CS 170: Problem Set 4

Due: September 23, 2011, 4:00 p.m.

Instructions: Turn this in to the class homework boxes in 283 Soda by 4:00 p.m. on Friday, September 23, 2011. Please begin your answer to each question on a new sheet of paper and make sure each sheet is labeled with your name, SID, section number, GSI name, the assignment number, the question number, and "CS 170 – Fall 2011."

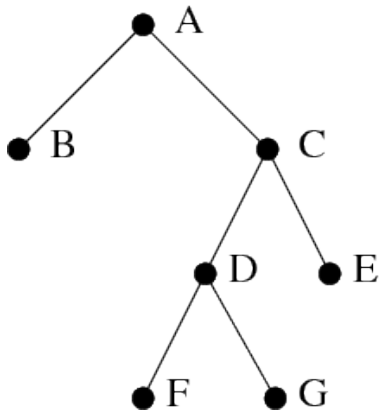
Because each problem will be graded by a different reader, please **turn in each question in a different box in 283 Soda.** Question i goes in the box labeled "CS 170 — i."

Please read the class webpage for rules regarding collaboration (encouraged!) and cheating (forbidden!) on homework.

DPV = Dasgupta, Papadimitriou, and Vazirani.

1. **Tree-Splitting.** Design an algorithm to answer the following query: given a tree T with n nodes and an integer k, find an edge (u, v) in the tree such that deleting this edge splits the tree into two connected components with k and $n - k$ nodes. If no such edge exists, your algorithm should report this fact. Your algorithm should run in $O(n)$ time.

   Example. Consider the following tree T:

   If $k = 4$, we could delete the edge (C, D) to split T into one connected component with 4 nodes (namely, A, B, C, and E) and one connected component with 3 nodes (namely, D, F, and G).

   If $k = 5$, we can delete the edge (A, C) to split T into one connected component with 5 nodes (namely, C, D, E, F, and G) and one connected component with 2 nodes (namely, A, and B).

2. DPV 3.2

3. DPV 3.3

4. DPV 3.4 (i)

5. DPV 3.8

6. DPV 3.12

7. DPV 3.26

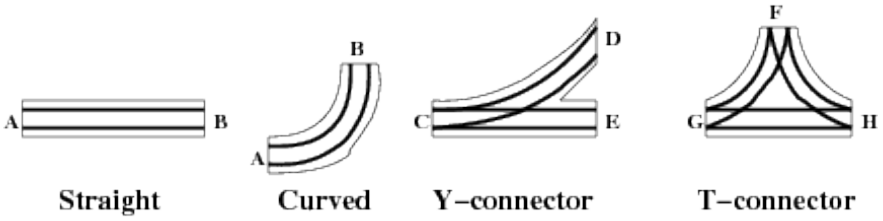
8. **Toy Train.** A precocious 4-year-old M likes to play with her toy trains. She has the following 4 kinds of pieces to compose a track:

   (i) straight pieces (one end is labeled A and the other B),

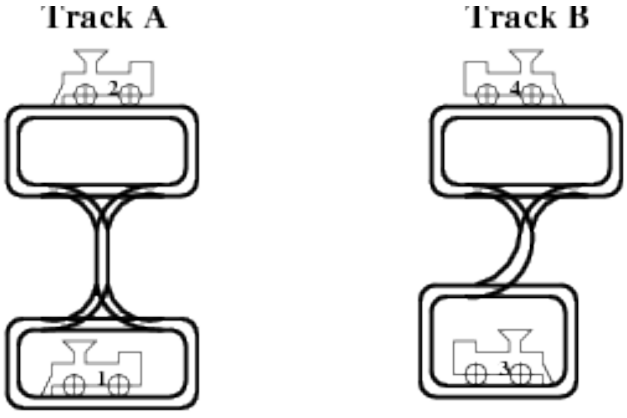
   (ii) curved pieces (one end is labeled A and the other B),

   (iii) Y-pieces, where if you enter at C, you can exit either at D or E, and if you enter at D or E, you must exit at C

   (iv) T-pieces, where no matter where you enter (say F), you can exit at either of the other two places (G or H)

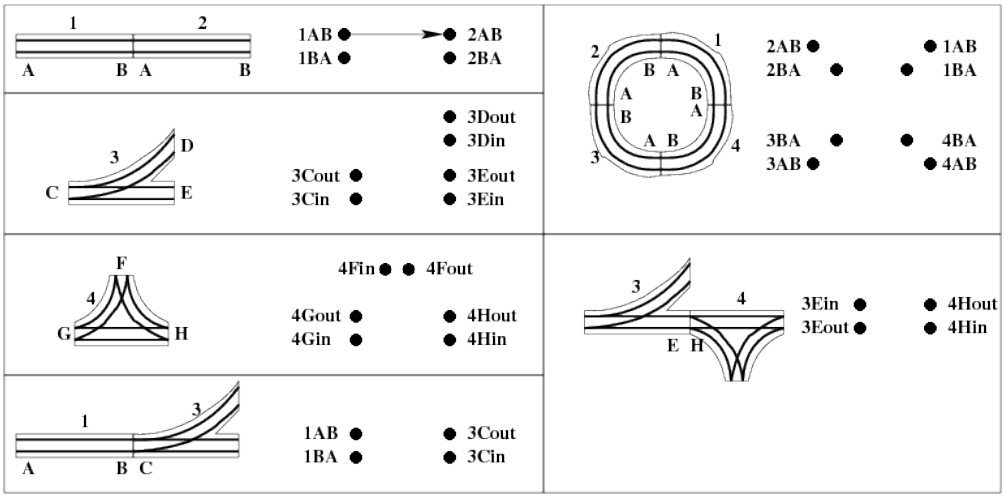


After building lots of different train tracks, M realizes not all are equally fun to play with: On some tracks you can keep moving the train forward as long as you like without running out of track, and on others you can't. On some you can get to any track piece from any other track piece by moving the train in its forward direction. On some you can even get the train to face in either direction on any track piece just by moving forward (see Track A below, where Train 1 can get to the same position as Train 2). In some other tracks you can't even reach some track pieces depending on where you start (see Track B below, where Train 3 stays on the lower track as long as it moves forward).



   (a) Given a train track, we will define a graph G that describes the train track, and how trains can move if they are only allowed to move forward. There will be a pair of vertices to represent each straight or curved piece: one vertex representing a train moving forward from A to B, and the other vertex representing a train moving forward from B to A. (Label these vertices for piece i vertices iAB and iBA, respectively.) Similarly, there will be a pair of vertices for each of the 3 connectors of a Y- or T-piece. For example, two of the vertices for a T-piece j are labeled $jFin$ and $jFout$,

for trains moving toward the T-piece at F and moving away from the connector at F, respectively. There will be edges representing possible traffic on the track both between and within pieces. Show how to define these graphs by filling in the missing edges in the figures below. Note that vertices are missing in the figure, you can omit all edges to these vertices. For illustration, one edge has already been added for you: the edge from 1AB to 2AB means that a train on piece 1 facing from A to B can move forward to piece 2, facing from A to B.



(b) If there are a total of s straight or curved pieces and c Y- or T-pieces, give the exact number of vertices n and an upper bound on the number of edges e (in terms of s and c) in your graph G.

(c) Let x, y be chosen straight or curved track pieces. Describe an efficient algorithm to determine whether M can put a train on x (facing in a specific direction) and move it forward to eventually reach y (facing in a specific direction). For example, Train 3 on Track B above cannot reach the position of Train 4 above moving forward only, but Train 4 can reach the position of Train 3. Similarly, Train 1 on Track A can reach the position of Train 2. Give the complexity (using $O()$) in terms of the total number s of straight or curved pieces and the total number c of Y- or T-pieces. Justify your answer.

(d) Describe an efficient algorithm for figuring out whether M can put a train on a chosen straight or curved track piece (facing in a chosen direction) and then play as long as she wants, i.e. keep moving the train forward without running out of track, provided she makes the correct turns at connectors. Give the complexity (using $O()$) in terms of the total number s of straight or curved pieces and the total number c of Y- or T-pieces. Justify your answer.

(e) Describe an efficient algorithm for answering the last question when M plays with her eyes closed after putting the train down on a chosen track piece facing in a chosen direction, i.e. may take any possible turn at any connector. Is she guaranteed to be able to keep the train moving forward as long as she likes without running out of track, regardless of what turns she makes at connectors? Give the complexity (using $O()$) in terms of the total number s of straight or curved pieces and the total number c of Y- or T-connectors. Justify your answer.

(f) M's little brother N shows up and starts playing, but moves the train either forward or backward as he pleases. Answer question (c) above again for this different situation.