Introduction to Algorithms May 14, 2004
Massachusetts Institute of Technology 6.046J/18.410J
Professors Erik Demaine and Shafi Goldwasser Handout 26

Problem Set 8 Solutions

This problem set is **not due** and is meant as practice for the final. *Reading:* 26.1, 26.3, 35.1

Problem 8-1. Prove these problems are NP-Complete:

(a) SET-COVER: Given a finite set $\mathcal{U}$, a collection $\mathcal{S} = \{S_1, S_2, \ldots S_m\}$ of subsets of $\mathcal{U}$, and an integer k, determine whether there is a sub-collection of $\mathcal{S}$ with cardinality k that covers $\mathcal{U}$. In other words, determine whether there exists $\mathcal{S}' \subset \mathcal{S}$ such that $|\mathcal{S}'| = k$ and $\bigcup_{S_i \in \mathcal{S}'} S_i = \mathcal{U}$.

Solution: The problem is in NP: to prove that there exists a sub-collection with cardinality k that covers the set $\mathcal{U}$, one can use the sub-collection $\mathcal{S}'$ as the certificate, which has polynomial size. To verify the certificate, check that the cardinality of $\mathcal{S}'$ is k and that every element in S is in one of the subsets in $\mathcal{S}'$. It takes polynomial time to verify the certificate.

The NP-Complete problem VERTEX-COVER is a special case of SET-COVER: the set $\mathcal{U}$ corresponds to the set of edges in the graph and each subset in $\mathcal{S}$ corresponds to a node in the graph and contains all the edges attached to the node. Since SET-COVER generalizes VERTEX-COVER, SET-COVER is NP-Hard.

(b) DIRECTED-HAMILTONIAN-PATH: given a directed graph $G = (V, E)$ and two distinct vertices $u, v \in V$, determine whether G contains a path that starts at u, ends at v, and visits every vertex of the graph exactly once. (Hint: Reduce from HAM-CYCLE: 34.5.3 in CLRS.)

Solution: It is clear that this is a problem in NP, since in order to prove that a digraph has a Hamiltonian path, it is sufficient to provide an ordering of the vertices; and to verify that a given ordering of the vertices represents a Hamiltonian path, it is sufficient to check that any two consecutive vertices v_i, v_{i+1} are joined by an edge (v_i, v_{i+1}).

We now need to show that this is an NP-hard problem. We use a reduction from HAM-CYCLE problem.

First, reduce Hamiltonian cycle to directed Hamiltonian cycle: suppose we are given an undirected graph $G = (V, E)$. Create a directed graph $G' = (V, E')$, where $(u, v), (v, u) \in E'$ if $(u, v) \in E$. If this new graph has a directed Hamiltonian cycle, then the original graph G must have a Hamiltonian cycle, and the other way around.

Then, reduce directed Hamiltonian cycle to directed Hamiltonian path: suppose we are given a digraph $G = (V, E)$. Construct a new graph $G' = (V', E')$ as follows: Pick and arbitrary vertex $v \in V$ and split it into two vertices: v_0 and v_1. Let $V' =$

$(V - v) \cup \{v_0, v_1\}$. For all edges $(v, u) \in E$, add an edge $(v_0, u) \in E'$. For all edges $(u, v) \in E$, add an edge $(u, v_1) \in E'$. For all other edges in E, copy them to E'.

We claim that G' has a Hamiltonian path from v_0 to v_1 if and only if G has a directed Hamiltonian cycle.

Suppose G has a Hamiltonian cycle. Then it is clear from the construction that G' will have a Hamiltonian path from v_0 to v_1.

Suppose G' has a Hamiltonian path from v_0 to v_1. Then there is a sequence of edges in E' that start at v_0 and end in v_1 and visit every node exactly once. From the way E' was constructed, it follows that there exists a path in G that starts at v and ends at v such that it visits every node $u \neq v$ exactly once and v exactly twice. Such a path in G is its Hamiltonian cycle.

Therefore it follows that the directed Hamiltonian path problem is NP-complete.

Problem 8-2. MAX-CUT Approximation

A *cut* $(S, V - S)$ of an undirected graph $G = (V, E)$ is a partition of V into two disjoint subsets S and $V - S$. We say that an edge $(u, v) \in E$ *crosses* the cut $(S, V - S)$ if one of its endpoints is in S and the other is in $V - S$. The MAX-CUT problem is the problem of finding a cut of an undirected connected graph $G = (V, E)$ that maximizes the number of edges crossing the cut. Give a deterministic approximation algorithm for this problem with a ratio bound of 2. *Hint:* Your algorithm should guarantee that the number of edges crossing the cut is at least half of the total number of edges.

Solution: The following algorithm maintains two disjoint subsets L and R of V, both initially empty. At each iteration, the algorithm looks at a new vertex v and places it either in L or R so as to maximize the number of edges going across these sets. That is, if v is adjacent to more vertices of L than of R, then v is placed in R, otherwise, it is placed in L.

```
APPROX-MAX-CUT(V, E)
1  L ← ∅
2  R ← ∅
3  for each vertex v ∈ V
4       do if |{u ∈ L : (v, u) ∈ E}| > |{u ∈ R : (v, u) ∈ E}|
5             then R ← R ∪ {v}
6             else L ← L ∪ {v}
7  return (L, R)
```

The final value of (L, R) is a cut of G, since every vertex is placed either in L or in R (but not in both). The time complexity of this algorithm is $O(|V| + |E|)$, since each edge is examined twice.

To see that the number of edges crossing the cut (L, R) is at least half the total number of edges, we prove the following invariant.

Invariant: Before and after each iteration, the number of edges going across (L, R) is at least as large as the number of edges inside L and R.

Proof. We prove this invariant by induction. The basis is trivial since L and R are empty. For the inductive step, assume that the invariant is true before the iteration that considers some vertex v. Denote by l the number of edges going from v to the vertices of L, and by r the number of edges going from v to the vertices of R. If $l > r$ then the algorithm places v in R. The number of edges going across (L, R) is increased by l, and the number of edges within L and R is increased by r. Thus, the invariant hold after the iteration as well. The case of $l \leq r$ is symmetric. This completes the proof.

The invariant states that the number of edges going across the cut is at least the number of edges inside L and R. This means that the number of edges crossing the cut is at least half of the total number of edges (this solves the hint). Since the number of edges crossing any cut of G, including the maximum one, is at most the total number of edges, this algorithm has a ratio bound of 2.

Problem 8-3. Global Edge Connectivity of Undirected and Directed Graphs

(a) The global edge connectivity of an *undirected* graph is the minimum number of edges that must be removed to disconnect the graph. Show how the edge connectivity of an undirected graph $G = (V, E)$ can be determined by running the maximum-flow algorithm $|V| - 1$ times, each on a flow network with $O(|V|)$ vertices and $O(|E|)$ edges.

Solution: Construct a directed graph G' from G by replacing each edge $\{u, v\}$ in G by two directed edges (u, v) and (v, u) in G'. Let $g(u, v)$ be the maximum flow value from u to v through G' with all edge capacities equal to one. Pick an arbitrary node u and compute $g(u, v)$ for all $v \neq u$. We claim that the edge connectivity equals $c^* = \min_{v \neq u} g(u, v)$. Therefore the edge connectivity of G can be computed by running the maximum-flow algorithm $|V| - 1$ times on flow networks each having $|V|$ vertices and $2\,|E|$ edges.

Suppose k is the edge connectivity of the graph and Q is the set of k edges such that removal of Q will disconnect the graph in two non-empty subgraphs G_1 and G_2. Without loss of generality assume the node $u \in G_1$. Let w be a node in G_2. Since $u \neq w$ the value $g(u, w)$ will be computed by the algorithm. By the max-flow min-cut theorem, $g(u, w)$ equals the min cut size between the pair (u, w), which is at most k since Q disconnects u and w. Therefore, we have

$$c^* \leq g(u, w) \leq k.$$

But c^* cannot be smaller than k since that would imply a cut set of size smaller than k, contradicting the fact that k is the edge connectivity. Therefore $c^* = k$ and the algorithm returns the edge connectivity of the graph correctly.

(b) The global edge connectivity of a *directed* graph G is the minimum number of directed edges that must be removed from G so that the resulting graph is no longer strongly connected. Show how the edge connectivity of a directed graph $G = (V, E)$ can be determined by running the maximum-flow algorithm $|V|$ times, each on a flow network with $O(|V|)$ vertices and $O(|E|)$ edges.

Solution: Label the $|V|$ nodes as $v_0, v_1, \ldots, v_{|V|-1}$ in an arbitrary order. Let $g(u, v)$ be the maximum flow value from u to v through G with all edge capacities equal to one. We claim that the global edge connectivity c^* for G is the following:

$$c^* = \min\{g(v_0, v_1), g(v_1, v_2), \ldots, g(v_{|V|-1}, v_0)\}$$

This implies that the global edge connectivity can be determined by $|V|$ max-flow computations on G.

First we prove that $c^* \leq \min\{g(v_0, v_1), g(v_1, v_2), \ldots, g(v_{|V|-1}, v_0)\}$:

$$\begin{aligned} c^* &= \min_{i,j \in V} g(i, j) \\ &\leq \min\{g(v_0, v_1), g(v_1, v_2), \ldots, g(v_{|V|-1}, v_0)\} \end{aligned}$$

Next we prove that $c^* \geq \min\{g(v_0, v_1), g(v_1, v_2), \ldots, g(v_{|V|-1}, v_0)\}$.

Suppose S is the minimum set of edges that needs to be removed from G to remove its strong connectivity. Let G' be the resulting graph. Since it is not strongly connected, there exists $u, v \in G'$ such that v is not reachable from u. Let P be the set of vertices in G' reachable from u and $Q = V \setminus P$. It follows that any $y \in Q$ is not reachable from any $x \in P$.

We show that there exists $i \in \{0, 1, 2, \ldots, |V|-1\}$ such that $v_i \in P$ and $v_{(i+1 \bmod |V|)} \in Q$. Let j be the smallest index such that $v_j \in Q$. If $j > 0$, then $v_{j-1} \in P$ and $v_j \in Q$ and we are done. Otherwise, $j = 0$ and we have two cases. If $v_{|V|-1} \in P$, then we are done as $v_{|V|-1} \in P$ and $v_0 \in Q$. Otherwise, let k be the smallest index such that $v_h \in Q$ for all $h \geq k$. Such k must exist since $v_{|V|-1} \in Q$. Also, we know that $k > 0$ since otherwise P would be empty. Therefore, we have $v_{k-1} \in P$ and $v_k \in Q$.

Finally we prove that the flow value $g(v_i, v_{i+1})$ is at most $|S| = c^*$. Suppose $g(v_i, v_{i+1}) > |S|$. By Max-Flow Min-Cut Theorem the minimum number of edges that must be removed from G to disconnect v_{i+1} from v_i is greater than $|S|$. This contradicts the fact that v_{i+1} is unreachable from v_i in G'. Therefore,

$$\begin{aligned} c^* &\geq g(v_i, v_{i+1}) \\ &\geq \min\{g(v_0, v_1), g(v_1, v_2), \ldots, g(v_{|V|-1}, v_0)\} \end{aligned}$$

Problem 8-4. Perfect Matching in Regular Bipartite Graph

A bipartite graph $G = (V, E)$, where $V = L \cup R$, is *d-regular* if every vertex $v \in V$ has degree exactly d.

(a) Prove that for every d-regular bipartite graph, $|L| = |R|$.

Solution: Every edge in the graph connect a node in L and a node in R. Therefore total number of edges can be expressed as both $|L|d$ and $|R|d$, which implies $|L| = |R|$.

(b) Model the maximum d-regular bipartite matching as a max-flow problem as in Section 26.3 in CLRS. Show that the max-flow value from s to t in the formulation is $|L|$.

Solution: Reduce the bipartite matching problem to a network flow problem, as in figure 26.3 in CLRS. Every edge has unit capacity. Consider the following flow function: every edge out of s or into t has unit flow, and all the other edges have flow $1/d$. This is a valid flow, and is maximum since every edge from s is saturated. Therefore, the max-flow value is $|L|$.

(c) Prove that every d-regular bipartite graph has a matching of cardinality $|L|$.

Solution: Since every edge in the max-flow formulation has integral capacity, the integrality theorem (Theorem 26.11) guarantees that there is an integral flow with value $|L|$. Such integral flow corresponds to a matching of cardinality $|L|$.