Extra Assignment + practice questions

If you are happy of your grade, you can skip submitting this assignment. If you look for an extra assignment, you can submit this assignment. This is an **individual** assignment.

Problem 1.

We are given as input a set of n jobs, where job j has a processing time p_j, a deadline d_j. Given a schedule (i.e., an ordering of the jobs), consider that each job j has the completion time C_j; we define the lateness l_j of job j as the amount of time $C_j - d_j$ after its deadline that the job completes, or as 0 if $C_j \leq d_j$. Our goal is to minimize the maximum lateness, $\max l_j$. Consider the following greedy rules for producing an ordering that minimizes the maximum lateness. For each rule, please explain why it gives the optimal ordering or give a counterexample. You can assume that all processing times and deadlines are distinct.

(a) Schedule the requests in increasing order of processing time p_j.

(b) Schedule the requests in increasing order of the product $d_j \times p_j$.

(c) Schedule the requests in increasing order of deadline d_j.

Problem 2. Explain Karatsuba's algorithm first. Then use it to solve the following multiplication problem. Show all your work. Apply recursive calls of the multiplication function until the base case, where each number has only one digit.

X= 86
Y= 27
XY=?

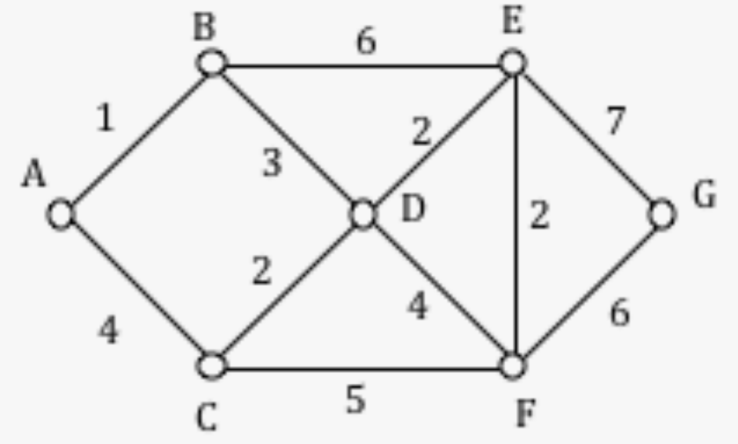
Problem 3: Use Strassen Matrix Multiplication algorithm to multiply the following matrices:

$$A=\begin{bmatrix}1 & 2 & 1 & 4\\ 2 & 1 & 3 & 2\\ 3 & 1 & 2 & 4\\ 1 & 3 & 1 & 2\end{bmatrix}, \quad B=\begin{bmatrix}2 & 6 & 3 & 4\\ 2 & 1 & 1 & 2\\ 3 & 1 & 2 & 1\\ 2 & 2 & 1 & 2\end{bmatrix}$$

Show all your work. Base case for recursive calls is once you get to one number, ilike a 1 by 1 matrix.

Extra Practice questions: No need to submit your answers to the following questions but practice them to get ready for the exam.

Problem 4. Use Dijkstra's algorithm to solve the shortest path problem from node A to ALL other nodes in the below graph. Show all your work and explain. Draw the graph again after each step.
What is the final shortest path tree with the distances from s to each node?
(We will learn this concept next week)



Problem 5. Consider the following recurrence relation and solve it to come up with a precise function of n in closed form (that means you should resolve all sigmas, recursive calls of the function T, etc.). **An asymptotic answer is NOT acceptable**. Justify your solution and show all your work.

T(n) = T(n/2) + 2n where $T(1) = 1$ and $n = 2^k$ for a non-negative integer k.

Problem 6. Count the precise number of "fundamental operations" executed in the following code. Again, your answer should be a function of n (n ≥ 0) in closed form. No asymptotic bound is accepted.

```
for(int i = 0; i <= n; i++)
{
        Perform 1 fundamental operation;
        for(int j = i+1; j <= n; j++)
                Perform 1 fundamental operation;
          //endfor j
}//endfor i
```

Problem 7.
A binary tree's “maximum depth” is the number of nodes along the longest path from the root node down to the farthest leaf node. Given the root of a binary tree, write a complete program in C++/Java that returns three’s maximum depth. What is the time-complexity of your algorithm in the worst-case once you have n nodes in the tree. Analyze and clearly discuss your reasoning.
Paste your complete program in the solution file.

Problem 8. Implement the **Depth first Search** and **Breath first Search** Algorithms for trees in C++/Java program.

Problem 9. Implement merge sort algorithm as studied in the course.

Problem 10. Now design a merge-sort algorithm that, instead of splitting the array into two subparts, splits the array into three subparts and recursively sorts the arrays as before. Write a code in C++/Java to implement your algorithm. Show it works.
Analyze your algorithm by writing the recurrence relation and solving the relation for an upper bound (Big-O).

Problem 11. Quick sort algorithm uses a divide and conquer strategy to sort the given list. Assume the pivot is always defined such that it divides up the list into two subarrays of equal size. Analyze your algorithm by writing the recurrence relation. Apply Master Theroem to bound the relation asymptotically.

Problem 12.
What are the similarities and differences of the following trees? How does an insertion/deletion of an element work in each.

1) Binary Search Trees
2) AVL trees
3) Heaps