AoA assignment 5

amitdharmadhikari41

December 2018

1 Question 1

1.1 (a)

We need to construct a graph G = (V,E). In the set of vertices, we will have a super-source node, four supply nodes (for each A, B, AB and O blood types), four demand nodes (similar to supply nodes - one for each blood type) and a super-sink node. We will construct an edge from the super-source node to each supply node with a capacity set to the available supply of blood of that type. Similarly, we will construct an edge between each demand node and the super-sink node, with a capacity set to the demand for that type. Between each supply node and demand node, we will construct an edge if the demand node can receive blood from the supply node and set its capacity to the demand for that blood type.

Now, we compute an integer-valued maxflow on this graph. The graph has constant size. Hence, the scaling maxflow algorithm takes O(log C) time, where C is the total supply. The Preflow-Push algorithm takes constant time.

There is sufficient supply for the projected need if and only if the edges from the demand node to the sink are all saturated in the maxflow. If there is sufficient supply in which x_{ST} units are sent from the supply node of type S to the demand node for type T, and respect all edge capacities. Conversely, if there is a flow which saturates all edges between demand nodes and the super-sink node, then there is an integer flow with this property. If it sends f_{ST} units of blood of type S from the supply node of type S to the demand node for type T, then we can use f_{ST} units of blood of type S for patients of type T.

1.2 (b)

For this subquestion, we will consider a cut containing the source, supply and demand nodes for B and A. The capacity of this cut is 50 + 36 + 10 = 99, and hence all 100 units of demand cannot be satisfied.

For clinic administrators:

There are 87 people with demand for blood types O and A. This demand can only be satisfied with blood donors of types O and A, and there are only 86 such donors.

2 Question 2

We need to build a flow network to solve this problem. We will make a node p_i for each patient i, a node h_j for each hospital j, and an edge (p_i, h_i) of capacity 1 if patient i is within a half hour drive of hospital j. We then make a super-source node s and connect it to each of the patient nodes by an edge of capacity 1. Similarly, we make a super-sink node t and connect each of the hospital nodes to it by an edge of capacity $\lceil n/k \rceil$.

There is a feasible way to send all patients to hospitals if and only if there is an s-t flow of value n. If there is a feasible way to send a patient i to hospital j, then we send one unit of flow from s to t along the edges (s, p_i), (p_i, h_i) and (h_i, t). This does not violate the capacity conditions as we have put load constraints on the edges connecting the hospital node to the super-sink node. Conversely, if there is a flow of value n, then there is one with integer values. We send patient i to hospital j if the edge (p_i, h_i) carries one unit of flow, and we the capacity condition makes sure that none of the hospitals is overloaded. The runtime of this algorithm is the time required to solve a maxflow problem on a graph with O(n+k) nodes and O(nk) edges.

3 Question 3

We are restricted to at most O(k log n) pings. However, the actual runtime of the algorithm is not restricted. This problem can be solved using the Ford-Fulkerson algorithm taught in class.

The maximum flow f* has value k and every arc has capacity 1. Hence, f* can be decomposed into exactly k arc-disjoint paths P_1 , P_2 , ... , P_k. We can find these paths in time O(mk) using the Ford-Fulkerson algorithm.

The attacker has destroyed exactly k arcs. No two paths share an arc, and there are no s-t paths left after removing these arcs. Hence, there is exactly one arc destroyed on each of these paths. Each path P_1 , P_2 , ... , P_k has length O(n) so in order to find which arc on P_i was removed we use a bisection-search ping algorithm on the path. That is, ping the middle vertex on P_i . If that is found ping the vertex one quarter of the way along P_i; otherwise ping the vertex three-quarters of the way along P_i, etc. So we use O(log n) pings per path as required to detect all the removed edges, i.e. between the last vertex on the path that we can ping and the first vertex we can't ping must be the removed edge. Lets call the removed edges E*. Since we know what the graph looks like originally, we perform a BFS in O(m) on $G - E*$ to find all the vertices that are reachable in the network. And return those that aren't.

4 Question 4

4.1 (a)

We can solve this problem by converting the coverage expansion problem into a a maximum flow problem. Given a bipartite graph G = (V_1 U V_2, E), matching M and integer k, we use the following algorithm.

Add source and sink nodes s, t.
Add directed edges from s to each node in V_1.
Add directed edges from each node in V_2 to t.
Add edges from V_1 to V_2.
Set the capacity of each edge in matching M to 0.
Set the capacity of all the other edges to 1.

```
Coverage$_expansion_solver(G, M, k) : if k < 0 then :$
    $return false$
$if k == 0 then :$
    $return M$
$G' = transform G into a network flow G' as shown above$
$f = Ford - Fulkerson(G')$
$if |f| \geq$ —M— + k then:
    M' = pick disjoint edges, where each edge has the form e = (v,u) such
that $v \epsilon V_1, u \epsilon V_2$
    M' = M U M'
    return M'
else:
    return false
```

4.2 (b)

Consider a graph with 4 vertices and edges (v1,v2), (v2,v3) and (v3,v4). It is a bipartite graph whose nodes are partitioned into A = v1, v3 and B = v2, v4. Let M = (v3, v2) be the matching. Assume k=1. Let this be an instance of the coverage expansion problem.
—M— = 1
—M'— = —M— + k = 2
Let M' = (v1,v2), (v3,v4) M' has more edges than M and every node covered in M is aldo covered in M'. Apart from that, the edges in M are not a subset of the edges in M'.

5 Question 5

This is an instance of the Assignment problem.