Aashish Yadavally
Spring 2021

Homework 4

Question 1

It is given that $(u, v) \notin E$ and $(v, u) \notin E$.

From the definition of flow, each edge has a non-negative capacity, and for any edge not in E, the capacity is equal to 0. Hence, $c(u, v) = 0$.

Let us assume that $f(u, v) \geq 0$.

Then, $0 \leq f(u, v) \leq c(u, v) = 0$. Therefore, $f(u, v) = 0$.

From the skew-symmetric condition, we know that for all $u, v \in V$, $f(u, v) = -f(v, u)$.

Since $f(u, v) = 0$, therefore, $f(v, u) = 0$. Hence proved.

Question 2

Given that there exist two distinct maximal flows f_1 and f_2.

We know that a flow has maximum value if and only if it has no augmenting paths. Thus, both f_1 and f_2 have no augmenting paths.

Based on this criterion, for a given constant c such that $0 < c < 1$, any linear combination of f_1 and f_2, i.e. $c.f_1 + (1 - c).f_2$ also has no augmenting paths. Thus for different values of c, there exist infinitely many maximum flows.

Question 3

This problem could be formulated as a max flow problem in the following way:

- Let the professor's house be the source node and the school be the sink node.
- Let all the other corners that his children can visit be denoted by c_i. If there is a path from corner c_i to corner c_j, let $f(c_i, c_j) = 1$, i.e. each edge has a capacity one.
- Let the nodes connected to the corners or nodes interconnected have a weight greater than 1. This ensures that the minimum weight for each edge is 1.

Thus, from the source to the sink, if there exists a maximum flow whose value is 2, then we can conclude that there must exist a non-contradictory path which both of the children could take to attend the same school.

Question 4

Given that M is the maximal matching of the graph G. Let it's size be denoted by $|M|$. Consider an edge (u, v) in a matching of the graph M'. For any such edge (u, v), atleast one of the vertices u or v must be present in the maximal matching M. Otherwise, the edge can be added to M and it will no longer be maximal. Since an edge can cover atmost 2 vertices, the number of vertices covered by a maximal matching M must be greater than or equal to $\frac{|M'|}{2}$, i.e. $|M'| \leq 2|M|$.

Question 5

The maximum flow for the network $G = (V, E)$ can be solved using the Ford-Fulkerson Algorithm. Let H_f be the subgraph from the source node to the sink node induced by arcs with positive flow value.

Let m be the minimum arc flow value on the path. There must exist an arc (u, v) on the path such that it's flow is equal to the minimum capacity of the arc on the path, i.e. $f(u, v) = c(u, v)$. Let us assume that $f_1(x, y) = f(x, y) - m$ for every arc (x, y) on the path. Also, for all the arcs not on the path, let $f_1(x, y) = f(x, y)$. Thus, H_{f1} is a subgraph of H_f with atleast one less arc (u, v). This process can be repeated until there are no more such arcs left, resulting in $|E|$ subgraphs, and consequently $|E|$ paths. As the flow on each of the $|E|$ paths can be obtained from one augmentation on the path, the maximum flow can be obtained by a sequence of at most $|E|$ augmenting paths. Hence proved.

Question 6

The bipartite graph can be converted into a flow problem as follows: add a source vertex s and a sink vertex t. Connect s to every vertex $u \in L$ with an arc (s, u). Also connect every vertex $v \in R$ to the sink vertex t with arc (v, t). Convert every edge (u, v) for $u \in L$ and $v \in R$ to an arc (u, v) with unit capacity.

Let us consider a cut (S, T) such that $s \in S$ and $t \in T$. Let us denote $L \cap S$ by L_s and $R \cap S$ by R_s. There are atleast $|L_s| - |R_s|$ number of edges going from L_s to $R \cap T$. Based on the set of all neighbors of L_s, we can deduce that there must be atleast the same number of neighbors in R. If this is not the case, some of the vertices in R would need to have a degree higher than d. Since there are only $|R_s|$ number of neighbors of L_s which are also in S, the size of the set of neighbors of L_s that are in T is atleast $|L_s| - |R_s|$. Since each of these neighbors has an edge crossing the cut, we have that the total number of edges taht the cut breaks is atleast $(|L| - |L \cap S|) + (|L \cap S| - |R \cap S|) + (|R \cap S|) = |L|$. Thus, we can conclude that every cut has a value of atleast $|L|$. This is sufficient to prove that the maximum flow of the constructed flow network is $|L|$. Hence, every d-regular bipartite graph has a matching of size $|L|$.

Question 7

Given that $U = \{u_1, u_2, ..., u_n\}$ and $V = \{v_1, v_2, ..., v_n\}$.

A given bipartite graph $G = (U, V, E)$ is *convex* if $(u_i, v_k), (u_j, v_k) \in E$ with $i < j$ implies that $(u_h, v_k) \in E$ for all $h = i, i+1, ..., j$.

The idea behind generating a maximum watching in a convex bipartite graph is as follows: suppose each v_k is connected to u_h for $s_k \leq h \leq d_k$, sort all v_k in ordering $d_1 \leq d_2 \leq ... \leq d_n$. Connect all v_k into a list L based on the above ordering. This will generate the maximum watching in the convex bipartite graph. The pseudo-code for this greedy algorithm is as follows:

```
function MAX-MATCHING
    M := 0
    for i = 1 to n do begin
        from list $L$, find the first $v_k$ such that $s_k \leq i \leq d_k$;
        if such a $v_k$ exists
            M := M $\cup$ $(u_i, v_k)$
            Delete $v_k$ from list $L$;
    end-for
    return M
end-function
```

Correctness of Algorithm:

Let $M*$ be a maximum matching. Suppose $s_u = min_{1 \leq i \leq n} s_i$. Let the vertex corresponding to s_u in U be u_{s_*}. Let k_u, $1 \leq k_u \leq n$, satisfy $d_{k_*} = min\{d_k | s_k = s_u, 1 \leq k \leq n\}$.

If the edge $(u_{s_*}, v_{k_*}) \in M*$, then through mathematical induction, it can be extended to the bipartite subgraph H formed by the vertex subsets $(U - \{u_{s_*}\}, V - \{v_{k_*}\})$ and the edge set $E - (u_{s_*}, v_{k_*})$. Thus, for the graph G, if (u_{s_*}, v_{k_*}) is the first edge chosen by the algorithm, it results in subsequently running the algorithm for the bipartite subgraph H. $M* - (u_{s_*}, v_{k_*})$ has the same size of matching obtained by the algorithm. Therefore, the matching obtained by the algorithm is $|M*|$.

If the edge $(u_{s_*}, v_{k_*}) \notin M*$, we have two cases:

- $M*$ contains two edges (u_{s_*}, v_k) and (u_i, v_{k_*}). Since $s_k = s_*$ and $d_k \geq d_{k_*}$, $s_k \leq i \leq d_k$ and the edge (u_i, v_k) exists. Thus, by replacing (u_{s_*}, v_k) and (u_i, v_{k_*}) by u_{s_*}, v_{k_*} and (u_i, v_k) we obtain from $M*$ a maximum matching of size $|M*|$ containing the edge (u_i, v_k).
- $M*$ contains only one edge to one of u_{s_*} or v_k. In $M*$, if we replace this edge by (u_i, v_{k_*}) as described above, we get a maximum matching of size $|M*|$ containing (u_i, v_{k_*}).

Thus, in each of the cases, the algorithm generates a maximum matching of size $|M*|$. Hence the algorithm is correct.

Question 8

This is equivalent to solving a matching problem where the size of the maximal matching is given by a non-negative integer function $c : V \to N$. For a given graph $G(V, E)$, a subgraph H is one in which $H = (V, F)$ where $F \subseteq E$. Furthermore, a node in H is considered to be free if it's degree is less than it's value for the non-negative integer function, i.e. $deg(v) < c(v)$ for some vertex v in H.

Let us assume that M is the matching for H. For all the free and non-free nodes in the subgraph H, alternating paths can be defined such that it's edges belong alternately to the matching M, when an edge is considered between two free nodes, and not belonging to the matching M, when an edge is considered between a free node and a non-free node. Any edge between two non-free nodes can not be added to the existing matching as the condition on the degree for both the non-free nodes will be defied. This covers all the possible edges between the vertices, thus proving that any additional edge cannot be added to the existing matching. Thus, the matching is maximal, and the subgraph H so computed has a maximum number of edges. Hence proved.