CS 261 – Recitation 6

Binary Search tree

Binary: Every node maximum 2 children
Search: Construction based on Binary search

What is binary search?
Example: The way we search in a dictionary

Not dictionary.com!

Binary search trees are binary tree’s where every node’s object value is:
1. Greater than or equal to all its descendants in the left sub tree
2. Less than or equal to all its descendants in the right sub tree
Example of tree

What is the root?
What is the parent of nodes 4 and 9?
What are the interior nodes?
What are the leaf nodes?
What is the sibling of 9?
What are the descendents of 7?
What is the height of 7?
Is this a binary search tree?
The data Structure: BST

- Distinguishing properties of the tree as WHOLE:
  1. Root: every tree has one and only one root.
  2. Num of nodes
- So, we have

- Properties of individual nodes:
  1. Value: Each existing node has a certain value
  2. Left and Right children: These can be null. Eg. 0 in the adjacent tree
- So, we have:
Operations on a bst

- void initBSTree(struct BSTree *tree)
  - {
    - tree->cnt = 0;
    - tree->root = 0;
  - }

- struct BSTree *newBSTree()
  - {
    - struct BSTree *tree = (struct BSTree *)malloc(sizeof(struct BSTree));
    - assert(tree != 0);
    - initBSTree(tree);
    - return tree;
  - }

int main() {
    struct BSTree *tree1 = newBSTree();
}
Operations on bst (contd..)

- Lets see the operations in action.
- Note: The above applet does not accept duplicate entries in the BST (like most of BST implementations)
- Try following functions in the applet:
  - void addBSTree(struct BSTree *tree, TYPE val);
  - int containsBSTree(struct BSTree *tree, TYPE val);
  - void removeBSTree(struct BSTree *tree, TYPE val);
In-depth analysis of remove

- Find leftmost node of the right child of the node to be removed
- Which is leftmost node of the right child of 7?
  - 8
- So, free(node 7) and replace it by (node 8)

Node `removeNode(Node current, TYPE value)`

  if `value = current.value`
  
  if `right child is null`
  
  return left child

  else
  
  replace value with leftmost child of right child
  
  set right child to be `removeLeftmost(right)`

  else if `value < current.value`
  
  left child = `removeNode(left child, value)`

  else right child = `removeNode(right child, value)`

  return current node
In the earlier examples, we considered integer values like 3, 4, 8 as nodes.
But in real-world examples we will have to arrange complex data structures in a tree according to one of its properties.
Eg. Arrange pizzas on the basis of number of toppings.
Here, we have to store all the data regarding the pizza as a node.
But the arrangement of these nodes will be based on a single property (numToppings)
Important tasks

Figure out way to compare instances of data structures

int compare(TYPE left, TYPE right)
{
    /*
       if left < right return -1
       if left > right return 1
       if left = right return 0
    */
}
C communicates with files using a new datatype called a file pointer. This type is defined within stdio.h, and written as FILE *

Usage:

```c
FILE *output_file;
```
Opening a file pointer

- Your program can open a file using the `fopen` function, which returns the required file pointer.

- If the file cannot be opened for any reason then the value `NULL` will be returned.

- Usage:
  ```c
  output_file = fopen("filename.txt", "w");
  if (output_file != NULL) {
      .... /* do something */
  }
  ```
Opening a file pointer

- `fopen` takes two arguments:

1. the name of the file to be opened (filename.txt).
2. an access character, which is usually one of:
   - “r” - open for reading
   - “w” - open for writing (creates file if it doesn't exist). Deletes content and overwrites the file.
   - “a” - open for appending (creates file if it doesn't exist)
   - Also, r+, w+ & a+. (Please explore on your own)
Reading from a file

- You can read a single character using the function `fgetc`.
  
  ```c
  int fgetc( FILE *fp );
  ```

- String values are read from a file or from a console using the function `fgets`. The function takes as argument a character array, the size of the array, and a file pointer.
  
  ```c
  char buffer[100];
  fgets(buffer, 100, stdin); //stdin means Standard i/p
  printf("You just typed %s\n", buffer);
  ```

- But, next time you read, `fgets` overwrites the previous value stored in the array (buffer []). To solve this, we should copy a string value into a new array which can then be stored in data structure.
  
  ```c
  char * newStr (char * charBuffer) {
      char * p = (char *) malloc(1 + strlen(charBuffer));
      strcpy (p, charBuffer);
      return p;
  }
  ```

CS 261 – Data Structures
Writing to a file

- You can use the `fputc` function to write a single character
  
  ```c
  int fputc( int c, FILE *fp );
  ```

- To write a line into a file, use `fputs` function
  
  ```c
  int fputs( const char *s, FILE *fp );
  ```
Closing a file pointer

- The **fclose** command is used to disconnect a file pointer from a file.
- Usage:
  
  ```c
  fclose(output_file);
  ```
- Make sure to close any open files once you are done working on them to avoid surprises.
GUESS THE ANIMAL GAME

- Implementing a binary tree
  - Interior nodes: Questions
  - Leaf nodes: Animal names
  - “Yes” answer: Move to left subtree
  - “No” answer: Move to right subtree

Does it mew?
  - yes
  - no

Is it domestic?
  - yes
  - no

Does it bark?
  - yes
  - no
Example run

Is the animal a cat?
N

Really? What was your animal?
Dog

Please enter a question that is true for a cat but not of a Dog.
Does it mew?

You Win!! I'll remember that. Please play again later.
Now we have some data in the file

**Case 1:**
Does it mew?
\( \text{Y} \)

Is the animal a **cat**
\( \text{Y} \)

I win! Thanks for playing!

**Case 2:**
Does it mew?
\( \text{Y} \)

Is the animal a **cat**
\( \text{N} \)

Really? What was your animal?

**Big Cat**

Please enter a question that is true for a **cat** but not of a **Big Cat**

**Is it domestic?**

You win!! I'll remember that. Please play again later.