**CSCI 174**

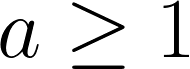
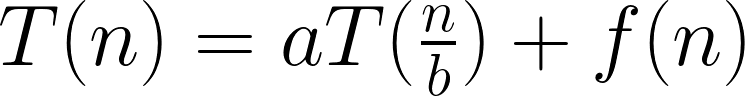
**Assignment 2**

**Master Method**

**Questions**

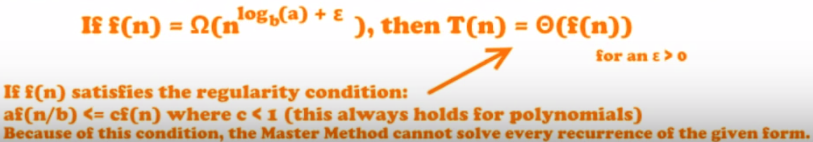
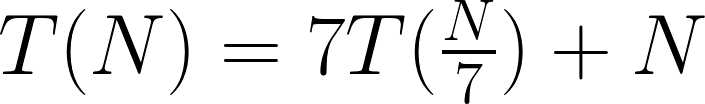
**Name: Abhishek Gupta**

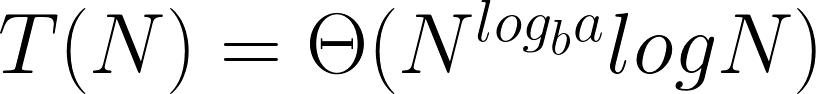
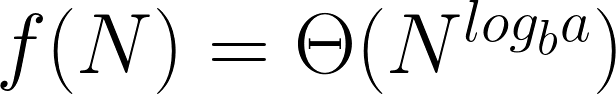
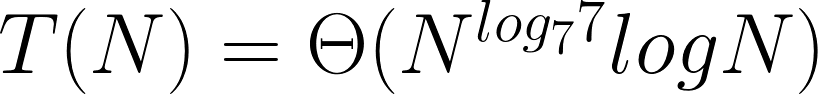
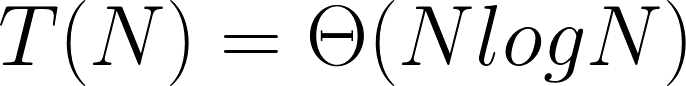
**ID Number:**

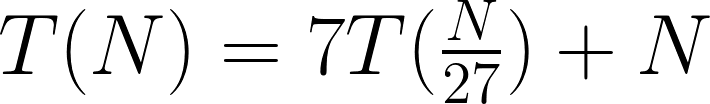
**C:\Users\joopj\AppData\Local\Microsoft\Windows\INetCache\Content.Word\CodeCogsEqn (2).pngMaster Method Rules:**

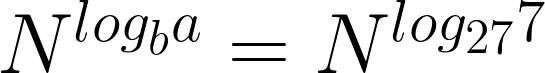
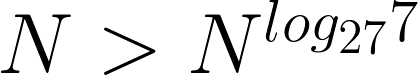
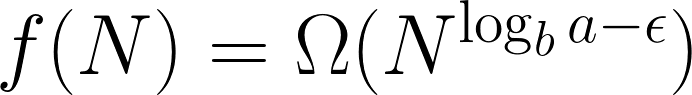
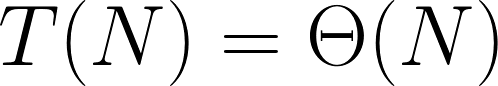
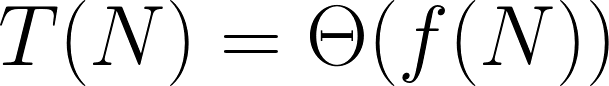
* n = size of current problem.
* a = number of subproblems in the recursion.
* n/b = size of each subproblem.
* f(n) = cost of the work that has to be done outside the recursive calls (cost of dividing + merging).

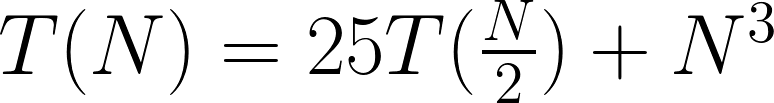
**3 Cases for f(n):**

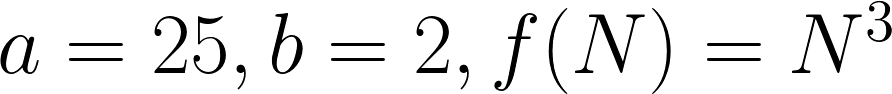
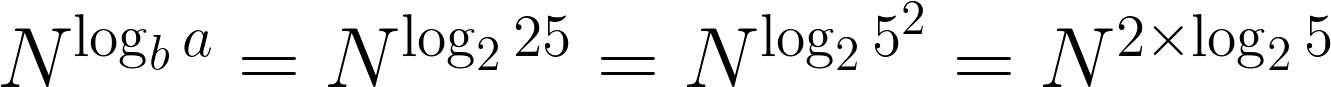
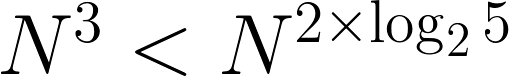
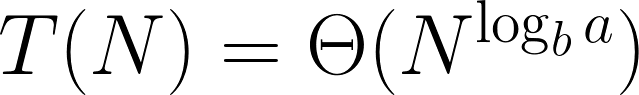
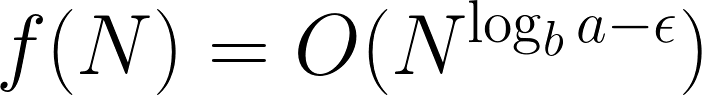
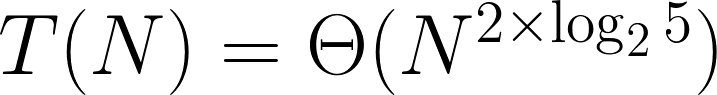
1. Runtime is dominated by cost at the leaves.  
   
2. Runtime is evenly distributed throughout tree.  
   C:\Users\joopj\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2.png
3. Runtime is dominated by the cost at the root.   
   
4. Use the Master Method to solve the recurrences below.
5. 

STEP 1: Extract a, b, and f(N)  
 a = 7, b = 7, f(N) = N  
STEP 2: Compute   
  
   
  
STEP 3: Compare f(N) with   
  
 N = N, therefore we look at CASE 2  
STEP 4: If   
  
 then

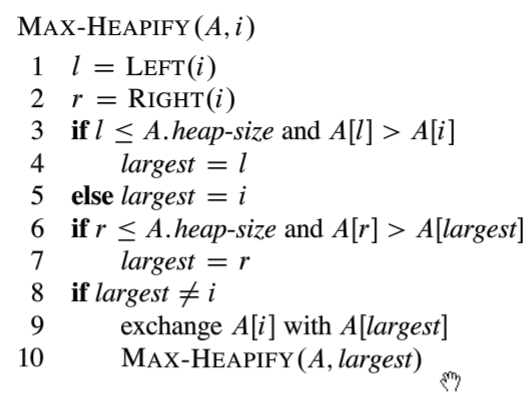
1. 

 STEP 1: Extract a, b, and f(N)  
 a = 7, b = 27, f(N) = N  
 STEP 2: Compute   
  
   
  
 STEP 3: Compare f(N) with   
  
   
therefore we look at CASE 3  
 STEP 4: If   
  
 then

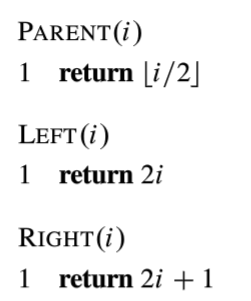
1. 

STEP 1: Extract a, b and f(N)  
   
  
STEP 2: Compute   
  
   
  
STEP 3: Compare f(N) with   
  
 therefore CASE 1   
STEP 4: If   
 then

1. Consider the following pseudo-code:

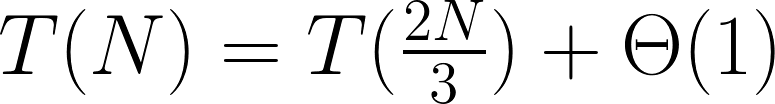
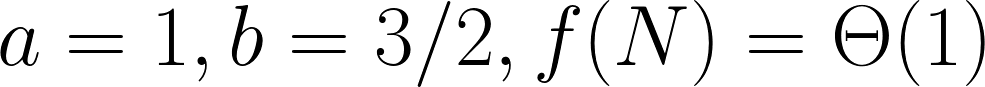
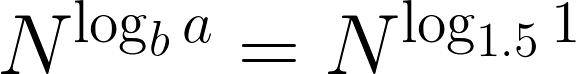


With the utility functions:



Model this algorithm with a recurrence relation, then solve it using master method.

  
STEP 1: Extract a, b and f(N)  
  
  
STEP 2: Compute   
  
  
   
STEP 3: Compare f(N) with   
  
STEP 4:

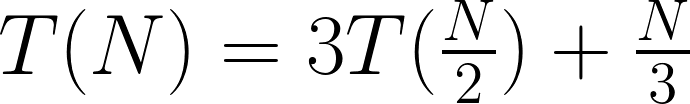


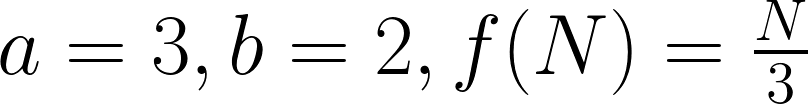
A = 1, b = 1.5, f(N) = O(1)

Compute

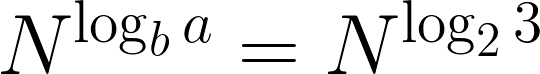
1. Consider an algorithm where we take in a problem and create 3 subproblems each half a large as the original problem. Then the subproblem solutions are combined with a single loop through a list 1/3 the size of the original problem.

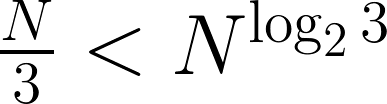
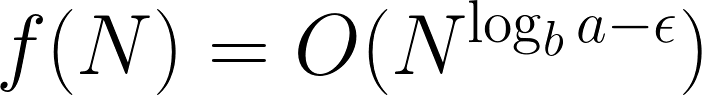
Model this problem with a recurrence relation and solve using the master method.

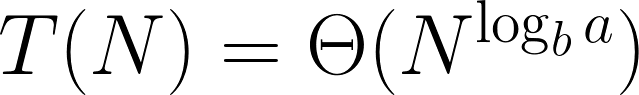
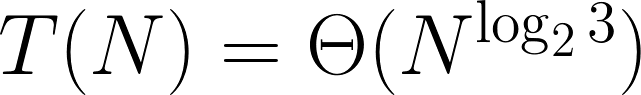


STEP 1: Extract a, b and f(N)

STEP 2: Compute



STEP 3: Compare f(N) with   
  
 therefore CASE 1

 STEP 4: If   
  
   
 then

**Works Cited**

1. **VIDEO:** Master Method ( incl. Step-By-Step Guide and Examples ) – Analysis  
   **LINK:** <https://www.youtube.com/watch?v=6CX7s7JnXs0>
2. **VIDEO:** Algorithms 6.2 – max heapify  
   **LINK:** <https://www.youtube.com/watch?v=tydfy_rLGmI>

<https://stackoverflow.com/questions/9099110/worst-case-in-max-heapify-how-do-you-get-2n-3>

<http://www.cems.uvm.edu/~rsnapp/teaching/cs124/notes/cs124notes_031914.pdf>

<https://www.quora.com/How-is-the-reccurence-relation-of-Max-Heapify-T-n-T-2n-3-+-theta-of-1>

<https://math.stackexchange.com/questions/181022/worst-case-analysis-of-max-heapify-procedure>