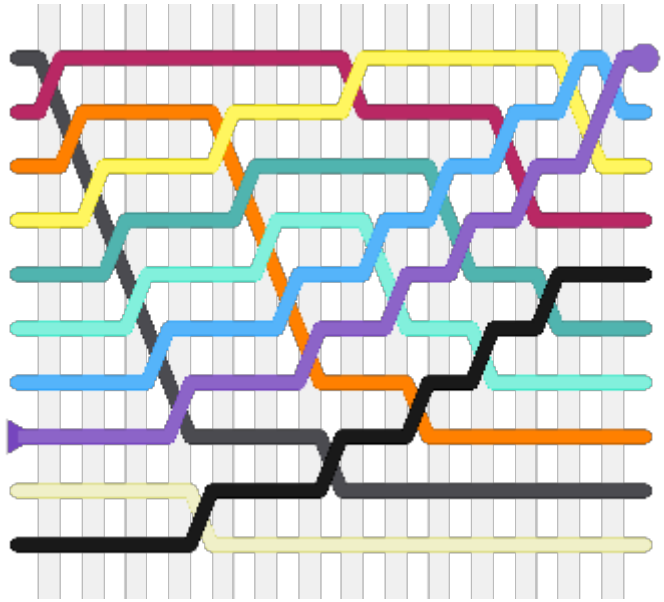


Sorting Algorithms!

Lab 06

Sorting Algorithms

- This is one of the few times Wikipedia is your friend.
  - https://en.wikipedia.org/wiki/Bubble_sort
  - https://en.wikipedia.org/wiki/Selection_sort
  - https://en.wikipedia.org/wiki/Insertion_sort

- YouTube and other sites also have lots of visualisations of sorting algorithms which will walk you through how they work

Bubble Sort

- It is called bubble sort because it looks like it is 'bubbling' larger numbers to the top (right) of a list

- Bubble Sort works by repeatedly looping through a list and swapping adjacent numbers which aren't in order until the list is sorted
  - E.g. [..., 4, 2, ...] -> [..., 2, 4, ...]

- Every time we loop through the list swapping pairs of numbers, we bring the next largest number to its sorted place in the top of the list

Bubble Sort – How it Works

- Bubble Sort works by repeatedly looping through a list and swapping adjacent numbers which aren't in order until the list is sorted

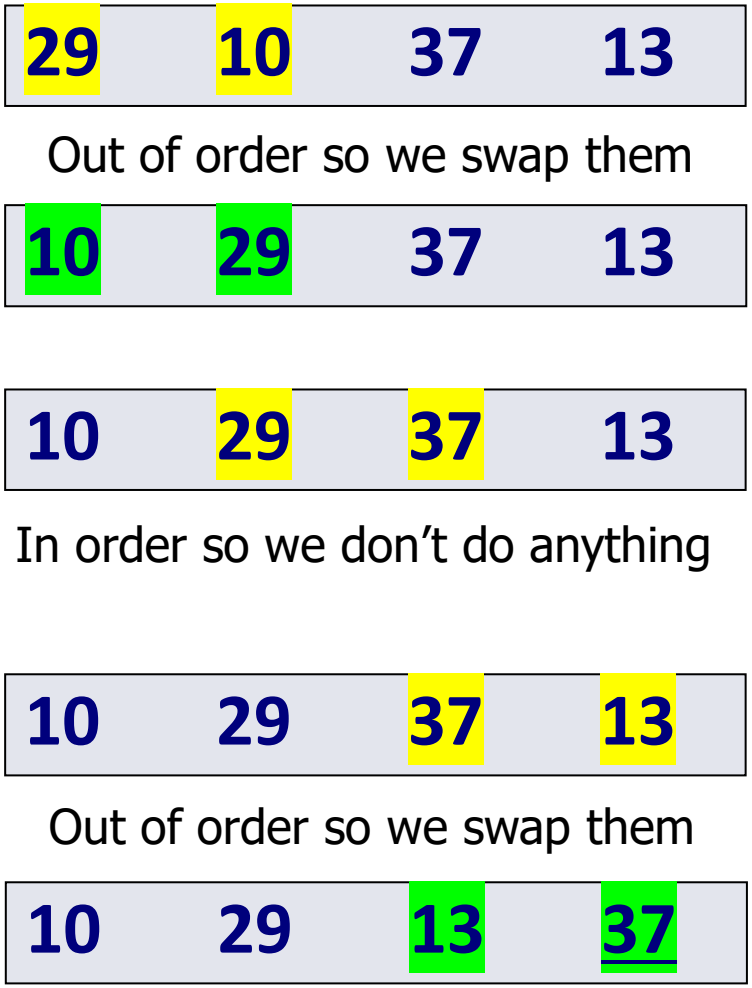
- Given a list of numbers containing n items
- Repeat the following n-1 times:
  - Loop through all pairs of adjacent numbers in a list except for the top i numbers in the list where i is the number of passes we have made through the list already:
    - If two adjacent numbers are out of order:

      Swap those two numbers in the numbers list

Bubble Sort – Example

- Given a list of numbers containing 4 items
- Repeat the following 3 times: # 1st Pass
  - Loop through all pairs of adjacent numbers in a list except the top i numbers:
    - If two adjacent numbers are out of order:

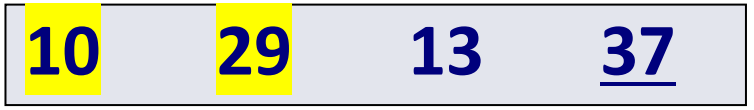
      Swap those two numbers in the numbers list



Bubble Sort – Example

- Given a list of numbers containing 4 items
- Repeat the following 3 times: # 2nd Pass
  - Loop through all pairs of adjacent numbers in a list except the top i numbers:
    - If two adjacent numbers are out of order:

      Swap those two numbers in the numbers list



In order so we don't do anything

Out of order so we swap them

We don't need to check the top number because we know it is in the correct position

Bubble Sort – Example

- Given a list of numbers containing 4 items
- Repeat the following 3 times: # 3rd Pass
  - Loop through all pairs of adjacent numbers in a list except the top i numbers:
    - If two adjacent numbers are out of order:

      Swap those two numbers in the numbers list

10	13	29	37

In order so we don't do anything

10	13	29	37

We don't need to compare the top two numbers because we know they are in order

10	13	29	37

We know 10 must be sorted because the larger numbers are all above it on the last pass

Selection Sort

- We repeatedly loop through a list of numbers and select the n^{th} largest number and will swap that number with the number in the $-n^{th}$ position in the list.
- E.g.
  - [2, 5, 4, 1, 3]
    5 is the 1^{st} largest number in the list, so we swap it with the number in the -1^{st} (last) position in the list
    -> [2, 3, 4, 1, 5]

Selection Sort – How it Works

- We repeatedly loop through a list of numbers and select the n^{th} largest number and swap that number with the number in the $–n^{th}$ position in the list.
- Given a list of numbers containing n items
- Repeat the following n-1 times:
  - Find the index of the largest number from the first n-i items (where i is the number of passes we have made through the list)
  - Swap that number with the number in the $–(i+1)^{th}$ place in the list

Selection Sort – Example

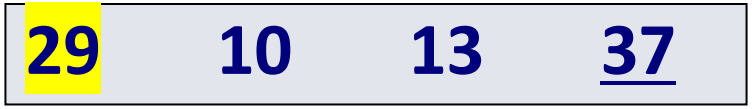
- Given a list of numbers containing n items
- Repeat the following n-1 times: # 1st pass
  - Find the index of the largest number from the first n-i items
  - Swap that number with the number in the –(i+1)th place in the list

37 is the 1st largest number

We swap 37 with the last number in the list

Selection Sort – Example

- Given a list of numbers containing n items
- Repeat the following n-1 times: # 2nd pass
  - Find the index of the largest number from the first n-i items
  - Swap that number with the number in the –(i+1)th place in the list



29 is the 2nd largest number (or the largest number of the first 3 numbers)

We swap 29 with the second to last number in the list

Selection Sort – Example

- Given a list of numbers containing n items
- Repeat the following n-1 times: # 3rd pass
  - Find the index of the largest number from the first n-i items
  - Swap that number with the number in the –(i+1)th place in the list

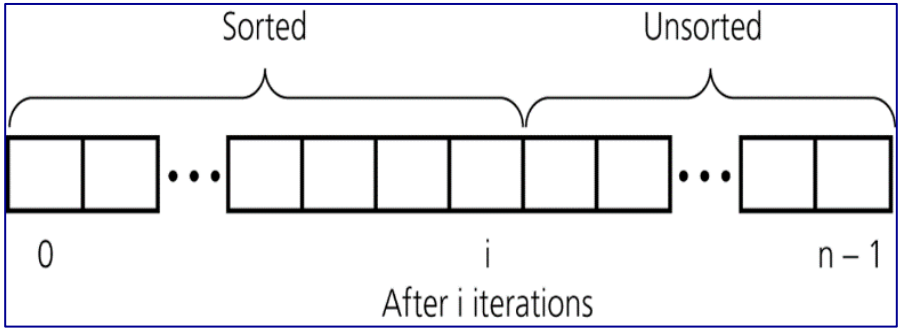
13 is the 3rd largest number (or the largest number of the first 2 numbers)

We swap 13 with the third to last number in the list

Like with bubble sort – we now know 10 is also in the right position

Insertion Sort

- We slowly build up a list of sorted numbers by inserting the unsorted numbers from the input list into the correct position one at a time
- However, we do this by shifting numbers around in the same list and keeping track of what part is sorted and what part is unsorted



Insertion Sort – How it Works

- We slowly build up a list of sorted numbers by inserting the unsorted numbers from the input list into the correct position one at a time

- Given a list of numbers containing n items
- Repeat the following n-1 times:
  - Divide list into sorted (left – i+1 elements) and an unsorted part (right) where i is the number of passes we have currently done
  - In each pass take left most element from unsorted part and place it into correct position of sorted part

Insertion Sort – Example

- Given a list of numbers containing n items
- Repeat the following n-1 times:# 1st pass
  - Divide list into sorted (left – i+1 elements) and an unsorted part (right) where i is the number of passes we have currently done
  - In each pass take left most element from unsorted part and place it into correct position of sorted part

10 is the first unsorted element

We insert 10 into the correct position in the sorted list

Insertion Sort – Example

- Given a list of numbers containing n items
- Repeat the following n-1 times:# 2nd pass
  - Divide list into sorted (left – i+1 elements) and an unsorted part (right) where i is the number of passes we have currently done
  - In each pass take left most element from unsorted part and place it into correct position of sorted part

10 29 | 37 13

37 is the first unsorted element

10 29 37 | 13

We insert 37 into the correct position in the sorted list

Insertion Sort – Example

- Given a list of numbers containing n items
- Repeat the following n-1 times:# 3rd pass
  - Divide list into sorted (left – i+1 elements) and an unsorted part (right) where i is the number of passes we have currently done
  - In each pass take left most element from unsorted part and place it into correct position of sorted part

13 is the first unsorted element

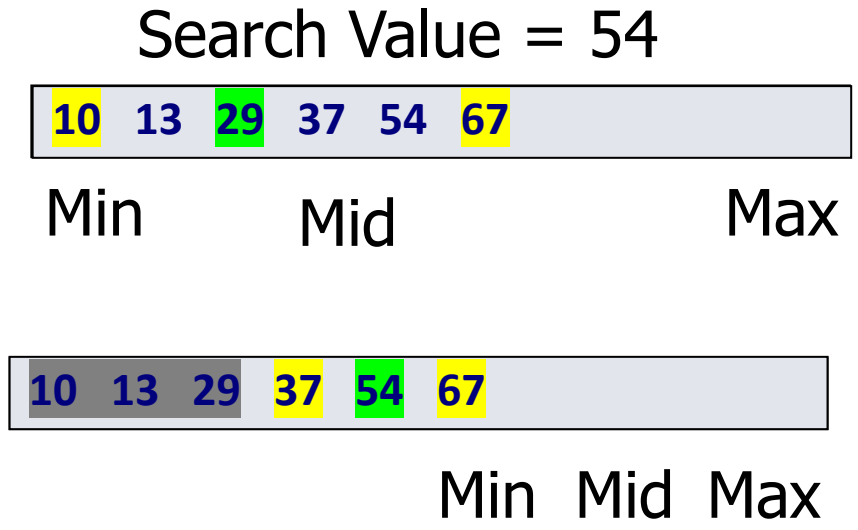
We insert 13 into the correct position in the sorted list

Binary Search!

- Useful for searching through sorted data quickly (log(n) time!!!)
- Pros: Quick
- Cons: Data has to be sorted

Binary Search Example

- Given a sorted list of n items
- Set the current max_index to the last index
- Set the current min_index to the first index
- While the max_index is bigger than the min_index:
  - Calculate the mid_index by getting the average of the max and min indicies
  - If the value at the mid_index == the search value we can return
  - If the value at the mid_index is less than the search value we know the search value is going to be above the mid_index.
    - Set the min_index to the index above the mid_index
  - Otherwise:
    - Set the max_index to one below the mid_index
- If we get outside the while loop the value is not in the list



CodeRunner

- Bubble Sort
  - Test case
  - Implement the bubble sort
- Selection Sort
  - Test case
  - Get the largest element
  - Implement the selection sort
- Insertion Sort
  - Test case
  - Shifting elements
  - Implement the insertion sort
- Binary search function

IMPORTANT CODERUNNER TIP!!

- A lot of the code for these questions might be shorter than you think!
- It is *really really* helpful to get a pencil and paper and try doing these algorithms by hand thinking carefully about what operations you need to do in what order if you're not familiar with them
- If your code is starting to get super duper long – it might be time to rethink your approach