DIVIDE-ET-IMPERA / DIVIDE-AND-CONQUER

A TECHNIQUE FOR SPEEDING UP ALGORITHM

SUPPOSE THAT MERGE SORT ON ARRAYS OF LENGTH n TAKES TIME $\leq T(n)$. WHAT IS $T(n)$?

```
DEF MERGE SORT (V):            // SORTS THE ARRAY V
                               n = LEN(V)
O(n)    IF LEN(V) <= 1 :
            RETURN V
        L = V[:LEN(V)/2]
        R = V[LEN(V)/2:]
T(n/2)  L = MERGE SORT (L)
T(n/2)  R = MERGE SORT (R)
        // MERGE L AND R
O(n)    i = 0
        j = 0
        WHILE i+j < LEN(V):     (LEN(V) = n)
            IF i == LEN(L):  // I HAVE ALREADY FINISHED SCANNING L
                V[i+j] = R[j]
                j += 1
            ELIF j == LEN(R):
                V[i+j] = L[i]
                i += 1
            ELIF L[i] <= R[j]:
                V[i+j] = L[i]
                i += 1
            ELSE:  // L[i] > R[j]
                V[i+j] = R[j]
                j += 1
        RETURN V
```

$V=[2,1,5,10,3,7]$

$L=[2,1,5]$ $\quad$ $R=[10,3,7]$

$L=[1,2,5]$ $\quad$ $R=[3,7,10]$

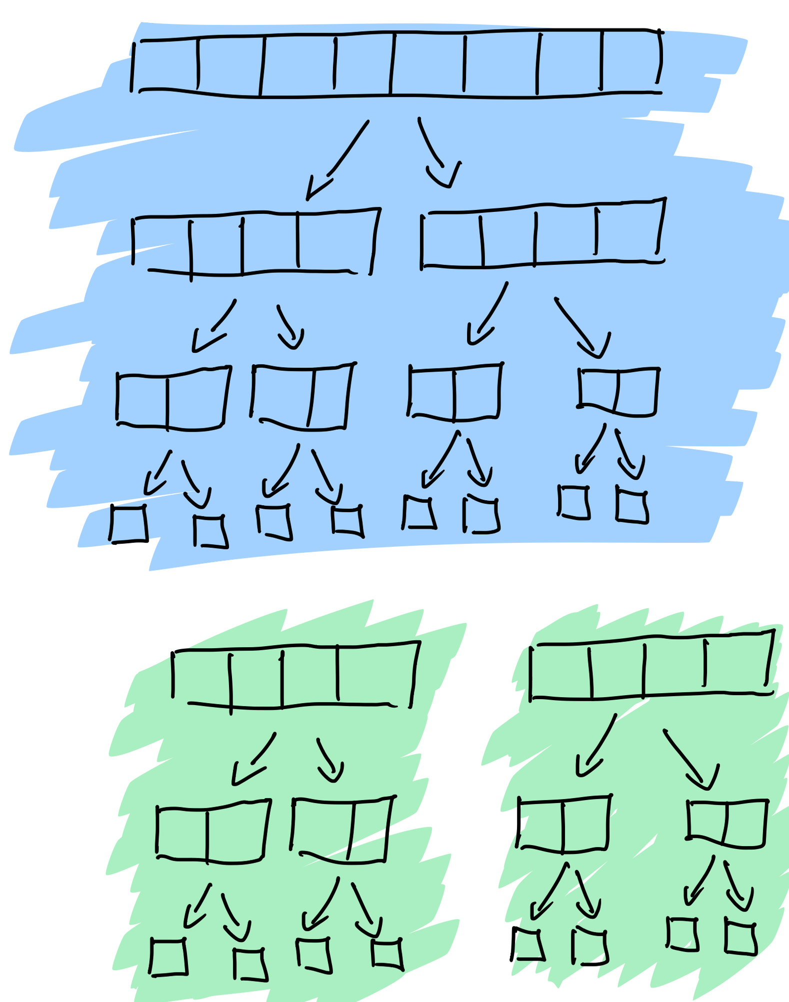
$1,2,3,5,7,10$

THM: MERGE SORT (V) RETURNS V SORTED.

(BY INDUCTION: IF $LEN(V) \leq 1$, THEN V IS SORTED. IF MERGESORT WORKS UP UNTIL LENGTH i, THEN IF I RUN IT ON AN ARRAY V OF LENGTH $i+1$, L AND R WILL HAVE LENGTH $\leq i$. THUS, THE TWO RECURSIVE CALLS WILL SORT L AND R. THE MERGING STEP, THEN, PRODUCES THE SORTED V.)

AS FOR RUN TIME, MERGESORT ON AN ARRAY OF LENGTH n TAKES $\leq 2T(\frac{n}{2}) + O(n)$

THEN, $T(n) \leq 2T(\frac{n}{2}) + c \cdot n$, WHERE c IS SOME POSITIVE CONSTANT.



$T(8) \leq 2T(4) + c \cdot 8$

$T(4) \leq 2T(2) + c \cdot 4$

LET $T(n)$ BE THE WORST-CASE RUNNING TIME OF MERGESORT ON INSTANCES OF SIZE n.

WE ASSUME FOR SIMPLICITY THAT $n = 2^t$, FOR A NON-NEGATIVE INTEGER t. (IF n DOES NOT HAVE THIS PROPERTY WE CAN PAD THE ARRAY WITH AT MOST n MANY "$+\infty$").

THEN, WE HAVE PROVED THAT:

$$\begin{cases} T(n) \leq 2\,T(\frac{n}{2}) + c \cdot n & \forall n > 2 \\ T(0), T(1), T(2) \leq c \end{cases}$$

THIS IS A RECURRENCE:

$$T(n) \leq \begin{cases} 2\,T(\frac{n}{2}) + c\,n & \text{IF } n \geq 3 \\ c & \text{IF } n \leq 2 \end{cases} \qquad \boxed{R_2}$$

WE WANT TO TRANSFORM THIS UPPER BOUND ON THE RUNTIME INTO SOMETHING LIKE WHAT WE USED TO HAVE $(O(n), O(n^2), O(n \log n), \ldots)$

APPROACH

UNROLL THE RECURRENCE FOR SOME NUMBER OF LEVELS, AND SEARCH FOR A PATTERN THAT SOLVES THE RECURRENCE.

OR

GUESS THE SOLUTION AND CHECK THAT IT WORKS.

LET US GUESS THAT $T(n) \leq a \cdot n \log_2 n$, FOR SOME CONSTANT $a > 0$.

$$T(n) \leq \begin{cases} 2\,T(\frac{n}{2}) + c\,n & \text{IF } n \geq 3 \\ c & \text{IF } n \leq 2 \end{cases}$$

IF $n \leq 2$ THEN THE INEQUALITY HOLDS IF $2a \geq c$

(IF $n=2$, THEN $a\,n \log_2 n = 2$, THUS $2a \geq c$ IS SUFFICIENT).

SUPPOSE, NOW, THAT $n \geq 3$. BY INDUCTION, WE KNOW THAT $T(m) \leq a\,m \log_2 m \quad \forall m \leq n-1$.

WE WANT TO PROVE THE SAME INEQUALITY FOR n.

$$T(n) \overset{\text{RECURRENCE}}{\leq} 2\,T\left(\frac{n}{2}\right) + c\,n$$

$$\overset{\text{INDUCTIVE HYPOTHESIS}}{\leq} \cancel{2}\left(a \frac{n}{\cancel{2}} \log_2 \frac{n}{2}\right) + c\,n$$

$$= a\,n \log_2 \frac{n}{2} + c\,n$$

$$= a\,n\,(\log_2(n) - 1) + c\,n$$

$$= a\,n \log_2 n - a\,n + c\,n$$

$$= a\,n \log_2 n + (c-a)\,n$$

$$\leq a\,n \log_2 n \quad \square$$

$\log \frac{a}{b} = \log a - \log b$

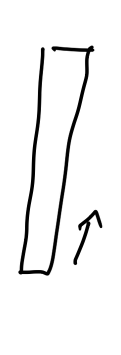
THUS,

THM: MERGE SORT TAKES $O(n \log n)$.

EX: SUPPOSE YOU LIVE IN A BUILDING WITH n FLOORS. YOU HAVE K "BOXES" THAT YOU CAN THROW OUT OF THE WINDOW. THE GENERIC "BOX" WILL BREAK IF THROWN OUT OF A WINDOW AT FLOOR $i \geq i^*$.

HOW CAN YOU FIND i^*?

WITH 1 BOX YOU CAN DO IT WITH n "EXPERIMENTS".



$\frac{n}{2}$

WITH 2 BOXES YOU CAN DO IT WITH $\leq O(\sqrt{n})$ EXPERIMENTS.

HOW?