Lab 13 & HW 13

Data Structure

Lab13 (due on the Lab Session)

1. Do p13_1.c

HW13 (due on the day before the next Lab Session)

1. Do p13_2.c

Evaluation criteria

Category	Evaluation	
p13_1	50	
p13_2	50	
Total	100	

- *Use GCC 4.8 version or GCC 5.4 version.*
- *No score will be given if the gcc version is different.*

Lab13 – DFS & BFS

- You should finish p13_1 (BFS) during the lab session and submit it on **portal site (assignment)** before you leave.
- For p13_2 you have to submit on **portal site (assignment)**
- code name: p13_1, p13_2
- No score, if the code names are wrong.
- No score, if it does not use FILE I/O
- Each code will be tested by 5 different input files.
- 10 score for each input, if you don't get the answer you get 0 score.

Lab13 – DFS & BFS

- **graph makeGraph(FILE* fi)** Create a graph with nodes and edges.
- **void DFS_recursive(graph g, int start, int end)** depth_first search by recursive approach.
- **void DFS_iterative(graph g, int start, int end)** depth_first search by iterative using stack.
- **void BFS (graph g, int start, int end)** breadth_first search by iterative using queue.

Lab13 – DFS & BFS

- **stack* create_stack(int num)** Create a stack
- **void push(stack* s, int value)** push a new element at the end of the element in the stack. If you stack is full, just print an error message.
- **int pop(stack* s)** pop the element in the end of the stack. If stack does not have any element, just print an error message.
- **void close_stack(stack* s)** free all the memory allocated to stack.
- **queue* create_queue(int size)** create a new queue with the size.
- **void enqueue(queue* q, int value)** a new element at the end of the element in the queue. If you queue is full, just print an error message.
- **int dequeue(queue* q)** the node in the front. If your list does not have any element, just print an error message.
- **void close_queue(queue* q)** free all the memory allocated to queue.

Lab13 – DFS & BFS

- Structure

```
typedef struct Graph{
            int  num;
      int** weight;
      int* check_visit;
}graph;
```

```
typedef struct Stack{
            int∗  content;
            int  top;
      int max_stack_size;
}stack;
```

```
typedef struct CircularQueue{
            int∗  content;
            int  first;
            int  rear;
            int  qsize;
      int max_queue_size;
}queue;
```

Lab13. DFS & BFS

```
void main(int argc, char* argv[])
{
        FILE *fi = fopen(argv[1], "r");
        graph g = makeGraph(fi);

      int start, end;
      fscanf(fi,"%d-%d",&start,&end);

      printf("DFS recursive : ");
        DFS_recursive(g, start, end);
        if(g.num < end)
                printf("cannot find");
        else if(g.check_visit[end-1] == 0)
                printf("cannot find");
        printf("\nDFS iterative : ");
        DFS_iterative(g, start, end);

        printf("\nBFS : ");
        BFS_search(g, start, end);
             printf("\n");
/**
        free all the memory
      close the file
        **/

}
```

Lab13. DFS & BFS - BFS

- program name : p13_1.c
- input : an input file name is given as a command line argument. See the example.
- output : the visit path of BFS in the standard output

Lab13. DFS & BFS - BFS

- input file : Lab13_input1.txt

```
7
1-2  1-4  2-5  2-4  3-1  3-6  4-3  4-6  4-7  4-5  5-7  7-6
1-6
```

- Result

```
BFS : 1 2 4 5 3 6
```

Lab13. DFS & BFS - BFS

- input file : Lab13_input2.txt

```
7
1-2  1-4  2-5  2-4  3-1  3-6  4-3  4-6  4-7  4-5  5-7  7-6
1-10
```

- Result

```
BFS : 1 2 4 5 3 6 7 cannot find
```

Lab13. DFS & BFS – BFS, DFS_recursive, DFS_iterative

- program name : p13_2.c
- input : an input file name is given as a command line argument. See the example.
- output : the visit path of all version in the standard output

Lab13. DFS & BFS – BFS, DFS_recursive, DFS_iterative

- input file : Lab13_input1.txt

```
7
1-2  1-4  2-5  2-4  3-1  3-6  4-3  4-6  4-7  4-5  5-7  7-6
1-6
```

- Result

```
DFS recursive : 1 2 4 3 6
DFS iterative : 1 4 7 6
BFS : 1 2 4 5 3 6
```

Lab13. DFS & BFS – BFS, DFS_recursive, DFS_iterative

- input file : Lab13_input2.txt

```
7
1-2  1-4  2-5  2-4  3-1  3-6  4-3  4-6  4-7  4-5  5-7  7-6
1-10
```

- Result

```
DFS recursive : 1 2 4 3 6 5 7 cannot find
DFS iterative : 1 4 7 6 5 3 2 cannot find
BFS : 1 2 4 5 3 6 7 cannot find
```