**Searching Algorithms**

**Pair work:**

Linear Search:

1. Describe the steps of the linear search algorithm. (Sentences)

It can be an un ordered list, start from first item in list, examine it, if it is the item you are looking for, return index and the item saying it was found and end search, if not move on to the next item and repeat, if the item isn’t in the data, return -1 and say its doesn’t exist in the data.

1. Write down pseudocode for a linear search algorithm. The algorithm should return the array index of the item searched for, or -1 if the search item does not exist in the list.
2. def linearSearch(array, arrayLength, requestedItem):
3. pointer = 0
4. while True:
5. if array[pointer] == requestedItem:
6. return array[pointer], pointer
7. elif pointer == arrayLength -1:
8. return -1
9. else:
10. pointer +=1
11. *#endif*
12. *#endwhile*
13. *#endfunction*
14. Write down the worst, best, and average case time complexity of linear search.

Worst = item at the very end of list

Best = item is the first item

Average = item is in the centre of list

Binary search:

1. Describe the steps of the binary search algorithm. (Sentences)
2. Write down pseudocode for a binary search algorithm. The algorithm should return the array index of the item searched for, or -1 if the search item does not exist in the list.
3. def binarySearch(array, arraylength, requestedItem):
4. startPointer = 0
5. endPointer = arraylength - 1
6. middlePointer = (endPointer - startPointer)//2
7. current = array[middlePointer]
8. while True:
9. if current == requestedItem:
10. return array[middlePointer], middlePointer
11. elif requestedItem > current:
12. startPointer = middlePointer
13. middlePointer += ((endPointer - startPointer)//2)
14. elif requestedItem < current:
15. endPointer = middlePointer
16. middlePointer -= ((endPointer - startPointer)//2)
17. else:
18. return -1
19. *#endif*
20. *#endwhile*
21. *#endfunction*
22. Write down the worst and best case time complexity of binary search.

Worsts = item either at very beginning, very end

Best = item is at the very centre of list (first middle pointer)

Average = item is either in the ¼ or ¾ length of list positions.

Write down the maximum number of array items that must be checked when conducting a binary search for an array of length:

1. 16 5
2. 140 9
3. 600 11
4. 1,000,000 21

How would you express the method for calculating the maximum number of array items that must be checked for an array of length n?

Log base 2 to the n then add 1

Use Cases:

* Describe a list for which linear search would be more appropriate.

List is unordered, of a very short list

* Describe a list for which binary search would be more appropriate.

List is ordered from medium to large size