Homework 6 Solutions
CSI 503 Spring 2021

RAHUL BHENJALIA

May 10 2021

1. **Suppose that, in addition to edge capacities, a flow network has vertex capacities. That is each vertex v has a limit l(v) on how much flow can pass though v.**

   (a) **Show how to transform a flow network G = (V, E) with vertex capacities into an equivalent flow network G' = (V', E') without vertex capacities, such that a maximum flow in G0 has the same value as a maximum flow in G.**

   (b) **How many vertices and edges does G' have?**

   ANSWER:

   First, we construct G' by splitting each vertex v of G in three vertices: v_1: where all incoming edges are now incoming edges to v_1. v_2: where all outgoing edges are now outgoing edges from v_2. And they are joined by an edge capacity $l(v)$

   More clearly, we can say $G' = (V', E')$ has capacity function which is defined as follows:

   - For every $v \in V$ we are creating two vertices in $V' -> v_1, v_2$
   - We add an edge (v_1, v_2) in E' with $c'(v_1, v_2) = l(v)$
   - So, for every edge $(u, v) \in E$, we create an edge (u_2, v_1) in E' with capacity $c'(u_2, v_1) = c(u, v)$
   - We introduce s_1 and t_2 as new source and target vertices in G'
   - Clearly, from this we can say $|V'| = 2|V|$ (number of vertices)and $|E'| = |E|+|V|$ (number of edges), hence this answers our question in Part 1(b)

   Now for Part 1(a), let f be a flow in G that respects vertex capacities. Now, we create a flow function f' in G' with following conditions:

   - For each edge (u, v) belongs to G let $f'(u_2, v_1) = f(u, v)$.
   - For each vertex $u \in V–t$, let $f'(u_1, u_2) = \Sigma_{v \in V} f(u, v)$.
   - Also, let $f'(t_1, t_2) = \Sigma_{v \in V} f(v, t)$

- So without hesitation we can say that there is one to one correspondence between flows that respect vertex capacities in G and flows in G'.
- If we look at capacity constraint, every edge in G' of the form (u_2, u_1) has a corresponding edge in G with corresponding capacity and flow and thus, satisifies capacity constraint.
- Therefore, for $u \in V–s, t$ we have $f'(u_1, u_2) = \Sigma_{v \in V} f(u, v) \leq l(u) = c'(u_1, u_2)$.
- We also have $f'(t_1, t_2) = \Sigma_{v \in V} f(v, t) \leq l(t) = c'(t_1, t_2)$

---

2. **The edge connectivity of an undirected graph is the minimum number k of edges that must be removed to disconnect the graph. For example, the edge connectivity of a tree is 1, and the edge connectivity of a cyclic chain of vertices is 2. Show how to determine the edge connectivity of an undirected graph G = (V, E) by running a maximum-flow algorithm on at most —V— flow networks, each having O(V) vertices and O(E) edges.**

   ANSWER:

   First, we focus on finding edge connectivity, which is a problem in which there is application of maximum-flow minimum-cut theorem. Minimum-cut of a network G is the cut whose capacity is the least of all possible cuts of G. So, if the capacities of all edges are 1, then the $min - cut = edge - capacity$. But, if $min - cut = k$, edge connectivity is also k, which means G can be disconnected on removing k-edges.

   .

   Now, let us consider that maximum-flow algorithm runs on a network and return a max-flow of a residual graph. Hence, the edge connectivity problem can be solved using the max-flow algorithm by implementing following steps:

   - Let us say that for any two vertices u and v in G, a flow network G_{uv} with directed edges of capacity 1, can be defines such that, $s = u$ and $t = v$.
   - Here we can use a max-flow algorithm like FORD-FULKERSON, that can be applied to find max-flow of G_{uv}.
   - Now, as we established that all capacities are 1, the number of edges crossing a cut equals the capacity of the cut.
   - The following algorithm run with time complexity of $O(V)$ and finds the capacity of all possible cuts.

   .

   .

EDGE-CONNECTIVITY(G)
1. initially set $k = \infty$
2. arbitrarily select any vertex $u \in V$
3. for each vertex $v \in G.V - u$ do
4. setup the flow network G_{uv}
5. $f_{uv} = FORD - FULKERSON(G_{uv}, u, v)$ // finding max-flow of G_{uv}
6. $k = min(k, f_{uv})$
7. return k

- The algorithm finds minimum of all these capacities in **line-6**, which means that k edges are to be removed to disconnect the network.

---

3. **Let G = (V, E) be a flow network with source s, sink t and integer capacities. Suppose that we are given a maximum flow in G. Suppose that we increase the capacity of a single edge (u, v) ∈ E by 1. Give an O(V + E)-time algorithm to update the maximum flow.**

   ANSWER:

   - The solution can be achieved by executing one iteration of FORD-FULKERSON algorithm.
   - The edge (u, v) in E with increased capacity ensures that the edge (u, v) is in the residual graph. Hence, we must look for an augmenting path and update the flow if a path is found.
   - But we need to consider the cases in which edge (u, v) is or is not an edge that crosses a minimum cut
     - If the edge (u, v) is not an edge that crosses the minimum cut, then increasing its capacity does not change the capacity of any minimum cut. Hence, the values of max-flow does not change.
     - If the edge (u, v) does cross a minimum cut, then increasing its capacity by 1 increases the capacity of minimum cut by 1. Hence, possibly the value of the max-flow by 1. In this case, there is either no augmenting path, or the augmenting path increases flow by 1.
   - Conclusively, no matter what, one iteration of FORD-FULKERSON algorithm suffices, and the time complexity would be $O(V + E) = O(E)$, if we find the augmenting path with either DFS or BFS

---

4. **Show that if HAM-CYCLE ∈ P, then the problem of listing the vertices of a hamiltonian cycle, in order, is polynomial-time solvable.**

ANSWER:

.

. Let us start by defining an algorithm to find Hamiltonian cycles. Let us consider a $HAM-CYCLE = <G>$: $GisaHamiltoniangraph$ and $HAM-CYCLE \in P$. For each node, we know exactly two incident edges that participate in the cycle.

.

CHECK-HAM-CYCLE(G')
1. select a node $v \in V$
2. E_v is the edges that are incident to the node v
3. calculate the pair $e_1, e_2 \in E_v$ such that
4. $G' = (V, (E - E_v) \cup e_1, e_2)$ contains a Hamiltonian cycle.
5. If Hamiltonian Cycle exists then
6. return TRUE
7. else
8. return FALSE

. We also must design another algorithm GENERATE-HAM-CYCLE(G) that generates the order of a legitimate Hamiltonian cycle of G, if exists.

.

GENERATE-HAM-CYCLE(G):
1. for each $w \in V$
2. If CHECK-HAM-CYCLE(G) is TRUE then
3. print the Hamiltonian cycle of G

.

We can conclusively say by looking at the CHECK-HAM-CYCLE algorithm that the calculation of Hamiltonian cycle with all possible pairs requires polynomial time. So the algorithm is polynomial time algorithm.

.

Including the GENERATE-HAM-CYCLE algorithm which iterates with time complexity of $O(V)$ and implements the algorithm CHECK-HAM-CYCLE once per iteration.

.

Hence, the total running time complexity of the algorithm would be the product of V and the polynomial time of CHECK-HAM-CYCLE algorithm, which in result would still be a polynomial time where the degree of the polynomial is just raised by 1.

.

Therefore, if $HAM-CYCLE \in P$, then the problem of listing the vertices of a Hamiltonian cycle in order is solvable in polynomial time.

.

.

5. **The subgraph-isomorphism problem takes two undirected graphs G1 and G2, and it asks whether G1 is isomorphic to a subgraph of G2. Prove that the subgraph-isomorphism problem is NP-complete. (Hint: You can reduce the CLIQUE problem, i.e., whether a clique of size k exists in a given graph.)**

ANSWER:

Defining Graph Isomorphism: Two graphs G_1 and G_2 are isomorphic to each other if they have same number of edges and vertices with edge connectivity being retained. There is a bijection between the set of vertices of the graphs which shows that two vertices u, v are adjacent to each other if and only if $f(u), f(v)$ are adjacent to B where f is a bijection.

.

To prove that a problem is NP-Complete, we must show that it belongs to both NP and NP-Hard Classes since NP-Complete problems are NP-Hard problems which also belong to NP.

.

Proving if the problem belongs to NP class

First, we approach the problem of subgraph isomorphism to see that if it belongs to NP class, which requires the problem to have a polynomial time verifiability. Using a certificate, we should be able to verify that.

.

CERTIFICATE: Let G be a subgraph of G_2. Also, we know the mapping between the vertices of G_1 and G.

VERIFICATION: Foremost, we must see if G_1 is isomorphic to G or not.

- First we check if the mapping is a bijection.
- Then we verify if, for every edge (u, v) in G_1, there is an edge $f(u), f(v)$ in G, takes polynomial time.

.

Thus, the Subgraph Isomorphism problem has polynomial time verifiability and belongs to the NP class.

.

Proving if the problem belongs to NP-Hard class

Now, we focus on proving if the problem of subgraph isomorphism belongs to NP-hard class.

.

A problem P is in NP-hard class if every NP problem if reducible to P, in polynomial time.

.

In order to prove that the problem of Subgraph Isomorphism (S) is NP-hard, we try reducing an already known NP-hard problem to S for a particular situation. If the mentioned reduction is possible in polynomial time, the S is also an NP-hard problem.

.
Here we introduce the Clique Decision Problem (C), which can be reduced to the Subgraph Isomorphism problem. Clique Decision Problem (C) is an NP-complete problem, which indicates that all the problem in NP can be reduced to C in polynomial time.
.

- Let us start by defining the input of Clique Decision Problem which is (G, k).
- The output would be true if the graph G contains a clique of size k, where k is a subgraph of G.
- Let H be a complete graph of k vertices and G_2 be G, where G_1, G_2 are inputs to the Subgraph Isomorphism problem.
- We can see that $k \leq n$, where n is the number of vertices in $G(= G_2)$.
- If $k > n$, then a clique of size k cannot be a subgraph of G.
- The total time taken to generate G_1 is $O(k^2) = O(n^2)$ (since $k \leq n$), because the number of edges in a complete graph of size $k = C_2^k = (k * (k-1))/2$.
- The graph G has a clique of size k, if and only if H is a subgraph of G_2, since G_1 itself is a subgraph of G_2 and every graph is isomorphic to itself and hence the result of Subgraph Isomorphism Problem is true. Thus, G_1 is isomorphic to a subgraph of G_2.
- Hence, if the Clique Decision Problem is true, the Subgraph Isomorphism Problem is true and vice versa.
- Therefore, the Clique Decision Problem can be reduced to the Subgraph Isomorphism Problem in polynomial time for a particular instance.
- Thus, the Subgraph Isomorphism Problem is an NP-Hard class problem

.
.
Conclusively, we can say that Subgraph Isomorphism Problem is an NP class as well as an NP-Hard class problem, which in the end makes it an NP-complete problem.

-