OJ review 2

Jialiang Lu, Huangjie Zheng

06/06/2018

Search in Permutated Sorted Array

Search in Permutated Sorted Array

- Core Idea: BST
  - Split the array into two parts, one part is non-permutated, another one is permutated.
  - Check if the given index is in the non-permutated sorted part
    - If yes, then BST
    - If not, keep splitting

Find the permutation index

- Initialize: l = left, h = right

- while(l<h){
- m=(l+h)/2
- if(A[m]>A[h]) l=m+1
- else h=m
- }

Find Repeated DNA Sequences

Find Repeated DNA Sequences

- Check for every consecutive 10 character-long substring
  - If not in the set
    - Add to the set
  - If exists in the set
    - Output

Pseudo-code

- Initialize: set{}
    - If input in set
        - Output
    - Else
        - Set{input}

Find the shortest way

Find the shortest way

- Open question
  - BFS or DFS
  - Some existing algorithm
    - A*
    - Dijkstra

Summary of the data structure

Review of DS

- Assignment 1
  - Overloading
    - Why overload (class operation)
    - How overload (define a member function)

- Assignment 2, 3, 4
  - Binary tree
    - Preorder, postorder, inorder
    - Build heap, priority queue
    - Huffman coding

Review of DS

- Assignment 5, 6
  - Binary search tree
    - The characteristics of BST (Assignment 5)
    - One typical application (Assignment 6)
- Assignment 7
  - Hash map
- Assignment 8
  - Graph
  - BFS, DFS

Recommended structure: Binary tree

- Binary tree construction
  - 1. Build a node

```
struct node {
    int data;
    struct node* left;
    struct node* right;
}
```

  - 2. Build the tree

```
struct node* NewNode(int data) {
        struct node* node = new(struct node);
        node->data = data;
        node->left = NULL;
        node->right = NULL;

        return(node);
}
```

Recommended structure : Binary tree

- According to your need:
  - Define the member functions
    - **size()**
    - **maxDepth()**
    - **minValue()**
    - **printTree()**

    …

Recommended structure: BST

- BST is built on same node w.r.t. Binary tree

- Difference is the way to insert

```
void BSTTree::Insert(node* & loc,int value)
{
    if (loc == 0)
        struct node* node = new(struct node);
    else if (value<loc->data)
    {
        Insert(loc->left, value);
    }
    else if (value > loc->data)
    {
        Insert(loc->right, value);
    }
    else
        cout << "redundant number!" << endl;
}
```

Thanks

http://huangjiezheng.com/teaching/2018-spring-teaching