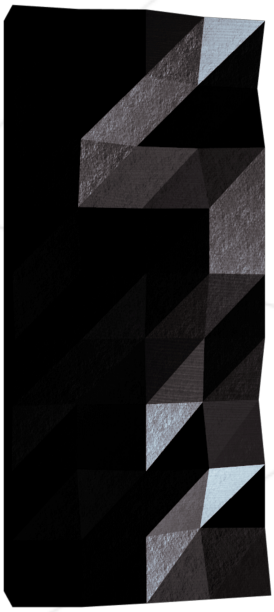

Starfleet

Interview

Staff 42 pedago@42.fr

Summary: This document is an interview question for the Starfleet Piscine.

Contents

Chapter I

General rules

- The interview should last between `45 minutes`.
- Both the interviewer and the interviewed student must be present.
- The interviewed student should write his code using a `whiteboard`, with the language of her/his choice.
- At the end of the interview, the interviewer evaluates the student based on the provided criteria.

Read carefully the interview question and solutions, and make sure you `understand` them before the interview. You can't share this document with other students, as they might be interviewed on the same question. Giving them the answer would prevent them from having to solve an unknown question during an interview.

I.1 During the interview

During the interview, we ask you to :

- Make sure the interviewed student `understands` the question.
- Give her/him any `clarification` on the subject that she/he might need.
- Let her/him come up with a solution before you guide her/him to the best solution given the constraints (time and space).
- Ask the student what is the `complexity` of her/his algorithm ? Can it be improved and how ?
- `Guide` her/him to the best solution without giving the answer. You may refer to the `hints` for that.
- You want to evaluate how the interviewed student thinks, so ask her/him to `explain everything` that she/he thinks or writes (there should be no silences).
- If you see a mistake in the code, wait untill the end and give her/him a chance to correct it by her/himself.
- Ask the student to show how the algorithm works on an `example`.
- Ask the student to explain how `limit cases` are handled.
- Bring out to the student any mistake she/he might have done.
- Give `feedback` on her/his performances after the interview.
- Be `fair` in your evaluation.

As always, stay mannerly, polite, respectful and constructive during the interview. If the interview is carried out smoothly, you will both benefit from it !

Chapter II

Kth smallest element in a BST

II.1 Interview question

`Find the Kth smallest element in a Binary Search Tree.`

EXAMPLE :

For this BST, the Kth smallest element is 27 where k = 6.

II.1.1 Hints

- What is the particularity of a binary search tree?
- How would you traverse a BST to visit the nodes in an increasing order?

II.2 Acceptable answers (no constraint)

II.2.1 Recursive in-order DFS

When you do an in-order traversal of a binary search tree, you visit the nodes from the smallest to the greatest (this is valid only for BSTs). Of course, it requires a Depth First Search (DFS).

The solution here is to perform in-order traversal of the BST and stop when k nodes have been visited.

```
O(k) time , O(k) space
```

code:

```
struct s_node {
        int           value;
        struct s_node *right;
        struct s_node *left;
};

void kthDFS(struct s_node *node, int *k, int *value) {
        if (!node || *k < 1)
                return ;
        kthDFS(node->left, k, value);
        if (*k == 1)
                *value = node->value;
        (*k)--;
        kthDFS(node->right, k, value);
}

int kthSmallest(struct s_node *bst, int k) {
        int value = INT_MIN;

        if (!bst || k < 1)
                return (value);
        kthDFS(bst, &k, &value);
        return (value);
}
```

II.3 Follow up question

You are specifically asked not to use recursion. You must implement an iterative in-order DFS

II.3.1 Hints

- To implement an iterative Breadth First Search (BFS) you use a queue. Which data structure will you use for an iterative DFS?
- Which data structure has properties similar to recursion?

II.4 Best solution

II.4.1 Iterative in-order DFS

Using a stack is the obvious way to traverse a tree without recursion. An iterative in-order DFS follow these steps:

- Initialize an empty stack.
- Repeat until you are done:

1. Push all nodes to the left until you reach a leaf node.
2. Pop the last node added to the stack. If the stack is empty, you are done.
3. Visit that node
4. Set the current node to be its right child.

As before, the solution is to perform in-order traversal of the BST and stop when k nodes have been visited.

```
O(k) time , O(k) space
```

code:

```
struct s_stackNode {
        void *content;
        struct s_stackNode *next;
};

struct s_stack {
        struct s_stackNode *top;
};

struct s_stack *init(void);

void *pop(struct s_stack *stack);

void push(struct s_stack *stack, void *content);

void *peek(struct s_stack *stack);

int isEmpty(struct s_stack *stack);


struct s_node {
        int           value;
        struct s_node *right;
        struct s_node *left;
};

int kthSmallest(struct s_node *bst, int k) {
        struct s_stack *stack;
        struct s_node *current;
        int done = 0;

        if (!bst || k < 1)
                return (INT_MIN);

        stack = init();
        current = bst;
        while (!done) {
                if (current) {
                        push(stack, current);
                        current = current->left;
                } else {
                        if (!isEmpty(stack)) {
                                current = pop(stack);
                                if (k == 1)
                                        return (current->value);
                                current = current->right;
                                k--;
                        } else {
                                done = 1;
                        }
                }
        }
        return (INT_MIN);
}
```