

Programming Assignment #3

2016025187 김도은

The screenshot shows a macOS file manager window titled "HW3_2016025187" with a sidebar containing folders "project1", "project2", and "project3". The main pane shows a file tree with "HW#3" (containing "HW3_2016025187" and "HW3_report"), "HW3_screenShot", and a list of files: "3-1.cpp", "3-1.out", "3-2.cpp", "3-2.out", "3-3.cpp", "3-3.out", "input3-1.txt", "input3-2.txt", and "input3-3.txt". Below the file manager is a terminal window with the same title bar. The terminal shows the following commands and output:

```
~/Documents/2-2/2-2학기/알고/hw/project3/HW3_2016025187
[> g++ -o 3-1.out 3-1.cpp

~/Documents/2-2/2-2학기/알고/hw/project3/HW3_2016025187
[> g++ -o 3-2.out 3-2.cpp

~/Documents/2-2/2-2학기/알고/hw/project3/HW3_2016025187
[> g++ -o 3-3.out 3-3.cpp

~/Documents/2-2/2-2학기/알고/hw/project3/HW3_2016025187
[> ./3-1.out
(1, 2) tree edge
(2, 5) tree edge
(5, 4) tree edge
(4, 2) back edge
(1, 4) forward edge
(3, 5) cross edge
(3, 6) tree edge
(6, 6) back edge
1 2 5 4 3 6

~/Documents/2-2/2-2학기/알고/hw/project3/HW3_2016025187
[> ./3-2.out
1 2 5 3 6 4
1: 1
2: 1
3: 2
4: 3
5: 1
6: 2

~/Documents/2-2/2-2학기/알고/hw/project3/HW3_2016025187
[> ./3-3.out
1
7 3 2 1 4 6 5

~/Documents/2-2/2-2학기/알고/hw/project3/HW3_2016025187
[> ls
3-1.cpp      3-2.cpp      3-3.cpp      input3-1.txt input3-3.txt
3-1.out      3-2.out      3-3.out      input3-2.txt

~/Documents/2-2/2-2학기/알고/hw/project3/HW3_2016025187
[> 
```

The terminal window title bar shows the path: "HW3_2016025187 — ~/Documents/2-2/2-2학기/알고/hw/project3/HW3_2016025187". The bottom of the image shows a breadcrumb trail: "문서 > 2-2 > 2-2학기 > 알고 > hw > project3 > HW3_2016025187".

3-1 : DFS with edge classification

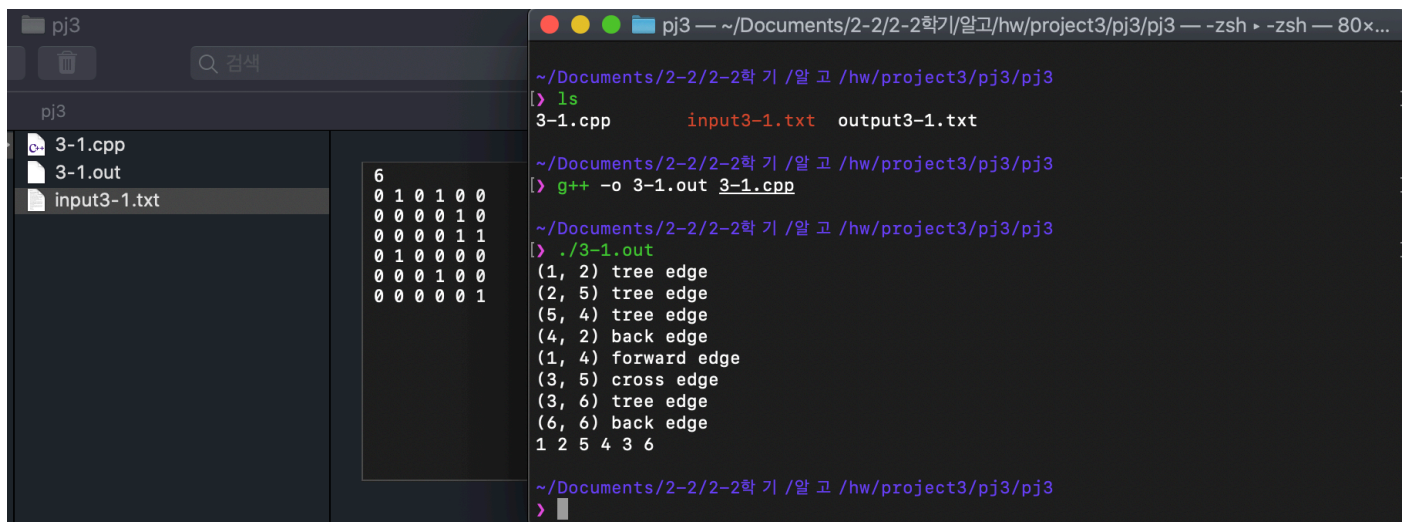
2016025187 김도은

함수

1. 트리간선 (tree edge) :
처음 발견하는 정점으로 가는 경우
2. 순방향 간선 (forward edge) :
이미 발견했던 정점으로 가는 경우 + 이어진 정점보다 먼저 발견된 경우
3. 역방향 간선 (back edge) :
이미 발견했던 정점으로 가는 경우 + 이어진 정점보다 늦게 발견된 경우 + 이어진 정점의 탐색이 종료되지 않은 경우
4. 교차 간선 (cross edge) :
이미 발견했던 정점으로 가는 경우 + 이어진 정점보다 늦게 발견된 경우 + 이어진 정점의 탐색이 종료된 경우

```
29 void dfs(int now) {
30     order.push_back(now+1);
31     discovered[now] = cnt++;
32
33     for (int i = 0; i < adj[now].size(); i++){
34         int next = adj[now][i];
35         printf("(%d, %d) ", now+1, next+1);
36
37         if (discovered[next] == -1) {
38             printf("tree edge\n");
39             dfs(next);
40         }
41         else if (discovered[now] < discovered[next]){
42             printf("forward edge\n");
43         }
44         else if (finished[next] == 0){
45             printf("back edge\n");
46         }
47         else{
48             printf("cross edge\n");
49         }
50     }
51
52     finished[now] = 1;
53 }
```

G++ 환경 : mac os / terminal



```
~/Documents/2-2/2-2학기/알고/hw/project3/pj3/pj3
ls
3-1.cpp      input3-1.txt  output3-1.txt

~/Documents/2-2/2-2학기/알고/hw/project3/pj3/pj3
g++ -o 3-1.out 3-1.cpp

~/Documents/2-2/2-2학기/알고/hw/project3/pj3/pj3
./3-1.out
(1, 2) tree edge
(2, 5) tree edge
(5, 4) tree edge
(4, 2) back edge
(1, 4) forward edge
(3, 5) cross edge
(3, 6) tree edge
(6, 6) back edge
1 2 5 4 3 6
```

3-2 : DFS with connected component identification

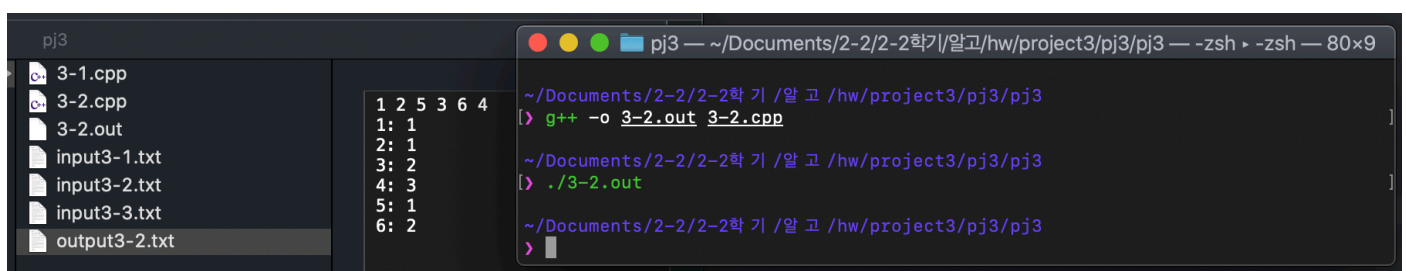
2016025187 김도은

함수

1. 전역변수 component변수를 이용하여 해당 node의 소속 component값을 체크한다

```
28 void dfs(int now) {
29     if (check[now] != 0){
30         return;
31     }
32     check[now] = component;
33     printf("%d ", now);
34     for (int i = 0; i < graph[now].size(); i++) {
35         if (check[graph[now][i]] == 0){
36             dfs(graph[now][i]);
37         }
38     }
39     return;
40 }
41
42 int main() {
43
44     input = fopen("input3-2.txt", "r");
45
46     fscanf(input, "%d", &node);
47     for(int i=1; i<=node; i++){
48         for(int j=1; j<=node; j++){
49             fscanf(input, "%d", &arr[i][j]);
50             if(arr[i][j] == 1){
51                 graph[i].push_back(j);
52             }
53         }
54     }
55
56     for (int i=1; i<=node; i++) {
57         if (check[i] != 0){
58             continue;
59         }
60         dfs(i);
61         component++;
62     }
63
64     printf("\n");
65     for (int i=1; i<=node; i++) {
66         printf("%d: %d\n", i, check[i]);
67     }
68     return 0;
69 }
70
```

G++ 환경 : mac os / terminal



3-3 : Topological sort

2016025187 김도은

함수

1. 방향 그래프에 존재하는 각 정점들의 선행 순서를 위배하지 않으면서 모든 정점을 나열하자
2. DAG(directed acyclic graph)란 간선에 방향이 있고, 사이클이 없는 그래프를 말한다
3. 시작점과 끝점이 같아지는 sequence가 있다면 cycle이 있다고 한다

```
30 void dfs(int x){
31     check[x]++;
32
33     //종료되는 시점에서 해당 정점과 연결된 정점들에 대해 모든 depth를 보았을 때 종료
34     //(간선이 없거나 이미 방문한 경우)
35     if (visit[x]){
36         return;
37     }
38     else{
39         visit[x] = 1;
40         for (int i=1; i<=node; i++){
41             if (graph[x][i]){
42                 dfs(i);
43             }
44         }
45         //dfs종료 시점에서 출력한 역순은 topSort의 결과가 된다
46         topSort[node_tmp--] = x;
47         return;
48     }
49 }
50
51 void topological_sort(void){
52
53     //초기 : 선행 정점을 가지지 않는 정점에 대해
54     for (int i=1; i<=node && DAG; i++){
55         if (!visit[i]){
56             for (int j=1; j<=node; j++){
57                 check[j] = 0;
58             }
59
60             dfs(i);
61
62             //u, v간의 dependency가 v->u라고 가정하자
63             //dfs(v)에서 v와 연결된 간선들의 집합 E(v)에 대해
64             //dfs(u)가 먼저 종료되어야 한다
65             //그렇지 않다면 주어진 그래프가 DAG가 아닌 cycle이 존재한다
66             if (check[i] > 1){
67                 DAG = 0;
68             }
69         }
70     }
71 }
```

G++ 환경 : mac os / terminal

```
~ /Documents/2-2/2-2학기/알고/hw/project3/pj3/pj3 — zsh — zsh — 80x...
~ /Documents/2-2/2-2학기/알고/hw/project3/pj3/pj3
> ls
3-1.cpp      3-3.cpp      input3-1.txt  input3-3.txt  output3-3.txt

~ /Documents/2-2/2-2학기/알고/hw/project3/pj3/pj3
> g++ -o 3-3.out 3-3.cpp

~ /Documents/2-2/2-2학기/알고/hw/project3/pj3/pj3
> ./3-3.out

~ /Documents/2-2/2-2학기/알고/hw/project3/pj3/pj3
> cat output3-3.txt
1
7 3 2 1 4 6 5

~ /Documents/2-2/2-2학기/알고/hw/project3/pj3/pj3
>
```