|  |  |
| --- | --- |
| **Algorithms**  An algorithm is a set of steps or instructions for completing a task.  An algorithm must have a clear problem statement – must define input and output expected  **What makes a good algorithm?**  An algorithm is correct when on all runs the algorithm takes an input and produces an expected output. – Additionally for any input the algorithm should also terminate.  Efficiency has two measures – time and space.   * Efficiency on time or Time complexity is a measure of how long the algorithm takes to run. * Efficiency on space deals with the amount of memory used in a computer.   Efficiency balances time and space.  **Running time** is used to define time complexity.  Three ways to measure Algorithms   * Best case scenario * Average Case Scenario * Worst Case Scenario   **Order of Growth** – how an algorithm grows as the input set increases.  **Big 0 –** is a theoretical definition of the complexity of an algorithm as a function of the size. It is used to define complexity. E.g O(n)  O – Order of magnitude of complexity  (n) – A function of the size – measures as input size grows.  O(n) – measures worst case scenario  **Polynomial Runtimes (O(n^k)) – efficient Algos**  Linear search = O(n) – linear time  Binary Search = O(log n) – logarithmic time – how many times youll divide the problem to get the answer  Constant time = O(1) – reading a value in a list    Quadratic runtime – O(n^2) –  Cubic runtime – O(n^3)  Quasilinear Runtimes = O(n log n) – for every value of n we run log n operations  **Exponential Runtimes (O(X^n)) – Inefficient Algos**  Brute force – O(x^n  Combinatorial Runtime O(n!)  **DATA STRUCTURES**  It is a way of storing data and the relationship between the data and the operations.  **Array**  Is a data structure that stores a collection of values referenced using keys.  Heterogeneous arrays in python  Java and swift have homogenous – one data type |  |