

Algorithm Design

Algorithm

- Origins of the word:
 - 9th century Muslim mathematician **Abu Abdullah Muhammad ibn Musa al-Khwarizmi** whose works introduced Arabic numerals and algebraic concepts.
 - The word *algorism* originally referred only to the rules of performing arithmetic using Arabic numerals
 - Evolved via European Latin translation of **al-Khwarizmi**'s name into *algorithm* by the 18th century.
- The word evolved to include *all definite procedures for solving problems or performing tasks.*

2

Algorithm Design

- An **algorithm** is an ordered set of unambiguous, executable steps, defining a terminating process
- May be described
 - **Abstractly** using human language (pseudocode) describing the steps for carrying out some procedure using a computer
 - Using a **programming language** of your choice
 - By providing a set of **machine instructions** to be executed

3

Algorithm

- An **algorithm** is an ordered set of unambiguous, executable steps, defining a terminating process
- Is the following an algorithm?

Calculate $1/3$ exactly

- No, because $1/3 = 0.333333\dots$ and this algorithm does not **terminate**

4

Algorithm

- An **algorithm** is an ordered set of unambiguous, executable steps, defining a terminating process
- Is the following an algorithm?

Go for a walk

- No, because it is **ambiguous**: duration, location of the walk unspecified, how does one walk?.

5

Algorithm

- An **algorithm** is an ordered set of unambiguous, executable steps, defining a terminating process
- Is the following an algorithm?

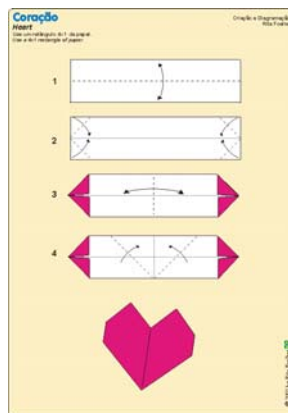
Find the third largest number in the list {3, 5}

- No, because it is not **executable**

6

Algorithm

- An **algorithm** is an ordered set of unambiguous, executable steps, defining a terminating process
- Is the following an algorithm?



7

Algorithm Design

- **Pseudocode** is a **programming language independent** description of the sequence of steps necessary to solve a problem
- Algorithms that are written in pseudo-code may be then **translated into a particular programming language** to make a **computer program**
- A programmer may come up with his/her own algorithm, or (s)he may *implement* an existing algorithm

8

Algorithm Design

- Given a **problem** (e.g., find the largest number in a list of numbers, arrange the given numbers in sequence, convert from binary to decimal, etc.) seek the **an algorithm that solves the problem**
- Issue to be concerned with: **efficiency**
- Want to come up with a solution that uses **computational resources** (CPU cycles, storage) as efficiently as possible

9

Algorithm Design

- Coming weeks:
 - No more programming
 - Giving English-language (pseudocode) solutions to problems
 - Study examples of (relatively simple) problems in searching, sorting, graph theory, text encoding, computational geometry, ...

10

Searching

- **Given:** a list of names of students in a class and their favourite colour
- **Want:** find the favourite colour of the student Alice, if she is in this class

1. Bob 'black'
2. Mary 'red'
3. Carol 'yellow'
4. Allison 'blue'
5. Alice 'yellow'
6. Joe 'green'
7. Joseph 'purple'

11

Sequential Search

Process each list entry from first to last

Check if each entry processed in this order is the entry for Alice

If we found the 'Alice' entry,

Note Alice's favourite colour.

Stop searching.

- How many entries in the list are processed before Joe is found?

1. Bob 'black'
2. Mary 'red'
3. Carol 'yellow'
4. Allison 'blue'
5. Alice 'yellow'
6. Joe 'green'
7. Joseph 'purple'

12

Sequential Search

- How many entries in the list are processed before Joe is found?

1. Bob 'black'
2. Mary 'red'
3. Carol 'yellow'
4. Allison 'blue'
5. George 'green'
6. Billy 'white'
7. Walter 'yellow'
8. Geoffrey 'pink'
9. Alice 'yellow'
10. Joe 'green'
11. Joseph 'purple'

13

Binary Search

- Looking up a word in the dictionary
- Do you use sequential search?
- Why/why not?

14

Insertion Sort

- A list with a single name is sorted
- A list with 2 names is sorted by:
 - swapping the 2 names if necessary
- A list with 3 names is sorted by:
 - sorting the first 2 names
 - inserting the 3rd name into the correct location and moving the first 2 names down by one location
- A list with 4 names is sorted by:
 - sorting the first 3 names
 - ...

15

Insertion Sort

1. Bob 'black'	1. Bob 'black'	1. Bob 'black'	1. Allison 'blue'
2. Mary 'red'	2. Mary 'red'	2. Carol 'yellow'	2. Bob 'black'
3. Carol 'yellow'	3. Carol 'yellow'	3. Mary 'red'	3. Carol 'yellow'
4. Allison 'blue'	4. Allison 'blue'	4. Allison 'blue'	4. Mary 'red'
5. Alice 'yellow'	5. Alice 'yellow'	5. Alice 'yellow'	5. Alice 'yellow'
6. Joe 'green'	6. Joe 'green'	6. Joe 'green'	6. Joe 'green'
7. Joseph 'purple'	7. Joseph 'purple'	7. Joseph 'purple'	7. Joseph 'purple'

16

Insertion Sort

1. Alice 'yellow' 1. Alice 'yellow' 1. Alice 'yellow'
2. Allison 'blue' 2. Allison 'blue' 2. Allison 'blue'
3. Bob 'black' 3. Bob 'black' 3. Bob 'black'
4. Carol 'yellow' 4. Carol 'yellow' 4. Carol 'yellow'
5. Mary 'red' 5. Joe 'green' 5. Joe 'green'
6. Joe 'green' 6. Mary 'red' 6. Joseph 'purple'
7. Joseph 'purple' 7. Joseph 'purple' 7. Mary 'red'

17

Sequential Search on Sorted List

- Can you speed up sequential search if the list is sorted?

1. Alice 'yellow'
2. Allison 'blue'
3. Bob 'black'
4. Carol 'yellow'
5. Joe 'green'
6. Joseph 'purple'
7. Mary 'red'

18

Sequential Search on Sorted List

- What if you are looking for Isabelle?

1. Alice 'yellow'
2. Allison 'blue'
3. Bob 'black'
4. Carol 'yellow'
5. Joe 'green'
6. Joseph 'purple'
7. Mary 'red'

19

Binary Search

- Works on sorted lists
- How do you find a word in the dictionary?

20

Binary Search

- Search(dictionary)

- Look at the **middle** page of the dictionary
- If the word you are looking for comes **after** the words on this page, **search** among the pages of the dictionary that come after this page
- If the word you are looking for comes **before** the words on this page, **search** among the pages of the dictionary that come before this page
- If the word you are looking for is **on** this page, stop searching.

21

Binary Search: Joe

1. Alice 'yellow'
2. Allison 'blue'
3. Bob 'black'
4. Carol 'yellow'
5. Joe 'green'
6. Joseph 'purple'
7. Mary 'red'



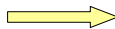
Check Carol, then Joseph, then Joe

3 items examined vs 5 for sequential search

22

Binary Search: Walter

1. Alice 'yellow'
2. Allison 'blue'
3. Billy 'white'
4. Bob 'black'
5. Carol 'yellow'
6. Geoffrey 'pink'
7. George 'green'
8. Joe 'green'
9. Joseph 'purple'
10. Mary 'red'
11. Walter 'yellow'



23

Worst-Case Complexity

- Binary search seems faster than sequential search for sorted lists
- Formal notion: **time complexity**
- Interested in analyzing the worst-case time complexity of algorithms
- If there are 7 elements in a list, then performing sequential search looks at ??? elements of the list in the worst case? Answer: **7**

24

Worst-Case Complexity

- What about binary search?

1. Alice 'yellow'
2. Alisson 'blue'
3. Bob 'black'
4. Carol 'yellow'
5. Joe 'green'
6. Joseph 'purple'
7. Mary 'red'



- Here, at most 3 elements of the list need to be analyzed

25

Worst-Case Complexity

- So, binary search is much faster than sequential search!
- Example:
 - To search a list of 8,388,607 names using **sequential search** examines all 8,388,607 names
 - To search a list of 8,388,607 names using **binary search** examines 23 names

26

But!

- Binary search only works on **sorted** lists!
- To sort using insertion sort a list with 7 names examines this many elements (in the worst case):
 - 2 elements to swap first 2 names
 - 3 elements to insert the 3rd name
 - 4 elements to insert the 4th name
 - 5 elements to insert the 5th name
 - 6 elements to insert the 6th name
 - 7 elements to insert the 7th name

 $2+3+4+5+6+7 = 27$ names examined

27

Binary vs. Sequential Search

- To search a list of 7 elements, **binary search** examines 3 elements, while **sequential search** examines all 7
- But, in order to **sort the list**, must examine 27 list elements
- What is the advantage of binary vs. sequential search?

28

Binary vs. Sequential Search

- Need to **sort only once** and can then search the sorted list using binary search as many times as we want.
- In case don't want to do **multiple searches** on one list, better off just doing a sequential search whose performance isn't affected by whether or not the list is sorted.

29

We Just Learned

- ✓ What is and is not an algorithm
- ✓ Sequential search, insertion sort, binary search algorithms
- ✓ By counting the number of times each element of the list was examined in the worst case, how to compare search algorithms in terms of their *worst-case time complexity*
- ✓ Situations where binary search is better than sequential search and vice versa

30