SCC120 Fundamentals of Computer

Science Introduction to Algorithms

Week 7 Workshop

1) First, do you have any questions about this week's lecture material?

2) We have the following input, problem, and output:

Input: Given an array of integers

Problem: Find the minimum integer in array

Output: Return the index of the minimum integer in array (just find first one if the minimum integer occurs more than once in array)

Answer: See Week 7, Monday's lecture slides.

(a) Write a simple but real example of the input (as in an array with some actual values of integers) so you can see and think with an example.

[12,14, 18, 6, 5]

(b) Write the function/code for "find min index", starting with this:

See slide 53 on Moodle

(c) How would it be different if it were "find max"?

The '<' changes to '>'

(d) Perform operation counting, as we did in lecture.

(e) What is the average cost T(N) of the function?

See slide 53 on Moodle

3) Recall that logarithmic functions are the inverse of exponential functions. So $b^c = a$ is equivalent to $\log_b a = c$ (for $a > 0$ and $b > 0$). Compute the following:

(a) $\log_{10} 10000 = 4$

(b) $\log_2 4 = 2$

(c) $\log_2 5 = 2.321$ (If you can't do it on your calculator, use formula $\log_b x = \log_d x / \log_d b$

(d) $\log_2 32 = 5$

(e) $\log_2 16777216 = 24$

4) Let $a = \log_2 n$. What are the values of n if a = 1, 2, 3, …, 10?

Let $b = \log_{10} n$. What are the values of n if b = 1, 2, 3, …, 10?

Observe the patterns of how n increases in these two cases.

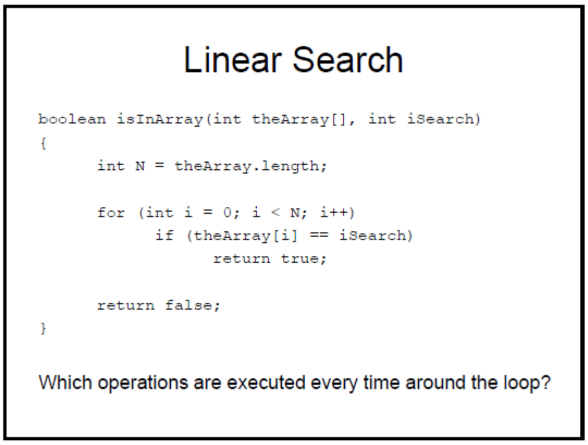
5) Plot these functions in a graph.

(a) T(N) = 3 x N + 4

(b) T(N) = 10

(c) T(N) = 2 x $\log_2 N$ + 6

6) Work with someone else for this question. Try to go over the slides below and understand them together. In particular, work through the details of operation counting (for example, why those numbers of operations).



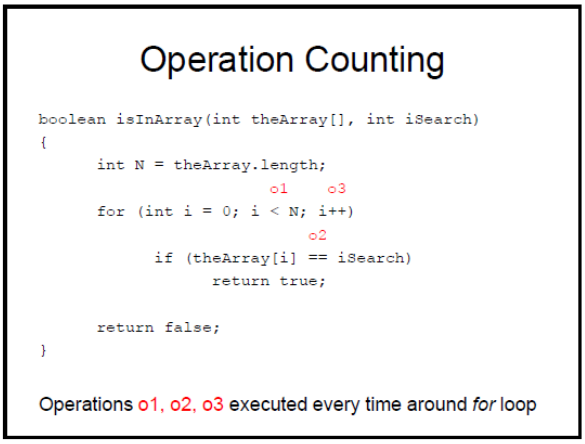
Linear Search

```
boolean isInArray(int theArray[], int iSearch)
{
      int N = theArray.length;

      for (int i = 0; i < N; i++)
            if (theArray[i] == iSearch)
                  return true;

      return false;
}
```

Which operations are executed every time around the loop?

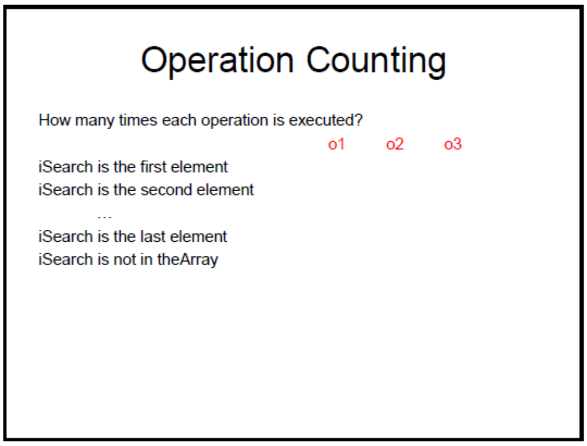


Operation Counting

```
boolean isInArray(int theArray[], int iSearch)
{
      int N = theArray.length;
                       o1    o3
      for (int i = 0; i < N; i++)
                            o2
            if (theArray[i] == iSearch)
                  return true;

      return false;
}
```

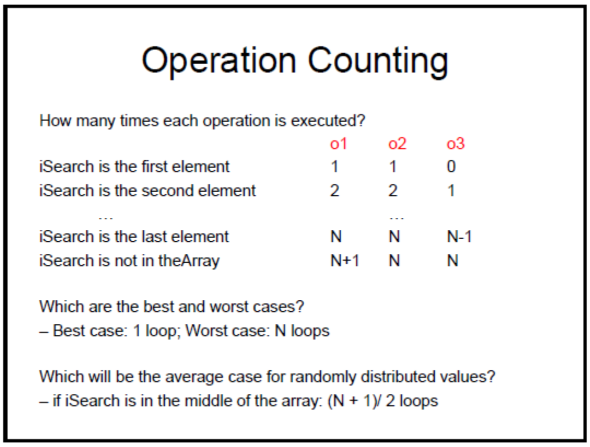
Operations o1, o2, o3 executed every time around *for* loop



Operation Counting

How many times each operation is executed?

	o1	o2	o3
iSearch is the first element			
iSearch is the second element			
…			
iSearch is the last element			
iSearch is not in theArray			



Operation Counting

How many times each operation is executed?

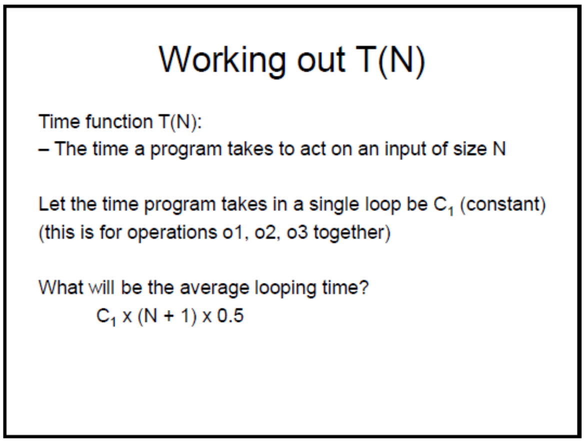
	o1	o2	o3
iSearch is the first element	1	1	0
iSearch is the second element	2	2	1
…		…	
iSearch is the last element	N	N	N-1
iSearch is not in theArray	N+1	N	N

Which are the best and worst cases?

– Best case: 1 loop; Worst case: N loops

Which will be the average case for randomly distributed values?

– if iSearch is in the middle of the array: (N + 1)/ 2 loops



Working out T(N)

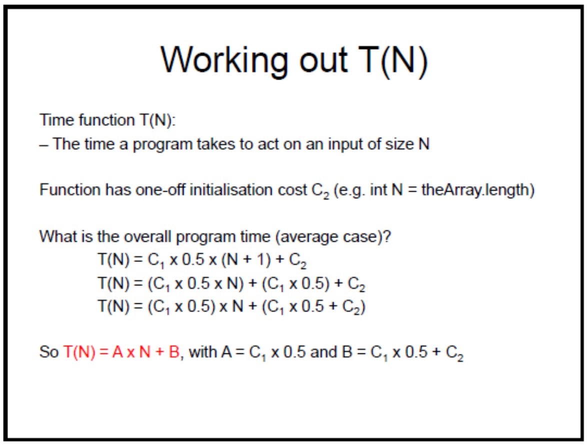
Time function T(N):

– The time a program takes to act on an input of size N

Let the time program takes in a single loop be C_1 (constant) (this is for operations o1, o2, o3 together)

What will be the average looping time?

C_1 x (N + 1) x 0.5



Working out T(N)

Time function T(N):

– The time a program takes to act on an input of size N

Function has one-off initialisation cost C_2 (e.g. int N = theArray.length)

What is the overall program time (average case)?

$T(N) = C_1 \times 0.5 \times (N + 1) + C_2$

$T(N) = (C_1 \times 0.5 \times N) + (C_1 \times 0.5) + C_2$

$T(N) = (C_1 \times 0.5) \times N + (C_1 \times 0.5 + C_2)$

So T(N) = A x N + B, with A = C_1 x 0.5 and B = C_1 x 0.5 + C_2