Comp 11 Lectures

Mike Shah

Tufts University

July 26, 2017

Please do not distribute or host these slides without prior permission.

Searching and Sorting Data

Comp 11 - Pre-Class warm up

Searching and Sorting Data is one of the fundamental computer science problems. You can argue that computers are just simply searching and sorting machines.

Famous Computer Scientist

Ada Lovelace

Figure 1: English mathematician and writer who worked on Charles Baggage's Analytical Engine. She carried out the first algorithm, and is widely known as the first programmer.

Peanut Butter and Jelly Sandwich

- Lets make a peanut butter and jelly sandwich
- Someone give me the instructions?

Lecture

Guessing Game

- Early on, we programmed a simple guessing game.
- The game was to guess a number between 1-10 and the computer would hint to either go higher or lower if you were not correct.
- And the computer could help us, because our data was sorted.
- Inherently, the numbers 1 through 10 (1,2,3,..9,10) are in sorted order.

Guessing Game Algorithm

- What number am I thinking of?
- Let us say the secret answer is 2
- The *smart* guess(given an even distribution of random numbers) is 5.

1	2	3	4	**5**	6	7	8	9	10

Guessing Game Algorithm 2

- Incorrect! The computer tells us lower, so we eliminate half of the possibilities (including 5).
- The *smart* guess (given an even distribution of random numbers) is 2 or 3–we choose 3[1].

1	2	**3**	4	5	6	7	8	9	10

[1] We always choose a higher number when there are an even number of elements in this algorithm

Guessing Game Algorithm 3

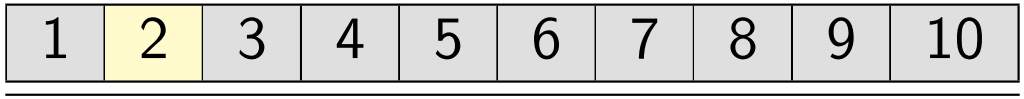
- Incorrect! The computer tells us lower, so we eliminate half of the possibilities (including 3).
- Then we choose 2^2.

1	**2**	3	4	5	6	7	8	9	10

[2]We always choose a higher number when there are an even number of elements in this algorithm

Guessing Game Algorithm 4

- Bingo, we finally got the right answer!
- It took us 3 guesses (5, 3, and 2)



Logarithmic Complexity

- What you may not have realized at the time, is that you solved something known as logarithmic complexity. *Pats on the back*
- If we can eliminate half of the possibilities every time we make progress towards a solution, that is considered logarithmic behavior.
- If we have 10 possibilities(1 through 10 as possible guesses), then mathematically
- $log_2 10 = 3.321928$
- The floor of this equation(rounding down): $\lfloor log_2 10 \rfloor = 3$
- That means, you can always solve this problem in 3 guesses following our algorithm.

Logarithmic Behavior Refresher

2^N	Result
2^1	2
2^2	4
2^3	8
2^4	16
2^5	32
2^6	64

Table 1: 10 Raised to a power

$log_b N$	Result
$log_2 2$	1
$log_2 4$	2
$log_2 8$	3
$log_2 16$	4
$log_2 32$	5
$log_2 64$	6

Table 2: Log base 2

Exponential behavior is the inverse of logarithmic behavior.

Log graph of guessing game

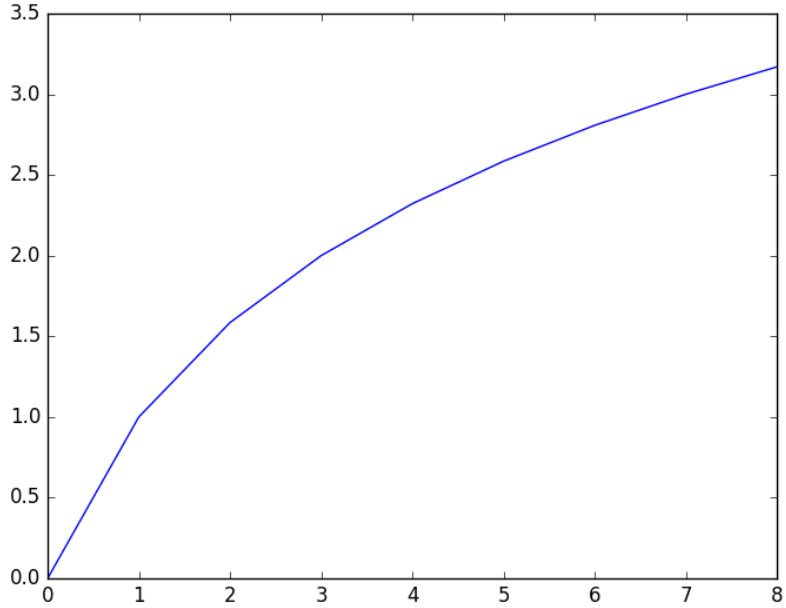


Figure 2: Logarithmic growth. The x-axis is the number of sorted items we are searching, and the y-axis is how many guesses it will take. We generally like this behavior, of a slow growth.

What if the data was not sorted?

- That is, we think about picking the number that the computer picked out of a hat, and that represents our guess.
- The computer can no longer tell us "guess higher" or "guess lower"
- So if my numbers were in random order, what is the algorithm?

7	2	4	3	5	10	1	8	9	6

Randomized picking

- One possibility is to just keep randomly guessing.
- That is, keep picking a number out of a hat, and then putting it back in the hat.
- This means we can get unlucky and guess over and over again the same number.

7	2	4	3	5	10	1	8	9	6

Discarding Strategy

- Another possibility: Linearly scan each item one at a time from the start.
- This time taking out each number and discarding it so it cannot be guessed again
- In the worst case, it takes 10 guesses.

7	2	4	3	5	10	1	8	9	6

Linear

- In this way, we say that the order of magnitude is linear.
- The number of guesses in the worst-case depends on how many times we can pull a number from a hat and discard it.
- In the worst case, it could still take 10 guesses.

```
1 int secretAnswer = 2;
2 int guesses[] = {7,2,4,3,5,10,1,8,9,6};
3
4 for(int i =0; i < 10; i ++){
5   if (guesses[i] == secretAnswer){
6       std::cout << "Correct in: " << i << " guesses";
7     }
8 }
```

Listing 1: Simple linear iteration

When data is sorted, it can help give us some guarantees

- We saw that the logarithmic approach (eliminating half of the possibilities each step) has a worst-case much less than our other strategy (only 3 guesses).
- And typically as a computer scientist, we care about the worst-case.
- Murphy's law paraphrased: if it can happen it will happen (especially in computers), and we want to account for this.

Linear Growth

- As our program grows, perhaps with 1000 guesses, now look at the worst-case number of guesses needed!

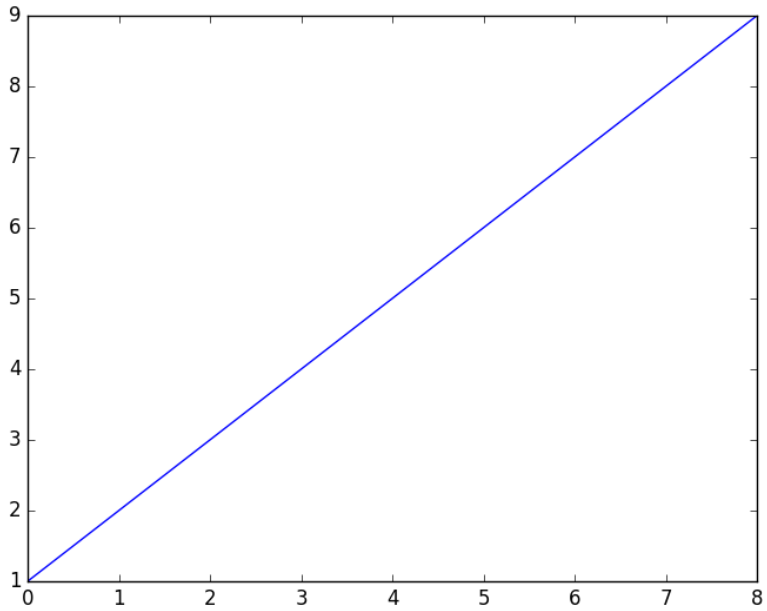


Figure 3: Linear growth. The x-axis is the number of unsorted items we are searching, and the y-axis is how many guesses it will take. Linear is (generally) the best we can do for unsorted data, because we have to look at everything

Linear versus logarithmic growth in search

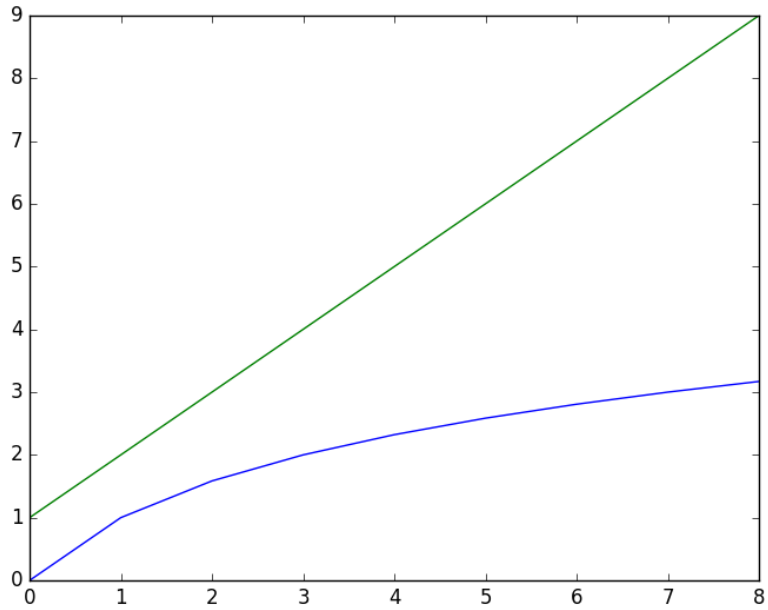


Figure 4: You will notice logarithmic versus linear growth is not even close when plotted. The log line is very close to the y-axis here!

Pop Quiz!

- Find a book in a library of unsorted books?
- How would you find the book you want?

Pop Quiz! – sort!

- Sort the books using some sorting algorithm X,Y, or Z
- If the items are sorted, then we can use our logarithmic strategy of eliminating half of the books each step.
- This was an actual interview question I had long ago, and generally you start by sorting the books and then finding the one you want.

Algorithm X, Y, or Z

- There are actually quite a few sorting algorithms.
- Bubble Sort
- Radix Sort
- Insertion Sort
- Merge Sort
- Heap Sort
- Quick Sort
- Many many more

The first one we will learn!

- There are actually quite a few sorting algorithms.
- **Bubble Sort**
- Radix Sort
- Insertion Sort
- Merge Sort
- Heap Sort
- Quick Sort
- Many many more

Bubble Sort

https://www.youtube.com/watch?v=k4RRi_ntQc8

Figure 5: Former United States President Obama knows about Bubble Sort, so you should too

Terminology

Algorithm

A step by step set of rules(e.g. a recipe) or a process that solves a problem. In Computer Science, algorithms are a set of computations that help us solve problems.

Pseudocode

Not actual code that you can compile. But a code-like form that expresses an algorithm or in a general manner such that you can implement it in a language of your choice.

Brute Force Sort Algorithm

Algorithm 1 Brute Force Sort pseudocode

```
1: elements[N] is an unsorted array
2: for i = 0 to N do
3:     for j = 0 to N do
4:         if elements[i] > elements[j]) then
5:             swap(elements[i], elements[j])
6:         end if
7:     end for
8: end for
```

Question what is the complexity? Analysis

- I have two nested-loops
- They are each of size N

Algorithm 2 Brute Force Sort pseudo code

```
1: elements[N] is an unsorted array
2: for i = 0 to N do
3:     for j = 0 to N do
4:         if elements[i] > elements[j]) then
5:             swap(elements[i], elements[j])
6:         end if
7:     end for
8: end for
```

Question what is the complexity? Explanation

- Well if it is N by N, then N^2 operations must take place
- So for an unsorted list, it will not be sorted until all N^2 operations take place.
- And within each of those loops, some swap operation is occurring.
- Swap is constant time, meaning there are a fixed number of operations taking place.

Algorithm 3 Brute Force Sort pseudo code

```
1: elements[N] is an unsorted array
2: for i = 0 to N do
3:     for j = 0 to N do
4:         if elements[i] > elements[j]) then
5:             swap(elements[i], elements[j])
6:         end if
7:     end for
8: end for
```

Constant time operation

- Let's assume swap looks something like this
- A total of three operations.
- Let's call them T.

Algorithm 4 Swap function code

1: temp= i
2: i= j
3: j= $temp$

Precise analysis

- N^2 loops
- Within the loop there are T operations taking place.
- So T*N^2 is a more precise analysis

Algorithm 5 Brute Force Sort pseudo code

```
1: elements[N] is an unsorted array
2: for i = 0 to N do
3:     for j = 0 to N do
4:         if elements[i] > elements[j]) then
5:             swap(elements[i], elements[j])
6:         end if
7:     end for
8: end for
```

Dominating term

- T*N^2 for Brute Force Sort
- Generally we only care about the biggest factor here(N has a degree of 2), in which case is the N^2.
- So we say Brute Force Sort takes N^2 time.

Ordering

- What happens if I switch the comparison sign in the Brute Force Sort algorithm?
- That is, was I sorting in ascending or descending order?

Dominating term - Examples

As N gets really large, one term will end up dominating, so we typically only care about that term.

- N^2 + N is considered O(N^2)
- N^{999} + N^{998} is considered O(N^{999})
- N^{999} + N^{998} + N^{997} + N^{996} is still O(N^{999})
- $log_2 N$ + 100000000 + N is O(N)

Worst-case analysis

- What we have done here is a worst-case analysis, this is known as Big-O analysis. $\mathcal{O}(N^2)$
- There is average case analysis. $\Theta(N^2)$
- And finally best case analysis. $\omega(N^2)$
- The Big-O runtime is what you will most often see. $\mathcal{O}(complexity)$

In-Class Activity

http://www.mshah.io/comp/11/activities/activity13/activity.pdf

Activity Discussion

Review of what we learned

- (At least) Two students
- Tell me each 1 thing you learned or found interesting in lecture.

5-10 minute break

To the lab!

Lab: `http://www.mshah.io/comp/11/labs/lab13/lab.pdf`
[3]

[3] You should have gotten an e-mail and hopefully setup an account at `https://www.eecs.tufts.edu/~accounts` prior to today. If not–no worries, we'll take care of it during lab!

Python fun

You can plot graphs of functions using the following python code.

```
1  import matplotlib.pyplot as plt
2  import math
3
4  fig, ax = plt.subplots()
5
6  logGrowth = []
7  linearGrowth = []
8  quadraticGrowth = []
9  for i in range(1,1000):
10   linearGrowth.append(i)
11   logGrowth.append(math.log(i,2))
12   quadraticGrowth.append(i*i);
13
14 ax.plot(logGrowth)
15 ax.plot(linearGrowth)
16 ax.plot(quadraticGrowth)
17
18 plt.show()
```

Listing 2: Plot in the Python Language

Glossary

Algorithm A step by step set of rules(e.g. a recipe) or a process that solves a problem. In Computer Science, algorithms are a set of computations that help us solve problems. 29

pseudocode Not actual code that you can compile. But a code-like form that expresses an algorithm or in a general manner such that you can implement it in a language of your choice. 29