ADVANCED DATA STRUCTURES (COP 5536)

# RISING CITY

# REPORT

November 28, 2019

Suhrudh Reddy Sannapareddy

UFID: **6485 - 1063**

Email : suhrudhr.sannapa@ufl.edu

Department of Computer & Information Science & Engineering University of Florida - Fall 2019

# Structure

- risingCity.cpp : This contains the logic for the rising city problem statement. The project begins at this place, working of which is explained in the working of the project.
- Structure.cpp & Structure.h : This contains the implementation of the heap and maintains the root node. All the operations of Red Blac Tree , heap and risingCity Problem statement are in this file.
- Nodes.h :
  This has the node structure of the Red Black node("*RedBlackElement*") and Min Heap Node("*MinHeapElement*").
- Building.h :
  Is the class which maintains the triplet information, which is inherited in "MinHeapElement".

# Working of The Project

**The rising city implementation logic:**

- Global counter, input command processing is implemented in "*risingCity.cpp*". The input is sequentially loaded into "*currentCommand*". Whenever a command gets executed the next command is loaded into "*currentCommand*".
- The day counter is incremented by a unit using while loop, at every iteration commands can be executed or a new building can be picked up either by disposing the old one or inserting the old one back in to the min heap.
- The building is chosen as per the problem statement, "**void** removeMin()" is executed whenever a building is chosen.This allows us to dispose it if it gets completed after the 5 days or before, or insert the node back into the tree.
- This process is continued until both the root is null and the input commands are empty.

```
//%%%%%%%%%%%%%%%%%% RISING CITY FUNCTIONS %%%%%%%%%%%%%%%%%%
void InsertBuilding(int building_num, int total_time);
void PrintBuilding(int building_num, ofstream& output);
void ExecuteCommand(vector <string> command, ofstream& output);
string PrintBuildingForRange(int building_num1,int building_num2,RedBlackElement* node, string result);
```

Above are the functions used for risingCity problem statement

- **void InsertBuilding(int building_num, int total_time):**
  This function creates a RedBlackElement and a HeapElement and stores pointers to each other in one another.invokes "heapInsert()" and "RBTinsert()" respectively.
- **void PrintBuilding(int building_num, ofstream& output):**
  This function searches in the red black tree to find the node with building Num, to access the Heap
- **void ExecuteCommand(vector <string> command, ofstream& output):**
  This function searches in the red black tree to find the node with building Num, to access the Heap
- **string PrintBuildingForRange(int building_num1,int building_num2,RedBlackElement* node, string result):**
  This function searches on the red black tree and

accumulates the node data if the building number is in the range.

**Red Black Tree:**

```
//************ START OF RED BLACK FUNCTIONS ***************

//Functions used for Insert
RedBlackElement* ParentSibling(RedBlackElement* node);
void BSTinsert(RedBlackElement* root1,RedBlackElement* node);
 void RBTinsert(RedBlackElement* node);
 void RecursiveCorrect(RedBlackElement* node);

//Rotation
 void LeftRotationWithColorChange(RedBlackElement* node);
 void RightRotationWithColorChange(RedBlackElement* node);
 void LeftRotation(RedBlackElement* node);
 void RightRotation(RedBlackElement* node);

//Deletion
 void RBTdelete(RedBlackElement* node);
 RedBlackElement* InorderSuccesor(RedBlackElement* node);
 void SwapNodes(RedBlackElement* node1,RedBlackElement* node2);
 void DoubleBlackCases(RedBlackElement* node);
 RedBlackElement* Sibling(RedBlackElement*node);
 void DeleteDoubleBlackNullNode(RedBlackElement* DoubleBlackNull);

 //Searching - returns null if not found
static RedBlackElement* SearchTreeByValue(int building_num,RedBlackElement* root);

//************ END OF RED BLACK FUNCTIONS ***************
```

Above are the functions used, description about some non-intuitive functions are below

- void RBTinsert(RedBlackElement* node) :
  Implements BST insert and then handles the cases using "LeftRotationWithColorChange","RightRotationWithColorChange" and "RecursiveCorrect".
- void RecursiveCorrect(RedBlackElement* node) :
  This function is used as a part of insertion where it contains all the actions that needs to be done after recoloring.
- void RBTdelete(RedBlackElement* node) :
  The implementation is divided into several cases and classifies a black node with two black child into DoubleBlack case, which is further divided into cases which transform into each other.these are handled using "DoubleBlackNodeCases".

## Min Heap:

```
//---------- START OF HEAP FUNCTIONS ---------

// returns parent position
int Parent(int position);
// returns left child position
int Left(int position);
//returns right child position
int Right(int position);

//Removes the min Element
void removeMin();

// corrects the heap from the input position downward
void heapDown(int position);

//Insert Element into the heap
void heapInsert(MinHeapElement *key);

//heapifying upward by decreasing a particular value
void heapDecreaseKey(int position,int valueDecreasedTo);

//swaps the minheap pointers in the minheap array
void swapElements(int position1,int position2);

// returns first element of the array
MinHeapElement* minElement();

// ---------- END OF HEAP FUNCTIONS ----------
```

Above are the functions used for implementation of heap.