LEC 37 12/01/99

SORTING

S IS A FINITE LINEARLY ORDERED SET
L IS AN INITIAL LIST OF ALL ELEMENTS OF S IN AN ARBITRARY ORDER. A SORTING IS A REORDERING L INTO L' WHERE ALL THE ELEMENTS ARE IN INCREASING ORDER

$$2,4,1,5,3 \longmapsto 1,2,3,4,5$$

THE KEY QUESTION IS THE COMPLEXITY OF SORTING

BINARY COMPARISON = THE COMPARISON OF TWO ELEMENTS AT A TIME

- TWO POSSIBLE OUTCOMES OF EACH COMPARISON "$a<b$" OR "$b<a$"

2-BRANCHING AT EACH NONTERMINAL VERTEX OF A DECISION TREE   DECISION TREE IS BINARY

\# OF LEAVES = \# OF ALL POSSIBLE INPUTS =
= \# OF PERMUTATIONS OF n ELEMENTS = $n!$

$n!$ LEAVES

height $\geq \lceil \log n! \rceil$
WORST-CASE COMPLEXITY = min heights OF DECISION TREES

INDEED,
\# OF LEAVES $\leq 2^h$
$n! \leq 2^h$
$\log n! \leq h$

Th.1 ANY SORTING ALGORITHM BASED ON BINARY COMPARISONS REQUIRES AT LEAST $\lceil \log n! \rceil$ COMPARISONS

<u>COR.</u> NO SORTING ALGORITHM THAT USES COMPARISONS AS THE METHOD OF SORTING CAN HAVE WORST-CASE COMPLEXITY THAT IS BETTER THAN $O(n \log n)$

<u>PROOF</u>. $n! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot \dots \cdot (n-1) \cdot n$

$$\lceil n/2 \rceil \cdot \dots (n-2) \cdot (n-1) \cdot n \le n! \le \underbrace{n \cdot n \cdot \dots \cdot n}_{n \text{ TIMES}} = n^n$$

$$\lceil n/2 \rceil^{\lceil n/2 \rceil} \le n! \le n^n$$

} TAKE log

$$\frac{n}{2} \cdot \log \frac{n}{2} \le \lceil n/2 \rceil \cdot \log \lceil n/2 \rceil \le \log n! \le n \cdot \log n$$

$\log \frac{n}{2} > \log \sqrt{n}$, SINCE $\frac{n}{2} > \sqrt{n}$ <u>FOR $n > 4$</u>

$$\frac{n}{2} \cdot \log \frac{n}{2} > \frac{n}{2} \cdot \log \sqrt{n} = \frac{n}{2} \cdot \log n^{1/2} = \frac{n \cdot \log n}{4}$$

$$\boxed{\frac{n \cdot \log n}{4} < \log n! < n \cdot \log n}$$ (FOR $n > 4$)

THE BUBBLE SORT

CONSEQUENT PASSES OF THE WHOLE LIST INTERCHANGING A LARGER ELEMENT WITH A SMALLER ONE FOLLOWING IT

4	3	3	3
3	4	1	1
1	1	4	2
2	2	2	4

PASS # ONE, GUARANTEES THE LAST POSITION IS CORRECT!

3	1	1
1	3	2
2	2	3
4	4	4

PASS # TWO GUARANTEES THE LAST TWO POSITIONS (AT THE LEAST)

1	1
2	2
3	3
4	4

PASS # TREE

COMPLEXITY = # OF COMPARISONS =

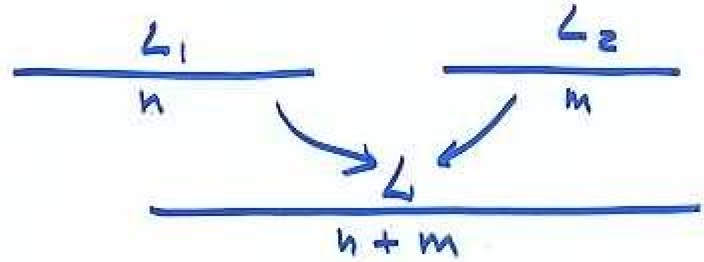
$$= (n-1) + (n-2) + \cdots + 2 + 1 = \frac{(n-1)\cdot n}{2} = O(n^2)$$

DOES NOT REACH $O(n \log n)$ COMPLEXITY.
SIMPLE BUT NOT FAST!

THE MERGE SORT

SORTED LISTS L_1 AND L_2

THEIR SORTED MERGE L



FASTER THAN TO SORT $n+m$ LIST

METHOD: COMPARE THE SMALLEST ELEMENTS OF THE REMAINING LISTS L_1, L_2, ADD THE SMALLEST OF TWO TO THE END OF THE LIST L.

L_1: 2 4 5 6 L_2: 1 3 7 L: 1 2 3 4 5 6 7

COMPARISONS

	2 v 1	1
	2 v 3	2
	4 v 3	3
6	4 v 7	4
	5 v 7	5
	6 v 7	6

IF ONE OF THE LISTS IS EMPTY ADD THE REMAINING ONE TO THE END OF THE MERGE

LEMMA: THE ABOVE ALGORITHM MERGES TWO SORTED LISTS OF m AND n ELEMENTS USING NOT MORE THAN $m+n-1$ COMPARISONS

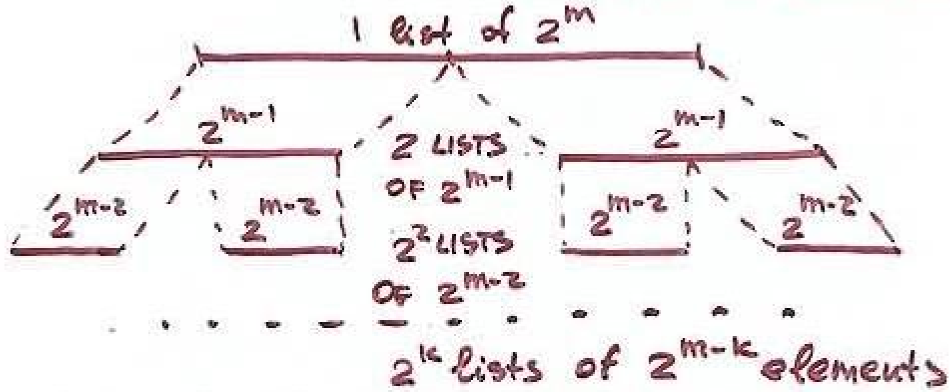
<u>PROOF</u>. EACH COMPARISON REDUCES THE COMBINED LENGTH OF L_1, L_2 BY 1. NO COMPARISONS NEEDED WHEN THE COMBINED LENGTH $= 1$.

THE MERGE SORT ALGORITHM : RECURSIVE DEFINITION

1. SPLIT A LIST INTO TWO SUBLISTS OF (APPROXIMATELY) EQUAL SIZE
2. SORT EACH PART BY THE MERGE SORT ALGORITHM
3. MERGE TWO SORTED PARTS

Th. THE NUMBER OF COMPARISONS NEEDED TO SORT A LIST BY THE MERGE SORT ALGORITHM IS $O(n \cdot \log n)$

PROOF. ASSUME $n = 2^m$ TO SIMPLIFY THE ARGUMENT (WITHOUT LOSS OF GENERALITY)



$2^m/2 = 2^{m-1}$ $\qquad$ 2^m lists of 1 element

COMPARISONS ON THE BOTTOM LEVEL. ON LEVEL K:

$2^k/2$ MERGES OF PAIRS OF 2^{m-k} LISTS $= 2^{k-1}(2^{m-k+1}-1)$ COMPARISONS

$$N = \sum_{k=1}^{m} 2^{k-1}(2^{m-k+1}-1) = \sum_{k=1}^{m}(2^m - 2^{k-1}) = m \cdot 2^m - (2^m - 1) = n \cdot \log n - n + 1 = O(n \cdot \log n)$$