Homework 9

PB17111623 范睿

2019 年 12 月 26 日

1

1.1 （欧拉回路）强连通有向图 G = (V,E) 中的一个欧拉回路是指一条遍历图 G 中每条边恰好一次的环路。不过，这条环路可以多次访问同一个结点。

1.1.1 a. 证明：图 G 中有一条欧拉回路当且仅当对于图中的每个结点 v, 有 $in_{degree}(v) = out_{degree}(v)$。

证明：($\Rightarrow$)

若G中存在一条欧拉回路，说明G中存在一条遍历图中每条边一次的环路。设此欧拉回路为$p = v_{q_0}v_{q_1}...v_{q_n}$，其中，$n = |E|$且$q_i \in V(i = 1...|V|)$。

$\forall v \in V$：

若v不是路径上两端的点，那么v在路径上出现的每一处，都是以v_pvv_q的形式，即v的前后都有某个不是v的顶点，其中v_pv是v的一条入边，vv_q是v的一条出边，一条入边一定与一条出边对应，则$in_{degree}(v) = out_{degree}(v)$。

若v是路径上两端的点，那么如果除去两端，v的情形与非两端的点的情形相同。设起始边为vv_q，终结边为v_pv，此两边相对应，那么当算上两端点时，$in_{degree}(v) = out_{degree}(v)$也成立。

($\Leftarrow$) 若图中每个顶点的入度=出度，那么从任何一个顶点出发，选择任何一条没有在环路中的边遍历，直到回到次顶点，一定可以找到一条欧拉回路。

1.1.2 b. 给出一个复杂度为 O(E) 的算法来找出图 G 的一条欧拉回路。

Algorithm 1

输入: $G = (V, E)$

输出: 欧拉回路p

```
let p[1...—E—] be a new array
index ←
Random select a vertex as v
while index is not |E| + 1 do
    p[index] ← v
    name the first not-in-path edge of v is (v, u)
    mark (v, u) as in-path
    v ← u
    index++
end while
return p
```

2

题目：（套利交易）套利交易指的是使用货币汇率之间的差异来将一个单位的货币转换为多于一个单位的同种货币的行为。例如，假定 1 美元可以购买 49 印度卢比，1 印度卢比可以购买 2 日元，1 日元可以购买 0.0107 美元。那么通过在货币间进行转换，一个交易商可以从 1 美元开始，购买 49*2*0.0107=1.0486美元，从而获得 4.86%的利润。
假定给定n种货币$c_1, c_2, ..., c_n$和一个 $n \times n$ 的汇率表 R, 一个单位的 c_i 货币可以购买 $R[i, j]$ 单位的c_j 货币。

2.1 a. 给出一个有效的算法来判断是否存在一个货币序列 $< c_{i_1}, c_{i_2}, ..., c_{i_k} >$, 使得$R[i_1, i_2] \times R[i_2, i_3] \times ... \times R[i_{k-1}, i_k] \times R[i_k, i_1] > 1$请分析算法运行时间。

思想：若想找到一个货币序列满足

$$R[i_1, i_2] \times R[i_2, i_3] \times ... \times R[i_{k-1}, i_k] \times R[i_k, i_1] > 1$$

相当于找到一个货币序列满足:

$$\log R[i_1, i_2] + \log R[i_2, i_3] + ... + \log R[i_{k-1}, i_k] + \log R[i_k, i_1] > \log 1 = 0$$

相当于找到一个货币序列满足:

$$(-\log R[i_1, i_2]) + (-\log R[i_2, i_3]) + ... + (-\log R[i_{k-1}, i_k]) + (-\log R[i_k, i_1]) < 0$$

那么利用汇率表构造一个有向带权完全图$G = (V, E)$，其中，$|V| = n$，即每个顶点代表一种货币。有一条边从i指向j，说明第i种货币可以被换成j货币，那么定义ij边的权重$\omega_{ij} = -\log R[i, j]$。此时问题变成了计算该图中是否存在负环。

Algorithm 2

输入: 汇率表R

输出: 是否存在满足条件的货币序列

```
1: G = CreateGraph(R)
2: for each vertex v ∈ G.V do
3:     ClearVertex(G.V) //initialize all vertex's distances as ∞
4:     v.d ← 0
5:     for i = 1 to |G.E| − 1 do
6:         Relex every edge in G.E
7:     end for
8:     if v.d < 0 then
9:         return True
10:    end if
11: end for
12: return False
```

复杂度分析：
创建图时间$\mathcal{O}(E+V)$，ClearVertex时间$\mathcal{O}(V)$,循环Relex的时间$\mathcal{O}(E^2)$，所以总时间为$\mathcal{O}(E+V+V*(V+E^2)\approx\mathcal{O}(V^2)$

2.2 b. 给出一个有效算法来打印出这样一个序列（如果存在这样一种序列），分析算法的运行时间。

思想：
若存在这样一个序列，即在上一问中的图中存在一个负环，那么如果再对G中的边所有进行一次按照负环的顺序的relax操作，负环上的顶点的d值还会改变。利用此性质找出负环。

复杂度分析：最坏情况下，该路径遍历了所有的顶点，且每个顶点在选下一个顶点的时候都是算到最后一个邻接点的时候才找到，这时，复杂度为$\mathcal{O}(E+V)$

Algorithm 3

输入: 图G，源s

输出: 负环p

let p[1...|$G.V$|] be a new array
$index \leftarrow 1$
$v \leftarrow s$
$p[index] \leftarrow s$
do
    index++
    for $(v, u) \in G.E$ **do**
        $OriginDistance \leftarrow u.d$
        $Relax(v, u, \omega_{vu})$
        if u.d is not OriginDistance **then**
            $p[index] \leftarrow u$
        end if
    end for
while p[index] is not s
return p

3

题目：假定在一个权重函数为ω的有向图上运行 Johnson 算法。证明：如果图 G 包含一条权重为 0 的环路 c, 那么对于环路 c 上的每条边 (u, v),$\hat{\omega}_{uv} = 0$。

证明：
设G中权重为0的环路为$p = v_1 v_2 ... v_{p-1} vp$，有：

$$\omega_{v_1 v_2} + \omega_{v_2 v_3} + ... + \omega_{v_{p-1} vp} = 0$$

运行完Jonhson算法后，设第i个顶点的对应值为$h(i)(i = 1...G.V)$，那么有：

$$\hat{\omega}_{v_1 v_2} = \omega_{v_1 v_2} + h(v_1) - h(v_2)$$
$$\hat{\omega}_{v_2 v_3} = \omega_{v_2 v_3} + h(v_2) - h(v_3)$$
$$...$$
$$\hat{\omega}_{v_{p-1} v_p} = \omega_{v_{p-1} v_p} + h(v_{p-1}) - h(v_p)$$

左右相加：

$$\hat{\omega}_{v_1 v_2} + ... + \hat{\omega}_{v_{p-1} v_p} = \omega_{v_1 v_2} + ... + \omega_{v_{p-1} v_p} + h(v_1) - h(v_2) + h(v_2) - ... + h(v_{p-1}) - h(v_p) =$$
$$\omega_{v_1 v_2} + ... + \omega_{v_{p-1} v_p} = 0$$
*由于是环路，所以$v_p = v1$

得到，此环路上所有边的新权重相加为0。
又由于新权重均大于等于0，所以环路上的每条边的新权重均为0。证毕。

4

题目：设 G = (V,E) 是一个源结点为 s 汇结点为 t 的流网络，其容量全部为整数值。假定我们已经给定 G 的一个最大流。

4.1 a. 如果将单条边 $(u, v) \in E$ 的容量增加 1 个单位，请给出一个 $\mathcal{O}(V + E)$ 时间的算法来对最大流进行更新。

Algorithm 4

输入： 已找到最大流的网络G，增加容量的单边(u,v)

输出： 更新流后的网络G

```
1: if (u,v).c > (u,v).f then
2:     (u,v).c++
3:     return G
4: else
5:     (u,v).c++
6:     if there exists a path p from s to t in the residual network G then
7:         c_f(p) = min{c_f(u,v) : (u,v)is in p}
8:         for each edge(u,v) in p do
9:             if (u,v) ∈ G.E then
10:                (u,v).f = (u,v).f + c_f(p)
11:            else
12:                (v,u).f = (v,u).f − c_f(p)
13:            end if
14:        end for
15:    end if
16:    return G
17: end if
```

4.2 b. 如果将单条边$(u, v) \in E$的容量减少 1 个单位，请给出一个 $\mathcal{O}(V + E)$ 时间的算法来对最大流进行更新。

Algorithm 5

输入: 已找到最大流的网络G，减少容量的单边(u,v)

输出: 更新流后的网络G

if $(u,v).c > (u,v).f$ **then**
    (u,v).c– –
    return G
else
    (u,v).c– –
    (u,v).f– –
    BFS find a path p1 from s to u
    for each edge (u,v) in p1 **do**
        (u,v).f– –
    end for
    BFS find a path p2 from v to t
    for each edge (u,v) in p2 **do**
        (u,v).f– –
    end for
    if there exists a path p from s to t in the residual network G **then**
        $c_f(p) = \min\{c_f(u,v) : (u,v) is\ in\ p\}$
        for each edge(u,v) in p **do**
            if $(u,v) \in G.E$ **then**
                $(u,v).f = (u,v).f + c_f(p)$
            else
                $(v,u).f = (v,u).f - c_f(p)$
            end if
        end for
    end if
    return G
end if