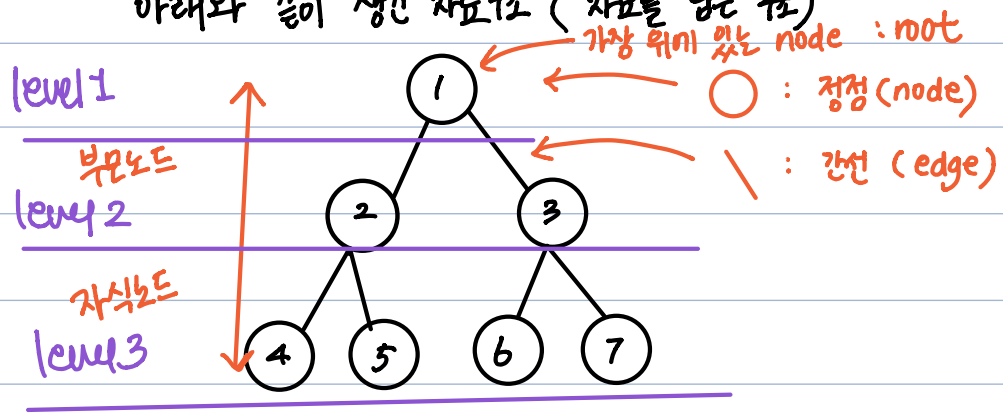
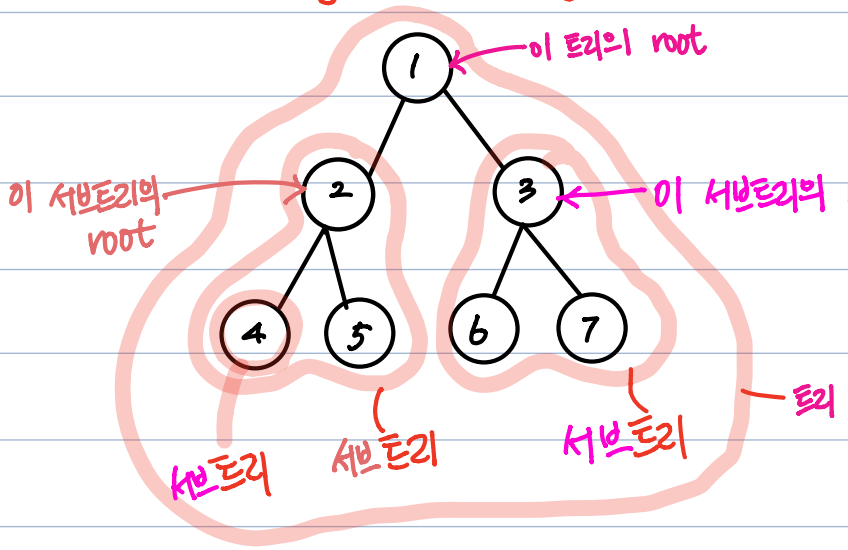
자료구조 ③ 트리

아래와 같이 생긴 자료구조 (자료를 담는 구조)



트리의 성질) ☆ 트리의 재귀적 성질 : 트리는 그 안에 또 트리가 존재한다.



트리의 순회) 트리 내에 어떤 자료가 담겨있는지를 알기 위함. by 트리의 재귀적 성질을 이용해서.

* 위와 같이 자식노드가 2개 이하인 트리를 가지고 순회를 배워보자.

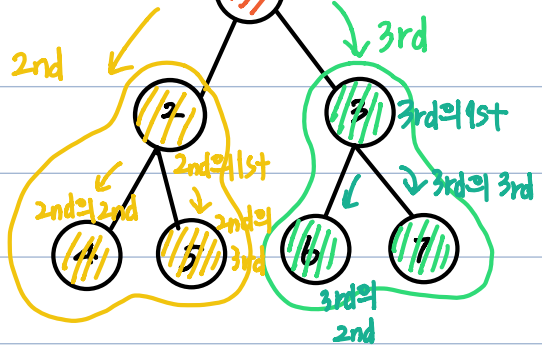
↳ 이진트리 (binary tree)

이진트리 순회방법) ① 전위순회 (Root - L - R)
② 중위순회 (L - Root - R)
③ 후위순회 (L - R - Root)

목적은 같은데, 순서만 다른 뿐. (특성)

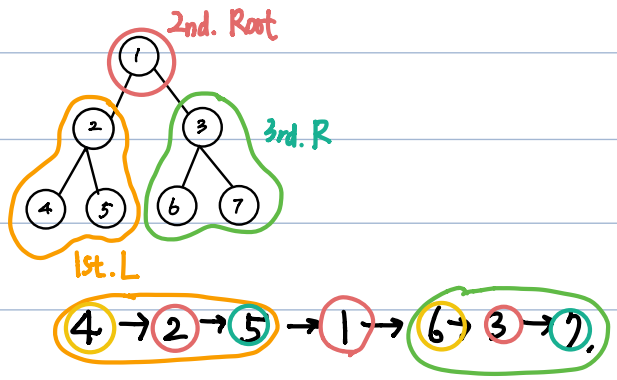
① 전위순회 (Preorder) : Root node 먼저 check → Left subtree 를 순회 → Right subtree 를 순회

예제) 1st ⇒ 아~ 1이 있구나



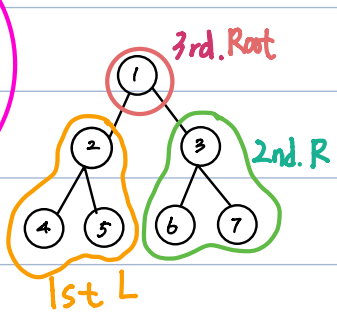
→ 그래서 순회되는 순서는 1→2→4→5→3→6→7

② 중위순회 (inorder)



가장많이 쓰임.

③ 후위순회 (Postorder)



4 → 5 → 2 → 6 → 7 → 3 → 1

이진트리 순회의 구현)

[예제 1] 이진트리 순회

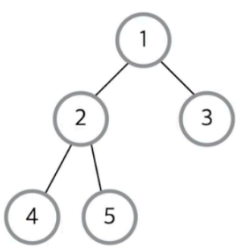
* 이진 트리가 주어질 때, 트리를 순회한 결과를 출력하라
  단, 노드 번호는 1부터 N까지로 주어진다.

입력의 예

```
5
1 2 3
2 4 5
3 -1 -1
4 -1 -1
5 -1 -1
```

출력의 예

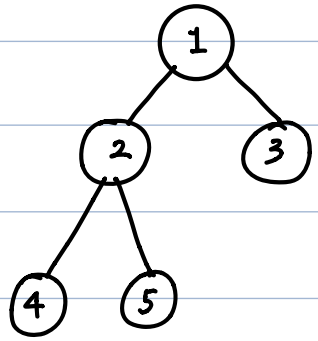
```
1 2 4 5 3   전위순회
4 2 5 1 3   중위순회
4 5 2 3 1   후위순회
```



Tree tree[5];

node	left	right
1	2	3
2	4	5
3	-1	-1
4	-1	-1
5	-1	-1

→ -1은 아무것도 없다는 의미.




```c
#include <stdio.h>

const int MAX = 100;

struct Tree {
  int left;
  int right;
};


Tree tree[MAX];
//tree[i] = 노드 i의 정보를 담고 있음.
//tree[i].left = 노드 i의 왼쪽의 노드 번호.
//tree[i].right = 노드 i의 오른쪽의 노드 번호.

void preorder(int x){
  //preorder : x를 루트로 하는 서브트리를 전위순회하여 출력하는 함수.

  //기저조건 : 왼쪽과 오른쪽이 모두 -1일때 (노드가 아무것도 없을 때)
  if(tree[x].left == -1 && tree[x].right == -1)
    printf("%d ", x);
  else{//기저조건이 아니면, Root -> Left -> Right
    printf("%d ", x);//1st. Root

    if(tree[x].left != -1)
      preorder(tree[x].left);//2nd. Left
    if(tree[x].right != -1)
      preorder(tree[x].right);//3rd. Right

  }
}

```

```c
void inorder(int x){
  //inorder : x를 루트로 하는 서브트리를 중위순회하여 출력하는 함수

  if(tree[x].left == -1 && tree[x].right == -1)
    printf("%d ", x);
  else {
    if(tree[x].left != -1)
      inorder(tree[x].left);//1st. Left

    printf("%d ", x);//2nd. Root

    if(tree[x].right != -1)
      inorder(tree[x].right);//3rd. Right
  }
}

void postorder(int x){
  //postorder : x를 루트로 하는 서브트리를 후위순회하여 출력하는 함수
  if(tree[x].left == -1 && tree[x].right == -1)
    printf("%d ", x);
  else {
    if(tree[x].left != -1)
      postorder(tree[x].left);//1st. Left

    if(tree[x].right != -1)
      postorder(tree[x].right);//2nd. Right

    printf("%d ", x);//3rd. Root


  }
}

int main() {

  //이진트리 순회 구현

  int n;//tree의 노드 개수.
  scanf("%d", &n);

  for(int i=0;i<n; i++){
    int a, b, c;
    scanf("%d %d %d", &a, &b, &c);
    //a : 노드 a의 값,
    //b : 노드 a의 왼쪽 값,
    //c : 노드 a의 오른쪽 값

    tree[a].left = b;
    tree[a].right = c;

  }

  preorder(1);
  printf("\n");

  inorder(1);
  printf("\n");

  postorder(1);
  printf("\n");

  return 0;
}
```

tree[a].left = b
tree[a].right = c

C/C++ 12 실행

```
5
1 2 3
2 4 5
3 -1 -1
4 -1 -1
5 -1 -1
```

```
1 2 4 5 3
4 2 5 1 3
4 5 2 3 1
```

↑* 위의 Preorder 알고리즘 수도코드 (내가 이해한 방식)

1. 일단 Tree 짜야하네?

```
Struct Tree{
    Tree tree[MAX]
    int left
    int right
};
```

2. Const int MAX = 100

3. main 함수 먼저.

```
int main(){
    int n
    Scanf(n)
    for(int i=0~<n)
        Scanf(&a, &b, &c) // 입력받고
        tree[node].left = b // tree[node]가 root, root의 left에 집어넣고
        tree[node].right = c // root의 right에 집어넣는다.
Preorder(1) // 1번 node부터 전위순회.
```

4. Preorder 함수를 만들어야 함.

```
void Preorder(int node){
    //기저조건
    if( b, c 둘다 -1)
        Printf(a) // root 출력하고 끝
    else 아니면
        Printf(a) 1st. Root
        if( b != -1 이면)
            Preorder(b) 2nd. left
        if( c != -1 이면)
            Preorder(c) 3rd. right
```

* a b c
↑ ↑ ↑
node left right
를 했다가
Tree.left .right 와
계속해서 error.

백준 1991 - 트리 순회

```
#include <stdio.h>

const int Max = 30;
int n;

struct Tree{
    char left;
   char right;
}

Tree tree[Max]; //왜 여기서 컴파일 에러가 계속 뜰까 .. 위의 것과 같이 했는데.. struct는 int만 되는 건가..?

void Preorder(int x){
   if(tree[x].left == '.' && tree[x].right == '.'){
      printf("%c", (char)(x+'A'))
   }
   else{
      printf("%c", (char)(x +'A'));
      if(tree[x].left != '.')
         Preorder(tree[x].left);
      if(tree[x].right != '.')
         Preorder(tree[x].right);
   }
   return;
}

void Inorder(int x){
   if(tree[x].left == '.' && tree[x].right == '.')
      printf("%c", (char)(x+'A'));
   else{
      if(tree[x].left != '.')
         Inorder(tree[x].left);
      printf("%c", (char)(x+'A'));
      if(tree[x].right != '.')
         Inorder(tree[x].right);
   }
   return;
}


void Postorder(int x){
   if(tree[x].left == '.' && tree[x].right == '.')
      printf("%c", (char)(x+'A'));
   else{
      if(tree[x].left != '.')
         Postorder(tree[x].left);
      if(tree[x].right != '.')
```

```
        Postorder(tree[x].right);
    printf("%c", (char)(x+'A'));
  }
  return;
}


int main(){
  scanf("%d", &n);

  for(int i=1; i<=n; i++){
    char c1, c2, c3;
    scanf("%c %c %c", &c1, &c2, &c3);

    tree[c1-'A'] = c1;
    tree[c1-'A'].left = c2;
    tree[c1-'A'].right = c3;
  }

  Preorder(0); //'A'- 'A'ㄴ | ㄲ ㅏ.
  printf("\n");
  Inorder(0);
  printf("\n");
  Postorder(0);
  printf("\n");

  return ;
}
```

$\perp$