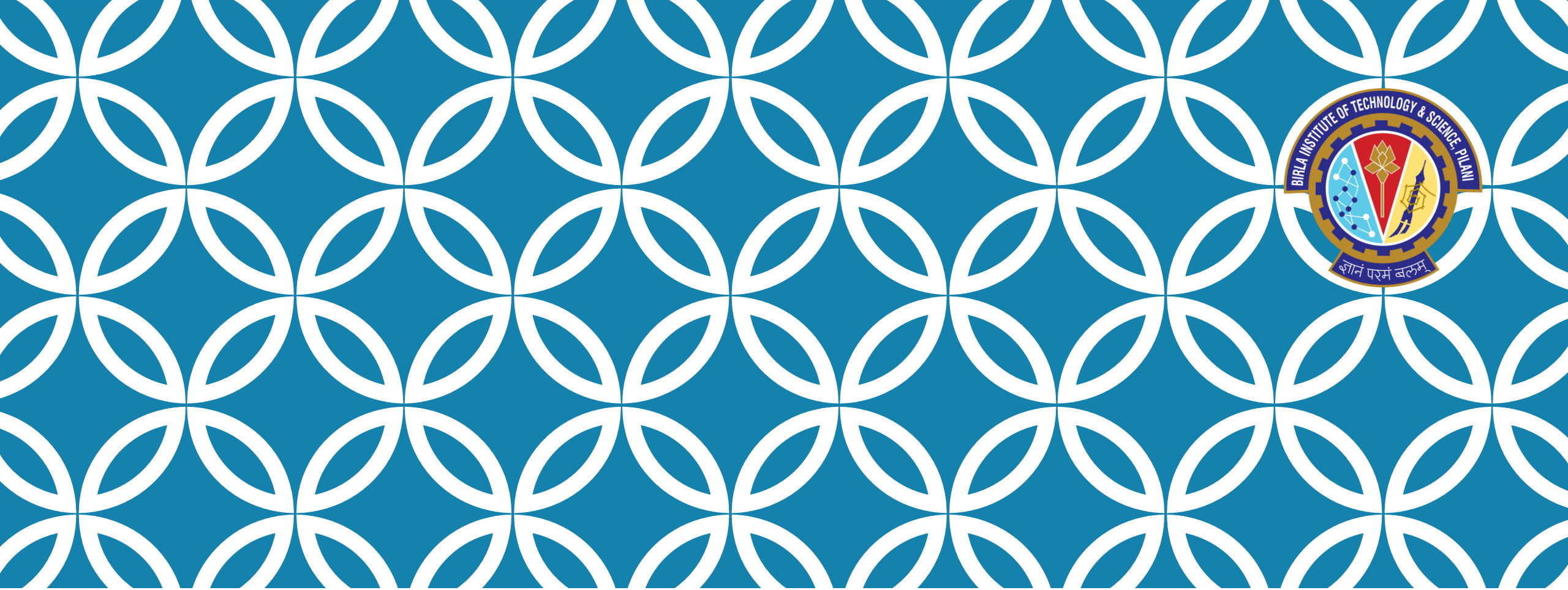


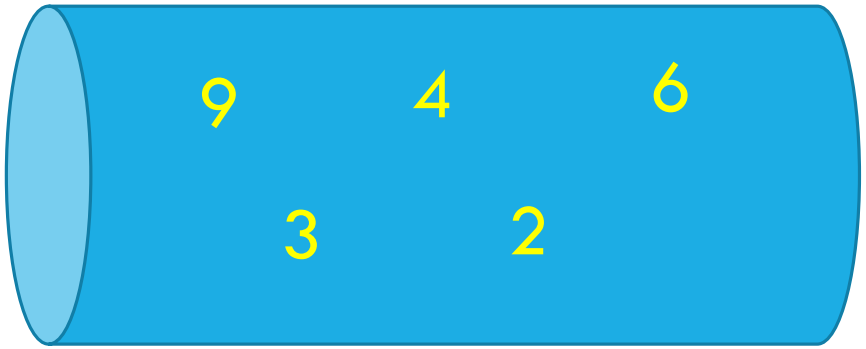
BITS F232: FOUNDATIONS OF DATA STRUCTURES & ALGORITHMS (1ST SEMESTER 2023-24) PRIORITY QUEUE ADT

Chittaranjan Hota, PhD
Sr. Professor of Computer Sc.
BITS-Pilani Hyderabad Campus
hota[AT]hyderabad.bits-pilani.ac.in

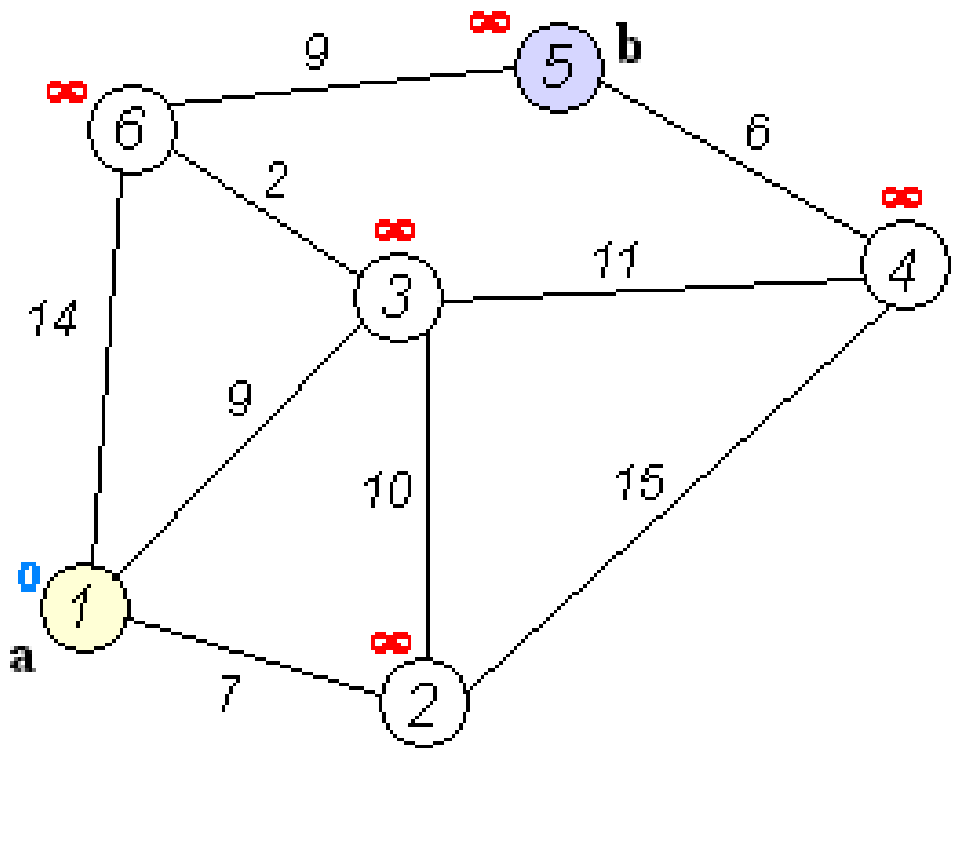
PRIORITY QUEUE ADT

- A priority queue stores a collection of entries.
- Typically, an entry is a pair (key, value), where the key indicates the priority.
- Main methods of the Priority Queue ADT
  - insert(e): inserts an entry e.
  - removeMin(): removes the entry with smallest key.
- Additional methods
  - min(): returns, but does not remove, an entry with smallest key.
  - size(), empty()

APPLICATIONS OF PRIORITY QUEUES



(A priority queue)



PQ-SORT(S, C) ALGORITHM

Input: S, C for the elements of S
Output: S sorted in increasing order

P ← priority queue with comparator C
while ¬S.empty ()
    e ← S.front();
    S.eraseFront();
    P.insert (e);

while ¬P.empty()
    e ← P.removeMin();
    S.insertBack(e);

```
26  template <typename E, typename C>
27  void ListPriorityQueue<E,C>::insert(const E& e) {
28      typename list<E>::iterator p;
29      p = L.begin();
30      while (p != L.end() && !(e <*p)) ++p;
31      L.insert(p, e);
32  }
33
34  template <typename E, typename C>
35  const E& ListPriorityQueue<E,C>::min() const
36      { return L.front(); }
37
38  template <typename E, typename C>
39  void ListPriorityQueue<E,C>::removeMin()
40      { L.pop_front(); }
```

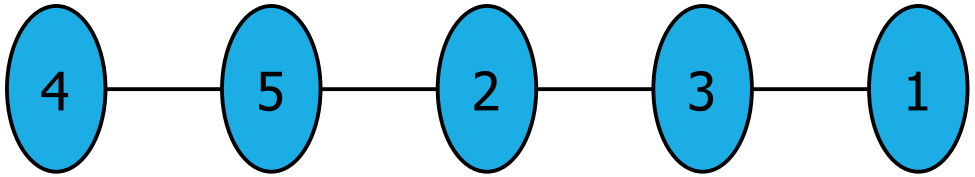
(Lab 9: Next week's lab)

CONTINUED...

```
1 : Insert
2 : Get size
3 : Check if empty
4 : Get minimum element
5 : Remove minimum element
6 : Exit
1
Enter element to be inserted : 56

1
Enter element to be inserted : 34

1
Enter element to be inserted : 89

1
Enter element to be inserted : 10

2
Size is : 4
3
The list is not empty
4
Minimum element : 10
5
Removing minimum element
4
Minimum element : 34
```

SEQUENCE-BASED PRIORITY QUEUE

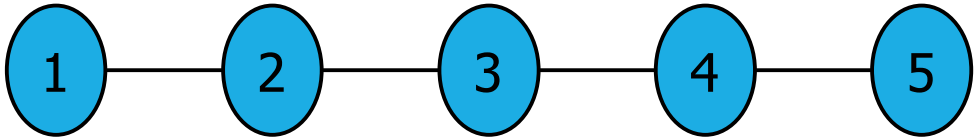
Implementation with an unsorted list:



Performance:

- insert takes ***how much time?***
- removeMin and min take ***how much time?***

Implementation with a sorted list:



Performance:

- insert takes ***how much time?***
- removeMin and min *take how much time?*

SELECTION SORT

Selection-sort is the variation of PQ-sort where the priority queue is implemented with an unsorted sequence.

Let us see an example!

Complexity of Selection sort?

INSERTION SORT EXAMPLE

Insertion-sort is the variation of PQ-sort where the priority queue is implemented with a sorted sequence:

Insertion-sort runs in $O(n^2)$ time.

- Can we do better by balancing the running times of both the phases? **Heaps**

STL PRIORITY QUEUE CLASS

```cpp
#include <bits/stdc++.h>

using namespace std;

int main()
{
    int value;
    priority_queue<int,vector<int>,greater<int> >pq;
    pq.push(1);
    pq.push(2);
    pq.push(3);

    while(!pq.empty())
    {
        value = pq.top();
        pq.pop();
        cout<<value<< " ";
    }
    return 0;
}
```

```
1 2 3
```

```cpp
#include<iostream>
#include <queue>
using namespace std;
void display_priority_queue(priority_queue<int> pq);
int main() {
  priority_queue<int> numbers;

  numbers.push(25);
  numbers.push(50);
  numbers.push(10);
  cout << "Initial Priority Queue: ";
  display_priority_queue(numbers);

  numbers.pop();
  cout << "Final Priority Queue: ";
  display_priority_queue(numbers);

  return 0;
}

void display_priority_queue(priority_queue<int> pq) {
  while(!pq.empty()) {
    cout << pq.top() << ", ";
    pq.pop();
  }

  cout << endl;
}
```

```
Initial Priority Queue: 50, 25, 10,
Final Priority Queue: 25, 10,
```