离散数学习题课

2022 年 12 月 8 日

1 基本概念

1.1 一些定义

定义 1 一个无向图 G 是一个有序二元组 $G=(V(G),E(G))$, 其中,

- $V(G)=\{v_1,\dots v_n\}$ 是顶点集合, 任给 $v\in V(G)$ 称为一个顶点.
- $E(G)=\{e_i=\{v_{ia},v_{ib}\}|v_{ia},v_{ib}\in V\}$ 是边集合, 任给 $e\in E(G)$ 称为一条边.

一些其他定义:

- 邻居 $N(v)=\{v_i|\exists e\in E(G),e=\{v,v_i\}\}$
- 简单图.
- 完全图 K_n.
- 二分图 $V(G)=X\cup Y,x\cap Y=\varnothing$，$X$ 中任意两顶点不相邻，Y 中任意两顶点不相邻.
- 完全二分图 $K_{m,n}$.
- r 分图，完全 r 分图 $K_{n_1,\dots,n_r}$.
- 顶点度数 $d(v)$，连接 v 的边的个数，$\sum_{v\in V(G)}d(v)=2|E(G)|$.
- 同构 $G\cong H$，存在一一映射 $\theta:V(G)\to V(H)$，使得 $\{v_i,v_j\}\in E(G)\Leftrightarrow\{\theta(v_i),\theta(v_j)\}\in E(H)$.
- 子图，H 是 G 的子图：$V(H)\subseteq V(G),E(H)\subseteq E(G)$，记作 $H\subseteq G$.
- 道路（路径），$w=v_0\xrightarrow{e_1}v_1\xrightarrow{e_2}v_2\dots\xrightarrow{e_k}v_k$.
- 行迹，边不重复的道路称为行迹.
- 轨道，顶点不重复的道路称为轨道.
- 回路，起点与终点相同的道路称为回路.
- 圈，除了起点与终点相同之外, 没有相同顶点的回路称为圈.

- 距离 $d(u,v)$，u 到 v 最短轨道的长度。
- 连通，u,v 之间存在道路.
- 连通片，$V(G)=\bigcup_{i=1}^{\omega(G)} V_i, V_i\neq\varnothing, V_i\cap V_j=\varnothing, i\neq j$，使得 u,v 连通等价于存在 $V_i, u,v\in V_i$.
- 连通图，$\omega(G)=1$
- 树，无圈连通图.
- 叶子，树中度数为 1 的顶点.
- 生成树，$T\subseteq G$，T 是一棵树且 $V(T)=V(G)$.
- 二叉树，二叉正则树，二叉完全树.
- 匹配，完备匹配，最大匹配.
- 连通，强连通，单连通，弱连通.
- 强连通片，类比连通片.

1.2 一些定理

定理 1 G 是二分图 $\Leftrightarrow G$ 没有奇圈。

证明略

2 树

定理 2 设 $G=(V(G),E(G))$ 是简单无向图，则以下命题等价*:*

- G 是树*;*
- G 的任意两个顶点之间有且仅有一条轨道*;*
- G 不含圈，且 $|E(G)|=|V(G)|-1$*;*
- G 是连通图，且 $|E(G)|=|V(G)|-1$*;*
- G 是连通图，且删去任意一条边后都不连通*;*
- G 不含圈，且任意添加一条边后恰好含一个圈.

2.1 生成树

问题 1 如何求图 G 的生成树?

深度优先搜索（DFS），广度优先搜索（BFS），细节略。

问题 2 如何求图 G 的最小生成树?

2.1.1 Kruskal 算法

基于连通图的边数最多的无圈子图必然是生成树的思想，Kruskal(1956) 设计了一个求最小生成树的算法，对任意实数权都有效。此算法可称为"加边法"。初始最小生成树边数为 0，每一步加边都避开圈，在此基础上选择一条权最小的边加入到最小生成树的边集中。细节略。

定理 2.6 由 Kruskal 算法得到的生成子图 $T^*=(V(G),\{e_1,e_2,\cdots,e_{\nu-1}\})$ 是最小生成树.

证明 不难证明, 若 G 是连通图, Kruskal 算法结束时一定能够选出 $\nu-1$ 条边, 其选出的边构成的生成子图 $T^*=(V(G),\{e_1,e_2,\cdots,e_{\nu-1}\})$ 恰有 $\nu-1$ 条边且不含圈. 由定理 2.1 知, T^* 是一棵生成树. 下面用反证法证明, T^* 是最小生成树. 对于 G 的生成树 T, 若 T 与 T^* 不同, 则 T 与 T^* 的边不尽相同, 我们可以记录 T^* 中不在 T 中的边的最小下标, 即令 $f(T)=\min\{i|e_i\notin E(T)\}$. 若 T^* 不是最小生成树, 则对于所有的最小生成树 T 来说, T^* 与 T 的边集合一定不同, 从而上面定义的 $f(T)$ 存在.

设生成树 T' 是一棵使 $f(T')$ 最大的最小生成树. 因为 T^* 不是最小生成树, 所以 $\{e_1,e_2,\cdots,e_{\nu-1}\}$ 中必有不在 $E(T')$ 中的边. 设 $f(T')=k$, 即 $e_1,e_2,\cdots,e_{k-1}$ 同时在 T' 和 T^* 中, 但 e_k 只在 T^* 中, 不在 T' 中. 由定理 2.1, $T'+e_k$ 包含唯一的圈 C. 因为 T^* 是树, 不含圈, 所以圈 C 上至少有一条边不在 T^* 中, 不妨设为 e_k'. 因为 e_k' 在 $T'+e_k$ 的圈上, 所以 $T''=(T'+e_k)-e_k'$ 仍然连通, 而且恰有 $\nu-1$ 条边, 由定理 2.1 知, T'' 是 G 的另一棵生成树, 而 $w(T'')=w(T')+w(e_k)-w(e_k')$.

由 Kruskal 算法知, e_k 是使 $G[\{e_1,e_2,\cdots,e_{k-1},e_k\}]$ 为无圈图的权最小的边. 由于 $G[\{e_1,e_2,\cdots,e_{k-1},e_k'\}]$ 是 T' 的子图, 所以也不含圈, 故 $w(e_k)\leqslant w(e_k')$. 因此有 $w(T'')\leqslant w(T')$, 所以 T'' 也是一棵最小生成树. 而 $T''=(T'+e_k)-e_k'$, 意味着 $e_1,e_2,\cdots,e_{k-1},e_k\in E(T'')$, 所以 $f(T'')>k=f(T')$, 与 T' 的选法矛盾. 因此假设不正确, T^* 是一棵最小生成树. 证毕.

2.1.2 Prim 算法

Prim(1957) 提出了另一种不需要验证圈的最小生成树算法，称为"加点法"。每次迭代选择一条边，该边的一个端点已经被访问，另一个端点没有被访问，且权最小，将该边及其未访问的端点加入到最小生成树中。算法从某个顶点 s 开始，逐渐扩大直到覆盖整个连通图的所有顶点。细节略。

定理 2.7 由 Prim 算法得到的图 $T^* = (V', E')$ 是最小生成树.

证明 Prim 算法每次迭代从已访问顶点与其余未访问的顶点之间的边集 $(V', \overline{V'})$ 中选边, 因此所得子图 $T^* = (V', E')$ 不含圈, 且 $|E'| = |V'| - 1$, 由定理 2.1 知 T^* 是一棵生成树. 下面证明 T^* 是最小生成树.

假设 T 是 G 的一棵最小生成树. 若 $E(T) = E(T^*)$, 结论成立. 否则, 设 $e \in (V', \overline{V'})$ 是 Prim 算法生成 T^* 过程中第一条不在 T 中的边, 其中 $V' \subset V(G)$, 则 $T+e$ 含圈, 设为 C. $e \in E(C) \cap (V', \overline{V'})$, 则 T 中存在另一条边 $e' \in E(C) \cap (V', \overline{V'})$. 由 Prim 算法知, $w(e) \leqslant w(e')$. 因为 T 是最小生成树, 所以 $w(e) < w(e')$ 不成立, 否则将得到权更小的生成树 $T + e - e'$, 矛盾. 所以 $w(e) = w(e')$. 这样我们得到 G 的一棵生成树 $T + e - e'$, 使得 $w(T + e - e') = w(T)$, 且 T^* 的边 e 在 $T + e - e'$ 上. 类似地, 对每一条 $E(T^*) - E(T)$ 中的边重复上述过程, 最终使得 T^* 和 T 的边完全相同, 因此 T^* 也是最小生成树. 证毕.

问题 3 这两种算法的时间复杂度分别是多少?

2.1.3 破圈法

破圈法，是区别于避圈法 (Prim 算法和 Kruskal 算法) 的一种寻找最小生成树的算法，由 Rosenstiehl 和管梅谷分别于 1967 年和 1975 年给出，它"见圈破圈"，如果看到图中有一个圈，就去掉这个圈的一条边，直至图中不再有圈为止。细节略。

2.2 Huffman 树

定义 2 设二叉树 T 有 t 片树叶 $v_1, v_2, \ldots, v_t$，其权值分别为 $w_1, w_2, \ldots, w_t$ ，T 的加权路径长度 *(weighted path length)* 定义为: $WPL(T) = \sum\limits_{i=1}^{t} w_i L(v_i)$, 其中 $L(v_i)$ 为 v_i 的深度.

给定树叶的一组权值，可以构造出不同的二叉树，加权路径长度最短的树称为**最优二叉树**。

问题 4 *Huffman* 算法是什么? 时间复杂度是多少? 为什么 *Huffman* 算法得到的是一棵最优二叉树?

3 匹配

3.1 一些定义

定义 3 设 M 是图 G 的边子集，且 M 的任意两条边在 G 中都不相邻，则称 M 是 G 的一个匹配。M 中同一条边的两个端点称为在 M 中相配。M 中边的端点称为被 M 许配。若 G 中所有的顶点都被 M 许配，则称 M 是 G 的完备匹配。G 中边数最多的匹配称为 G 的最大匹配。若 M 是 G 的最大匹配，则称 M 中的边数 $|M|$ 为 G 的匹配数，记作 $\alpha(G) = |M|$.

定义 4 设 M 是图 G 的匹配，$P = v_0e_1v_1e_2\cdots e_kv_k$ 是 G 中的一条轨道，若 $e_1, e_2, \cdots, e_k$ 在 M 与 $E(G) - M$ 中交替出现，则称 P 是 G 中关于 M 的**交错轨道**。

设 $P = v_0e_1v_1e_2\cdots e_{2k+1}v_{2k+1}$ 是 G 中关于 M 的交错轨道, 若 $e_1, e_3, \cdots, e_{2k+1} \notin M, e_2, \cdots, e_{2k} \in M$, 且 v_0 与 v_{2k+1} 没有被 M 许配, 则称 P 是 G 中关于 M 的**可增广轨道**.

定理 3 M 是 G 的最大匹配, 当且仅当 G 中没有关于 M 的可增广轨道。

证明略。

3.2 二分图中的最大匹配算法

Algorithm 1 匈牙利 (Hungarian)

Input: 连通二分图 $G = X \cup Y$

1: 在 G 中任取初始匹配 M.
2: 令 X' 是 X 中没有被 M 许配的点的集合。
3: 若 $X' = \varnothing$, 止, M 即为 G 的最大匹配; 否则在 X' 中任取 u, 令 $S = \{u\}, T = \varnothing$.
4: 若 $N(S) = T, X' = X' - \{u\}$ 转 (3); 否则取 $y \in N(S) - T$.
5: 若 y 被 M 许配, 设 $yz \in M, S \leftarrow S \bigcup \{z\}, T \leftarrow T \bigcup \{y\}$, 转 (4) ; 否则取可增广轨 $P(u, y)$, 令 $M \leftarrow M \oplus E(P)$, 转 (2).

Output: 最大匹配 M

问题 5 匈牙利算法的时间复杂度? 正确性?

问题 6 二分图中匹配与覆盖有什么关系?

4 有向图

定义 5 一个有向图 G 是一个有序二元组 $G = (V(G), E(G))$, 其中,

- $V(G) = \{v_1, \ldots v_n\}$ 是顶点集合, 任给 $v \in V(G)$ 称为一个顶点.
- $E(G) = \{e_i = (v_{ia}, v_{ib}) | v_{ia}, v_{ib} \in V(G)\}$ 是边集合, 任给 $e \in E(G)$ 称为一条有向边.

定义 6 设 G 是有向图, 若存在从 u 到 v 的有向路径, 则称 u 可达 v。若 $\forall u, v \in V(G), u$ 可达 v 而且 v 可达 u, 即 u 与 v 双向可达, 则称 G 是**强连通的**; 若 $\forall u, v \in V(G), u$ 可达 v 或 v 可达 u, 则称 G 是**单向连通的**; 若 G 的底图是连通的无向图, 则称 G 是**弱连通的**。

定义 7 与无向图的连通类似, 双向可达在有向图 G 的顶点集 $V(G)$ 上也是一个等价关系。根据双向可达关系可以确定 $V(G)$ 的一个划分 $(V_1, V_2, \cdots, V_\omega)$, 由它们导出的有向子图 $G[V_1], G[V_2], \cdots, G[V_\omega]$, 称为 G 的**强连通片**。如果 G 只有一个强连通片, 则它是强连通的。

Algorithm 2 Kosaraju

Input: 有向图 G

1: 初始化 $SCC = \{\}$
2: 在 G 上运行 DFS 并记录每个节点 u 的完成时间 f_u
3: 对 $V(G)$ 按 f_u 降序排列
4: 建立图 G^T：将 G 所有边翻转
5: **repeat**
6: 　按 3中顺序对 G^T 进行 DFS
7: 　将得到的树添加进 SCC
8: **until** 所有节点都被访问

Output: 强连通分量 SCC

4.1 Kosaraju 算法

Kosaraju 的算法（也称为 Kosaraju-Sharir 算法）是线性时间的算法来找到一个有向图的强连通分量。

问题 7 *算法正确性如何保证？可不可以只用一次 DFS 得到结果？*

5 网络流理论

定义 8 *一个网络可以定义为一个四元组 $N = (G, s, t, c)$，其中：*

- *G 是一个弱连通的有向图；*
- *$s, t \in V(G)$，分别称为源与汇；*
- *$c : V(G) \times V(G) \to \mathbf{R}$ 为容量函数，任给 $e = (u, v) \in E(G), c(u, v) \geqslant 0$ 为边 e 的容量，对于 $(u.v) \notin E(G), c(u, v) = 0$.*

定义 9 *网络 $N = (G, s, t, c)$ 上的**流函数**为 $f : V(G) \times V(G) \to \mathbf{R}$，要求满足：*

- *任给 $u, v \in V(G)$，都有 $c(u, v) \geqslant f(u, v)$；*
- *斜对称，$f(u, v) = -f(v, u)$；*
- *任给 $u \in V(G) - \{s, t\}$，都有 $\sum\limits_{v \in V(G)} f(u, v) = 0$*

定义 10 *网络 $N = (G, s, t, c)$ 上的流函数为 f 的**流量**定义为：$|f| = \sum\limits_{v \in V(G)} f(s, v)$*

5.1 最大流问题

对于一个网络 $N = (G, s, t, c)$ ，我们想要求 N 上的流函数 f^* 使得流量 $|f^*|$ 最大化。

5.1.1 Ford-Fulkerson 算法

定义 11 一个网络 $N=(G,s,t,c)$ 关于一个流函数 f 的剩余**剩余网络**定义为 $N_f=(G_f,s,t,c_f)$，其中:

- $c_f(u,v)=c(u,v)-f(u,v)$ 为网络的剩余容量;
- $G_f=(V(G),E_f),E_f=(u,v)|c_f(u,v)>0$;

Algorithm 3 Ford-Fulkerson

Input: $N=(G,s,t,c)$

在 N 上任取一个流函数 f

repeat

    建立剩余网络 N_f

    在 G_f 上找到一条 s 到 t 的轨道 $P=s,v_1,\dots,t$，否则停止

    定义 $c_f(P)=\min\{c_f(u,v)|(u,v)\in P\}$

    定义 $f_P(u,v)=\begin{cases} c_f(P), & (u,v)\in P \\ -c_f(P), & (v,u)\in P \\ 0 \end{cases}$

    更新 $f(u,v)=f(u,v)+f_P(u,v)$

until 找不到 P

Output: 最大流函数 f

问题 8 *Ford-Fulkerson* 时间复杂度是多少，在哪些情况下可以保证找到最大流，可以怎么优化?

定义 12 给定一个网络 $N=(G,s,t,c)$，N 上的割是一个二元组 (S,T)，满足要求:

- $S\cap T=\varnothing,S\cup T=V(G)$
- $s\in S,t\in T$

$$\text{割的容量}c(S,T)=\sum_{u\in S,v\in T}c(u,v)$$
$$\text{割的流量}f(S,T)=\sum_{u\in S,v\in T}f(u,v)$$

定理 4 最大流等于最小割，即 $\max|f|=\min c(S,T)$

问题 9 最大流问题与二分图的最大匹配有什么联系?