Arrays –

1. Contiguous blocks in memory
2. M/R Address – 100 (Array – 300 bytes long): 100, 400, 700…
3. Every element in array occupies same amount of space in memory
4. Base M/R Address – x, size of each element – y, ith element address: x + i \* y
   1. Retrieve with Index – O (1)
   2. Retrieve without Index – O (n)
   3. Add element to full array – O (n)
   4. Add element to end of array (has space) – O (1)
   5. Insert or update element at specific index – O (1)
   6. Delete an element by setting it to null – O (1)
   7. Delete an element by shifting element – O (n)

Sort Algorithms –

Bubble Sort –

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 20 | 35 | -15 | 7 | 55 | 1 | -22 |

Unsorted partition Index = 6 – this is the last index of the unsorted partition

i=0 – index used to traverse the array from left to right [0 1] x no swap

i=1 – index used to traverse the array from left to right [1 2] swap -15 35

and so on…

If i=unsorted partition index then, stop

Unsorted partition Index = 5, 4, 3 …

Reset i to 0 again, and repeat the process

* In place algorithm (logically partitioning the array but not creating new array)
* O (n^2) – Quadratic time complexity
* 10 -> 100, 100 -> 10000, 1000 -> 1000000
* Algorithm degrades quickly

Unstable Sort –

Unsorted:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 | 9 | 3 | 9 | 8 | 4 |

Sorted:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 4 | 5 | 8 | 9 | 9 |

Sorting order not maintained as Violet 9 should occur after the black 9

Bubble Sort is a stable sort algorithm, because we swap when element of i is greater than i+1

Selection Sort –

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 20 | 35 | -15 | 7 | 55 | 1 | -22 |

Last Unsorted Index = 6 – this is the last index of the unsorted partition

i=1 – index used to traverse the array from left to right 35 > 20 (i=2) -15 >35 x (i=3)

largest = 0 – index of largest element in unsorted partition -> 1 change largest element always

swap (55 and -22): swap largest with last element in unsorted index

largest = 0 and I =1

Selection sort because on each traversal we are selecting a largest element and swapping.

* In place algorithm
* O (n^2) – quadratic time complexity
* Doesn’t require as much swapping as bubble sort. (Once per traversal)
* Unstable algorithm, if we have duplicate elements then their regular order might not be retained.

Insertion Sort –

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 20 | 35 | -15 | 7 | 55 | 1 | -22 |

Fist unsorted Index = 1 – this is the first index of the unsorted partition

I=0 – index used to traverse the sorted partition from right to left

New element = 35 – the value we want to insert into the sorted partition – array[firstUnsortedIndex]

Change new element always and I =1, new element: -15 < 35 (35 -> right pos2)

-15 <20 (-15 at starting and shift 20)

7<35 (35 -> right), 7<20 (shift 20 to right), found pos for 7 and so on…

* In-place algorithm
* O (n^2) – quadratic time complexity
* Stable algorithm

Shell Sort –