REPORT B+ Tree implementation

NAME : **Sourabhreddy Medapati**

UFID : 17439921

email : smedapati@ufl.edu

Code Structure

```
main.cpp
includes/
     main.h
     BPTree.h
     BPTreeNode.h
     BPTreeNodeElement.h
classes/
     BPTree.cpp
     BPTreeNode.cpp
     BPTreeNodeElement.cpp
tests/
     bin/
     data/
          input1.txt
          input2.txt
          output1.txt
          output2.txt
     test1.cpp
     test2.cpp
docs/*
Makefile
```

Installation

run make in the directory to create treesearch executable . Provide treesearch the path to input file with queries and it will dump output into output_file.txt in the same directory. Supported types for (key, value) pairs in this B+ tree are (double, string)

Testing and documentation

run make test to run tests located in the tests folder. Documentation can be found in the docs folder, it has been auto generated using Doxygen.

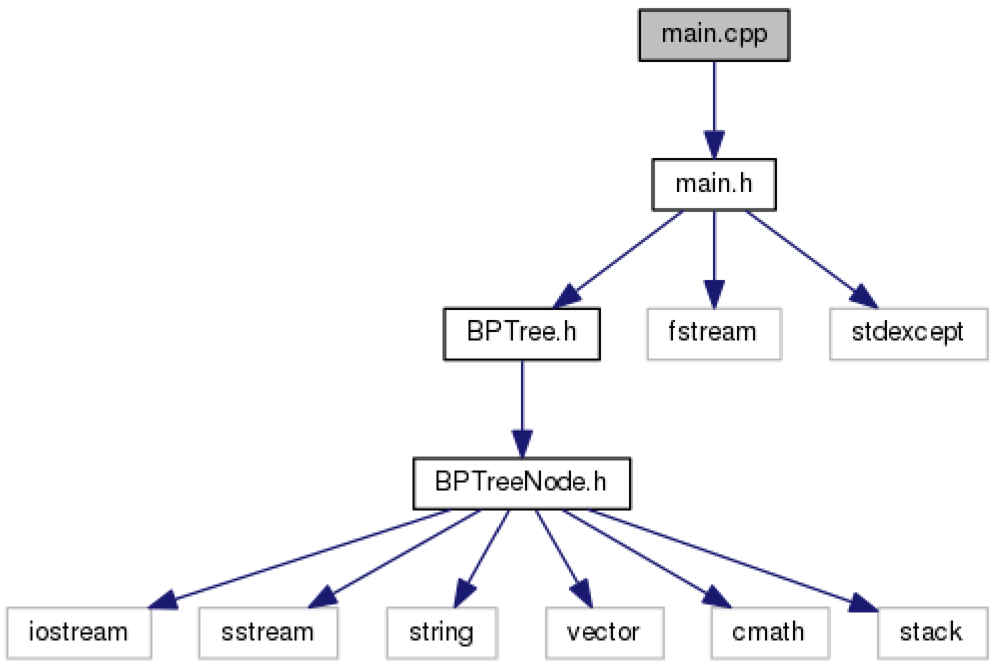
Code explanation

I have created 3 classes BPTree, BPTreeNode and BPTreeNodeElement. BPTree object has the root pointer to the B+ tree and has each node in the tree is a BPTreeNode object. Every BPTreeNode object has its own array of BPTreeNodeElements which are the key objects. Every key object has a left and right pointer pointing to lower BPTreeNodes. The source code is properly annotated and the doxygen html docs also provide a nice interface to the same. The html docs can be accessed from docs/html/index.html

Prototypes of functions and dependency graphs

Main.cpp

dependency graph

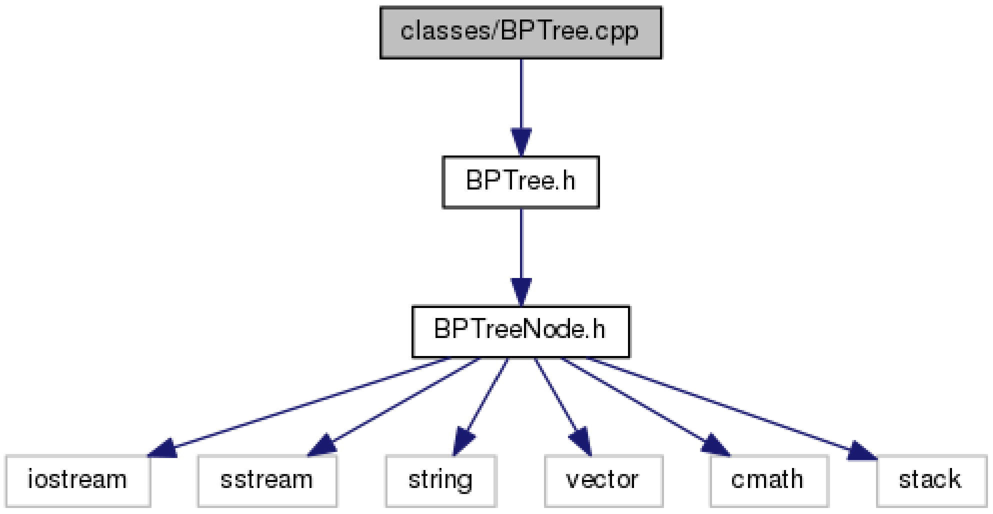


Methods:

```
void parseInput(int, char**);
void processQueries(string);
vector<string> buildQuery(string);
```

BPTree.cpp

dependency graph



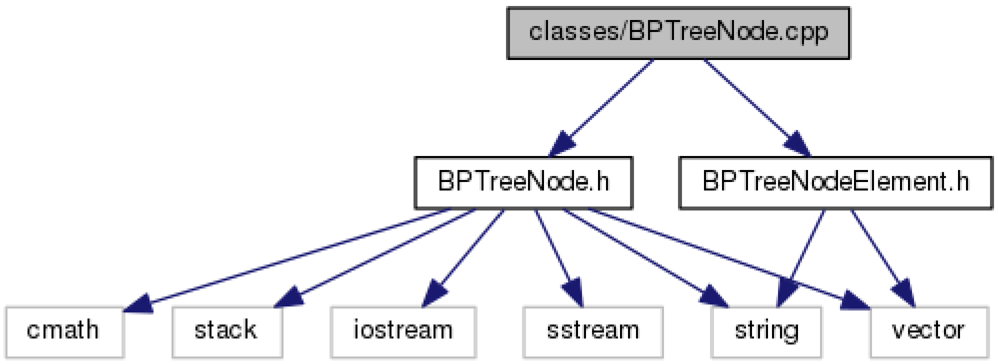
```
class BPTree{
     int order;
     BPTreeNode* root;   /** reference to this B+ tree */

     public:
          BPTree(int);   /** constructor */

          void insert(double, string);
          string search(double);
          string rangeSearch(double, double);
          string vec2string(vector<string>&);
};
```

BPTreeNode.cpp

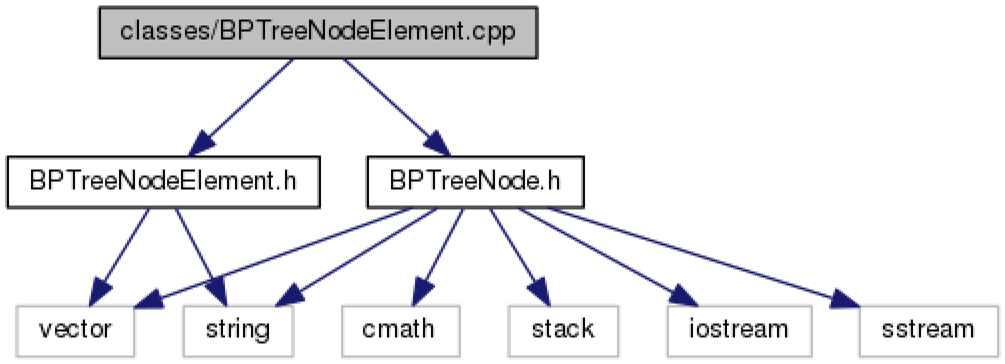
dependency graph



```
BPTreeNode(int, bool);
BPTreeNode(int, bool, BPTreeNodeElement*);
void getState();
void setFilled(int);
BPTreeNode* getNext();
BPTreeNode* getPrev();
void setNext(BPTreeNode*);
void setPrev(BPTreeNode*);
bool isOverflow();
void insert(double, string, stack<BPTreeNode*>&, BPTreeNode*&);
void insert(BPTreeNodeElement*, stack<BPTreeNode*>&, BPTreeNode*&);
void splitNode(stack<BPTreeNode*>&, BPTreeNode*&);
void search(double, vector<string>&);
void rangeSearch(double, double, vector<string>&);
BPTreeNode* findPosition(double);
```

BPTreeNodeElement.cpp

dependency graph



```
BPTreeNodeElement();
BPTreeNodeElement(const BPTreeNodeElement&);
void initialize(double, BPTreeNode*, BPTreeNode*);
BPTreeNode* getLeft();
BPTreeNode* getRight();
double getKey();
void setKey(double);
void setLeft(BPTreeNode*);
void setRight(BPTreeNode*);
void setValues(vector<string>);
void insert(string);
vector<string> getValues();
```