d. Describe an algorithm that finds a maximum feasible flow in G. Denote by $MF(|V|, |E|)$ the worst-case running time of an ordinary maximum flow algorithm on a graph with $|V|$ vertices and $|E|$ edges. Analyze your algorithm for computing the maximum flow of a flow network with negative capacities in terms of MF.

26-7 The Hopcroft-Karp bipartite matching algorithm

In this problem, we describe a faster algorithm, due to Hopcroft and Karp, for finding a maximum matching in a bipartite graph. The algorithm runs in $O(\sqrt{V}E)$ time. Given an undirected, bipartite graph $G = (V, E)$, where $V = L \cup R$ and all edges have exactly one endpoint in L, let M be a matching in G. We say that a simple path P in G is an ***augmenting path*** with respect to M if it starts at an unmatched vertex in L, ends at an unmatched vertex in R, and its edges belong alternately to M and $E - M$. (This definition of an augmenting path is related to, but different from, an augmenting path in a flow network.) In this problem, we treat a path as a sequence of edges, rather than as a sequence of vertices. A shortest augmenting path with respect to a matching M is an augmenting path with a minimum number of edges.

Given two sets A and B, the ***symmetric difference*** $A \oplus B$ is defined as $(A - B) \cup (B - A)$, that is, the elements that are in exactly one of the two sets.

a. Show that if M is a matching and P is an augmenting path with respect to M, then the symmetric difference $M \oplus P$ is a matching and $|M \oplus P| = |M| + 1$. Show that if $P_1, P_2, \ldots, P_k$ are vertex-disjoint augmenting paths with respect to M, then the symmetric difference $M \oplus (P_1 \cup P_2 \cup \cdots \cup P_k)$ is a matching with cardinality $|M| + k$.

The general structure of our algorithm is the following:

```
HOPCROFT-KARP(G)
1  M ← ∅
2  repeat
3       let 𝒫 ← {P_1, P_2, ..., P_k} be a maximum set of vertex-disjoint
              shortest augmenting paths with respect to M
4       M ← M ⊕ (P_1 ∪ P_2 ∪ ··· ∪ P_k)
5    until 𝒫 = ∅
6  return M
```

The remainder of this problem asks you to analyze the number of iterations in the algorithm (that is, the number of iterations in the **repeat** loop) and to describe an implementation of line 3.

b. Given two matchings M and M^* in G, show that every vertex in the graph $G' = (V, M \oplus M^*)$ has degree at most 2. Conclude that G' is a disjoint union of simple paths or cycles. Argue that edges in each such simple path or cycle belong alternately to M or M^*. Prove that if $|M| \le |M^*|$, then $M \oplus M^*$ contains at least $|M^*| - |M|$ vertex-disjoint augmenting paths with respect to M.

Let l be the length of a shortest augmenting path with respect to a matching M, and let $P_1, P_2, \ldots, P_k$ be a maximum set of vertex-disjoint augmenting paths of length l with respect to M. Let $M' = M \oplus (P_1 \cup \cdots \cup P_k)$, and suppose that P is a shortest augmenting path with respect to M'.

c. Show that if P is vertex-disjoint from $P_1, P_2, \ldots, P_k$, then P has more than l edges.

d. Now suppose that P is not vertex-disjoint from $P_1, P_2, \ldots, P_k$. Let A be the set of edges $(M \oplus M') \oplus P$. Show that $A = (P_1 \cup P_2 \cup \cdots \cup P_k) \oplus P$ and that $|A| \ge (k+1)l$. Conclude that P has more than l edges.

e. Prove that if a shortest augmenting path for M has length l, the size of the maximum matching is at most $|M| + |V| / l$.

f. Show that the number of **repeat** loop iterations in the algorithm is at most $2\sqrt{V}$. (*Hint:* By how much can M grow after iteration number $\sqrt{V}$?)

g. Give an algorithm that runs in $O(E)$ time to find a maximum set of vertex-disjoint shortest augmenting paths $P_1, P_2, \ldots, P_k$ for a given matching M. Conclude that the total running time of HOPCROFT-KARP is $O(\sqrt{V} E)$.

Chapter notes

Ahuja, Magnanti, and Orlin [7], Even [87], Lawler [196], Papadimitriou and Steiglitz [237], and Tarjan [292] are good references for network flow and related algorithms. Goldberg, Tardos, and Tarjan [119] also provide a nice survey of algorithms for network-flow problems, and Schrijver [267] has written an interesting review of historical developments in the field of network flows.

The Ford-Fulkerson method is due to Ford and Fulkerson [93], who originated the formal study of many of the problems in the area of network flow, including the maximum-flow and bipartite-matching problems. Many early implementations of the Ford-Fulkerson method found augmenting paths using breadth-first search; Edmonds and Karp [86], and independently Dinic [76], proved that this strategy yields a polynomial-time algorithm. A related idea, that of using "blocking flows,"