Introduction to Data Structures

Jinkyu Lee

Dept. of Computer Science and Engineering,
Sungkyunkwan University (SKKU)

Homework 4B

- 40 points for coding evaluation
  - Submission format
    - File name: yourid_HW4B.c
      - Example: 2000123456_HW4B.c
    - File type: .c (NOT .cpp)
  - Submission site: https://icampus.skku.edu
    - Week 11: [Homework] 4B (code)

- 1 point for report
  - The report is not evaluated in detail but evaluated as Pass/Fail
  - Template: Homework Report Template.docx
  - Submission format
    - File name: yourid_HW4B.pdf
      - Example: 2000123456_HW4B.pdf
    - File type: .pdf
  - Submission site: https://icampus.skku.edu
    - Week 11: [Homework] 4B (report)

- Due date
  - 11/24 23:59 (no late submission accepted)

Rules for homework

- You should follow instructions.
  - Complier
    - You will get no/less point if your program cannot be complied with the specified complier
  - Input/output format
    - You will get no/less point if TA's automatic evaluation program cannot parse your input or output.
  - Permitted modification scope
    - You will get no/less point if you modify code outside of the permitted modification scope
  - All other rules
    - You will get severe penalty or no/less point if you violate the given rules.

Complier and input/output rules for homework

- Every implementation homework will be evaluated by TA's automatic evaluation program with the following complier.
  - Complier: GCC 7.X, 8.X, 9.X or 10.X
    - https://gcc.gnu.org/
  - You will get no/less point if your program cannot be complied with GCC 7.X, 8.X, 9.X or 10.X.
    - For example, do not rely on visual studio.
  - You can use standard library such as *stdlib.h* and *math.h*.
- Input/output format
  - You will get no/less point if TA's automatic evaluation program cannot parse your input or output according to the following rules.
  - Use `stdin` and `stdout`

Problem

- Problem: Implementation of graph traversal
  - Your data is either ‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’, … ‘Z’, meaning that your data is an upper-case letter.
  - ~~For given graph, count the number of each element through graph traversal~~
  - Node id starts from 0 and then 1, 2, 3, ….
  - Stack and queue code will be served (see “stackqueue.h”)
  - You can assume that each graph is a connected component.

  - You should construct and ~~print a graph~~ with the “adjacency list” that every vertex has a linked list.
  - The first node to be visited is the node with 0.
  - You should output the results of the Depth-First Search(DFS) and Breadth-First Search(BFS) on the graph.
    - For DFS, you should visit the node with the largest id first when you have multiple adjacent nodes.
    - For BFS, you should visit the node with the smallest id first when you have multiple adjacent nodes.

Input/Ouput

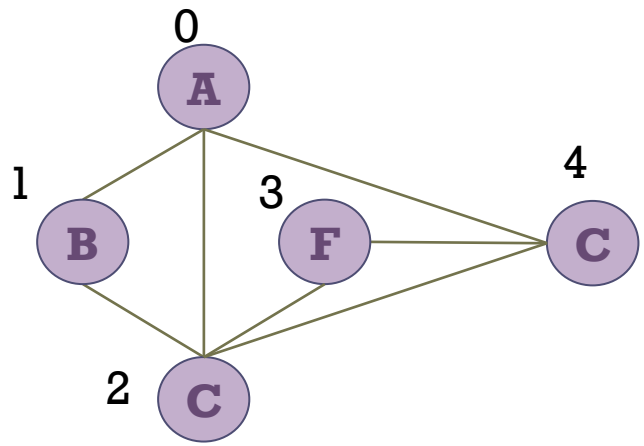
- Input

```
5 7
ABCFC
0 1
0 2
0 4
1 2
2 3
2 4
3 4
```

5 7 ---------- # of nodes # of edges

ABCFC ---------- Data of nodes 0, 1, 2, …

edges



- Output

0 4 3 2 1 ---------- DFS result

0 1 2 4 3 ---------- BFS result

For DFS, you should visit the node with the largest id first when you have multiple adjacent nodes.
For BFS, you should visit the node with the smallest id first when you have multiple adjacent nodes.

Template (4B_template.c)

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "stackqueue.h “
/* Modify from here */


/* Modify to here */


#define MAX_SIZE 100
typedef enum {false, true} bool;

typedef struct _GNode {
        int id;
        char data;
        struct _GNode *next;
} GNode;

typedef struct {
        int num;
        GNode **heads;
} Graph;

void CreateGraph(Graph *pgraph, int num, char
data[]);

void DestroyGraph(Graph *pgraph);

void AddEdge(Graph *pgraph, int src, int dest);

void PrintGraph(Graph *pgraph);

void DFS(Graph *pgraph);

void BFS(Graph *pgraph);

void GetInput();
/* Modify from here */


/* Modify to here */
```

```c
int main() {

        Graph g;
        CreateGraph(&g, 5, "ABCFC");
        AddEdge(&g, 0, 1);
        AddEdge(&g, 0, 2);
        AddEdge(&g, 0, 4);
        AddEdge(&g, 1, 2);
        AddEdge(&g, 2, 3);
        AddEdge(&g, 2, 4);
        AddEdge(&g, 3, 4);


        DFS(&g);
        BFS(&g);

        DestroyGraph(&g);

        // GetInput();
        /*
        5 7
        ABCFC
        0 1
        0 2
        0 4
        1 2
        2 3
        2 4
        3 4
        */

        return 0;

}
```

```c
void GetInput(){
        int node, edge, src, des;
        char *a;

        scanf("%d %d",&node, &edge);
        getchar();

        a = malloc(sizeof(char)*(node+1));
        scanf("%s",a);
        getchar();

        Graph g;
        CreateGraph(&g, node, a);
        for(int i = 0; i<edge; i++){
                scanf("%d %d", &src, &des);
                AddEdge(&g, src, des);
                getchar();

        }


        DFS(&g);
        BFS(&g);

        DestroyGraph(&g);

}

/* Modify from here */


/* Modify to here */
```

Template (stackqueue.h)

- “stackqueue.h”

```
#define MAX_STACK 100
#define MAX_QUEUE 100

typedef enum { false, true } bool;

typedef struct {
        int items[MAX_STACK];
        int top;
} Stack;

typedef struct {
        int front, rear;
        int items[MAX_QUEUE];
} Queue;

void InitStack(Stack *pstack) {
        pstack->top = -1;
}

bool IsSFull(Stack *pstack) {
       return pstack->top == MAX_STACK - 1;
}

bool IsSEmpty(Stack *pstack) {
        return pstack->top == -1;
}
```

```
 int SPeek(Stack *pstack) {
        if (IsSEmpty(pstack))
                exit(1); //error: empty stack
        return pstack->items[pstack->top];
}

void Push(Stack *pstack, int item) {
        if (IsSFull(pstack))
                exit(1); //error: stack full
        pstack->items[++(pstack->top)] = item;
}

int Pop(Stack *pstack) {
        if (IsSEmpty(pstack))
                exit(1); //error: empty stack
        int item = pstack->items[pstack->top];
        --(pstack->top);
        return item;
}

void InitQueue(Queue *pqueue) {
        pqueue->front = pqueue->rear = 0;
}

bool IsQFull(Queue *pqueue) {
        return pqueue->front == (pqueue->rear +
1) % MAX_QUEUE;
}
```

```
bool IsQEmpty(Queue *pqueue) {
        return pqueue->front == pqueue->rear;
}

int QPeek(Queue *pqueue) {
        if (IsQEmpty(pqueue))
                exit(1); //error: empty stack
        return pqueue->items[pqueue->front];
}

void EnQueue(Queue *pqueue, int item) {
        if (IsQFull(pqueue))
                exit(1); //error: stack full
        pqueue->items[pqueue->rear] = item;
        pqueue->rear = (pqueue->rear + 1) %
MAX_QUEUE;
}

int DeQueue(Queue *pqueue) {
        if (IsQEmpty(pqueue))
                exit(1); //error: empty stack
        int item = pqueue->items[pqueue->front];
        pqueue->front = (pqueue->front + 1) %
MAX_QUEUE;
        return item;
}
```

Template

- You cannot modify the template except the space between /*Modify from here*/ and /*Modify to here*/
  - Do not remove /*Modify from here*/ and /*Modify to here*/
  - TA will copy the space and evaluate your code.
  - You may add user-defined functions and header files between /*Modify from here*/ and /*Modify to here*/.

  - In the space, you need to implement the following functions. (Next page)
    - void CreateGraph(Graph *pgraph, int num, char data[]);
    - void DestroyGraph(Graph *pgraph);
    - void AddEdge(Graph *pgraph, int src, int dest);
    - void PrintGraph(Graph *pgraph); // for your validation, you don't need to implement
    - void DFS(Graph *pgraph);
    - void BFS(Graph *pgraph);

Evaluation

- Evaluation
  - TA will test several cases by changing the main function.
  - Read Pages 7~9 (regarding template) carefully.
  - For each test case,
    - If your C code results in an answer within 10 seconds on a platform with average computing power,
      - If your output is perfect for both DFS and BFS,
        - You get 100%.
      - Else if your output is correct for either one of DBF and BFS,
        - You get 50%
      - Else,
        - You get 0%.
    - Else,
      - You get 0%.