

---

50 marks

CS 601 Quiz 1

8:15-9:25, 18/8/10

You have about one minute per mark. Keep this in mind while deciding how long you need to write. Crisp answers will receive more credit. You need not repeat anything proved in the book or in class (just state as much), unless the question explicitly asks exactly that.

---

**Problem 1:** Consider a network with nodes numbered 1 to  $n$ . Suppose 1,  $n$  are  $s, t$  respectively. There are bidirected edges of capacity 1 between  $i$  to  $i + 1$  for  $1 \leq i < n$ . Further are bidirected edges between 2 to  $j$  of capacity 1 for all  $4 \leq j < n$ .

(a)[5 marks] In this network, how long will it take for the Ford-Fulkerson algorithm to find the max flow in the worst case?

$O(n)$  time.

(b)[5 marks] In this network, how long will it take for the Goldberg-Tarjan algorithm to find the max flow in the *best* case?

$O(n)$  time,  $O(1)$  pushes – you can say any of these two things.

(c)[10 marks] Show that the Goldberg-Tarjan algorithm could take  $\Omega(n^2)$  pushes in the worst case. Explain briefly. Don't just start writing, but think how you can make your point briefly. For example, you could show the state of the program after a carefully chosen number of steps. Whatever you do, try to be brief and clear.

The flow would be pushed on  $(2, n - 1)$  in the first step. Then backwards towards 2, for a total of  $1 + (n - 1 - 2) = n - 2$  pushes. Then it might be pushed to  $n - 2$  and so on. So this unit flow would circulate and take  $(n - 2) + (n - 3) + \dots = \Omega(n^2)$  pushes.

**Problem 2:**[20 marks] The Indian Amateur League has decided to form cricket teams in various towns in India. It has been decided that all players of a team need not be from the town associated with the team; it is sufficient if at least 8 are from that town. Each team must have between 15 and 20 players. You are to write a program that takes as input a list of teams with their hometowns, and a list of players with their hometowns, and a list for each player giving the teams he would like to play for. In addition, it is given whether each player is junior or senior. You are also required to ensure that overall at least 100 juniors are selected in some team or another.

Formulate this as a maxflow problem of appropriate kind (state which). Give a picture. Characterize the time required to solve the problem as a function of the input size.

Unless mentioned, edges have capacity infinite, lower bound 0. Vertex demands are also 0 by default.

level 1:  $s$

level 2:  $J, S$  for junior/senior. Edge from  $s$  to  $J$  of lower bound 100. Edge from  $s$  to  $S$ .

level 3: one vertex per player. Edges from  $J$  to each junior player with capacity 1. Edge from  $S$  to each senior player, with capacity 1.

level 4: two vertices per team  $t$ .  $t_1$  for the hometown players,  $t_2$  for the outstation players. An edge from each player to all the teams that he wants to play for. The edge is to  $t_1$  if the player has the same hometown as the team, to  $t_2$  if the player has different hometown.

level 5: one vertex  $t$  per team  $t$ . Edge from  $t_1$  to  $t$  with lower bound 8. Edge from  $t_2$  to  $t$ .

Lever 6: vertex  $t$ . Edge from each team vertex to  $t$  with lower bound 15 and capacity 20.

Need a feasible flow for this network.

Time: Using Ford-Fulkerson:  $O(FE)$ .  $F = O(\text{number of teams})$ .  $E = \text{number of preferences that the players have, summed over all players}$ .

---

Grading scheme: correctly uses player preferences to allocate players to teams 3 marks, at least 15 to each team at most 20, 3 marks. Ensures the hometown constraint 5. Ensures the junior/senior constraint 5. State that you need a feasible flow: 1 marks. Estimate time: 3 marks as a function of the input with 2 marks to estimate  $F$ .

**Problem 3:**[10 marks] Remember that in the image relabelling problem you were given an  $n \times n$  binary image  $Q$  and were supposed to generate another  $n \times n$  binary image  $R$  which minimizes a penalty function:  $P(Q, R) = x * \text{number of pixels having different values in } Q, R + y * \text{number of adjacent pairs of pixels in } R \text{ having different values}$ . Say  $x = 1, y = 0.6$ .

Consider the following algorithm for solving this problem:

1.  $R = Q$
2. Let  $p$  be a pixel in  $R$ . Complement the pixel (i.e. 0 goes to 1 or 1 goes to 0). If the penalty reduces, then retain the new value, else leave the pixel at the original value.
3. Repeat the previous step while the penalty can be reduced.

Give as simple an example as possible for which this algorithm does not produce an  $R$  of minimum possible penalty. You will get more marks, the simpler your solution is.

A pattern such as the one below will stay unchanged, while mincut will turn the 0s to 1s, for a penalty drop of  $1.2 * 4 - 4 = 0.8$  This gets 10 marks.

```
1111
1001
1001
1111
```

There are other patterns for which a bad final answer can be reached, but need not always be reached, e.g.

```
101
010
101
```

In this if the 1s are considered first then the penalty drop is  $1.2 * 4 + 2.4 - 5 = 2.2$ , whereas if the 0s are considered first it is  $1.8 * 4 - 4 = 3.2$ . Other results are also possible. On the other hand, the mincut algorithm will pick the best, i.e. penalty drop of 3.2. In this case you get 8 marks.