Algorithms: Complexity (Big Oh) of the codes

The RAM Model of Computation

Algorithms are an important and durable part of computer science because they can be studied in a machine/language independent way.

This is because we use the **RAM model of computation** for all our analysis.

- Each "simple" operation (+, -, =, if, call) takes 1 step.
- Loops and subroutine calls are *not* simple operations. They depend upon the size of the data and the contents of a subroutine. "Sort" is not a single step operation.
- Each memory access takes exactly 1 step.

We measure the run time of an algorithm by counting the number of steps.

This model is useful and accurate in the same sense as the flat-earth model (which *is* useful)!

• Assumption

• Each operation will take same time

for(int i=0; i<5; i++) printf("%d", i); c1 = 1 + 6 + 5 = 12 c2 = 5*1 = 5Total c = c1 + c2 = 12 + 5 = 17 $= O(n^0) = O(1)$

• Assumption

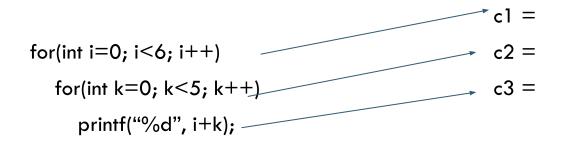
• Each operation will take same time

for(int i=0; ic1 = 1 + (n+1) + n = 2n + 2
printf("%d", i);
$$c2 = 1*n = n$$

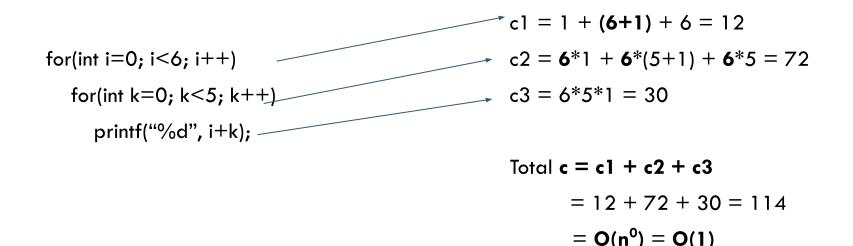
Total c = c1 + c2 = 3n + 2 = O(n)

• Assumption

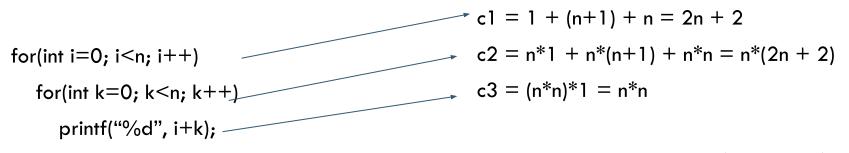
• Each operation will take same time



- Assumption
 - Each operation will take same time



- Assumption
 - Each operation will take same time

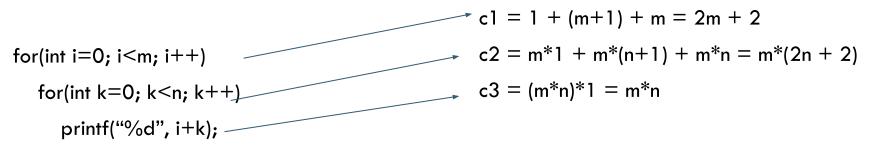


Total $c = c1 + c2 + c3 = (2n+2) + n^*(2n+2) + n^*n$

= 3*n*n + 3*n + 2

= O(n*n)

- Assumption
 - Each operation will take same time

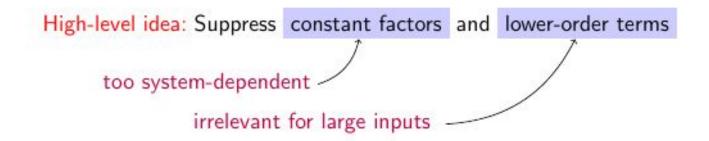


Total $c = c1 + c2 + c3 = (2m+2) + m^*(2n+2) + m^*n$

= 3*m*n + 3*m + 2

= O(m*n)

Asymptotic Analysis



Example: Equate $6n \log_2 n + 6$ with just $n \log n$.

Terminology: Running time is $O(n \log n)$ ["big-Oh" of $n \log n$] where n = input size (e.g. length of input array).

Example: One Loop

Problem: Does array A contain the integer t? Given A (array of length n) and t (an integer).

Algorithm 1

- 1: for i = 1 to n do
- 2: if A[i] == t then
- 3: Return TRUE
- 4: Return FALSE

Example: One Loop

Problem: Does array A contain the integer t? Given A (array of length n) and t (an integer).

Algorithm 1

- 1: for i = 1 to n do
- 2: if A[i] == t then
- 3: Return TRUE
- 4: Return FALSE

Question: What is the running time?

A) O(1) C) O(n)B) $O(\log n)$ D) $O(n^2)$

Example: Two Loops

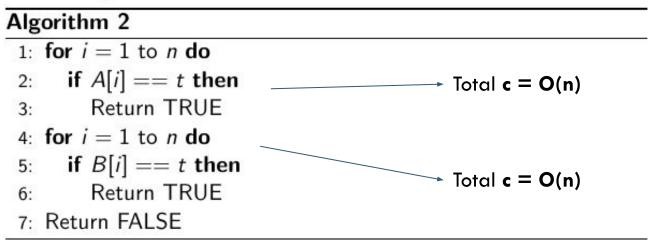
Given A, B (arrays of length n) and t (an integer). [Does A or B contain t?]

Algorithm 2

- 1: for i = 1 to n do
- 2: if A[i] == t then
- 3: Return TRUE
- 4: for i = 1 to n do
- 5: **if** B[i] == t **then**
- 6: Return TRUE
- 7: Return FALSE

Example: Two Loops

Given A, B (arrays of length n) and t (an integer). [Does A or B contain t?]



A)
$$O(1)$$
 C) $O(n) = O(n) + O(n)$
B) $O(\log n)$ D) $O(n^2)$

Example: Two Nested Loops

Problem: Do arrays A, B have a number in common? Given arrays A, B of length n.

Algorithm 3

- 1: for i = 1 to n do
- 2: **for** j = 1 to *n* **do**
- 3: **if** A[i] == B[j] **then**
- 4: Return TRUE
- 5: Return FALSE

Example: Two Nested Loops

Problem: Do arrays A, B have a number in common? Given arrays A, B of length n.

Algorithm 3		
1: for <i>i</i>	= 1 to <i>n</i> do	
2: for	j = 1 to n do	
3: i	f A[i] == B[j] then	
4:	Return TRUE	
5: Retur	n FALSE	

Question: What is the running time?A) O(1)C) O(n)B) $O(\log n)$ D) $O(n^2)$

Example: Two Nested Loops (II)

Problem: Does array A have duplicate entries? Given arrays A of length n.

<u> </u>	rithm 4	
1: for $i = 1$ to n do		
2:	for $j = i+1$ to n do	
3:	if $A[i] == A[j]$ then	
4:	Return TRUE	
5: F	Return FALSE	

Example: Two Nested Loops (II)

Problem: Does array A have duplicate entries? Given arrays A of length n.

Algorithm 4

- 1: for i = 1 to *n* do
- 2: for j = i+1 to n do

3: if
$$A[i] == A[j]$$
 then

- 4: Return TRUE
- 5: Return FALSE

Question: What is the running time?

A) O(1) C) O(n)B) $O(\log n)$ D) $O(n^2)$