

CS 146 Final Exam 1

Name: _____

Grade:

The test will be open book, open notes, 2 hours and 15 minutes time limit. Calculators allowed for numerical, non-programmed calculations, but not computers. Please write your answers on the exam sheet. 17 problems, 4 points per problem, 68 points total. Partial credit given only when the problem has parts (a,b,c, etc.)

1. *Analysis of non-recursive algorithms.* Consider the problem of counting the number of points with integer coordinates (x,y) on the circle of radius R , where R is a positive integer.

(a) Write code to do this without using any floating-point arithmetic. Efficiency is not a concern here.

(b) Use Θ notation to estimate the running time of your code in part (a) in terms of R .

(c) With floating-point arithmetic allowed, write code that will be faster for large R .

(d) Use Θ notation to estimate the running time of your code in part (c) in terms of R :

2. Indicate, for each pair of expressions (A,B) in the table below, whether A is O , or Θ of B. Write “Yes” or “No” in each blank box of the table. [You will only receive points if you answer more than 50% correct, since you can get 50% correct by guessing.]

A	B	O	Θ
2^n	n^3		
\sqrt{n}	$n^5 + \sqrt{n}$		
\sqrt{n}	$\lg n^5$		
\sqrt{n}	$\lg n + \sqrt{n}$		
$n^5 + \sqrt{n}$	n^5		
n^{23}	$\sqrt{(n^{56})}$		
n^{23}	n^{28}		
$n \lg n$	n^2		
$n \lg n$	n		

3. *Linked lists.* The class *StringList* has these members:

```
{ String key;
  StringList next;
}
```

Write a method *void append(StringList x)* that appends *x* to the end of *this*. For example, if *this* is (“cat”, “dog”) and *x* is (“giraffe”, “anteater”), then after running *append*, *this* has the value (“cat”, “do”, “giraffe”, “anteater”).

4. *Analyzing recursive code using recurrence relations.*

A *recursive palindrome* is a string that it is a palindrome, and in addition both its left half and its right half (if they have more than one character) are recursive palindromes. The “left half” and “right half” of a string of odd length $2n + 1$ or of even length $2n$ are the first n characters and the last n characters. For example, WOWWOW and ABACABA are recursive palindromes but MADAM is not. For another example, WOWWOWXWOWWOW is a recursive palindrome. Assume only upper-case letters will be used.

We will consider two problems about recursive palindromes: One, testing a string of length n to see if it is or is not a recursive palindrome. The function to do this can be

called *IsRP*. The second problem is counting the number of recursive palindromes of length n . The function to do this can be called *CountRP*.

(a) Write a recurrence relation for the worst-case running time $T(n)$ of *IsRP*.

(b) Solve that recurrence relation to give an estimate for $T(n)$. (“Estimate” means that your answer uses big-O notation.) (Write only your answer)

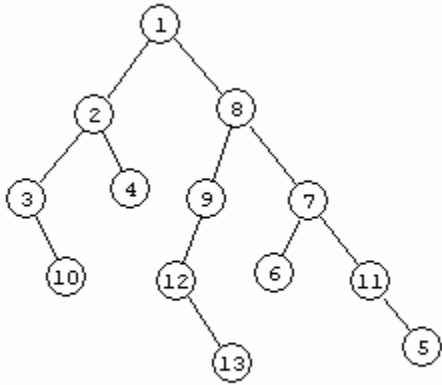
(c) Write a recurrence relation for the worst-case running time $T(n)$ of *CountRP*.

(b) Solve that recurrence relation to give an estimate for $T(n)$. (“Estimate” means that your answer uses big-O notation.) (Write only your answer)

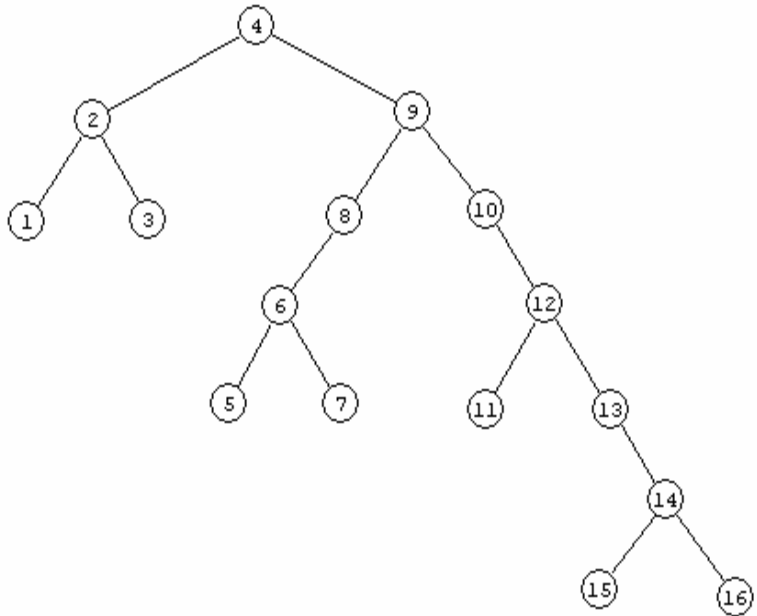
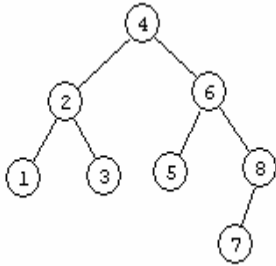
5. *Binary Search Trees.*

The picture represents a binary search tree. The numbers shown are arbitrary node labels, not numbers representing the contents of the nodes. **The contents are not shown.** If node 1 is deleted, using binary search tree deletion, what will be the new root node?

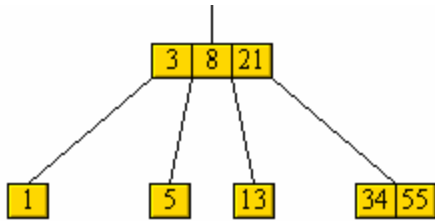
Answer _____ [It is not required to draw the tree after the deletion.]



6. *Red-black trees.* For each of the following two trees, either indicate how to color the nodes red and black (use R and B to label the nodes) to make the tree a red-black tree, or explain why that is not possible.



7. *B-tree insertion.* A new key 4 is to be inserted into the following B-tree. This B-tree has 2, 3, or 4 children per internal node.



Draw a similar diagram showing the B-tree after the insertion of 4. Use the one-pass insertion algorithm, i.e. the one given in your textbook

8. *Merge sort.* The following array is to be sorted:

4	6	8	1	10	5	15	13	2	16	14	7	3	9	11	12
---	---	---	---	----	---	----	----	---	----	----	---	---	---	----	----

(a) If *mergeSort* is called to sort this array, how many recursive calls will there be (not counting the top-level call)? ____ (*Remark: the textbook's mergeSort does get called on arrays of length 1.*)

(b) What will be the maximum recursion depth, i.e., the maximum number of nested calls that have not been returned from yet? ____ (*Here we DO count the top-level call.*)

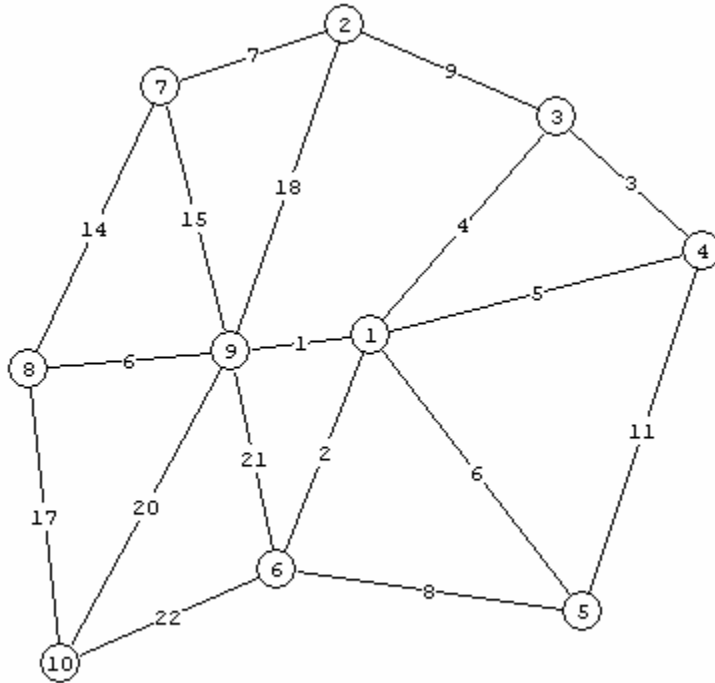
(c) What will the array look like immediately after the return from the first recursive call?

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

9. *Quick sort.* The following array is to be sorted (it's the same as the one above):

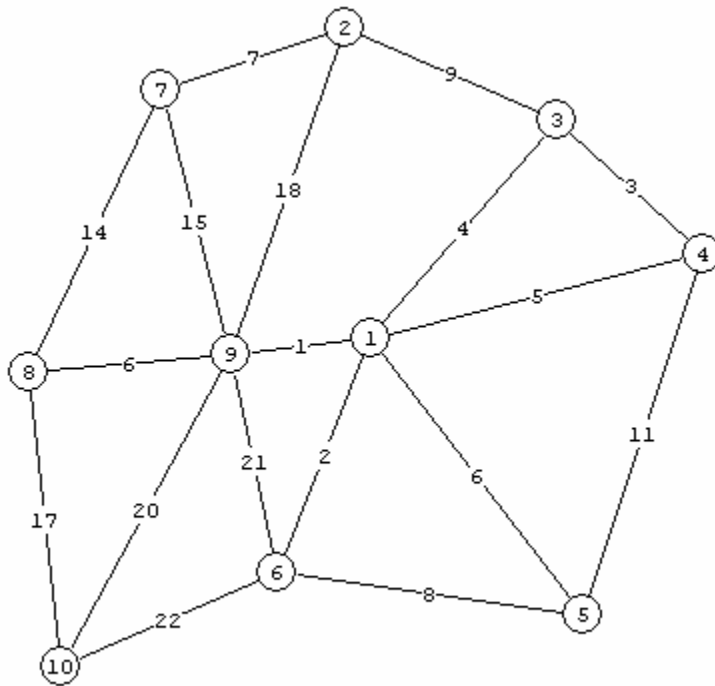
4	6	8	1	10	5	15	13	2	16	14	7	3	9	11	12
---	---	---	---	----	---	----	----	---	----	----	---	---	---	----	----

11. Execute Prim's algorithm to find a minimal spanning tree in the following graph, starting from vertex 2. List the vertices in the order they come off the queue, and then highlight the minimal spanning tree by drawing it right on the diagram.



Vertices in the order they come off the queue:

12. Execute Kruskal's algorithm to find a minimal spanning tree in the following graph (which is, incidentally, the same graph as in the previous problem).



List the edges in the order they are added to the answer, and also highlight the minimal spanning tree by drawing it right on the diagram. Since the edges have unique weights, you can identify an edge by its weight, e.g. you can write "11" instead of (4,5).

Edges:

13. If we run Dijkstra's algorithm on the graph shown below (which is the same as the graph in the previous problem, but here is another picture of it), starting from node 2,

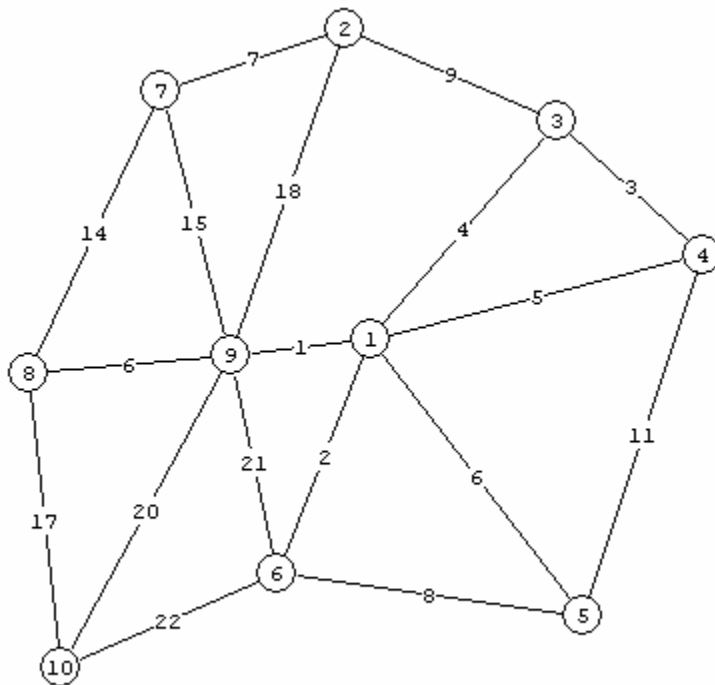
(a) how many nodes will be removed from the queue before every node gets its correct distance from node 2 calculated?

Answer [just one number] : _____

(b) List the nodes in the order of their removal from the queue.

Answer [a list of numbers]: _____

The diagram is provided for you to use if you wish, but it's not needed to communicate the answer.



14. Needleman-Wunsch.

(a) What are the asymptotic time and space requirements of the Needleman-Wunsch algorithm? Express your answers in terms of the maximum length n of the two input strings, using big-O notation.

Time:

Space:

(b) Execute the Needleman-Wunsch algorithm to find the best alignment of

COPE and CAP, scoring exact matches as 1 and non-matches as 0, with a gap penalty of 1 (i.e. matching a letter with a gap scores -1). Use the following table to execute the algorithm, showing the predecessor arrows as was done in class.

The best alignment is _____

and the score of that alignment is _____.

15. *Basic properties of some algorithms we studied*

(a) What two algorithms we studied used a priority queue?

_____ and _____

(b) Which algorithm has a recursive version, depth-first search or breadth-first search?

(c) Which search algorithm uses a stack?

(d) Which algorithm is used for finding, in one run, the shortest distance between all possible pairs of points in a weighted directed graph?

(e) Which algorithm can detect negative cycles in a graph?

16. *Basic properties of some data structures*

(a) What kind of tree that we studied has this property? any two leaf nodes have same depth. _____

(b) True or false: If A and B are the left and right children of some node of a heap, then we must have $A \leq B$. _____

(c) What kind of tree is used in the implementation of databases? _____

(d) True or false: in a binary search tree, the height of the tree is logarithmic in the number of nodes. _____

(e) What is the worst-case running time for the insertion algorithm for binary search trees? _____

(f) What is the worst-case running time for the insertion algorithm for red-black trees? _____

17. *Graphs and their representations.* How would you represent the air transport network of the world by a weighted graph?

(a) The nodes would be _____

(b) The edges would be _____

(c) The weights would be _____

(d) How could this graph, plus some additional information, be used to run a ticket-selling website where you would enter your starting and destination cities, and travel dates, and get the best flights? What algorithms would be used?