Name: Ke Liu PID: A5327422

Q1: Graph cohesiveness

1. Give a polynomial-time algorithm that takes a rational number α and determines whether there exists a set S with cohesiveness at least α.

HLD: We have graph $G = (V, E)$ where $V = \{x, y, z, ...\}$ denotes the set of people and $E = \{(x, y), (y, z), ...\}$ denotes the friendship between two people.

1). Construct a flow network $G' = (V', E')$ as shown in figure 1.

i. Nodes: V' includes s, t, u_{xy} if x and y are friends (that is, $for\ all\ (x, y) \in E$) and x for all $x \in V$.

ii. Edges: (s, u_{xy}) with capacity 1. (u_{xy}, x) and (u_{xy}, y) with capacity ∞. (x, t) with capacity α.

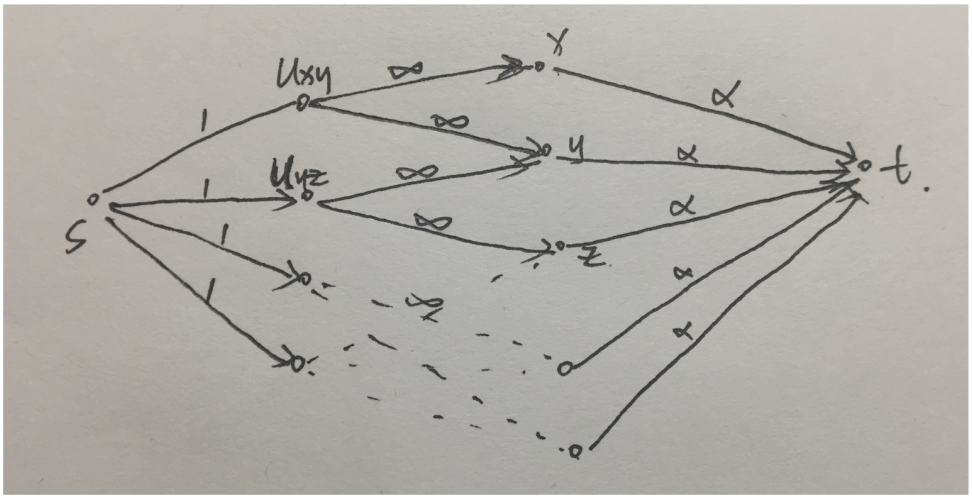

Figure 1

2). Find the max-flow min-cut of the flow network with *Preflow-Push Maximum-Flow algorithm*. We denote the max flow as Z.

$Z \leq |E|$ since $C = \sum_{su \in E} c(su) = |E|$. The max flow must be no larger than the capacity sum of s and its adjacent nodes.

If $Z < |E|$, then there exists a set S with cohesiveness at least α.

We have $\min_{cut(A,B)} = C(A, B)$. $A' = A - \{s\}$. A' includes two parts: nodes like x that represent vertices in V and nodes like u_{xy} which represents edges among nodes. The first part will be the set *S* and the second part is edges in *S*, that is, *e(S)*.

Proof of Correctness:

Claim 0.1: If $Z < |E|$, then there exists a set S with cohesiveness at least α.

If $Z < |E|$, then the edges leaving s are not all saturated. And since the cut edges have finite capacity, edges like (u_{xy}, x) cannot exist in the cut. So there must be at least an

cut edge (w,t) where w is a node in V. According to the max-flow min-cut algorithm, $f(wt) = c(wt) = \alpha$. Thus w must be in set S where could find a cohesiveness with α.

To formulate the min cut, we should note that if $u_{xy} \in A$, then x and y must be in A because the cut has finite capacity. And if x and y in A, then $u_{xy} \in A$, otherwise, the cut capacity will increase $c(s, u_{xy})$ and thus the cut will not be the min cut. We set S' to be $A \cap V$, that is, the vertices of V that belong to A.

Then we will figure out the formula of min_cut(A, B):

$$\begin{aligned}\min\_cut(A,B) &= C(A,B)\\ &= \sum_{u_{xy}\notin A} 1 + \sum_{x\in A} \alpha\\ &= \sum_{u_{xy}\notin A} 1 + \sum_{x\in A} \alpha + \sum_{u_{xy}\in A} 1 - \sum_{u_{xy}\in A} 1\\ &= |E| + \sum_{x\in A} \alpha - \sum_{u_{xy}\in A} 1\\ &= |E| + \alpha|S'| - e(S')\end{aligned}$$

If the min cut Z<|E|, we could obtain $|E| + \alpha|S'| - e(S') < |E|$, after simplifying, we can get $\alpha < e(S')/|S'|$. Thus the set S' has cohesiveness at least α.

Claim 0.2 If a set S has cohesiveness at least α, then the max flow Z in our network will be less than |E|.

We will do this by proof of contradiction. Because the capacity of edges leaving s sums up to |E|. According to the capacity constraint, the max flow will be no larger than |E|. So we assume the max flow is |E|. The min cut=$|E| + \alpha|S'| - e(S')$=|E|. We will obtain $\alpha = e(S')/|S'|$, which is a contradiction of our precondition that S has cohesiveness at least α. Hence the max flow will be less than |E|.

According to the two claims, we obtain that if $Z < |E|$, then there exists a set S with cohesiveness at least α.

Time complexity:

There are $(|V| + |E|)$ nodes in G'(except s and t). So the Preflow-Push algorithm takes $O((|V| + |E|)^3)$ time.

2. Give a polynomial-time algorithm to find a set S of nodes with maximum cohesiveness.

HLD: We could use the algorithm in question 1 to find out the set S with cohesiveness at least α. To find the set with maximum cohesiveness, we should determine the value of α. Since α is a rational number, we cannot decrease from an integer one by one. Hence we need to figure out all possible α for graph $G = (V, E)$. There are at most |V| nodes in set S. Assume for a set S with n nodes, the maximum $e(S)$ is C_n^2. So for a set with n nodes, the possible $e(S)$ may be $0, 1, \dots, n(n-1)/2$. The maximum α should be $\frac{e(S)}{S} = \frac{C_{|V|}^2}{|V|} = \frac{||V|-1|}{2}$. We begin from the maximum α and use the algorithm in question 1 to check if the max flow is less than |E|. We apply this method to all possible $\frac{e(S)}{S}$ from largest to smallest and once we find a min cut value less than |E|, we could return the maximum cohesiveness.

Proof of correctness: We have enumerated all possible cohesiveness. The maximum cohesiveness must be in one of them. Then we apply the algorithm in question 1 which return the result that for a particular α, if there exists a set S with cohesiveness at least α. Since we check with the descending order of α, then the first time when meets the requirement will return the maximum cohesiveness.

Time complexity: We need to find out all possible α in $O(|V|^3)$ time. And then apply the Preflow-Push algorithm in question 1 which takes $O((|V| + |E|)^3)$ time. So the total time complexity is $O(|V|^3(|V| + |E|)^3))$.

Q2. Number Puzzle

HLD: The purpose is to check given input, if we can find a matrix that satisfies all the constraints. So first we should check if the inputs are valid. Since the output is a matrix $a_{i,j}$ of integers between 1 and M so that $\sum_i a_{i,j} = c_j \; for \; 1 \le j \le n$ and $\sum_j a_{i,j} = r_i \; for \; 1 \le i \le n$, so we need to check:

the sum of a row or a column should be no less than n;

the sum of a row or a column should be no larger than Mn;

the sum of all rows should be the same to the sum of all columns.

If $\exists r_i \; or \; c_j, \; (r_i < n) \; or \; (r_i > Mn) \; or \; (c_j < n) \; or \; (c_j > Mn)$ or $(\sum_i r_i \neq \sum_j c_j)$, then output impossible.

After checking the input, we try to find all the n^2 entries in the matrix. To solve the puzzle, we first construct a flow network as shown in Figure 2. Like Bipartite max matching, we split the nodes in two group, *Row* and *Col*, where each node represents a row or a column. In directed graph $G = (V, E)$, we have:

$V = \{s\} \cup Row \cup Col \cup \{t\}$

$Edges \; will \; include \; three \; parts$:

$\{(s, R_i) \; for$ all $R_i \in Row,$ with capacity $r_i - n\}$

$\{(R_i, C_j)$ for $all \; R_i \in Row \; and \; C_j \in Col, \; with \; capacity \; M - 1\}$

$\{(C_j, t) \; for \; all \; C_j \in Col, \; with \; capacity \; c_j - n\}$

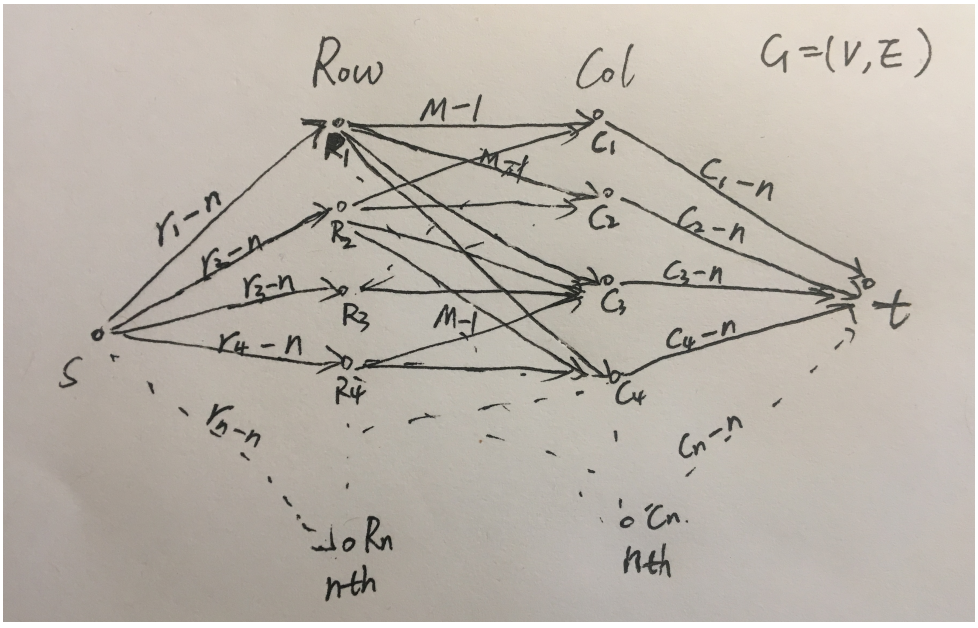

Figure 2

For an edge (R_i, C_j), we set the capacity as $M - 1$. The *flow* in that edge is exactly $(a_{i,j} - 1)$. The reason is that the flow in that edge could be 0 while $1 \le a_{i,j} \le M$. So

we reduce the capacity to be $M-1$ and thus the capacity of (s, R_i) is $r_i - n$.
Then we apply Ford-Fulkerson algorithm to find the max-flow min-cut. If the max flow is not $\sum_i (r_i - n)$, output impossible. Otherwise, we could construct a matrix with $a_{i,j} = f(R_i, C_j) + 1$.

Proof of Correctness:

Claim 0.1 If the max flow is $\sum_i (r_i - n)$, then we could find a matrix that satisfies all constraints.

Since $\sum_i (r_i - n) = \sum_j (c_j - n)$, if the max flow is $\sum_i (r_i - n)$, then all edges that leaving s and all edges that reaching t are saturated. So that $\sum_i a_{i,j} = c_j \; for \; 1 \leq j \leq n$ and $\sum_j a_{i,j} = r_i \; for \; 1 \leq i \leq n$ is satisfied. The flows in (R_i, C_j) is within [0,M-1]. So the value of every entries in matrix is within [1,M]. All these two constaints are met, so the matrix is a valid one.

Claim 0.2 If there exists a matrix that satisfies all constraints, then the max flow is $\sum_i (r_i - n)$.

If the max flow is less than $\sum_i (r_i - n)$, then there must be at least one edge, let's say (s, R_i), such that $f(s, R_i) < c(s, R_i)$. In that case, the sum of row i is less than r_i, which means there does not exist a matrix that satisfies all constrictions imposed by the input.

Time Complexity:

Constructing the flow network needs $O(n^2)$ time. And applying the Ford-Fulkerson algorithm needs polynomial time. Thus the total algorithm needs polynomial time complexity.

Q3. Database Projections

First we need to construct the graph $G = (V, E)$ as shown in Figure 3. In the graph, we have two sets of nodes, set1=$\{S_1, S_2, \dots, S_k\}$ and the copy of these k subsets $set2 = \{S_1', S_2', \dots, S_k'\}$.

Node: $\{s, t\} \cup set1 \cup set2$

Edges have three parts:

{(s,S_i) for S_i in set1, with capacity 1}

$\{(S_i, S_j')\ if\ S_i \subset S_j,\ with\ capacity\ \infty\}$

$\{(S_j', t) for\ S_j'\ in\ set2,\ with\ capacity\ 1\}$

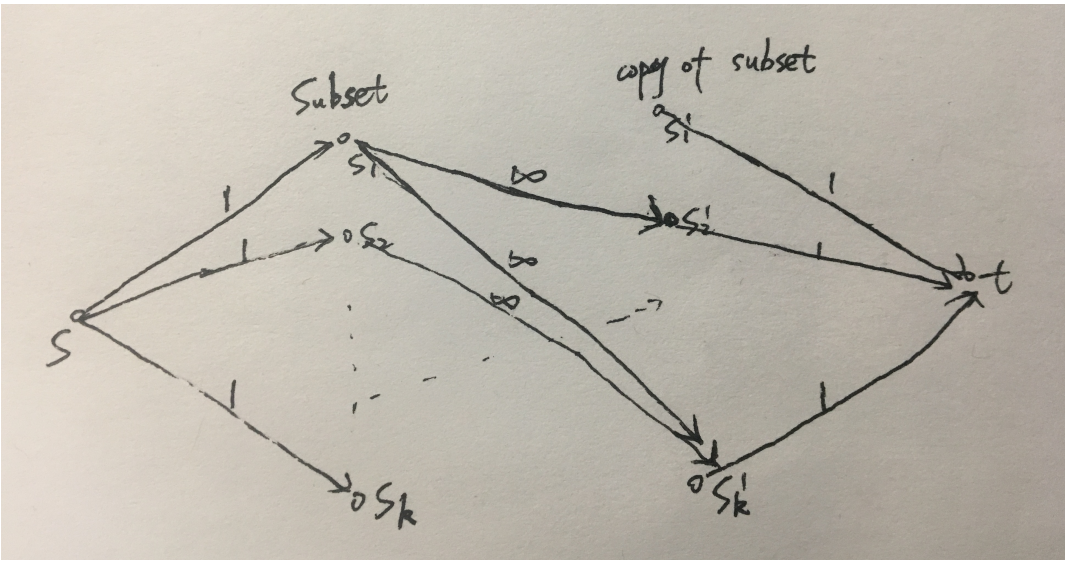

Figure 3

Then we use Ford-Fulkerson algorithm to find the max-flow min-cut of the flow network. Suppose we have max flow f, then if $l \geq k - f$, we could find l permutations of columns $p_1, p_2, \dots, p_l$ for every one of subset S_i, there exists that S_i constitute a prefix of at least one of the permuted tables.

Arrange subset in $k - f$ permutations. If the flow of an edge (S_i, S_j') is 1, then we will have a chain with $S_i \subset S_j$ in it. $S_i\ and\ S_j$ will be in the same permutation. We find all the $edge(S_i, S_j')$ with flow=1. All the subsets that have subset($\subset$) relationship will be set in one permutation. For example, $S_1 \subset S_2 \subset S_3$, we will have a permutation that begins with elements in S_1 and then elements in $S_2 - S_1$ and then $S_3 - S_2$. By doing this, we can find all $k - f$ permutations.

Proof of Correctness:

If $S_i \subset S_j \subset S_k$, then all these subsets can be covered in one chain (permutation). The permutation starts with the elements in S_i and then appends with elements in $S_j - S_i$, at last appends the elements in $S_k - S_j$. So we need to find the possible subset relationship between subsets $(S_1, S_2, \dots, S_k)$ to determine permutations.

Claim 0.1 We could cover all subsets $(S_1, S_2, \dots, S_k)$ in (k-f) permutations.

Since the finite capacities are all 1s, we are assured that each flow is either 1 or 0. There cannot be a node in $set1$ having flow=1 in two edges from $set1$ to $set2$ because of the conservation property. By the same time, the flow into nodes in $set2$ cannot exceed 1 since the capacity from node in $set2$ to t is 1. Assume we have a chain $S_i \subset S_j \subset S_k$, in order to get the max flow, there will be a flow=1 from S_i to S_j' and from S_j to S_k'. If we have flow from S_i to S_k', then the flow on edge (S_j, S_k') will be 0 because of the reason we just mentioned, thus the flow will only decrease. So the max flow will ensure that for a chain, there will only be flow from a subset to its successive subset in our network.

As shown in Figure 4, at first, we may have k permutations for all k subsets. Suppose that there is a path $s \to S_2 \to S_3' \to t$ with flow=1. We can have a chain with $S_2 \subset S_3$ in it. So in the first layer, we only need to consider $(k-1)$ subsets(permutations) because S_3 will be "attached" to S_2. In that way, as the flow between $set1$ to $set2$ increases one, there will be a subset disappear in the first layer as shown in Figure 4. The max flow of our graph is f, so the subsets(permutations) in the first layer will be $(k-f)$. All the other subsets have been "attached" to its previous subset in a chain. Hence we can cover all subsets in $(k-f)$ permutations.

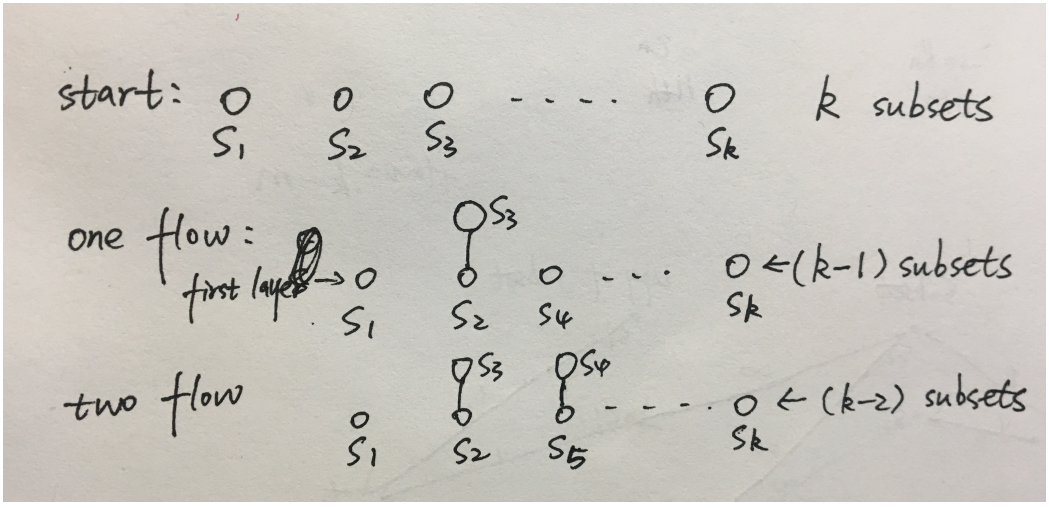

Figure 4

Claim 0.2 We could not cover all subsets $(S_1, S_2, \dots, S_k)$ in less than $(k - f)$ permutations.

We will prove by contradiction. Assume we can cover all subsets in $(k - w)$ permutations while $w > f$. Therefore we will have w subsets "attached" to other subsets. So there will be w paths of $s \to S_i \to S_j' \to t (i \neq j, i, j \in [1, k])$ with flow=1. Since the flow leaving S_i and the flow into S_j' are at most 1 because of the conservation property, there will be at least w subsets S_i in $set1$ have flow=1. So the max flow will be at least w, which is a contradiction of max flow is f.

Through these proofs, we can conclude that we could cover all subsets $(S_1, S_2, \dots, S_k)$ in at least $(k - f)$ permutations. So if $l \geq k - f$, we can find l permutations to cover all the k subsets.

Time Complexity: We need to find all the subset relationship between the k subsets which takes $O(k^2)$ time to compare. And the Ford-Fulkerson could have at most k augmenting path. The time to find max flow is $O(k^3)$. So the total time complexity is $O(k^3)$.

Q4. Maximum likelihood points of failure

HLD: Let $X = -log \prod_{i \in F} p_i = \sum_{i \in F}(-logp_i)$. We could find the minimum X in order to get the maximum failure probability.

Given a directed graph $G = (V, E)$, we will construct a flow network $G' = (V', E')$ as shown in figure 5. In this network, we have two sets of nodes(except s and t): $set1$ which includes all vertices in V and $set2$ which is a copy of $set1$, that is, for any vertex u_i in V, we obtain two nodes u_i in $set1$ and u_i' in $set2$.

Nodes: $\{s, t\} \cup set1 \cup set2$

Edges include four parts:

$\{(s, u_i)\ if\ there\ is\ an\ edge\ ((s, u_i))\ in\ V, with\ capacity\ \infty\}$

$\{(u_i, u_i')\ for\ u_i\ in\ set1\ with\ capacity - logp_i\}$

$\{(u_i', u_j), (u_j', u_i)\ if\ there\ is\ an\ edge\ (u_i, u_j) in\ V, with\ capacity\ \infty\}$

$\{(u_i', t) if\ there\ is\ an\ edge\ ((u_i, t)) in\ V, with\ capacity\ \infty\}$

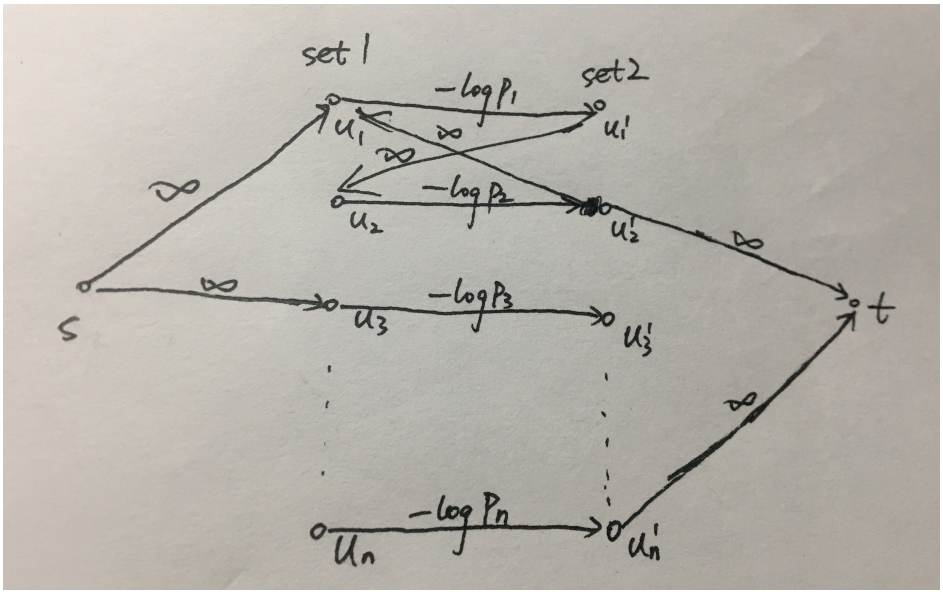

Figure 5

Since we are dealing with rational values, we will find the max-flow min-cut with Preflow-Push algorithm. We should note that if there is no direct edge from s to t, then we could find a min-cut, since if we delete all nodes except s and t, we can disconnect t from s. Because of finite capacity, the edges that cross the min cut will all be edges like (u_i, u_i'). The failure point F will be all u_i that have a cut edge (u_i, u_i').

We denote the max flow (min cut) as m. We have $m = -log \prod_{i \in F} p_i = X$. The maximum failure probability $\prod_{i \in F} p_i = 2^{-m}$.

Proof of Correctness:

Claim 0.1 If we have a cut(A, B) with capacity m, then we have a failure probability 2^{-m}.

Since the cut has finite capacity, the edges that cross the cut must be like (u_i, u_i'). After removing these cut edges, there won't be edges from A to B, otherwise, this is not a cut. So we could disconnect t from s by removing these edges, that is, removing vertices u_i if (u_i, u_i') is a cut edge. So these removed nodes form a failure point F with failure probabity $\prod_{i \in F} p_i = 2^{-m}$.

Claim 0.2 If we have a maximum failure probability 2^{-m}, then the min cut in the network is m.

A maximum failure point F is defined as $\prod_{i \in F} p_i$. If we add other vertices in V into F, the failure probability will only decrease since $0 \leq p_i \leq 1$.

We can cut the connection between s and t by cutting all edges (u_i, u_i'), but cutting redundant edges will only decrease the failure probability. So we need to find the min cut such that there is no other edges from A to B.

These failure point can disconnect t from s, so we can assure that if we delete those nodes(cut the edges between node in F and its corresponding node in set2), there s and t will be disconnected.

There will be a set of nodes in F such that $\prod_{i \in F} p_i = 2^{-m}$. Since it's the maximum failure probability, so $X = -log \prod_{i \in F} p_i$ will be minimized. And after cutting the edges of corresponding nodes, we can cut the graph. So m=X is the min cut.

By showing these two proofs, we could find the maximum failure probability by finding the min cut.

Time Complexity: The are (2|V|+2) nodes in the network. The Preflow-Push Algorithm will take $O(|V|^3)$ time to find the min cut. So the time complexity is $O(|V|^3)$.