3. Compare heaps vs array vs linked list (sorted and unsorted) to implement Priority

queue in terms of performance of PQ operations. Tabulate the Performance on PQ

operations such as insertItem, removeMin, and minKey(Retrieve the minkey

without removal).

Answer: Tabular Representation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S N | Data Structure | Insert Item | Remove Min | Min Key Retrieve |
| 1 | Sorted Array | O(n) | O(1) | O(1) |
| 2 | Unsorted Array | O(1) | O(n) | O(n) |
| 3 | Sorted Linked List | O(n) | O(1) | O(1) |
| 4 | Unsorted Linked List | O(1) | O(n) | O(n) |
| 5 | Min Heap | O(n log n) | O(n log n) | O(l) |

4. Carry out the steps of RadixSort to sort the following array:

{80, 27, 72, 1, 27, 8, 64, 34, 16}.

Use 9 as your radix.

Solution: Radix = 9, Buckets = 2 [ Max digits ], initial array : 80, 27, 72, 1, 27, 8, 64, 34, 16

Step 1: array [i] % 9

Input: 80 % 9 = 8th index🡺80-72 = 8

27 % 9 = 0th index 🡺27-27 = 0

72 % 9 = 0th index 🡺 72-72= 0

1 % 9 = 1th index 🡺1

27 % 9 = 0th index 🡺27-27 = 0

8 % 9 = 8th index🡺8

64 % 9 =1th index🡺64-63 = 1

34 % 9 =7th index 🡺34-27=7

16 % 9 = 7th index 🡺16-9 =7

order now: 27, 27, 72, 27, 1, 64, 34, 16, 80, 8

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 27  27  72  27 | 1  64 |  |  |  |  |  | 34  16 | 80  8 |

**Step2 = array[i] / 9**

Input: 27 / 9 = 3rd index

27 / 9 = 3rd index

72 / 9 = 8th index

1 / 9 = 0th index

27 / 9 = 3rd index

8 / 9 = 0th index

64 / 9 = 7th index

34 / 9 = 3rd index

16/ 9 = 1st index

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1  8 | 16 |  | 27  27  27  34 |  |  |  | 64 | 72  80 |

Order now: [1, 8, 16, 27, 27, 27, 34, 64, 72, 80] 🡺 Sorted Output