Aadit Kapoor

013413530

Problem 4:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| E | A | S | Y | Q | U | E | S | T | I | O | N |
| A | E |  |  |  |  |  |  |  |  |  |  |
|  |  | S |  |  |  |  |  |  |  |  |  |
| A | E | S |  |  |  |  |  |  |  |  |  |
|  |  |  | Q | Y |  |  |  |  |  |  |  |
|  |  |  |  |  | U |  |  |  |  |  |  |
|  |  |  | Q | U | Y |  |  |  |  |  |  |
| A | E | Q | S | U | Y |  |  |  |  |  |  |
|  |  |  |  |  |  | E | S |  |  |  |  |
|  |  |  |  |  |  |  |  | T |  |  |  |
|  |  |  |  |  |  | E | S | T |  |  |  |
|  |  |  |  |  |  |  |  |  | I | O |  |
|  |  |  |  |  |  |  |  |  |  |  | N |
|  |  |  |  |  |  |  |  |  | I | N | O |
|  |  |  |  |  |  | E | I | N | O | S | T |
| A | E | E | I | N | O | Q | S | S | T | U | Y |

Problem 5: Quicksort on average takes about 2N compares when sorting an array with distinct elements. We can see the performance of quick sort depends on the pivot selection. If we have distinct elements then the pivot selection is random and we can see the best case for quick sort comes out to be O(nlogn) and the worst case comes out to be O(n^2). The pivot selection in the case of all identical elements will always be a median since elements are same so as a result no swap takes place which in turn gives a time complexity of O(nlogn). (Quick sort 3 way)

If we see the traditional way of performing the sort which partitions into two will have O(n^2) on same elements.

Problem 6:

P

RP

RPI

RPOI

REMOVE : R

POI

RPOI

REMOVE: R

POI

REMOVE:P

OI

OII

REMOVE:O

II

TII

REMOVE: T

II

YII

REMOVE:Y

II

REMOVE I

I

REMOVE I

Q

UQ

UQE

REMOVE:U

QE

REMOVE:Q

E

REMOVE:E

U

E

REMOVE:E

Order:

R

R

P

O

T

Y

I

I

U

Q

E

U

E

Problem 7:

Heap data structure is a data structure that can viewed as a binary tree. It is filled from left to right. There are two kinds of heap: Max Heap and Min Heap. In Max, each node is at most the value of its parent and in Min, each node is less than or equal to each of its children. Glancing at the heap we can see that the height if O(logn). For heap construction, we have different ways to achieve the task:

1. If we perform a swim operation (left to righ, nlogn time), we can achieve a heap ordered complete tree. If we go from right to left, we can achieve this in an efficient way as we can make sub heaps on the go. If two children of the nodes are heaps then calling sink makes the subtree a heap.