OCR Computer Science A Level

2.3.1 Analysis, Design and Comparison of Algorithms

Concise Notes

Analysis of Algorithms

- There are two things you check when developing an algorithm:
  - Time Complexity
  - Space Complexity

Time of Complexity

- How much time an algorithm requires to solve a particular problem
- The time complexity is measured using a notation called big-o notation, it shows the effectiveness of the algorithm
- It shows the amount of time taken relative to the number of data elements given as an input
- This is good because it allows you to predict the amount of time it takes for an algorithm to finish given the number of data elements

Big O Notation	Name	What is means
$O(1)$	Constant time complexity	The amount of time taken to complete an algorithm is independent from the number of elements inputted.
$O(n)$	Linear time complexity	The amount of time taken to complete an algorithm is directly proportional to the number of elements inputted.
$O(n^2)$	Polynomial time complexity (example)	The amount of time taken to complete an algorithm is directly proportional to the square of the elements inputted.
$O(n^n)$	Polynomial time complexity	The amount of time taken to complete an algorithm is directly proportional to the elements inputted to the power of n
$O(2^n)$	Exponential time complexity	The amount of time taken to complete an algorithm will double with every additional item.
$O(\log n)$	Logarithmic time complexity	The time taken to complete an algorithm will increase at a smaller rate as the number of elements inputted.

Logarithms

- A logarithm is similar to the inverse of an exponential
- the logarithms is an operation that determines how many times a certain number (base) is multiplied by itself to reach another number
- It might help to check the extra resources for more information on this

x	y = log(x)
1 (2^0)	0
8 (2^3)	3
1024 (2^{10})	10

Space Complexity

- The space complexity of an algorithm is the amount of storage the algorithm takes
- Space complexity is commonly expressed using Big O ($O(n)$) notation.
- Algorithms store extra data whenever they make a copy, this isn't ideal
- When working with lots of data, it's not a good idea to make copies. As this will take lots of storage which is expensive.

Designing Algorithms

- An algorithm is a series of steps that completes a task
- When you design an algorithm your main objective is to complete a task, the next objectives are to get the best time complexity and the best space complexity
- When you try to minimise the time and space complexity you might get conflicted thinking about which one of the two complexities are more important. It is entirely dependant on the situation
- To reduce the space complexity, you make sure perform all of the changes on the original pieces of data.
- To reduce the time complexity, try to reduce the number of embedded for loops as possible
- Try to reduce the number of items you have to complete the operations on, for example the divide a conquer algorithm accomplishes this and results in a logarithmic algorithm.

Comparison of Algorithms

Linear Search Algorithm

- An algorithm which traverses through every item one at a time until it finds the item its searching for
- The Big-O notation for a linear search algorithm is $O(n)$

Binary Search Algorithm

- A divide and conquer algorithm, this means it splits the list into smaller lists until it finds the item it's searching for.
- Since the size of the list is halved every time it's a Big-O notation of $O(\log(n))$

Bubble Sort Algorithm

- Passes through the list evaluating pairs of items and ensuring the larger value is above the smaller value
- It has a polynomial Big-O notation, $O(n^2)$