Assignment 5: CSE548: Analysis of Algorithms

Shivasagar Boraiah — #112077826

December 6, 2018

Problem 1. Textbook [Kleinberg & Tardos] Chapter 7, page 415, problem #8.

Proof. ... (a)
Input: Supply chain with nodes s_o, d_A, s_B, s_{AB} and demand chain with nodes d_o, d_A, d_B, d_{AB}
Approach for (a): We can minimize the above supply-demand problem by constructing a graph similar to the problem we discussed in class, workers-machine problem and reduce it to the max flow problem.

- As a base case, we can construct a graph by connecting source node, a to all supply chain nodes, s (input set, s) and connect sink node, b to all demand chain nodes, t (input set d).
- First, construct an edge s, t if type t can retrieve the blood from s and set its capacity to the t.
- Second, Draw an edge a, s, and set the capacity to the availability of s.
- Third, construct an edge with sink-demand chain too and set the capacity accordingly

Now, on applying Ford-Fulkerson algorithm, the maximum flow to balance this supply-demand problem is

$$\sum_{i=1}^{4} demand - d_i$$

with E=9 edges in the graph matching the capacity.
Output: Give a polynomial-time algorithm to evaluate if the blood on hand would suffice for the projected need.
Correctness: With the help of supporting claim that, The need will be satisfied if and only if the edges from the demand nodes to the sink are all saturated in the resulting max-flow, the correctness can be achieved. We can observe that, if there is enough blood supply in which a_{sd} from the supply node s is used for type t, then we can certainly use supply of a_{sd} from the supply node s to demand node d and respect all capacity conditions. Converse, holds true as well.
Time Complexity: With the calculated max flow function, f as above and edgelist, E, the time complexity is, $O(fE)$, running polynomial time. □

Proof. ... (b)
Approach for (b): From the supply-demand distribution it is difficult not possible to satisfy 100 units of demand if we have 105 units on hand. To achieve this, its allocation should be found by running the max flow algorithm. With the values given in the table which are flow values, Using Min cut Max flow algorithm, the actual demand for A is 42 units whereas the real supply to A is 41 units. So, not all patients can receive the blood. Even the below configuration is possible satisfying the maximum flow capacity conditions. As the table shows, O blood type patient can be in deficit of 1 unit of blood as the demand for O blood type is 45. □

Problem 2. Textbook [Kleinberg & Tardos] Chapter 7, page 415, problem #9.

Proof. ...
Input: n patients and k hospitals.
Output: polynomial time to maintain the flow of patients to each hospital should not be more than n/k
Approach: This problem is similar to the one we discuss din the class where we need to construct flow network between patients, n and hospitals, k. Network graph is constructed between count n and k where a separate node for n_i for i hospital and k_j is constructed for j hospital. Also, there is a source node, s connected to n nodes with a capacity of 1 and a sink node, k which is a end point of k hospitals. There is a node n_i for each patient i, a node k_j for each hospital j, and an edge (n_i, k_j) of capacity 1 if patient i is with in a half hour drive of hospital j. On applying max-flow algorithm, the above graph is stabilized.
Correctness: The above graph can be proved that, every hospital has (n/k) patients with a claim that, in the entire graph there is a s-t flow of value n. If there is a feasible way to send patients, then sending one unit of flow from s to t along the path s, n_i, k_j, t, where patient i is sent to hospital j.
Complexity: To sove this max flow problem on a graph with $O(n + k)$ nodes and $O(nk)$ ediges is n. □

Problem 3. Textbook [Kleinberg & Tardos] Chapter 7, page 415, problem #17.

Proof. ...
Input: directed graph G = (V , E), in which the capacity of each edge is 1 and in which each node lies on at least one path from s to t.
Output: an algorithm that accomplishes the explained task using only O(k log n) pings.
Approach: Constructing a max flow network, f which has a value k and every edge has a capacity 1. This now tells us that f can be decomposed into k non-overlapping paths. Using Ford-Fulkerson algorithm, the above problem can be solved in a polynomial time $O(mk)$. Also, If the malicious attacker tries to destroy k arcs. There shall only be one arc destroyed on each of these paths the reason for it being that no two paths have an overlapping arc. And now that the malicious attacker has destroyed the arc there are no s - t paths left after removing these arcs. Each path A1,A2,A3,......,Ak has length O(n).
Correctness: Using Bisection Search Algorithm, we can check whether arc, A_i has been removed by a attacker. Where, we can ping the middle vertex A_i, If it is found ping the

vertex one quarter of the way along A_i. Otherwise, ping the vertex three-quarters of the way along A_i.
Complexity: As I have already stated, Using Ford-Fulkerson algorithm, the above problem can be solved in a polynomial time $O(mk)$. □

Problem 4. Textbook [Kleinberg & Tardos] Chapter 7, page 415, problem #18.

Proof. ... (a)
Input: Bipartite Graph, $G = (V, E)$
Output: Polynomial time solution to return of there exists any coverage expansion and either returns a solution M or M'.
Approach for (a): Using the input, Bipartitie graph, G=(V, E), first we should transform the problem into maximum flow network. As we did before, add source and sink nodes, s, t. Then, add directed edges from s to to each node of $V1$ and add directed edge form each node $V2$. Set the capacity of all the edges in the problem to $O(M)$. The overall transformation will happen in $O(E)$ time.

```
if k ! = 0;
  return FALSE;
if k == 0;
  return M;
f = ford-fulkerson(G');
if $|f| >= |M| + k$ then;
  pick k disjoint edges;
  M' = M U M;
  return M';
else;
  return FALSE;
```

Correctness: Let M and M' be the two cuts formed using Bi-partite graph. By performing Union operation, we can directly say that, every node y belongs to Y i.e covered M is also covered by M'. Using the claim, $|f| >= |M| + k$ iff there are at least k disjoint edges between V 1 and V 2 which are also disjoint to the edges in M. This can be acheived.
Complexity: Due to Ford-Fulkerson algorithm, the overall time complexity of the algorithm is $O(VE)$. □

Proof. ...(b)
Input: Bipartite Graph, $G = (V, E)$
Output: Retrieve an example for the situation, where the subset of M doesn't form a subset in M'.
Approach for (b): Using the above constructed Max flow network for a case where, $k = 1$, where V2 and V1 are covered but they dont have common edges between between M and M'. □

Proof. ... (c)
Input: Bipartite Graph, G
Output: Prove that in fact K1 = K2; that is, we can obtain a maximum matching even if were constrained to cover all the nodes covered by our initial matching M.

Approach: Let K1 be the largest size of subgraph problem, M. K2 be the largest matching possible for the graph G on which M is a matching. With that, we can see, even if were constrained to cover all the nodes covered by our initial matching M.
Correctness: The largest matching is exactly the size of the largest coverage expansion of any matching. This can be proved, Recall the network used to find a maximal matching of value K2. □

Problem 5. Textbook [Kleinberg & Tardos] Chapter 7, page 415, problem #22.

Proof. ... (a)
Input: mxn matrix with values either 0 or 1.
Output: mxn matrix that is non rearrangabale.
Matrix:

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$ □

Proof. ... (b)
Input: mxn matrix with values either 0 or 1.
Output: Give a polynomial-time algorithm that determines whether a matrix M with 0 1 entries is rearrangeable
Approach: This problem can be pictured as bipartite network flow where for every row and column, if $r_{ij} = 1$, then create a edge between r_i and c_j. With the above network flow, of there is a perfect flow then the matrix is rearrangable.
Correctness: To prove its correct, construct a matrix, M which has only 1's across diagonal. with $M_{ij} = 1$ implies that the graph has horizontal lines in its bipartite graph. So, if there is a perfect matching every vertex is incident to exactly one edge, which by swapping rows and columns we can achieve a graph which is exactly that of matrix which has only 1s on the diagonal. Also, bipartite matching problem is solvable using the network flow and ford-fulkerson alogorithm.
Complexity: As ther are n^2 edges the the complexity for max flow using bipartite graph, $O(n^3)$. □