Sort:

Selection Sort, Insertion Sort, Shell Sort, Bubble Sort, Heap Sort, Merge Sort, Quick Sort, Radix Sort, and Counting Sort.

• Bonus exploration: Binary Insertion Sort, Shaker Sort, Flash Sort

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Selection Sort | Insertion Sort | Shell Sort | Bubble Sort |
| IDEA | Find the minimum/ maximum and put it in correct position (swap minimum with | Compare element with all elements before (until the element less/greater than element). And put it in correct (movement all elements greater/less than element after 1 position and put in in correct – not swap) | Same Insertion Sort. But Insertion Sort have a bad case that when the less/greater then element so far away with element. So Shell Sort, u will check subarray with gap (until gap = 1) -> reduce changes have bad case | Compare element with all elements after it and swap it if it wrong position (same insertion sort but u don’t stop when u meet the element greater/less than element) |
| Ưu điểm | U can take top n when u don’t finish sorted full array  Easy to implement | Easy to implement | Easy to implement  The an improved version of insertion sort | Easy to implement |
| Nhược điểm | U need to find minimum/ maximum many time. So previous comparisons are not used in the best way.  U need to traverse all element (except elements that correct position) even the element u want to find at in the first position. | U need to find the correct its position and move it in this (so u need to move elements the wrong after its position -> u need to many swap)  Different selection. U can stop loop when u meet the element greater/less than it in sorted subarray. But u also need to go through all elements. And if your array have minimun at the end of array u need to n step and n swap so that u can put it in correct position | When u sort array with gap. U can solve somecase that have minimum/maximum at the end of array. But it is not optimal. And it can a bad version of the insertion sort. If u have a sorted array – insertion sort : o(n). But shell sort: o(n^2 / k) with k is the Number of Interval Divisions (gap) of array. | U need to compare the element with all elements after it and swap if it wrong position. So u need to many swap in your code -> it is not good. If the elements u compare is the ith mininum, u still have to check it with others -> it is so bad. |
| Độ phức tạp | *N(n+1)/2 = ½ n^2;*  O(n^2) | Best case (sorted array): O(N)  Worse case (reverse – sorted array)  O(N^2)  Average case : O(N^2) | O (n^2 / k) | O(N^2) |
|  | In-place, stable (base on your implement) | In – place, stable (base on your implement) | In-place, stable (base on your implement) | In – place, stable (base on your implement) |

Vd: Selection Sort

7 5 8 3 1

Step 1: 1 5 8 3 7

Step 2: 1 3 8 5 7

Step 3: 1 3 5 8 7

Step 4: 1 3 5 7 8

Vd: Insertion Sort:

7 5 8 3 1

Step 1: 5 7 8 3 1

Step 2: 5 7 8 3 1

Step 3: 5 7 8 3 1

Step 4: 3 5 7 8 1

Step 5: 1 3 5 7 8

Vd: Shell Sort  
7 5 8 3 1

Step 1 (Gap = 5/2 = 2) => 7 5 8 3 1 -> 7 3 8 5 1 -> 7 1 8 3 5

Step 2 (Gap = 2/2 = 1) => 1 7 8 3 5 -> 1 7 8 3 5 -> 1 7 8 3 5 -> 1 3 7 8 5 -> 1 3 5 7 8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Heap Sort | Merge Sort | Quick Sort | Binary Insertion Sort |
| IDEA | Build an array have rule is the element always greater/less than element, element. And than u will find the minimum/maximum (same Selection sort, but u don’t need to visit other element – u only check the parent element and put it in the first position and swap it with element at the position where its position corrects) | U need to divide array into two subarray and u will sort them. Add u will merge two sorted subarray int a sorted array | U need to choice pivot. And u will move elements < pi into left and >= pi into right . and u will repeat with subarrays => u will have an sorted array | With insertiong sort, u will need to move elements into its position. But u can meet the case that the its position is so far -> u have shell sort to improve that case. But it not good -> u can find its position by means of use binarysearch in sorted subarray. |
| Ưu điểm | Minimize the number of comparisons because you only need to compare to the parent element | U will divide and conquer. So u will reduce the length of a array that u solve -> Minimize the number of comparisons. And then u only merge them. Each element don’t need to compare with all element | U will divide and conquer. But it is not same merge sort. U will use pivot so don’t need to merge two sorted subarray. U only need to put them together. | it can more efficient than shell sort. |
| Nhược điểm |  |  |  |  |
| Độ phức tạp |  |  |  |  |
|  | In-place |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Counting Sort | Radix Sort | Shaker Sort |
| IDEA | U need to have an array have max elements (max is the max value of the array that u want sorted)  U will count elements that have val = i and put it in a[i]. And then u will put it in the main array with i from 0 to max  (u will meet to difficult with max is so highest) | U will take advantage of counting sort. But u will counting sort with a digit of the element of array (so u can avoid the difficult with max is so highest – u only need to 10 bucket ; but u need to repeat the action of your many time with all digit of the max value) | When u use bubble sort, u need to swap continuously with 2 element if they wrong position and u will repeat it with each element many time. U can improve this case by means of neglect the elements that correct its position.  U will move the minimum / maximum into correct position. And then u will sort the subarray not include the correct elements. |
| Ưu điểm | U will sort quickly  Non-comparsion | U will sort quickly  Non – comparsion |  |
| Nhược điểm | U cann’t u it with highest data  Need to big space memory | Need to big space memory |  |
| Độ phức tạp |  |  |  |
|  |  |  |  |