THE THEORY SIDE OF THIS COURSE WILL COVER

- MATHEMATICAL PRELIMINARIES
  - ASYMPTOTIC GROWTH RATES OF FUNCTIONS
  - RECURRENCES
  - INDUCTION PROOFS
- STANDARD ADT's
  - ELEMENTARY DATA STRUCTURES (STACKS, QUEUES, LISTS)
  - HASH TABLES
  - BINARY SEARCH TREES
  - RED-BLACK TREES
  - DISJOINT SETS
  - GRAPHS
- ALGORITHMS ASSOCIATED WITH THESE ADT's
- 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 $n = length[A]$.

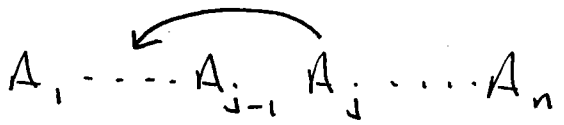
WE WRITE $A[i \cdots j]$ THE SUBARRAY FROM INDEX i TO INDEX j. IF $i > j$ WE UNDERSTAND THIS TO BE AN EMPTY ARRAY (LENGTH 0.) THE FULL ARRAY IS THEN $A[1 \cdots n]$.

InsertionSort(A)

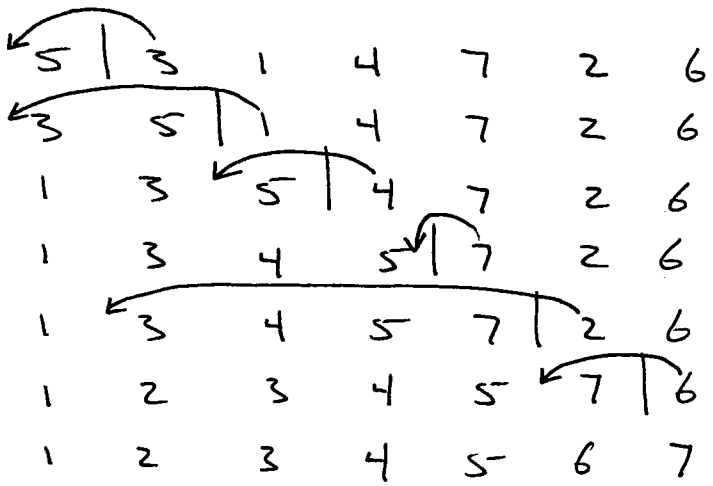
```
1.) for j ← 2 TO n
2.)     temp ← A[j]
3.)     i ← (j-1)
4.)     while i>0 & A[i] > temp
5.)         A[i+1] ← A[i]
6.)         i ← (i-1)
7.)     A[i+1] ← temp
```

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

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



EX.



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

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

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 $t_j = 1$ $(2 \le j \le n)$ 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 $t_j = j$ $(2 \le j \le n)$, SO

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

AND HENCE

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

$$-(c_2+c_3+c_4+c_7) .$$

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 $A[1 \cdots (j-1)]$ ARE LESS THAN $A[j]$, AND HALF ARE GREATER.

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

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

AND SO

$$T(n) = \frac{1}{4}(c_4+c_5+c_6)n^2 + (c_1+c_2+c_3-\frac{3}{4}c_5-\frac{3}{4}c_6+c_7)n$$
$$+ (-c_2-c_3-\frac{c_4}{2}+\frac{c_5}{2}+\frac{c_6}{2}-c_7)$$

	$T(n)$	CLASS
BEST	$an+b$	$\Theta(n)$
WORST	cn^2+dn+e	$\Theta(n^2)$
AVERAGE	fn^2+gn+h	$\Theta(n^2)$

THE CONSTANTS $a-h$ DEPEND ON THE PARTICULAR COMPUTING DEVICE USED. WE SEEK A MEASURE OF RUNNING TIME WHICH IS INDEPENDENT OF CHOICE OF MACHINE

THE REQUIRED MEASURE IS CALLED THE <u>ASYMPTOTIC GROWTH RATE</u> OF $T(n)$. IT IS A MEASURE OF HOW $T(n)$ "SCALES UP" WITH n.

CONSIDER FOUR ALGORITHMS A-D WHOSE RUN TIME ON INPUTS OF SIZE n ARE

A:	n^2	$\Theta(n^2)$
B:	$10n^2$	
C:	$10n^2+2n+100$	
D:	$1000n+10,000$	$\Theta(n)$

D is superior for large n, and worst for small n. A, B, C are classified as equivalent. The lower order terms in C are negligible for large n, and A, B can be equalized by running B on a machine which is 10 times faster.

Returning to Insertion Sort, since constants $a-h$ (and hence c_1-c_7) are not critical, we make no effort to calculate them explicitly. Instead we pick a representative basic operation (sometimes called a <u>barometer</u>) and count the number of times it is executed on inputs of fixed size n.

In sorting algorithms it is customary to count the number of array comparisons performed, i.e. line 4 of Insertion Sort.

<u>Exercise</u>

Show that Insertion Sort does $n-1$, $\frac{n(n-1)}{2}$, $\frac{n(n-1)}{4}$ comparisons in best, worst, and average cases, on input arrays of length n.