Exam 3 (120 minutes)
CS 330 Discrete Structures
Summer Semester, 2016

Name:
A-ID:

• Only a pen and whatever provided by the instructor are permitted throughout the exam.
• You have to show your work. You will not get partial credits if the grader cannot figure out how you arrived at the answer.

<table>
<thead>
<tr>
<th>Question</th>
<th>Maximum Points</th>
<th>Earned Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
1. The PERT diagram for ‘Wearing clothes for CS330 lecture’ is given below.

Do the topological sorting for this PERT diagram and explain how you achieved the order.

One of the possible DFS traversals is:

Then, the linear order is the sorted list by the decreasing order of the finish time:
2. Find the distances of all shortest paths from the top vertex to everyone else in the following graph.
After applying the Dijkstra’s algorithm, we can get:
3. (a) Form a binary search tree (BST) from the words of the sentence: “This test is not so difficult”, using alphabetical order. Insert the words in the order they appear in the sentence.

(b) What is the printed sentence if the above BST is traversed by

- Preorder?:
  
  this test is difficult not so

- Inorder?:
  
  difficult is not so test this

- Postorder?:
  
  difficult so not is test this
4. Find the minimum spanning tree in the following graph.

(Hint: before you start, think about which algorithm will be easier to manage in this graph.)

Either Kruskal’s or Prim’s algorithm will work, but since there are too many edges in the graph,
Kruskal’s algorithm may be hard to manage by hand. After applying the Prim’s algorithm, we can get the following MST (highlighted).
5. Prove that, in a simple connected graph, all of its Minimum Spanning Trees (MSTs) must contain the edge with minimum weight.

(Hint: prove this by contradiction)

Assume that there exists an MST \( T \) that does not contain the edge with minimum weight (which is denoted as \( e_{\text{min}} \)). If \( e_{\text{min}} \) is added to \( T \), a cycle must have been formed because of \( e_{\text{min}} \). In that cycle, if we remove any other edge, the cycle disappears and we get another spanning tree \( T' \). Since \( e_{\text{min}} \) is the one with the minimum weight, the weight of the removed edge must be larger than \( e_{\text{min}} \) (recall that \( T \) didn’t have any edge with the minimum weight in the first place). That means the total weight of \( T' \) must be smaller than that of \( T \) since a larger edge is removed and a smaller edge is added. This contradicts the assumption that \( T \) was an MST, which completes the proof.

6. Consider the following finite state machine in which state \( A \) is the starting state and state \( B \) (shaded) is the only accepting state:

![Finite State Machine Diagram]

(a) What is the number of 0s and 1s in the language recognized by the above Finite State Machine (FSM)?

The number of 1s is unlimited, and the number of 0s is \( 1 \mod 3 \).

(b) Construct a regular expression for that language.

\[ 1^*0(1^*01^*01^*01^*)^* \]

7. Construct Finite State Machines recognizing the following languages.

(a) The set of bit strings that do not contain two consecutive 0s.

![Finite State Machine Diagram for non-consecutive 0s]

State 1 denotes that no consecutive 0s so far and current bit is 1. State 2 denotes that no consecutive 0s so far and current bit is 0. These are the only accept states.
(b) The set of bit strings that contain even number of 1s and even number of 0s.

A state ‘x, y’ denotes that the number of 0s and 1s is $x \mod 2$ and $y \mod 2$ respectively. The gray state is the only accept state.

8. Prove that the set $\{0^n + 21^{n-2} | n = 0, 1, 2, \ldots \}$ is not a regular language.

We prove this by contradiction. Suppose the language is regular, and it is accepted by an FSM with $k$ states. Let’s consider the string $0^{k+2}1^{k-2}$ which should be accepted by that FSM. According to the pigeon hold theorem, at the time $0^k$ is read by the FSM, there has been $k$ state transitions and therefore at least one state in the FSM has been visited for twice, which forms a cycle in the FSM starting and end at this state. Then, we can insert as many 0s as possible before proceeding to the rest of the string (i.e., $0^21^{k-2}$) and moves to the accept state finally. But then the strings accepted by the FSM are much different from $\{0^n + 21^{n-2} | n = 1, 2, 3, \ldots \}$ which is a contradiction.