Anand Mahadevan - SID 862132182

CS141 Homework 2

due Thursday, August 19th 11:59 PM

Problem 1.
a) T(n) = 3 * T(n/3) + O (n)
b) Case: {1 = $log_3 3$}
T(n) = O(n * lg n)
c) The running time of this algorithm matches the run time of the regular merge sort algorithm described in lecture.

Problem 2.
Strassen's algorithm passes in two **square** matrices and outputs the result of multiplication performed on both matrices. X is a kn x n matrix and Y is a n x kn matrix.

a) Algorithm for X times Y:
1) Divide X into k sub matrices row wise where every n / k rows constitutes a sub matrix, each being size n x n.
2) Divide Y into k sub matrices column wise where every n / k columns constitutes a sub matrix, each being size n x n.
3) There will be k^2 calls to Strassen's algorithm with inputs being the corresponding sub matrix from X and Y, outputting to corresponding part of the output matrix.
In terms of runtime, we are calling Strassen's algorithm k^2 times.
T(n) = $O(k^2 * n^{lg7})$
This algorithm will correctly calculate the product because this is simply row times column matrix multiplication being outputted into the proper location into an output matrix. The problem is simply being divided into k^2 sub problems in order to utilize Strassen's algorithm which require two square matrices for input to multiply together.

b) Algorithm for Y times X:
1) Divide Y into k sub matrices column wise where every n / k columns constitutes a sub matrix, each being size n x n.
2) Divide X into k sub matrices row wise where every n / k rows constitutes a sub matrix, each being size n x n.
3) There will be k calls to Strassen's algorithm with inputs being the corresponding sub matrix from X and Y, outputting to corresponding part of the output matrix.
In terms of runtime, we are calling Strassen's algorithm k times.
T(n) = $O(k * n^{lg7})$
This algorithm will correctly calculate the product because this is simply row times column matrix multiplication being outputted into the proper location into an output matrix. The problem is simply being divided into k sub problems in order to utilize Strassen's algorithm which require two square matrices for input to multiply together.

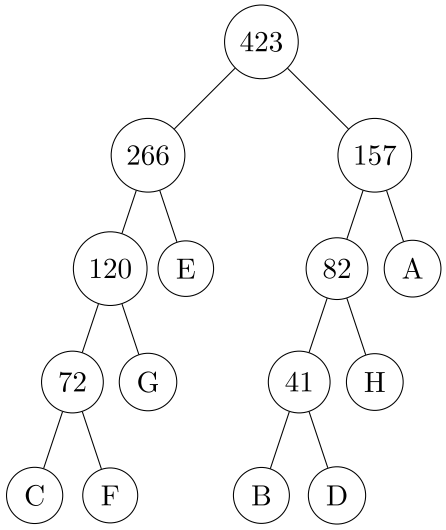
Problem 3.
a) Sort the tasks in order of lowest time taken to highest time taken.
b) O($n * logn$) - Mergesort worst case
c) Assume all t_i are distinct, where i is the number of the task
Assume $t_1 < t_2 < ... < t_n$
Thus the greedy schedule from my algorithm σ is simply 1, 2, 3, ..., n
Let $\sigma*$ be an optimal schedule better than σ, where there are consecutive tasks i and j where i > j thus $t_i > t_j$
Focusing on these two tasks i and j in $\sigma*$, the waiting time for task i is the sum of all times before task i, and the waiting time for task j is the sum of all times before task i **plus** t_i.

If we were to make a new schedule from $\sigma*$, we switch the order of these tasks i, j to have task j performed before i. The waiting time for task j is now the sum of all times before task j, and the waiting time for task i is the sum of all times before task j **plus** t_j.

The Cost of swap: total waiting time increases by t_j.
The Benefit of swap: total waiting time decreases by t_i.
This is a **contradiction** since $t_i > t_j$, meaning there is a larger benefit from the swap of tasks i and j from $\sigma*$ compared to the cost.

This means that our assumption that $\sigma*$ was the optimal schedule was incorrect, and in fact our algorithm produces the optimal schedule.
Q.E.D.

Problem 4.
a)
Strategy: Merge 2 lowest frequency nodes together until only 1 node is remaining
1) Merge B and D into BD (freq. 41)
2) Merge C and F into CF (freq. 72)
3) Merge BD and H into BDH (freq. 82)
4) Merge CF and G into CFG (freq. 120)
5) Merge BDH and A into ABDH (freq. 157)
6) Merge CFG and E into CEFG (freq. 266)
7) Merge CEFG and ABDH into ABCDEFGH (freq. 423)



b)

A: 11

B: 1000

C: 0000

D: 1001

E: 01

F: 0001

G: 001

H: 101

c)

HEADACHE would be encoded to

101 01 11 1001 11 0000 101 01

d)

No, the frequencies of letters used in HEADACHE are vastly different from the originally given frequencies. In fact, letters like B, F, G aren't even used in HEADACHE, so there would be no need to encode those letters.