

## Lab 13-1 (5/25) Topological Sort

**Due : ~2022.05.30.(Mon) 23:59, Late Submission : ~2022.05.31.(Tue) 23:59**

- **Topological Sorting Implementation using Queue.**
- 아래의 사진과 같이 [input].txt 파일을 입력받아 **Topological Sort**를 수행하고 결과값을 [output].txt 파일에 출력하여 저장한다.
- **Implement Topological Sorting Algorithm by receiving the [input].txt file as input commands as below, and the output result is stored in the [output].txt file.**

The screenshot shows a terminal window with two tabs: 'input1.txt' and 'output1.txt'. The 'input1.txt' tab contains the following text:

```
1 2 3 6 5 7
1-2 1-6 2-5 2-6 2-3 3-5 5-6 7-3 7-5
```

The 'output1.txt' tab contains the following data:

	1	2	3	5	6	7
1	0	1	0	0	1	0
2	0	0	1	1	1	0
3	0	0	0	1	0	0
5	0	0	0	0	1	0
6	0	0	0	0	0	0
7	0	0	1	1	0	0

Below the matrix, the sorted nodes are listed: 1 7 2 3 5 6.

- **<input>** :
  - 첫째줄: Graph에서 각 Node(vertex)들의 Key 값.
  - 둘째줄: Graph에서 Edge들의 정보. (**SourceNode-TargetNode**의 형식, ex: “7-3” : (7)---->(3) 의미)
- **1st line: Key values of the Nodes(vertice) in the Graph.**
- **2nd line: Information of the Edges in the graph. (in the format of **SourceNode-TargetNode**, ex: “7-3” : (7)---->(3))**
- **<output>** :
  - Graph의 Adjacent Matrix 출력
  - Graph의 Topological Sorting 결과 출력
  - print out the Adjacent Matrix of the graph.
  - print out the result of Topological Sorting.

**<Structure & Function Format>**

## Structure

```
typedef struct _Queue {  
    int* key;  
    int first;  
    int rear;  
    int qsize;  
    int max_queue_size;  
}Queue;
```

```
typedef struct _Graph {  
    int size;  
    int* node;  
    int** matrix;  
}Graph;
```

## Function

```
Graph* CreateGraph(int* nodes, int n);  
void InsertEdge(Graph* G, int a, int b);  
void PrintGraph(Graph* G);  
void DeleteGraph(Graph* G);  
void Topsort(Graph* G);  
Queue* MakeNewQueue(int X);  
int IsEmpty(Queue* Q);  
int Dequeue(Queue* Q);  
void Enqueue(Queue* Q, int X);  
void DeleteQueue(Queue* Q);
```

- 위 사진과 같은 Struct 구조체를 사용하셔야 합니다.
- 위 사진과 같은 함수들을 형식에 맞게 구현해주시면 됩니다.
- **Struct format above should be used for implementation**
- **Functions should be implemented in appropriate format as above.**

### <File Name Format>

- [StudentID].c      ex) 20XXXXXXXXXX.c

### <Execution>

- gcc 20XXXXXXXXXX.c -o 20XXXXXXXXXX
- ./20XXXXXXXXXX [input\_file\_name] [output\_file\_name]
- !!! 꼭 제공되는 testcase로 실행시켜보시기 바랍니다!!!!,
- !!! Run your solution code with the provided test case above and check whether it works properly !!!
- 

### <Issue>

- 코드 작성시 주석을 적어주시기 바랍니다. 주석이 없는경우 Cheating으로 간주될 수 있습니다.
- 제공된 testcase는 채점 case에 포함됩니다. 모두 알맞게 나오는지 확인해보시기 바랍니다.
- 파일 입출력은 `argv[]`를 사용하여 구현해주시기 바랍니다.
- 제출 마감 시간 이전의 가장 최신 버전의 commit을 기준으로 채점할 예정입니다.
- 제출 파일과, 폴더 naming 은 꼭 지정된 형식으로 해주셔야 합니다.
- **Please write down the detailed comments when writing the code. If there is no comment, it might be considered cheating.**
- **Provided test case is included in the test case for grading. Please check to see if it makes a proper result.**
- **Do not use a fixed file name when inputting and outputting files, but implement it using argv[] as in skeleton code.**
- **Scoring will be based on the latest version of commit before the deadline.**
- **The names of the .c file and directory should be named in proper format.**

- **Adjacent Matrix** 출력시 위 사진과 같이 **graph[SourceNode][TargetNode]** 형태로 출력하셔야 하고, **Topological Sorting** 결과 출력시, **priority**가 같은 **node**들의 경우 작은 **key**값의 **node**부터 오름차순으로 출력하셔야 합니다.
- 출력시 꼭 정해진 형식에 맞게 출력해주셔야 합니다. **Matrix** 출력시 **모든 공백은 띄어쓰기 2칸입니다, Sorting** 출력시 **모든 공백은 띄어쓰기 1칸입니다.** 모든 출력 메시지의 알파벳은 소문자만 사용하여 출력 합니다.
- **Graph**에서 **Cycle**이 존재하는 경우, **Topological Sorting** 결과 대신 에러메시지를 출력하셔야 합니다. 에러메시지의 형식은 “**sorting error : cycle!**” 입니다.
- -> **Adjacent Matrix** 출력시 위 사진과 같이 각 Node의 Key값도 함께 출력 (**Node의 순서는 Key값 기준 오름차순으로 정렬하여 출력.**)
- -> (**Adjacent Matrix** 출력시 첫번째 줄에서 Node의 Key값 출력시 첫 공백은 공백 3칸.)
- -> **Adjacent Matrix** 출력과, **Topological Sorting** 결과 출력 사이에 한줄 공백 존재.
- -> 마지막줄의 경우, **sorting** 결과 출력후에 줄바꿈 후 **EOF**.
- -> **Graph**에서 **Cycle** 존재시, **Sorting** 결과대신 “**sorting error : cycle!**” 출력.
- When printing the **Adjacent Matrix**, it should be in the format of **graph[SourceNode][TargetNode]** as an example above.
- When printing the result of **Topological Sorting**, if there exists multiple nodes that have the same priority, key values should be printed in ascending order.
- All the messages must be printed out according to the appropriate format as shown in the example above. **When printing the Matrix, all the spaces are “two spaces”, When printing the sorting result, all the spaces are “one space”.** Only lowercase letters should be used for the alphabets in output messages.
- When the cycle exists in the Graph, the error message should be printed out instead of the sorting result in the format of “**sorting error : cycle!**”.
- -> When printing the **Adjacent Matrix**, all the key values for each Node should be printed together as an example above(nodes should be printed in ascending order for the key values.)
- -> (When printing the first line of the **Adjacent Matrix**, space before the first key value is “three spaces”)
- -> A single line(“\n”) exists between the **Adjacent Matrix** and the result of **Topological Sorting**.
- -> **newline(\n) after the last line(sorting result), and EOF.**
- -> If the cycle exists in the Graph, “**sorting error : cycle!**” instead of the sorting result.

#### <Directory Format>

- 아래와 같이 git 프로젝트 폴더에 “lab13-1” 폴더 생성후, “lab13-1” 폴더 안에 “20XXXXXXXXXX.c” 파일을 위치시키시면 됩니다.

- After creating the "lab13-1" directory in the git-project-directory as below, place the "20XXXXXXXXXX.c" file in the "lab13-1" directory.

2022

Lab 11 /

2020xxxxxxxxx.e

—

---lab12/

-----2020XXXXXX.c

— · · ·

---lab13-

-----2020XXXXXX.c

23