Teach A level Computing: Algorithms and Data Structures

Eliot Williams

@MrEliotWilliams

Queen Mary University of London · SUPPORTED BY MAYOR OF LONDON · COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE · KING'S College LONDON

Course Outline

1	**Representations of data structures: Arrays, tuples, Stacks, Queues,Lists**
2	Recursive Algorithms
3	Searching and Sorting - EW will be late!
4	Hashing and Dictionaries, Graphs and Trees
5	Depth and breadth first searching ; tree traversals

Queen Mary University of London · SUPPORTED BY MAYOR OF LONDON · COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE · KING'S College LONDON

Searching, sorting and algorithm analysis

Queen Mary University of London SUPPORTED BY MAYOR OF LONDON COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE KING'S College LONDON

Algorithm Analysis

- An algorithm is a self-contained sequence of actions to be performed
- We can analysis an algorithms efficiency using the following measures
  - The time it takes to finish
  - How much memory it needs

Queen Mary University of London SUPPORTED BY MAYOR OF LONDON COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE KING'S College LONDON

Measuring time

- Firsttime.py

- Run the program how long do the simple expressions take
- Change the numbers and operations in the first three statements

- Uncomment the further statements (each block at a time)

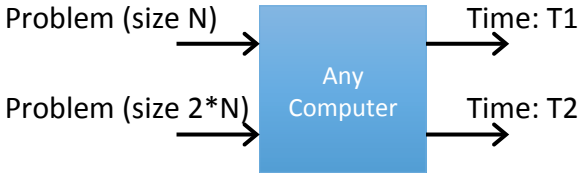
Measuring time

- Firsttime.py

- Different computers will produce different times
- Time taken to execute a command is not equal on the same computer
- Time taken to execute a sequence of commands grows with the length of the sequence

We don't need absolute time measures

- We want to understand fundamental properties of our algorithms, not things specific to a particular input or machine.



- We therefore only care about how the time increases
  - Maybe the time stays the same
  - Maybe doubling the size, doubles the time
  - Maybe doubling the size, more than doubles the time

Big O Notation

- Big O Notation is a formal way of stating how the time taken by an algorithm relates to an input size

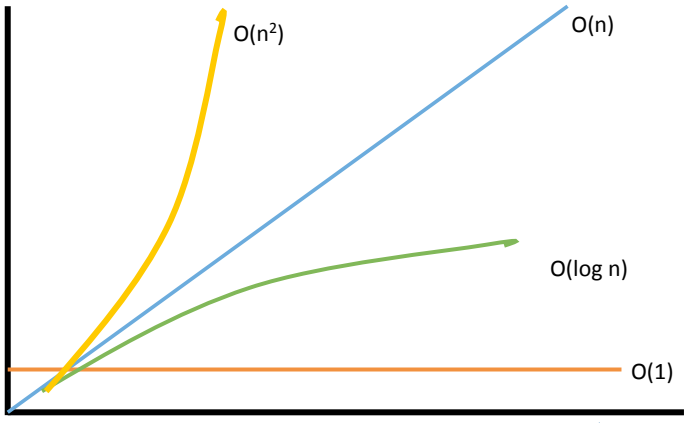
- Try secondtime.py
- and secondtimen.py

- O(1) input size does not effect the time
- O(n) input grows linearly with input size

Big O Notation

- Big O Notation is a formal way of stating how the time taken by an algorithm relates to an input size –we only need to consider the largest term

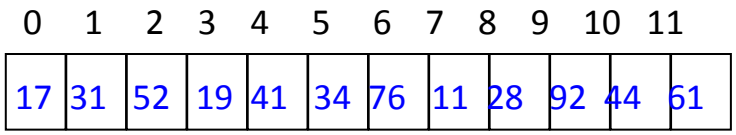
- O(1) input size does not effect the time
- O(log n) logarithmic growth
- O(n) time grows linearly with input size
- $O(n^2)$ time is directly proportional to the square of the input size
- $O(n^y)$ Exponential growth
- $O(2^n)$ polynomial growth – intractable problems….



Queen Mary University of London SUPPORTED BY MAYOR OF LONDON COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE KING'S College LONDON

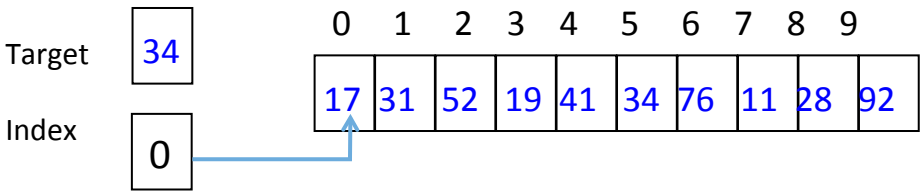
Search: The Problem

- Find a target value in the list
  - Is it there?
  - If so, at what index?



- Target = 41, found at index 4
- Target = 27, not found

Linear Search



- Idea: look at each entry in turn
- Steps
  - Start at index = 0
  - Is Array[Index] equal to the target?
  - If yes, stop; otherwise increment the index
  - Stop when index is one less than the length

Exercise 1.1 and 1.2

Linear Search – Algorithm

- Algorithm in pseudo code
- Array is A

```
index = 0
while index < length of array
   if A[index] equals target
      return index
   index = index + 1
return -1 to show not found
```

```
for i is 0 to length of array-1
   if A[i] equals target
      return i
return -1 to show not found
```

Binary Search

Searching a sorted list

Queen Mary University of London SUPPORTED BY MAYOR OF LONDON COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE KING'S College LONDON

Searching a Sorted List

- Question: why are books in the library kept in order?

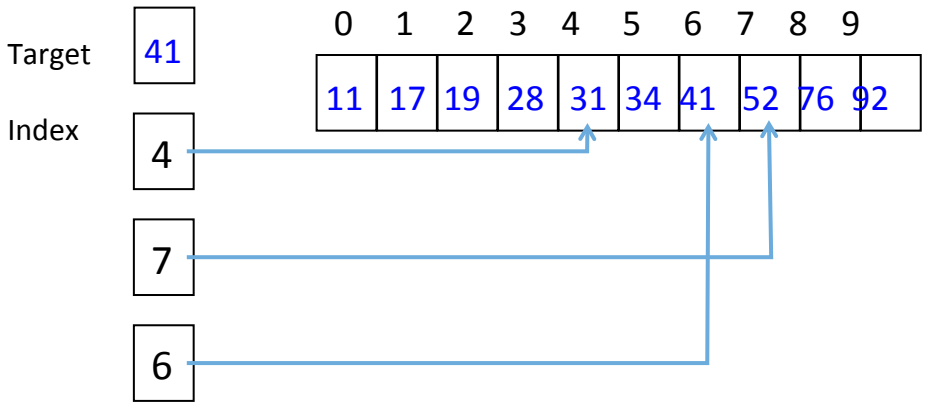
Queen Mary University of London SUPPORTED BY MAYOR OF LONDON COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE KING'S College LONDON

Searching a Sorted List

- Question: why are books in the library kept in order?
- *In an ordered array, we do not have to look at every item*
  - "Before this one"
  - "After this one"
  - … quickly find the correct location
- What is the best algorithm for looking?

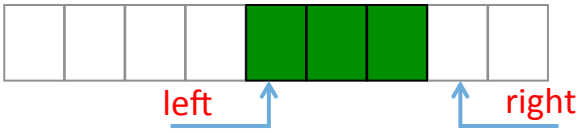
Binary Search – Sorted Lists

- Which half is it in? Look in the middle.

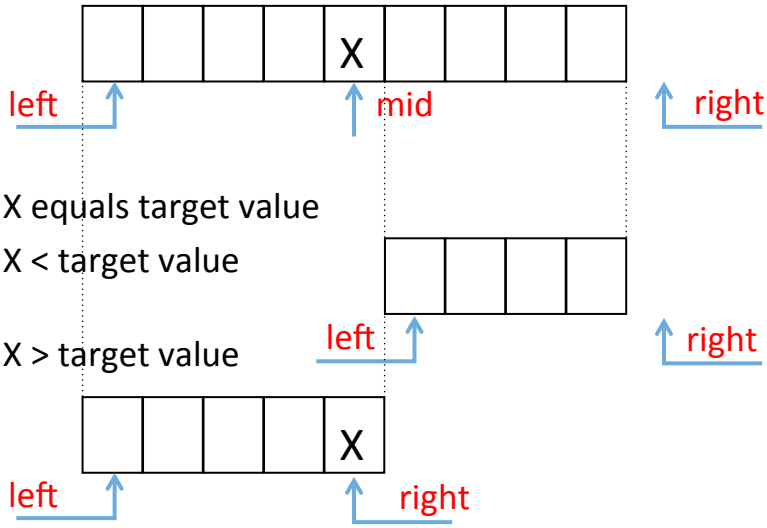


Binary Search – Algorithm

- Key idea: in which part of the array are we looking?



Binary Search – 3 Cases



- Case 1: X equals target value
- Case 2: X < target value
- Case 3: X > target value

Binary Search – Algorithm

```
left = 0
right = length of array
while right > left:
   mid = average of left and right
   if A[mid] equals target
      found it at 'mid'
   if A[mid] < target
      search between mid+1 & right
   otherwise
      search between left & mid
return not found
```


Binary Search – Python

```
def BSearch(A, target):
   left = 0
   right = len(A)
   while right > left:
      mid = (left + right) // 2
      if A[mid] == target:
         return mid
      elif A[mid] < target:
         left = mid+1
      else:
         right = mid
   return -1
```

Binary Search – Complexity

0	**0**	**0**	**0**	1	1	1	1
0	0	**1**	**1**	0	0	1	1
0	1	**0**	1	0	1	0	1

11	17	19	28	31	34	41	52

- Number of steps = number of binary digit to index
- O(log N)

Binary Search (Recursive)

- Search a sorted array – is E in the array?

- Base case:
  - empty array or E found
- Recursive case:
  - Compare E with the middle element
  - If E smaller, search the left half
  - If E larger, search the right half

Exercise 1.3

- *Complete the iterative and recursive binary search procedures in binarysearch.py*

Queen Mary University of London SUPPORTED BY MAYOR OF LONDON COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE KING'S College LONDON

Sorting

Queen Mary University of London SUPPORTED BY MAYOR OF LONDON COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE KING'S College LONDON

Sorting: The Problem

0	1	2	3	4	5	6	7	8	9
17	31	52	19	41	34	76	11	28	92

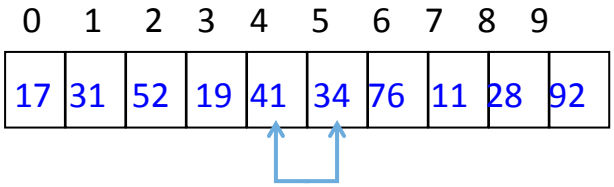
11	17	19	28	31	34	41	52	76	92

- Arrange array in order
  - Same entries; in order – swap entries
- Properties
  - Speed, space, stable,

Discussion

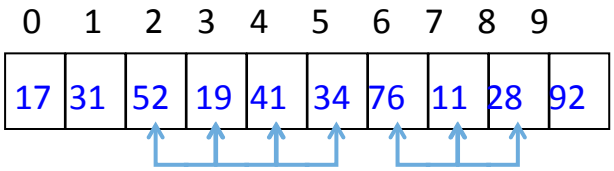
- Sort a pack of top trumps
- Describe how you do it

Bubble Sort – Insight



- Librarian finds two books out of order
- Swap them over!
- Repeatedly

Bubble Sort – Description



- Pass through the array (starting on the left)
- Swap any entries that are out of order
- Repeat until no swaps needed

Quiz: show array after first pass

Bubble Sort – Algorithm

- Sorting Array A
  - Assume indices 0 to length-1

```
while swaps happen
   index = 1
   while index < length
      if A[index-1] > A[index]
         swap A[index-1] and A[index]
      index = index + 1
```


Exercise 2.1 Bubble Sort

- Complete the table to show the successive passes of a bubble sort

Queen Mary University of London SUPPORTED BY MAYOR OF LONDON COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE KING'S College LONDON

Demo

sortingDemo.py

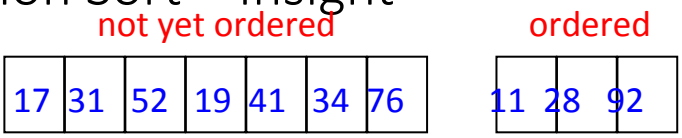
Bubble Sort – Properties

- Stable
- Inefficient
- $O(N^2)$
  - Double length – time increases 4-fold

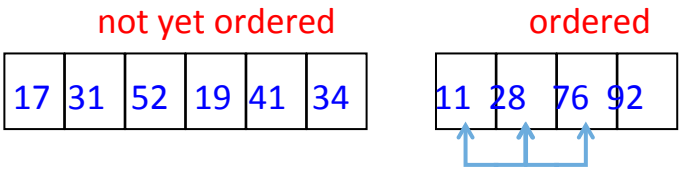
http://www.sorting-algorithms.com/bubble-sort

Queen Mary University of London SUPPORTED BY MAYOR OF LONDON COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE KING'S College LONDON

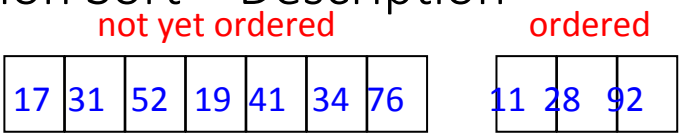
Insertion Sort – Insight



- Imagine part of the array is ordered
- Insert the next item into the correct place



Insertion Sort – Description



- Start with one entry – ordered
- Take each entry in turn
- Insert into ordered part by swapping with lower values
- Stop when all entries inserted

Insertion Sort – Algorithm

- Sorting Array A
  - Assume indices 0 to length-1

```
index = 1
while index < length of array
   ix = index
   while A[ix] < A[ix-1] and ix > 0
      swap A[ix] and A[ix-1]
      ix = ix – 1
   index = index + 1
```

- A[0:index] ordered
- Same values

Inner loop: insert into ordered list

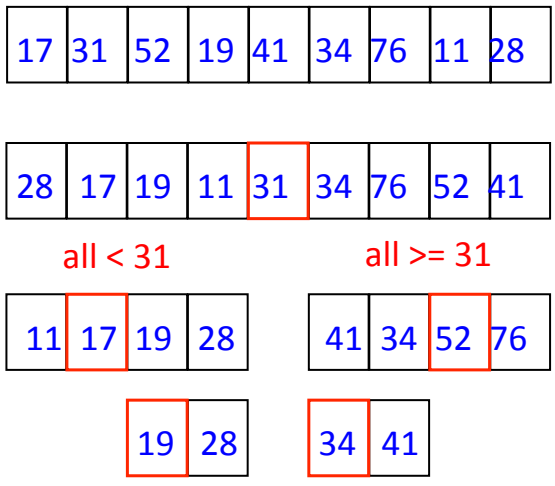
Exercises 3.x

Quicksort – Insight

- How could we share out sorting between two people?
  - Choose a value V
  - Give first person all values < V
  - Give second person all values > V

  - When there is only a single entry – it is sorted

Quicksort Example



Quicksort Description

- Choose a **pivot value**
- Partition the array
  - Values less than the pivot to the left
  - The pivot
  - Values greater than the pivot to the right
- Repeat process on each partition
- … until partition has no more than one value

- **Work done in partition**

Tasks

- Quickfirststep.py – going through only once
  - quickS1
    - Separate into two new lists and put back together with pivot
    - Checks basic understanding

Queen Mary University of London SUPPORTED BY MAYOR OF LONDON COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE KING'S College LONDON

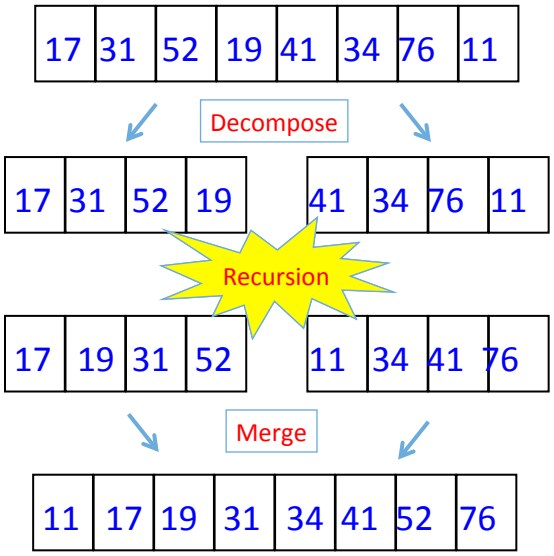
Recursive Sorting

- Concept
  - Split array in two halves, sort each half
  - Combine two sorted arrays
  - Single item sorted (base case)
- Two algorithms
- Merge sort
  - Halve array – merge two sorted lists
- Quicksort
  - Partition array – combine easy

Merge Sort – Insight

- How could we share out sorting between two people?
  - Half it and sort each half
  - Merge the two sorted lists
  - When there is only a single entry – it is sorted

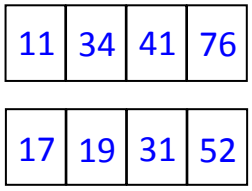
Merge Sort

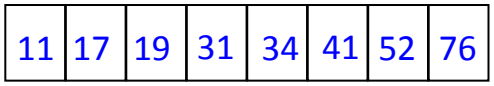


Queen Mary University of London SUPPORTED BY MAYOR OF LONDON COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE KING'S College LONDON

Merging

- Work done in the merge
- Repeatedly select smallest





Queen Mary University of London SUPPORTED BY MAYOR OF LONDON COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE KING'S College LONDON

Properties

- Insertion sort http://www.sorting-algorithms.com/insertion-sort
  - $O(N^2)$ – same as bubble sort
  - Stable
- Quicksort http://www.sorting-algorithms.com/quick-sort
  - More efficient: O(N logN)
  - Not stable
- Merge sort http://www.sorting-algorithms.com/merge-sort
  - O(N log N) – same as quick sort but extra space
  - Stable