

This exam has 13 questions worth 80 points. Please make sure that you have them all.

1. Use back substitution to solve this recurrence exactly:
 $T(0) = 1$, and $T(n) = T(n - 1) + 2$ for $n > 0$.

$$\begin{aligned} T(n) &= T(n - 1) + 2 \\ &= T(n - 2) + 2 + 2 \\ &= T(n - 3) + 2 + 2 + 2 \\ &= \dots \\ &= T(0) + 2 + 2 + \dots + 2. \end{aligned}$$

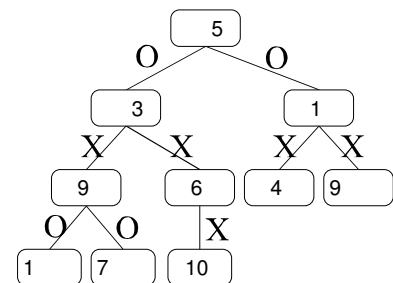
The question is, how many 2s are there? We observe that the number of 2s is in each case the same as the number subtracted from n in the $T(n - \text{something})$ expressions. Thus, by the time we get down to $T(0)$, we have $T(n - n)$, and thus have n 2s. the answer is $T(0) = 2n$, or $2n + 1$.

2. Use the Master Theorem to solve each of these recurrences. In each case, show all work.
- $T(n) = 3T(n/4) + n$ In this case, we compare $\log_4 3$ with 1 (since $n = n^1$). Since $\log_4 3 < 1$, we have case 3 of the MT, and the answer is $T(n) = \Theta(n)$.
 - $T(n) = 4T(n/3) + n$ In this case, we compare $\log_3 4$ with 1 (since $n = n^1$). Since $\log_3 4 > 1$, we have case 1 of the MT, and the answer is $T(n) = \Theta(n^{\log_3 4})$.
 - $T(n) = 21T(n/2) + n^4$ In this case, we compare $\log_2 21$ with 4. Since $\log_2 21 > 4$, we have case 1 of the MT, and the answer is $T(n) = \Theta(n^{\log_2 21})$.
3. Why doesn't the Master Theorem apply to the following recurrence: $T(n) = 3T(n / 3) + n \log n$?
 Although $n \log n$ is asymptotically greater than n , it is not polynomially larger.
4. Suppose the function T satisfies the recurrence $T(n) = 3T(n / 2) + 2n^2$. Prove, without using the Master theorem, that $T(n) = O(n^2)$. Make sure that you explicitly give constants c and n_0 .

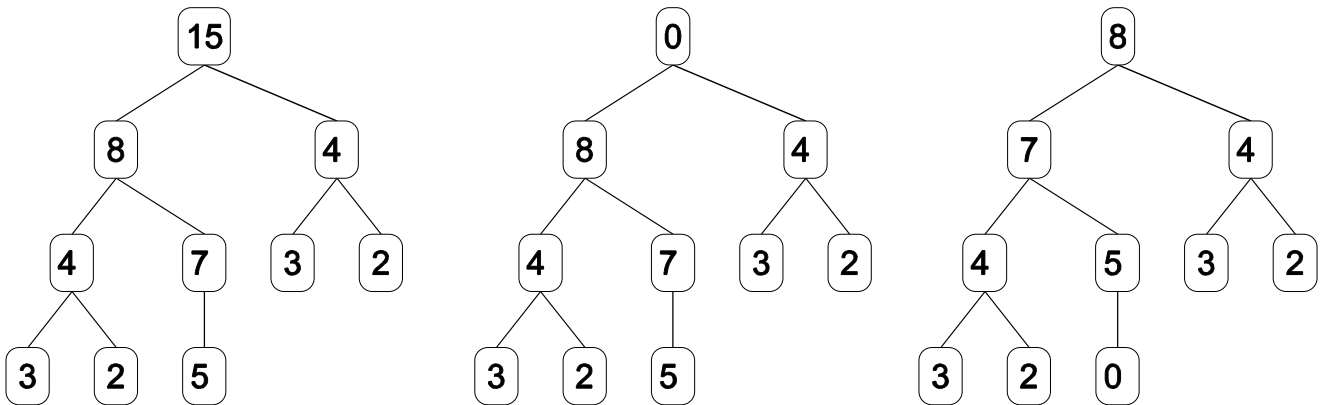
*We want to prove that there is some constant c and some constant n_0 so that for $n > n_0$, $T(n) \leq cn^2$.
 Let us assume that for $k < n$, we have $T(k) \leq ck^2$.*

Consider $T(n)$. By the recurrence, we have $T(n) = 3T(n / 2) + 2n^2$, and by our assumption about $T(k)$ for values of $k < n$, we have $T(n) = 3T(n / 2) + 2n^2 \leq 3 \times c \times (n/2)^2 + 2n^2 = (3c/4 + 2)n^2$, which we want to be less than or equal to cn^2 . For this to happen, we want $(3c/4 + 2) \leq c$, which is true when $c \geq 8$. Since I didn't give a starting value for $T(0)$ or $T(1)$, it doesn't matter what n_0 you select.

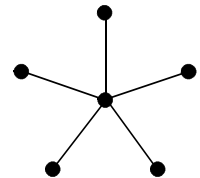
5. Select 10 random numbers from 1 to 50, and place them in a heap ■
 to the right, so that it is neither a max heap nor a min heap.
- Indicate with an "X" a place where your heap fails to have the max heap property.
 - Indicate with an "O" a place where your heap fails to have the min heap property.



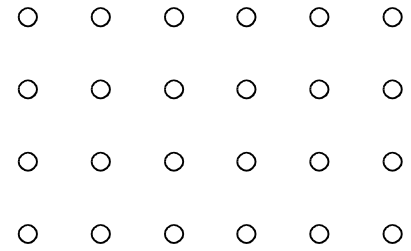
6. Below are three copies of a heap.
- In the first one, draw a max heap with all elements between 1 and 20.
 - The second should be the same as the first, except change the top element to a "0."
 - And the third should show the result after calling MAX-HEAPIFY on the top element of this heap, after all (if any) recursive calls that MAX-HEAPIFY makes to itself.



7. In our BUILD-MAX-HEAP function, we start with a heap and then work backwards from some position in the array back to the beginning of the array.
- Where do we start? *Half way, at position $n/2$*
 - Show that the worst-case running time for BUILD-MAX-HEAP is $O(n)$. *See pages 1 and 2 of the online notes for 2/27/06*
8. We can treat a max heap as a max priority queue.
- What are the operations that we need to implement in a max priority queue?
Insert, Increase-Key, and Extract
 - Show how to implement the operation that extracts the maximum element from a max priority queue.
Swap the first and last elements of the queue, decrement heapsize, Max-Heapify the top element of the queue, and return $A[\text{heapsize} + 1]$
9. Describe in words, with no pseudocode, how to use a max heap to sort an array from smallest to largest.
Swap $A[\text{heapsize}]$ with $A[1]$, decrement heapsize, max-heapify $A[1]$, and iterate until heapsize = 1.
10. Suppose that T is a tree with a vertex v of degree 5
- Prove that T must have at least 5 leaves. *From v , we have five different directions in which to walk out of v . Each of them will eventually dead-end, since T has no cycles, so each of these will lead to at least one leaf.*
 - Show that T need not have more than 5 leaves. *See the figure to the right.*



11. To the right is an array of dots. **[This problem has been modified from its original form. It has been formatted to be correct.]**



- a. For each dot *except the top one in the leftmost column*, do the following:
- If it's in the top row, connect it to its left neighbor
 - If it's in the left column, connect it to its top neighbor
 - Otherwise, mentally flip a coin, and if the coin comes up heads, connect the dot to the dot directly above it, and if it comes up tails, connect it to the dot directly to its left.

You go ahead and draw these connections. It's therapeutic!

- How many connections did you draw? *23.*
- Did your figure have any cycles? *Nope.*
- Prove that no matter how your coins are tossed, the resulting figure will be a tree.

To prove that a graph on n vertices is a tree, you can prove any two of the three following conditions:

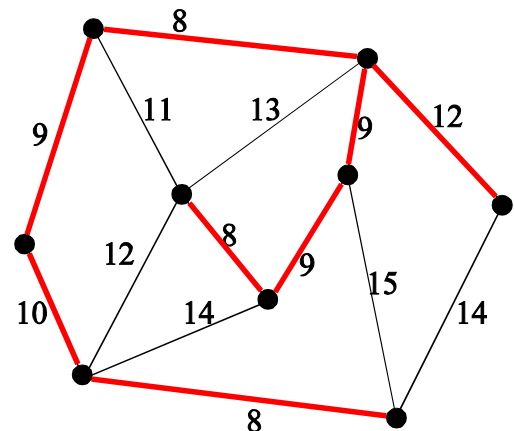
- It is connected*
- It is acyclic*
- It has $n - 1$ edges*

We can easily prove it is connected by observing that from every vertex there must be a path from that vertex to the root (root = top-left vertex). This is because out of each vertex other than the root, we can step either up or left, ending only when we get to the root. Finally, since each vertex has a path to the root, we can get from any vertex to any other vertex by going through the root, thus proving connectedness.

Finally, it is clear that the graph has $n - 1$ edges, since an edge was drawn from every vertex, except the root.

12. Find a minimum weight spanning tree on the weighted graph to the right.

- Label the edges in your spanning tree, and list below the order in which you selected your edges. *The red (bold) edges are the edges of the tree. They were selected by weight, lowest to highest.*
- What algorithm did you use?
Kruskal's algorithm, breaking ties arbitrarily.



Prim's algorithm started from the leftmost vertex would have selected them as follows: (I'll give the weights of the edges. No ties needed to be broken. Since the tree must stay connected at all times, this uniquely determines the order in which I built the tree.) 9-8-9-9-8-10-8-12.