Eg Illustration of the operations of the merge-sort (Refer book)

Analyzing divide-and-conquer algorithms

- when an algorithm calls itself (recursive calls), then its running time can be described a recurrence equation.
- This recurrence relation describes the overall running time on a problem of size n in terms of the running time on smaller inputs.

Further, by solving the recurrence, we can provide bounds on the performance of the algorithm.

- A recurrence for the running time of a D&C algorithm is based on the following 3 steps (of the basic paradigm)

Let $T(n)$ = the running time on a problem of size n

Step (I) if the problem size is small enough, say $n \leq c$ for some constant c, the straightforward solution takes constant time $\Theta(1)$

Step (II) we divide the problems into a subproblems, each of which is $1/b$ the size of original.

let $D(n)$: time to divide the problem into subproblems

Step (III) let $C(n)$ = time to combine the solutions to the subproblems into the solution to the original problem.

Then,

$$T(n) = \begin{cases} \Theta(1) ; & \text{if } n \le c \\ a\,T(n/b) + D(n) + C(n) ; & \text{otherwise} \end{cases}$$

- For merge sort, both $a = b = 2$ and merge sort takes constant time when we have just one element i.e. $n = 1$, when $n > 1$ we calculate the running time as follows:

Divide : The divide step just computes the middle of the subarray, which takes constant time. Thus $D(n) = \Theta(1)$

Conquer : we recursively solve two subproblems, each of size $n/2$ which contributes $2T(n/2)$ to the running time.

Combine : we know that the Merge procedure on an n-element subarray takes $\Theta(n)$, so $C(n) = \Theta(n)$.

So
$$T(n) = \begin{cases} \Theta(1) & \text{if } n=1 \\ 2T(n/2) + \Theta(1) + \Theta(n) & \text{if } n>1 \end{cases}$$

$$= \begin{cases} \Theta(1) & \text{if } n=1 \\ 2T(n/2) + \Theta(n) & \text{if } n>1 \end{cases}$$

using Master's Theorem

$$T(n) = \Theta(n \lg n)$$

Without Master's Theorem

let c = time required to solve the problem of size 1.

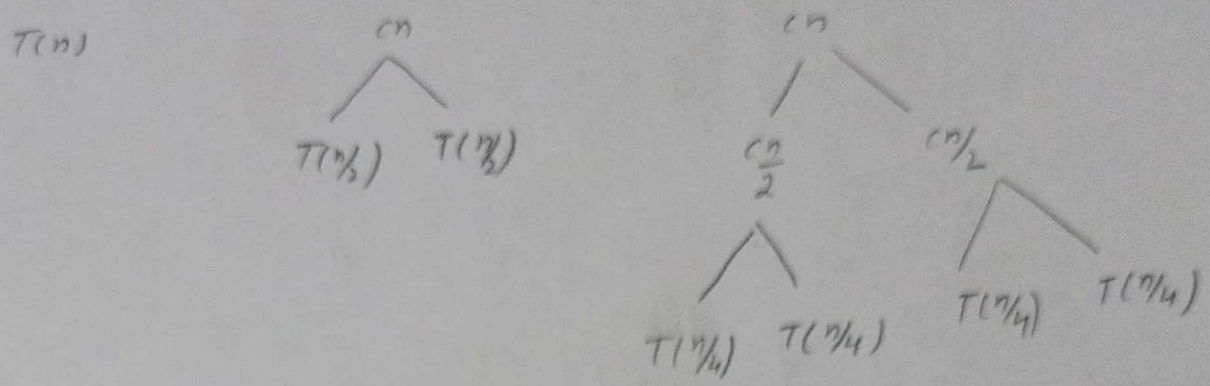
also it is assumed that time c^* is needed for every array element in divide & combine steps.

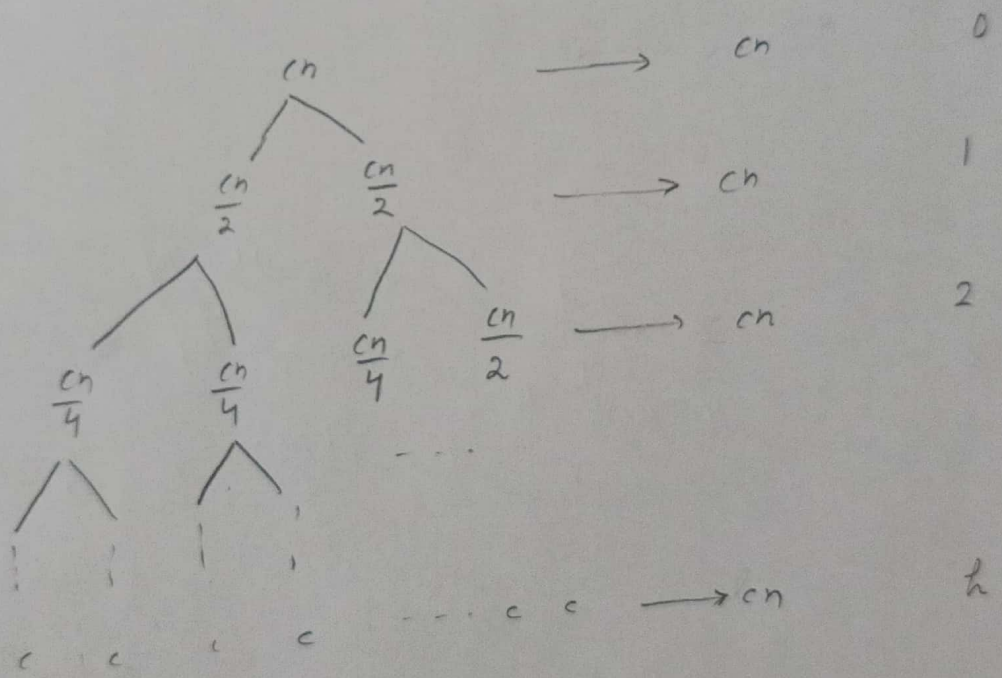
$$T(n) = \begin{cases} c & \text{if } n=1 \\ 2T(n/2) + cn & \text{if } n>1 \end{cases}$$

$$\begin{cases} c_1 \\ 2T(n/2) + c_2 + an+b \end{cases}$$

so we can assume it to be either upper or lower bound.

Further, for simplification, we assume that original problem size is a power of 2, so that each divide step yields two subsequences of size $n/2$.





level i has 2^i nodes and each node contributing a cost of $c\left(\frac{n}{2^i}\right)$.

So at which cost will be c?

$\Rightarrow \quad c\left(\frac{n}{2^i}\right) = c$

$\Rightarrow \quad i = \log_2 n$

So height $= \lg n$

So there will be: $(\lg n + 1)$ levels.

So total cost $= cn(\lg n + 1)$

$= cn\lg n + cn$

$= \Theta(n \lg n)$