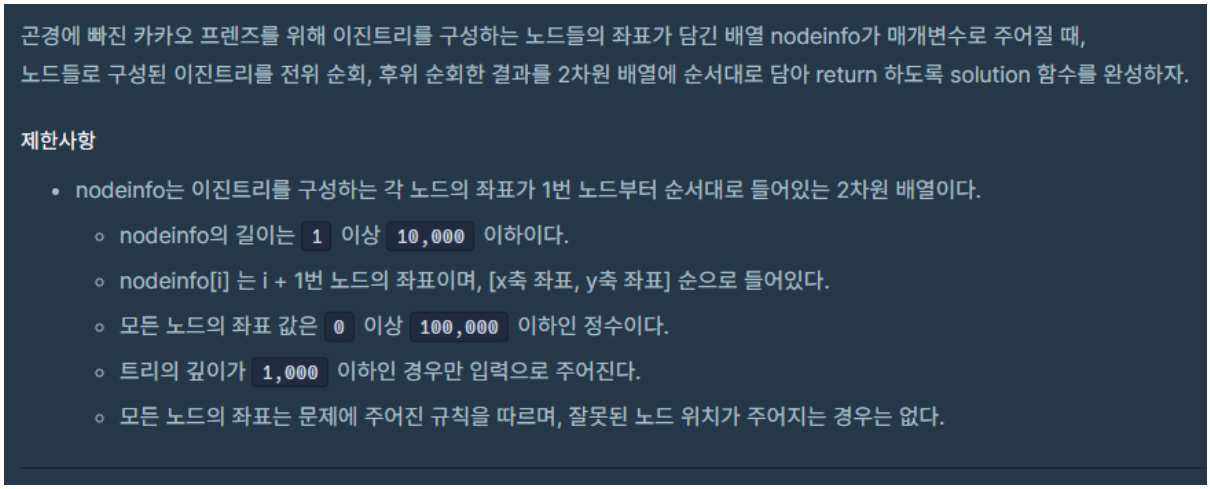
과제 5: 길찾기 게임 문제 202012468 김현서

1. 문제에 대한 자신의 알고리즘 설명

저는 우선 arrays에 문자열을 입력받고, programmers 조건에 나와있는대로 숫자들만nodeinfo 2차원 배열을 이용해 각 노드들의 좌표를 입력받았습니다.


곤경에 빠진 카카오 프렌즈를 위해 이진트리를 구성하는 노드들의 좌표가 담긴 배열 nodeinfo가 매개변수로 주어질 때,
노드들로 구성된 이진트리를 전위 순회, 후위 순회한 결과를 2차원 배열에 순서대로 담아 return 하도록 solution 함수를 완성하자.

제한사항

- nodeinfo는 이진트리를 구성하는 각 노드의 좌표가 1번 노드부터 순서대로 들어있는 2차원 배열이다.
  - nodeinfo의 길이는 `1` 이상 `10,000` 이하이다.
  - nodeinfo[i] 는 i + 1번 노드의 좌표이며, [x축 좌표, y축 좌표] 순으로 들어있다.
  - 모든 노드의 좌표 값은 `0` 이상 `100,000` 이하인 정수이다.
  - 트리의 깊이가 `1,000` 이하인 경우만 입력으로 주어진다.
  - 모든 노드의 좌표는 문제에 주어진 규칙을 따르며, 잘못된 노드 위치가 주어지는 경우는 없다.

그리고 nodeinfo에 각 데이터도 추가했습니다. 1번째 노드면 1번, 2번째 노드면 2번 이런 식으로 데이터를 nodeinfo[I][2]에 추가하였습니다. 그 다음, for문을 이용해 y좌표를 내림차순으로 정렬하고, 만약 y좌표의 값이 같다면 x좌표의 오름차순으로 정렬하였습니다.

그 다음 tree를 선언하였습니다. 12주차 강의에서 배웠던 BinarySearchTree를 응용했는데, BTnode들만 사용해서는 트리를 연결시킬 수 없겠더라구요. 노드들을 하나씩 명명하고, 데이터를 지정시켜주고 x, y좌표에 맞게 입력시켜주려면 더 큰 tree 구조를 먼저 만들어서 노드를 추가하는 방식(연결리스트 때 배웠던 것 응용)으로 알고리즘을 짜면 좋을 것 같다는 생각이 들어서 Tree를 먼저 선언하고, Tree 안에도 head, tail, cur, before함수를 두어서 Tree의 node 간 이동이 가능하게 하였습니다.

Tree를 선언한 이후, 이미 정렬을 해둔 이후이기 때문에 1번째 노드(Root노드)를 입력하고, for문을 이용해 code의 주석으로 달아놓은 경우의 수들을 모두 고려하여 새 노드의 위치를 잡아갔습니다.

```
Tree tree;  // tree 선언

TreeInit(&tree);    // tree 초기화(아무 노드도 없음)

TreeFInsert(&tree, nodeinfo[0][2], nodeinfo[0][0], nodeinfo[0][1]); // nodeinfo[0][2]를 data로 갖고, xpos : nodeinfo[0][0], ypos : nodeinfo[0][1]인 1번째 노드 입력

for (int k = 1; k < node_num; k++) {    // k가 node_num-1일때까지
    TreeInsert(&tree, nodeinfo[k][2], nodeinfo[k][0], nodeinfo[k][1]);  // nodeinfo[k][2]를 data로 갖고, xpos : nodeinfo[k][0], ypos : nodeinfo[k][1]인 노드 입력

    int temp_original_y = nodeinfo[0][1], temp_original_x = nodeinfo[0][0], temp_n = 0; // y좌표, x좌표 임시저장(가장 큰 root노드, 1번째 노드를 첫 값으로 지정)
    for (int n = 0; n < k; n++) {
        if (nodeinfo[n][1] > nodeinfo[k][1]) {      // 새 노드의 y좌표보다 큰 이전 노드들
            if (temp_original_y >= nodeinfo[n][1]) {    // 새 노드보다 한 단계 더 큰 y좌표를 갖는 노드들
                temp_original_y = nodeinfo[n][1];
                if (temp_original_x > (nodeinfo[n][0] - nodeinfo[k][0])) {  // 그 중 새 노드에 가까운 x좌표
                    temp_original_x = abs(nodeinfo[n][0] - nodeinfo[k][0]);
                    temp_n = n;
                }
            }
        }
    }
    //printf("%d %d %d\n", nodeinfo[temp_n][2], nodeinfo[temp_n][0], nodeinfo[temp_n][1]);

    TSearch(&tree, nodeinfo[temp_n][2], nodeinfo[temp_n][0], nodeinfo[temp_n][1]);  // nodeinfo[temp_n][2]를 data로 갖고, xpos : nodeinfo[temp_n][0], ypos : nodeinfo[temp_n][1]인 노드 찾아 cur로

    if (nodeinfo[k][0] - nodeinfo[temp_n][0] > 0)   // 만약 찾은 노드의 xpos 값이 새로 입력한 노드의 xpos보다 작다면
        MakeRightSubTree(&tree);    // 찾은 노드의 오른쪽 서브 노드에 새 노드를 이어줌
    else if (nodeinfo[k][0] - nodeinfo[temp_n][0] < 0)  // 만약 찾은 노드의 xpos 값이 새로 입력한 노드의 xpos보다 크다면
        MakeLeftSubTree(&tree);     // 찾은 노드의 왼쪽 서브 노드에 새 노드를 이어줌

}
```

그 후, 8장에서 배웠던 전위 순회함수와 후위 순회 함수를 이용해 값을 출력했습니다.

2. 수업시간에 배운 트리의 ADT 변경시 무엇을 왜 변경했는지 설명

위에서도 말했듯, 수업시간에 배운 트리에 head, tail, cur, before 등의 포인터를 추가하였고, xpos, ypos를 포함한 NodeInfo를 선언했습니다. 수업시간에 배웠던 것은 data의 대소를 이용해 트리의 노드를 추가하는 것이었는데, x좌표와 y좌표를 트리에 노드를 추가하는 기준으로 하려면 NodeInfo로 선언을 해주고, BTreeNode와 BTData에 입력해주는 방법밖에는 없다고 판단했습니다.

그리고, 포인터들을 이용해 트리 내에서 노드 간 자유로운 이동이 가능하게 하면서 노드를 추가하는 데 어려움을 덜었습니다. 또한 이 부분에서 연결리스트 때 배웠던 것들이 생각이 나서 TreeInsert, TFirst 등의 함수를 만들었는데, 이 함수들을 이용함으로써 Tree에 node들을 추가하기 한층 수월했습니다. 왜냐하면 부모노드를 찾거나, 새 노드가 부모노드의 left 서브노드로 들어가는지 right 서브노드로 들어가는지 쉽게 판단 가능했기 때문입니다.