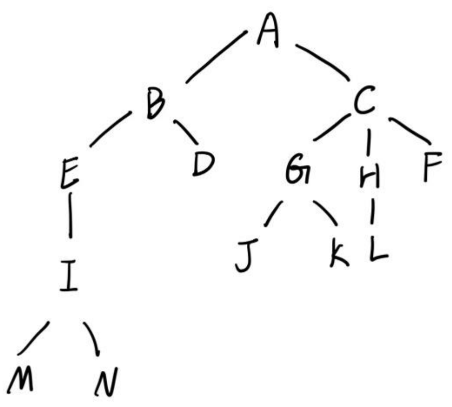
树和二叉树作业

一、基础题

1、



（1）A（2）D M N J K L（3）C（4）C A（5）J K（6）I M N（7）①D ②G H（8）2 14（9）5（10）3

2、 ⑴ k^(i-1)，其中 i=1,2,...,H

⑵ (p-1)/k

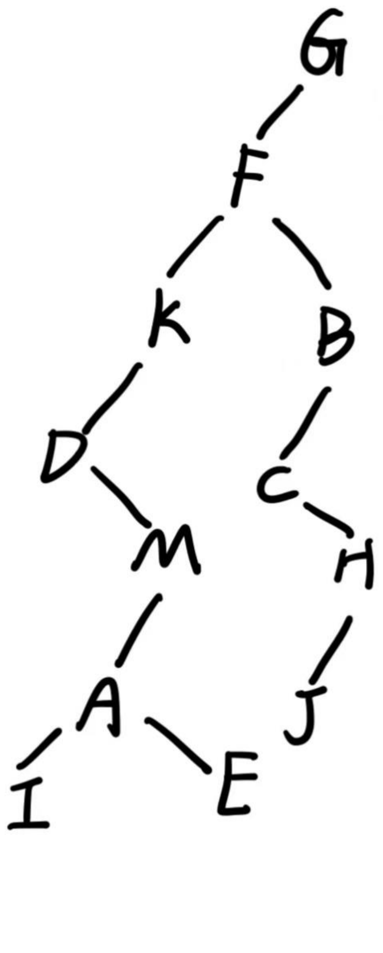
⑶ pk+i

⑷条件：p 不是该层上的最后一个结点，其右兄弟的编号是 p+1。

3、$n_2+2*n_3+3*n_4+\cdots+(k-1)*n_k+1$

4、最大深度为 log(k)n，最小深度为 log(k)n/2

5、



二、算法题

1、

```
#include <iostream>
#include <vector>

using namespace std;

bool is_descendant(int u, int v, vector<vector<int>>& tree) {
    if (u == v) {
        return true;
    }

    for (int child : tree[v]) {
        if (is_descendant(u, child, tree)) {
            return true;
        }
    }

    return false;
}
```

2、

```
#include <iostream>
#include <stack>

using namespace std;

struct TreeNode {
    int value;
    TreeNode* left;
    TreeNode* right;
    int mark; // 0: not visited, 1: visited left, 2: visited both
};

void postorder_traversal(TreeNode* tree) {
    stack<TreeNode*> nodeStack;
    nodeStack.push(tree);

    while (!nodeStack.empty()) {
        TreeNode* node = nodeStack.top();

        if (node->mark == 0) {
            if (node->left && node->left->mark == 0) {
                node->mark = 1;
                nodeStack.push(node->left);
            } else if (node->right && node->right->mark == 0) {
                node->mark = 2;
                nodeStack.push(node->right);
            } else {
                node->mark = 2;
                cout << node->value << " "; // Output or process the node value
            }
        } else if (node->mark == 1) {
            if (node->right && node->right->mark == 0) {
                node->mark = 2;
                nodeStack.push(node->right);
            } else {
                node->mark = 2;
                cout << node->value << " "; // Output or process the node value
            }
        } else {
            nodeStack.pop();
        }
    }
}
```
3、
```
#include <iostream>

using namespace std;

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

void swap_left_right(TreeNode* node) {
    if (node != nullptr) {
        swap(node->left, node->right);
        swap_left_right(node->left);
        swap_left_right(node->right);
    }
}
```
4、
```
#include <iostream>
#include <queue>

using namespace std;
```

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

void level_order_traversal(TreeNode* root) {
        if (!root) {
                return;
        }

        queue<TreeNode*> nodeQueue;
        nodeQueue.push(root);

        while (!nodeQueue.empty()) {
                TreeNode* node = nodeQueue.front();
                cout << node->value << " "; // Output or process the node value
                nodeQueue.pop();

                if (node->left) {
                        nodeQueue.push(node->left);
                }
                if (node->right) {
                        nodeQueue.push(node->right);
                }
        }
}
```