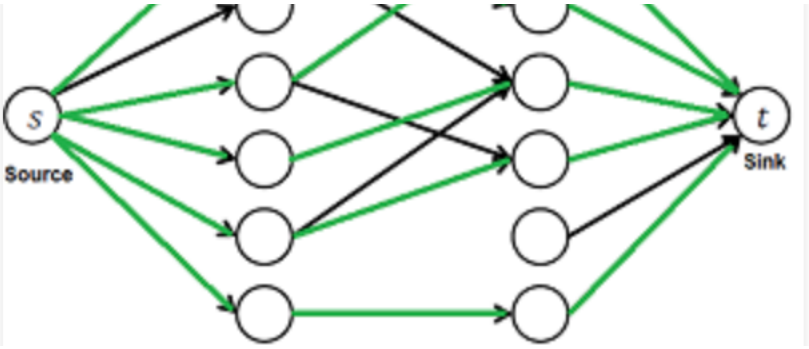
Ali Mirabzadeh
305179067
CS 180 – HW8
22

a. 1 1 0
   0 0 1
   0 0 1
   The above matrix is not rearrangeable.

b. This problem can be deduced to finding a perfect matching in a bipartite graph. Indeed, construct the following bipartite graph (A, B). Let A be the vertices corresponding to rows and B be the vertices corresponding to columns. Join a ∈ A and b ∈ B with an edge only when the corresponding entry in the matrix is 1. It is easy to see that matrix will be re-arrangeable if and only if this bipartite graph has a perfect matching. So we create a network flow graph by adding sources, s, and sink, t, node to do so.



This picture is from https://www.geeksforgeeks.org/maximum-bipartite-matching/

Ali Mirabzadeh
305179067
CS 180 – HW8
23.
First let's design an auxiliary algorithm A which computes the minimum cut (S, T). Run the Ford-Fulkerson algorithm to find the maximum flow and look at the final residual graph G then run DFS from the source s. Suppose DFS observed the vertex set S. We claim that (S, V − S) is the desired cut. Indeed, notice that for every vertex in S which is a leaf found by BFS, there is an edge of full capacity that comes in from some vertex of V − S and these are the only non-zero capacity edges between S and V – S. Hence, this cut is a minimum one and S has the minimum cardinality because of the property of DFS - we found a cut with S that has the first vertices in the path from s from which only saturated edges are going further. Algorithm A clearly is polynomial. Now run algorithm A twice. First on our original flow network, say the output cut is (S, V −S). We claim that S is the set of upstream vertices. Run it second time on the "reversed network" and obtain the cut (T, V −T). T is the set of downstream vertices. It is not hard to show that all the remaining vertices (i.e. V − S − T) are central.

Ali Mirabzadeh
305179067
CS 180 – HW8
34

a. To check for the back-up devices that can be found, the phones that are in Pi are connected fully. The graph G= (X U Y, E), birpartite, represents conncetion of X phone with Y stations. We call the distance of an edge d. G is in perfect matching state, if all phones are connected to the base station. Based on bipartite graph, it takes O(n^3) to check for a perfect match state.
   A position t is such that it lies on path of phone Pi. We know that it is either the first phone to encounter the base or is the first to go off range. Also, we know that the path cam be tracked in a linear format and the k set of devices in contact with base Bj is in a circular of radius d. Also, it is obvious that a line can cross a circle two times; there is only 1 event in which P1 is in contact with base Bj and exactly 1 event in which goes off base Bj. Therefore, for each phone to maintain contact with base it are 2*n. events in total. Each phone can be traced by Its co-ordinates.
   The bipartite graph G doesn't change between consecutive events interval. It changes for any event Ti, when an edge which is incident to Pi is added or removed from the contact set. G[Ti] is the state of graph just after event Ti has taken place. Hence, the graph G can be said in perfect matching if and only if each sub-state G; are all inperfect matching state.

b.

   I use the same explanation to solve this part.
   The graph G can be tested in O(n^4) time for perfect matching state. This time can be reduced because a perfect matching M for graph state G[Ti] can act matching state for graph state G[Ti+1], if any of edge in M is removed during event Ti. To evaluate it, a single augmenting path is traced to decide if matching size n-1 exits or not, which can raised to perfect match. The time taken for single augmenting path check is O(|E|) = O(n^2). Hence the time for checking is reduced to O(n^3), because for initial graph state G there is no check made and for all other states it is O(kn^2). Also, as k<= 2*n, the total time becomes O(n^3)