Assignment 4 - Hints and Clarifications

COMP3121/9101 21T3

Released November 3, due November 17

This document provides some hints to help you solve the problems in Assignment 4. You are *not* required to follow these hints, and there may be alternate solutions which are equally correct.

Also included are the clarifications listed in the Assignment 4 FAQ on the Ed forum. Further clarifications may be added after this document is released.

General clarifications:

- **What is the time complexity of the Ford-Fulkerson algorithm? What about the Edmonds-Karp algorithm?**

  In a flow network, you can safely assume that $|V| \leq |E| + 1$. Otherwise, there are vertices other than the source which don't have any incoming edges, or vertices other than the sink which don't have any outgoing edges. No flow passes through such vertices, so they could be safely removed from the graph. We therefore simplify $O(|V|+|E|)$ (for example arising from a graph search such as breadth-first or depth-first search) to $O(|E|)$.

  The Ford-Fulkerson algorithm has worst case time complexity of $O(|E|f)$, where f is the value of a maximum flow. In general if there are m edges in the graph, each of capacity up to C, then f may be up to mC. However, the edge capacities are specified using only $m \log_2 C$ bits, so the Ford-Fulkerson algorithm does not run in polynomial time in general.

  However, in some circumstances the maximum flow is bounded by some smaller quantity. For example, in the standard flow graph constructed to represent a maximum bipartite matching problem, the maximum flow cannot exceed the number of vertices of the original bipartite graph, so the time complexity of the Ford-Fulkerson algorithm is $O(|E||V|)$.

  The Edmonds-Karp algorithm is a specialisation of Ford-Fulkerson, so its complexity is also bounded by $O(|E|f)$. One can also prove that it finds $O(|V||E|)$ augmenting paths, each in $O(|E|)$ time using breadth-first search, so an alternative bound for its time complexity is $O(|V||E|^2)$. Therefore the time complexity can be written as $O(\min(|V||E|^2, |E|f))$. In general, the first term of this minimum is a more useful constraint, but in some particular cases such as maximum bipartite matching, the second term dominates.

	General max flow	Max bipartite matching
Ford-Fulkerson	$O(\|E\|f)$	$O(\|E\|\|V\|)$
Edmonds-Karp	$O(\min(\|V\|\|E\|^2, \|E\|f))$	$O(\|E\|\|V\|)$

- **Do I need to prove the correctness of my chosen max flow algorithm?**

  No.

- **Do I need to account for the time complexity of constructing a flow network?**

  Technically yes, but the construction time will most likely be eclipsed by the runtime of a max flow algorithm anyway.

- **When constructing a flow network, is a worded explanation sufficient to describe the construction?**

  Yes, but if you feel that a diagram will aid in your explanation then you can provide it (and will make your marker's job easier).

1. (20 points) 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.

   Clarifications:

   - **What is the input?**

     The number of rooms n, the number of students x, the number of corridors m, and for each corridor i:

     – the room numbers of the two rooms it joins, and
     – the limit l_i of this corridor.

     You can assume that $m \geq n - 1$.

   - **Do the students in a wave have to remain together as a group while making their way from room 1 to room n?**

     No. They can disperse immediately; even when leaving room 1, they do not all have to take the same corridor.

   - **Are the rooms and corridors arranged in a straight line, with a corridor from room 1 to room 2, another from room 2 to room 3, and so on?**

     No. In particular, a room may have more than two corridors attached.

   Hint: think one wave at a time - how many students can be part of the first wave?

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.

   Clarifications:

   - **What is the input?**

     The number of rows r and columns c, the maximum jump distance d, and for each pair (i, j):

     – the initial height h_{ij} of this stone, and
     – the initial number of lizards a_{ij} on this stone.

   - **Does a lizard have to be along the edge of the grid to jump out?**

     No. They can jump out directly if they are within d distance of a lattice point outside of the grid.

   - **Can a stone (i, j) hold more than a_{ij} lizards in the future?**

     Yes. a_{ij} is the *initial* number of lizards on that stone. Stones do not have any 'lizard capacity' that cannot be exceeded.

   Hint: First solve the case where initially there is only one lizard. Then generalise to allow several lizards all starting on the same stone, and finally to the full problem. Note that each initial height h_{ij} provides a constraint on the number of lizards which can use (i.e. jump from) that stone.

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.

   Clarifications:

   - **What is the input?**
     The number of spies n, the number of strongholds m, the coordinates of each spy and each stronghold, the speed v_i of each spy and the time T.
   - **What is the output?**
     You should provide a maximal assignment of spies to strongholds, or equivalently a procedure to generate this assignment.

   Hint: Start by working out for each spy which strongholds they can reach within T minutes. Create a graph to represent these spy-stronghold pairs.

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 $p_i = q_i = 1$ for all i (i.e. all customers have only one favourite drink and one favourite dessert) will earn up to 10 points.

   Clarifications:

   - **How has the question changed?**

     Each customer may now have several favourite drinks and favourite desserts. The 10 point subtask is the original problem.

   - **What is the input?**

     The number of types of drink n, and for each drink type i, the number of cups a_i that can be made.

     The number of types of dessert m, and for each dessert type j, the number of servings b_j that can be made.

     The number of customers k, and for each customer i:

     - their number of favourite drinks p_i,
     - their favourite drinks $c_{i,1}, \ldots, c_{i,p_i}$
     - their number of favourite desserts q_i
     - their favourite desserts $d_{i,1}, \ldots, d_{i,q_i}$

   Hint: For the $p_i = q_i = 1$ case, we need to make pairs of drinks and desserts, where each pair satisfies a customer without exceeding the limit on each drink type and dessert type.

   The most straightforward generalisation of the subtask solution considers pairs (drink, dessert) where the drink is a favourite drink of customer i and the dessert is a favourite dessert of customer i. However, there can be many such pairs for each customer, and we need to be careful to avoid assigning several drinks and several desserts to a single customer. How can you enforce that each customer is only given one of their favourite drinks and one of their favourite desserts?

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.

   Clarifications:

   - **What is the input?**

     The number of warehouses n, days d and deliveries m.

     For each delivery i:

     - the integer A_i (initial number of items at warehouse i) and
     - the integer B_i (final number of items required at warehouse i).

     For each delivery k:

     - w_k (ID of departure warehouse)
     - w'_k (ID of arrival warehouse)
     - t_k (day of departure)
     - t'_k (day of arrival)
     - c_k (max items carried by this delivery)

     You can assume that $m \geq d$. This doesn't change the problem (since a day with no deliveries has no impact), but it makes it quite a bit easier to prove that the intended algorithm runs in polynomial time.

   - **What is the output?**

     Either true or false. Your algorithm should determine whether all quotas B_i can be satisfied together.

   - **Can there be multiple deliveries from warehouse i to warehouse j leaving on the same day?**

     Yes.

   Hint: A path in the flow network should represent the status of an item across the d days. However, simply keeping track of the location of the item is insufficient, as this doesn't allow us to construct appropriate edges in the flow network. What else should be encoded in the vertices?

   Try first solving for the case $d = 2$. In this case, all deliveries depart on evening 1 and arrive on morning 2.