1. Name 5 sorting algorithms, also write their time complexities(best, average, worst).

|  |  |
| --- | --- |
| Algorithm | Time Complexity |
| Selection Sort | Best Case : Ω(n^2)  Worst case : O(n^2) |
| Merge Sort | Best Case : Ω(nlogn)  Worst Case : O(nlogn) |
| Bubble Sort | Best Case : Ω(n)  Worst Case : O(n^2) |
| Insertion Sort | Best Case : Ω(n)  Worst Case : O(n^2) |
| Quick Sort | Best Case : Ω(nlogn)  Worst Case : O(n^2) |

2) Implement a selection sort algorithm using Python.

1. A=[int(i) for i in input().split()]

for i in range(len(A)):

min\_idx = i

for j in range(i+1, len(A)):

if A[min\_idx] > A[j]:

min\_idx = j

A[i], A[min\_idx] = A[min\_idx], A[i]

for i in range(len(A)):

print("%d" %A[i])

3) Implement pop operation of the stack

1. **Pop():** It removes the element on top of the stack. It also takes O(1).*O*(1) time as the top contains the index of the most recently added element.

def pop(stack):

if (isEmpty(stack)):

return str(-maxsize -1)

return stack.pop()

4) Implement dequeue operation of the queue

1. def DeQueue(self):

if self.isEmpty():

print("Empty")

return

print("% s dequeued from queue" % str(self.Q[self.front]))

self.front = (self.front + 1) % (self.capacity)

self.size = self.size -1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |