ASSIGNMENT - I

Name : Mahalakshmi. V

Reg NO : 346924009053

Department & Sec: II - BCA - 'A'

Subject : Data structure And Algorithm

Submission Date: 09 - 07 - 2025

1. PRIORITY QUEUE:

* A priority queue is an abstract data type that behaves similarly to the normal queue except that each element has some priority.

ie: the element with the highest priority would come first in a priority queue. the priority of the elements in a priority queue will determine the order in which elements are removed from the priority queue.

* the priority queue support only comparable elements. which means that the elements are either arranged in an ascending or descending order.

* for example: suppose we have some values like 1, 3, 4, 8, 14, 22 inserted in a priority queue with an ordering imposed on the values is form least to the greatest.

Therefore, the 1 number would be having the highest priority while 22 will be having the lowest priority.

Characteristics of a priority Queue:-

A priority queue is an extension of a queue that contains the following characteristics:

* Every element in a priority queue has some priority associated with it.
* An element with the higher priority will be deleted before the deletion of the lesser priority.
* If two elements in a priority queue have the same priority, they will be arranged using the FIFO principle.

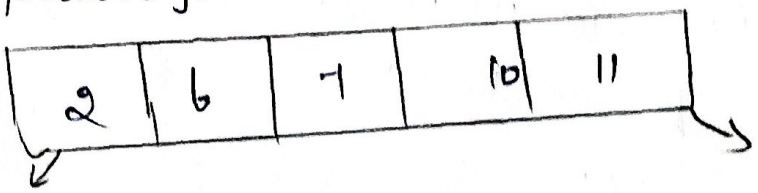
Types of priority Queue:-

1. Ascending order priority Queue
2. Descending order priority Queue.

Ascending order Priority Queue:

* In ascending order priority queue, a lower priority number is given as a higher priority in a priority.

* For example: we take the number from 1 bus arranged is an ascending order like 1,2,3,4,5; Therefore the smallest number i.e 1 is given highest priority.



element with the highest priority

elements with the lowest priority

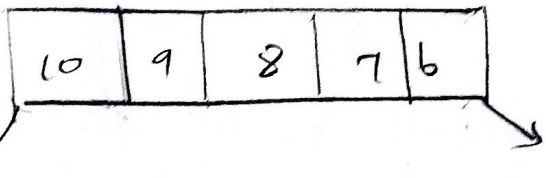
Fig: Ascending order priority queue.

Descending order priority Priority Queue:

* In descending order priority queue, a higher priority number is given as a higher priority in a priority.

* For example: we take the number from 1 bus arranged in descending order like 5.4.3.2.1:

Therefore the largest number, ie. 5 is given as the highest priority in a priority queue.



element with the highest priority

element with the lowest priority

fig: Descending order priority queue

2. Priority Queue operation:

the common operation that we can perform on a priority queue are:

-> Insertion
-> Deletion and
-> peek

Insertion the element in a priority queue:

If we insert an element in a priority queue. It will more to the empty slot by looking from top to bottom and left to right.

If the element is not in a correct place then it is compared with the parent node; if it is found out of order, elements are swapped. this process continues until the element is placed in a correct position.

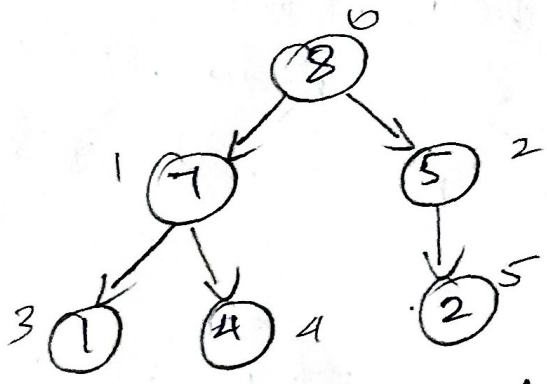
Algorithm:

Start
if (no node):
    create node
Else:
    Insert node at end of heap

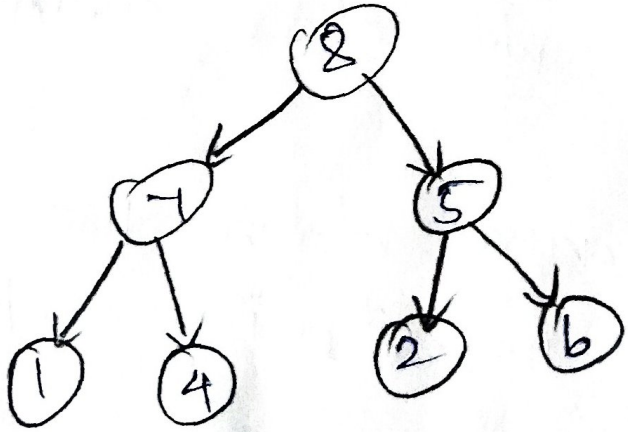
Heap ify
End

Example:

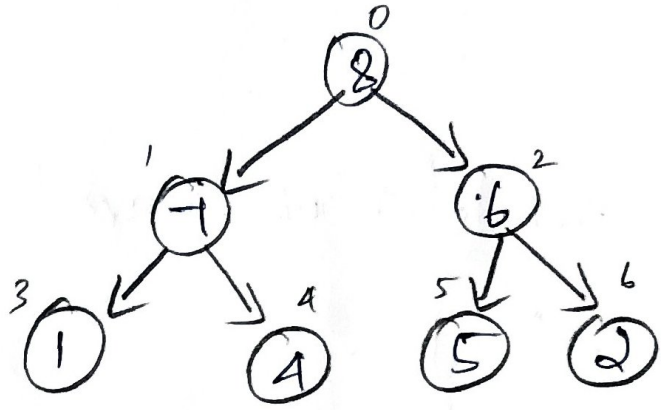
Let's say the elements are 1, 4, 2, 7, 8, 5. The max-heap of these heap of element would like



Now, Let's try to insert a new element 6, Since there are nodes present in the heap we insert this node at the end of heap so it looks like this:-



then, heapify operation is implemented after which, the heap will look like this;



Removing the minimum element from the priority queue:

* In a max heap, the maximum element is the root node. when we remove the root node, it creates an empty slot.

* the last inserted element will be added in this empty slot.

* then, this element is compared with the child nodes, i.e, left, child and right child and swap with the smaller of the two.

* It keep moving down the tree until the heap property is restored

Algorithm:

Start

If node that needs to be delete is a leaf node;

Remove the node.

Else:

Swap node that needs to be delete with the last leaf node present.

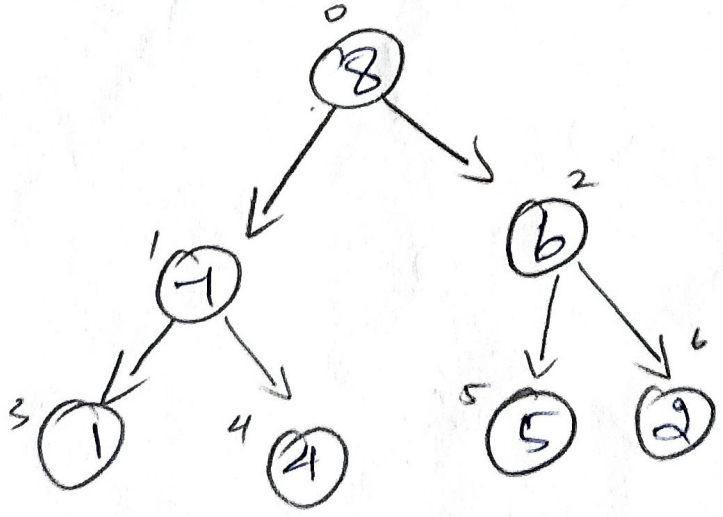
Remove the node

Heapify

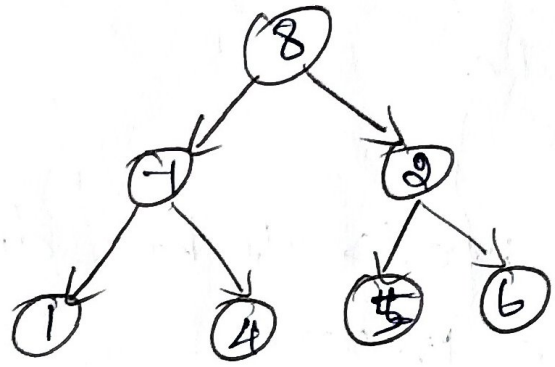
End.

Example:

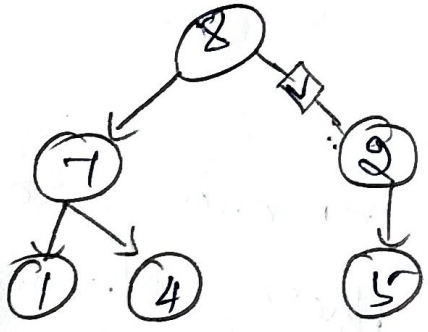
* Let's Say the elements are 1,4,2,7,8,5,6 the max heap of these elements would look like:



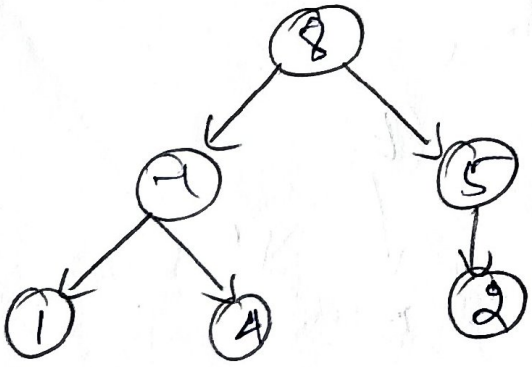
* Now let's try to delete an element since this is not a leaf node we swap it with the last leaf no so if looks like this



* then, we remove the leaf node. so it looks like this:



* then, heapify operation is implement after which, the heap will look like this.



peeking the elements from priority queue:-

* this will returns the maximum element if a max heap is used and the minimum number if a min. heap is used

* To do both of these, we returns root node, this is because in the max-heap or the min-heap the maximum or minimum element will be present at the root node respectively.

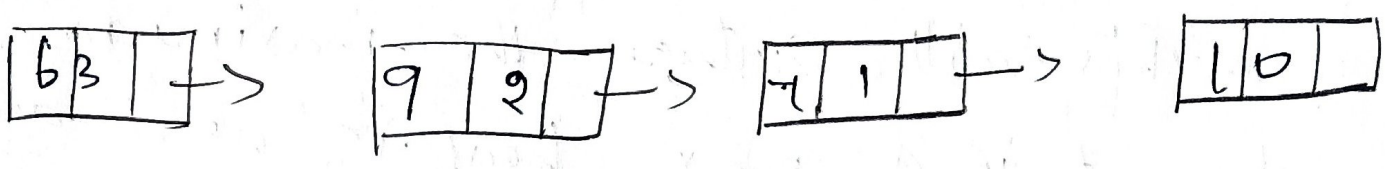
Polynominal manipulation:-

* Linked list can be used to present polynominal and the difference operation that can be performed on.

polynomial Representation:-

* Consider a polynomial $6x^3 + 9x^2 + 7x^1$ Every individual term in a polynomial consists of two parties, a coefficient a power. Here 6, 9, 7, and 1 are coefficient of the terms that have 3, 2, 1 and 0 as their power respect.

Linked list Representation of a polynomial:



* polynomial manipulation, all operation such as "addition, subtraction; and differentiation" etc... can be performed using linked list.

* Declaration for linked list implementation of polynomial ADT

```
struct poly
{
int coeff;
int power;
struct poly *next;
};
*List 1, *list 2, *list 3;
```

operation on polynomial:

* Creation
* Display
* Addition
* subtraction
* multiplication
* Evaluation.

Creation:

* Build a polynomial by String coefficient and powers

Algorithm:

-> Initialize head = NULL
-> Repeat for each for term.

a. create a new node with coefficient and exponent
b. Insert node in sorted order (deleting (deviding power)

-> Return head

Display: Print the polynomial in human readable format.

~~Algorithm:~~

Addition:

add to polynomial by combination like terms.

Algorithm:

-> Initidize result = NULL

-> Traverse poly, and poly 2

a) if exponents equal, and coefficient Store in result.

b) If exponents from poly1 > poly 2. copy term from poly.

c) Else, copy term from poly 2

-> Append remaining terms from longer polynomial

-> Return result.

Addition of two polynomial example:

$$P1(x) = 3x^2 + 5x + 6$$

$$P2(x) = 6x^2 + 3x + 4$$

Diagram:

P1: [3/2] -> [5/1] -> [6/0]

P2: [6/2] -> [3/1] -> [4/0]

Result:

[9/2] -> [13/1] -> [10/0]

Subtraction:

* Subtract two polynomial by negative one and adding.

Algorithm:

-> Negate coefficient of second polynomial
-> use addition algorithm.

Example:

$P_1[x] = 5x^2 + 4x + 2$

$P_2[x] = 2x^2 + 3x + 1$

Diagram:

P_1 : [5/2] -> [4/1] -> [2/0]

P_2 : [2/2] -> [3/1] -> [1/0]

Result :

[3/2] -> [1/1] -> [1/0]

multiplication:

multiply two polynomial by multiply each term

Algorithm:

-> for each term in poly 1:

a) for each term in poly 2:

* Multiply coefficient
* Add exponents
* Store in temporary results list
→ Combine like terms in the result.
→ Return result.

Multiplication of two polynomial

* Each term of P_1 is multiplied with every term of P_2

example:

$P_1(x) = x + 1$

$P_2(x) = x + 2$

Diagram:

$(x)\ (x) = x^2$

$(x)\ (2) = 2x$

$(1)\ (x) = x$

$(1)\ (2) = 2$

Combine $x^2 + 3x + 2$

Step-by-Step:

[x/1] [x/1]

[1/0] [2/0]

multiply each and add. result:

$[x^2] \rightarrow [3x] \rightarrow [2]$

Result:

[1/2] -> [3/1] -> [2/0]

Evaluation:

* Evaluate the polynomial for a value of x

Algorithm:

-> Input value of x

-> Result = 0

-> Traverse polynomial

a) result += coefficient * (x ^ exponent)

-> Return result.

Example:

Substitute value of x into polynomial

$P(x) = 2x^2 + 3x + 4, \quad x = 2$

value $= 2 \cdot (x^2) + 3(2) + 4 =$

$= 8 + 6 + 4 = 18$

Diagram:

[2/2] -> [3/1] -> [4/0]

Step:

2 * (2 ^ 2) -> 8 3 * (2 ^ 1) -> 6

4 * (4 ^ 0) -> 4

Result:

Sum = 18