**Collections report by shimaa hamdy (PD)**

|  |  |  |  |
| --- | --- | --- | --- |
| **collection** | **desc** | **performance** | **use** |
| Dictionary | Store data in the form of  (Key, Value)  Doesn’t allow double keys  Can access elements by key --> map[key]  ------------------------------------  -calculated a key hash value  -insert Value to a bucket related to the hash value  -check for equality to get the right data  Capacity is not the actual number of elements | With good  (GetHashCode)  Implementation  We can have:  O(1) for insert  O(1) for access  O(1) for delete  With bad (GetHashCode) it will depend on the depth of the bucket | -when have a unique key like (ID)  -when data is not edited a lot  -when we search for data with key many times and that key depend on some states |
| LinkedList | doubly linked contains (node points) each node refers to the next node and backward to the previous node.  -store data in sequence  -allow duplicate elements  -does not allow search or access elements directly  Capacity is the actual number of elements | O(1) for insert first  O(1) insert last  O(1) for delete first  O(1) delete last  O(n) for access element as we have to loop through data  O(n) for delete or insert in specific space in the middle | -accessing sequential data from first or final |
| List | Store data that can be accessed by an index.  It is a dynamically sized array  Capacity is also not the actual number of elements as it grows dynamically  -allow duplicate elements  -can access elements directly by index  Ex--> list[index] | O(1) for access or edit element by index  O(1) for delete or add an element at last  O(n) for insert or remove from middle or first elements | -in most data we use lists as it provides both fast accessing and dynamic sizes and the cost of resizing is negligible in most times  -in random data accessing |
| HashSet | Represent a set of unique elements  Does not allow duplicate values  Store data in form of (value)only  -order of insertion of items is not relevant.  -we cannot access the element directly  -capacity of HashSet is the number of objects it can hold  -used for some mathematical calculations that are done with sets | O(1) for search for element  O(1) for insert | -we use for store data that do not have duplicate values most of the time to do some Mathematical operations on data as a whole like check, union, sub |
| Stack | Last-In-First-Out (LIFO)  Last item enter is the first item we can access  -use array internally so it resizes if array is full | O(1) for push & pop  O(n) for search for specific items | temporary storage for information  if you need to access the data in reverse order. |
| Queue | First-In-First-Out (FIFO) same as Stack | O(1) for enqueue & dequeue | Same as stack but if want to access data in order it stored |
| Priority Queue | Store data in order according to its priority  Pop the element with higher priority | O(1) for pop  O(nlogn) for insert  As it uses heap sort algorithms in insertion |  |
| Sorted Dictionary | Store also the data as (key, value)  class but it is a binary search tree with O(log n) retrieval  -sort date with every insertion  must also have a unique key.  -can access elements with key | O(log n) for accessing  O(nlogn) for insert & remove | faster insertion and removal operations for unsorted data |
| Sorted List | Also store the date as (key, value)  -no duplicate key  has two arrays to store the elements one array for the keys and another array for the associated values. | O(1) for index access  O(log n) for key access  O(n) for insert & remove | When we want less memory and can access data with index |