TASK-3

Algorithm :-

Priority_enqueue (Front, priority , vehicle.No :,Time)

```
① if front = = NULL   // constant time
        front = create new node   // constant time
        set front->priority = priority. // constant time
      copy (Front-> vehicle.No from vehicle.No) // constant time
      copy ( Time, Front->Time) // constant time
      set Front -> link = NULL // constant time
      Exit   // constant time
2. Else
3. current = create new node  // constant time
4. set current-> priority = priority  // constant time
5. ~~set~~ copy ( vehicle.No: to current->vehicle No )  } // constant
6.        copy( time to current -> TIME)              }    time
7. set current->link = NULL. and set ptr = Front, prev = NULL.
                                                  // constant time
8. Repeat step 9,10 while ptr ≠ NULL & ptr->priority <= priority
9.    prev = ptr;          } This loop excutes "n times" if the
10.   ptr = ptr->link;     } element inserted is at last of linked list
                             of size n.
11. if (ptr = NULL) /* node inserted is of least priority */
12. set prev->link = current. // constant time

13. if (ptr ≠ NULL)  and   // constant time
14.        if( ptr = Front) /* node inserted is of highest pri */
             set current->link = ptr.  }  // constant time
              Front = current           }
              Exit
```

```
15. Else
16.     set  prev→link = current   }  //constant time
17.     set  current→link = ptr;   }
18. Exit.
```

TIME COMPLEXITY:-

Total time taken = k(constant time) + n

$f(n) = K*C + n$

$\boxed{f(n) = k + n}$

$k+n \leq 2n$ $\quad \boxed{C=2} \quad \boxed{n_0 = K}$

$2n - n \geq k$ $\quad \because$ We can able to show that

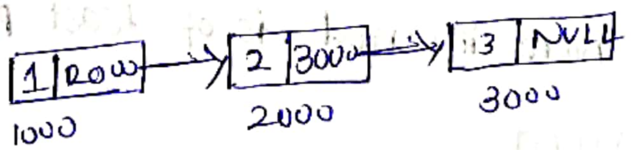
$n \geq k$ $\quad n+K \leq 2n$ for all $n > n_0$

where $n_0 = K$

Time complexity will be $\underset{\text{upper bound.}}{\underset{\downarrow}{O(n)}}$.

SAMPLE TEST CASE:-

say priority queue with given inputs is made



Front = 1000

① New node inserted is of priority 2' (i/e same priority but new one).

ptr = 1000

current→priority = 2.

current→link = NULL.

Prev = NULL

condition (ptr ≠ NULL && ptr→priority <= priority)

	ptr→priority		priority		prev	ptr
i=1	1	<	2	✓	1000	2000
i=2	2	=	2	✓	2000	3000
i=3	3	>	2	✗	loop terminates	

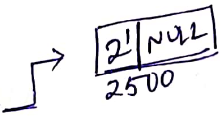
prev = 2000

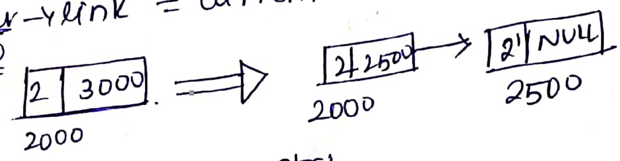
ptr = 3000

since ptr ≠ NULL it enters in Line.No 13 and

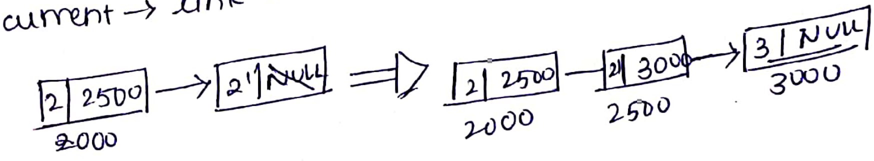
ptr ≠ Front
(3000) (1000)

so, prev→link = current

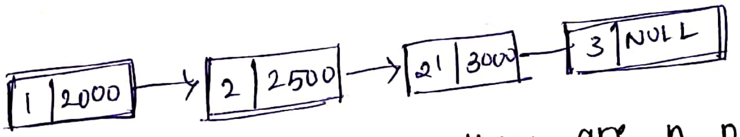




current → link = ptr;



so, new node is inserted.



say here instead of 3 nodes there are n nodes and we have insert in position n-k then we need traverse n-k nodes where Total (time) will be n-k i.e

$$f(n) = n-k$$

which is a average case

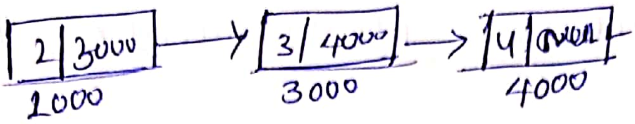
$n-k \leq 2n$ $\boxed{c=2}$

$n \geq -k$

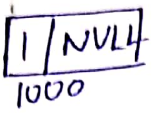
which means no is arbitary & n will always be greater than a ~~the~~ -k since n is positive

∴ Average case time complexity will be "O(n)"

② say node inserted is of highest priority



New



Front = 2000

current = 1000

ptr = 2000

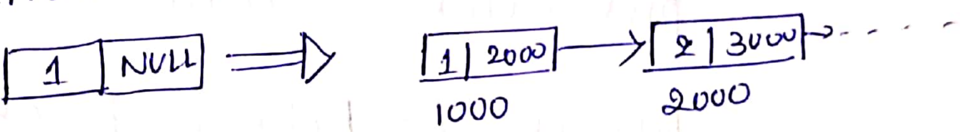
condition (ptr ≠ NUL ~~L~~ && ptr → priority <= priority)

	ptr→priority		priority	prev	ptr
					2000
i=1	2	>	1	NULL	
	(break)				

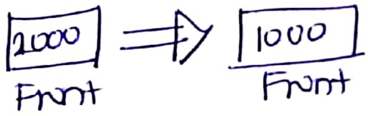
since ptr ≠ NULL it enter in to ~~se~~ second if check

since ptr = Front

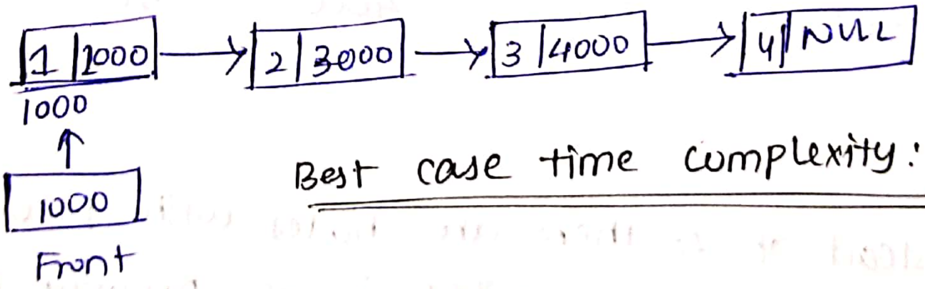
current → link = ptr



Front = curerent

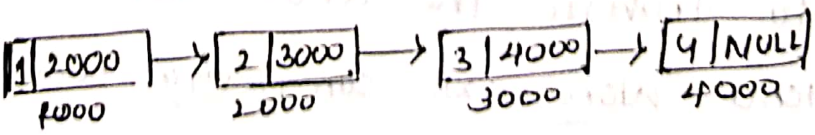


Result

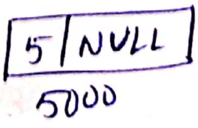


Best case time complexity : O(1)

③ say inserted node is of least priority



I New node priority = 5



ptr = 1000   current = 5000   priority = 5   prev = NULL

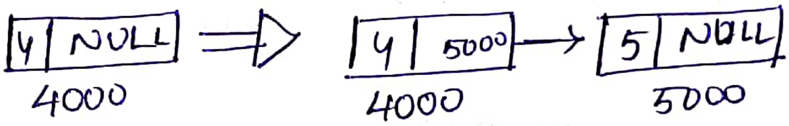
condition(ptr ≠ NULL && ptr→priority <= priority)

	ptr→ priority		priority	prev	ptr
i=1	1	<	5 ✓	1000	2000
i=2	2	<	5 ✓	2000	3000
i=3	3	<	5 ✓	3000	4000
i=4	4	<	5 ✓	4000	NULL

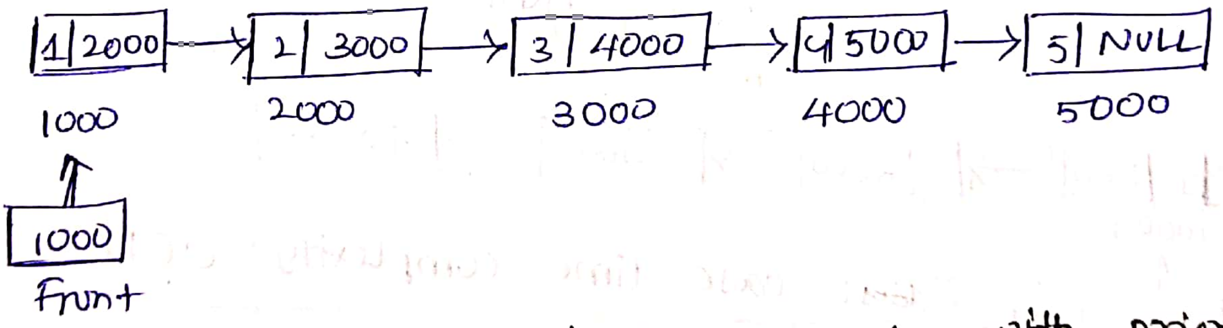
i=5   since ptr = NULL loop break.

since ptr = NULL 1st if condition will be executed

prev→ link = current



List



Say here instead of 4 there are nodes with priorities 1 to n-1 and the node inserted is of priority n then we need to traverse the whole list.

Hence "Worst case complexity would be O(n)"

Space complexity :-

space complexity refers to the total amount of memory space used by an algorithm which includes the space of input values for execution

In the algorithm the amount of memory used is constant and does not depends on the data that is processing space complexity "O(1)".

Algorithm:-

Dequeue (Front)

```
① If(Front = NULL) then   // constant time
        print: Overflow, and Exit. // constant. time

② PTR = Front   // constant time

③ Front = Front→Link  // constant time

④ Free ptr  // constant time.

⑤ SET ptr = NULL  // constant time

⑥ Exit.   // constant time
```

TIME COMPLEXITY:-

Total time taken $f(n) = K$ (constant time)

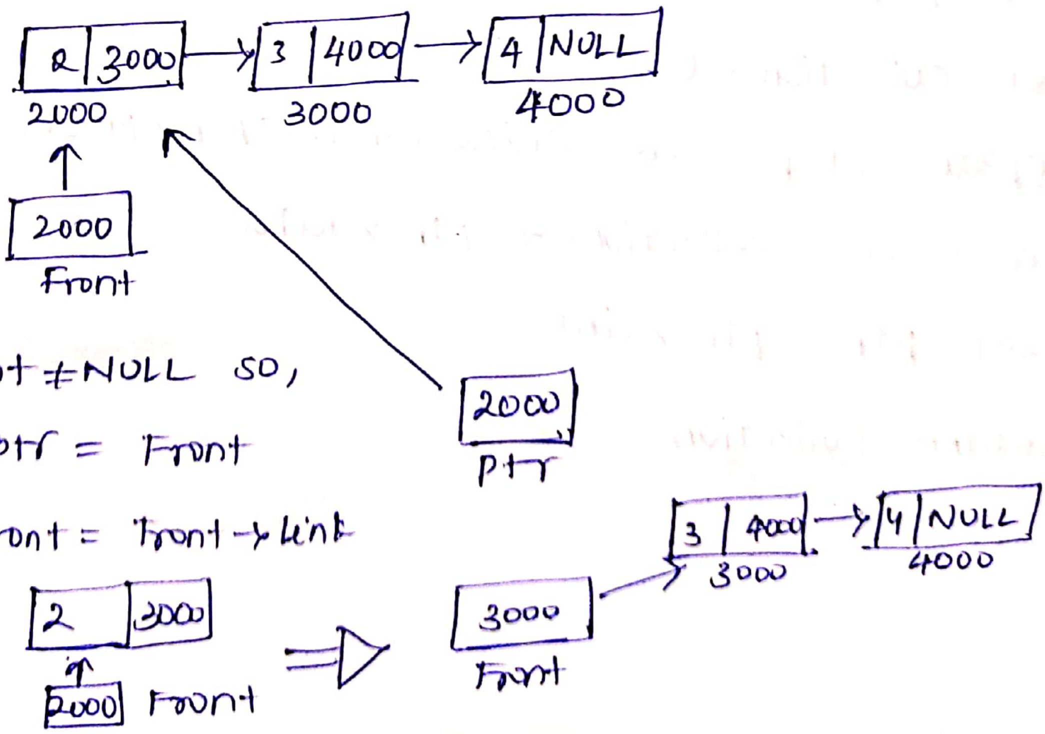
$f(n) = K(1)$

$f(n) = K$

$K \leq K$ for all $n > 1$

$\therefore K = O(1)$ with $c = 1$ and $n_0 = 1$

SAMPLE TEST CASE:-

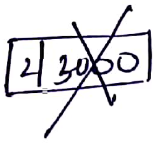


① Front ≠ NULL so,

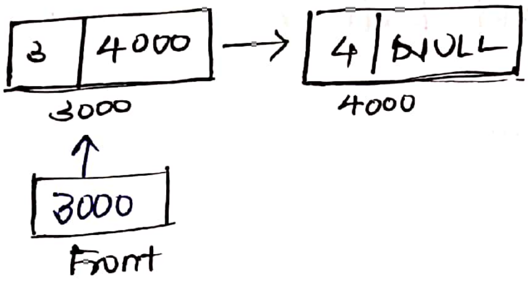
② ptr = Front

③ Front = front→link

free ptr



New list



Best case Time complexity :- O(1)

Average case Time complexity : O(1)

Worst case Time complexity: O(1)

space complexity : O(1)

Algorithm:-

```
show_collection( Front)
① if (Front == NULL)   // constant time
        print "NO COLLECTION RECEIVED", exit // constant time

② set ptr = Front.   // constant time
③ set collection = 0   // constant time
④ Repeat step 5,6 ~~while~~ while (ptr ≠ NULL)
⑤    collection = collection + ptr→data
⑥    set ptr = ptr→link
                  } This loop runs n time if size of Link List in n
⑦ return collection   // constant time
⑧ Exit.   // constant time
```

TIME COMPLEXITY:-

Total time taken be function $f(n)$

where $f(n) = K$ (constant time) $+ n$

$f(n) = n + K$

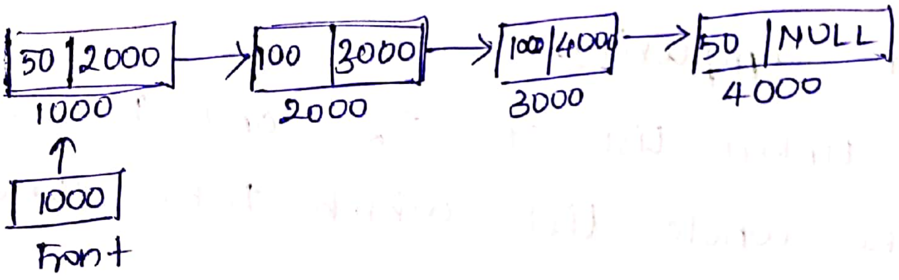
$n + K \leq 2n$ $\boxed{c = 2}$ $\forall$ ~~$n_0 \geq K$~~ $n \geq n_0$

$n \geq K$ where $n_0 = K$

∴ Time complexity of this function is "$O(n)$"

SAMPLE TEST CASE:

say linked list is as follows



Front ≠ NULL $\boxed{1000}$ ptr collection = 0;

so ptr = 1000

condition (ptr ≠ NULL)

	collection	ptr→data	ptr
i=1	0 + 50 = 50	50	2000
i=2	50 + 100 = 150	100	3000
i=3	150 + 100 = 250	100	4000
i=4	250 + 50 = 300	50	NULL

collection = 300/-

Best case complexity:-

say if my Linked is Containing on 1 node the I need to traverse it only 1 time

which is of constant time

Time complexity would be $O(1)$

Best case Time complexity "$O(1)$"

Average case complexity :-

say if my Linked is of size $n-k$ and also then it would be a $f(n)=n-k$

again $O(n)$.

Worst case time complexity:-

say if my linked list of size and I need to traverse the whole list which take time

$f(n)=n$

$n \le 2n$ $\boxed{c=2}$ $\forall n > n_0$

$n \ge 0$ where $\boxed{n_0=0}$

Hence Time complexity in worst case is "$O(n)$"

space complexity:-

All the variables used would a space of $O(1)$

$K(O(1))$ is also $O(1)$

Hence space complexity is "$O(1)$".