Lec5_Scheduling and Binary Search Trees

Scheduling Problem

- Runway Reservation System
  - Airport with single runway
  - “Reservations” for future landing
  - When plane lands, it is removed from set of pending events
  - Reserve req specify “requested landing time” t
  - Add t to the set if no other landings are scheduled within k mins either way
    Assume that k can vary

  ⇒ Goal : Run this System efficiently in O(lg n) time

- How to solve
  - Sorted List
    Appending and sorting : Θ(n lg n)
    Insertion : Θ(n)
    Check : O(1)
  - Sorted array
    Find position to insert : Θ(lg n)
    Insertion : Θ(n)
  - Unsorted list/array
    Check : O(n)
  - Min-Heap
    Insertion : O(lg n)
    Check : O(n)
  - Dictionary
    Insertion : O(1)
    Check : Ω(n)

Binary Search Trees (BST)

- Properties
  Each node x in the binary tree has a key *key(x).* Nodes other than the root have a parent *p(x).* Nodes may have a left child *left(x)* and/or a right child *right(x)*. These are pointers unlike in a heap
  The invariant is : for any node x, for all nodes y in the left subtree of x, *key(y)* <= *key(x)*. For all nodes y in the right subtree of x *key(y)* >= *key(x)*
- Functions
  - Insertion : insert(val)

    - Finding a value in the BST if it exists : find(val)
    - Finding the minimum element in a BST : findmin()
- Complexity
    - All operations are O(h) → h : height of the BST
    - Function
        - Next-larger

```
if right child not NIL, return minimum(right)
else y = parent(x)

while y not NIL && x = right(y)
    x = y; y = parent(y)

return (y)
```

- Height h of the tree should be O(lg n)
  → **Balancing is needed!!**