Lesson-8

Priority Queue:

A priority queue is designed for applications in which elements have a *priority value,* or *key,* and each time we need to select an element from *S*, we want to take the one with highest priority.

A priority queue is a data structure that maintains a set of elements *S*, where each element *v* is in the *S* has an associated value key(*v*) that denotes the priority of element *v*; smaller keys represent higher priorities.

Example: Amazon Prime member to get the quick delivery. Prime member in an airline.

Key Value

SSN Person Details

Student ID Student Information

How to implement a Priority Queue for the List S

1. Sequence based ADT(Linked list) – Analysis – Refer Slide 7
    a. Implement as sorted list
    b. Implement as an unsorted lsit
2. Heap based Implementation(min-heap) – Analysis Slide - 14

Array bases Sorted List Performance using Sequence ADT

4	6	8	9	10	12	14	16	
0	1	2	3	4	5	6	7	8

Analysis: Insert(2) – 2 will go to the first index, you need shift up all the elements. O(n)

Remove() – front, remove 4. You have to shift down the elements. O(1). If you want to shift you need O(n).

Unsorted List Performance

14	6	8	19	10	31			
0	1	2	3	4	5	6	7	8

Analysis: Insert(31), Inserting at the end O(1)

Remove() – In PQ, always remove the min, which has the high priority. O(n). After removal will take O(n) for shifting.

Remove() – delete 6. Finding a min is O(n).

Linked List implementation of sequence ADT for PQ (Best Way to use it)

Unsorted List

Insertion – O(1) – Adding in the end

Removal – O(n) – You need find the min.

Sorted List

Insertion – O(n), you need find the right position to insert O(n)

Removal – O(1), Remove from the front.

Heap based PQ

Construction: O(n)

Upheap & downheap: log n

Both Insertion and deletion will take O(n log n)

Comparison based sorting

All In_place sort, PQ Sort we are sorting the elements comparing.

Non-comparison based sorting

Bucket sort – O(n) – Linear time (not an In-place sort)

Radix sort

Bucket Sort

Input array A: [5, 8, 1, 2, 3, 5,7] , size is n= 7

Maximum/biggest value = 8, Array Capacity to perform sorting which called as Bucket size = 9. Always default value of int array is 0. Bucket size is m.

0	1	1	1	0	1+1	0	1	1
0	1	2	3	4	5	6	7	8

i= 7, A[i] = A[i] + 1 = 0 + 1 = 1, replace 0 with 1 in the index 7

Output Array (A: [5, 8, 1, 2, 3, 5,7] got sorted and the output is 1 2 3 5 5 7 8. Result array size is n

1	2	3	5	5	7	8
0	1	2	3	4	5	6

Time Complexity is O(n + m)

If m is equal to n then complexity will be O(n) – Linear Time

Worst Case

Input size is 1000, Max value is 1000000.

N = 1000

M= 1000000 => (n^2)

Time complexity is O(n^2), do not use Bucket sort.

Another Worst Case Scenario Example

Arr = [5 2 3 7 9 10 100 11 45 20] n = 10

Bucket size m is = 100, which is equal to n^2

Handling Duplicates with Key and Value

Student Score(Key), name(value)

If you take SD421 Scores from the final, Your input is an Object type.

Key are duplicates.

95 a, 95 b

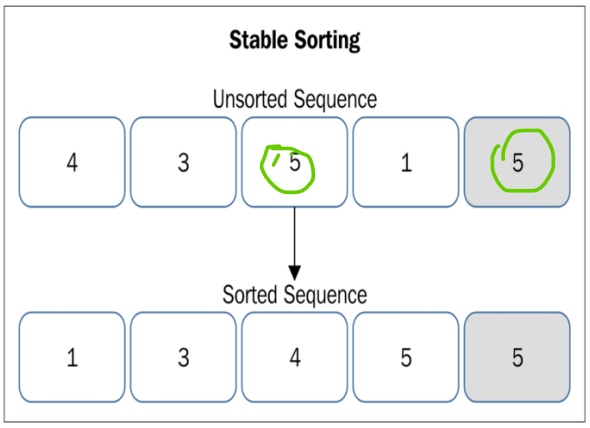
99, k

88 c, 88 e, 88 i

How we are going to handle the duplicate keys in the Bucket Sort. To overcome this we use an Linked list on each array index. Array of type Linked List.

Stable sorting

A stable sort algorithm maintains the relative ordering of elements of equal values in a sorted sequence. It can be understood using the following diagram:



Original List:

1. Alice, 25
2. Bob, 22
3. Carol, 25
4. Dave, 23
5. Eve, 22

Sort based on Age
Sorted List

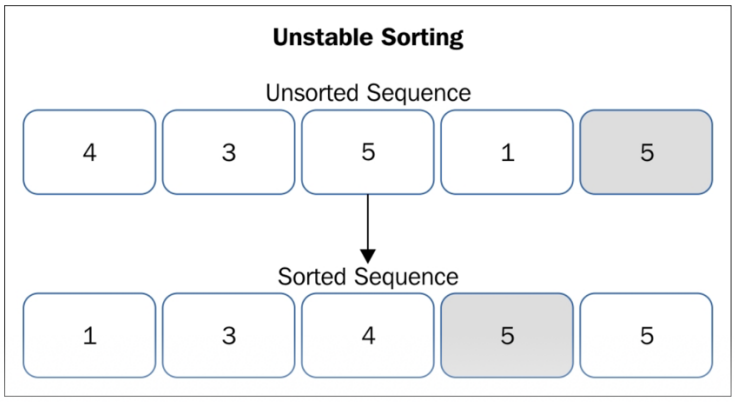
1. Bob, 22
2. Eve, 22
3. Alice, 25
4. Carol, 25
5. Dave, 23

As the diagram depicts, our unsorted list has two fives. The first **5** is in a white slot and the second one is in a gray slot. After sorting, in the sorted sequence also, the **5** in the white slot remains before the **5** in the gray slot. This is an example of a stable sort.

Bubble sort, Insertion Sort, Merge Sort, Radix Sort and Bucket Sort are Stable sorting.

Unstable sorting

Unstable sorting algorithms do not maintain the relative ordering of elements of equal values in a sorted sequence. The following diagram will help in understanding unstable sorting:



Quick Sort and Selection sort are Unstable.

Lesson-8- Review Questions

1. Purpose of Priority Queue and its ADT operations. (Slide-4)
2. Compare the performance of Sequence based Priority Queue implementation with sorted and unsorted list. (slide-7)
3. Analyze the Heap based Priority Queue performance. (slide-14)
4. How to perform a bucket sort with or without duplicates and its performance analysis.
5. How to perform Radix sort with the given Radix and its performance analysis.
6. What is stable sort. Give examples of sorting under stable sort.
7. Summary of sorting algorithms learned in this lesson. (Slide-41)