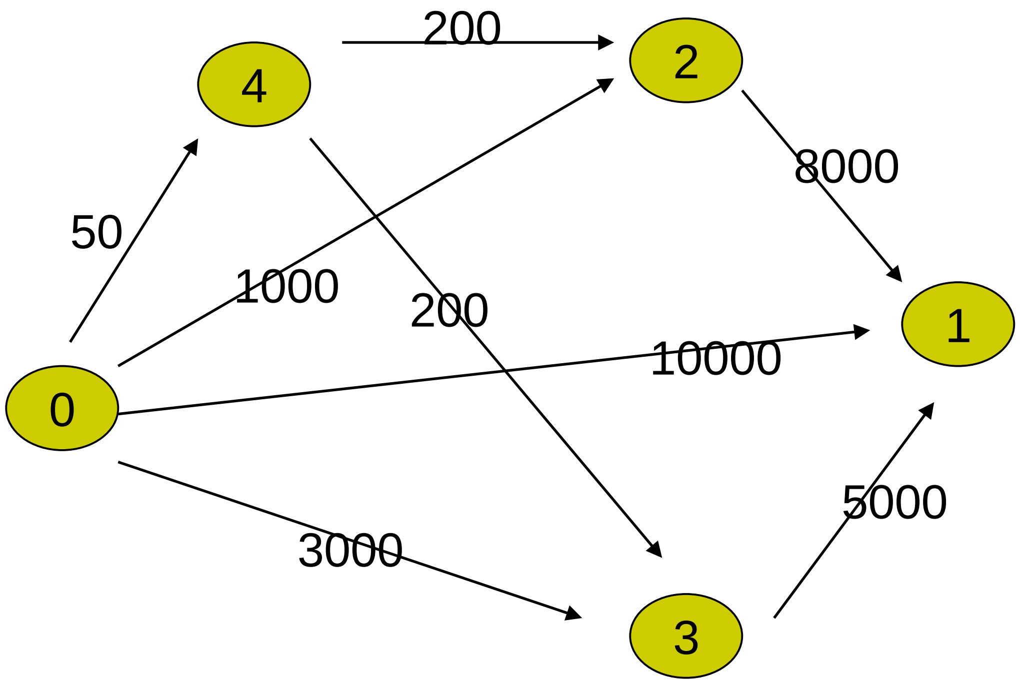
Dijkstra's Algorithm

- d[i]表示i点到起点s的距离
- 初始令d[s]=0，d[i]=+∞，P=∅
- 找到点i∉P，且d[i]最小
- 把i添入P，对于任意j∉P，若d[i]+cost(i,j)<d[j]，则更新d[j]=d[i]+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(ElgV)
- 用斐波那契堆可以做到O(VlogV+E)

- 若要输出路径，则设置prev数组记录每个节点的前趋点，在d[i]更新时更新prev[i]



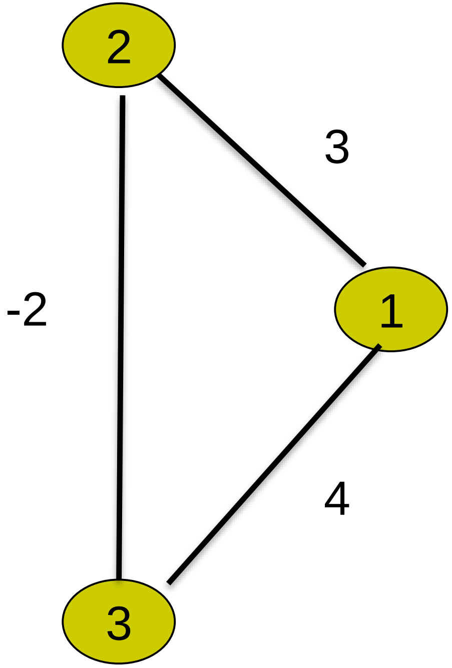
v	Dist[v]
0	0
1	[illegible]
2	[illegible]
3	[illegible]
4	50

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



Dijkstra算法实现

- 已经求出到V0点的最短路的点的集合为T
- 维护Dist数组，Dist[i]表示目前Vi到V0的“距离”
- 开始Dist[0] = 0, 其他Dist[i] = 无穷大, T为空集

- 1) 若|T| = N，算法完成，Dist数组就是解。否则取Dist[i]最小的不在T中的点Vi, 将其加入T，Dist[i]就是Vi到V0的最短路长度。
- 2) 更新所有与Vi有边相连且不在T中的点Vj的Dist值：
- Dist[j] = min(Dist[j],Dist[i]+W(Vi,Vj))
- 3) 转到1)

POJ3159 Candies

有N个孩子（N<=3000)分糖果。
有M个关系(M<=150,000)。每个关系形如：

A B C (A,B,C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

用heap实现 dijkstra + 堆的 POJ 3159 Candies

```
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 = 100000000
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
```

用heap实现 dijkstra + 堆的 POJ 3159 Candies


```
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 )
```

有向图的强连通分量

美国加州1号公路

有向图的强连通分量

- 在有向图G中，如果任意两个不同的顶点相互可达，则称该有向图是强连通的。有向图G的极大强连通子图(再加任何一个顶点就会不再强连通的子图）称为G的强连通分量。

转置图的定义

- 转置图的定义：将有向图G中的每一条边反向，形成的图称为G的转置G^T。图G和G^T的强连通分量是一样的。

Korasaju算法求有向图强连通分量

1) 深度优先遍历G，算出每个结点u的结束时间f[u],起点如何选择无所谓。

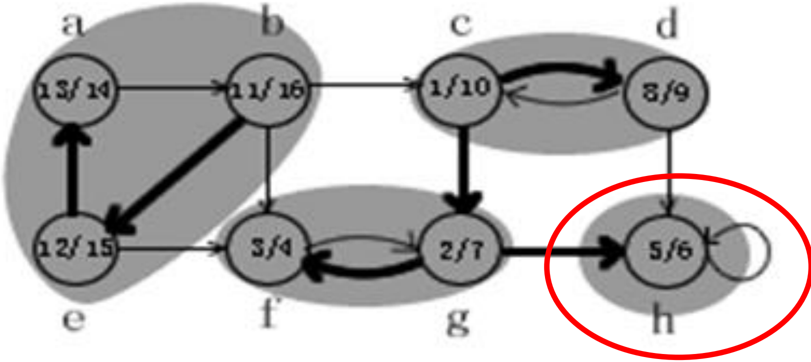
2) 深度优先遍历G的转置图G^T，选择遍历的起点时，按照结点的结束时间从大到小进行。遍历的过程中，一边遍历，一边给结点做分类标记，每找到一个新的起点，分类标记值就加1。

3) 第2)步中产生的标记值相同的结点构成深度优先森林中的一棵树，也即一个强连通分量

Korasaju算法求有向图强连通分量

(a)为有向图G：

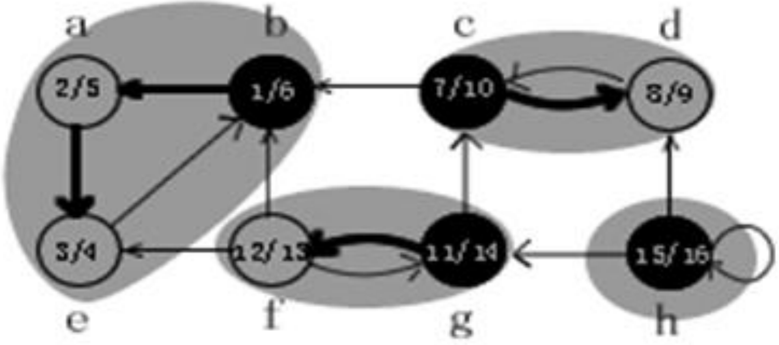
阴影部分是G的强连通分量，每个顶点都标出了其开始时间与结束时间，黑色边为深度优先搜索的树枝；



(a)

(b)G的转置图G^T：

依次以b,c,g,h为起点做DFS, 得到4个强连通分量



(b)

算法复杂度分析

- 深度优先搜索的复杂度:Θ(V + E)
- 计算G^T的复杂度:0或者Θ(V + E)(临接表)
- 所以总的复杂度为:Θ(V + E)

例题:POJ2186:Popular Cows

- 给定一个有向图，求有多少个顶点是由任何顶点出发都可达的。
- 顶点数<= 10,000,边数 <= 50,000

有用的定理:

有向无环图中唯一出度为0的点，一定可以由任何点出发均可达(由于无环，所以从任何点出发往前走，必然终止于一个出度为0的点)

POJ2186: 解题思路

- 1. 求出所有强连通分量
- 2. 每个强连通分量缩成一点，则形成一个有向无环图DAG。
- 3. DAG上面如果有唯一的出度为0的点,则该点能被所有的点可达。那么该点所代表的连通分量上的所有的原图中的点，都能被原图中的所有点可达，则该连通分量的点数，就是答案。
- 4. DAG上面如果有不止一个出度为0的点，则这些点互相不可达，原问题无解，答案为0

POJ2186: 解题思路

- 缩点的时候不一定要构造新图，只要把不同强连通分量的点染不同颜色，然后考察各种颜色的点有没有连到别的颜色的边即可(即其对应的缩点后的DAG图上的点是否有出边）。