COL351 Assignment 4

Avval Amil - 2020CS10330
Ujjwal Mehta - 2020CS10401

November 2022

1 Hitting Set

Part 1

To prove that the Hitting Set problem is in NP class, we give an algorithm which when given an instance of the decision version of the problem, verifies whether the given certificate is a valid hitting set or not.
Specifically, given a set $U = \{u_1, u_2 \dots u_n\}$, a collection of subsets $A_1, A_2 \dots A_m$, a positive integer k and a given subset of U called C (which has to be verified), we have to check whether $(C \cap A_i \neq \phi)\ \forall i$ or not.

Algorithm and Time Complexity

For each element $e \in C$ and each subset A_i of U, check if $e \in A_i$ or not. Notice that we can check whether an element belongs in a set or not by repeatedly comparing it with all the elements in the set and returning true if there is an equality.
Doing this for a single set and element pair will take time proportional to the size of the set. But as the sets here are subsets of U, the maximum size of the sets will be equal to that of U, which is n. Also, as there are m subsets, checking for a single element of C will take $\mathcal{O}(mn)$ time. As we have to check for all elements in C, which can be of maximum size n, we will have to spend at most $\mathcal{O}(mn^2)$ time. Since this time is polynomial, we have shown that the Hitting Set problem is in NP.

Part 2

To show that the Hitting Set problem is NP-complete, we will reduce the decision version of Vertex Cover to the decision version of Hitting Set. We prove the two directions now -

Claim 1 (Forward Direction)

For every instance of the vertex cover problem (G, k) where G is a graph in which there is a vertex cover of size at most k, there is an instance of problem $(U, A_1, A_2 \dots A_m, k)$ such that there is a set $C \subseteq U$ which is a hitting set of size at most k.

Proof

Let $G = (V, E)$ be the graph such that $V = \{v_1, v_2 \dots v_n\}$ is the vertex set and $E = \{e_1, e_2 \dots e_m\}$ is the edge set. Now, construct the set $U = \{u_1, u_2 \dots u_n\}$ such that each element $u_i \in U$ corresponds to the vertex $v_i \in V$. Also, construct $A_i = \{x, y\}$ where $x \in U$ and $y \in U$ such that it corresponds to the edge e_i and the elements x and y correspond to the end-points of e_i.
Let VC be a vertex cover of size at most k of G. Construct C such that for every $v \in VC$, we take its corresponding element $u \in U$ and add it to the set

C. The set C so constructed is claimed to be a hitting set.
This can be proved by contradiction. Assume that C is not a hitting set. Then there exists some A_i for which $A_i \cap C = \phi$. This means that if $A_i = \{x, y\}$ then neither $x \in C$ nor $y \in C$. But, as x and y were the two endpoints corresponding to the edge $e_i \in E$, and the vertex cover VC must have contained at least one of them, it is not possible that C does not contain both of them. Hence, we have proved that C is a hitting set. As the size of C is the same as the size of VC, we can say that a hitting set of size at most k has been constructed.
We can also see that this transformation is of polynomial time in n and m, as we are constructing objects to the vertex and edge sets of the graph with exactly the same size.

Claim 2 (Backward Direction)

For every instance of the vertex cover problem (G, k) where G is a graph and the corresponding instance of problem $(U, A_1, A_2 \ldots A_m, k)$ such that there is a set $C \subseteq U$ which is a hitting set of size at most k, there is a vertex cover of G of size at most k.

Proof

We know that by definition, the hitting set of size at most k (let it be called C), has non-empty intersection with all the subsets A_i. Also, for every $u \in C$, we can construct a set VC such that if u corresponds to the vertex $v \in V$ (vertex set of G), then $v \in VC$. This correspondence is as described in the proof of Claim 1. We can claim that VC so constructed is a vertex cover of the graph G. This can again be proved by contradiction. Let VC not be a vertex cover of G. Then there exists an edge $e_i = \{v, w\}$ such that neither v nor w are in VC. But, the subset A_i corresponding to the edge e_i must have had elements corresponding to v and w, and by the definition of the hitting set C must have contained at least one of these elements otherwise the intersection would be empty. This would cause the set VC to contain either v or w, as their corresponding elements belonged to C, which is a contradiction. Again, as the size of VC is the same as the size of C, we can say that we have constructed a vertex cover of size at most k.

The two claims proved above prove that the Hitting Set problem is NP-complete, as it is in NP class and another NP-complete problem (Vertex Cover) can be reduced to it in polynomial time.

2 Tracking Shortest Paths

Part 1

To show that the tracking shortest paths problem lies in the NP class, we prove a property of tracking sets as a whole, as the number of shortest paths can be exponential in the worst case between the source vertex s and the target vertex t, so we need an alternate way of verifying a given certificate rather than just using brute force. To do this, we first prove a few claims -

Claim 1

The shortest path lengths between all pairs of vertices in an undirected, unweighted graph can be computed in polynomial time.

Proof

By applying BFS from every vertex, we can compute the shortest paths from each vertex to every other vertex. This will take $\mathcal{O}(n(m+n))$ time, which is polynomial. There are more efficient ways possible, but the main takeaway is that this is possible in polynomial time.

Claim 2

A pair of vertices u and v distinct from both s and t lie on a shortest path between s and t (we can assume without loss of generality that u appears before v in the path) iff $d(s,u)+d(u,v)+d(v,t)=d(s,t)$. Here, $d(a,b)$ represents the shortest path length between vertices a and b. All such pairs can be computed in polynomial time.

Proof

First, let's assume that u and v (in order), do appear on some $s-t$ shortest path. Then, as all subpaths of this path are going to be shortest paths (otherwise we could replace that respective subpath by a shorter path and get a shorter path - which is a contradiction), we can say that the subpaths $s-u, u-v$ and $v-t$ are shortest paths between their respective endpoints as well. On adding these shortest path lengths, we get that $d(s,u)+d(u,v)+d(v,t)=d(s,t)$.
For proving the converse , we assume that $d(s,u)+d(u,v)+d(v,t)=d(s,t)$. Now, we just concatenate the paths which have lengths $d(s,u), d(u,v)$ and $d(v,t)$ to get a path between s and t of length $d(s,u)+d(u,v)+d(v,t)$. As this length is the same is the $s-t$ shortest path length $d(s,t)$, this is another shortest path between the source and target, and u and v lie in this path. Hence, proved.
To prove that all such pairs can be computed in polynomial time, we simply compute all pairs shortest path (using Claim 1 in polynomial time), and then check whether $d(s,u)+d(u,v)+d(v,t)=d(s,t)$ or not. If yes, then the pair occurs in some shortest path, otherwise it doesn't.

Claim 3

The number of shortest paths between two vertices u and v in a graph $G = (V, E)$ can be found out in polynomial time.

Proof

First, compute the shortest path length between u and v using BFS. Let the adjacency matrix of G be A, and the shortest path length found be k. Then, in the matrix $B = A^k$, the entry $B[u][v]$ will be the number of paths between u and v which have length exactly k (in other words, the number of shortest paths). This can be computed in polynomial time (done in class), and the proof of correctness follows by the proof of the fact that in $B = A^i$, the entry $B[x][y]$ represents the number of paths between x and y of length i (done in class). Hence, proved.

Claim 4

A set T is a tracking set iff for every pair of vertices u and v which lie on an $s-t$ shortest path (in order), in the graph $G' = (G \backslash (T \cup \{s,t\})) \cup \{u,v\}$ (here, removing a set of vertices means removing their associated edges too, and adding vertices back means adding their edges back which are possible; also note that G' is dependent on the choice of u and v), there does not exist more than one shortest path between u and v with length equal to $d(u,v)$ ($d(a,b)$ represents shortest path length in the original graph; note that shortest path in G' cannot be less than shortest path in G as G' is a subgraph of G).

Proof

First, let's assume that T is a tracking set. For sake of contradiction, assume that there exist at least two shortest paths between u and v in G' with length equal to $d(u,v)$. Let two of these be P_1 and P_2. Also, in the original graph G, let P_u be a shortest $s-u$ path, and P_v be a shortest $v-t$ path. Then, in the original graph G, the two paths $P_a = P_u + P_1 + P_v$ and $P_b = P_u + P_2 + P_v$ represent two shortest paths between s and t (note that '+' here refers to path concatenation, also note that a path that exists in G' also exists in G as G' is a subgraph of G). Also, we can note that $V(P_a) \cap T = V(P_b) \cap T$. This is because P_a and P_b only differ in their respective portions of P_1 and P_2, but those portions do not contain any vertex in T (as otherwise the path would not exist in G'). But this is a contradiction as T was a tracking set.
For the converse, we can assume that T is not a tracking set and then show that there exists at least two paths of length $d(u,v)$ in G' between some pair of vertices u and v which were on some $s-t$ shortest path in G. As T is not a tracking set, then by definition there exists two paths P_a and P_b in G such that $V(P_a) \cap T = V(P_b) \cap T$. Let the vertices in $T \cup \{s,t\}$ which are encountered in the path P_a in order be $(s = v_0, v_1, v_2 \dots v_k, v_{k+1} = t)$. This will also be the order in which they are encountered in P_b as we know that $V(P_a) \cap T = V(P_b) \cap T$

so all of these vertices also belong to P_b, and the ordering can't change as then the path would no longer be the shortest path (contradiction). Let $P_a(v_i, v_j)$ and $P_b(v_i, v_j)$ denote the subpaths between the vertices v_i and v_j in the paths P_a and P_b respectively. We choose i such that it is the smallest value for which $P_a(v_i, v_{i+1}) \neq P_b(v_i, v_{i+1})$. Note that such an index will necessarily exist otherwise the two paths P_a and P_b would be identical. Note that except for the end-points v_i and v_{i+1}, all the vertices in $P_a(v_i, v_{i+1})$ and $P_b(v_i, v_{i+1})$ are not in T. This implies that all the vertices of $P_a(v_i, v_{i+1})$ and $P_b(v_i, v_{i+1})$ are in $G' = (G \backslash (T \cup \{s,t\})) \cup \{v_i, v_{i+1}\}$. Also, as the lengths of both $P_a(v_i, v_{i+1})$ and $P_b(v_i, v_{i+1})$ are equal to $d(v_i, v_{i_1})$ (as they were part of some $s-t$ shortest path in G), we can say that there exists one pair of vertices which has at least two shortest paths in G' of length equal to the shortest length in the original graph. Hence, proved.

Now, we give an algorithm to verify whether a given set T is a tracking set of size at most k or not for a graph $G = (V, E)$ (decision version (G, k) of the tracking shortest paths problem).

Algorithm 1: Verification of Tracking Set

```
$T$ is a given set (to be verified), and $G$ and $k$ are given.
if ($|T| > k$) then
    Return False
end
$X$ = Set of pairs of vertices which are in some $s - t$ shortest path in $G$
for each pair of vertices $u$ and $v$ in $X$ do
    $d$ = shortest distance between $u$ and $v$ in $G$ (by BFS)
    $G' = (G \backslash (T \cup \{s,t\})) \cup \{u, v\}$
    $d'$ = shortest distance between $u$ and $v$ in $G'$ (by BFS)
    if ($d' = d$) then
        $num$ = Number of shortests paths between $u$ and $v$ in $G'$
        if ($num > 1$) then
            Return False
        end
    end
end
Return True
```

Proof of Correctness

By Claim 4, we can see that a set is a tracking set iff for every pair of vertices u and v which lie on an $s-t$ shortest path (in order), in the graph $G' = (G \backslash (T \cup \{s,t\})) \cup \{u, v\}$, there does not exist more than one shortest path between u and v with length equal to $d(u, v)$. This is precisely what the algorithm is checking for. Further, the algorithm also makes sure that the size of T is at most k, and hence returns the correct answer. Hence, proved.

Time Complexity Analysis

The set X can be computed in polynomial time (by Claim 1 we can compute all pair shortest distances and then check for the equality condition in Claim 2 for each pair of vertices). The size of X is also polynomial in n, and hence the number of iterations of the for loop is also polynomial in the input. The shortest distances and the subgraph can be computed in polynomial time as well (as BFS is polynomial and the subgraph construction only involves deleting and adding edges and vertices). By Claim 3, the number of shortest paths can also be found out in polynomial time.
As all the steps take polynomial time, and the number of iterations is also polynomial in the input, we can conclude that the overall verification algorithm takes polynomial time in the input size.
Hence, the tracking shortest paths problem is in NP class.

Part 2

We will prove that the tracking shortest paths problem is NP-complete by reducing the vertex cover decision problem to the tracking shortest paths decision problem in polynomial time. In the following analysis, we will assume that the graph does not have any isolated vertices (this is because isolated vertices can be handled in polynomial time as they will always be part of a vertex cover and so they do not add to the hardness of the vertex cover problem). We prove both the directions now -

Claim 1 (Forward Direction)

For every instance of the vertex cover problem (G, k) where $G = (V_g, E_g)$ ($|V_g| = n$ and $|E_g| = m$) is a graph in which there is a vertex cover of size at most k, there is an instance of problem $(G', m + k)$ such that there is a tracking set of G' of size at most $m + k$.

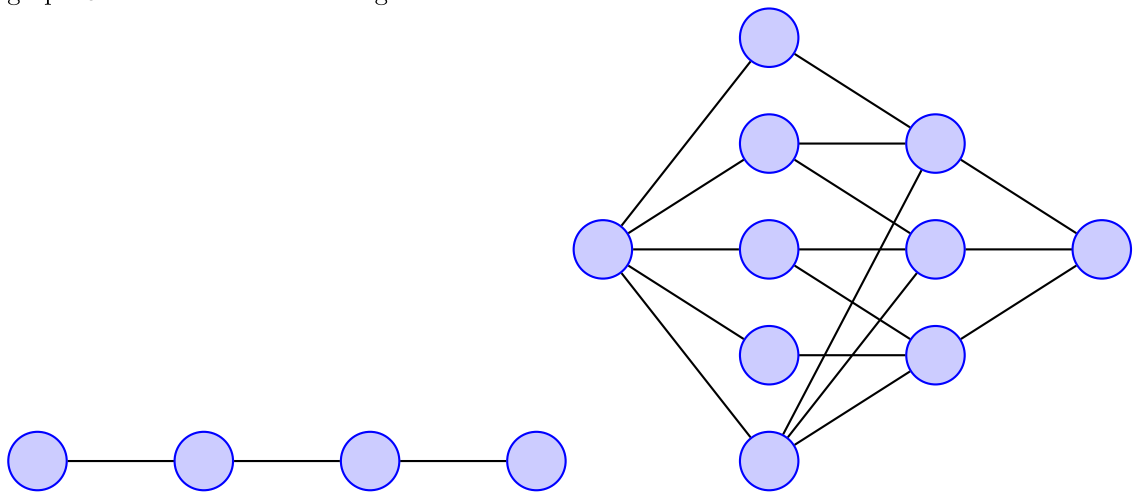
Proof

First, we construct a graph $G' = (V, E)$ in which we will create an instance of the tracking shortest paths problem. Let the graph for which the vertex cover problem is formulated be $G = (V_g, E_g)$. For each $v_g \in V_g$, create a vertex $v \in V$. Also, for each edge $e_g \in E_g$, create a vertex $e \in V$. Apart from this we have another vertex $a \in V$, and we have two vertices s and t which will be the source and target for the tracking set problem in G' respectively. The edges in G' will be as follows -

- $(s, v) \in E$ for all vertices v corresponding to some $v_g \in V_g$
- $(s, a) \in E$

- $(e, t) \in E$ for all vertices e corresponding to some $e_g \in E_g$
- $(v, e) \in E$ for all vertices e and v corresponding to $e_g \in E_g$ and $v_g \in V_g$ respectively such that v_g is an endpoint of e_g
- $(a, e) \in E$ for all vertices e

For example, for the simple graph G given below to the left, the constructed graph G' is also shown in the right.



In the left, the graph G has 4 vertices and 3 edges. In the right graph G', the first vertex denotes s, while the 5 vertices in the next layer correspond to the 4 vertices in the graph G and the bottommost vertex which is a. The vertices in the next layer correspond to the 3 edges of G (notice the connections), and finally the last vertex in the right graph is t. This was just an example for illustration of our construction, now we continue with our proof.

Let the vertex cover of G of size at most k be represented by VC. Now, for the tracking set of G', we consider the following vertices -

- $v \in V$ such that v corresponds to $v_g \in V_g$ and $v_g \in VC$
- All except any one $e \in V$ corresponding to $e_g \in E_g$
- $a \in V$

We can see that the size of this set (let it be called T) is $|VC| + (m - 1) + 1$ so if the size of the vertex cover was at most k, then the size of this set will be at most $m + k$. Now, we need to prove that T is a tracking set of G'.

For this, first notice that any $s-t$ shortest path in G' is of length 3, and is of either the form $s-v-e-t$ or $s-a-e-t$, where e, v are vertices of G' corresponding to some $e_g \in E_g$ and $v_g \in V_g$, and a is the additional vertex added by construction. By breaking into cases, we examine all possible pairs of shortest paths P_1 and P_2 from s to t, and show that their intersection with the set T is different.

1. $P_1 = (s - v_1 - e_1 - t)$ and $P_2 = (s - v_2 - e_2 - t)$, where $e_1 \neq e_2$ (v_1 and v_2 need not be distinct, and can also be equal to a)

   In this case, note that at least one of e_1 and e_2 has to be present in T because T is missing exactly one $e_i \in V$. As both of them cannot be absent from T, the sets $V(P_1) \cap T$ and $V(P_2) \cap T$ are necessarily distinct (say that e_1 was present in T, then the first set has e_1 while the second set does not).

2. $P_1 = (s - v_1 - e - t)$ and $P_2 = (s - v_2 - e - t)$, where $v_1 \neq v_2$ and neither of them equal a.

   In this, case we can note that at least one of v_1 or v_2 must belong to T. This is because all the vertices corresponding to $v_g \in VC$ were part of T, and as the vertices corresponding to v_1 and v_2 are the endpoints of the edge corresponding to e, at least one of them must be T otherwise it will be contradicted that VC was a vertex cover. Now, since at least one of them is in T, the intersections $V(P_1) \cap T$ and $V(P_2) \cap T$ are necessarily distinct.

3. $P_1 = (s - a - e - t)$ and $P_2 = (s - v - e - t)$ where $v \neq a$

   In this case, we simply note that as $a \in T$, the intersection $V(P_1) \cap T$ contains a, while the set $V(P_2) \cap T$ does not contain a, and hence the two are distinct.

The three cases above cover all possible pairs of shortest $s-t$ paths in G', and we have shown that their intersections with T are different in each case. This completes the proof that T is indeed a tracking set of G' of size at most $m+k$. Hence, proved.

Before proving the opposite direction, we prove a lemma.

Lemma 1

For any graph $G = (V, E)$ (note that we still are considering that there are no isolated vertices in the graph), where $|V| = n$, the minimum vertex cover has size at most $n - 1$.

Proof

Assume for sake of contradiction that the minimum vertex cover VC in some graph G has size n. Then it implies that $VC = V$. But, on taking any single vertex $v \in VC$ and considering the edges incident on v, we see that all these edges are covered by the other endpoints of v (as they are also in V and hence in VC). So, the removal of v from VC will not result in any edge being uncovered. So, the set VC was not a minimum vertex cover. Hence, proved.

Claim 2 (Backward Direction)

Using the construction given in proof of Claim 1, if a set T is a tracking set of G' of size at most $m + k$, then there exists a vertex cover of G of size at most k.

Proof

First, consider the paths $P_1 = (s - a - e_1 - t)$ and $P_2 = (s - a - e_2 - t)$ such that $e_1 \neq e_2$. We can see that if both $e_1 \notin T$ and $e_2 \notin T$, then $V(P_1) \cap T = V(P_2) \cap T$ and hence T cannot be a tracking set. This implies that at most one $e \in V$ which corresponded to some edge $e_g \in E_g$ can be missing from T. Thus, at least $m - 1$ elements belong in T.
Now, consider the paths $P_1 = (s - a - e - t)$ and $P_2 = (s - v - e - t)$, such that $v \neq a$. This implies that at least one of a or v must be in T, otherwise $V(P_1) \cap T = V(P_2) \cap T$. We consider both of these cases below -

1. $a \notin T$
   In this case, for every vertex $v \in V$ and corresponding to a vertex $v_g \in V_g$, we must have that $v \in T$. This is because if any $v \notin T$, since we know that $a \notin T$, we can consider the paths $P_1 = (s - a - e - t)$ and $P_2 = (s - v - e - t)$ where e is connected to v, and see that their intersections with T will be the same, which is a contradiction. So, in this case the size of T is at least $(m - 1) + n$ or $m + (n - 1)$ (at least $m - 1$ elements were shown before, and here an additional n elements are also shown to belong to T). But, since we know that the minimum vertex cover of G must have at most $n - 1$ elements from Lemma 1, we have shown that there exists a vertex cover of size at most $n - 1 \leq |T| - m$ in G. So, if $|T| = m + k$, then we have shown that there exists a vertex cover of G of size at most $|T| - m = k$. Hence, for this case the reverse direction is proved.

2. $a \in T$
   In this case, we consider two paths $P_1 = (s - v_1 - e - t)$ and $P_2 = (s - v_2 - e - t)$, where v_1 and v_2 correspond to the endpoints of the edge to which e corresponds. Also, both v_1 and v_2 are not equal to a. We can see that either v_1 or v_2 must belong to T, otherwise $V(P_1) \cap T = V(P_2) \cap T$ which is a contradiction as T was a tracking set. Hence, at least one vertex corresponding to one endpoint of the edge to which e corresponds to must be in T. Since this is true for all $e \in V$, we can say that there are at least as many more vertices $v \in V$ which correspond to some $v_g \in V_g$ and also belong to T as the size of some vertex cover of G (say VC). So, we have shown that the size of T in this case is at least $|T| \geq (m-1) + 1 + |VC|$ ($m-1$ vertices were shown to be part of T earlier, and a is also a part of T in this case, and finally we also need at least as many vertices as that in a vertex cover of G). But as the size of T was at most $m + k$, we have shown that $|VC| + m \leq T \leq m + k$ or that $|VC| \leq k$. So, in this case as well, we have shown that there exists a vertex cover of G of size at most k.

As for both the cases, the reverse direction has been proved, we can say that the reverse direction always holds.

We have shown both the directions now, and so we conclude that the vertex cover decision problem can be reduced to an instance of the tracking shortest paths decision problem in polynomial time. As the tracking shortest paths problem was in NP, and as the vertex cover problem is NP-complete, we finally conclude that the tracking shortest paths problem is NP-complete. Hence, proved.

3 Flows and Cuts

Part 1

We will firstly construct a flow network (V', E', c) using $G = (V, E)$ where $|V| = n$ and $|E| = m$. Let the vertex set $V' = V \cup \{s, t\}$ such that s is the source node of the flow network while t is the sink node of the flow network. Also, for every undirected edge $e = (x, y) \in E$, include two directed edges $e_1 = (x, y)$ and $e_2 = (y, x)$ (each of capacity 1) in E'. In addition to these edges, for all $v \in V$, add an edge $(s, v) \in E'$ such that $c(s, v) = m$; and an edge $(v, t) \in E'$ such that $c(v, t) = m + 2\alpha - d(v)$, where α is the rational constant given in the question, and $d(v)$ denotes the degree of the vertex v in graph G. Now, we prove a few claims.

Claim 1

Let $A \subseteq V$ be a set of vertices in graph G. Then, the number of edges that have both endpoints in A ($E(A)$) is given by -

$$E(A) = \frac{\sum_{v \in A} d(v) - N_{out}}{2} \tag{1}$$

Where N_{out} is the number of edges which have exactly one endpoint in A.

Proof

Consider edges which have both endpoints in A, they will be counted twice in the sum $\sum_{v \in A} d(v)$, once for each endpoint. Also, consider the edges which have exactly one endpoint in A, these will only be considered once in the sum $\sum_{v \in A} d(v)$, only when the endpoint that is present in A is summed over. Thus, we can conclude that $\sum_{v \in A} d(v) = 2E(A) + N_{out}$, and rearranging this equation gives us the desired result. Hence, proved.

Claim 2

Let $A \subseteq V$ be a set of vertices in graph G and A^c denote its complement $V \backslash A$. Then $(A \cup \{s\}, A^c \cup \{t\})$ is an $s - t$ cut of G' with capacity equal to $mn + 2|A|(\alpha - D(A))$ where $D(A)$ denotes the density of the set A i.e. $D(A) = \frac{E(A)}{|A|}$.

Proof

To compute the capacity of the cut, we have to sum the capacities of all the edges e of the following forms -

- $e = (s, b)$ where $b \in A^c$. These edges all have capacity m and the number of such edges is $|A^c| = n - |A|$. So, the sum of capacities of these edges is $m(n - |A|)$.

- $e = (a, t)$ where $a \in A$. These edges have capacities $m + 2\alpha - d(v)$ for each $v \in A$. The summation of these over all the elements in A is therefore - $\sum_{v \in A}[m + 2\alpha - d(v)] = |A|(m + 2\alpha) - \sum_{v \in A} d(v)$
- $e = (a, b)$ where $a \in A$ and $b \in A^c$. These edges all have capacity 1, and so the sum of their capacities is $\sum_{(v,w) \in A \times A^c} 1$. However, notice that this is simply the number of edges in G which have exactly one endpoint in G, and hence this sum is equal to N_{out}.

Summing all of the three categories above, we get that the capacity of the cut C is -

$$C = m(n-|A|)+|A|(m+2\alpha)-\sum_{v\in A} d(v)+N_{out} = mn+2|A|(\alpha-(\sum_{v\in A} d(v)-N_{out})/(2|A|))$$

In the above equation, replacing the expression for the density of the subset A using Claim 1, we get that -

$$C = mn + 2|A|(\alpha - D(A)) \tag{2}$$

Hence, proved

Claim 3

Let $A \subseteq V$ be such that $(A \cup \{s\}, A^c \cup \{t\})$ is a minimum capacity cut. Then, there exists a subset with density at least α if $|A| \neq 0$. And if $|A| = 0$, then the density of all subsets is less than or equal to α.

Proof

Let C denote the capacity of the cut $(A \cup \{s\}, A^c \cup \{t\})$. Notice that we can create a cut $(s, V \cup \{t\}$ with capacity mn. Since, C is a minimum capacity cut, we must have that $C < mn$. Using Claim 2, this inequality can be written as -

$$mn + 2|A|(\alpha - D(A)) \leq mn$$

Which implies that

$$|A|(\alpha - D(A)) \leq 0$$

First, assume that $|A| \neq 0$. Then this implies that $(\alpha - D(A)) \leq 0$ or $D(A) \geq \alpha$. Hence, there exists a subset which has density equal to at least α.
Next, for proving the converse, let's assume that $|A| = 0$. This implies that the minimum capacity cut has capacity equal to mn. For the sake of contradiction, assume that the maximum density subset of the graph has density $D > \alpha$. Let the maximum density subset be M (Note that M has to be non-empty). Then the capacity of the cut $(M \cup \{s\}, M^c \cup \{t\})$ is given by $mn + 2|M|(\alpha - D) < mn$ (as $D > \alpha$). But, this cannot be possible as the minimum capacity of any cut was mn. Hence, by contradiction if $|A| = 0$, then we must have that the density of all subsets is less than or equal to α.
Hence, proved.

Claim 4

The possible number of densities that the subsets can have are polynomial in n and m.

Proof

Let the (distinct) densities of any two subsets be m_1/n_1 and m_2/n_2, where m_i and n_i are the number of edges with endpoints in the subset i and the number of vertices in n_i respectively. Notice that $m_i \in \{0, 1 \ldots m\}$ and $n_i \in \{1, 2 \ldots n\}$ and hence the number of values the density of a subset can take is at most $(m+1)n$ which is polynomial in m and n.

Algorithm

The following is the algorithm to find out in a graph whether there is a subset of V with density at least α.

1. Create the graph G' as described in the beginning of this section. Using the Ford-Fulkerson algorithm, find out the maximum flow in G', and using the residual graph, find out the cut of minimum capacity (the vertices reachable from s in the residual graph form one set of the cut, and the complement is the other set - done in class).
2. If the cut of minimum capacity has vertices other than s, then we know that there exists some subset with density greater than or equal to α, so return True.
3. Else return False

Proof of Correctness

The proof of correctness follows by the correctness of Claim 3 and the Ford-Fulkerson algorithm (done in class).

Time Complexity Analysis

The Ford-Fulkerson algorithm runs in polynomial time here, as the max flow coming out of the source s is bounded by mn which is polynomial in the input size, and as all the edge capacities are rationals, the algorithm is confirmed to terminate. After this, we only have to apply BFS from s in the residual graph, which will again take polynomial time to check for the size of the minimum capacity cut set. So, the entire algorithm takes polynomial time only.

Part 2

Algorithm

The algorithm to find maximum density subgraph is as follows -

1. Since there are only a polynomial number of possible densities (as shown in Claim 4), we find all the possible densities (a/b such that $a \in \{0, 1 \dots m\}$ and $b \in \{1, 2 \dots n\}$) and sort these into an array A.
2. Now, for each $\alpha \in Arr$, we can find if there exists a subgraph of G with density at least α or not. We keep doing this till we reach the last element in Arr, or we get to an element in Arr for which we cannot find a subgraph with density greater than or equal to it. Let the last element for which there there was a subset with density greater than or equal to it be α_0. Then, the only possible candidates that we can have for the maximum density are α_0 and the element just after α_0 (let it be called α_1; also note that if α_0 is the last element in Arr then the maximum density is automatically equal to it). This is because the elements before α_0 cannot be the maximum density as the maximum density is greater than or equal to α_0, and the elements after α_1 also cannot be the maximum density as the maximum density is less than or equal to α_1 (Arr is sorted).
3. Now, again check if there exists a subgraph of G with density at least $\alpha = (\alpha_0 + \alpha_1)/2$ or not using the algorithm in Part 1. If there exists, then we can be sure that this density is equal to α_1, otherwise it is equal to α_1. This is because the densities can only be out of these two, and if we know if it less than equal to or greater than equal to their mid-point, then we can pinpoint which one it is.
4. Once we know the maximum density α, we know that since the algorithm developed in Part 1 returns a minimum capacity cutset $(A \cup \{s\}, A^c \cup \{t\})$ such that the density of A is at least α (done in proof of part 1), we can conclude that on running the Ford-Fulkerson algorithm a last time and finding A, we have found a subset of maximum density (as the density is greater than or equal to the maximum density, so it must be equal to the maximum density. This is the required set S in the question.

Proof of Correctness

The proof of correctness follows from the arguments made in the claims as well as the arguments made in the steps above.

Time Complexity

Since all the steps (sorting, Ford-Fulkerson and the final finding of the set A) can be carried out in polynomial time, and the number of times these must be performed is also polynomial (the size of sorted array Arr), we can conclude that the overall running time of this algorithm is polynomial in the size of the input. Hence, proved.