CHAPTER 3 : EFFICIENCY OF ALGORITHMS

ATTRIBUTES OF ALGORITHMS

THE ISSUE OF CORRECTNESS IS REALLY A MATHEMATICAL ONE : AN ALGORITHM SHOULD PRODUCE CORRECT RESULTS ON ALL POSSIBLE INPUTS. TO PROVE CORRECTNESS THEN, IT IS NOT SUFFICIENT TO TRACE EXECUTION ON ANY FINITE NUMBER OF INPUTS.

SOMETIMES A RIGOROUS PROOF OF CORRECTNESS MAY BE REQUIRED, BUT NOT IN THIS CLASS. WE WILL BE SATISFIED WITH AN OCCASIONAL TRACE.

WE SHOULD STRIVE TOWARDS READABILITY AND EASE OF UNDERSTANDING IN OUR ALGORITHMS BY CHOOSING GOOD DESCRIPTIVE VARIABLE NAMES AND BY CHOOSING LOGICAL CONSTRUCTS WHICH CLARIFY MEANING.

EX. IF $i > n$ ... } FROM LINE 9 OF SEQUENTIAL SEARCH

BETTER: IF FOUND = FALSE ... } FROM LINE 9 OF SEQUENTIAL SEARCH

STILL BETTER: IF NOT FOUND ... } FROM LINE 9 OF SEQUENTIAL SEARCH

ELEGANCE IS RELATED TO READABILITY, AND IS SOMETIMES IN CONFLICT WITH IT

ESSENTIALLY ELEGANCE MEANS SIMPLICITY AND BREVITY.

EX

GIVEN: $n \geq 1$

FIND: THE SUM $1+2+3+\cdots+(n-1)+n$

1.) SET i TO 1
2.) SET SUM TO 0
3.) REPEAT UNTIL $i > n$
4.) SET SUM TO SUM + i
5.) SET i TO $i+1$
6.) END LOOP
7.) PRINT SUM
8.) STOP

THIS IS A STRAIGHT FORWARD AND OBVIOUS APPROACH, BUT YOU MAY NOT KNOW THAT THERE IS A FORMULA

$$1+2+3+\cdots+(n-1)+n = \frac{n(n+1)}{2}$$

PROOF (C.F. GAUSS)

$$S = 1+2+\cdots+(n-1)+n$$
$$S = n+(n-1)+\cdots+2+1$$
$$2S = (n+1)+(n+1)+\cdots+(n+1)+(n+1) = n(n+1)$$

$$\therefore \quad S = \frac{n(n+1)}{2}$$

THUS WE HAVE THE MORE ELEGANT ALGORITHM:

1.) PRINT $n \cdot (n+1)/2$
2.) STOP

OF COURSE, UNLESS ONE KNOWS THE FORMULA, THIS IS DIFFICULT TO UNDERSTAND.

EFFICIENCY IS THE TERM USED TO DESCRIBE AN ALGORITHM'S CAREFUL (OR NOT SO CAREFUL) USE OF RESOURCES. THE TWO RESOURCES WE ARE CONCERNED WITH ARE SPACE (i.e. MEMORY) AND TIME.

SPACE EFFICIENCY CAN BE JUDGED BY THE AMOUNT OF INFORMATION THE ALGORITHM STORES IN ORDER TO DO ITS JOB, i.e. HOW MANY AND WHAT KIND OF VARIABLES ARE USED. SOME MEMORY IS USED FOR THE INPUT VALUES. IF AN ALGORITHM USES JUST A FEW MORE VARIABLES TO PROCESS THE DATA, IT IS CONSIDERED SPACE EFFICIENT. IF IT USES AS MUCH SPACE OR MORE THAN THE INPUT ITSELF, IT WOULD BE CONSIDERED SPACE INEFFICIENT.

WE WILL BE PRIMARILY CONCERNED WITH TIME EFFICIENCY. ONE WAY TO APPROACH THE PROBLEM OF MEASURING TIME EFFICIENCY IS TO WRITE A PROGRAM TO IMPLEMENT THE ALGORITHM, RUN IT ON SOME COMPUTER WITH SOME INPUTS, AND TIME THE RESULTS.

THERE ARE TWO PROBLEMS WITH THIS APPROACH:

1.) ARE WE MEASURING THE TIME EFFICIENCY OF THE ALGORITHM, OR JUST THE SPEED OF THE PARTICULAR COMPUTER (OR COMPUTER LANGUAGE)? IN FACT WE'RE MEASURING ALL THESE THINGS.

2.) WE CAN EXPECT TO GET DIFFERENT RESULTS FOR DIFFERENT INPUT. HOW DO THESE DIFFERENT RESULTS RELATE TO THE TIME EFFICIENCY OF THE ALGORITHM?

WE NEED A MEASURE OF TIME EFFICIENCY WHICH IS INDEPENDENT OF ANY PARTICULAR COMPUTER OR COMPUTER LANGUAGE, AND WHICH TELLS US SOMETHING ABOUT ALL POSSIBLE INPUTS.

TO DEAL WITH (1) WE MEASURE TIME NOT BY COUNTING SECONDS, BUT BY COUNTING THE NUMBER OF INSTRUCTIONS EXECUTED. OF COURSE NOT ALL INSTRUCTIONS WILL (NECESSARILY) BE COUNTED EQUALLY.

EX.
```
SET  a TO b  ...
IF   a < b   ...
END OF LOOP  ...
```

TO DEAL WITH (2) WE HAVE THREE MEASURES WHICH CONSIDER ALL POSSIBLE (i.e. VALID) INPUTS.

WORST CASE is THE MAXIMUM TIME TAKEN OVER ALL INPUTS OF A GIVEN SIZE

BEST CASE is THE MINIMUM TIME TAKEN OVER ALL INPUTS.

AVERAGE CASE is THE AVERAGE TIME TAKEN, WHERE WE CONSIDER ALL INPUTS TO BE EQUALLY LIKELY.

IN ORDER TO EXPLORE THESE IDEAS, WE CONSIDER SEVERAL DIFFERENT ALGORITHMS WHICH SOLVE THE SAME PROBLEM.

DATA CLEAN-UP PROBLEM

GIVEN A LIST OF $n \geq 1$ INTEGERS $a_1, a_2, \ldots, a_n$, FIND ALL OCCURANCES OF 0 IN THE LIST AND REMOVE THEM, THEN RETURN THE NEW LIST.

INPUT: $n \geq 1$, $a_1, \ldots, a_n$
OUTPUT: A SUBSET OF $a_1, \ldots, a_n$ WHICH CONTAINS NO ZEROS.

EX.											LENGTH
	0	3	1	7	0	2	4	11	0	5	10
	3	1	7	2	4	11	5				7

THE SHUFFLE LEFT ALGORITHM

GO THROUGH THE LIST FROM LEFT TO RIGHT; EVERY TIME WE FIND A ZERO, SQUEEZE IT OUT OF THE LIST BY COPYING EACH REMAINING DATA ITEM ONE CELL TO THE LEFT.

		LENGTH
EX	0 3 1 7 0 2 4 11 0 5	10
	3 1 7 0 2 4 11 0 5 | 5	9

INTERMEDIATE STEPS:

0 3 1 7 0 2 4 11 0 5
3 5 1 7 0 2 4 11 0 5
3 1 1 7 0 2 4 11 0 5
3 1 7 7 0 2 4 11 0 5
3 1 7 0 0 2 4 11 0 5
3 1 7 0 2 2 4 11 0 5
3 1 7 0 2 4 4 11 0 5
3 1 7 0 2 4 11 11 0 5
3 1 7 0 2 4 11 0 0 5
3 1 7 0 2 4 11 0 5 | 5

SUBSEQUENT STEPS:

	LENGTH
3 1 7 0 2 4 11 0 5 | 5	9
3 1 7 0 2 4 11 0 5 | 5	
3 1 7 0 2 4 11 0 5 | 5	
3 1 7 0 2 4 11 0 5 | 5	
3 1 7 2 4 11 0 5 | 5 5	8
3 1 7 2 4 11 0 5 | 5 5	
3 1 7 2 4 11 0 5 | 5 5	
3 1 7 2 4 11 0 5 | 5 5	
3 1 7 2 4 11 5 | 5 5 5	7

INPUT: $n \geq 1$, $a_1, \ldots, a_n$

OUTPUT: MODIFIED LIST WITH NO ZEROS, & length

1.) SET LENGTH TO n
2.) SET i TO 1
3.) WHILE i ≤ LENGTH
4.) IF $a_i \neq 0$
5.) SET i TO i+1
6.) ELSE
7.) SET LENGTH TO (LENGTH - 1)
8.) SET j TO i+1
9.) WHILE j ≤ n
10.) SET a_{j-1} TO a_j
11.) SET j TO j+1
12.) END LOOP
13.) END LOOP
14.) STOP

NOTE THAT THIS ALGORITHM DOES ALOT OF COPYING. MANY ITEMS MAY BE MOVED MULTIPLE TIMES.

THE COPY OVER ALGORITHM

SCAN THE LIST FROM LEFT TO RIGHT AND COPY NON-ZERO VALUES INTO A NEW LIST.

EX. 0 3 1 7 0 2 4 11 0 5