CS 112 – Introduction to Computing II

Wayne Snyder
Computer Science Department
Boston University

Today: Administrivia and Motivation

- Administrative Matters:
  - Review of course design and course policies
- Motivation: Two Algorithms for Searching An Array
  - Sequential Search and Binary Search compared

Next Time: From Python to Java

Reading assignment will be posted on Piazza!

Computer Science

Motivation: Two Algorithms for Searching an Array

Computer Science

This course is about learning to be computer scientists and programmers. Our two main goals are

Learn object-oriented programming using Java;

Develop the tools to think scientifically about the design and analysis of data structures and algorithms.

The second goal is by far the most important, and we will get a sense for what the **science** of algorithms is today; next time we will start to study Java in detail...

2

Motivation: Two Algorithms for Searching an Array

Computer Science

How do we think scientifically about the programs we write? Mostly this is by analyzing how they use resources (time, space, hardware, power, other algorithms). We will focus in this class on understanding the **running time** of algorithms.

As an example, let's consider an *unsorted* list of integers:

0	1	2	3	4	5	6	7	8	9
78	25	2	15	26	38	7	45	12	19

How would we determine if a given integer, say **19**, is in the list?

This problem is called "sequential search" or "linear search."

Let's consider a Python implementation......

3

Motivation: Two Algorithms for Searching an Array

Computer Science

Sequential Search in Python:

```
>>> def seqSearch(A,n):
...     for i in range(len(A)):
...             if A[i] == n:
...                     print 'Found: ', n;
...                     return;
...     print 'Not found: ', n;
...
>>> A = [78, 25, 2, 15, 26, 38, 7, 45,  12, 19];
>>> seqSearch(A,19);
Found 19
>>> seqSearch(A,29);
Not found: 29
```

4

Motivation: Two Algorithms for Searching an Array

Computer Science

How would we analyze this algorithm?

We are basically interested in how long it takes to find an arbitrary member of the list......

0	1	2	3	4	5	6	7	8	9
78	25	2	15	26	38	7	45	12	19

Here are the kinds of questions we want to answer:

How many "basic operations" (e.g., comparing one integer to another) does it take to find the integer (or not), expressed as a function of N = number of data items.

In the worst case?
In the best case?
In the average case?

5

Motivation: Two Algorithms for Searching an Array

Computer Science

How would we analyze this algorithm mathematically?

We are basically interested in how long it takes to find an arbitrary member of the list......

0	1	2	3	4	5	6	7	8	9
78	25	2	15	26	38	7	45	12	19

N = number of data items = 10

Here are the kinds of questions we want to answer:

How many "basic operations" (e.g., comparing one integer to another) does it take to find the integer (or not), expressed as a function of N = number of data items.

1+2+...+N = N(N+1)/2

In the worst case?	N	10
In the best case?	1	1
In the average case?	(N+1)/2	5.5

6

Motivation: Two Algorithms for Searching an Array

Computer Science

Digression on summing the series 1 + 2 + ... + N:

I really like the proof of

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

in which $1 + 2 + \cdots + (n-1) + n$ is written forwards then backwards and summed. It is claimed that Gauss had come up with this when he was just a child, although contested.

The proof

Let

$$s = 1 + 2 + \cdots + (n-1) + n.$$

Clearly,

$$s = n + (n-1) + \cdots + 2 + 1.$$

Sum to get

$$2s = \underbrace{(n+1) + (n+1) + \cdots + (n+1) + (n+1)}_{n \text{ times}}.$$

Hence,

$$2s = n(n+1),$$

and

$$s = \frac{n(n+1)}{2}.$$

7

Motivation: Two Algorithms for Searching an Array

Computer Science

Now let's consider how things change when we sort the list into ascending order:

0	1	2	3	4	5	6	7	8	9
2	**7**	**12**	**15**	**19**	**25**	**26**	**38**	**45**	**78**

Now how would we determine if a given integer, say **15**, is in the array?

The best way to do this is called "binary search."

Again let's consider the Python implementation......

8

Motivation: Two Algorithms for Searching an Array

Computer Science

```
>>> def binSearch (A,n):
...     return binSearchAux(A,n,0,len(A)-1)
...
>>> def binSearchAux(A,n,left,right):
...     if right < left:
...         return False;
...     else:
...         mid = (left+right) // 2;
...         if n == A[mid]:
...             return True;
...         elif n < A[mid]:
...             return binSearchAux(A,n,left,mid-1);
...         else:
...             return binSearchAux(A,n,mid+1,right);
...

>>> A = [2, 7, 12, 15, 19, 25, 26, 38, 45, 78 ];


>>> binarySearch(A,15);
True


>>> binarySearch(A,29);
False
```

9

Motivation: Two Algorithms for Searching an Array

Computer Science

How would we analyze this algorithm mathematically?

Again, we want to count the number of "basic operations" such as comparisons:

0	1	2	3	4	5	6	7	8	9
2	**7**	**12**	**15**	**19**	**25**	**26**	**38**	**45**	**78**

In the worst case?
In the best case?
In the average case?

10

Motivation: Two Algorithms for Searching an Array

Computer Science

RECALL:

$\log_A B = C$

iff

$A^C = B$

floor(M) = largest integer ≤ M

How would we analyze this algorithm mathematically?

Again, we want to count the number of "basic operations" such as comparisons:

0	1	2	3	4	5	6	7	8	9
2	**7**	**12**	**15**	**19**	**25**	**26**	**38**	**45**	**78**

In the worst case?	$\text{floor}(\log_2(N)) + 1$	floor(3.32) + 1 = 4
In the best case?	1	1
In the average case?	approx. $0.87 * \log_2(N)$	2.89

11

Motivation: Two Algorithms for Searching an Array

Computer Science

RECALL:

$\log_2 B = C$

iff

$2^C = B$

log is the functional inverse of the exponential function:

$$B \underset{2^x}{\overset{\log_2(x)}{\rightleftarrows}} C$$

$\log_2(2^x) = x$

$2^{\log(x)} = x$

How to derive the log(N) bound on the worst case:

Let us count the **approximate size of the sublist** to be searched in the worst case at each call of the function:

Original List:	N/1	=	$N/2^0$
After 1 comparison:	N/2	=	$N/2^1$
After 2 comparisons:	N/4	=	$N/2^2$
	.....		
After X comparisons:	N/N = 1	=	$N/2^x$

So for what X does $N = 2^x$? Clearly, $X = \log_2(N)$.
The answer to the question "how many times can I divide N by 2 before I get 1" is "approximately $\log_2(N)$."

A more precise analysis gives us $\text{floor}(\log_2(N)) + 1$, but we will be satisfied with the approximate answer of $\log_2(N)$ in this class......

12

Motivation: Two Algorithms for Searching an Array

Computer Science

How would we analyze this algorithm **experimentally**?

Again, we want to count the number of "basic operations" such as comparisons, but in this case we will **run the program with sample data** (randomly generated lists) and actually count the number of comparisons.

Hopefully, our mathematical and experimental results are consistent!!

13

Experiments with Sequential and Binary Search

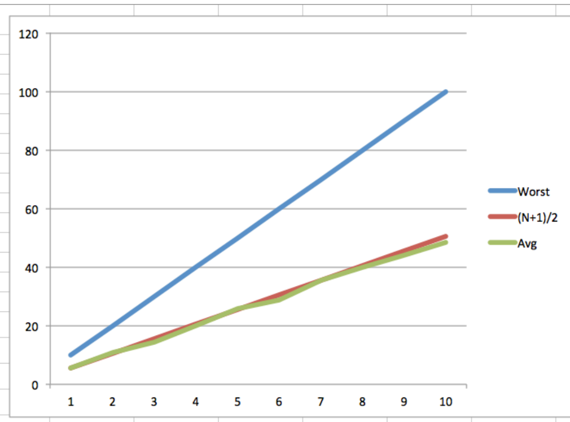
Computer Science

Sequential Search

N	Worst	(N+1)/2	Avg
10	10	5.5	5.49
20	20	10.5	10.92
30	30	15.5	14.33
40	40	20.5	19.92
50	50	25.5	25.78
60	60	30.5	28.76
70	70	35.5	35.61
80	80	40.5	39.99
90	90	45.5	44.11
100	100	50.5	48.37

Worst = N

Avg = ~ (N+1)/2

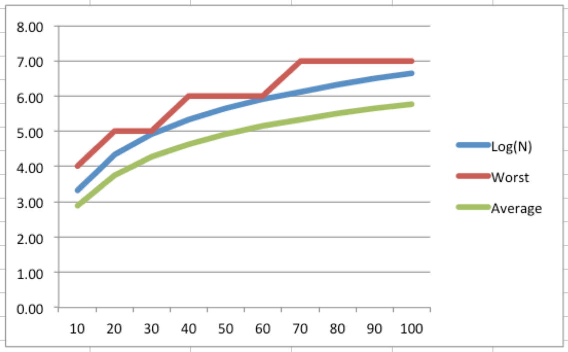


14

Experiments with Sequential and Binary Search

Computer Science

N	log(N)	Worst	Average
10	3.32	4	2.89
20	4.32	5	3.76
30	4.91	5	4.27
40	5.32	6	4.63
50	5.64	6	4.91
60	5.91	6	5.14
70	6.13	7	5.33
80	6.32	7	5.50
90	6.49	7	5.65
100	6.64	7	5.78
110	6.78	7	5.90
120	6.91	7	6.01
130	7.02	8	6.11



Worst = floor($\log_2$(N)) + 1

Avg = ~ 0.87 * $\log_2$(N)

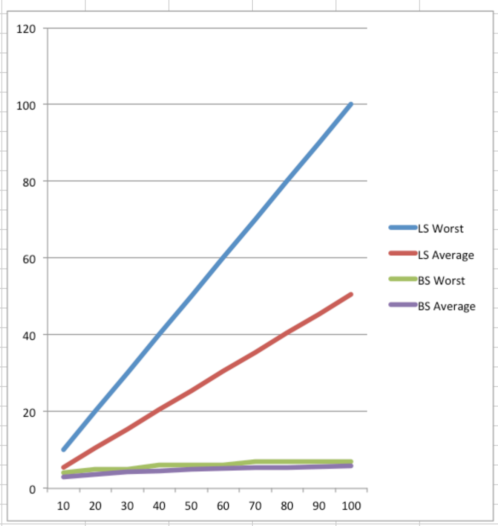
15

Experiments with Sequential and Binary Search

Computer Science

		Linear Search		Binary Search	
N	log(N)	Worst	Average	Worst	Average
10	3.32	10	5.50	4	2.89
20	4.32	20	10.50	5	3.76
30	4.91	30	15.50	5	4.27
40	5.32	40	20.50	6	4.63
50	5.64	50	25.50	6	4.91
60	5.91	60	30.50	6	5.14
70	6.13	70	35.50	7	5.33
80	6.32	80	40.50	7	5.50
90	6.49	90	45.50	7	5.65
100	6.64	100	50.50	7	5.78

The Punchline:
Understanding data structures and algorithms scientifically can make the difference between a good solution and a bad solution, or between success and failure for a programming task!
It also will enable you to survive your first “coding interview”!



16