2018-19 Resit Paper

1)

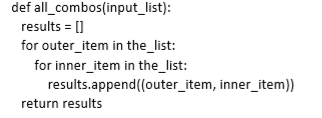
a. Describe the main features of von Neumann architecture and explain how this architectural approach is useful in the analysis of data structures and algorithms. (7 Marks)

The main features of Von Neumann architecture are

* A central processing unit (cpu)
* Arithmetic and logic unit (alu)
* Processor registers
* Control unit (containing instruction register and program counter)
* Memory (storing data and instructions)
* Input/output (IO) mechanisms.

All these features are interconnected through several busses.

This approach is useful in the analysis of data structures and algorithms as von Neumann architecture allows for many things to be tracked such as number of clock cycles the cpu takes to complete an action or how much memory is used when carrying out an instruction.

b. Using “Big O” Notation describe the complexity of the following code:

In your answer you should explain any assumptions you’ve made in your calculations

(10 marks)

The following code would be run in Linear time according to Big O notation. Linear notation means that the time taken to run the algorithm is directly proportional to the input size. So, in this case the bigger the input\_list the longer the code will take to run. In Big O notation the complexity would be represented as O(n).

c. Whilst solving a programming problem you must choose between two algorithms, one that runs in linear time O(n) and another that runs in quadratic time O(n^2). Which would you choose? In your answer you should justify your context and explain the meaning of the terms linear and quadratic in this context, and identify an example algorithm for each.

(8 marks)

If a program runs in linear time (O(N)) that means that the algorithms run time will be directly proportional to the input size given. If a program runs in Quadratic time (O(N^2)) then the algorithms execution time is proportional to the square of the input size. I would choose the algorithm that runs in linear time (O(N)) as the run time would be shorter than that of a quadratic algorithm as the input size squared would obviously be higher than just the input size so the linear algorithm would be faster. One example of an algorithm that works in linear time is the linear search algorithm. An example of an algorithm that works in quadratic time is the Bubble sort algorithm.

2)

a. Describe how stack data structures can be used to implement undo and redo features. In your answer you should identify the operations associated with stacks, the effect that these operations have on the data stored therein, and how these considerations help you to solve the undo/redo problem. (9 marks)

To implement an undo and redo feature using a stack you would start by pushing the value onto the stack for each turn played. You would do this by using the push stack function to push the value onto the stack. As a stack is LIFO (last in first out) this means that the last value pushed onto the stack would be the first to popped. Pop is another function used in a stack to remove the value from the stack. So if the pop function is ran on a stack the item that will be popped will be the most recent item added to the stack. However in the case of an undo and redo system we may want to undo and redo multiple times meaning that if we popped the value to undo then we would lose that value on the stack. To prevent this the stack has another feature known as Peek(). Peek allows you to view the values on the stack without the need to pop() the value off the stack. So when undoing and redoing the peek() function could be used to look back through the stack to undo and redo. To know which value you want to undo or redo to you would have to have a variable to keep track of the amount of turns which have passed so you can correctly undo or redo the right values.

b. Compare and contrast the binary search algorithm to another search algorithm (such as linear search) In your answer you should identify the advantages and disadvantages of each algorithm and the circumstances under you might choose one over the other. (8 marks)

The biggest difference between the linear search algorithm and the binary search algorithm is that the linear search algorithm does not require the data to be sorted whereas the binary search algorithm requires the data to be sorted before it is used. Binary search is an example of a divide and conquer algorithm, linear search is not. When using linear search there is a trade-off between the setup/sort time that other algorithms such as binary search require versus the cost of searching with linear search as it can be quite a lot slower than other search algorithms, but it doesn’t require the initial setup. If you have a larger amount of data, then you may opt to use binary search as linear search may take a very long time as there is a lot of unsorted data to get through. However, if the data set is not very big then linear search may be faster as you do not need to take the time to sort the data beforehand. Linear search runs in linear time whereas binary search runs in logarithmic time. So, if the data set is large then you may want to use binary search as the time increase to data increase is less proportional than in linear search where it is directly proportional.

c. Sort the following array of numbers into ascending order using the insertion sort algorithm : [8,2,7,3,4] For each iteration of the algorithm write out the state of the partially sorted array and identify the element under consideration. (8 marks)

Iteration 1 – element under consideration 2, unsorted list [8,7,3,4], sorted list [2]

Iteration 2 – element under consideration 3, unsorted list [8,7,4], sorted list [2,3]

Iteration 3 – element under consideration 4, unsorted list [8,7], sorted list [2,3,4]

Iteration 4– element under consideration 7, unsorted list [8], sorted list [2,3,4,7]

Iteration 5 – element under consideration 8, unsorted list [] sorted list [2,3,4,7,8]

3)