Homework 2

Algorithm Design

- 3-2 Give an algorithm to detect whether a given undirected graph contains a cycle. If the graph contains a cycle, then your algorithm should output one. (It should not output all cycles in the graph, just one of them.) The running time of your algorithm should be O(m + n) for a graph with n nodes and m edges.

  Solution

  运用BFS算法以任意一个顶点为根结点，生成一棵BFS树，名为T

  如果图G的所有边都出现在了T中，那么说明G也是一棵树，并且G没有形成circle。

  若存在一些边 e(v, w) 属于 G 而不在 T 中，那么寻找v,w的公共祖先节点，且该节点的深度尽可能大，这样我们就能得到一个circle

- 3-7 Some friends of yours work on wireless networks, and they're currently studying the properties of a network of n mobile devices. As the devices move around (actually, as their human owners move around), they define a graph at any point in time as follows: there is a node representing each of the n devices, and there is an edge between device i and device j if the physical locations of i and j are no more than 500 meters apart. (If so, we say that i and j are "in range" of each other.)

  They'd like it to be the case that the network of devices is connected at all times, and so they've constrained the motion of the devices to satisfy the following property: at all times, each device i is within 500 meters of at least n/2 of the other devices. (We'll assume n is an even number.) What they'd like to know is: Does this property by itself guarantee that the network will remain connected?
  Here's a concrete way to formulate the question as a claim about graphs.

  Claim: Let G be a graph on n nodes, where n is an even number. If every node of G has degree at least n/2, then G is connected.

  Decide whether you think the claim is true or false, and give a proof of either the claim or its negation.

  Solution

  True.

  proof: Assume G is not connected, then G must have at least two SCC. let S be the SCC with fewest nodes, then $|S| < \frac{n}{2}$. Consider any node u in S, all its neighbor must also in S. Given the condition that every node of G has degree at least $\frac{n}{2}$, u has at least $\frac{n}{2}$ neighbors, then $|S| = \frac{n}{2} + 1$, which is a contradiction to $|S| < \frac{n}{2}$

- 3-9 There's a natural intuition that two nodes that are far apart in a communication network—separated by many hops—have a more tenuous connection than two nodes that are close together. There are a number of algorithmic results that are based to some extent on different ways of making this notion precise. Here's one that involves the susceptibility of paths to the deletion of nodes.
  Suppose that an n-node undirected graph G = (V , E) contains two nodes s and t such that the distance between s and t is strictly greater than n/2. Show that there must exist some

node v, not equal to either s or t, such that deleting v from G destroys all s-t paths. (In other words, the graph obtained from G by deleting v contains no path from s to t.) Give an algorithm with running time O(m + n) to find such a node v.

Solution

使用BFS算法，从s开始逐层搜索。设BFS算法搜索的每一层分别为$L_1, L_2, \ldots, L_d$,且由于s、t的距离大于$\frac{n}{2}$,所以$d > \frac{n}{2}$。这可以得出结论，必然存在1个L_i仅含1个顶点；否则，若每一层至少含两个顶点，从$L_1, \ldots, L_d$至少会有2 * $\frac{n}{2} = n$个顶点，再加上s和t顶点， 顶点总数矛盾。

根据BFS特性，设集合 $X = s \cup L_1 \cup L_2 \cup \ldots \cup L_{i-1}$, 可得 $v \notin X$, 所以离开X的边都会进入L_i中。而L_i又只包含1个顶点，所以删去该顶点后，L_{i-1}层不会有任何路径可以前往L_{i+1}，所以无法抵达t，所以破坏顶点v会破坏s-t所有路径。

算法概论

-

3.28. In the 2SAT problem, you are given a set of *clauses*, where each clause is the disjunction (OR) of two literals (a literal is a Boolean variable or the negation of a Boolean variable). You are looking for a way to assign a value `true` or `false` to each of the variables so that *all* clauses are satisfied – that is, there is at least one true literal in each clause. For example, here's an instance of 2SAT:

$$(x_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee \overline{x}_3) \wedge (x_1 \vee x_2) \wedge (\overline{x}_3 \vee x_4) \wedge (\overline{x}_1 \vee x_4).$$

This instance has a satisfying assignment: set x_1, x_2, x_3, and x_4 to `true`, `false`, `false`, and `true`, respectively.

(a) Are there other satisfying truth assignments of this 2SAT formula? If so, find them all.

(b) Give an instance of 2SAT with four variables, and with no satisfying assignment.

The purpose of this problem is to lead you to a way of solving 2SAT efficiently by reducing it to the problem of finding the strongly connected components of a directed graph. Given an instance I of 2SAT with n variables and m clauses, construct a directed graph $G_I = (V, E)$ as follows.

- G_I has $2n$ nodes, one for each variable and its negation.
- G_I has $2m$ edges: for each clause $(\alpha \vee \beta)$ of I (where α, β are literals), G_I has an edge from from the negation of α to β, and one from the negation of β to α.

Note that the clause $(\alpha \vee \beta)$ is equivalent to either of the implications $\overline{\alpha} \Rightarrow \beta$ or $\overline{\beta} \Rightarrow \alpha$. In this sense, G_I records all implications in I.

(c) Carry out this construction for the instance of 2SAT given above, and for the instance you constructed in (b).

(d) Show that if G_I has a strongly connected component containing both x and $\overline{x}$ for some variable x, then I has no satisfying assignment.

(e) Now show the converse of (d): namely, that if none of G_I's strongly connected components contain both a literal and its negation, then the instance I must be satisfiable. (*Hint:* Assign values to the variables as follows: repeatedly pick a sink strongly connected component of G_I. Assign value `true` to all literals in the sink, assign `false` to their negations, and delete all of these. Show that this ends up discovering a satisfying assignment.)

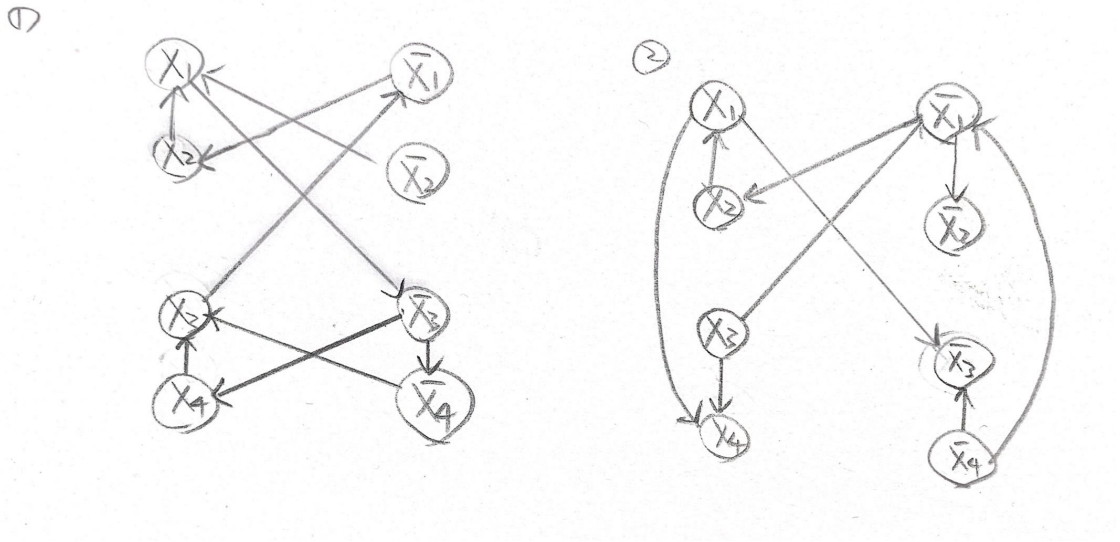
(f) Conclude that there is a linear-time algorithm for solving 2SAT.

Solution

(a) $x_1 = true,\ x_2 = true,\ x_3 = false,\ x_4 = true$

(b) $(x_1 \vee x_2) \wedge (x_1 \vee \bar{x_2}) \wedge (x_3 \vee x_4) \wedge (x_3 \vee \bar{x_4}) \wedge (\bar{x_1} \vee \bar{x_3})$

(c)



(d)

此命题显然成立。因为 x 与 $\bar{x}$ 在同一强连通分量中，因此存在x到$\bar{x}$的路径，也存在反向的路径。而又因为x与$\bar{x}$ 必有一者为true，不管是题中定义的激活路径还是一般的有向图路径，一定有一者是激活的，所以经过链式逻辑推导，命题x$\Rightarrow$ $\bar{x}$(或反之)必然成立是不可能的

(e) 只需要证明找到一种组合方式，为所有变量分配一个唯一值，且不至于产生冲突，即可解决2SAT问题。可以注意到，每次对汇强连通分量中的变量全部分配了true，对其反变量分配false，因为汇强连通分支为真时不会影响其他顶点的值，为假时不会影响子孙的取值。所以，可以把该强连通分支和其顶点的反变量从图中删除，剩余的图仍满足。

(f) 按照(e)方法执行。每找到汇强连通分量后，就执行赋值，删除操作。删除可采用标记法删除以节省时间。因为寻找强连通分量复杂度为O(m+n), 且赋值操作之多访问所有顶点一次，故为O(n)，所以总体的复杂度为O(m+n),所以复杂度为线性

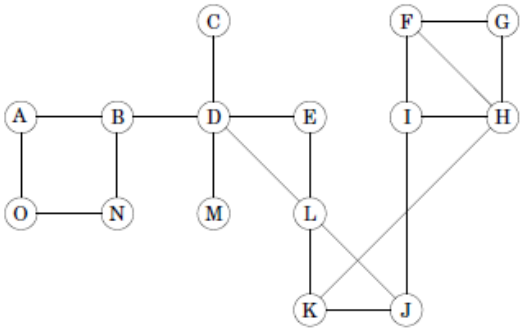
-

3.31. *Biconnected components* Let $G = (V, E)$ be an undirected graph. For any two edges $e, e' \in E$, we'll say $e \sim e'$ if either $e = e'$ or there is a (simple) cycle containing both e and e'.

(a) Show that $\sim$ is an equivalence relation (recall Exercise 3.29) on the edges.

The equivalence classes into which this relation partitions the edges are called the *biconnected components* of G. A *bridge* is an edge which is in a biconnected component all by itself.
A *separating vertex* is a vertex whose removal disconnects the graph.

(b) Partition the edges of the graph below into biconnected components, and identify the bridges and separating vertices.



Not only do biconnected components partition the edges of the graph, they also *almost* partition the vertices in the following sense.

(c) Associate with each biconnected component all the vertices that are endpoints of its edges. Show that the vertices corresponding to two different biconnected components are either disjoint or intersect in a single separating vertex.

(d) Collapse each biconnected component into a single meta-node, and retain individual nodes for each separating vertex. (So there are edges between each component-node and its separating vertices.) Show that the resulting graph is a tree.

DFS can be used to identify the biconnected components, bridges, and separating vertices of a graph in linear time.

(e) Show that the root of the DFS tree is a separating vertex if and only if it has more than one child in the tree.

(f) Show that a non-root vertex v of the DFS tree is a separating vertex if and only if it has a child v' none of whose descendants (including itself) has a backedge to a proper ancestor of v.

(g) For each vertex u define:

$$\texttt{low}(u) = \min \begin{cases} \texttt{pre}(u) \\ \texttt{pre}(w) \end{cases} \quad \text{where } (v, w) \text{ is a backedge for some descendant } v \text{ of } u$$

Show that the entire array of `low` values can be computed in linear time.

(h) Show how to compute all separating vertices, bridges, and biconnected components of a graph in linear time. (*Hint:* Use `low` to identify separating vertices, and run another DFS with an extra stack of edges to remove biconnected components one at a time.)

(a)

- 自反性：显然成立
- 对称性：若$e \neq e'$，它们始终在同一环中，故对称性成立
- 传递性：设存在 x y, y z, 设$x \neq y \neq z$，即x,y属于同一环C_1, y,z属于环C_2，因为C_1, C_2存在公共边y，所以任意C_1上的顶点都可以通过y到C_2，所以C_1, C_2构成大环，x z 传递性成立

(b) bridges: (B, D), (D, C), (D, M), separating vertices: B, D, L

(c)

若双连通分量C_1, C_2有超过1个公共点，则设2个顶点为u、v。根据双连通分量的定义。u、v分别在C_1, C_2中的路径为P_1, P_2,所以u，v形成了一个环。并且该环连接C_1, C_2，推出C_1, C_2也是双连通，矛盾。所以两个双连通分支不可能存在一个以上的公共点。若仅存在一个，显然他们不可能存在公共边，因为有公共边意味着至少存在2个公共点。因此，任何从C_1出发的路径必须经过公共点u才能进入C_2，反之亦然；删去该点会导致图不连通，因此该公共点为separating vertex

(d)

双连通分量除separating vertex顶点与separating vertex能到达的其他顶点间一定不存在边，于是将双连通分量汇集到其割点上构成的meta-node间不存在回路，即它们构成一棵树

(e)

充分性：若DFS数根节点为separating vertex，则删去该点后，图将不连通，将产生多个连通分量，这些连通分量原来的父节点均是根结点，所以DFS树根节点必有多余一棵子树

必要性：根据DFS算法，算法只有在完全遍历完根节点的一棵字数后，才会转向根节点的下一棵子树，所以，它们的父节点均是根节点，除去根节点后，它们互不连通，因此若根节点有多余一棵子树，则其为separating vertex

(f)

由于非根节点v有1个子节点v'，其所有子孙节点都与v的祖先节点连通考虑分别来自v' 以及子孙节点集与其余节点集中的亮点，期间的路径必然通过点v，将v移除时，上述两部分点不在连通，所以v为 separating vertex

(g)

增加一个数组pos[i]用于记录被访问的次序，则对图开始DFS。初始让low[i] = pos[i]。当节点u访问过试图访问u的子节点v时，若v已被访问，说明u-v为回边。若pos[v] < low[u], low[u] = pos[v], 在回溯的过程中不断更新节点low直到回溯结束，重复上述判断。这样可在1次DFS中获得low[], 时间复杂度为O(V+E) 为线性

(h)

准备3个数组分别存储separating vertex，bridge与biconnected components

对图进行DFS求出low与pos数组

对图中的每一个顶点u判断其子节点v, 若存在 pre (u) $\leq$ low(v),则说明u是一个separating vertex

之后执行第二次DFS，选取一条边e'入栈，此后对于栈顶的边e(a,b)判断：若a不为separating vertex， 将a的未访问邻边入栈；若b不为separating vertex，将b的未访问邻边入栈；若a，b均为separating vertex 或a，b全部邻边已被访问，则e出栈

每当栈空时，就是找到一个双连通分量，此后将该分量的所有顶点与边均删去，并且如果该双连通分量仅含有一条边，则他是一个bridge。

该算法是基于DFS，故总复杂度不超过O(V+E)为线性