

THE THEORY SIDE OF THIS COURSE WILL COVER

- MATHEMATICAL PRELIMINARIES
 - ASYMPTOTIC GROWTH RATES OF FUNCTIONS
 - RECURRENCES
 - INDUCTION PROOFS
- STANDARD ADTs
 - ELEMENTARY DATA STRUCTURES (STACKS, QUEUES, LISTS)
 - HASHTABLES
 - BINARY SEARCH TREES
 - RED-BLACK TREES
 - DISJOINT SETS
 - GRAPHS
- ALGORITHMS ASSOCIATED WITH THESE ADTs
- TIME COMPLEXITY ANALYSIS OF THESE ALGORITHMS

(2.1) SOME SORTING ALGORITHMS

A TYPICAL PROBLEM ASSOCIATED WITH LISTS IS SORTING. LET OUR LIST BE STORED IN AN ARRAY A WITH INDICES RANGING FROM 1 TO $n = \text{length}[A]$.

WE WRITE $A[i \dots j]$ THE SUBARRAY FROM INDEX i TO INDEX j . IF $i > j$ WE UNDERSTAND THIS TO BE AN EMPTY ARRAY (LENGTH 0.) THE FULL ARRAY IS THEN $A[1 \dots n]$.

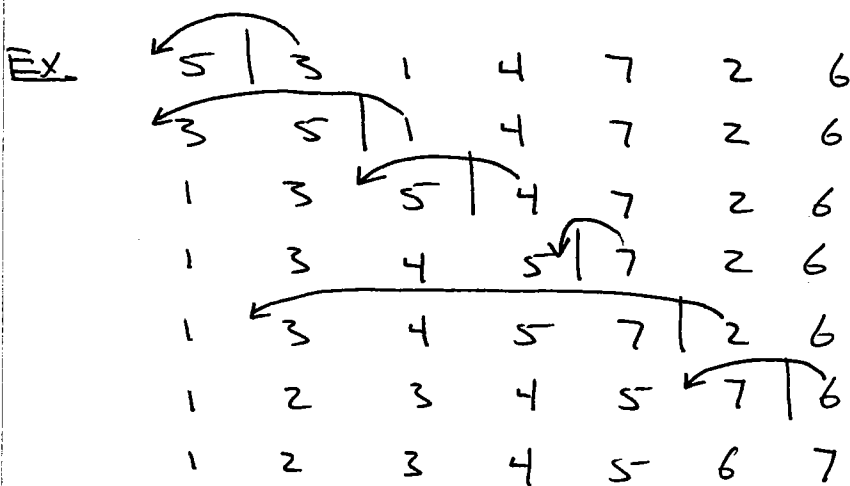
Insertion Sort (A)

- 1.) for $j \leftarrow 2$ to n
- 2.) $temp \leftarrow A[j]$
- 3.) $i \leftarrow (j-1)$
- 4.) while $i > 0 \ \& \ A[i] > temp$
- 5.) $A[i+1] \leftarrow A[i]$
- 6.) $i \leftarrow (i-1)$
- 7.) $A[i+1] \leftarrow temp$

(READ PSEUDO-CODE CONVENTIONS P. 19-20.)

ON THE j TH ITERATION OF LOOP 2-7 THE SUBARRAY $A[1 \dots (j-1)]$ IS SORTED, WHILE $A[j \dots n]$ IS UNSORTED. STEPS 3-7 INSERT $A[j]$ INTO THE SORTED SUBARRAY $A[1 \dots (j-1)]$.

$A_1 \dots A_{j-1} \quad A_j \dots A_n$



2.2 ANALYSIS

WE WISH TO FIND THE RUN TIME OF THIS ALGORITHM AS A FUNCTION OF THE INPUT SIZE n . WE SHOULD MAKE THIS ANALYSIS (AS MUCH AS POSSIBLE) MACHINE INDEPENDENT.

ASSUME STEP k TAKES TIME c_k , AND THAT THE WHILE LOOP TEST (LINE 4) EXECUTES t_j TIMES ON THE j^{TH} EXECUTION OF LOOP 1-7. THE TOTAL RUN TIME IS THEN

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_3(n-1) + c_4 \sum_{j=2}^n t_j + c_5 \sum_{j=2}^n (t_j - 1) \\ &\quad + c_6 \sum_{j=2}^n (t_j - 1) + c_7(n-1) \\ &= (c_4 + c_5 + c_6) \sum_{j=2}^n t_j + (c_1 + c_2 + c_3 - c_5 - c_6 + c_7)n \\ &\quad + (c_5 + c_6 - c_2 - c_3 - c_7) \end{aligned}$$

WE SEE THAT $T(n)$ DEPENDS ON THE NUMBERS t_j , WHICH DEPEND ON THE PARTICULAR INPUT LIST.

IN BEST CASE THE LIST IS ALREADY SORTED, SO $t_j = 1$ ($2 \leq j \leq n$) AND

$$\sum_{j=2}^n t_j = n-1,$$

THEREFORE

$$T(n) = (c_1 + c_2 + c_3 + c_4 + c_7)n - (c_2 + c_3 + c_4 + c_7).$$

A MORE USEFUL ANALYSIS CONCERNS THE WORST CASE, WHICH OCCURS WHEN THE LIST IS SORTED IN REVERSE ORDER. IN THIS CASE $t_j = j$ ($2 \leq j \leq n$), SO

$$\sum_{j=2}^n t_j = \left(\sum_{j=1}^n j \right) - 1 = \frac{n(n+1)}{2} - 1$$

AND HENCE

$$T(n) = \frac{1}{2}(c_4 + c_5 + c_6)n^2 + \left(c_1 + c_2 + c_3 + \frac{c_4}{2} - \frac{c_5}{2} - \frac{c_6}{2} + c_7 \right)n - (c_2 + c_3 + c_4 + c_7).$$

TO ANALYSE THE AVERAGE CASE WE ASSUME ALL INPUTS OF A GIVEN SIZE n ARE EQUALLY LIKELY. THIS SUGGESTS THAT, ON AVERAGE, HALF THE ELEMENTS IN $A[1 \dots (j-1)]$ ARE LESS THAN $A[j]$, AND HALF ARE GREATER.

THUS, ON AVERAGE $t_j = \frac{j}{2}$ ($2 \leq j \leq n$) SO

$$\sum_{j=2}^n t_j = \frac{1}{2} \left(\sum_{j=1}^n j - 1 \right) = \frac{1}{4}n^2 + \frac{1}{4}n - \frac{1}{2}.$$

