算法导论习题课讲义

作业解答

Author
舒文炫
USTC

June 28, 2023

Contents

1 hw4

1.1 P1

给定一颗树 T，在 $O(|T|)$ 的时间内找出 T 的最长非平凡路径。

Solution: 考虑从任意顶点 v 出发先找到距离其最远的顶点，然后再从该顶点找到离它最远的顶点，则这两顶点之间的路径就是最长的非平凡路径。

正确性说明：由于树的边数有限，最长路径总是存在的，并且最长路径端点必定是叶子节点，否则再加上该端点的孩子可以得到更长的路径。

我们将证明拆为两个简单的引理。

引理 1. *如果某顶点 v_1 恰好是某最长路径的一个端点，那么距其最远的点一定在其对应的某一条最长路径上。*

证明: 设从 v_1 出发距其最远的点为 v_n, 其中的路径 $P_1 = v_1...v_n$, 则 v_n 必定是其另一个端点，否则存在一个其他的端点 x, 使得 $P = v_1v_2...v_i...x$ 是该最长路径, 其中 v_i 是 P 首次离开 P_1 的节点. 结合 v_n 距离 v_1 最远，可以得到 $len(v_i...x) \geq len(v_i...v_n)$, 而 P 是最长路径，从而 $len(v_i...x) \leq len(v_i...v_n)$. 从而 $len(P_1) = len(P)$, 即得到 P_1 也是一条最长路径。即 v_n 在某条最长路径上。

引理 2. *从任意一个顶点 v_1 出发，距离其最远的顶点一定在某条最长路径上。*

证明: 由引理，如果 v_1 恰好是某最长路径的端点的情况，已经得证。

我们证明 v_1 不是最长路径的端点的情况。此时设从 v_1 出发距其最远的点为 v_n, 其中的路径 $P_1 = v_1...v_n$。

假设还有一条最长路径 $P_2 = s_1s_2...s_m$。则 P_1, P_2 有两种情况，其一，相交, 即存在一个 i_1, j_1, l 使得 $v_{i_1+h} = s_{j_1+h}, \forall h \leq l$。然后根据 v_n 距离 v 最远，可以得到 $len(v_{i_1+l}...v_n) \geq len(v_{i_1+l}...s_m)$。从而可以得到

$$len(s_1...v_{i_1})+len(v_{i_1}...v_{i_1+l})+len(v_{i_1+l}...v_n) \geq len(s_1...v_{i_1})+len(v_{i_1}...v_{i_1+l})+len(v_{i_1+l}...s_m)$$

记 $P_3 = s_1...v_{i_1}...v_{i_1+l}...v_n$, 也即 $len(P_3) \geq len(P_2)$ 而 P_2 是最长的路径，可以得到 $len(P_2) \geq len(P_3)$，从而 $len(P_3) = len(P_2)$，也即 P_3 是一条最长路径。而 v_n 是其中端点。得证。

第二种情况，不相交。注意到 v_n 肯定是叶子节点，从 v_n 到 s_1 有一条路径，和 P_1 交于 v_{i_2} 和 P_2 交于 s_{j_2}。类似的分析可以得到 $P_4 = s_1...s_{j_2}...v_{i_2}...v_n$ 是一条最长路径，从而同样得证。

结合这两个引理，原问题的正确性可以得到说明。

时间复杂度，由于只需要两次 dfs，时间复杂度为 $O(|T|)$. 算法大致如下

Algorithm 1: DFS

```
Input: 树 T，节点 V, 当前距离 l
Output: 最长非平凡路径 P
// 这里考虑用一般图的数据结构，方便从任意节点开始做 DFS，树还得用 child，parent
全局变量 maxl 初始化为 0，P1,P 初始化为空,P1 保存当前路径，P 保存最长路径;
if u==NULL then
    if l>maxl then
        maxl=l;
        P=P1;
        Return ;
    end
    P1.append(v);
    for u in v.adjancent do
        DFS(T,u,l+1);
        P1.delete(u);
    end
end
```

Algorithm 2: 最长非平凡路径

```
Input: 树 T
Output: 最长路径 P
任取一个节点 v;
P=DFS(T,v,-1);
P 最后一个节点即为最远节点，记为 u;
P=DFS(T,u,-1);
Return P;
```

1.2 P2

1 给定前序中序确定后序, 要求时间复杂度 $O(n)$。

简单分析，前序遍历第一个节点为树的根节点，如果可以在中序遍历中找到这个节点位置为 s，那么在其左边的部分为左子树中序遍历，右边为右子树中序遍历，先序遍历序列之后的 s 长度为左子树的先序遍历，剩下的为右子树的先序遍历, 从而可以很自然用递归的方式恢复出整颗树，自然也能给出其后序遍历序列，后序遍历序列为左子树

后序遍历序列 + 右子树后序遍历序列 + 根节点。算法如下。

Algorithm 3: PTW

Input: 序列 $P[1:l]=p_1p_2...p_l$，序列 $I[1:l]=i_1...i_l$, 当前序列长度 l

Output: 序列 $W[1:l]=w_1...w_l$

```
// find(A,a) 返回,a 在序列 A 中的位置
s=find(I,P[1]);
// P[2:s+1] 表示取出 P 的 2 到 s+1 构成的新序列
W1=PTW(P[2:s], I[1:s-1],s-1);
W2=PTW(P[s+1:l],I[s+1:l],l-s);
W=w1+W2+P[1];
return W;
```

Algorithm 4: 给定前序中序确定后序

Input: 序列 $P[1:n]=p_1p_2...p_n$，序列 $I[1:n]=i_1...i_n$, 当前序列长度 n

Output: 序列 $W[1:n]=w_1...w_n$

```
W=PTW(P[1:n],I[1:n],n);
return W;
```

时间复杂度因为每个节点只会访问一次故为 $O(n)$, 满足题目要求.

2 给定前序后序确定中序。

实际上给定前序后序无法唯一确定一颗树，故无法给出其后序排列。反例很容易，这里不再赘述。

1.3 P3

给定无向带权图 $G=(V,E,w)$, 找出代价最小的两颗生成树 T_1,T_2。

如果只要一颗最小生成树比较容易得到，可以使用 Prim 算法，使用二叉最小堆，时间复制度为 $O(ElogV)$。如果有不同的最小生成树，需要返回两个不同的最小生成树，否则返回的是一个最小生成树以及一个次小生成树。要得到次小生成树，我们考虑基于如下的定理。

Theorem 1.1. 如果 $|V|\le|E|$, 设 T 为 G 的一颗最小生成树，则图 G 中包含边 $(u,v)\in T$ 和边 $(x,y)\notin T$, 使得 $(T-\{(u,v)\})\cup\{(x,y)\}$ 是 G 的一颗次优生成树。

证明: 考虑在生成最小生成树的过程中产生的一个切割 $(S,V-S)$, 跨越切割的边如果存在两条或以上，在生成最小生成树的过程中，我们选择的是权值最小的边，可以记为 (u,v), 为了生成次优的生成树，我们选择另一条边 (x,y), 满足 $w_{xy}\ge w_{uv}$，否则我们只会得到最小生成树。则我们得到一个新的生成树 $T'=(T-\{(u,v)\})\cup\{(x,y)\}$。如果改变了多条边只会使得权值更大，不满足次优的定义。因此这里有且只有一条边被改变。所有这样得到的生成树取总权值最小的，即得到了次优生成树。

从而我们的算法可以考虑从最小生成树出发，每次添加一条不在最小生成树中的边，此时会生成一个环，找到环除去加入的边之外权值最大的边，将其删去，记录此时的权值，最后找到所有这样生成的树权值最小的，即为我们所要的代价最小的第二颗生

成树。算法如下

Algorithm 5: 代价最小两颗生成树

Input: $G = (V, E, w)$

Output: 代价最小的两颗生成树 T_1, T_2

// 先调用 `prim` 算法得到最小生成树以及其对应的权值。

T_1,weight=Prim(G);

// 这里实现可以给所有边初始化 `flag=0`，然后在 `prim` 算法中对所有在最小生成树的边标记一个 `flag=1`，然后从任意点进行 `DFS`，只选出 `flag=0` 的点

for *e=(u,v) in* E/T_1 **do**

  p=Find_max_weight(u,v， T_1);

  // 返回最大边权值对应的边

  weight1=INF;

  weight2=weight-p.weight+e.weight;

  if *weight2<weight1* **then**

    weight1=weight2;

    $T_2 = T_1 - p + e$;

  end

end

Return T_1, T_2;

Algorithm 6: Find_max_weight

```
Input: u, v, T
Output: 最大权值对应的边
对每一个节点初始化 2^i 级节点祖先数组 v.father[i], 考虑递推公式
  v.father[i+1]=(v.father[i]).father[i]。;
初始化节点的深度，保存在 v.depth 中;
初始化节点到 2^i 节点祖先路径中最大权值对应的边的数组 v.max[i], 递推公式
  v.max[i+1]=argmax(v.max[i],v.father[i].max[i]);
if u.depth<v.depth then
    swap(u,v);
end
i=floor(log(u.depth));
maxe=u.max[0];
while u.depth!=v.depth do
    if u.father[i].depth<v.depth then
        i–;
    end
    else
        maxe=argmax(maxe,u.max[i]);
        u=u.father[i];
    end
end
for i=floor(log(u.depth));i>=0;i– do
    if u.father[i]!=v.father[i] then
        maxe=argmax(maxe,u.max[i],v.max[i]);
        u=u.father[i];
        v=v.father[i];
    end
end
return maxe;
```

这里考虑使用倍增的方法寻找最近公共祖先，并且在寻找最近公共祖先的过程中保持当前已知最大权值对应的边。初始化时间只需要 $VlogV$, 最后只需要 $logV$ 时间即能实现返回最大权值对应的边，最多执行 E 次, 所以总时间复杂度 $O(ElogV)$。

1.4 P4

考虑有向带权图 $G=(V,E,w)$，其中 w 表示 E 上的权重，有正有负。对图的路径 $(v_1,v_2,...v_k)$，定义它的乘积: $\prod_{i=1}^{k} w(v_i,v_{i+1})$ 其中要求 $(v_i,v_{i+1})\in E$。输入 G 和点对 (s,t)，请设计快速算法找出 s 至 t 的一条乘积最大路径。如果这样的路径不存在，请输出 N/A。

方法一： 首先分析一下路径中有环的情况，记环 C 的权值的乘积为 W_C，那么可以分为下面五种情况讨论，如果 $W_C>1$ 那么不断经过这个环可以使乘积不断增大，此

时乘积最大路径不存在。如果 $-1 \leq W_C < 1$, 此时经过此环会使乘积绝对值减小，故乘积最大的路径不会经过这个环。如果恰好为 1，乘积不会变，所以是否经过此环不影响结果。如果 $W_C < -1$ 经过此环偶数次可以是乘积不断增大，所以乘积最大路径不会存在。如果 $-1 < W_C < 0$ 此时经过此环偶数次会减小，所以乘积最大路径不能经过这个环。

综上，可以考虑修改 Bellman-Ford 算法，我们所需要注意的只有经过权值乘积绝对值大于 1 的环，对于负的环只需要经过两次就可以让权值增大，所以我们需要运行 BF 算法，其中需要松弛 $2|V|$ 次。为了判断乘积最大路径是否存在，还需要再运行一次这样的 BF 算法看权值乘积是否继续增加。

因为这里权值有正有负，所以在松弛时需要考虑每个顶点最小的乘积和最大的乘积，初始化为正无穷和负无穷。用 v.pmax 保存最大乘积路径的前驱，v.pmin 保存最小乘积路径的前驱。

Algorithm 7: relax

Input: 节点 u，v, 边权值 w(u,v)

if $v.max < max(u.max \times w(u,v), u.min \times w(u,v))$ **then**
    $v.max = max(u.max \times w(u,v), u.min \times w(u,v))$ v.pmax=u **if** $u.max \times w(u,v) > u.min \times w(u,v)$ **then**
        v.choose=max
    end
    else
        v.choose=min
    end
end
if $v.min > min(u.max \times w(u,v), u.min \times w(u,v))$ **then**
    $v.min = min(u.max \times w(u,v), u.min \times w(u,v))$ v.pmin=u **if** $u.max \times w(u,v) < u.min \times w(u,v)$ **then**
        v.choose=max
    end
    else
        v.choose=min
    end
end

最后如果 t.max 为负无穷，则 s 不可达 t，如果权值还会更新，则存在绝对值大于 1 的环，乘积最大路径不存在，其他情况下根据 choose 来判断通过 max 回溯还是 min 回溯，最后通过回溯得到乘积最大路径。

方法二： 考虑到权值中没有 0，同时权值中有正有负，我们先考虑绝对值的情况，如果有一个乘积绝对值最大的路径

$$|w_1 w_2 ... w_n| = max$$

，同时取一个对数，再加一个符号，即转化为求权值为 $-ln|w_i|$ 求和的最小值，也即将原图的权值替换为为 $-ln|w_i|$ 的最短路径。之后想去掉绝对值，考虑出现奇数个符号最后乘积为负，偶数个符号最后乘积为正，我们可以将这两种情况分开考虑。将边的权

值拆分成权值的绝对值 e.weight 和权值的符号 v.symbol。通过记录经过符号的次数的奇偶来分为正路径和负路径, 分别存储在 v.distance[0] 和 v.distance[1] 中。这里可以直接使用 Bellman-Ford 算法来更新，这样如果存在正的最短路径，就是我们想要的结果。如果原图没有正的最短路径, 也就是更新到最后 v.distance[0]=inf，但是存在负的路径，我们其实需要负的最长路径，这是可以通过重新考虑权值为 $ln|w_i|$ 的图，寻找负的最短路径得到。如果两种路径都不存在，那就不存在输出 N/A。当然如果 Bellman-Fold 判断出了负环，也输出 N/A。必要的 relax 需要调整如下

Algorithm 8: relax

Input: 节点 u，v, 边 e

if $e.symbol = 1$ **then**
    if $v.distance[1] > u.distance[0] + e.weight$ **then**
        $v.distance[1] = u.distance[0] + e.weight$;
        v.p[1]=u;
    end
    if $v.distance[0] > u.distance[1] + e.weight$ **then**
        $v.distance[0] = u.distance[1] + e.weight$;
        v.p[0]=u;
    end
end
else
    if $v.distance[1] > u.distance[1] + e.weight$ **then**
        $v.distance[1] = u.distance[1] + e.weight$;
        v.p[1]=u;
    end
    if $v.distance[0] > u.distance[0] + e.weight$ **then**
        $v.distance[0] = u.distance[0] + e.weight$;
        v.p[0]=u;
    end
end

实际上有环的话回溯也不是很方便，可能得用一个栈来记录多次经过同一个节点所选择的边。

2 hw5

2.1 P1

1. 给定二分图，设计算法找出最大匹配。分析时间复杂度与正确性。

方法一，匈牙利算法，图论里面已经介绍过，时间复杂度 O(VE)

方法二，考虑到转化为图的最大流问题，添加一个源点 s 和一个汇点 t，对于二分图 (X, Y, E), 添加 s 到 X 中所有点的有向边，将 X,Y 之间的无向边替换为 X 到 Y 的有向边，然后添加 Y 中点到 t 的有向边。对每条边我们赋最大容量为 1. 这样我们得到了新图 G，以及图 G 上每条边的容量限制。我们有如下引理：

引理 3. *如果二分图 (X,Y,E) 有一个匹配 M，那么 G 中存在一个整数值的流 f 使得 $|f| = |M|$。反之 G 中的整数值流 f 也可以对应二分图中的一个匹配 M，使得 $|f| = |M|$。*

如果二分图存在一个匹配 M，我们定义流 f 如下：如果 $(u, v) \in M$，则 $f(s, u) = f(u, v) = f(v, t) = 1$ 对于其他的边定义 $f(u, v) = 0$。容易验证 f 满足容量限制和流量守恒。横跨切割 $(\{s\} \cup X, \{t\} \cup Y)$ 的流量 $|f| = |M|$

如果 G 存在一个整数值的流 f，我们定义匹配

$$M = \{(u, v) : u \in X, v \in Y, f(u, v) > 0\}.$$

因为每个节点 u 只有一条进入的边 (s, u), 所有 u 至多有一个单位流量进入，根据流量守恒，至多有一个流量流出，但是 f 是整数值的，故流入 u 的一个单位流量至多只能从一条边流出，也就是恰好存在一个节点 $v \in Y$ 使得 $f(x, y) = 1$。对每个 $v \in Y$ 也有对称的结论，从而 M 是一个匹配。且 $|f| = |M|$.

为了说明说明可以从最大流推出最大匹配，我们还需要证明如果容量函数只能取整数值，那么用 Ford-Fulkerson 方法生成的最大流只能是整数值的，且对所有的节点 u, v 都有 $f(u, v)$ 是整数。

这点我们只需要注意 Ford-Fulkerson 算法在每次迭代是寻找增广路径，所增加的流量是增广路径上容量的最小值，而这个值为整数。从而迭代每次归纳可以得到结果。

最后我们只需要在新图 G 上运行 Ford-Fulkerson 算法，然后取出其中流量大于 0 且端点不是 s，t 的边，即构成原来二分图的最大匹配。转化时间复杂度 $O(V+E)$,Ford-Fulkerson 算法时间复杂度 $O(VE)$, 最后得到最大匹配时间复杂度 $O(V+E)$，从而总时间复杂度 $O(VE)$.

2. 给定二分图，请找出它的最大独立集，给出时间复杂度并提供严格的正确性分析。提示：考虑最小割与最大独立集的关系。

首先我们如果得到图的一个覆盖集 C，将该覆盖集中顶点从原图中删去，就能得到原图的一个独立集。且在原图中对独立集取补相应的也能得到原图的覆盖集。那么我们要找最大独立集，实际上就是找原图的最小覆盖集，然后取顶点集的补集。

为了找到原图的最小覆盖，我们证明最小覆盖数等于最大匹配数。

首先给定一个匹配 M，以及任意一个覆盖 C，我们有如果 $e = (u, v) \in M$, 即有 $u \in C$ 或者 $v \in C$, 从而 $|C| \geq |M|$。

如果我们从最大匹配 M 构造出一个点覆盖 C，使得 $|C| = |M|$ 即证明了最小覆盖数等于最大匹配数。

考虑二分图 (X, Y), 我们用第一题的算法先得到了该图的最大匹配 M，那么从 Y 中未被匹配且未被标记的点出发，按照经过一条未匹配边，一条匹配边的顺序遍历尽可能所有路径，将所有可能路径经过 X，Y 的点标记，最后取 X 中被标记的点和 Y 中未被标记的点取出来，即为该二分图的一个覆盖. 这个操作只需要从每个 Y 中寻找不在 M 中的且未被标记的点，做 DFS，并在选择边时按照交错的方法选择即可，时间复杂度在 O(VE).

如果 Y 中有点没有被匹配且没有被标记，那么一定会被选择作为交错路径的起点而被标记，所以 Y 中所有没有被标记的点一定被 M 匹配，同时对应 X 中的点如果没有被匹配，交错路径一定不会经过这个点，从而我们选出的点一定为匹配 M 其中边的一个端点，且由交错路径的选取不会存在选出 XY 的点为 M 中一条边的两个端点的情况，所有这个点集大小为 $|M|$.

下面证明所有边都被这样的点覆盖，实际上只需要证不存在顶点在 X 中未被标记而 Y 中被标记的边即可。如果这条边不在 M 中，我们就会从 Y 中该点出发经过该边到底 X 中对应的点，那么 X 中点会被标记，如果这条边在 M 中，那么我们就不会将其作为交错路径的起点，但是其有标记，那么只能是从 X 中经过这一条边过来，这样 X 中对应的点一定会被标记。从而所有的边都被该点集覆盖了。

综上我们的算法，先使用 1 中算法得到最大匹配 M，基于该最大匹配 M 从 Y 中未被匹配且未被标记的顶点出发走交错路径，将沿途顶点标记，取 X 中标记顶点和 Y 中未被标记顶点，即为所求最小覆盖，最后将最小覆盖关于点集求补集得到最大独立集

2.2 P2

给定有向图 G = (V, E) 以及 $f : E \to R^+$ 与 $g : V \to R^+$ ，其中 f 代表经过边 e 的代价，g 代表访问节点 v 的收益。设计算法找出有向环 C 使得收益比 $\frac{\sum_{v\in C} g(v)}{\sum_{e\in C} f(e)}$ 最大。分析时间复杂度与正确性。

考虑如果存在一个最优解 r^*, 那么对于任意的环 C 我们都有

$$\frac{\sum_{v\in C} g(v)}{\sum_{e\in C} f(e)} \leq r^*$$

, 从而问题转化为

$$r^* \sum_{e\in C} f(e) - \sum_{v\in C} g(v) \geq 0$$

对所有的环都成了。反过来如果 $r \leq r^*$, 那么必存在一个环 C' 使得

$$r \sum_{e\in C'} f(e) - \sum_{v\in C'} g(v) \leq 0$$

，对于一个给定的 r, 考虑图 $G' = (V.E, w)$ 为其中 V, E 原图 G 的一个复制, 边 $e = (u, v)$ 权值定义为 $rf(e) - g(u)$, 这样如果 G' 中存在负环 C, 我们就有 $r \sum_{e\in C} f(e) - \sum_{v\in C} g(v) \leq 0$. 这时只需要取 $r = \frac{max_v g(v)}{min_e f(e)}$ 开始对图 G' 运行 Bellman-ford 算法，判断其中是否有负环，然后用二分的方法，最后可以在给定精度 ϵ 下得到结果，时间复杂度为 $O(VElog)\frac{max_v g(v)}{min_e f(e)}$. 这里取 $r = \frac{max_v g(v)}{min_e f(e)\epsilon}$ 是因为对于 $a, b, c, d \in R^+$, 如果 $\frac{a}{b} \leq \frac{c}{d}$ 那么 $\frac{a}{b} \leq \frac{a+c}{b+d} \frac{c}{d}$。

2.3 P3

问题 3 (25 分). 给定（无向）图 $G = (V, E)$ 以及 $f : E \to \mathbb{R}^+$ 表示每条边的距离。已知下列线性规划问题可用于求解 (s, t)-最短路径。

$$\max d_t$$

suject to

$$d_s = 0,$$
$$d_v \leq d_u + f((u, v)) \quad \forall (u, v) \in E,$$
$$d_v \geq 0 \quad \forall v \in V.$$

给出上述线性规划问题的对偶问题，并尝试通过最小费用流问题 (Min-Cost Flow) 解释该对偶问题可用于求解 (s, t)-最短路径。

这里感觉得是一个有向图，如果是无向图就会需要同时满足 $d_u \leq d_v + w(u, v)$ 和 $d_v \leq d_u + w(v, u)$, 这只能 $d_u = d_v$,$w(u, v) = w(v, u) = 0$ 感觉很奇怪

先把问题转成标准型

$$\begin{aligned}
&max\, d_t \\
&d_s \leq 0 \\
&d_u - d_v \leq f(u, v)\, \forall (u, v) \in E \\
&d_v \geq 0\, \forall v \in V
\end{aligned}$$

那么其对偶问题

$$\begin{aligned}
&min \sum_{(u,v) \in E} f(u, v) y_{(u,v)} \\
&\sum_v y_{(s,v)} + y_0 \geq 0 \\
&\sum_v y_{(u,v)} - \sum_v y_{(v,u)} \geq 0\, \forall (u, v), (v, u) \in E, u \in V/\{s, t\} \\
&\sum_v y_{(s,v)} - \sum_v y_{(v,s)} \geq 1 \\
&y_{(u,v)}, y_0 \in \{0, 1\}\, \forall (u, v) \in E
\end{aligned}$$

这里多出来的变量 y_0 是因为原问题有 $|E| + 1$ 个约束，不过在目标函数中并没有体现 y_0 因为其前系数为 0，所以可以不考虑这个变量。

如果我们对 $(u, v) \notin E$, 设置 $c(u, v) = 0, c(v, u) = 0$, 对 $(u, v) \in E$, 设置 $c(v, u) = 1, c(u, v) = 0$ 该条边的费用为 $f(u, v)$. 对于 s 设置 $c(s, v) = 0 \forall v \in V$. 那么上面对偶问

问题 4 (25 分). 考虑如下的线性规划问题 P:

$$\max c^T x$$

subject to

$$a_i^T \leq b_i \text{ for } 1 \leq i \leq m',$$
$$a_i^T = b_i \text{ for } m'+1 \leq i \leq m,$$
$$x_j \geq 0 \text{ for } 1 \leq j \leq n',$$
$$x_j \in \mathbb{R} \text{ for } n'+1 \leq j \leq n.$$

将上述问题表示成 $\max c'^T x'$ subject to $A'x \leq b'$, $x' \geq 0$ 的标准形式，并给出其对偶问题 D。
继续给出 D 的对偶问题，并证明其等于 P。

Figure 1: Caption

题还可以写为

$$\begin{aligned}
&min \sum_{(u,v)\in E} f(u,v) y_{(u,v)} \\
&y_{(u,v)} \leq c(u,v)\, \forall u,v \in V \\
&\sum_v y_{(u,v)} - \sum_v y_{(v,u)} \geq 0\, \forall u \in V/\{s,t\} \\
&\sum_v y_{(t,v)} - \sum_v y_{(v,t)} \geq 1 \\
&y_{(u,v)}, y_0 \geq 0, \forall (u,v) \in E
\end{aligned}$$

这里因为需要求最小值，我们只需要约束条件里面取等号就可以了，这其实就是认为将原图中所有边反向之后，求从 t 点净流量为 1，汇入到 s,$y_{(u,v)}$ 表示通过的每一条反向边的流量，每条边的费用为 $f(u,v)$ 的最小费用流问题。其结果即为 s 到 t 的最短路径长度。

2.4 P4

题目约束那一块少了 x_i, 应该是 typo

这一题按照标准型一步步转化为对偶，以及对偶的对偶即可，最后验证对偶的对偶恰好就是原问题，不想码公式了 orz

3 hw6

3.1 P1

证明欧拉定理

证明： 考虑 Z_n* 中所有元素，设为

$$x_1, x_2, ..., x_{\phi(n)},$$

其中每一个 x_i 满足 $gcd(x_i, n) = 1$。对每一个 x_i 考虑 $y_i = ax_i \, mod \, n$。由于 Z_n* 是模 n 乘法群，故 $y_i \in Z_n*$，下证每一个 y_i 各不相同，否则 $\exists i,j$,$y_i - y_j = 0$ 即 $a(x_i - x_j) \equiv 0 \, mod \, n$。

也即 $n|a(x_i - x_j)$ 但是 $gcd(n, a) = 1$, 由贝祖定理，$\exists x, y \in Z$,s.t.$xn + ay = 1$ 两边同乘 $(x_i - y_i)$, 可以证明 $n|(x_i - y_i)$ 然而这是不可能的，因为 $|x_i - y_i| < n$。推出矛盾，也即 y_i 各不相同，从而 y_i 是 x_i 的一个置换。那么我们有

$$
\begin{aligned}
x_1...x_{\phi(n)} &= y_1...y_{\phi(n)} \\
&= a^{\phi(n)} x_1..x_{\phi(n)}\, mod\, n
\end{aligned}
$$

从而 $a^{\phi(n)} \equiv 1 \mod n$

3.2 P2

设 (e, n) 与 (d, n) 为一对 RSA 公钥与私钥。张三将密文 P 加密为明文 $C = P^e\, mod\, n$，准备发送给李四。过程中，王五截获明文 C，将其伪造为 $S = Cr^e\, mod\, n$，并发送给李四。李四尝试通过私钥解密该信息，得到 $P = S^d\, mod\, n$。李四认为 P 是垃圾信息并丢弃，被王五得到。

1. 为什么李四会认为 P 是垃圾信息?

solution: 考虑数字签名，张三在发送消息 P 时会先使用函数 h 作用在 P 上，得到 $h(P)$ 然后使用私钥对其加密，并将这个与加密后的明文同时发送给李四。李四在接受到消息后，通过公钥解密消息摘要，然后解密明文计算哈希，对比发现哈希值不同，故认为这是垃圾信息。

2. 王五能不能通过已知信息破译 P?

这取决于 r 的选取

3. 为使王五能破译成功，r 需要如何选取?

因为李四按照私钥解密得到的是 $(Pr)^{ed} \equiv Pr\, mod\, n$, 这个被王五得到，如果我们可以使得 $Pr \equiv P\, mod\, n$, 那么王五就可以破译出密文 P。这时只需要选择 $r \equiv 1\, mod\, n$ 即可。

3.3 P3

给定一个无向图 G 和正整数 k，判断 G 中是否存在一条长度至少为 k 的简单路径 (经过每个顶点至多一次)。证明该问题是 NP 完全的。

solution: 定义 LONGEST-PATH={<G,k>: 图 G 存在一条长度至少为 k 的简单路径}，首先证明其为 NP 的。因为给定了一条长度为 k+1 的顶点串 $v_1 v_2 ... v_{k+1}$ 我们很容易在多项式时间内验证其是否为图 G 中的一条长度为 k 的简单路径。遍历一遍判断是否有重复的顶点并且对应的边是否在边集中即可。

此时限制 $k = |V| - 1$, 添加一个新顶点 u 同时对于图 G 中的所有顶点 v 连边 (v,u), 构成新的图 G ‘。这个操作只需要在 O(V) 的时间内做完。从而在 G’ 上如果存在哈密顿回路 $P = v_1 v_2 ... v_i u v_{i+1} ... v_n$, 那么将 $(v_i, u), (v_{i+1}, u)$ 删去，就得到原图 G 中长为 $|V| - 1$ 的的简单路径。反之也对。从而我们可以在多项式时间将哈密顿回路问题规约到存在长度为 $|V| - 1$ 的简单路径问题。由于哈密顿回路问题是 NPC 的，从而 LONGEST-PATH 是 NPC 的。

3.4 P4

已知问题 A 和问题 B 可互相多项式时间内 Karp 归约，若 A 是 NP 完全的，能否得出结论 B 也是 NP 完全的？若是请给出证明，若否请给出反例。

B 是 NP 完全的，下面给出证明。

首先证 B 是 NP 的，因为 $B \leq_P A$, 所以存在一个多项式时间可计算的函数

$$f\{0,1\}* \to \{0,1\}^*,$$

满足对所有的 $x \in \{0,1\}^*$, $x \in B$ 当且仅当 $f(x) \in A$. 而 A 是 NPC 的，所有 A 是 NP 的，其存在一个多项式时间算法 P，和一个多项式规模的证书 y，验证语言 A. 那么对于 $x \in B$, 我们可以在多项式时间内将其变成 $f(x) \in A$, 并通过 A 的多项式时间算法 P 验证 $f(x)$, 从而将 $P(f(x), y)$ 的输出作为 B 的验证算法的输出，这样我们得到了 B 的多项式时间验证算法，也即 B 是 NP 的。

然后证任意 NP 问题可以多项式时间规约到 B，这是因为任意 NP 问题 L$L \leq_P A$, 同时我们有 $A \leq_P B$, 由规约的传递性，我们得到 $L \leq_P B$.

综上，B 是 NPC 的。

3.5 P5

给定一个无向图 G = (V, E) 和正整数 k，G 的一个 k-因子是 G 的一个子图 H，满足对于任意 v   V , v 出现在 H 中且 v 的度数为 k。（1-因子即为图的完美匹配）给出判断一个图是否存在一个 k-因子的多项式时间算法

我们可以在多项式时间内解决一般图完备匹配问题 (blossom algorithm)，所以这里我们注意考虑将 k-factor 问题转化为另一个图上的完备匹配问题

考虑这样的规约，对于所有的 $v \in G$, 我们考虑拆成 $X(v) = \{x_1, ..v_{d(v)}\}$ 和 $Y(v) = \{y_1, ...y_{d(v)-k}\}$ 构成的完全二分图，这里是合理的，因为如果存在 $d(v) < k$, 那么可以直接判断 k-factor 不存在。同时对于 X(v) 中的点，还要和 X(u) 中对应的点连边，如果 $(u, v) \in E$。所有的 $X(u), Y(u)$ 以及对应的边构成图 G’。这样只需要关于图 G 规模的多项式时间构造出图 G‘.

下面证明图 G’ 上存在完美匹配等价与原图 G 上存在 k-factor。

如果 G’ 上存在完美匹配 M’，对每个 u 其 $Y(u)$ 中顶点只能和 $X(u)$ 中顶点相配，这样 $X(u)$ 中只能剩下 $d(u) - (d(u) - k) = k$ 个点要与其他 v 对应的 $X(v)$ 中的点相配，且每个 $X(v)$ 中有且只有一个点与 $X(u)$k 个点之一相配, 将这样的 $X(u)$ 中的点缩成原图中的 u 并且如果存在 $x_{ui} \in X(u), x_{vj} \in X(v)$ 使得 $(x_{ui}, x_{vj}) \in M'$, 我们就有 $(u, v) \in M$. 那么这样 M 就是原图 G 的一个 k-factor。

如果原图 G 存在一个 k-factor 记为 M, 那么对于 $e = (u, v) \in M$, 我们可以得到 $(x_{ui}, x_{vj) \in M'}$ 对于某个 i, j。那么对于 $X(u)$ 中剩下的 $d(u) - k$ 个点我们让其和 $Y(u)$ 对应的点分配相配，将其加入 M’ 中，这样我们得到了 M‘为图 G’ 的完备匹配。

综上，我们得到了多项式时间判断是否存在 k-factor 的算法。

References