과목명	자료구조 및 알고리즘	분반	x	담당교수	김화성 교수님
학과	전자통신공학과	학번	2016707079	이름	하상천
H/W 4: 8-Queens Problem					

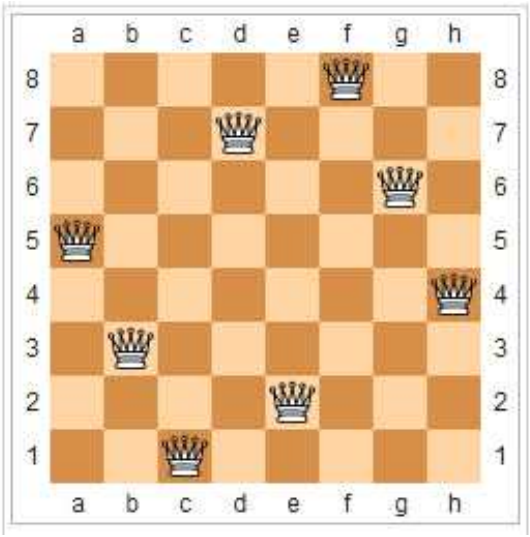
1, 과제설명 (과제에 대한 설명 및 목표)

인터넷 검색을 통해 8-queens problem이 무엇인지 이해하고,
이문제의 solution을 찾는 프로그램을 구현하라.

(solution은 1개만 있는 것이 아님. 가능한 모든 solution을 찾기 바랍니다.)

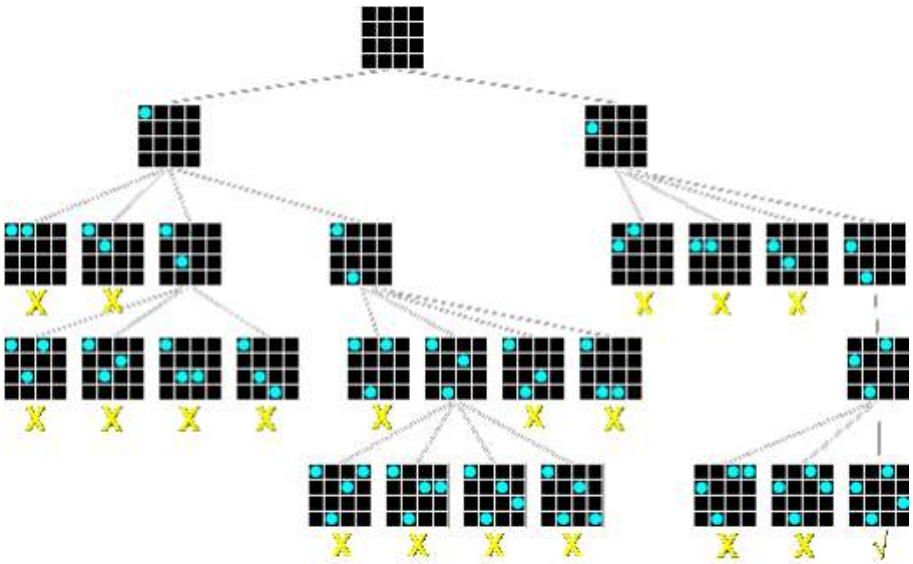
구현은 C언어를 사용하여 구현함.

2, 이론 (과제와 관련한 일반적인 내용)



어떠한 행, 열, 대각선에 퀸이 겹치지 않도록 놓는 방법의 수

3. 알고리즘 및 자료구조 설계 내용 기술



백트래킹이란 어떤 노드의 유망성을 점검하고 유망하지 않으면 그 노드의 부모 노드로 돌아간 후 다른 자손의 노드를 검색하는 것을 말한다. 즉, 스택에 자식노드를 넣기 전에 유망한지(해답이 될 가능성이 있는지) 확인하고 스택에 넣음을 말한다.

4. 소스코드 설명 (직접 작성한 소스코드중에 핵심 부분을 발췌하여 설명)

```
#define MAXSIZE 8 //최대 값
#define true 1 //보기 쉽게 true, false를 1, 0 으로 정의
#define false 0
typedef int boolean;  //가독성을 높이기 위해 int 자료형을 boolean 자료형으로 쓰겠다.
int count = 0;// 전역변수
int matrix[MAXSIZE];  //전역변수
```

가독성을 높이기 위해 true를 1로 정의하고, false를 0으로 정의했다. 또한 int 자료형을 boolean 자료형으로 쓰도록 치환했고, MAXSIZE를 8로 정의하여 count와 matrix[MAXSIZE] 전역변수를 만들었다.

```
while (k < num && flag)
        {       //같은 열이거나 대각선이면 배치 못한다.
                if (matrix[num] == matrix[k] || abs(matrix[num] - matrix[k]) == num - k)
{
                        flag = false;
                }
                k++;
        }
```

k를 1씩 증가시키면서 같은 열이거나 대각선이면 퀸을 배치하지 못하도록 한다. 만약 조건문에 일치하면 flag를 false로 바꾸어주면서 while문을 탈출한다. break문을 통해 반복문을 탈출해도 되지만, 리턴값으로 true, false를 리턴해줘야 하기 때문에 그냥 flag를 false로 바꾸어주면서 반복문을 탈출했다.

```
void put_queen(int num) {
	if (check(num)) {
		if (num == (MAXSIZE-1)) {  //판이 완성 되었으니까 count 1증가
			count++;
		}
		else {
			for (int j = 0; j < MAXSIZE; j++) {
				matrix[num + 1] = j;  //반복문을 통해 각 열에 하나씩 배치, 배열의 인덱스가 행이고 값이 열이다.
				put_queen(num + 1); // 재귀 함수와 num+1을 통해 위에서 아래의 행으로 내려간다.
			}
		}
	}
}
```

check(num)이 false(값으로는 0)이면 함수가 바로 종료되고, true(값으로는 1)이면 다시 조건문으로 들어간다. 만약 마지막 행이면 count++을 통해서 count를 1증가 시킨다. 반복문을 통해 각 열에 하나씩 배치하고, matrix 배열의 인덱스가 행이고, 값이 열이다. put_queen(num + 1)에서는 재귀함수와 num + 1을 통해 위에서 아래의 행으로 내려간다.

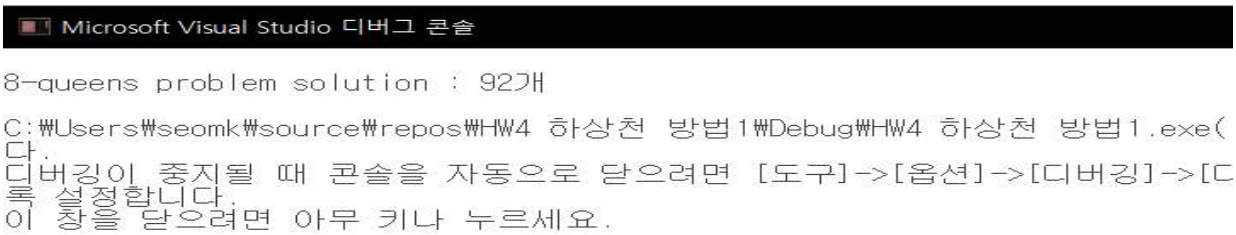
```
boolean check(Stack *pStack) {    //배치 가능한지 검사
	int k = 0;
	boolean flag = true;
	while (k < pStack->top && flag)
	{	//같은 열이거나 대각선이면 배치 못한다.
		if (pStack->stack[pStack->top] == pStack->stack[k] || abs(pStack->stack[pStack->top] - pStack->stack[k]) == (pStack->top) - k) {
			flag = false;
		}
		k++;
	}
	if (flag == true) {
		return flag;
	}
	else if (flag == false) {
		pStack->top--;  //조건에 맞지 않은 경우 다시 top에서 1을 빼준다.
		return flag;
	}
}
```

top을 인덱스라고 생각하고 stack에 들어간 값을 열이라고 생각한다. 같은 열이거나 대각선이면 배치를 못하도록 flag 를 false로 바꾸어주고, 다시 top을 -1해준다. 왜냐하면 stack에 값을 넣어줄 때, 즉 push 함수를 사용할 때 top이 +1 되고 stack에 값이 입력되기 때문이다.

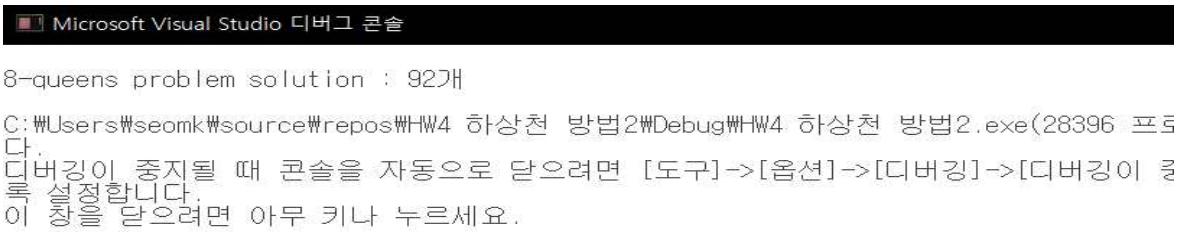
```
void put_queen(Stack *pStack) {
	if (check(pStack)) {
		if (pStack->top == (MAXSIZE-1)) {  //판이 완성 되었으니까 count 1 증가
			pStack->count++;
			pStack->top--; //top을 다시 1 빼준다.
		}
		else {
			for (int j = 0; j < MAXSIZE; j++) {
				push(pStack, j);  //반복문을 통해 각 행에서 퀸이 놓여질 열 값을 stack에 저장한다.
				put_queen(pStack); // 재귀 함수와 push함수에서의 top 증가를 통해 위에서 아래의 행으로 내려간다.
				if (j == (MAXSIZE -1)) {  //j가 7까지 입력된 이후에도 규칙에 어긋난다면 다시 top에서 1을 빼준다.
					pStack->top--;
				}
			}
		}
	}
}
```

top이 마지막이 되면 판이 완성되었으니까 count를 1 증가 시키고 다시 마지막 전 행으로 가야하니까 top--를 해준다. 반복문을 통해 각 행에서 퀸이 놓여질 열 값을 stack에 저장하고 재귀 함수와 push함수에서의 top증가를 통해 위에서 아래의 행으로 내려간다. 만약 j가 7까지 입력된 이후에도 규칙에 어긋난다면 다시 top--를 통해 전 행으로 돌아간다.

5. 실행결과 및 설명 (실행 결과를 캡쳐하여 첨부한 후 설명)



방법 1을 이용해서 실행한 결과화면입니다.



방법 2를 이용해서 실행한 결과화면입니다.

(그림을 문서에 포함, 글자처럼 취급 옵션, 잉크 절약과 잘 보이게 하기위해 그림 반전)

6. 고찰 (과제를 진행하면서 배운점 이나, 시행 착오 내용, 기타 느낀점)

이번 과제는 8-queens problem을 이해하고, 이 문제의 solution을 찾는 프로그램을 c언어로 구현하는 것이었다. 처음에는 어떻게 접근해야 할지 몰라서 노트에 경우의 수를 다 적어보며 생각해보았다. 결국 반복된다는 것을 알게 되었고, 재귀함수를 써야 된다고 생각하였다. 인터넷에 이 문제를 찾아보니 유명한 알고리즘 문제였다. 보통 백트래킹이라는 방법으로 접근을 많이 했는데, 백트래킹이란 어떤 노드의 유망성을 점검하고 유망하지 않으면 그 노드의 부모 노드로 돌아간 후 다른 자손의 노드를 검색하는 것이다. 다시 말하면 스택에 자식노드를 넣기 전에 해답이 될 가능성이 있는지 확인하고, 스택에 넣음을 의미한다. 이 알고리즘을 이용해서 c언어로 구현해보았더니 금방 해결이 되었다. 처음에는 전역변수로 배열을 만들고, count 변수를 만들어서 구현을 하였고, 두 번째 방법으로는 stack을 이용해서 구현해 보았다. 위의 소스코드 설명에도 설명하였듯이, check 함수를 통해 퀸이 같은 열과 대각선에 놓이지 않도록 체크하였고 배열의 인덱스를 행으로 값을 열로 해서 대입하였다. 두 번째 방법으로는 push함수를 통해서 0행부터 7행까지의 퀸의 열값을 stack에 저장하였다. 또한 top이 1씩 증가하거나 감소하면서 인덱스, 즉 행의 위치를 조절하였다. 보통 while(true){ }를 하고 조건문에서 break을 통해 반복문을 탈출하는데, 이번에는 flag 라는 것을 이용해서 반복문을 탈출했다. 또한 flag를 true, false로 보면 더 가독성이 좋다고 생각하여 1을 true로 0을 false로 치환하였다. 그리고 int 자료형을 boolean 자료형으로 쓸 수 있도록 치환하였다. 재귀함수를 이용하면 피보나치수열을 구하는 것과 같이 빅오표기법의 시간복잡도가 좋지 않기 때문에 재귀함수를 이용하지 않고 구현해보는 연습을 해보았지만 쉽게 되지 않았다. 이번 과제를 하면서 이런 유명한 알고리즘 문제를 처음 보았다는 것에 나 자신에게 조금 실망을 하였다. 앞으로는 과제뿐만 아니라 다른 알고리즘 문제들도 많이 풀어봐야겠다.

7. 전체 소스코드 (글자크기 9에 줄간격을 120%로 유지하고 한 줄이 너무 길지 않게 작성)

```
<방법 1>

#include <stdio.h>
#include <math.h> //abs 절댓값을 사용하기 위한 헤더파일
#define MAXSIZE 8 //최대 값
#define true 1 //보기 쉽게 true, false를 1, 0 으로 정의
#define false 0
```

```
typedef int boolean;  //가독성을 높이기 위해 int 자료형을 boolean 자료형으로 쓰겠다.
int count = 0;// 전역변수
int matrix[MAXSIZE];  //전역변수

boolean check(int num) { //배치 가능한지 검사
	int k = 0;
	boolean flag = true;
	while (k < num && flag)
	{	//같은 열이거나 대각선이면 배치 못한다.
		if (matrix[num] == matrix[k] || abs(matrix[num] - matrix[k]) == num - k)
{
			flag = false;
		}
		k++;
	}
	return flag;
}
void put_queen(int num) {
	if (check(num)) {
		if (num == (MAXSIZE-1)) {  //판이 완성 되었으니까 count 1증가
			count++;
		}
		else {
			for (int j = 0; j < MAXSIZE; j++) {
				matrix[num + 1] = j;  //반복문을 통해 각 열에 하나씩 배치,
배열의 인덱스가 행이고 값이 열이다.
				put_queen(num + 1); // 재귀 함수와 num+1을 통해 위에서 아
래의 행으로 내려간다.
			}
		}
	}
}
int main(void) {
	for (int i = 0; i < MAXSIZE; i++)
	{
		matrix[0] = i;
		put_queen(0);
	}
	printf("₩n8-queens problem solution : %d개₩n", count);
	return 0;
}
```

<방법 2>

```
#include <stdio.h>
#include <math.h>  //abs 절대값 사용하기 위한 헤더파일
#include<stdlib.h>
#define MAXSIZE 8 //최대 값
#define true 1 //보기 쉽게 true, false를 1, 0 으로 정의
#define false 0
typedef int boolean;  //가독성을 높이기 위해 int 자료형을 boolean 자료형으로 쓰겠다.
typedef int element;
typedef struct {
        element *stack;  //array to store elements
        int max_size;  //maximum size
        int top;    //stack top
        int count;
} Stack;

Stack* CreateStack(int size) {      //Stack 동적 메모리 할당, size 만큼 stack 동적 메모리할당
        Stack *pStack = (Stack*)malloc(sizeof(Stack));
        if (pStack == NULL)
        {
                return NULL;
        }
        pStack->stack = (element*)malloc((size) * sizeof(element));
        if (pStack->stack == NULL) {
                free(pStack);
                return NULL;
        }
        pStack->max_size = size;
        pStack->top = -1;
        pStack->count = 0;
        return pStack;
}

void push(Stack *pStack, element item) {
        if (pStack->top == pStack->max_size - 1) {
                printf("Stack is full");
                return;
        }
        pStack->stack[++pStack->top] = item;
}
element pop(Stack *pStack) {
        if (pStack->top < 0) {
```

```
                printf("Stack is empty");
                return 0;
        }
        return pStack->stack[pStack->top--];
}
element top(Stack *pStack) {
        if (pStack->top < 0) {
                printf("Stack is empty");
                return 0;
        }
        return pStack->stack[pStack->top];
}

void DestroyStack(Stack *pStack) {
        free(pStack->stack);
        free(pStack);
}

boolean check(Stack *pStack) {    //배치 가능한지 검사
        int k = 0;
        boolean flag = true;
        while (k < pStack->top && flag)
        {       //같은 열이거나 대각선이면 배치 못한다.
                if      (pStack->stack[pStack->top]      ==      pStack->stack[k]      ||
abs(pStack->stack[pStack->top] - pStack->stack[k]) == (pStack->top) - k) {
                        flag = false;
                }
                k++;
        }
        if (flag == true) {
                return flag;
        }
        else if (flag == false) {
                pStack->top--;  //조건에 맞지 않은 경우 다시 top에서 1을 빼준다.
                return flag;
        }
}
void put_queen(Stack *pStack) {
        if (check(pStack)) {
                if (pStack->top == (MAXSIZE-1)) {  //판이 완성 되었으니까 count 1 증가
                        pStack->count++;
                        pStack->top--; //top을 다시 1 빼준다.
                }
                else {
                        for (int j = 0; j < MAXSIZE; j++) {
```

```
                                push(pStack, j);  //반복문을 통해 각 행에서 퀸이 놓여질 열 값을 stack에 저장한다.
                                put_queen(pStack); // 재귀 함수와 push함수에서의 top 증가를 통해 위에서 아래의 행으로 내려간다.
                                if (j == (MAXSIZE -1)) {  //j가 7까지 입력된 이후에도 규칙에 어긋난다면 다시 top에서 1을 빼준다.
                                        pStack->top--;
                                }
                        }
                }
        }
}
int main(void) {
        Stack *pStack = CreateStack(8);
        for (int i = 0; i < MAXSIZE; i++,pStack->top = -1) //반복문이 한바퀴 돌면 top을 다시 -1로 초기화해준다.
        {
                push(pStack, i); //반복문을 통해 0번째 행에 퀸이 놓여질 열 값을 stack에 저장한다.
                put_queen(pStack);
        }
        printf("₩n8-queens problem solution : %d개₩n", pStack->count);
        DestroyStack(pStack);
}
```

(글자크기는 10으로 유지하고 줄간격도 160%를 유지할 것)