

1. You may not use notes or a calculator.
2. Tear off this sheet and use it to keep your answers covered at all times. If you need more scratch paper, I have some.
3. Write your name on the back of the test by the staple to facilitate returning the exams. Do not write your name on the front of the exam.
4. No question is meant to have a syntax error. If it does, fix it or ask me how I want the error resolved.
5. You are graded not only on the correct answer, but also the best answer.
6. Look at the number of points a problem is worth in deciding how much time to spend answering it. A question with five points probably doesn't warrant three pages of proof (unless you have finished answering everything else).
7. The amount of space provided is an indication of how much room *I* needed to answer the question.

The master method for determining recursive complexity:

$$T(n) = c \text{ if } n < d \\ aT(n/b) + O(n^k) \text{ if } n \geq d$$

$a > b^k$  then  $T(n)$  is  $O(n^{\log_b a})$

$a = b^k$  then  $T(n)$  is  $O(n^k \log n)$

$a < b^k$  then  $T(n)$  is  $O(n^k)$

## CS 5050 Spring 2005

### ***Fill in the blank (2 points each)***

1. \_\_\_\_\_ recursively solves an optimization problem by caching sub problem solutions rather than recomputing them.
2. A \_\_\_\_\_ algorithm is one that proceeds in a simple and obvious way, but may require a huge number of steps to complete.
3. A \_\_\_\_\_ algorithm repeatedly makes the choice that seems the best given local information.
4. A graph which does not have more than one edge between the same pair of vertices and contains no self loops is termed \_\_\_\_\_.
5. A \_\_\_\_\_ graph is a directed graph in which you can reach every node from every other node.

### ***Multiple Choice (3 points each)***

1. Which is true of a breadth first spanning tree on an undirected graph?
  - a. There are only discovery edges.
  - b. There are discovery edges and back edges
  - c. There are discovery edges and cross edges.
  - d. There are discovery, cross, forward, and back edges.
  - e. None of the above
2. Which of the following represents an example of coin denominations such that the greedy change making algorithm will not use the minimum number of coins:
  - a. 1 5 12 50 75
  - b. 1 7 8 21 35
  - c. 1 13 22 40
  - d. all of the above
  - e. none of the above
3. Recall, in the knapsack problem you have items' weights and values; you want best set of items for you knapsack within a specified weight limit. The algorithm is optimal under a greedy heuristic if
  - a. All weights are integer
  - b. All weights are positive
  - c. A fraction of an item can be used
  - d. All items have the same value per pound
  - e. None of the above
4. The most efficient way to find the median (element in which half the elements are higher and half lower) of an array of n numbers is
  - a. Sort the elements using quicksort and pick the middle.
  - b. Use a variant of quicksort, where you throw away half of the nodes after a partition.
  - c. Find the average of the numbers. Check to see if the average is also the median. If so, you are done. If not, guess another number and try again.
  - d. Use a max heap and deleteMax  $n/2$  times.

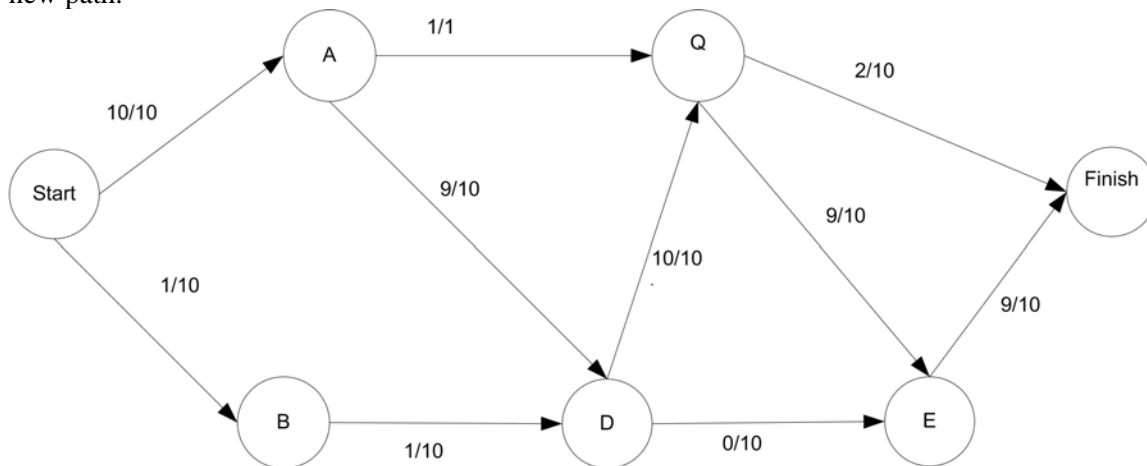
5. You need to find the **longest** directed path between two points. There are no cycles. Which method is most efficient:
- Use Dijkstra's algorithm for shortest paths, but always select the largest total distance rather than the smallest.
  - Consider nodes in topological order, recording the longest path.
  - Use a breadth first search.
  - Use a variant of Bellman-Ford
  - All of the above
6. Which defines a greedy algorithm
- An algorithm that always takes the best immediate, or local, solution while finding an answer.
  - An algorithm which is sub-optimal.
  - An algorithm which finds a globally optimal solution.
  - A brute force algorithm.
7. Dynamic programming techniques
- Can often be used to change an exponential algorithm into a polynomial algorithm.
  - Require the computation (and storage of results) of many sub problems
  - Are useful in that they eliminate the redundant calculations which are often a part of recursive solutions
  - All of the above
  - None of the above
8. What is the complexity of the algorithm below (assuming that  $n$  and  $m$  are approximately equal in value)?
- ```
int doit (int n, m)
{ if (n*m==0) return n;
  return max(doit(n-1,m), doit(n-1,m-1), doit(n,m-1)) +1;
}
```
- $O(n)$
  - $O(n \log n)$
  - $O(n^2)$
  - $O(2^n)$
9. What is the complexity of the algorithm below?
- ```
int doit(int n,int m)
{ System.out.println("Today");
  if (n <= m) return n+1;
  return doit(n/2,n%2);
}
```
- $O(\log n)$
  - $O(n)$
  - $O(n \log n)$
  - $O(n^2)$
10. What is true of the minimal cost spanning tree algorithm (for a connected, undirected graph) when there are negative edges present?
- There is no problem. The original algorithm handles this case without modification.
  - You must add a constant to each edge before solving the problem in the traditional manner. Just subtract the constant after you are finished.
  - The algorithm will not be able to give any answer.
  - In some cases, there is no such thing as a minimal cost spanning tree with negative edges.

11. We need to determine if a directed graph is cyclic. This can be accomplished most efficiently by
  - a. Doing a regular traversal. If you reach a node which has already been visited, the graph is cyclic.
  - b. Do a graph traversal keeping track of the nodes which are on the path (in the dfs tree) from the root to the node. If a node has an edge to anything that is already on the path, the graph is cyclic.
  - c. Perform a topological ordering. If it fails, the graph is cyclic.
  - d. none of the above will work.
12. We need to find the  $k^{\text{th}}$  largest element of an array containing  $n$  elements. This can be performed most efficiently by
  - a. Sorting the elements and selecting the  $k^{\text{th}}$  from the bottom.
  - b. Making  $k$  passes over the array, each time removing the biggest.
  - c. Partitioning the set as in quicksort, and recursively searching the appropriate portion.
  - d. None of the above.
13. I need to ensure that every house has electricity, but I want to minimize the cost of the wire used in connected them to the power plant. Which algorithm would I use?
  - a. All pairs shortest path.
  - b. Single source shortest path.
  - c. Hamiltonian tour.
  - d. Minimal cost spanning tree.
  - e. Topological ordering

**Short Answer**

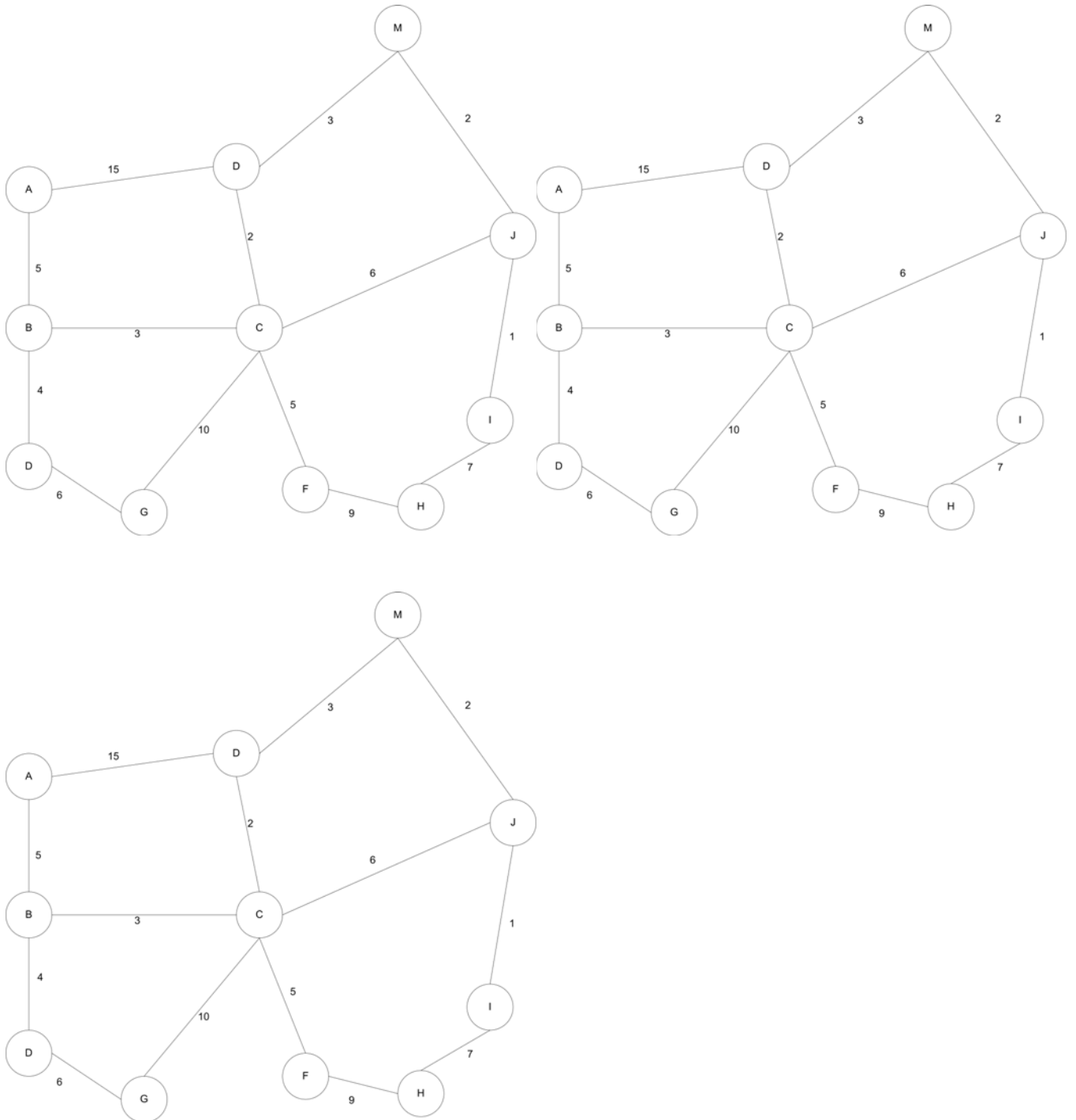
1. (15 points) Consider the network flow diagram below.

a. Show the augmenting path that would be chosen by an algorithm seeking to maximize the flow of the new path.



b. If the path from D to Q were suddenly reduced to half capacity, briefly explain how you could update the flow (on appropriate edges) without starting over.

2. (10 points) Consider the diagram below. Using Baruvka's algorithm for finding minimal cost spanning tree, show the selected edges after each iteration of the algorithm. **What is the complexity of Baruvka's algorithm?**



3. (16 points) You have an undirected graph stored as an adjacency list. Write the java-like code to determine if the graph is bipartite. **What is the complexity of your solution?** *Recall a bipartite graph is an undirected graph where vertices can be divided into two sets such that no edge connects vertices in the same set.*

```
Node [] nodes;
int nodeCt; // maximum subscript of nodes

class Node {
    int id; // the subscript in nodes[]
    int set; // initialized to 0
    EdgeList adj; // all edges touching Node in adjacency list form
}

class Edge {
    int from; // endpoint of start node in terms of subscript in nodes[]
    int to; // endpoint of end node in terms of subscript in nodes[]
}

class EdgeList{
    Edge edge; // edge
    EdgeList next; // pointer to rest of edge list
}
```

4. (10 points) Consider the following undirected graph. It has been labeled with a depth first spanning tree (in which back edges are shown as dotted lines) to facilitate computing articulation points.

a. Which are articulation points?

b. Label the graph with low when node 1 is the starting point.

