Assignment 4 Solutions

COMP3121/9101 21T3

Released November 3, due November 17

This document gives model solutions to the assignment problems. Note that alternative solutions may exist.

1. A school is doing a fire drill. The school consists of n rooms and m corridors ($m \geq n-1$), where each corridor connects two rooms. There are x students who must be moved from room 1 (their classroom, with teacher A) to room n (a safe room, where teacher B is waiting).

   The teachers decided to divide the class of x students into several waves. Each wave will be released from room 1 by teacher A, and make their way through the corridors. To prevent overcrowding, each corridor has a limit l_i, which is the maximum number of students in a single wave who can use this corridor. Once all students in this wave have reached room n, teacher B will call teacher A and instruct them to send the next wave.

   Design a polynomial time algorithm which determines the minimum number of waves that must be formed and the number of students in the largest wave.

   Construct a flow network with n vertices representing the rooms. The ith corridor is represented by a pair of directed edges between the two rooms it connects, both with capacity l_i. We omit edges to vertex 1 and edges from vertex n, then denote vertex 1 as the source and vertex n as the sink. Using the Edmonds-Karp algorithm, we find a maximum flow, and let its value be y. Note that the amount of flow through each edge i is at most the capacity l_i, so any valid flow obeys the only constraint: no more than l_i students from each wave can use corridor i.

   Therefore y is the largest possible size of a wave, so the number of waves is minimised by sending y students at a time. The minimum number of waves will then be $\lceil x/y \rceil$, and the largest wave has $\min(x, y)$ students.

   Constructing the graph takes $O(n+m)$ time and running the Edmonds-Karp algorithm takes $O(nm^2)$ time, so the overall time complexity is $O(nm^2)$. Since $n \leq m+1$ and the input is of length at least m, this is a polynomial time algorithm.

2. (20 points) There are some stones in a rectangular grid with r rows and c columns. The stone in row i and column j has height h_{ij} and holds a_{ij} lizards. Both h_{ij} and a_{ij} are non-negative integers. If h_{ij} is zero, this denotes that there is no stone at (i, j) and hence a_{ij} is guaranteed to also be zero.

   Each lizard can jump between two stones if they are separated a distance of at most d, i.e. it can jump from a stone at (i_1, j_1) to a stone at (i_2, j_2) (which may be occupied by any number of lizards) if

   $$\sqrt{(i_1 - i_2)^2 + (j_1 - j_2)^2} \leq d.$$

   However, the stones are not stable, so whenever a lizard leaves a stone, the height of the stone is decreased by 1. If the new height of the stone is zero, the stone has disappeared below the surface. Any remaining lizards on this stone will drown, and lizards will no longer be able to jump onto this stone.

   We want to help as many lizards as possible to escape the grid. A lizard escapes if a jump of distance d takes them beyond the boundary of the grid.

   Design a polynomial time algorithm to find the maximum number of lizards that can escape from the grid.

   Construct a flow network with a (super-)source s, a (super-)sink t, and for each stone (i, j), two vertices $in_{i,j}$ and $out_{i,j}$.

   - For each stone (i, j), place an edge from the source to $in_{i,j}$ with capacity $a_{i,j}$, representing the lizards starting at this stone.
   - For each stone (i, j), place an edge from $in_{i,j}$ to $out_{i,j}$ with capacity $h_{i,j}$, representing the capacity of this stone (i.e. the number of lizards which can depart it).
   - For each pair of distinct stones (i, j) and (i', j') (of which there are $O((rc)^2)$), place an edge of infinite capacity from $out_{i,j}$ to $in_{i,j}$, representing any number of lizards moving from (i, j) to (i', j').
   - For each stone (i, j) within d of the boundary (i.e. $i \leq d$, $(r + 1) - i \leq d$, $j \leq d$ or $(c+1)-j \leq d$), place an edge of infinite capacity from $out_{i,j}$ to t, representing any number of lizards escaping from this stone.

   Each unit of flow in this network represents one of the starting lizards jumping between stones without drowning and eventually escaping the grid, so the maximum number of lizards escaping is the size of the maximum flow. We therefore find the answer by running the Edmonds-Karp algorithm on this flow network.

   There are $2rc + 2 = O(rc)$ vertices and $rc + rc + O((rc)^2) + O((rc)^2) = O((rc)^2)$ edges in this flow network. Therefore constructing the graph takes $O((rc)^2)$ time, and running Edmonds-Karp takes $O((rc)^5)$ time. The length of the input is at least rc, so the algorithm runs in polynomial time.

3. (20 points) You are the head of n spies, who are all wandering in a city. On one day you received a secret message that the bad guys in this city are going to arrest all your spies, so you'll have to arrange for your spies to run away and hide in strongholds.

   You have T minutes before the bad guys arrive. Your n spies are currently located at

   $$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

   and your m strongholds are located at

   $$(a_1, b_1), (a_2, b_2), \ldots, (a_m, b_m).$$

   The ith spy can move v_i units per minute, and each stronghold can hold only one spy.

   Design a polynomial time algorithm which determines which spies should be sent to which strongholds so that you have the maximum number of spies hiding from the bad guys.

   First, for each spy i check which strongholds j are reachable in T minutes. Each of mn pairs can be checked in constant time by directly comparing the distance between spy i and stronghold j to v_iT, the distance that spy i can travel.

   Then, we observe that matching spies with strongholds they can go to is a problem of finding the maximum bipartite matching, since no two spies can go to the same stronghold. We therefore construct a flow graph, with each spy i represented by a vertex p_i and each stronghold j represented by a vertex q_j, as well as a source s and a sink t. We place edges of capacity 1 from s to each p_i, and edges of capacity 1 from each q_j to t, and finally for each pair (i, j) such that spy i can reach stronghold j, we place an edge of capacity i from p_i to q_j. Running the Ford-Fulkerson algorithm on this graph finds a maximum flow, and hence the size of the maximum bipartite matching. We inspect the flow function found by this algorithm to recover the actual maximum matching – for each edge (p_i, q_j) carrying flow, we send spy i to stronghold j.

   There are $n + m$ total nodes and up to $nm + n + m = O(nm)$ edges, so constructing the graph takes $O(nm)$ time. Further, the value of a maximum flow is bounded above by $\min(n, m)$, so finding the maximum flow and hence the maximum matching takes $O(nm \cdot \min(n, m))$ time. The input is of size at least $n + m$, so the algorithm runs in polynomial time.

4. (20 points) Alice is the manager of a café which supplies n different kinds of drink and m different kinds of dessert.

   One day the materials are in short supply, so she can only make a_i cups of each drink type i and b_j servings of each dessert type j.

   On this day, k customers come to the café and the ith of them has p_i favourite drinks $(c_{i,1}, c_{i,2} \ldots, c_{i,p_i})$ and q_i favourite desserts $(d_{i,1}, d_{i,2}, \ldots, d_{i,q_i})$. Each customer wants to order one cup of any one of their favourite drinks and one serving of any one of their favourite desserts. If Alice refuses to serve them, or if all their favourite drinks or all their favourite desserts are unavailable, the customer will instead leave the café and provide a poor rating.

   Alice wants to save the restaurant's rating. From her extensive experience with these k customers, she has listed out the favourite drinks and desserts of each customer, and she wants your help to decide which customers' orders should be fulfilled.

   Design a polynomial time algorithm which determines the smallest possible number of poor ratings that Alice can receive.

   A solution for the case where all p_i and all q_i are 1 (i.e. all customers have only one favourite drink and one favourite dessert) will earn up to 10 points.

   For the full problem, construct a flow graph with:

   - Two vertices, S and T, for the source and the sink.
   - A vertex A_i for each drink i, with an edge of capacity a_i from S to A_i, to restrict the number of available cups of this drink.
   - A vertex B_j for each dessert j, with an edge of capacity b_j from B_j to T, to restrict the number of available servings of this dessert.
   - Two vertices C_i and D_i for each customer, with an edge of capacity 1 from C_i to D_i to ensure that each customer either has both their drink and dessert, or has neither. Note that we ignore serving them only one, as that is equivalent to serving them nothing in terms of ratings.
   - For each favourite drink $c_{i,j}$ of customer i, an edge of capacity 1 from $A_{c_{i,j}}$ to C_i for any drink they would accept.
   - For each favourite dessert $d_{i,j}$ of customer i, an edge of capacity 1 from D_i to $B_{c_{i,j}}$ for any dessert they would accept.

   Each unit of flow through this graph assigns a different customer one of their favourite drinks and one of their favourite desserts. Running the Edmonds-Karp algorithm on this graph then gives us the maximum flow, i.e. the maximum number of customers that we can satisfy, and so k minus this value is the minimum number of poor ratings.

   Our flow graph has $V = 2 + n + m + 2k$ vertices and $E = n + m + k + \sum_{i=1}^{k}(p_i + q_i) \leq n + m + k + (n + m)k$ edges. Both of these are polynomial in terms of the size of the input (which is at least $n + m + k$), so the complete algorithm of constructing the graph in $O(V + E)$ then running Edmonds-Karp in $O(VE^2)$ also runs in polynomial time.

   For the restricted version of the problem with $p_i = q_i = 1$ worth 10 points, we can use maximum flow in the same way but with a simpler flow graph. Construct a flow graph with a vertex A_i for each drink, a vertex B_j for each dessert, then two extra vertices for a source S and a sink T. For each drink i, add an edge with capacity a_i from S to A_i. For each dessert j, add an edge with capacity b_j from B_j to T. Finally, for each customer, add an edge of capacity 1 from $c_{i,1}$ to $d_{i,1}$. Again, the answer is k minus the maximum flow, found using Edmonds-Karp.

5. (20 points) You will be in charge of a delivery network consisting of n warehouses for d days. During this time, your job is to redistribute items between these warehouses – specifically, the ith warehouse starts with A_i items and must have at least B_i items by the end of the dth day. All items are identical, so this requirement can be fulfilled using items from any warehouse.

   You are also given a schedule of m deliveries ($m \geq d$) between warehouses that you can use to redistribute items. The kth delivery leaves warehouse w_k on the evening of day t_k, carrying at most c_k items, and drops them all off at warehouse w'_k on the morning of day t'_k. You can also keep an unlimited number of items at each warehouse overnight.

   Design a polynomial time algorithm which determines whether it is possible to have at least B_i items present at each warehouse i at the end of the dth day.

   Construct a flow network with:

   - a source S and a sink T.
   - a further dn vertices, each representing a warehouse i on day t, denoted as (t, i).
   - for each warehouse i, an edge from S to $(1, i)$ with capacity A_i, representing items starting at warehouse i.
   - for each warehouse i, an edge from (d, i) to T with capacity B_i, representing items finishing at warehouse i.
   - for each warehouse i and each day $1 \leq t < d$, an edge from (t, i) to $(t + 1, i)$ with infinite capacity, representing items left at warehouse i overnight from day t to day $t + 1$.
   - for each delivery k, an edge from (t_k, w_k) to (t'_k, w'_k) with capacity c_k, representing items carried by delivery k.

   Each unit of flow in this graph represents the locations of an item over the course of days 1 to d, ensuring that each delivery carries at most c_k items.

   If the value of a maximum flow in this network equals $\sum_i B_i$, then we determine that it is possible to have at least B_i items present at each warehouse i at the end of the d-th day, so the answer is YES. Otherwise the answer is NO.

   There are $dn + 2 = O(mn)$ vertices and $n + n + (d - 1)n + m = O(mn)$ edges, so it takes $O(mn)$ time to construct the graph and $O((mn)^3)$ time to run the Edmonds-Karp algorithm. The length of the input is at least $n + m$, so the algorithm runs in polynomial time.