Algorithms Tutorial 5
Solutions

1. In the country of Pipelistan there are several oil wells, several oil refineries and many distribution hubs connected by oil pipelines. To visualise Pipelistan's oil infrastructure, just imagine a directed graph with k source vertices (the oil wells), m sinks (refineries) and n vertices which are distribution hubs linked with one way pipelines. You are given the graph and the capacity $C(i, j)$ of each pipeline going from a vertex i to a vertex j. You want to instal the smallest possible number of flow meters on some of these pipelines so that the total throughput of oil from all the wells to all refineries can be computed exactly from the readings of all of these meters. Each meter shows the direction of the flow and the quantity of flow per minute. Design an efficient algorithm for deciding on which pipelines to place the flow meters.

Solutions: Note that the readings have to provide an accurate estimate of the total flow from all oil wells to all refineries for every flow, not just the maximal flow. This is ensured by constructing a flow network as a directed graph where the oil wells, oil refineries and hubs are the vertices of the graph and each pipeline is represented by TWO directed edges of opposite orientation and each of capacity equal to 1. The oil wells will play the role of multiple sources and the refineries will play the role of multiple sinks. Thus, we add a super source S and a super sink T and connect S with all the wells with edges of infinite capacity and similarly all the refineries with T. We now run the Edmons-Karp algorithm to find the maximal flow through such a network (note that we have ignored the actual capacities of the edges). After the algorithm has converged, we construct the last residual network flow and look at all the vertices to which there is a path from the source S. This will define a minimal cut, so we look at all the edges crossing such a minimal cut. The number of such edges (in the forward direction only) determines the minimal number of meters needed to accurately compute the total flow from S to T.

2. Given an undirected graph with vertices numbered $1, 2, \ldots, n$, partition the vertices into two disjoint subsets such that vertex 1 and n are in different subsets, and the number of edges with both ends in the same subset is maximised.

   Solution: We again make a corresponding network flow with the source at vertex 1 and the sink at vertex n and two directed edges replacing each undirected edge. The problem is now equivalent to minimising the number of edges between the two subsets; since all the edges appear in pairs of edges with the same end points but opposite direction, whenever there is an edge from the sink side to the source side, there will also be an edge in the opposite direction which min cut will take in the account.

3. Assume that you are given a network flow graph with a source s, a sink t and two other distinct vertices u and v. Design an algorithm which returns a smallest capacity cut among all cuts for which the vertex u is in the same side of the cut as the source s and vertex v is in the same side as the sink t and which runs in polynomial time.

   Solution: Connect the source s with vertex u with an edge of infinite capacity and similarly connect vertex v with the sink t with an edge also of infinite capacity. Use the Edmons-Karp algorithm to find the maximal flow through such a network and then the corresponding minimal cut. Clearly the two added edges of infinite capacity cannot belong to the min cut which ensures that u stays at the same side as s and v at the same side as t.

4. Assume that you are given a network flow graph with a source s, a sink t and two other distinct vertices u and v. Design an algorithm which returns a smallest capacity cut among all cuts for which vertices u and v are in the same side of the cut.

   Solution: Connect vertex u with vertex v and also vertex v with vertex u, both with directed edges of infinite capacity. Note that if we add only an edge of infinite capacity from u to v, v can still end up on the source side and u on the sink side and so the edge of infinite capacity will not belong to the cut because it is from the sink side to the source side. To prevent this we need edges between u and v in both directions.

5. You know that $n + 2$ spies S, $s_1, s_2, \ldots, s_n$ and T are communicating through certain number of communication channels; in fact, for each i and each j you know if there is a channel through which spy s_i can send a secret message to spy s_j or if there is no such a channel (i.e., you know what the graph looks like, with spies as vertices and communication channels as directed edges).

(a) Your task is to design an algorithm which finds the fewest number of channels which you need to compromise (for example, by placing a listening device on that channel) so that spy S cannot send a message to spy T through a sequence of intermediary spies without the message being passed through at least one compromised channel.

(b) Assume now that you cannot compromise channels because they are encrypted, so the only thing you can do is bribe some of the spies. Design an algorithm which finds the smallest number of spies which you need to bribe so that S cannot send a message to T without the message going through at least one of the bribed spies as an intermediary.

(a) Solution: This is just a max flow/min cut problem, in a flow network with an edge from a spy i to a spy j just in case there is a communication link from i to j, with a unit capacity. We find a maximal flow in such a network and the corresponding min cut. The edges that have to be compromised are those that cross the min cut.

(b) Solution: This is again a max flow/min cut problem with a unit vertex capacity and infinite capacity of all edges. Such problems are solved as explained in the lectures, by splitting each vertex other than the source and the sink into two vertices, one incident to all incoming edges and one incident to all outgoing edges, with these two vertices connected by an edge of capacity equal to the capacity of that vertex, in this problem the unit capacity.

6. There are N cities, labelled 1, 2, ..., N, connected by M bidirectional roads. Each road connects two different cities. A pair of cities may be connected by multiple roads. A well known criminal is currently in city 1 and wishes to get to city N via road. To catch the criminal, the police have decided to block the minimum number of roads possible to make it impossible to get from city 1 to city N. However, some roads are major roads. In order to avoid disruption, the police cannot close any major roads. Find the minimum number of roads to block to prevent the criminal from going from city 1 to city N, or output -1 if the police cannot stop the criminal.

Solution: We first construct a corresponding flow network with the cities as vertices and with city 1 as the source and city N as the sink. If there are k roads between two cities i and j but none of the roads is a major road, we connect i and j with two directed edges in opposite directions and of capacity each equal to k. If one of the roads is a major road the capacity of the two directed edges is set to $M+1$. We now find max flow in such a network; if it is

at most M then the problem has a solution and roads that need to be blocked are those that cross the corresponding min cut. If the flow is more than M then at least one edge of capacity $M + 1$ has been crossed, which indicates that every cut is crossed by a major road which cannot be blocked and thus we cannot catch the criminal and we must output -1.

7. An $n \times n$ grid is an undirected graph consisting of n rows, each row containing n vertices, with vertices connected with edges to all of their immediate neighbours (2 at all of the 4 corners, 3 at all of the 4 sides and 4 in the interior of the grid). Vertices with exactly 4 neighbours are called the internal vertices; vertices with exactly 3 neighbours are called the side vertices. The escape problem is, given $m \leq 4(n-1)$ many internal vertices in the grid and m side vertices on the 4 sides of the grid, connect each of m many distinct internal vertices with distinct m side vertices by non intersecting paths or return impossible when there is no such a solution. Note that any of m internal vertices can be connected to any of m side vertices.

Solution: This is very similar to a problem with blue and red vertices from the lecture slides. We add a super source vertex and a super sink vertex. Super source is connected to all of m internal chosen vertices and the super sink to all of m side vertices. Each edge of the grid is replaced with two directed edges in opposite directions, all at unit capacity. Finally, every internal vertex is assigned capacity of 1. We now find max flow in such a network using the vertex splitting method described on the lecture slides. All edges occupied with a flow form the connecting paths and if the flow is equal to m the problem has a solution.

8. A band of M criminals has infiltrated a secure building, which is structured as an $N \times N$ square grid of rooms, each of which has a door on all of its sides. Thus: from an internal room, we can move to any of the four neighbouring rooms, from a room on the side of the building, we can move to three other rooms or leave the building, and from a corner room, we can move to two other rooms or leave the building. The criminals were able to shut down the building's security system before entering, but during their nefarious activities, the security system became operational again, so they decided to abort the mission and attempt to escape. The building has a sensor in each room, which becomes active when an intruder is detected, but only triggers the alarm if it is activated again. Thus, the criminals may be able to escape if they can all reach the outside of the building without any two of them passing through a same room. Given the M different rooms which the criminals occupy when the

security system is reactivated, determine whether all M criminals can escape without triggering the alarm.

Solution: Producing the flow graph: First replace all the edges with two directed edges in opposite directions, thus making the graph directed and set the weights of all edges to 1. Either first assign capacity 1 to all vertices, or directly replace each vertex v with two vertices, $v1$ and $v2$; all edges which were incoming to vertex v will be now made incoming to vertex v_1 and all the edges outgoing from vertex v will be made outgoing from vertex v_2. Finally, connect vertex v_1 with vertex v_2 with an edge going from v_1 to v_2 also of capacity 1. Having done that for all vertices, add a supper source and connect it to the M internal vertices (where the robbers are) with edges of unit capacity as well; also add a super sink and connect all the side vertices to the super sink with edges of capacity 1 as well. Now find max flow in such a network; the problem has a solution if the flow is exactly equal to M. Note that our construction insures that every vertex and every edge can belong to only one path, so such paths will be disjoint, sharing neither edges not vertices.

9. Several families are coming to a birthday celebration at a restaurant. You have arranged that v many tables will serve only vegetarian dishes, p many tables will not serve pork and n many remaining tables will serve food with pork. You know that V many families are all vegetarians, P_1 many families do not eat pork but do not mind eating vegetarian dishes, P_2 many families do not eat pork but hate vegetarian dishes. Also R_1 many families have no dietary restrictions and would also not mind eating vegetarian dishes or food without pork, R_2 many families have no dietary restrictions but hate vegetarian dishes but can eat food without pork. Finally, S many families are from Serbia and cannot imagine not eating pork. You are also given the number of family members in each family and the number of seats at each table. Your conundrum is to place the guests at the tables so that their food preferences are respected and no two members from the same family sit at the same table. In case the problem has no solutions your algorithm should output statement "no solution".

Solution: This is a typical max flow problem. Make a bipartite graph with vertices on the left side representing families and vertices on the right side representing tables. Introduce a super source and a super sink and connect each family with the super source by a directed edge of capacity equal to the number of family members. Connect each table with the super sink with a directed edge of capacity equal to the number of seats at that table. Further, connect each family vertex with all the tables compatible with the dietary

preference of that family using directed edges of capacity 1 (to ensure that no table contains more than one family member from the same family. Now just use max flow algorithm and look at occupied edges to determine placement of guests.

10. Today was just a regular day for everyone in Krypton until a news flashed that a meteor is going to destroy Krypton in X days. Krypton has N cities, some of which are connected by bidirectional roads. You are given a road map of Krypton; for every two cities C_i and C_j which are connected by a (direct) road from C_i straight to C_j you are given the value $t(i, j)$ which is the number of days to travel from city C_i to city C_j. (You can of course also go from a city C_m to city C_k without a direct road from C_m to C_k by going through a sequence of intermediate cities connected by direct roads.) In each city C_i the Krypton Government built q_i pods to carry inhabitants in case of any calamity, which will transport them to Earth. City C_i has population p_i. As soon as the people hear this news they try to save themselves by acquiring these pods either at their own city or in other city before the meteor destroys everything. Note that a pod can carry only one person. Find the largest number of invaders the Earth will have to deal with. (20 pts)

Hint: First apply the Floyd Warshall algorithm to the weighted graph with cities as its vertices and edges representing roads, with each edge having a weight equal to the time $t(i, j)$ needed to traverse it. In this way we find the shortest distance from every node to any other node and determine which of these distances are less than X. Construct a bipartite graph; on the left there will be N nodes l_i, one for each city c_i representing population of city c_i; on the right there will be N nodes r_j, again one for each city c_j, but this time representing the set of pods available in city c_j.

11. You have been told of the wonder and beauty of a very famous painting. It is painted in the hypermodern style, and so it is simply an N by N grid of squares, with each square coloured either black or white. You have never seen this picture for yourself, but have been told some details of it by a friend. Your friend has told you the value of N and the number of white squares in each row and each column. Additionally, your friend has also been kind enough to tell you the specific colour of some squares: some squares are black, some are white, and the rest she simply could not remember. The more details she tells you, the more amazing this painting becomes but you begin to wonder that perhaps it's simply too good to be true. Thus, you wish to design a polynomial time algorithm that determines whether or not such a painting can exist.

Solution This problem can be viewed as a network flow problem in disguise. We begin by adjusting the count of the number of white squares in each row and column based on the location of the known (white) squares. The sum of the count in all rows must equal the sum of the count in all columns. Let this sum be X so we can infer that there are precisely X white squares among the squares of unknown colour. A common approach to such problems on a grid is to consider the bipartite graph where every row is a vertex on the left side of the graph and every column is a vertex on the right side, making 2N vertices in total. Every square in the grid which is of unknown colour forms a directed edge from its corresponding row to its corresponding column. We convert this graph into a flow network as follows. Each edge has capacity of 1. We add a source s and sink t to this graph. We add a directed edge from s to each row with capacity equal to the adjusted number of white squares in that row. Similarly, we add a directed edge from each column to t with capacity equal to the adjusted number of white squares in that column. It is evident that the saturated edges of the bipartite graph in any integer valued flow from the source to the sink describes a possible colouring of the grid. Any such flow has capacity at most X so a painting exists if and only if the maximum flow is equal to X.