1. (25 points) Design a decrease-and-conquer algorithm for efficiently computing \( \lfloor \log n \rfloor \) and express it in pseudocode. Determine its time efficiency.

2. (25 points) You have 9 gigabytes of data in a file on the disk. You are allowed to use only 1 gigabyte of RAM. How can you efficiently sort the data in the file given that a disk-access is very costly compared to a RAM-access. Describe your idea with as much specificity as you can. Assume 1 gigabyte = 1000 megabytes. \textbf{Hint}: Can you utilize the ideas of mergesort?

3. (25 points) We can multiply two \( 2 \times 2 \) matrices to get their product matrix in the following way

\[
\begin{bmatrix}
a & b \\
c & d
\end{bmatrix}
\cdot
\begin{bmatrix}
A & C \\
B & D
\end{bmatrix}
= 
\begin{bmatrix}
aA + bB & w + v + (a + b - c - d)D \\
w + u + d(B + C - A - D) & w + u + v
\end{bmatrix}
\]

where \( u = (c - a)(C - D) \), \( v = (c + d)(C - A) \), \( w = aA + (c + d - a)(A + D - C) \).

We could generalize the above idea for \textit{efficiently} multiplying two \( n \times n \) matrices where \( n \) is a power of 2, like we did for Strassen's algorithm in the class. \textbf{Hint}: Do not forget to reuse computations.

(a) Set up the recurrence relation for the number of multiplications including the base case. Solve it.

(b) Set up the recurrence relation for the number of additions/subtractions including the base case. Solve it.

(c) To what asymptotic efficiency class does this algorithm's complexity belong? Is it better than Strassen's algorithm? If yes, in what respect? If no, why is it inferior?

4. (25 points) Consider the following algorithm for transforming an array into a (max) heap:

\begin{verbatim}
Algorithm 1: Bottom up heap construction

input : An array \( H[1..n] \) of orderable items
output: A heap \( H[1..n] \)
1 for \( i \leftarrow \lfloor \frac{n}{2} \rfloor \) downto 1 do
2   \( k \leftarrow i \)
3   \( v \leftarrow H[k] \)
4   heap \leftarrow false
5   while not heap and \( 2 \cdot k \leq n \) do
6      \( j \leftarrow 2 \cdot k \)
7      if \( j < n \) then
8         if \( H[j] < H[j + 1] \) then
9            \( j \leftarrow j + 1 \)
10        end
11     end
12     if \( v \geq H[j] \) then
13        heap \leftarrow true
14     else
15        \( H[k] \leftarrow H[j] \)
16        \( k \leftarrow j \)
17     end
18 end
19 \( H[k] \leftarrow v \)
20 end
\end{verbatim}
(a) What changes would you make to Algorithm 1, in order to use it for transforming an array into a min heap, instead of a max heap. Recall that a min heap is an essentially complete binary tree in which, every key is less than or equal to the keys in its children.

(b) Perform heapsort on the array ⟨7, 8, 16, 4, 14⟩. Show the states of the array during iterations both for the “heap construction” and the “maximum deletions” stages.

1 Master Theorem

For the recurrence relation $T(n) = aT\left(\frac{n}{b}\right) + f(n)$, if $f(n) \in \Theta(n^d)$ where $d \geq 0$, then

$$T(n) \in \begin{cases} 
\Theta(n^d) & \text{if } a < b^d \\
\Theta(n^d \lg n) & \text{if } a = b^d \\
\Theta(n^{\log_b a}) & \text{if } a > b^d
\end{cases}$$

Analogous results hold for the $O$ and $\Omega$ notations.

2 Useful formulas

1. $\sum_{i=l}^{u} 1 = u - l + 1$
2. $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
3. $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$
4. $\sum_{i=0}^{n} a^i = \frac{a^{n+1} - 1}{a - 1} (a \neq 1)$
5. $\sum_{i=1}^{n} i^2 = (n-1)2^{n+1} + 2$
6. $\lg n = \frac{\ln n}{\ln 2}$
7. $\frac{d}{dn} n^m = mn^{m-1}$
8. $\frac{d}{dn} \ln n = \frac{1}{n}$
9. $\frac{d}{dn} a^n = (\ln a)a^n$