$4.1-1$ *Generalize MATRIX − MULTIPLY − RECURSIVE to multiply n x n matrix for which n is not necessarily an exact power of 2. Give a recurrence describing its running time. Argue that it runs in $\Theta(n^3)$ time in the worst case*

```
MATRIX-MULTIPLY-RECURSIVE(A, B, C, n)
if n == 1
// Base case.
    c11 = c11 + a11 · b11
    return
// Divide.
partition A, B, and C into n/2 × n/2 submatrices
    A11, A12, A21, A22; B11, B12, B21, B22;
    and C11, C12, C21, C22; respectively
// Conquer.
MATRIX-MULTIPLY-RECURSIVE(A11, B11, C11, n/2)
MATRIX-MULTIPLY-RECURSIVE(A11, B12, C12, n/2)
MATRIX-MULTIPLY-RECURSIVE(A21, B11, C21, n/2)
MATRIX-MULTIPLY-RECURSIVE(A21, B12, C22, n/2)
MATRIX-MULTIPLY-RECURSIVE(A12, B21, C11, n/2)
MATRIX-MULTIPLY-RECURSIVE(A12, B22, C12, n/2)
MATRIX-MULTIPLY-RECURSIVE(A22, B21, C21, n/2)
MATRIX-MULTIPLY-RECURSIVE(A22, B22, C22, n/2)
```

This algorithm is based on a matrix of size $2^b x\ 2^b$, where $b > 0$ and is an integer

let n be the size of our matrix, we know $\exists\ 2^{b+1}$ greater than n and 2^b lower than n. In this case We will construct a 2^{b+1} matrix in which the rows and columns from n to 2^{b+1} are 0's.
We would have:

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{n1} & \cdots & A_{nn} \end{bmatrix} \rightarrow \begin{bmatrix} A_{11} & \cdots & A_{1n} & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ A_{n1} & \cdots & A_{nn} & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} B_{11} & \cdots & B_{1n} \\ \vdots & \ddots & \vdots \\ B_{n1} & \cdots & B_{nn} \end{bmatrix} \rightarrow \begin{bmatrix} B_{11} & \cdots & B_{1n} & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ B_{n1} & \cdots & B_{nn} & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 \end{bmatrix}$$

This will be the preprocessing, after it we can call Matrix − Multiple − Recursive

We basically need to create another two arrays of size 2^{b+1} and we know that
$2^b < n < 2^{b+1} < 2n$. *The number elements created are* $2\left(2^{b+1}\right)^2 < 8n^2$
$\rightarrow$ *this creation takes* $\Theta(n^2)$

After it we will have a resulting matrix C, in which the submatrix composed by the first n rows and the first n columns are our desired matrix(We will prove it!),
all other element c_{ij} for $i > n$ or $j > n$ will be 0

After that We can extract our matrix trough n^2 operations.

Our pre processing and posprocessing are $\Theta(n^2)$, but our overall algorithm is $\Theta(n^3)$ so it remains $\Theta(n^3)$

Now lets prove why this algorithm we created still work

$We\ know\ that, for\ A\ and\ B, without\ being\ padded, we\ have\ c_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}$

$For\ our\ padded\ A\ and\ B\ we\ have\ c^*_{ij} = \sum_{k=1}^{2^{b+1}} a_{ik}b_{kj}\ , where\ c_{ij}\ are\ the\ elements\ of\ A \cdot B$

$and\ c^*_{ij}\ are\ elements\ of\ A(padded) \cdot B(padded)$

$we\ know\ that\ a_{ik}, b_{ik} = 0\ \forall i > n\ and\ k > n$

$In\ this\ way\ we\ have\ that\ c^*_{ij} = 0\ \forall i, or\ j > n. Now\ if\ we\ have\ them \le n:$

$$c^*_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj} + \sum_{k=n+1}^{2^{b+1}} a_{ik}b_{kj}\ , but\ for\ the\ second\ summation\ all\ we\ have\ is\ 0$$

$$So\ we\ have\ that\ c^*_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj} = c_{ij}\ \forall i\ and\ j \le n$$

4.1.2 How quickly can you multiple a $kn \times n$ matrix by a $n \times kn$ matrix, where $k \ge 1$, using Matrix − Multipl − Recursive as a subroutine? Answer the same question for multiplying $n \times kn$ by a $kn \times n$. Which is Asymptotically better?

1^{st} *case*:

$let\ A\ be\ a\ kn \times n. Divide\ it\ in\ k\ (n \times n)\ matrices\ and\ do\ the\ same\ for\ B, the\ n \times kn\ matrix$

We now multiply each one of the matrices produced by the division of A by each one of the matrices produced by the division of B. This will generate k^2 $n \times n$ submatrices. After we have each one we can build *our resulting matrix*

since we will generate k^2 $n \times n$ submatrices. we will call the subroutine k^2 times

What leads us to a $\Theta(k^2n^3)$ running time.

2^{nd} *case*:

We do the same. but now we have k matrices multiplications since we will multiplicate the i^{th} divided submatrix of A by the i^{th} divided submatrix of B. We will maitain this in memory and sumup all the resulting matrices. What leads us to k calls of the subroutine

this leads us to $\Theta(kn^3)$ to end the algorithm properly we need to sum all the k matrices. each one have n^2 entries, so we add a $\Theta(kn^2)$. Resulting in a $\Theta(kn^3)$, which is faster

Now we have to prove that the algorithms are correct. We have the following for the first case:

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{kn,1} & \cdots & A_{kn,n} \end{bmatrix}\ and\ B = \begin{bmatrix} B_{11} & \cdots & B_{1,kn} \\ \vdots & \ddots & \vdots \\ B_{n1} & \cdots & B_{n,kn} \end{bmatrix}$$

We will divide them into k submatrices in the form $n \times n$ as following:

$$A_{k1} = \begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{n,1} & \cdots & A_{n,n} \end{bmatrix};\ A_{k2} = \begin{bmatrix} A_{n+1,1} & \cdots & A_{n+1,n} \\ \vdots & \ddots & \vdots \\ A_{2n,1} & \cdots & A_{2n,n} \end{bmatrix} \ldots$$

$$A_{kk} = \begin{bmatrix} A_{(k-1)n+1,1} & \cdots & A_{(k-1)n+1,1,n} \\ \vdots & \ddots & \vdots \\ A_{kn,1} & \cdots & A_{kn,n} \end{bmatrix}$$

$$B_{k1} = \begin{bmatrix} B_{11} & \cdots & B_{1,n} \\ \vdots & \ddots & \vdots \\ B_{n,1} & \cdots & B_{n,n} \end{bmatrix};\ B_{k2} = \begin{bmatrix} B_{1,n+1} & \cdots & A_{1,2n} \\ \vdots & \ddots & \vdots \\ B_{n,n+1} & \cdots & A_{n,2n} \end{bmatrix} \ldots B_{kk} = \begin{bmatrix} B_{1,(k-1)n+1} & \cdots & B_{1,kn} \\ \vdots & \ddots & \vdots \\ B_{n,(k-1)n+1} & \cdots & B_{n,kn} \end{bmatrix}$$

$Now\ we\ want\ to\ show\ that\ A_{kb} \cdot B_{kj} = submatrix\ of\ A \cdot B\ referencing\ the\ elements$
$between\ the\ rows\ (b-1)n+1\ until\ bn\ and\ between\ the\ columns\ (j-1)n+1\ until\ jn$

$$= \begin{bmatrix} A_{(b-1)n+1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{bn,1} & \cdots & A_{n,n} \end{bmatrix} \cdot \begin{bmatrix} B_{1,(j-1)n+1} & \cdots & B_{1,jn} \\ \vdots & \ddots & \vdots \\ B_{n,,(j-1)n+1} & \cdots & B_{n,jn} \end{bmatrix}$$

$We\ also\ know\ that$

$c_{id} = \sum_{h=1}^{n} a_{ih} b_{hd}\ for\ the\ original\ matrix\ and\ c_{id}^{k} = \sum_{h=1}^{n} a_{ih}^{k} b_{hd}^{k}\ for\ the\ matrix\ resulted$
$from\ the\ multiplication\ above. Now, we\ know\ that$:

$a_{ih}^{k} = a_{(b-1)n+i,h}\ and\ b_{hd}^{k} = b_{h,(j-1)n+i} \rightarrow$

$$c_{id}^{k} = \sum_{h=1}^{n} a_{(b-1)n+i,h}\ b_{h,(j-1)n+i}\ \ for\ 1 \le i \le n\ and\ 1 \le d \le n$$

$$c_{id} = \sum_{h=1}^{n} a_{ih} b_{hd}\ \ for\ 1 \le i \le kn\ and\ 1 \le d \le kn.$$

$Now\ lets\ get\ the\ elements\ in\ which\ (b-1)n+1 \le i \le bn\ and\ (j-1)n+1 \le d \le jn \rightarrow$

$we\ can\ write\ i\ in\ the\ form:\ (b-1)n+K, K\ from\ 1\ to\ n.$
$we\ can\ write\ i\ in\ the\ form:\ (j-1)n+D, D\ from\ 1\ to\ n$

$We\ can\ write\ the\ following\ for\ all\ k, D \in Z, such\ 1 \le K, D \le n$

$$c_{(b-1)n+K,(j-1)n+D} = \sum_{h=1}^{n} a_{(b-1)n+K,h} b_{h,(j-1)n+D} = \sum_{h=1}^{n} a_{(b-1)n+K,h}\ b_{h,(j-1)n+D} = c_{K,D}^{k}$$

$c_{(b-1)n+K,(j-1)n+D} = c_{K,D}^{k}\ \forall\ k, D \in Z, such\ 1 \le K, D \le n$

$We\ can\ the\ veracity\ of\ the\ other\ algoroth\ in\ a\ analog\ manner$

$\boldsymbol{4.1.3}$ ***Supose that instead of partitioning matrices by index calculation in Matrix Algorithm you copy the elements of A, B and C into separates*** $\boldsymbol{n/2 \times \frac{n}{2}}$ ***submatrices. After the recursive calls, you copy the results from*** $\boldsymbol{C_{11}, C_{12}, C_{21}, C_{22}}$ ***back into the appropriate places in C. How oes the recurrence change and what is the solution?***

if I make a copy in each iteration, we would have $3 * n^2$ *access to the data, probabily multiplied by two because we have to read and write the data in a new place. After that we would need to copy all elements of* C_{11}, C_{12}, C_{21} *and* C_{22} *back to C. What would take more* $2n^2$

In this case our Recurrence would be $T(n) = 8 * T\left(\frac{n}{2}\right) + 8n^2$.
We can use the master theorem and show that this is $\Theta(n^3)$

$\boldsymbol{4.1.4}$ ***Write pseudocode for a divide and conquer algorithm MatrixAddRecurse that sums two*** $\boldsymbol{n \times n}$ ***matrices A and B by partitioning each of them into four*** $\boldsymbol{\frac{n}{2} \times \frac{n}{2}}$ ***submatrices and then recursively summing corresponding pairs of submatrices. Assume Assume that matrix partitioning uses*** $\boldsymbol{\Theta(1)}$ ***time index calculations. Write a recurrence for the worst − case running time of this algorithm and solve your recurrence. What happens if you use*** $\boldsymbol{\Theta(n^2)}$ ***time copying to implement the partitioning instead of index calculations?***

We could have:
$MATRIX - ADD - RECURSIVE(A, B, C, n)$

$if\ n == 1:$
    $c_{11} = c_{11} + b_{11} + a_{11}$
Copy or index calculation(A, B, C);
$MATRIX - ADD - RECURSIVE(A_{11}, B_{11}, C_{11}, n/2)$
$MATRIX - ADD - RECURSIVE(A_{12}, B_{12}, C_{12}, n/2)$
$MATRIX - ADD - RECURSIVE(A_{21}, B_{21}, C_{21}, n/2)$
$MATRIX - ADD - RECURSIVE(A_{22}, B_{22}, C_{22}, n/2)$
move $C_{11}C_{12}, C_{21}, C_{22}$ *to C #if we use the copy method*
The recurrente will be $T(n) = 4T\left(\frac{n}{2}\right) + \Theta(1)$*without copying and*
and $T(n) = 4T\left(\frac{n}{2}\right) + \Theta(n^2)$ *with copying*

$\boldsymbol{4.2.1}$ ***Use strassen's algorithm to compute the matrix product***: $\begin{pmatrix} \mathbf{1} & \mathbf{3} \\ \mathbf{7} & \mathbf{5} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{6} & \mathbf{8} \\ \mathbf{4} & \mathbf{2} \end{pmatrix}$

Strassen's algorithm is based on the idea of reducing the number of multiplications through a very clever way of produzing submatrices from the original ones.
It's developed to be used on $\boldsymbol{2^n \frac{row}{column}}$ ***size. for*** $\boldsymbol{n \in Z > 0}$

It's just a receipt, we have:

$S_1 = B_{12} - B_{22} = 6; S_2 = A_{11} + A_{12} = 4; S_3 = A_{21} + A_{22} = 12; S_4 = B_{21} - B_{11} = -2;$
$S_5 = A_{11} + A_{22} = 6; S_6 = B_{11} + B_{22} = 8; S_7 = A_{12} - A_{22} = -2; S_8 = B_{21} + B_{22} = 6$
$S_9 = A_{11} - A_{21} = -6; S_{10} = B_{11} + B_{12} = 14$

$P_1 = A_{11} \cdot S_1 = 6 \mid P_2 = S_2 \cdot B_{22} = 8 \mid P_3 = S_3 \cdot B_{11} = 72 \mid P_4 = A_{22} \cdot S_4 = -10$
$P_5 = S_5 \cdot S_6 = 48 \mid P_6 = -12 \mid P_7 = S_9 \cdot S_{10} = -84$

$C_{11} = P_5 + P_4 - P_2 + P_6 = 18$
$C_{12} = P_1 + P_2 = 14$
$C_{21} = P_3 + P_4 = 62$
$C_{22} = P_5 + P_1 - P_3 - P_7 = 66$

$C = \begin{pmatrix} 18 & 14 \\ 62 & 66 \end{pmatrix}$

$\boldsymbol{4.2.2}$ ***Write pseudocode for Strassen algorithm.***

$STRASSEN(A, B, C, n)$

$if\ n == 1$:
    $c_{11} = c_{11} + b_{11} + a_{11}$

Initialize all constants $S_1 \dots S_{10}$
$Strassen(A_{11}, S_1, P_1, n/2)$
$Strassen(S_2, B_{22}, P_2, n/2)$
$Strassen(S_3, B_{11}, P_3, n/2)$
$Strassen(A_{22}, S_4, P_4, n/2\ )$
$Strassen(S_5, S_6, P_5, n/2)$
$Strassen(S_7, S_8, P_6, n/2)$
$Strassen(S_9, S_{10}, P_7, n/2)$

$C_{11} = P_5 + P_4 - P_2 + P_6$
$C_{12} = P_1 + P_2$
$C_{21} = P_3 + P_4$
$C_{22} = P_5 + P_1$

Combine $(C_{11}, C_{12}, C_{21}, C_{22})$

$4.2.3$ ***What is the largest k such that if you can multiply 3x3 matrices using k multiplications(not assuming commutativity of multiplication), thenyou can multiply nxn matrices in*** $o(n^{\lg 7})$ ***time? What is the running time of this algorithm?***

Suppose you have a matrix n that is not a power of 3*, you know there exists a constant b such that*:
$3^b < n < 3^{b+1} < 3n$ *now you can pad this matrix creating the reminiscent elements.*
By this inequality we know this will take $(3^{b+1})^2 - n^2 \rightarrow n^2 < (3^{b+1})^2 < 9n^2 \rightarrow < 8n^2$

Now we can divide this new size, let say $k = 3^{b+1}$*, into* $\frac{n}{3}\, x \frac{n}{3}$ *and we can recurre in the following*

$T(n) = k\, T\left(\frac{n}{3}\right) + O(n^2)$*. if* $n = 3$ *then you realize* 3 *multiplications of size* 1*.*
Using master theorem in $n^{\log_3 k}$ *we have the following situations*:
$if\ \log_3 k = 2 \rightarrow\ and\ T(n) = \Theta(n^2 lgn). k = 9\ and\ T(n) = o(n^{\lg 7})$
$if\ \log_3 k < 2 \rightarrow\ and\ T(n) = \Theta(n^2). k < 9\ and\ T(n) = o(n^{\lg 7})$
$if\ \log_3 k > 2 \rightarrow\ and\ T(n) = \Theta(n^{\log_3 k}). k > 9\ and\ T(n) = o(n^{\lg 7})when\ k < 3^{\lg 7}. K\ must\ be\ 21$

$4.2.4$ ***V. Pan discovered a way of multiplying 68x 68 matrices using 132,464 multiplications, a way of multiplying 70 x 70 matrices using 143,640 multiplications and a way of multiplying 72 x 72 matrices using 155,424 multiplications. Which method yields the best asymptotic running time when used in a divid −and − conquer matrix − multiplication algorithm? How does it compare with Strassen's algorithm?***

$4.2.5$ ***Show how to multiply the complex numbers a + bi and c + di using only three multiplications of real numbers. The algorithm should take a, b, c and as input and produce the real component ac − bd and the imaginary component ad + bc separately***

let $a_1 = ac$
let $b_1 = bd$
let $c_1 = (a + b)(c + d)$

the real component is $a_1 - b_1$*. The imaginary component is* $c_1 - a_1 - b_1$

4.2.6 *Suppose that you have a $\Theta(n^\alpha)$time algorithm for squaring a n x n matrix where $\alpha \geq 2$. Show how to use that algorithm to multiply two different n x n matrices in $\Theta(n^\alpha)$*

Given two matrix A x B we want to multiply, create a $2n$ x $2n$ matrix. It have $4n^2$ elements So it takes $\Theta(n^2)$The matrix we want to create is $\begin{pmatrix} 0 & B \\ A & 0 \end{pmatrix}$. We them use the algorithm. That takes $\Theta(2^\alpha n^\alpha)$. So we have an algorithm that takes $\Theta(n^\alpha)$

4.3.1 *Use the substitution method to show that each of the following recurrences defined on the reals has the asymptotic solution specified*

a). $T(n) = T(n-1) + n$ *has solution* $T(n) = O(n^2)$

Supose $T(k) \leq ck^2\ \forall n_0 \leq k < n$

Then $T(n) = T(n-1) + n \leq c(n-1)^2 + n = cn^2 - 2cn + 1 + n \leq cn^2$

if $c(1-2n) + n \leq 0$ *and if* $n_0 + 1 < n$. *Lets choose* n_0 *and* $c = 1$ *Lets adjust the guess*

Supose $T(k) \leq k^2\ \forall\ 1 \leq k < n \rightarrow$

$T(n) \leq n^2 - n + 1 \leq n^2$. *If* $1 \leq n - 1 \rightarrow 2 \leq n$. *if* $n = 1$ $T(1) = 1 \leq 1$ *so*

Supose $T(k) \leq k^2\ \forall\ 1 \leq k < n$ *then we prooved that* $T(n) \leq n^2\ \forall n \geq 1$ *and holds for the base case*

b). $T(n) = T(n/2) + \Theta(1)$ *has solution* $T(n) = O(lg(n))$

Suppose $T(k) \leq clg(k)\ \forall n_0 \leq k < n$ *and* $k = 2^j$ *and* $n = 2^l$*for* $j, l \in Z$ *and* $j, l \geq 0$

Then $T(n) = T\left(\frac{n}{2}\right) + c_1 \leq clg\left(\frac{n}{2}\right) + c_1 = clg(n) - c + c_1$ *This if* $\frac{n}{2} \geq n_0 \rightarrow 2n_0 \leq n$ *and if* $c = c_1$

Now we need to prove that, given our assumptions that this works for $n_0 \leq n < 2n_0$

Lets get $n_0 = 1$. *We need that* $T(1) \leq c_1$. *let* $T(1) = c_1$. *So adjusting our hypothesis*

Suppose $T(k) \leq \Theta(1)lg(k)\ \forall 1 \leq k < n$ *and* $k = 2^j$ *and* $n = 2^l$*for* $j, l \in Z$ *and* $j, l \geq 0$

Then $T(n) \leq \Theta(1)lg(k)\ \forall\ 1 \leq n$

c) **$T(n) = 2T\left(\frac{n}{2}\right) + n$ *has solution* $T(n) = \Theta(nlgn)$**

Suppose $T(k) \leq cklg(k)\ \forall n_0 \leq k < n$ *and* $k = 2^j$ *and* $n = 2^l$*for* $j, l \in Z$ *and* $j, l \geq 0$

$T(n) = 2T\left(\frac{n}{2}\right) + n \leq cnlg(n) - cn + n$. *This if* $0\ n \geq 2n_0$. *lets take* $n_0 = 2 \rightarrow$

$T(2) = 1 \leq 2c$ *if* $c = 1$. *So lets adjust our hypothesis* :

Suppose $T(k) \leq klg(k)\ \forall 2 \leq k < n$ *and* $k = 2^j$ *and* $n = 2^l$*for* $j, l \in Z$ *and* $j, l \geq 0$

Then $T(n) \leq nlg(n)\ \forall n \geq 2, n = 2^i$ *for* $i \geq 0$ $i \in Z$.

We can consider the job to generate a 2^b *for any n. This would take a* $\Theta(n)$. *what wouldn'change The runnig time*

$\boldsymbol{d)\ T(n) = 2T\left(\frac{n}{2}+17\right)+n\ has\ solution\ T(n) = O(nlgn)}$

$Suppose\ that\ T(k) \le cklg(k)\ \forall n_0 \le k < n\ we\ know\ that\ \frac{n}{2}+17 \le \frac{3n}{4}\ \forall n \ge 68$

$Now\ T(n) = 2T\left(\frac{n}{2}+17\right)+n \le 2\cdot c\cdot\left(\frac{n}{2}+17\right)\lg\left(\left(\frac{n}{2}+17\right)\right)+n \rightarrow$

$cn\lg\left(\left(\frac{n}{2}+17\right)\right)+34c\lg\left(\left(\frac{n}{2}+17\right)\right)+n \rightarrow if\ \ n\ge 68 \rightarrow T(n) = 2T\left(\frac{n}{2}+17\right)+n$

$\le cn\lg\left(\frac{3n}{4}\right)+34clg(n)+n \rightarrow cnlg(n)-cnlog\left(\frac{4}{3}\right)+34clg(n)+n \rightarrow$

$T(n) \le cnlg(n)+(34clg(n)-n\left(clg\left(\frac{4}{3}\right)-1\right) \le cnlg(n).$

$This\ all\ holds\ if\ n\ge 68\ and\ +(34clg(n)-n\left(clg\left(\frac{4}{3}\right)-1\right)\le 0$

$(34clg(n)-n\left(clg\left(\frac{4}{3}\right)-1\right)\le 0 \rightarrow if\ c = 20 \rightarrow$

$n > 300\ this\ holds$

$Now\ lets\ adjust\ our\ hypothesis$:

$Suppose\ that\ T(k) \le 20klg(k)\ \forall 300 \le k < n\ and\ we\ know\ that\ \frac{n}{2}+17 \le \frac{3n}{4}\ \forall n \ge 68$

$Now\ T(n) = 2T\left(\frac{n}{2}+17\right)+n \le 2\cdot 20\cdot\left(\frac{n}{2}+17\right)\lg\left(\left(\frac{n}{2}+17\right)\right)+n \rightarrow if\left(\frac{n}{2}+17 \ge 300\right)$

$T(n) \le 20nlog(n)\ if\ (n \ge 566) What\ if\ 300 \le n < 566?\ We\ know\ that\ the\ algorithm\ always\ terminate$

$So\ we\ have\ a\ set\ F\{T(n); for\ 300 \le n < 566\}$

$by\ the\ well\ ordering\ principle\ we\ know\ it\ has\ a\ maximum, let\ say\ c_{special}$ = max(F) .

$Clearly\ T(n) \le c_{special} \le c_{\text{special}}$nlg(n). if $c_{\text{special}} \le 20$ we are done. if $c_{\text{special}} \ge 20$ we will use c as c_{special}. In this case we the n we choosed early can still be considered so we are done

$\boldsymbol{e)\ T(n) = 2T\left(\frac{n}{3}\right)+\Theta(n) has\ solution\ T(n) = \Theta(n)}$

$Suppose\ T(k) \le 3c_2k\ \forall n_0 \le k < n\ where\ k\ is\ 3^i$ for some $i \in \mathbb{Z}$ and $i > 0$ Then:

We know that $\exists c_1, n_0^*$ and c_2 we used above such that $0 \le c_1 n \le \Theta(n) \le c_2 n\ \forall n \ge n_0 \rightarrow$

$T(n) = 2T\left(\frac{n}{3}\right)+\Theta(n) \le \frac{2\cdot 3c_2 n}{3}+\Theta(n) \le 2c_2 n + c_2 n \rightarrow$

$T(n) \le 2c_2 n\quad if\ \ n_0 \le \left(\frac{n}{3}\right)$. $We\ have\ to\ see\ the\ cases\ n_0 \le n < 3n_0$. One more time they are finite and have a maximun

if c ≥ maximun, we are done. if not. get c as the maximun. $This\ proves\ O(n)$

$The\ proof\ for\ \Omega$ is analagous, using min $and\ the\ lower\ bound\ property$

$\boldsymbol{f)\, T(n) = \; 4T\left(\frac{n}{2}\right) + \Theta(n)\ has\ solution\ T(n) = \Theta(n^2)}$

$Suppose\ T(k) \leq ck^2 - \frac{c_1}{3} k for\ n_0 \leq k < n \rightarrow$
$T(n) = 4T\left(\frac{n}{2}\right) + \Theta(n) \leq 4c\left(\frac{n}{2}\right)^2 + c_1 n - \frac{4}{3}c_1 n \leq cn^2 - \frac{1}{3}c_1 n\ but\ this\ is\ valid\ only\ if \frac{n}{2} > n_0$
$We\ need\ to\ verify\ for\ n_0 \leq n < 2n_0.$
$Again\ we\ can\ choose\ c = \max\{T(n); for\ n_0 \leq n < 2n_0\}$
$The\ proof\ for\ lower\ bound\ is\ similar$

$\boldsymbol{4.3.2\ The\ solution\ to\ the\ recurrence\ \ T(n) = 4T\left(\frac{n}{2}\right) + n turns\ out\ to\ be}$
$\boldsymbol{T(n) = \Theta(n^2). Show\ that\ a\ substitution\ proof\ with\ T(n) \leq cn^2\ fails.}$
$\boldsymbol{Then\ show\ how\ to\ subtract\ a\ lower\ order\ to\ make\ a\ substitution\ proof\ work}$

$Suppose\ T(k) \leq ck^2\ for\ n_0 \leq k < n \rightarrow$
$T(n) = 4T\left(\frac{n}{2}\right) + n \leq cn^2 + n\ which\ isn't guarateed\ that\ T(n) \leq cn^2$
$Now\ lets\ subtract\ a\ lower\ term\ and\ we\ have\ T(n) \leq cn^2 - nc_1 - nc_1 + n$
$= cn^2 - nc_1 - n(c_1 - 1) \leq cn^2 - nc_1\ if\ c_1 \geq 1$
$We\ can\ do\ a\ simillar\ analysis\ for\ the\ lower\ bound:$

$\boldsymbol{4.3.3\ The\ recurrence\ T(n) = 2T(n-1) + 1\ has\ the\ solution\ T(n) = O(2^n)}$
$\boldsymbol{Show\ that\ a\ substitution\ proof\ fails\ with\ the\ assumption\ T(n) \leq c2^n is\ contant}$
$\boldsymbol{Then\ show\ how\ to\ subtract\ a\ lower\ order\ term\ to\ make\ a\ substitution\ proof\ work}$

$Suppose\ T(k) \leq c2^k\ for\ n_0 \leq k < n. Then\ we\ would\ have:$
$T(n) = 2T(n-1) + 1 \leq 2 \cdot c \cdot 2^{n-1} + 1 = c \cdot 2^n + 1$
$We\ can\ subtract\ a\ lower\ term, d, what\ leads\ us\ to:$
$T(n) = 2T(n-1) + 1 \leq 2 \cdot c \cdot 2^{n-1}(-2d + 1)$
$If\ we\ choose\ -2d + 1 \leq 0\ we\ are\ done$

$\boldsymbol{4.4.1 For\ each\ of\ the\ following\ recurrences, sketch\ its\ recursion\ tree, and\ guess\ a\ good\ asymptotic\ upper\ bound\ on\ its\ solution. Then\ use\ the\ substitution\ method\ to\ verify\ your\ answer}$

$\boldsymbol{a)\, T(n) = T\left(\frac{n}{2}\right) + n^3}$

$(n)^3 \rightarrow \left(\frac{n}{2}\right)^3 \rightarrow \left(\frac{n}{4}\right)^3 \rightarrow \cdots \rightarrow 1$
$There\ are\ \lg(n) + 1\ of\ size. The\ sum\ of\ the\ work\ done\ on\ each\ node\ is$

$$\sum_{k=0}^{lg(n)} \left(\frac{n}{2^k}\right)^3 = \sum_{k=0}^{lg(n)} \frac{n^3}{8^k} = n^3 \sum_{k=0}^{lg(n)} \frac{1}{8^k} = S.$$

$$Now\ S - \frac{1}{8}S = \sum_{k=0}^{lg(n)} \frac{1}{8^k} - \frac{1}{8^{k+1}} = \frac{7S}{8} = \left(1 - \frac{1}{8^{\lg(n)+1}}\right)n^3 \to S = \frac{1}{7}\left(8 - \frac{1}{8^{\lg(n)}}\right)n^3 \le \frac{8}{7}n^3$$

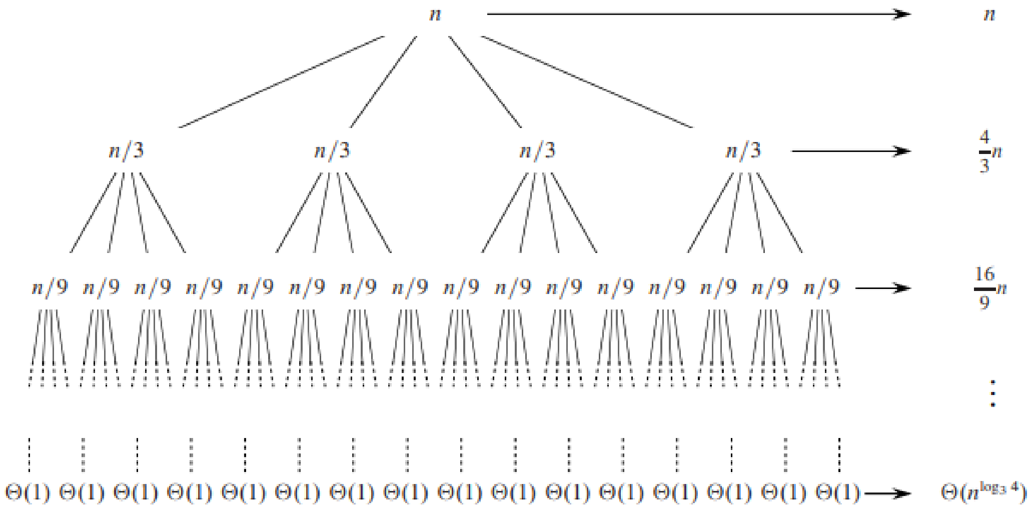
This could give us a better upper bound, but we will guess its only cubic by now

Suppose $T(k) \le ck^3$ *for* $n_0 \le k < n$ *then:*

$$T(n) = T\left(\frac{n}{2}\right) + n^3 \le \left(\frac{c}{8}\right)^1 n^3 + n^3 = \left(1 + \frac{c}{8}\right)n^3. Now\ we\ just\ need\ c\ such\ 1 + \frac{c}{8} \le c$$

This is valid for $\frac{n}{2} \ge n_0$, *we need to verify for* $n_0 \le n < 2n_0$. *We them use the same argument as before, get the* max *of* $T(n)$ *and use as the c*

b) $\boldsymbol{T(n) = 4T\left(\frac{n}{3}\right) + n}$



The height is $log_3(n)$

$$S = \sum_{k=0}^{\log_3 n} n\left(\frac{4}{3}\right)^k \to \frac{4}{3}S = \sum_{k=0}^{\log_3 n} n\left(\frac{4}{3}\right)^{k+1} \to \frac{1}{3}S = n\sum_{k=0}^{\log_3 n}\left(\frac{4}{3}\right)^{k+1} - \left(\frac{4}{3}\right)^k = \left(\frac{4}{3}^{\log_3 n+1} - 1\right)n$$

$$S \le 3n\left(\frac{4}{3}\right)^{\log_3 n+1} \le 4 \cdot 4^{\log_3 n} \to S \le 4n^{\log_3 4}$$

Now lets prove it: Suppose $T(k) \le c_1 k^{\log_3 4} - c_2 n$ *for* $n_0 \le k < n$

$$T(n) = 4 \cdot T\left(\frac{n}{3}\right) + n \le \frac{4c_1 n^{\log_3 4}}{4} + n - \frac{4}{3}c_2 n \le c_1 n^{\log_3 4}\ if\ 1 - \frac{4}{3}c_2 < -c_2$$

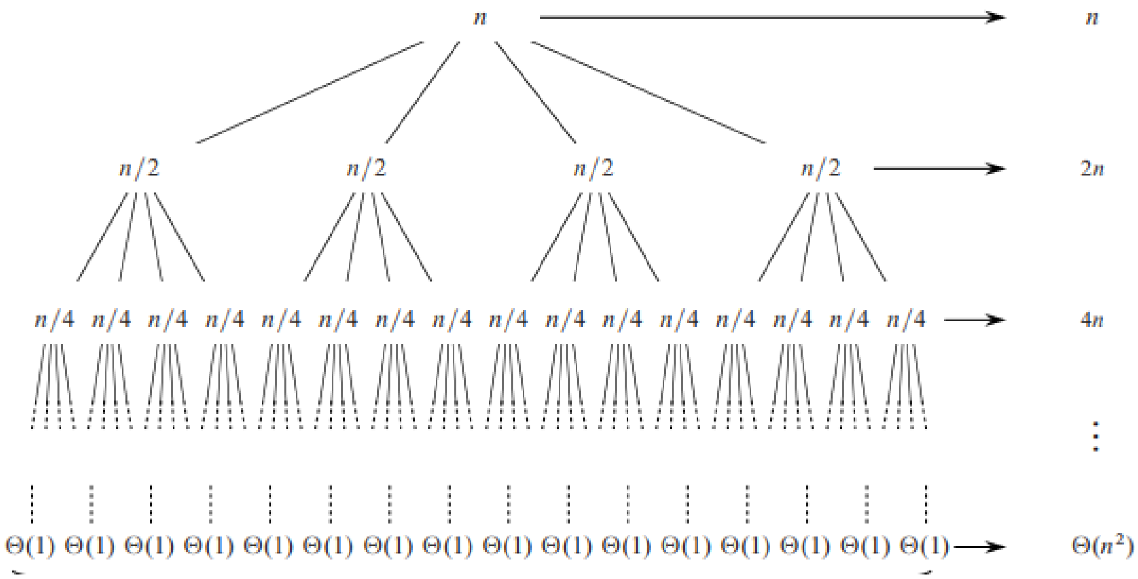
get $c_2 > 3$. *But this is valid only for* $n > 3n_0$.

Now get the c = max *of* $(T(n))$ *for* $n_0 \le n < 3n_0$.

$T(n) \le cn^{\log_3 4}$. *Now we can take a* c_3 *such* $T(n) \le c_3 n^{\log_3 4} \le cn^{\log_3 4} - c_2 n$

Basing on the last resolutions QED

$\boldsymbol{c)\ T(n) = 4T\left(\frac{n}{2}\right) + n \rightarrow}$



The size oh the tree is $\log_2 n + 1$ *the Running time can be estimated as*

$$S = \sum_{k=0}^{\lg(n)} 2^k n \rightarrow 2S - S = S = \sum_{k=0}^{\lg(n)} 2^{k+1}n - 2^k n = \left(2^{\log_2 n+1} - 1\right)n = 2n^2 - n$$

Our guess is that $T(n) = O(n^2)$*So Suppose* $T(k) \le c_1k^2 - c_2k$ *for* $n_0 \le k < n$ *then:*

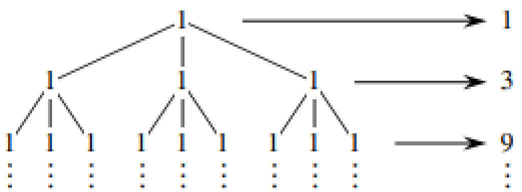
And choose c_2 *such* $4c_2 - 1 = c_2 \rightarrow 3c_2 = 1 \rightarrow c_2 = \frac{1}{3}$

$$T(n) = 4 \cdot T\left(\frac{n}{2}\right) + n \le 4 \cdot c_1\left(\frac{n}{2}\right)^2 + n - 4c_2n = c_1n^2 - n(4c_2 - 1) = c_1n^2 - c_2n$$

if $n_0 \le n < 2n_0$*.Taking* $n_0 = 2$ *we have* $2 \le n < 4 \rightarrow$ *lets analise for* $n = 2$ *and* $n = 3$

$T(2) = 1 \le 4c_1 - \frac{2}{3}$ *and* $T(3) = 1 \le 9c_1 - 1$*.Just take* $c_1 = 1$ *and we are done*

$\boldsymbol{d)T(n) = 3T(n-1) + 1}$



The height clearly will be n and our guess $S = \sum_{k=1}^{n} 3^n \rightarrow 2S = 3^{n+1} - 1 \rightarrow$ *our guess is*

$O(3^n)$*.Suppose* $T(k) \le c_1 3^k - c_2$ *for* $n_0 \le k < n \rightarrow T(n) = 3T(n-1) + 1$

$\le 3c_1 3^{n-1} - 3c_2 + 1 = c_1 3^n - c_2 + (1 - 2c_2) \le c_1 3^n - c_2 \left(\text{choose } c_2 = \frac{1}{2}\right)$

But this is valid for $n_0 \le n - 1 \rightarrow n \ge n_0 + 1$*. But for* $T(n_0) = 1$

$< c_1 3^{n_0} - c_2$*. Choose* $n_0 = 2$ *and a* c_1 *and* $c_1 = 1 \rightarrow T(2) = 1 \le 9 - 1/2$

$\mathbf{4.4.2}$ ***Use the substitution method to prove that reucrrence 4.15 has the asymptotic lower bound*** $\boldsymbol{L(n) = \Omega(n)}$. ***Conclude that*** $\boldsymbol{L(n) = \Theta(n)}$

$$\mathbf{4.15}: \boldsymbol{L(n)} = \begin{cases} \boldsymbol{1, if\,(n < n_0)} \\ \boldsymbol{L\left(\frac{n}{3}\right) + L\left(\frac{2n}{3}\right)} \end{cases}$$

$Suppose\ c_1 k \le L(k) for\ n_0^* \le k < n, take\ n_0 = 5n_0$

$L\left(\frac{n}{3}\right) + L\left(\frac{2n}{3}\right) = L(n)$

$c_1 n = \ c_1 \frac{n}{3} + c_1 \frac{2n}{3} \le L(n)\ for\ 5n_0 \le \frac{n}{3} \to for\ n \ge 15n_0\ let\ analyze\ for$

$5n_0 \le n < 15\ n_0\ \ we\ know\ that\ this\ will\ be\ bounded\ and\ we\ take\ the\ x$

$= \{\min of\ T(n)/\ max\ of\ \ n; 5n_0 \le n < 15\ n_0\}$

$now\ take\ the\ \mathrm{z} = \min x, c_1\ \ and\ we\ know\ that$:

$zn \le L(n)\ for\ 5n_0 \le n < 15\ n_0\ and\ we\ are\ done$

$\mathbf{4.4.3}$ ***Use the substitution method to prove the recurrence*** $\mathbf{4.14}$ ***has the solution*** $\boldsymbol{T(n) = \Omega\big(nlg(n)\big)}$. ***Conclude*** $\boldsymbol{T(n) = \Theta(nlgn)}$

$$\mathbf{4.14}:\ \boldsymbol{T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + \Theta(n)}$$

$Lets\ prove\ the\ lower\ bound\ first. We\ know\ exist\ n_0\ such\ that\ \Theta(n) \le c_1 n\ \forall n \ge n_0$

$Now\ lets\ suppose: T(k) \le cklg(k) - c_3 k\ for\ n_0 \le k < n\ then$:

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + \Theta(n) \le c\frac{n}{3}\lg\left(\frac{n}{3}\right) + c\frac{2n}{3}\lg\left(\frac{2n}{3}\right) + c_2 n - \frac{c_3 n}{3} - \frac{2c_3 n}{3}:$$

$$cn\left(\frac{1}{3}\lg\left(\frac{n}{3}\right) + \frac{2}{3}\log\left(\frac{2}{3}\right)\right) - n(c_2 - c_3) = cn\lg\left(\left(\frac{n}{3}\right)^{\frac{1}{3}}\left(\frac{2n}{3}\right)^{\frac{2}{3}}\right) - n(c_2 - c_3)$$

$$cn\lg\left(\frac{n^{\frac{1}{3}}}{3^{\frac{1}{3}}}\frac{2^{\frac{2}{3}}n^{\frac{2}{3}}}{3^{\frac{2}{3}}}\right) - n(c_2 - c_3) = cn\lg\left(\frac{n2^{\frac{2}{3}}}{3^1}\right) - n(c_2 - c_3) \le cn\lg(n) - n(c_2 - c_3)$$

$Now\ lets\ choose\ c_3 = \frac{c_2}{2} \to T(n) \le cnlg(n) - nc_2\ if\ n \ge 3n_0\ if\ n_0 \le n < 3n_0$.

$let\ c_{max} = \mathbf{100} * \max\big\{\{\mathrm{T(n); n_0 \le n < 3n_0}\}, \mathrm{c, c_2}\big\} \to \mathrm{T(n)} \le \mathrm{c}_{max} \le \mathrm{c_{max}nlg(n) - c_2 n}$

so $\mathrm{T(n) = O\big(nlg(n)\big)}$. We make an analogous proof for the lower bound :

Suppose $\mathrm{cklg(k)} \le T(k)\ for\ 0 \le k < n$

$$c\frac{n}{3}\lg\left(\frac{n}{3}\right) + c\frac{2n}{3}\log\left(\frac{2n}{3}\right) \le c\frac{n}{3}\lg\left(\frac{n}{3}\right) + c\frac{2n}{3}\log\left(\frac{2n}{3}\right) + c_2 n \le=\ T(n)$$

$$c\frac{n}{3}\lg\left(\frac{n}{3}\right) + c\frac{2n}{3}\log\left(\frac{2n}{3}\right) + c_2 n = cn\left(\log\left(\frac{n2^{\frac{2}{3}}}{3^1}\right) = \log(n) + \log\left(2^{\frac{2}{3}}\right) - \log(3)\right) \le \mathrm{T(n)}$$

$$cnlg(n) \le cnlg(n) + \frac{2}{3}cn - cnlog(3) + c_2 n \le T(n)\ if\ \frac{2}{3}cn - cnlog(3) + c_2 n \ge 0$$

$c\left(\frac{2}{3} - \log(3)\right) \ge -\mathrm{c_2} \to \mathrm{c}\left(\log(3) - \frac{2}{3}\right) \le \mathrm{c_2} \to \mathrm{c} \le \frac{\mathrm{c_2}}{\log(3) - \frac{2}{3}}$ and we are done for

$\mathrm{n_0 \le n < 3n_0}\ We\ now\ get\ the \min of\ T(n)/\max n$ and c so $\mathrm{T(n) < c * nlg(n)}$

$\mathbf{4.4.4}$ ***Use a recursion tree to justify a good guess for the solution to the recurrence*** $\boldsymbol{T(n) = T(\alpha n) + T((1-\alpha)n) + \Theta(n)}$***, where*** $\boldsymbol{\alpha}$ ***is a constant such*** $\mathbf{0 < alpha < 1}$

We know that the work done on the childrens are equivalent to the work done on the parent node now we need to find the size of it. Without loss of generality, take $\alpha > 1-\alpha$.
In this case we will have the deepest leaf at the node in which $n\alpha^{height-1} = 1 \rightarrow$

$(height - 1) = \log_a n^{-1} \rightarrow height = \log_a\left(\frac{1}{n}\right) + 1$ *. We do* $\Theta(n)$ *in each level So*

$$(\log_a a - \log_a n)\Theta(n) \rightarrow \left(\log_a \frac{a}{n}\right)cn \rightarrow \frac{\left(\log_{\frac{1}{a}} \frac{a}{n}\right)}{\log_{\frac{1}{a}} a} = \frac{\left(\log_{\frac{1}{a}} a - \log_{\frac{1}{a}} n\right)}{\log_{\frac{1}{a}} a} = \frac{-1 - \log_{\frac{1}{a}} n}{-1}$$

$= \left(1 + \log_{\frac{1}{a}} n\right)cn$ *we will assume its bounded by* $nlg(n)$

Suppose $T(k) \le cklg(k)$ *for* $n_0 \le k < n$
$T(n) = T(\alpha n) + T(n(1-\alpha) + \Theta(\mathrm{n})) \le$
$T(n) \le c(\alpha n)\log \alpha n + c(1-\alpha)\mathrm{n}\lg((1-\alpha))n + c_2 n$
$\le cn[\log(\alpha^{\alpha} n^{\alpha}) + \log(1-\alpha)n^{(1-\alpha)}] + cn$
$T(n) \le cn[\alpha\log(\alpha) + \alpha\log(n) + (1-\alpha)\log(n) + (1-\alpha)\log(1-\alpha)] + c_2 n$
$T(n) \le cn[\log(n) + \alpha\log(\alpha) + \log(1-\alpha)\log(1-\alpha)] + c_2 n \rightarrow$
$T(n) \le cnlog(n) + n[c\,[\log(1-\alpha)\log(1-\alpha) + \alpha\log(\alpha] + c_2)]$ *this is true if we have*:
$[c\,[\log(1-\alpha)\log(1-\alpha) + \alpha\log(\alpha] + c_2)] \le 0 \rightarrow$
$c \ge -\frac{c_2}{[\log(1-\alpha)\log(1-\alpha) + \alpha\log(\alpha])}$ *This value here is a constant, so get it*!
Now to hold we know that $0 \le \Theta(n) \le c_2 n$ *for* $n \ge n_0$*. We will get this* n_0 *be the same as our hipothesis* n_0
We showed by far that if $T(k) \le cklg(k)$ *for* $n_0 \le k < n$ *then* $T(n) \le cnlgn$ *if* $\alpha n \ge n_0$ *and* $(1-\alpha)n \ge n_0$ *get* n^* *as* $\min\{\alpha, (1-\alpha)\}$ *then we proved for* $n_0 \le n < n^* n_0$ *But for the same analysis we did previously we can get the minimun of the running times, compare with the constant c, find minimuns and maximuns and choose the values that guarantess that it all works*

$\mathbf{4.5.1}$ ***Use the master method to give tight asymptotic bounds for the following recurrences***
$\boldsymbol{a)T(n) = 2T\left(\frac{n}{4}\right) + 1}$

$a = 2; b = 4, f(n) = 1.\ 1 = O(n^{\log_4 2 - \mathrm{E}})$*. take* $\mathrm{E} = \frac{1}{3}$ *and this holds* $T(n)$ *is* $\Theta\left(n^{\frac{1}{2}}\right)$
$\mathbf{b)T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n}}$

$a = 2; b = 4, f(n) = \sqrt{\mathrm{n}}$ *There is no* E *such that* $n^{\frac{1}{2}} = O\left(n^{\frac{1}{2}-\mathrm{E}}\right)$ *since every* ϵ *would make the exponent of* $g(n) = n^{\frac{1}{2}-E}$*, less than* $\frac{1}{2}$*. We can prove it by contradiction.*
Suppose that exists such an E*. Then* $\exists c_1$ *and* n_0 *such that* $\forall n \ge_0$
$0 \le n^{\frac{1}{2}} \le c_1 n^{\frac{1}{2}-E} \rightarrow$ *this would imply that* $n^E \le c$ *for all* $n \ge n_0$ *now take* $n > c^{\frac{1}{E}}$
This would imply $n^E > c$*. Thus a contradiction.*

$Lets\ analize\ the\ second\ case\ of\ the\ Master\ Theorem$:
$There\ exists\ a\ constant\ K \geq 0\ such\ that\ n^{\frac{1}{2}} = \Theta\left(n^{\frac{1}{2}} \lg^{k} n\right)?\ Yes\ k = 0$
$This\ leads\ to\ T(n) = \Theta(n^{\frac{1}{2}}\lg(n))$

$\boldsymbol{c)\ T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n}lg^2 n}$

$a = 2; b = 4, f(n) = \sqrt{n} \lg^2 n\ \ There\ exists\ k \geq 0\ suchn^{\frac{1}{2}} \lg^2 n = \Theta\left(n^{\frac{1}{2}} \lg^k n\right), k = 2$
$Then\ T(n) = \Theta(n^{\frac{1}{2}} \lg^3 n)$

$\boldsymbol{d)\ T(n) = 2T\left(\frac{n}{4}\right) + n}$

$f(n) = n \rightarrow n = \Omega\left(n^{\frac{1}{2}+\frac{1}{2}}\right)\left(Here\ we\ take\ \mathrm{E} = \frac{1}{2}\right)\ Additionaly\ 2f\left(\frac{n}{4}\right) \leq cf(n) \rightarrow$
$\frac{n}{2} \leq cn\ for\ c = \frac{2}{3} Then\ T(n) = \Theta(n)$

$\boldsymbol{e)\ T(n) = 2T\left(\frac{n}{4}\right) + n^2}$

$f(n) = n^2 \rightarrow n^2 = \Omega\left(n^{\frac{1}{2}+\frac{3}{2}}\right)\left(Here\ we\ take\ \mathrm{E} = \frac{3}{2}\right)\ Additionaly\ 2f\left(\frac{n}{4}\right) \leq cf(n) \rightarrow$
$\frac{n^2}{8} \leq cn^2\ for\ c = \frac{2}{3} Then\ T(n) = \Theta(n^2)$

$\boldsymbol{4.5.2\ Professor\ Caesar\ wants\ to\ develop\ a\ matrix\ multiplication\ algorithm\ that\ is}$
$\boldsymbol{asymptoptically\ faster\ than\ Strassen\ algorithm. His\ algorithm\ will\ use\ the}$
$\boldsymbol{divide\ and\ conquer\ methodm\ dividing\ each\ matrix\ intro \frac{n}{4} x \frac{n}{4}\ submatrices\ and}$
$\boldsymbol{the\ divide\ and\ combine\ steps\ together\ will\ take\ \Theta(n^2) time. Suppose\ that\ the\ professor}$
$\boldsymbol{Algorithm\ creates\ a\ recursive\ subproblems\ of\ size \frac{n}{4}. What\ is\ the\ lasgest\ integer\ value\ of}$
$\boldsymbol{a}$

$We\ need\ largest\ a\ such\ that \log_4 a < \lg 7 \rightarrow a = 48$

$\boldsymbol{4.5.3\ Use\ the\ master\ method\ to\ show\ that\ th\ solution\ to\ the\ binary\ search\ recurrence}$
$\boldsymbol{T(n) = T\left(\frac{n}{2}\right) + \Theta(1) is\ such\ that\ T(n) = \Theta(\lg(n))}$

$a = 1 b = 2 f(n) = 1. Exists\ e\ such\ that\ 1$
$= O(n^{-E})? No\ we\ could\ prove\ it, but\ its\ obviously\ false$
$for\ case\ 2\ exists\ k \geq such\ that\ 1 = \Theta(1). take\ k = 0 \rightarrow \Theta\ (\lg(n));$

4.5.4 ***Consider the function*** $\boldsymbol{f(n) = \lg(n)}$. ***Argue that although*** $\boldsymbol{f\left(\frac{n}{2}\right) < f(n)}$, ***the regularity condition*** $\boldsymbol{af\left(\frac{n}{b}\right) \leq cf(n)}$ ***with*** $\boldsymbol{a = 1}$ ***and*** $\boldsymbol{b = 2}$ ***does not hold for any constant*** $\boldsymbol{c < 1}$. ***Argue further that for any*** $\boldsymbol{\mathrm{E} > 0}$, ***the condition in case 3 that*** $\boldsymbol{f(n) = \Omega\left(n^{\log_b a + \mathrm{E}}\right)}$ ***does not hold***

$lg\left(\frac{n}{2}\right) \leq clg(n) \rightarrow \lg(n) - 1 \leq clg(n) \rightarrow (1-c)\lg(n) \leq 1.$
This clearly does not hold since $\lg(n)$ *in unbounded*

Suppose it holds for some E, *then* $0 \leq c_1 n^E \leq \lg(n)$. *Then* $\frac{n^E}{\lg(n)} \leq c_2 \; \forall n \geq n_0 \rightarrow$
But $\lim_{n\to\infty} \frac{n^E}{\lg(n)} = \lim_{n\to\infty} \frac{\ln(2) E n^E}{1} = \infty$. *Thus is not bounded above. So there is no* E.

4.5.5 ***Show that for suitable constants a, b and E, the function*** $\boldsymbol{f(n) = 2^{l\lceil g(n)\rceil}}$ ***satisfies all the conditions in case 3 of the master theorem except the regularity condition***

Lets begin by showing that the regularity doesn't hold for any a, b, c $\rightarrow$
$a2^{\lg\left(\frac{n}{b}\right)} \leq a2^{\left\lceil \lg\left(\frac{n}{b}\right)\right\rceil} \leq 2a2^{\log\left(\frac{n}{b}\right)}$
$a2^{\log\left(\frac{n}{b}\right)} = \frac{a}{b} > 2c = 2c2^{\log(n)} \geq$

By Inspection we find thet the numbers $a = 1$ $b = sqrt(2)$ *and* $e = 1$ *guarantees the case 3. Now lets use this numbers in the regularity problem*

$a2^{\left\lceil \lg\left(\frac{n}{b}\right)\right\rceil} = c2^{\lceil \lg n\rceil} \rightarrow 2^{\left\lceil \lg n - \frac{1}{2}\right\rceil} = c2^{\lceil \lg n\rceil}$, *but exists n such that* $n = 2^k$ *for* $k \in Z > 0 \rightarrow$
$2^{\left\lceil \lg 2^k - \frac{1}{2}\right\rceil} = 2^{\left\lceil \lg 2^k - \frac{1}{2}\right\rceil}$ *but* $\left\lceil k - \frac{1}{2}\right\rceil = \lceil k\rceil$ *so c would need to be* > 1 *for this case. So doesn't hold*

4.1 ***Recurrence examples***
Give asymptotic tight upper and lower bounds for $\boldsymbol{T(n)}$ ***in each of the following algorithmic recurrences. Justify your answers***

$\boldsymbol{a)\ T(n) = 2T\left(\frac{n}{2}\right) + n^3}$
We will guess our $\Theta(g(n))$ *based on the tree method. in the first line we have* n^3
In the second line we have $\frac{n^3}{4}$, *in the third line we have* $4\frac{n^3}{4^3} = \frac{n^3}{16}$.
So in each level we can guess it will be $\frac{n^3}{4^{i-1}}$. *We expect that* $\frac{n}{2^{height-1}}$ *is* 1 in the last leaves
$n = 2^{height-1} \rightarrow log_2 n + 1 = height$. *So we expect to have the following* $T(n)$

$$T(n) = \sum_{k=0}^{log_2} \frac{n^3}{4^k} \rightarrow S = n^3 \sum_{k=0}^{log_2} \frac{1}{4^k} \rightarrow \frac{S}{4} = n^3 \sum_{k=0}^{log_2} \frac{1}{4^{k+1}} \rightarrow S - \frac{S}{4} = \sum_{k=0}^{log_{2(n)}} \frac{1}{4^k} - \frac{1}{4^{k+1}} \rightarrow$$

$\frac{3S}{4} = 1 - \frac{1}{4^{log_2(n)+1}} \rightarrow S = \frac{4}{3}\left(1 - \frac{1}{(2^2)^{log_2(n)+1}}\right) = S = \frac{4}{3}\left(1 - \frac{1}{(2^{log_2(n)+1})^2}\right)$

$S = \frac{4}{3}n^3\left(1 - \frac{1}{(2^{log_2(n)} \cdot 2)^2}\right) = \frac{4n^3}{3}\left(1 - \frac{1}{(2n)^2}\right) \rightarrow S = \frac{4n^3}{3} - \frac{n}{3}$

So we will guess that $T(n) = \Theta(n^3)$

Suppose that $T(k) \leq c_1 k^3$ *for* $n_0 \leq k < n$

Then $T(n) = 2T\left(\frac{n}{2}\right) + n^3 = \frac{2c_1 n^3}{8} + n^3 = n^3\left(\frac{1c_1}{4} + 1\right)$ *Now we choose* c_1 *such*

$\frac{c_1}{4} + 1 = c_1 \rightarrow \frac{3c_1}{4} = 1 \rightarrow c_1 = \frac{4}{3}$*. but this is valid for* $2n_0 \leq n$*. We need to verify for*

$n_0 \leq n < 2n_0$*. Lets choose* $n_0 = 2$*. Then* $\mathrm{T}(1) = 1 \leq \frac{4}{3}; T(2) = 1 \leq \frac{32}{3};$

$T(3) = 2 + 27[T(1) + T(2) + n^3] \leq \frac{108}{3}.$

So Assuming that $T(k) \leq \frac{4}{3}k^3$ *for* $2 \leq k < n$ *we can proove that* $T(n)$
$\leq \frac{4}{3}n^3$*. By the strong inductive argument this is true for* 2
$\leq n.$ *Since its true for* $T(2)$

This proves that $T(n) = \mathrm{O}(n^3)$*. Now lets prove that it is* $\Omega(n^3)$:

Suppose that $\frac{4}{3}\ k^3 \leq T(k)$ *for* $n_0 \leq k < n$ *then we would have*:

$T(n) = 2T\left(\frac{n}{2}\right) + n^3 \geq \frac{2 \cdot \frac{4}{3}n^3}{8} + n^3 = \frac{n^3}{3}\ + n^3 = \frac{4}{3}n^3$*. If* $n \geq 2n_0$ *We need to verify for*

$n_0 \leq n < 2n_0$*. Now lets choose* $n_0 = 2$ *again. In this case ne need to verify for* 2 *and* 3

$T(2) = 1 \leq \frac{4 \cdot 8}{3}$ *. Take* c_2 *equals* $\frac{1}{100}$*. Then* $T(2) = 1 \geq \frac{8}{100}$ *and* $T(3) = 1 \geq \frac{27}{100}$

In addition since c_2 *we choosed first was greater than our final* c_2 *our derivation remains valid*
In this case lets rearragen our hypothesis, what leads us to

Suppose that $\frac{1}{100}\ \ k^3 \leq T(k)$ *for* $2 \leq k$
$< n$ *then this would imply that this is valid for* n

Since its true for $T(2)$ *by the strong inductive argument we can guarantee that this holds for all* $n \geq 2$*. So this is* $\Omega(n^3)$*. Since it is* $O(n^3)$ *we can conclude that is* $\Theta(n^3)$

$\boldsymbol{b)\ T(n) = T\left(\frac{8n}{11}\right) + n}$

Lets use the master theorem in this case. We have $a = 1, b = \frac{11}{8}$*, and* $f(n) = n$

Lets analyze case 1*; Does exist a* $E > 0$ *such that* $n = O(n^{-E})$*. No. Suppose exists.*
Then would have the following situation:

$\exists c_1$ *and* $n_0 > 0$ *such that* $\forall n \geq n_0$ *we would have* $n \leq c_1 n^{-e}$
$\rightarrow$ *this would imply that the* $g(n) = \frac{n}{n^{-e}} = n^{1+e}$ *is bounded.*

Which is clearly unbounded. So lets see case 2:

Does exist $K \geq 0$ *such that* n
$= \Theta(lg^k n)$*? Obvisouly no. Lets prove that there is no such* k

Assume there is such K*, then we would have positive constants* $c_1 c_2$ *and* n_0 *that*
$\forall n \geq n_0$ *we would have* $0 \leq c_1 lg^k(n) \leq n \leq c_2 lg(k)(n)$*. This would imply that the*

$Function \dfrac{n}{lg^k(n)}$ $is\ bounded. But\ if\ we\ use\ L'hopitalin\ the\ limit:$

$$\lim_{n\to\infty} \frac{n}{lg^k(n)} = \frac{ln(2)}{k} \cdot \frac{x}{log^{k-1}(n)} = \left(\frac{\ln^2 2}{k\cdot(k-1)}\right) \cdot \frac{x}{log^{k-2}(n)} \ldots After\ k\ derivatives,$$

$we\ could\ prove\ the\ formula\ by\ induction\ on\ k \to = \lim_{n\to\infty} \dfrac{\ln^k 2}{k!}\ x$

$= \infty\ So\ its\ clearly\ unbounded. So\ there\ is\ no\ such\ K > 0$

$Lets\ analyse\ the\ last\ case\ Does\ exist\ a\ constant\ E\ such\ that\ n$

$= \Omega(n^E) and\ if\ f(n) additionally\ satisfies\ af\left(\frac{n}{b}\right)$

$\leq cf(n) for\ some\ constant\ c < 1\ and\ n \geq n_0?$

$Take\ E = 1\ and\ clearly\ n = \Omega(\mathrm{n}) Now\ f\left(\frac{8}{11}n\right) = \frac{8}{11}n \leq cn. Take\ c$

$= \dfrac{9}{11}\ and\ holds\ for\ all\ n.$

$Last\ case\ Holds, so\ we\ have\ T(n) = \ \Theta(\mathrm{n})$

$\boldsymbol{c)\ T(n) = 16T\left(\frac{n}{4}\right) + n^2}$

$Lets\ use\ the\ master\ theorem\ again\ a = 16, b = 4\ f(n) = n^2.$

$Case\ 1. There\ exists\ e > 0\ such\ that\ f(n) = \ O\left(n^{\log_b a - e}\right); n^2$

$= O(n^{2-e})? Clearly\ no. if\ so, we\ would\ have\ that\ n^e\ is\ bounded\ above.$

$Case\ 2. There\ exists\ K \geq 0\ such\ that\ f(n) = \Theta\left(n^{\log_b a} \log^k n\right)? Yes\ k = 0$

$So\ we\ have\ that\ T(n) = \Theta(n^2 \log(n))$

$\boldsymbol{D)\ T(n) = 4T\left(\frac{n}{2}\right) = n^2 \log(n)}$

$We\ will\ use\ the\ master\ theorem\ a = 4, b = 2, \log_b a = 2, and\ f(n) = n^2 \lg(n)$

$case\ 1, there\ exists\ e > 0\ such\ that\ f(n) = O(n^{2-E})? n^2 \lg(n)$

$= O(n^{2-E}) Clearly\ not. If\ we\ had\ such\ E > 0, this\ would\ lead\ us\ that$

$n^E \lg(n)\ is\ bounded, which\ is\ clearly\ false.$

$Case\ 2, there\ exists\ k \geq 0\ such\ thatn^2 \lg(n) = \Theta\left(n^2 \lg^k n\right). Yes, tale\ k = 1, Then$

$T(n) = \Theta(n^2 \log^2 n)$

$\boldsymbol{E)\ T(n) = 8T\left(\frac{n}{3}\right) + n^2}$

$a = 8, b = 3, \log_b a = \log_3 8, f(n) = n^2$

$Case\ 1\ There\ exists\ a\ constant\ such\ that\ n^2 = O\left(n^{\log_3 8 - E}\right)? No. \log_3 8 - E$

$< 2. this\ would\ imply\ that\ n^l\ for\ l$

$> 0\ is\ bounded\ above, which\ is\ clearly\ false$

$Case\ 3. There\ exists\ a\ constant\ E > 0 such\ that\ n^2 = \Omega\left(n^{\log_3 8 + \epsilon}\right)? Yes\ log_3 8 < 2. So$

$Exist\ E > 0\ such\ that \log_3 8 + k = 2\ which\ leads\ us\ to\ n^2\ \Omega(\mathrm{n}^2) lets\ analyze\ regularity$

$8f\left(\frac{n}{3}\right) = \frac{8}{9}n^2 \leq cn^2? take\ c = \frac{8.1}{9}. So\ we\ have\ that\ T(n) = \Theta(n^2)$

$Case\ 2\ There\ exists\ a\ constant\ such\ that\ n^2 = \Theta\left(n^{\log_3 8} \lg^k n\right)?$

$\boldsymbol{F)\ T(n) = 7T\left(\frac{n}{2}\right) + n^2 \lg(n)}$

$a = 7, b = 2, f(n) = n^2 \lg(n)$.

$Case\ 1, there\ exists\ E > 0\ such\ that\ n^2 \lg(n) = O\left(n^{\log_2 7 - E}\right)?\ yes$

$Lets\ find\ such\ c_1 and\ n_0 such\ that\ \forall n \geq n_0 we\ have$

$0 \leq n^2 \lg(n) \leq c_1 n^{\log_2 7 - E}. we\ need\ such\ an\ E\ that\ makes$

$\lim_{n\to\infty} n^{2+E-\log_2 7} \lg(n) \leq c_1. We\ know\ that\ log_2 7 \leq 2.800001 \rightarrow$

$\lim_{n\to\infty} n^{-0.8} \lg(n) \rightarrow \frac{\lg(\mathrm{n})}{n^{0.8}} = 0.$ There is such an E. Then we have $\mathrm{T(n)} = \Theta(\mathrm{n}^{\log_2 7})$

$\boldsymbol{G)\ \mathbf{T(n)} = \mathbf{2T}\left(\frac{\mathbf{n}}{\mathbf{4}}\right) + \sqrt{}(\mathbf{n}}$

Case 1: There exists $\mathrm{E} > 0$ such that $\mathrm{n}^{\frac{1}{2}}$

$$= \mathrm{O}\left(\mathrm{n}^{\frac{1}{2}-E}\right)?$$ No, this would imply that n^{E} is bounded above

$Case\ 3: There\ exists\ E > 0\ such\ that\ n^{\frac{1}{2}}$

$$= \Omega\left(n^{\frac{1}{2}+\mathrm{E}}\right). No. This\ would\ imply\ that\ n^E$$ Is bounded above.

Case 2: There exists K $\geq$ Such that $\mathrm{n}^{\frac{1}{2}} = \Theta\left(n^{\frac{1}{2}} lg^k n\right)? Yes\ take\ k = 0. In\ this\ case\ we\ have$

$In\ this\ case\ we\ have\ T(n) = \ \Theta\left(n^{\frac{1}{2}} \lg(n)\right)$

$\boldsymbol{H) T(n) = \ T(n-2) + n^2}$

$Lets\ guess\ a\ bound\ trhough\ thebound\ method. We\ have\ n^2\ done\ in\ the\ first,$

$(n-2)^2\ in\ the\ second, (n-4)^2\ in\ the\ third. Lets\ guess\ the\ height.$

$We\ have\ that\ (n - 2(height - 1)) = 1 \rightarrow \frac{n+1}{2} = height$

$We\ will\ guess\ that\ T(n) = \Theta(n^3)$

$Suppose\ that\ T(k) \leq c_1 k^3\ for\ \ n_0 \leq k < n\ then\ we\ would\ have:$

$T(n) = T(n-2) + n^2 \leq c_1(n-2)^3 + n^2 \rightarrow$

$T(n) \leq c_1 n^3 - 6c_1 n^2 + 12nc_1 + n^2 \rightarrow we\ want\ -6c_1 n^2 + 12nc_1 + n^2$

$$< 0\ we\ want\ 1 - 6c_1 < 0 \rightarrow c_1 > \frac{1}{6} guarantess\ that\ it\ goes\ to - \infty$$

$T(n) \leq c_1 n^3\ if\ n - 2 \geq n_0\ if\ n \geq n_0 + 2 \rightarrow if\ n_0 \leq n < n + 2.\ We\ need\ to\ analyze$

$Since\ the\ limit\ above\ goes\ to\ infity\ exists\ such\ n_0\ that\ guarantees\ it. We\ want\ now:$

$T(n_0) = 1\ \leq c_1 n_0^3 \rightarrow choose\ c_1 = 1\ and\ n_0 = \max\{previous\ n_0\ and\ 1\}$

$T(n_0 + 1) = 1 \leq T(n_0) \leq c_1(n_0 + 1)^3. This\ proves\ that\ its\ Upper\ bound\ is\ O(n^3).$

$We\ can\ do\ a\ similar\ analysis\ for\ the\ lower\ bound, we\ would\ have\ similarly:$

$c_2 k^3 \leq T(k) if\ n_0 \leq k < n\ then$

$c_2 n^3 \leq c_2 n^3 - 6c_2 n^2 + 12nc_2 + n^2 \leq \mathrm{T(K)} \rightarrow$ This happers if $-6c_2 n^2 + 12nc_2 + n^2$

$$> 0$$ for $\mathrm{n} \geq \mathrm{n}_0$. Just get c_2 such that $1 - 6\mathrm{c}_2 \geq 0\ \mathrm{c}_2 \leq \frac{1}{6}$

Just analyze the $\mathrm{n}_0$necessary for the quadratic equation above be positive always and analyze The values for $\mathrm{n}_0 \leq \mathrm{n} < \mathrm{n}_0 +$

$\boldsymbol{4.2\ Paramter - passing\ costs}$

$\boldsymbol{Throughout\ this\ book, we\ assume\ that\ parameter\ passing\ during\ procedure\ calls\ takes\ constant\ time, even\ if\ - element\ array\ is\ being\ passed.}$

$\boldsymbol{This\ aassumption\ is\ valid\ in\ most\ systems\ because\ a\ pointer\ to\ the\ array\ is\ passed, not\ the\ array\ itself, This\ problem\ examines\ The\ implications\ of\ three\ parameter - passing\ strategies}$:

1. $\boldsymbol{Arrays\ are\ passed\ by\ pointer. Time = \Theta(1)}$
2. $\boldsymbol{Arays\ are\ passed\ by\ copying. Time = \Theta(N), where\ N\ is\ the\ size\ of\ the\ array}$
3. $\boldsymbol{Arrays\ are\ passed\ by\ copying\ only\ the\ subrange\ that\ might\ be\ accessed\ by\ the\ called\ procedure. Time = \Theta(n) if\ the\ subarray\ contains\ n\ elements}$

$\boldsymbol{Consider\ the\ following\ htree\ algorithms}$:

$\boldsymbol{The\ recursive\ binary\ search\ algorithm\ for\ finding\ a\ number\ in\ a\ sorted\ array.}$

$\boldsymbol{The\ Merge - Sort\ procedure}$

$\boldsymbol{Give\ six\ recurrences\ T_{\alpha 1}(N,n) \dots T_{b3}\ for\ the\ worst\ case\ running\ time\ of\ each\ of\ the\ three\ algorithms\ above\ when\ arrays\ and\ matrices\ are\ passed\ using\ each\ of\ the\ three\ parameter - passing\ strategies\ above. Solve\ your\ recurrences\ giving\ tight\ asynptotic\ bounds}$

$\boldsymbol{Recurisve\ Binary}$:

$\boldsymbol{Arrays\ are\ passed\ by\ pointer}$

$$\boldsymbol{T(n) = T\left(\frac{n}{2}\right) + \Theta(1)}$$

$Now\ lets\ use\ the\ master\ theorem\ a = 1, b = 2\ f(n) = \Theta(1)$

$does\ exist\ e > 0\ such\ that\ \Theta(1) = O(n^{-e}).\ Clearly\ no, since\ the\ limit\ of\ n^{-e}\ is\ 0$

$Does\ exist\ k \geq 0\ such\ that\ \Theta(1) = \Theta(\lg^0 n)? Yes. \Theta(1) = \Theta(1). So\ T(n) = \Theta(\lg(n))$

$\boldsymbol{Arrays\ are\ passed\ by\ copying}$

$$\boldsymbol{T(n) = T\left(\frac{n}{2}\right) + \Theta(N)}$$

$Lets\ make\ a\ guess\ through\ the\ tree\ method. In\ the\ first\ node\ we\ have\ \Theta(N)$

$In\ the\ second\ we\ have\ \Theta(N) \dots The\ height\ of\ our\ tree\ is\ such\ that\ \frac{N}{2^{h-1}} = 1 \rightarrow$

$$\log(N) + 1 = h \rightarrow Then\ we\ have\ \sum_{h=1}^{logN+1} \Theta(N) = (\log(N) + 1)\Theta(N)$$

$We\ will\ then\ assume\ it\ isn \log(n)\ . Now\ lets\ prove\ it\ by\ strong\ induction$

$Suppose\ that\ T(k) \leq c_1 klg(k) for\ n_0 \leq k < N$

$$Then\ T(n) = T\left(\frac{n}{2}\right) + \Theta(N) \leq c_1 \frac{N}{2} \lg\left(\frac{N}{2}\right) + c_2 N = c_1 \frac{N}{2} \log(N) - c_1 \frac{N}{2} + c_2 N \leq$$

$$c_1 Nlog(N) - N\left(\frac{c_1}{2} - c_2\right) \leq c_1 Nlog(N). This\ if\ \frac{c_1}{2} - c_2 \leq 0 \rightarrow$$

$Just\ take\ c_1 \leq 2c_2.\ and\ n_o such\ that\ 0 \leq \Theta(N) \leq c_2 N\ \forall n \geq n_0$

$Ok\ but\ this\ to\ happen\ we\ need\ that\ \frac{n}{2} \geq n_0. we\ need\ to\ verify\ if\ it\ holds\ for$

$n_0 \leq n < 2n_0\ By\ the\ same\ argument\ as\ in\ previous\ exercises\ T(n)\ for$

$n_0 \leq\ n < 2n_0\ can\ be\ considered\ as\ constants, we\ can\ get\ the \max of\ them$

$and\ c_1. Now\ we\ do\ similar\ analysis\ for\ lower\ bound$

Arrays are passed by copying only the subrange:

$$\boldsymbol{T(n) = T\left(\frac{n}{2}\right) + \Theta(n)}$$

$Leta\ use\ the\ master\ theorem\ a = 1, b = 2, n^{\log_b a} = 1, f(n) = c_2 n \rightarrow$
$Does\ exist\ e > 0\ such\ that\ c_2 n = \mathrm{O}(\mathrm{n}^{-\mathrm{e}})?$ Obviously no, since this would imply $that\ c_2 n^{1+e}\ would\ be\ bounded\ above. Which\ clearly\ isn't.$
$Does\ exist\ k > 0\ such\ that\ c_2 n = \Theta(\log^k n)?\ No\ This\ would\ implet\ that$
$\lim_{n\to\infty} \frac{c_2 n}{log^k n} = \infty, but\ this\ limit\ is\ 0$
$Case\ 3, does\ exist\ E > 0\ such\ that\ \Theta(n) = \Omega(n^E). Yes\ take\ E = 1$
$Now\ lets\ verify\ the\ regularity\ \Theta\left(\frac{n}{2}\right) \le c_3 n?\ for\ c_3 < 1?\ Not\ necessarly$
$Lets\ guess\ it's\ \Theta(n). our\ work\ in\ each\ node\ is\ n, \frac{n}{2}, \frac{n}{4}..$
$Our\ height\ is\ such\ that\ \frac{n}{2^{height-1}} = 1 \rightarrow height = \log_2 n + 1 \rightarrow$

$$S = \sum_{k=0}^{\log_2 n} \frac{n}{2^i} \rightarrow \frac{1}{2}S = \sum_{k=0}^{\log_2 n} \frac{n}{2^{i+1}} \rightarrow \frac{1}{2}S = n - \frac{n}{2\cdot 2^{\log_2 n}} = n - \frac{1}{2} = 2n - 1$$

$We\ will\ suppose\ that\ T(n) = \Theta(n) \rightarrow So\ suppose\ that\ T(k) \le c_1 k\ for\ n_0 \le k < n$
$T(n) = T\left(\frac{n}{2}\right) + n \le c_1 \frac{n}{2} + n \rightarrow n\left(\frac{c_1}{2} + 1\right). Takec_1\ \frac{c_1}{2} + 1 = c_2 \rightarrow c_2 = 2. Take\ n_0 = 2$
$Now\ we\ need\ to\ analyse\ the\ case\ where\ n_0 \le n < 2n_0 \rightarrow n = 2\ and\ n = 3. In\ this\ cases$
$T(2) = T(3) = 1 \le 2 * 2 \le 2 * 3. So\ its\ proved\ that\ is\ O(n). We\ argue\ similarly$
$For\ the\ lower\ bound\ situation$

Merge Sort:

Arrays are passed by a pointer:

$$\boldsymbol{T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)}$$

$a = 1, b = 1, \log_b a = 1 \rightarrow \Theta(n) = O(n^{1-E})\ Obvisouly\ no, this\ wold\ imply\ that$
$n^{-E}\ is\ bounded\ above, which\ clearly\ isn't. Case\ 2. There\ exist\ k > 0\ such\ that$
$\Theta(n) = \Theta\left(nlog^k(n)\right)? Yes\ takek = 0 \rightarrow T(n) = \Theta(nlog(n))$

Arrays are passed by copying:

$$\boldsymbol{T(n) = 2T\left(\frac{n}{2}\right) + 2\Theta(N) + \Theta(n)}$$

Lets analyzethe work, We know the the height $= \log_2 n + 1$, *the work done is* $2\Theta(N) + c_1 n, \quad 4\Theta(N) + c_1\frac{n}{2}, 8\Theta(N) + c_1\frac{n}{4} \dots$

We will guess then that $S = \sum_{k=1}^{\log_2 n+1} 2^k\Theta(N) + \sum_{k=1}^{\log_2 n+1} c_1\frac{n}{2^{k-1}} =$

$$\mathrm{S} = \Theta(N)\sum_{k=1}^{\log_2 n+1} 2^k + c_1\sum_{k=0}^{\log_2 n}\frac{n}{2^k}$$

$$\frac{1}{2}S = \Theta(N)\sum_{k=1}^{\log_2 n+1} 2^{k-1}\Theta(N) + c_1\sum_{k=0}^{\log_2 n}\frac{n}{2^{k+1}} \to \frac{1}{2}S$$

$$= \Theta(N)\left(\sum_{k=1}^{\log_2 n+1} 2^k - 2^{k-1}\right) + c_1\left(\sum_{k=0}^{\log_2 n}\frac{n}{2^k} - \frac{n}{2^{k+1}}\right) \to$$

$$\frac{S}{2} = \Theta(N)(2^{\log_2 n+1} - 1) + c_1 n\left(1 - \frac{1}{2^{\log_2 n+1}}\right) \to S = \Theta(N)(2n-1) + c_1 n - \frac{c_1}{2}$$

We will suppose that it's N^2, *when beginning*, $n = N$

Suppose $T(k) \le c_1 n^2 - c_4 N - c_5 n$ *for* $n_0 \le k < n$. *Then*

$T(n) = 2T\left(\frac{n}{2}\right) + c_2 N + c_3 n \le \frac{c_1 n^2}{2} - 2c_4 N - c_5 n + c_2 N + c_3 n \le$

$c_1 n^2 - 2c_4 N - c_5 n + c_2 N + c_3 n = c_1 n^2 - (2c_4 - c_2)N - (c_5 - c_3)n.$

$-(2c_4 - c_2)N + c_3 n \le -c_4 N \to -2c_4 N + Nc_2 + c_3 n \le -c_4 N \to$

$c_4 N \ge Nc_2 + Nc_3 \ge Nc_2 + nc_3 \to$ *So if we force* $c_4 N \ge N(c_2 + c_3)$ *we are done.*

So get $c_4 = c_3 + c_2$ *and then* $T(n) \le c_1 n^2 - c_4 N$. *We need to verify the case* $T(n)$ *for* $n_0 \le n < 2n_0$. *We need to choose our* n_o *the* max n_0'*s that guarantees* $0 \le \Theta(N) \le c_4 N$ *and* $0 \le \Theta(n) \le c_3 n$

Now we need to verify $T(n)$ *for* $n \le n < 2n_0$. *Since its a well ordered set, it has a maximun. since* $c_4 N$ *is a constant, there exists* c_1 *that makes* $c_1 n^2$

For some c_1, *now just get the max, between this* c_1 *value and the* max *of the set of running times. With this we prove the upper bound to be* $O(n^2)$

We can prove the lower bound of this in a analogous manner.

Arrays are passed by copying only the subrange

$$\boldsymbol{T(N, n) = 2T\left(N, \frac{n}{2}\right) + \Theta(n) + \Theta(n)}$$

This is equal to $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$ *what is the typical mergesort strategy, by the master theorem we can verify that is* $\Theta\big(nlg(n)\big)$;

4.3 *Solving recurrences with a change of variable*
Sometimes a little algebraic manipulation can make an unknown reccurence similar to one you have seen before. lets sove the recurrence

$\boldsymbol{T(n) = 2T(\sqrt{n}) + \Theta(lg(n))}$

a) Define $\boldsymbol{m = lg(n)}$ and $\boldsymbol{S(m) = T(2^m)}$.
Rewrite recurrence above in termns of m and $\boldsymbol{S(m)}$

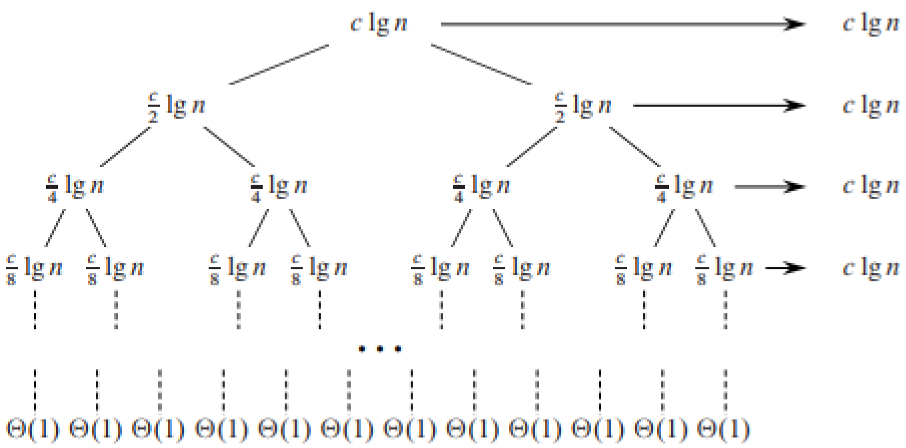
$S(m) = T(2^m) = 2T(\sqrt{2^m}) + \Theta(\lg(2^m)) = S(m) = T(2^m) = 2T(\sqrt{2^m}) + \Theta(\mathrm{m})$
$S(m) = 2S\left(\frac{m}{2}\right) + \Theta(m)$

b) Solve your recurrence for $\boldsymbol{S(m)}$
This recurrence is clearly $m(\lg(m))$

c) Use your recurrence for $\boldsymbol{S(m)}$ to conclude that $\boldsymbol{T(n) = \Theta(\lg(n)\lg(\log(n)))}$
$S(m) = T(2^m) = \log(n) log(\log(n)) = T(n)$

d) Sketch the recursion tree for recurrence above and use it to explain intuitively why The solution is $\boldsymbol{T(n) = \Theta(lg(n)lg(log(n)))}$

The recursion tree is shown below



We will do $c lg(n)$ *in each level and have a* $\lg(lg(n))$ *as height, Why?*
The size of the problems are $n, n^{\frac{1}{2}}, n^{\frac{1}{4}} \ldots$
So we have $n^{\frac{1}{2^{height-1}}} = 2$ *(We don't use 1 because we will never attain it, but we can Guarantee that 2 is the base case)* $\log(\log(n)) + 1 = height$

e) $\boldsymbol{T(n) = 2T(\sqrt{n}) + \Theta(1)}$

Let $n = 2^m$ *and* $S(m) = T(2^m) \rightarrow S(m) = 2T(\sqrt{2^m}) + \Theta(1) = S(m) = 2S\left(\frac{m}{2}\right) + \Theta(1)$
Lets use case 2 of the master theorem Then exist $k \geq 0$ *such that* $\Theta(1) = \Omega(\lg^0(\quad))$
This implies that $S(m) = \Theta(\lg(m)) \rightarrow T(n) = \Theta(\lg(n))$

$\boldsymbol{f)\ T(n) = 3T(\sqrt[3]{n}) + \Theta(\mathbf{n})}$

$We\ will\ use\ an\ analogous\ guess, take\ S(m) = T(3^m) and\ n = 3^m \rightarrow$

$S(m) = T(3^m) = 3T\left(3^{\frac{m}{3}}\right) + \Theta(3^m) \rightarrow S(m) = 3S\left(\frac{m}{3}\right) + \Theta(3^m)$

$Lets\ try\ to\ solve\ this\ recurrene\ using\ the\ tree\ method, In\ the\ first\ level\ we\ have\ 3^m,$

$in\ the\ second\ we\ have\ 3 \cdot 3^{\frac{m}{3}}, in\ the\ third\ level\ we\ have\ 3 \cdot 3 \cdot 3^{\frac{m}{9}}, we\ can\ make\ an$

$induction\ to\ prove\ the\ work\ on\ each\ node, but\ its\ not\ necessarily\ since\ we\ will\ prove\ by$

$Strong\ inuction. Now\ lets\ guess\ the\ height$

$\frac{m}{3^{i-1}} = 1 \rightarrow \log_3 m + 1 = height. So\ the\ sum\ of\ work\ done\ will\ be:$

$S = \sum_{k=0}^{\log_3 m} 3^k \cdot 3^{\frac{m}{3^k}} \rightarrow\ But\ the\ characteristic\ of\ the\ sum, it\ looks\ that\ the\ first\ term$

$is\ the\ most\ important\ so\ we\ will\ guess\ that\ S(m) = \Theta(3^m)$

$Now\ we\ need\ to\ prove\ this\ guess, Suppose\ that\ S(k) \leq c_1 3^m\ for\ n_0^* \leq k < s \rightarrow$

$S(m) = 3S\left(\frac{m}{3}\right) + \Theta(3^m) \leq 3c_1 3^{\frac{m}{3}} + c_2 3^m\ (We\ know, since\ its\ a\ \Theta(3^m) function\ that$

$\exists\ c_2\ and\ n_0\ such\ that 0 \leq \Theta(3^m) \leq c_2 3^m\ \forall n \geq n_0. we\ then\ choose\ n_0^* = n_0 \rightarrow$

$S(m) \leq 3c_1 \frac{3^m}{3^{\frac{2m}{3}}} + c_2 3^m = c_1 3^m \left(\frac{3}{3^{\frac{2m}{3}}} + \frac{c_2}{c_1}\right) \leq c_1 3^m. We\ need\ that\ \left(\frac{3}{3^{\frac{2m}{3}}} + \frac{c_2}{c_1}\right) < 1 \rightarrow$

$\frac{c_2}{c_1} < 1 + \frac{3}{3^{\frac{2m}{3}}} \rightarrow c_1 > \frac{c_2}{1 + \frac{3}{3^{\frac{2m}{3}}}} \rightarrow c_1 > \frac{c_2}{\frac{3^{\frac{2m}{3}} + 3}{3^{\frac{2m}{3}}}} \rightarrow c_1 > \frac{3^{\frac{2m}{3}} c_2}{3^{\frac{2m}{3}} + 3}. Now\ lets\ analyze$

$\lim_{m \to \infty} \frac{3^{\frac{2m}{3}} c_2}{3^{\frac{2m}{3}} + 3} = c_2. This\ means\ that\ there\ is\ a\ n_0^{**}\ such\ that\ if\ we\ take\ c_1$

$= 2c_2\ and\ make\ our\ n_0 = n_0^{**}\ we\ are\ safe\ and\ we\ guarantee$

$that\ , as\ long\ as\ m \geq 3n_0\ S(m) is\ valid, if\ all\ previous\ are\ valid. Now\ we\ need\ to$

$guarantee\ that\ n_0 \leq m$

$< 3n_0\ holds\ the\ situation, again\ we\ can\ use\ the\ well\ odering\ principle, deal\ with\ max's$

$and\ choose\ a\ proper\ c_1, Again\ do\ a\ similarly\ approach\ for\ the\ lower\ bound\ considering\ min's$

$We\ then\ can\ prove\ that\ S(m) = \Theta(3^m) \rightarrow S(m) = T(3^m) = T(n) = \Theta(n)$

$\boldsymbol{4.4\ More\ recurrences\ examples}$
$\boldsymbol{Give\ asynptotically\ tight\ upper\ and\ lower\ bounds\ for\ T(n) in\ each\ of\ the\ following\ recurrences. Justify\ your\ answers}$:

$\boldsymbol{a)\ T(n) = 5T\left(\frac{n}{3}\right) + nlg(n)}$

$a = 5, b = 3. Does\ exist\ e > 0\ such\ that\ nlgn = O\left(n^{\log_3 5 - e}\right)?\ if\ exists\ then:$

$0 \le n^{1+0.4} lg(n) \le c_1 n^{\log_3 5} \leftrightarrow \lim_{n \to \infty} \frac{lg(n)}{n^{0.065}} = 0\,. So\ take\ e = 0.4\ and\ the\ case\ 1\ holds$

$We\ then\ have\ T(n) = \Theta(\mathrm{n}^{\log_3 5})$

$\boldsymbol{b)\ T(n) = 3T\left(\frac{n}{3}\right) + \frac{n}{\lg(n)}}$

$Does\ exist\ e > 0\ such\ that \frac{n}{\lg(n)} = O(n^{-e})?\ if\ exist\ such\ an\ E\ than\ exist\ c_1 such\ that:$

$0 \le \frac{n}{\lg(n)} \le \frac{c_1}{n^e} \rightarrow This\ would\ imply\ that \frac{n^{1+e}}{\lg(n)} \le c_1 but\ this\ isn't\ the\ case, the\ limit\ of$

$the\ last\ fraction\ is\ \infty. Lets\ see\ case\ 2, does\ we\ have\ k \ge 0\ such\ that \frac{n}{\lg(n)} = \Omega\left(n \lg^k n\right).$

$obviously\ not. Lets\ see\ the\ case\ 3. Does\ exist\ e > 0\ such\ that \frac{n}{\lg(n)} = \Theta(n^{1+E}).$

$No. Lets\ make\ a\ guess\ about\ the\ tree. In\ our\ first\ node\ we\ have\ a\ work\ of \frac{n}{\lg(n)},$

$the\ second\ node\ we\ have \frac{n}{\lg\left(\frac{n}{3}\right)}, in\ the\ third\ we\ have \frac{n}{\lg\left(\frac{n}{9}\right)} \ldots The\ height\ is \log_3 n + 1$

$$\sum_{k=0}^{\log_3 n} \frac{n}{\log\left(\frac{n}{3^k}\right)} = n \sum_{k=0}^{\log_3 n} \frac{1}{\log\left(\frac{n}{3^k}\right)} = n \log_3 2 \sum_{k=0}^{\log_3 n} \frac{1}{\log_3\left(\frac{n}{3^k}\right)} = n \log_3 2 \sum_{k=0}^{\log_3 n} \frac{1}{\log_3 n - k} =$$

$n \log_3 2 \sum_{k=0}^{\log_3 n} \frac{1}{k}\ , we\ have\ a\ problem\ for\ the\ leaves. We\ do\ \Theta(n) in\ them, since\ each\ one$

$So\ S = \Theta(n) + n \log_3 2 \sum_{k=1}^{\log_3 n} \frac{1}{k}.$

$We\ will\ guess\ than\ that\ T(n) = O\left(n \sum_{k=1}^{\log_3 n} \frac{1}{k}.\right) \rightarrow$

$T(k) \le c_1 k \sum_{i=1}^{\log_3 k} \frac{1}{i}.\ for\ n_0 \le k < n \rightarrow$

$$T(n) = 3T\left(\frac{n}{3}\right) + \frac{n}{\lg(n)} \le 3c_1 \cdot \frac{n}{3} \sum_{i=1}^{\log_3 \frac{n}{3}} \frac{1}{i}. + \frac{n}{\lg(n)} =$$

$$c_1 n \sum_{i=1}^{\log_3 \frac{n}{3}} \frac{1}{i}. + \frac{n}{\lg(n)} + c_1 n \sum_{i=\log_3 \frac{n}{3}+1}^{\log_3 n} \frac{1}{i}. - c_1 n \sum_{i=\log_3 \frac{n}{3}+1}^{\log_3 n} \frac{1}{i}. =$$

$$c_1 n \sum_{i=1}^{\log_3 n} \frac{1}{i}. + \frac{n}{\lg(n)} - c_1 n \sum_{i=\log_3 \frac{n}{3}+1}^{\log_3 n} \frac{1}{i}. \leq c_1 n \sum_{i=1}^{\log_3 n} \frac{1}{i}.\ if\ \frac{n}{\lg(n)} - c_1 n \sum_{i=\log_3 \frac{n}{3}+1}^{\log_3 n} \frac{1}{i}. < 0$$

$c_1 > \dfrac{1}{\lg(n) \cdot \sum_{i=\log_3 \frac{n}{3}+1}^{\log_3 n} \frac{1}{i}}$. *The limit on the equation in the right goes to P So we can find a suitable* c_1 *given a valid* n_0, *after that we can analyze the values between* $n_0 \leq n < 3n_0$. *We can get the maximun of them and* c_1 *in a way that guarantees the The strong induction hypothesis. Now we have to do the same for the lower bound*

Suppose $T(k) \geq c_2 n \sum_{k=1}^{\log_3 n} \frac{1}{k}.$ *for* $n_0 \leq k < n$

Then we would have $T(n) = 3T\left(\frac{n}{3}\right) + \frac{n}{\lg(n)} \geq 3c_1 \cdot \frac{n}{3} \sum_{i=1}^{\log_3 \frac{n}{3}} \frac{1}{i}. + \frac{n}{\lg(n)} \rightarrow$

$$c_1 n \sum_{i=1}^{\log_3 n} \frac{1}{i}. \leq c_1 n \sum_{i=1}^{\log_3 n} \frac{1}{i}. + \frac{n}{\lg(n)} - c_1 n \sum_{i=\log_3 \frac{n}{3}+1}^{\log_3 n} \frac{1}{i}. \leq T(n)$$

$+\dfrac{n}{\lg(n)} - c_1 n \sum_{i=\log_3 \frac{n}{3}+1}^{\log_3 n} \frac{1}{i} > 0 \rightarrow c_1 \leq \dfrac{1}{\lg(n) + \sum_{i=\log_3 \frac{n}{3}+1}^{\log_3 n} \frac{1}{i}}$. *Agains this limit goes to a P*

We need again analyze the situation where $n_0 \leq n < 3n_0$. *We analyze the* c_1 *with the minimuns running time of the values in this range*

$\boldsymbol{c)\ T(n) = 8T\left(\frac{n}{2}\right) + n^3 n^{\frac{1}{2}}}$

Case 3, *lets analyze, does exist* $e > 0$ *such that* $n^{3.5} = \Omega(n^{3+0.5=e})$ *the regularity*:
$8\left(\frac{n}{2}\right)^{3.5} < cn^{3.5}\ \frac{8}{2^{3.5}} < 1$. *We can find such a* $c < 1$. *So* $T(n) = \Theta(n^{3.5})$

$\boldsymbol{d)\ T(n) = 2T\left(\frac{n}{2} - 2\right) + \frac{n}{2}}$

It looks the merge sort algorithm, maybe the 2 *doesn't make much difference*
Lets suppose $T(k) \leq c_1 k lgk - c_2 k\ \ n_0 \leq k < n$
$T(n) = 2T\left(\frac{n}{2} - 2\right) + \frac{n}{2} \leq 2c_1\left(\frac{n}{2} - 2\right)\log\left(\frac{n}{2} - 2\right) + \frac{n}{2}$:
$(c_1 n - 4c_1)\log\left(\frac{n}{2} - 2\right) + \frac{n}{2} - 2c_2\left(\frac{n}{2} - 2\right) \leq c_1 n log(n) + \frac{n}{2} - c_2 n \leq c_1 n log(n)$
just take $c_2 = 1$. *This is valid for* $2n_0 + 4 \leq n$. *So we need to analyze*:

$n_0 \leq n < 2n_0 + 4$. *By the well ordering principle we can have a maximun*
Based on that we can decide what is the best suitable constant for c_1
Now we need to prove the lower bound:
Suppose $c_2 k lgk \leq T(k)$ *for* $n_0 \leq k < n$ *then we would have*:

$$2c_1\left(\frac{n}{2}-2\right)\lg\left(\frac{n}{2}-2\right)+\frac{n}{2} \leq T(n)$$
$$cnlg\left(\frac{n}{4}\right)-4clg\left(\frac{n}{2}\right)+\frac{n}{2}[if\ n\geq 8] \leq c_1 n\lg\left(\frac{n}{2}-2\right)-4c_1\lg\left(\frac{n}{2}-2\right)+\frac{n}{2} \leq T(n)$$
$$cnlg(n)-2cn-4clgn+4c+\frac{n}{2} \leq cnlg\left(\frac{n}{4}\right)-4clg\left(\frac{n}{2}\right)+\frac{n}{2} \leq T(n)$$
$$cnlg(n) \leq T(n) as\ long\ as\ -2cn-4clgn+4c+\frac{n}{2} \geq 0 -.$$
$$\frac{n}{(4n-8+8\lg(n))} \geq c\ If\ we\ analyze\ the\ limit\ as\ n\to\infty\ we\ see\ its\ 0.25. So\ get$$
$$c=\frac{1}{10}\ and\ this\ will\ hold. Nos\ we\ have\ to\ analyze\ the\ cases\ where\ n_0 \leq n < 2n_0+4$$
We need to analyze the minimun and choose a proper c based on them

$$\boldsymbol{e)T(n)=2T\left(\frac{n}{2}\right)+\frac{n}{\lg(n)}}$$

It's analogous to the problem b, but now we are in base 2. the height is $\log_2 n+1$

In each level we have the following work done: $\frac{n}{\lg(n)} \to \frac{n}{\lg\left(\frac{n}{2}\right)} \to \frac{n}{\lg\left(\frac{n}{4}\right)} \ldots$

So we have the following sum for all levels:

$$\sum_{k=1}^{\log_2 n+1} \frac{n}{\lg\left(\frac{n}{2^{i-1}}\right)}, we\ just\ have\ a\ problem\ at\ the\ last\ level\ that\ has\ \Theta(1) cost\ and\ there\ are\ n$$

$$So\ we\ adjust\ the\ equation\ for\ \Theta(n)+\sum_{k=1}^{\log_2 n}\frac{n}{\lg\left(\frac{n}{2^{i-1}}\right)}=\Theta(n)+\sum_{k=0}^{\log_2 n-\mathbf{1}}\frac{n}{\lg\left(\frac{n}{2^{i}}\right)}=$$
$$n\sum_{k=0}^{\log_2 n-\mathbf{1}}\frac{1}{\lg(\mathrm{n})-\mathrm{i}}=n\sum_{k=0}^{\log_2 n-\mathbf{1}}\frac{1}{i}.$$

From here we have the same analysis as the b) case. The limit of this converges to a constant too

$$\boldsymbol{f)T(n)=T\left(\frac{n}{2}\right)+T\left(\frac{n}{4}\right)+T\left(\frac{n}{8}\right)+n}$$

Lets guess the Complexity of this recurrence. In the first level we do n

In the second we do $\frac{n}{2}+\frac{n}{4}+\frac{n}{8}$

$$=\frac{7n}{8}. In\ the\ third\ we\ do\ \left(\frac{7}{8}\right)^2 n. This\ is\ not\ a\ complete\ tree$$

We can have an upper and lower bound for the levels the all left leafs are $\frac{n}{2^{i-1}}$ *What gives us* $\log_2 n+1$ *. and for the lower bound we have* $log_8 n+1$

Lets analyze both sums:

$$S=n\sum_{k=0}^{\log_2 n}\left(\frac{7}{8}\right)^k \to \frac{\mathbf{7S}}{\mathbf{8}}=n\sum_{k=0}^{\log_2 n}\left(\frac{7}{8}\right)^{k+1} \to \boldsymbol{S}-\frac{\mathbf{7S}}{\mathbf{8}}=\frac{\boldsymbol{S}}{\mathbf{8}}=\sum_{k=0}^{\log_2 n}\left(\frac{7}{8}\right)^k-\left(\frac{7}{8}\right)^{k+1}=$$
$$S=8\cdot\left(1-\left(\frac{7}{8}\right)^{log_2 n+1}\right)n=8-\frac{8\cdot 7\cdot 7^{log_2 n}}{8\cdot(2^3)^{log_2 n}}=8n-\frac{7\cdot n^{2.8}}{n^2}\to\Theta(n)$$

$$Now\ lets\ analyze\ with\ the\ upper\ bound\ S=8\cdot\left(1-\left(\frac{7}{8}\right)^{log_8 n+1}\right)n=8n-7\cdot n^{0.93}.$$

$Our\ guess\ is\ that\ T(n)\ is\ \Theta(n). Let's\ make\ a\ strong\ inductive\ hypothesis$
$Suppose\ that\ T(k) \le c_1 n\ for\ n_0 \le k < n\ then:$
$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + T\left(\frac{n}{8}\right) + n \le \frac{c_1 n}{2} + \frac{c_1 n}{4} + \frac{c_1 n}{8} + n = n\left(\frac{c_1}{2} + \frac{c_1}{4} + \frac{c_1}{8} + 1\right) =$
$nc_1\left(\frac{7}{8} + \frac{1}{c_1}\right)\ take\ c_1 = 8\ Now\ we\ need\ to\ verify\ for\ n_0 \le n < 8n_0 \rightarrow$

$n_0. Again\ we\ can\ take\ c_1 = \max\{T(n), for\ n \in [n_0, 8n_0), c_1\}$
$In\ this\ way\ we\ guarantee\ that\ T(n) \le c \le cn\ and\ that\ nc_1 \le nc$

$We\ need\ to\ validate\ the\ lower\ bound\ now. Our\ guess\ will\ be\ the\ same\ have, so\ we\ have:$
$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + T\left(\frac{n}{8}\right) + n \ge \frac{c_1 n}{2} + \frac{c_1 n}{4} + \frac{c_1 n}{8} + n = n\left(\frac{c_1}{2} + \frac{c_1}{4} + \frac{c_1}{8} + 1\right) =$
$nc_1\left(\frac{7}{8} + \frac{1}{c_1}\right)\ take\ c_1 = 8\ Now\ we\ need\ to\ verify\ for\ n_0 \le n < 8n_0 \rightarrow$

$take\ c = \min\left\{\frac{T(n)}{n}, for\ n \in [n_0, 8n_0), c_1\right\} then \frac{T(n)}{n} \ge c \rightarrow cn \le T(n) and\ cn \le c_1 n$
$So\ we\ proved\ that\ T(n) is\ \Theta(n)$

$\boldsymbol{g)\ T(n) = T(n-1) + \frac{1}{n}}$

$Lets\ make\ our\ tree\ again, it\ clearly\ has\ n\ as\ height. The\ work\ done\ in\ each\ level\ is \frac{1}{n}$

So our guess will be $\mathrm{T(k)} \le c_1 \sum_{i=1}^{k} \frac{1}{i}$ for $\mathrm{n_0 \le k < n}$

$T(n) = T(n-1) + \frac{1}{n} \le c_1 \sum_{i=1}^{n-1} \frac{1}{i} + \frac{1}{\mathrm{n}}$. Take $\mathrm{c_1} = 1$, then $\mathrm{T(n)} \le \mathrm{c_1} \sum_{i=1}^{n} \frac{1}{i}$
$For\ the\ lower\ bound\ we\ have\ the\ same\ case\ and\ the\ proof\ is\ straight\ forward$
$And\ again, we\ need\ to\ check\ n_0 \le n < n_0 + 1. So\ we\ just\ need\ to\ check\ the\ base\ case,$
$That, since\ its\ an\ algorithmic\ recursion, is\ \Theta(1)$

$\boldsymbol{h)\ T(n) = T(n-1) + \lg(n)}$

$Again\ we\ have\ n\ as\ height\ but \log_2 n\ \ as\ work. Our\ guess\ will\ be:$
$T(k) \le c_1 \sum_{i=1}^{k} \log_2 i = \log_2\left(\prod_{i=1}^{k} i\right) \rightarrow T(k) \le c_1 \log_2 k! \le c_1 klog(k)$
$Ok, so\ suppose\ that\ T(k) \le c_1 klog(k)\ for\ n_0 \le k < n.\ \ Now\ we\ have:$
$T(n) = T(n-1) + \log(n) \le c_1(n-1)\log(n-1) + logn \rightarrow$
$c_1 nlog(n-1) - c_1 \log(n-1) + \log(n) \le c_1 nlog(n-1) + \log(n)$
$\le c_1 \log(n-1) + \log(n) \le \log\big(n(n-1)\big) \le \log(n!) \le nlog(n)$
$Just\ take\ c_1. We\ again\ would\ need\ to\ analyze\ for\ n_0 \le n$
$< n_0 + 1, which\ is\ the\ base\ case\ and\ will\ clearly\ hold$

$Now\ we\ need\ to\ analyze\ the\ lower\ bound. So\ suppose\ that\ c_2 \mathrm{klog}(k) \le T(k)$
$for\ n_0 \le k < n \rightarrow$
$T(n) = T(n-1) + \log(n) \ge c_2(\mathrm{n}-1)\log(n-1) + \log(n)$
$c_2(\mathrm{n}-1)\log(n-1) + \log(n) \le T(n) \rightarrow c_2 nlog(n-1) - c_2 \log(n-1) + \log(n) \le T(n)$
$Take\ \ n_0\ such\ that$

$c_2\log(n) \le c_2(\mathrm{n}-1)\log(n-1) + \log(n) \le T(n). Take\ c_2 = 0.1\ and\ n_0 \ge 2$
$Again\ we\ need\ to\ analyze\ for\ the\ case\ n_0 \le n < n_0 + 1\ which\ is\ just\ the\ base\ case$
$which\ is\ equal\ to\ 1, so\ we\ have 0.1\log(2) \le \Theta(2) = 1. So\ it\ holds$

$\boldsymbol{i)\ T(n) = T(n-2) + \frac{1}{\lg n}}$

$Lets\ analyze\ the\ recursion\ tree\ of\ the\ problem\ above.\ We\ have\ n - 2^{i-1}$
$for\ the\ i-th\ level. Then\ n - 2(i-1) = 2 \rightarrow n = 2height \rightarrow height = \frac{n}{2}$
$Now\ the\ work\ done\ on\ level\ 1\ is \frac{1}{\lg(n)}, \text{level } 2 = \frac{1}{\log(n-2)}, \text{in level } 3 \rightarrow \frac{1}{\log(n-4)} \rightarrow$
$\sum_{k=1}^{\frac{n}{2}} \frac{1}{\log(n-2(k-1))}. = \sum_{k=0}^{n-2} \frac{1}{\log(2+k)}. \text{The smallest term is} \frac{1}{\lg(n)} \text{ and there are} \frac{\mathrm{n}}{2} \text{ items}$
$We\ will\ assume\ that\ T(n) = \Theta\left(\frac{n}{\lg n}\right)$
$So\ suppose\ that\ T(k) \le \frac{c_1 k}{\log(k)}\ for\ n_0 \le k < n\ then\ we\ would\ have:$
$\mathrm{T(n) = T(n-1)} + \frac{1}{\lg(n)} \le \frac{\mathrm{c_1(n-2)}}{\log(n-2)} + \frac{1}{\log(n)} \le \frac{\mathrm{c_1 n}}{\log(n)}. \text{Take } \mathrm{c_1 = 10 \text{ and } n > n_0 = 5}$
$Now\ we\ need\ to\ analze\ for\ n_0 \le n$
$< 5n_0.\ By\ the\ well\ ordering\ principle\ there\ exist\ such\ c$
$the\ n_0 can\ remain, since\ the\ function\ on\ the\ right\ will\ grow\ fast\ for\ c_1, it\ will\ obviously\ grow$
$faster\ for\ a\ c_1\ bigger$

$Now\ lets\ prove\ the\ lower\ bound:$
$\frac{\mathrm{c_2 n}}{\log(n)} \le \frac{\mathrm{c_1(n-2)}}{\log(n-2)} + \frac{1}{\log(n)} \le \mathrm{T(n)}. Take\ c_2 = 10^{-5}\ and\ it\ hold\ for\ n > 3$
$Again\ we\ find\ a\ c_2 lower\ than\ what\ is\ need\ to\ contemplate\ the\ cases$
$for\ n_0 \le n < 3n_0$ and take it. the n will still hold
$\boldsymbol{j)T(n) = \sqrt{n}T(\sqrt{n}) + n}$
$we\ again\ will\ use\ the\ guess\ by\ the\ recursive\ tree\ call\ method. We\ wont\ achieve\ T(1).$
$So\ we\ will\ choose\ T(2) as\ our\ base\ case.\ then\ we\ have\ n^{\frac{1}{2^{height-i}}} = 2 \rightarrow$

$n = 2^{2^{height-1}} \rightarrow \log(n) = 2^{height-1} \rightarrow \log(\log(n)) = height - 1 \rightarrow height =$
$\log(\log(n)) + 1. Now\ our\ summation: is\ n\ in\ the\ first\ level\ n\ in\ the\ second$
$at\ level\ 3\ we\ have\ \mathrm{n}^{\frac{1}{4}} n^{\frac{1}{2}} = n^{\frac{3}{4}}\ \ nodes, each\ doing\ \mathrm{n}^{\frac{1}{4}}\ so\ n.$
$$\sum_{k=1}^{\log(\log(n))+1} n \rightarrow nlog(\log(n))$$
$Now\ we\ will\ make\ our\ inductive\ Hypothesis\ T(k) \le c_1 klog(\log(k)) for\ n_0 \le k < n:$
$T(n) = \sqrt{n}\,T(\sqrt{n})\ \ + n \le \sqrt{n} c_1\, \mathrm{n}^{\frac{1}{2}} \log\left(\frac{1}{2}\log(n)\right) + n =$
$c_1\, \mathrm{n}\log(\log(n)) - c_1 \mathrm{n} + \mathrm{n} \le c_1\, \mathrm{n}\log(\log(n)).\ if\ n > 2$
$So\ we\ just\ need\ to\ verify\ the\ cases\ n_0 \le n < 2n_0 \rightarrow T(2) = 1\ and\ T(3) = 1$
$The\ lower\ bound\ is\ similar$

$\boldsymbol{4.5\ Fibonacci\ number}$
$\boldsymbol{This\ problem\ develops\ properties\ of\ the\ Fibonacci\ numbers, which\ are\ defined\ by\ recurrence.}$
$\boldsymbol{We\ will\ explore\ the\ technique\ of\ generating\ functions\ to\ solve\ the\ Fibonacci\ recurrence}$
$\boldsymbol{Define\ the\ generating\ function\ F\ as}$

$$\boldsymbol{F(z) = \sum_{i=0}^{\infty} F_i z^i = 0 + z + z^2 + 2z^3 + 3z^3 + 5z^5 + 8z^6}$$

$\boldsymbol{a)\ Show\ that\ F(z) = z + z\big(F(z)\big) + z^2 F(z) \rightarrow}$

$$z + z\big(F(z)\big) + z^2F(z) = z + \sum_{i=1}^{\infty} F_{i-1}z^i + \sum_{i=2}^{\infty} F_{i-2}z^i \rightarrow z + F_0 z + \sum_{i=2}^{\infty} F_{i-1}z^i + F_{i-2}z^i \rightarrow$$

$$= z + F_0 z + \sum_{i=2}^{\infty} z^i(F_{i-1} + F_{i-2}) = F_1 z + F_0 z + \sum_{i=2}^{\infty} F_i z^i = \sum_{i=0}^{\infty} F_i z^i = F(z)$$

$\boldsymbol{b)\ Show\ that\ F(z) = \dfrac{z}{1 - z - z^2} = \dfrac{1}{\sqrt{5}}\left(\dfrac{1}{1 - \phi z} - \dfrac{1}{1 - \hat{\phi} z}\right)}$

$We\ showed\ that\ F(z) = z + z\big(F(z)\big) + z^2F(z) \rightarrow$

$$F(z) - z\big(F(z)\big) - z^2F(z) = z = F(z) = \frac{z}{1 - z - z^2}$$

$the\ roots\ of\ are\ the\ golden\ ration\ as\ showed\ in\ previous\ exercises$

$$F(z) = \frac{z}{(1-\phi z)(1-\bar{\phi} z)} = \frac{A}{(1-\phi z)} - \frac{B}{(1-\bar{\phi} z)} = \frac{A(1-\bar{\phi} z) - B(1-\phi z)}{(1-\phi z)(1-\bar{\phi} z)} =$$

$$\frac{A - B + B\phi z - A\bar{\phi} z}{(1-\phi z)(1-\bar{\phi} z)} \rightarrow A\ must\ be = B \rightarrow \frac{zB(\phi - \bar{\phi})}{(1-\phi z)(1-\bar{\phi} z)} = \frac{zB\sqrt{5}}{(1-\phi z)(1-\bar{\phi} z)} \rightarrow$$

$$B = \frac{1}{\sqrt{5}} \rightarrow F(z) = \frac{1}{\sqrt{5}}\left(\frac{1}{(1-\phi z)} - \frac{1}{(1-\bar{\phi} z)}\right)$$

$\boldsymbol{c)Show\ that}$

$$\boldsymbol{F(z) = \sum_{i=0}^{\infty} \frac{1}{\sqrt{5}}(\phi^i - \bar{\phi}^i)z^i}$$

$$F(z) = \frac{1}{\sqrt{5}}\left(\frac{1}{(1-\phi z)} - \frac{1}{(1-\bar{\phi} z)}\right) = \frac{1}{\sqrt{5}}\sum_{i=0}^{\infty}(\phi z)^1 - \sum_{i=0}^{\infty}(\bar{\phi} z)^1$$

$$\rightarrow F(z) = \sum_{i=0}^{\infty} \frac{1}{\sqrt{5}}(\phi^i - \bar{\phi}^i)z^i$$

$\boldsymbol{d)Use\ part\ c\ to\ prove\ that\ F_i = \dfrac{\phi^i}{\sqrt{5}}\ for\ i > 0, rounded\ to\ the\ nearest\ integer}$

$$\sum_{i=0}^{\infty} \boldsymbol{F_i z^i} = \sum_{i=0}^{\infty} \frac{1}{\sqrt{5}}(\phi^i - \bar{\phi}^i)z^i \rightarrow \sum_{i=0}^{\infty} z^i\left(F_i - \frac{1}{\sqrt{5}}(\phi^i - \bar{\phi}^i)\right) = 0 \rightarrow$$

$$F_i - \frac{1}{\sqrt{5}}(\phi^i - \bar{\phi}^i) = 0 \rightarrow F_i = \frac{1}{\sqrt{5}}\phi^i - \frac{1}{\sqrt{5}}\bar{\phi}^i \rightarrow$$

$Now\ we\ now\ that\ \bar{\phi}^i = -0.61 \rightarrow |\phi|^i < 1 \rightarrow$

$$\left|F_i - \frac{1}{\sqrt{5}}\phi^i\right| = \left|-\frac{1}{\sqrt{5}}\bar{\phi}^i\right| \le \frac{1}{\sqrt{5}} < \frac{1}{2} \rightarrow \left|F_i - \frac{1}{\sqrt{5}}\phi^i\right| < \frac{1}{2} \rightarrow -\frac{1}{2} < F_i - \frac{1}{\sqrt{5}}\phi^i < \frac{1}{2} \rightarrow$$

So rounding to the nearest integer gives us F_i

e) Prove that $F_{i+2} \geq \phi^i$

By the previous exercise we know that $F_{i+2} \geq \frac{\phi^{i+2}}{\sqrt{5}} - \frac{1}{2} = \phi^i\left(\frac{\phi^2}{\sqrt{5}} - \frac{1}{2\phi^i}\right) \rightarrow$

$F_{i+2} \geq \phi^i\left(\frac{\phi^2}{\sqrt{5}} - \frac{1}{2\phi^i}\right)$. *Now* $1 \leq \left(\frac{\phi^2}{\sqrt{5}} - \frac{1}{2\phi^i}\right) \leq \frac{\phi^2}{\sqrt{5}}$ *for some* $i \rightarrow$

$F_{i+2} \geq \phi^i\left(\frac{\phi^2}{\sqrt{5}} - \frac{1}{2\phi^i}\right) \geq \phi^i$. *if* $i > 3$ *the condition above is valid; if* $i = 0,1,2 \rightarrow$

$F_2 = 1 \geq 1. F_3 = 2 > 1.61, F_4 = 3 > \phi^2$ *So its proved*

4.6 *Chip testing*

Professor Diogenes has n supposedly identical integrated − circuit that in principle are capable of testing each other.
The professor's test jig accommodates two chipts at a a time.
When the jig is loaded, each chip tests the other and reports wheter it is good or bad, but the professor cannot trust the answer of a bad chip. Thus, the four possible outcomes of a test are as follows:

Chip A says	Chip B says	Conclusion
B is good	A is good	both are good, or both are bad
B is good	A is bad	at least one is bad
B is bad	A is good	at least one is bad
B is bad	A is bad	at least one is bad

a) Show that if at least $\frac{n}{2}$ ***chips are bad, the professor cannot necessarily determine which chips are good using any strategy based on this kind of pairwise test***
Assume that the bad chips can conspire to fool the professor

Suppose that there are $\frac{n}{2}$ *goods and* $\frac{n}{2}$ *bads.*
Each good chip will say that there is $\frac{n}{2} - 1$ *good chips and* $\frac{n}{2}$ *bad chips.*
the bad chip although can say them same. Being totally
Symetric, in this case there is no way to determine who is good and who is bad

b) Show that $\left\lfloor\frac{n}{2}\right\rfloor$ ***pairwise tests are sufficient to reduce the problem to one of nealy half the size. That is, show how to use*** $\left\lfloor\frac{n}{2}\right\rfloor$ ***pairwise tests to obtain a set with at most*** $\left\lceil\frac{n}{2}\right\rceil$ ***chips that still has the property that more than half of the chips are good***

Note that since we have more good chips than bad, if we take out one good and one bad chips this property remains the same guarantee that n is even so take one aside if needed
Now we will deal with $\left\lfloor\frac{n}{2}\right\rfloor$ *pairwise comparisons choosing 2 chips at a time*
If they don't say that both are good, them throw them away or you take 1 good and 1 bad

What will make remain the property. Or you take out 2 bads, what still remain the property
Now if both says the other is good, or both is good or both is bad
In this situation we discard only one chip, maybe you will discard a good chip. But since
There is a pair with two goods, in this case maybe you remain with the
same number of goods and bads, but in this situation the worst that can happen is that
in the remaining pairs to be analyzed there are more bads than goods. But in this situation
We necessarily have a pair formed by two bads, this will make us discard a bad and rebalance
the good we discarded early

c) Show how to apply the solution to part b recursively to identify one good chip.
Give and solve the recurrence that describes the number of tests needed to identify one good chip.

By the technique we described in the previous question, we can make $\frac{n}{2}$ *pairwise comparisons reducing the size of the set to at most* $\left\lceil\frac{n}{2}\right\rceil$ *through* $\left\lfloor\frac{n}{2}\right\rfloor$ *comparisons. so we have*

$$T(n) = T\left(\left\lceil\frac{n}{2}\right\rceil\right) + \left\lfloor\frac{n}{2}\right\rfloor$$

We will ignore the floors and ceilings, consider n to be 2^k. *So, lets apply the master theorem case 3 there exists* $e > 0$ *such that* $\frac{n}{2} = \Omega(n^e)$? *Yes take* $e = 1$

Now lets analyze the regularity test: $\frac{n}{4} \leq \frac{c_1 n}{2}$, *yes take* $c_1 = \frac{1}{8}$. *So we have that*

$$T(n) = \Theta\left(\frac{n}{2}\right) = \Theta(n)$$

d) Show how to identify all the good chips with an additional $\boldsymbol{\Theta(n)}$ ***pairwise tests***
Since you know a good chip, it will always say the truth, so just iterate over all other chips

---

4.7 *Monge arrays an m x n array A of real numbers is a Monge Array if for all, i, j, k and l, such that* $\boldsymbol{1 \leq i < k \leq m}$ ***and*** $\boldsymbol{1 \leq j < l \leq n}$:
$\boldsymbol{A[i,j] + A[k,l] \leq A[i,l] + A[k,j]}$
In other words, whenever we pick two rows and two column of a Monge array and consider the four elements at the intersection of the rows and the column, the sum of the upper left and lower right elements is less than or equal to the sum of the lower left and upper right elements

a) Prove that an array is Monge if and only if for all $\boldsymbol{i = 1, 2 \ldots m-1}$ ***and*** $\boldsymbol{j = 1, 2 \ldots . n-1}$ ***we have***
$\boldsymbol{A[i,j] + A[i+1,j+1] \leq A[i,j+1] + A[i+1,j]}$

if the matrix is Monge then its true, as shown belo:

let $i + 1 = k$ *and* $j + 1 = l \rightarrow$
$A[i,j] + A[k,l] \leq A[i,l] + A[k,j]$ *and we have* $1 \leq i < k \leq m$ *and* $1 \leq j < l \leq n$

Now lets make the other way. if
$A[i,j] + A[k,l] \leq A[i,l] + A[k,j]$ *and we have* $1 \leq i < k \leq m$ *and* $1 \leq j < l \leq n$ *then*
$1 \leq i < k \leq m$ *and* $1 \leq j < l \leq n: \rightarrow A[i,j] + A[k,l] \leq A[i,l] + A[k,j]$

We know that $A[k,j] + A[k+1,j+1] \leq A[k,j+1] + A[k+1,j] \rightarrow$
take $k = i + n$. *Suppose holds for n:*

$A[i,j] + A[k,j+1] \leq A[i,j+1] + A[k,j]$
$Then\ we\ have\ A[i,j] + A[k,j+1] + A[k,j] + A[k+1,j+1]$
$\leq A[k,j+1] + A[k+1,j] + A[i,j+1] + A[k,j] \rightarrow$
$A[i,j] + A[k+1,j+1] \leq A[k+1,j] + A[i,j+1] \rightarrow$
$A[i,j] + A[i+n+1,j+1] \leq A[i+n+1,j] + A[i,j+1]$
$\rightarrow So\ it\ implies\ it\ is\ a\ Monge\ array$

$\boldsymbol{b) THe\ following\ array\ is\ not\ Monge. Change\ one\ element\ in\ order\ to\ make\ it\ a\ Monge}$

$We\ know\ that\ if\ it\ A[i,j] + A[i+1,j+1]$
$\leq A[i,j+1] + A[i+1,j], holds\ then\ its\ monge$
$By\ inspection\ on\ each\ element\ we\ find\ that$
$23 + 7 \leq 22 + 6?\ No.\ but\ if\ we\ change\ 22\ to\ 24\ it\ holds\ for\ every\ other\ pair$

$\boldsymbol{c) Let\ f(i) be\ the\ index\ of\ the\ column\ containing\ the\ leftmost\ minimum\ element}$
$\boldsymbol{of\ row\ i. Prove\ that\ f(1) \leq f(2) \leq \cdots f(m) for\ any\ mxn\ Monge\ array}$

$For\ a\ monge\ Array\ we\ have:$
$A[i,j] + A[k,l] \leq A[i,l] + A[k,j]$
$Now\ suppose\ that\ a_j\ and\ row\ b_k\ are\ leftmost\ minimal\ element\ of\ row$
$a\ and\ b\quad and\ j > k (they\ represent\ the\ columns) and\ a$
$< b (In\ other\ words, suppose\ that\ the\ assumption\ we\ want\ to\ prove\ is\ false) \rightarrow$
$A[a,k] + A[b,j] \leq A[a,j] + A[b,k] \rightarrow$
$A[a,j] \leq A[a,k], since\ its\ the\ minimun$
$A[b,k] \leq A[b,j], since\ its\ the\ minimun$

$A[a,k] + A[b,j] \geq A[a,j] + A[b,k], which\ implies\ that$
$A[a,k] + A[b,j] = A[a,j] + A[b,k]. which\ impliest\ that$

$case: 1$
$if\ A[a,k] = A[a,j]\ we\ would\ have\ A[a,k]\ as\ the\ minimun, not\ j$
$The\ same\ case\ if A[b,k] = A[b,j]$
$case\ 2:$
$Now\ if\ A[a,k] > A[a,j] \rightarrow A[b,j] < A[b,k]\ Then\ k\ would\ not\ be\ the\ minimum\ for\ row\ b$
$case\ 3:$
$A[b,k] < A[b,j] \rightarrow A[a,k] < A[a,j]. So\ j\ would\ not\ be\ minimal\ for\ row\ a$

$\boldsymbol{d) Here\ is\ a\ description\ of\ a\ divide\ and\ conquer\ algorithm\ that\ computes\ the\ left\ most}$
$\boldsymbol{minimum\ element\ in\ each\ row\ of\ an\ Mxn\ Monge\ Array\ A:}$
$\boldsymbol{Construct\ a\ submatrix\ A' of\ A\ conssiting\ of\ the\ even - numbered\ rows\ of\ A.}$
$\boldsymbol{Recursively\ determine\ the\ leftmost\ minimum\ for\ each\ row\ of\ A'. Then\ compute\ the}$
$\boldsymbol{leftmost\ minimum\ in\ the\ odd - numbered\ rows\ of\ A}$
$\boldsymbol{Explain\ how\ to\ compute\ the\ leftmost\ minimum\ in\ the\ odd\ numbered\ rows\ of\ A}$
$\boldsymbol{(that\ the\ leftmost\ minimum\ of\ the\ even\ numbered\ rows\ is\ known) in\ \Theta(n+m)}$

$By\ the\ previous\ exercise\ we\ know\ that\ f(2i) \leq f(2i+1) \leq f\big(2(i+1)\big) \rightarrow$
$So\ we\ just\ need\ to\ iterate\ over\ f(2i) and\ f\big(2(i+1)\big). which\ leads\ us\ to$
$f\big(2(i+1)\big) - f(2i) + 1\ tries\ to\ find\ f(2i+1)\ We\ need\ to\ sum\ it\ for\ all$

$odd\ numbered\ rows\ So\ \sum_{k=0}^{\frac{m}{2}-1} f\big(2(i+1)\big) - f(2i) + 1 \rightarrow \frac{m}{2} + \sum_{k=0}^{\frac{m}{2}-1} f\big(2(i+1)\big) - f(2i) \rightarrow$

$$\sum_{k=0}^{\frac{m}{2}-1} f\big(2(i+1)\big) - f(2i) = \sum_{k=1}^{\frac{m}{2}} f(2i) - \sum_{k=0}^{\frac{m}{2}-1} f(2i) = f(m) - f(0) + \sum_{k=1}^{\frac{m}{2}-1} f(2i) - f(2i) + \frac{m}{2}$$

$So\ we\ have\ That\ We\ need\ to\ iterate\ over\ f(m) - f(0) + \frac{m}{2}. The\ maximun\ value$

$f(m) - f(0) can\ have\ is\ n \rightarrow O(n+m)$

$\boldsymbol{e)\ Write\ the\ recurrence\ for\ the\ running\ time\ of\ the\ algorithm\ in\ part\ d.}$
$\boldsymbol{Show\ that\ its\ solutionis\ O\big(m + nlg(n)\big).}$

$So\ the\ idea\ behind\ the\ algorithm\ is\ as\ follows:$
$I\ have\ a\ number\ the\ rows\ 1,2,3,4,5 \dots m\ We\ will\ compute\ the\ left\ most\ minimun\ element$
$By\ calling\ a\ recursive\ call\ on\ the\ submatrix\ 1.2.4,6.8..m(if\ m\ is\ even)\ and\ the$ work level $step$
$Is\ the\ calculation\ we\ have\ done\ in\ the\ letter\ d. So\ the\ calculation\ proceeds\ as:$

$T(m,n) = T\left(\frac{m}{2}, n\right) + \Theta(n+m). This\ is\ independent\ of\ n, since\ n\ won't\ change:$

$T(m) = T\left(\frac{m}{2}\right) + \Theta(n+m) \rightarrow It\ looks\ that\ its\ O\big(m + nlog(m)\big) \rightarrow$

$Suppose\ that\ T(k) \le c_1 k + c_1 nlog(k) for\ n_0 \le k < n \rightarrow$

$T(m) = T\left(\frac{m}{2}\right) + \Theta(n+m) \le c_1 nlog\left(\frac{m}{2}\right) + c_1\frac{m}{2} + c_2 m + c_2 n =$

$c_1 nlog(m) - c_1 n + \frac{c_1 m}{2} + c_2 m + c_2 n. Take\ c_1 = 2c_2$

$\rightarrow c_1 nlog(m) - 2c_2 n + c_2 n + c_2 m + c_2 m \le c_1 nlog(m) + c_2 m + c_2 m$
$= c_1 nlog(m) + c_1(m)$

$This\ guarantees\ only\ for\ n \ge 2n_0. we\ need\ to\ verify\ for\ n_0 \le n < 2n_0$
$We\ have\ done\ it\ a\ plenty\ of\ times. By\ the\ well\ ordering\ principle\ might\ exist\ a\ max$
$We\ can\ analyze\ the\ the\ values\ of\ T(m,n)\ and\ c_1 m$
$+ c_1 nlog(m) for\ the\ values\ commented\ above$