Aaryan Ajay Sharma 2022121001

Algorithm Analysis & Design

15/11/2022

# Problem set 4

Q.1 False.

Consider a graph with nodes $s, v_1, v_2, v_3, w, t$ edges $(s, v_i)$ & $(v_i, w)$ for each $i$, and an edge $(w, t)$.

There is a capacity of 4 on edge $(w, t)$, and a capacity of 1 on all other edges.

Then setting $A = \{s\}$ and $B = V - A$ gives a minimum cut, with capacity 3.

But if we add one to every edge then this cut has capacity 6, ~~and~~ more than the capacity of 5 on the cut with $B = \{t\}$ and $A = V - B$.

Q.2 We build the following flow network.

There is a node $v_i$, for each patient $i$, a node $w_j$ for each hospital $j$, and an edge $(v_i, w_j)$ of capacity 1 if patient $i$ is within a half hour drive of hospital $j$.

We then connect a super-source $s$ to each of the patient nodes by an edge of capacity 1, and we connect each of the hospital nodes to a super-sink $t$ by an edge of capacity $\lceil n/k \rceil$.

We claim that 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 ~~all~~ patients, ~~to hospital~~ then we send one unit of flow from s to t along each of the paths s, $v_i$, $w_j$, t where patients i is sent hospital j.

This does not violate the capacity constraints, in particular on the edge ($w_j$, t) due to the load constraints.

Conversely, if there is a flow of value of n, then there is one with integer values.

We send patients i to hospital j if the edge ($v_i$, $w_j$) carries one unit of flow, and we observe that the capacity condition ensures that no hospital is overloaded.

The running time is the time required to solve a max-flow problem on a graph with $O(n+K)$ vertices & $O(nK)$ edges.

Q.3 We will assume that the flow f is integer-valued.

Let $e^* = (v, w)$. If the edge $e^*$ is not saturated with flow, then reducing its capacity by one unit does not cause a problem.

So assume it is saturated

We first reduce the flow on $e^*$ to satisfy the capacity conditions. We now have to restore the capacity constraints.

We construct a path from $u$ to $t$ such that all edges carry flow, and we reduce the flow by one unit on each of these edges. We then do the same thing from $v$ back to $s$.

Note that because the flow $f$ is acyclic, we do not encounter any edge twice in this process, so all edges we traverse have their ~~process~~ flow reduced by exactly one unit, and the capacity constraint is restored.

Let $f'$ be the current flow. We have to decide whether $f'$ is a maximum flow, or whether the flow value can be increased. Since $f$ was a maximum flow, and the value of $f'$ is only one unit less than $f$, we attempt to find a single augmenting path from $s$ to $t$ in the residual graph $G_{f'}$. If we fail to find one, then $f'$ is maximum. Else the flow is augmented to have a value at least that of $f$; since the current flow network cannot have larger maximum flow value than the original one, this a maximum flow

_/_/_

Q.4 First observe that by removing any $k$ edges in a graph, we reduce the capacity of any cut by at most $k$, and so, the min-cut will reduce by at most $k$.

Therefore, the max-flow will reduce by at most $K$. Now we show that one can in fact reduce the max-flow by $K$.

To achieve this, we ~~make~~ take a min-cut $X$ and remove $K$ edges going out of it.

The capacity of this cut will now become $f-K$, where $f$ is the value of the max-flow.

Therefore, the min-cut becomes $f-K$, and so, the max-flow becomes $f-K$.

Q.5 First compute a minimum $s-t$ cut $C$, and define its value by $|C|$. Lets $e_1, e_2, e_3, \ldots e_K$ be the edges in $C$.

For each $e_i$, try increasing the capacity of $e_i$ by 1 and compute a minimum cut in the new graph. Let the new minimum cut be $C_i$, and denote its value me (in the new graph).

If $|C|' = |C_i|$ for some $i$, then clearly $C_i$ is also a minimum cut in the original

_/_/_

graph and $C \neq C_i$, so the minimum cut is not unique. Conversely, if there is a different minimum cut $C'$ in the original graph, there will be some $e_i \in C$ that is not in $C'$, so increasing the capacity of that edge will not change the volume of $C'$, so increasing the capacity of that edge will not change the volume of $C'$, thus $|C| = |C_i|$.

In conclusion, the graph has a unique minimum cut if and only if $|C| < |C_i|$ $\forall i$.

The algorithm takes at most $m+1$ computing of minimum cuts, and therefore runs in polynomial time.

Q. 6

We have $G = (V_1, V_2, E)$ $(|V_1| = |V_2| = n)$ a ~~bar~~ bipartite matching.

Also all $(v_1, v_2) \in E$ have capacity $= 1$.

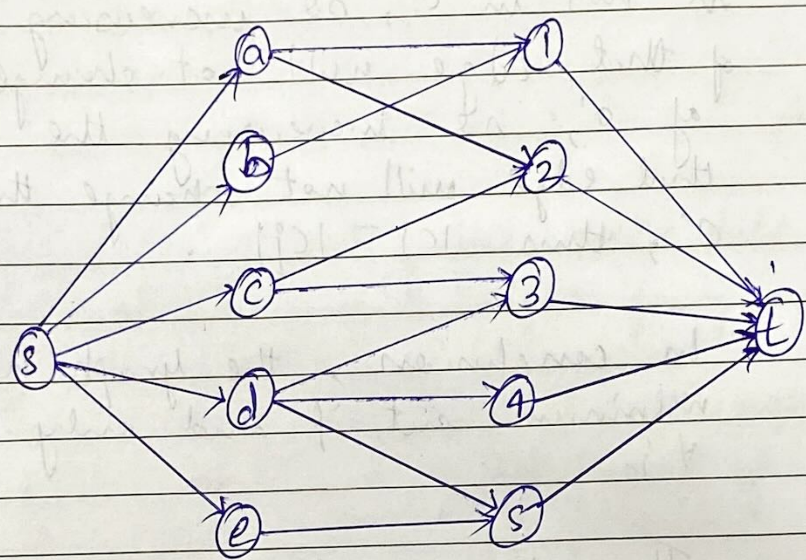
Now, since the capacities are integers our flow will also be an integer.

Also since capacity $= 1$. We'll either use the edge completely or not at all.

Let $M$ be the set of edges going from $V_1$ to $V_2$.

→ We'll show:

1. M is a matching
2. M is largest possible matching




We can choose one edge leaving any node in $V_1$ and one edge leaving any node in $V_1$ and one edge entering any node $V_2$

If we chose more than 1, we couldn't have balanced flow.

Now, if there's matching of k edges, there is a flow f of value k and since there's flow f of value k, there's matching with k edges.

Suppose we find the maximum flow f (with k edges) ⇒
This corresponds to matching M of k edges.

_/_/_

If there were a matching with $k >$ edges, we would have found a flow with values $> k$, contradicting that $f$ was maximum.

Hence $k$ gives maximum flow.

Now, since we know that $G$ is a perfect matching all the edges of $V_1$ & $V_2$ are covered upon matching.

So, we can say that $k$ will have value equal to that of no. of vertical in $V_1$ or $V_2 = n$.

Otherwise if that would not have been the case, there will be atleast one vertex $V_1$ and one in $V_2$ left contradicting the max flow condition.

$\therefore k = n$

Max-flow gives perfect matching.

**In Discussion With :**
**Ishan Kavathekar (2022121003)**
**Faizal Karim (2022121004)**
**Kunal Bhosikar (2022121005)**
**Yashpal Yadav (2022121007)**
**Prakhar Jain (202212008)**