EX1: WE HAVE n JOBS, THE i TH OF WHICH TAKES w_i SECONDS TO FINISH, AND HAS A VALUE OF v_i.

WE HAVE A CPU AVAILABLE FOR W SECONDS.

A SET OF JOBS S IS FEASIBLE

$$\sum_{i \in S} w_i \le W.$$

WHAT IS A SET OF JOBS S THAT IS FEASIBLE, AND THAT HAS MAXIMUM TOTAL VALUE

$$\sum_{i \in S} v_i \; ?$$

EX 2: CONSIDER THE FOLLOWING TWO CASES:

1. W IS "SMALL",
2. $\sum_i v_i$ IS "SMALL".

CAN YOU GIVE A FASTER ALGORITHM FOR 1, AND A SECOND FASTER " FOR 2?

DIVIDE-ET-IMPERA / DIVIDE-AND-CONQUER

A TECHNIQUE FOR SPEEDING UP ALGORITHM

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

$m = LEN(V)$

```
DEF MERGE SORT (V) :          // SORTS THE ARRAY V
    IF LEN(V) <= 1 :
        RETURN V
    L = V[:LEN(V)/2]
    R = V[LEN(V)/2:]
    L = MERGE SORT (L)
    R = MERGE SORT (R)
    // MERGE L AND R
    i = 0
    j = 0
    WHILE i+j < LEN(V):
        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
```

$O(n)$ $O(n)$ $T(\frac{n}{2})$ $T(\frac{n}{2})$ m

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

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

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

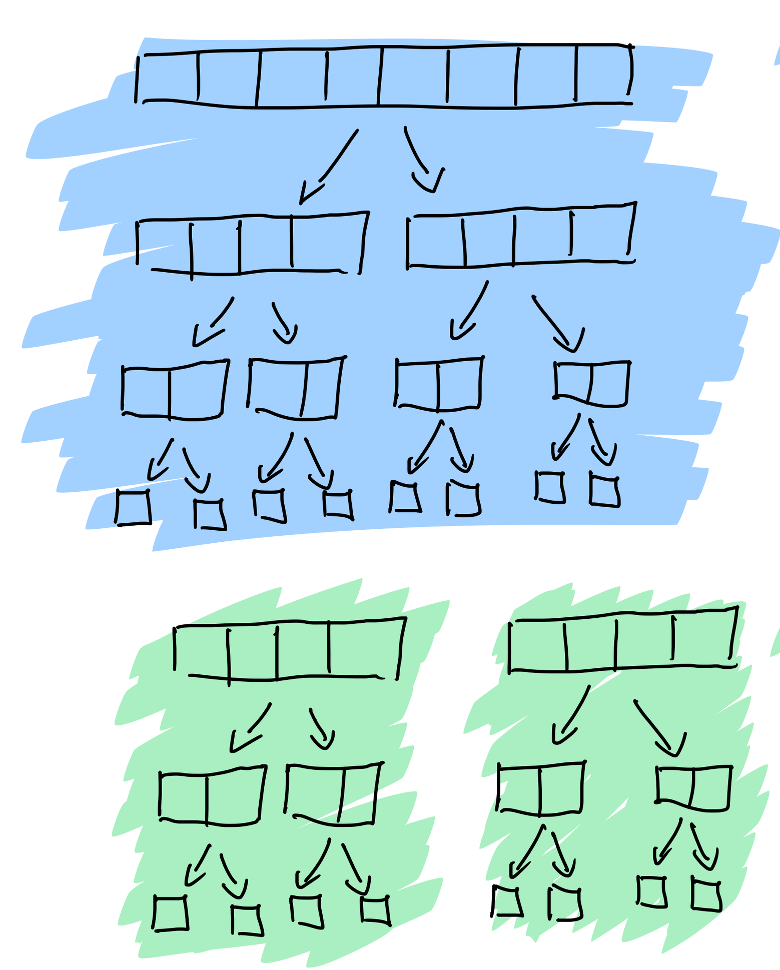
$1, 2, 3, 5, 7, 10$

THM: MERGE SORT (V) RETURNS V SORTED.

(BY INDUCTION: IF $LEN(V) \le 1$, 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 $\le i$. 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 $\le 2\,T(\frac{n}{2}) + O(n)$

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



$T(8) \le 2\,T(4) + c \cdot 8$

$T(4) \le 2\,T(2) + c \cdot 4$

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:

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

THIS IS A RECURRENCE:

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

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

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 $T(n) \le a \cdot n \lg_2 n$, FOR SOME CONSTANT $a > 0$.

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

IF $n \le 2$ THEN THE INEQUALITY HOLDS IF $2a \ge c$

(IF $n = 2$, THEN $a n \lg_2 n = 2$, THUS $2a \ge c$ IS SUFFICIENT).

SUPPOSE, NOW, THAT $n \ge 3$. BY INDUCTION, WE KNOW THAT $T(m) \le a\, m \lg_2 m \quad \forall m \le n-1$.

WE WANT TO PROVE THE SAME INEQUALITY FOR n.

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

$$\overset{\text{INDUCTIVE HYPOTHESIS}}{\le} \cancel{2}\left(a \frac{n}{\cancel{2}} \lg_2 \frac{n}{2}\right) + cn$$

$$= a\, n \lg_2 \frac{n}{2} + cn$$

$$= a\, n \left(\lg_2(n) - 1\right) + cn$$

$$= a\, n \lg_2 n - an + cn$$

$$= a\, n \lg_2 n + (c - a)\, n$$

$$\le a\, n \lg_2 n \quad \square$$

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

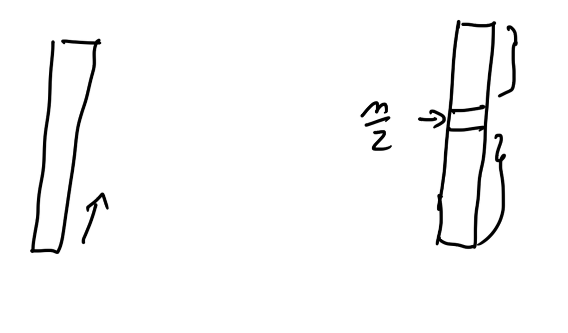
THUS,

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

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 $i \ge i^*$.

HOW CAN YOU FIND i^*?

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



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

HOW?