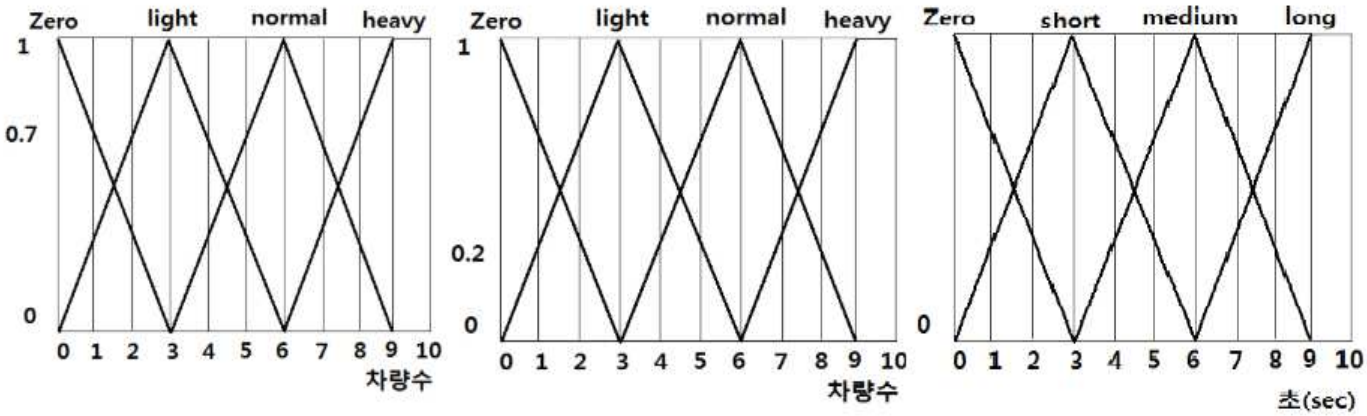
Fuzzy Theory-리포트

이름 : 박건호

학과 : 컴퓨터공학과

학번 : 201635964



목차

① 코드 흐름 설명

Step 0 : 변수 설정

```
//녹색 신호 진입 차량
int Azero = 0; // 0대
int Alight = 1; // 3대
int Anormal = 2; // 6대
int Aheavy = 3; // 9대

//적색 신호 대기 차량
int Bzero = 0; // 0대
int Blight = 1; // 3대
int Bnormal = 2; // 6대
int Bheavy = 3; // 9대

//녹색 신호 연장 시간
int Czero = 0; // 0초
int Cshort = 1; // 3초
int Cmedium = 2; // 6초
int Clong = 3; // 9초

//계산 규칙 적용 결과값
double result_arr[4][4][3];
```

Step 1 : 퍼지 규칙 정의

```
//진입 차량과 대기차량에 따른 녹색신호 확장시간 계산을 위한 규칙
static int arr[4][4] = {
    { Czero,Czero,Czero,Czero }, //Zero
    { Cshort, Czero,Czero,Czero }, //Light
    { Cmedium,Cshort,Czero,Czero }, //Normal
    { Clong,Cmedium,Cshort,Czero } //Heavy
};
```

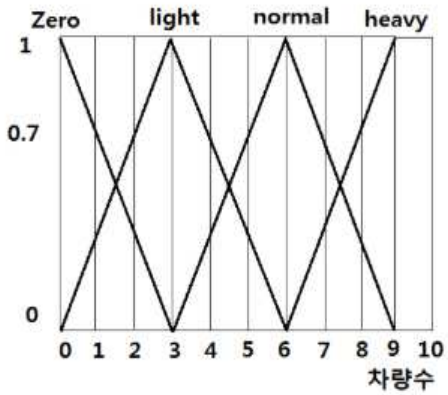
교차로에서 녹색 신호에 따라 진행하는 차량수가 적색 신호에 의해 대기 중인 차량수보다 적으면 녹색 신호를 더 이상 확장시키지 않고, 많은 경우에는 그 정도에 따라 확장을 시킨다. arr[4][4] 배열에서 진입 차량과 대기 차량에 따른 녹색 신호 연장시간을 나타내주고 이 규칙을 적용시킨 결과 값을 result_arr[4][4][3]에 저장 시킨다.

Step 2 : 삼각형 퍼지수를 이용하여 각각의 소속 함수 표현

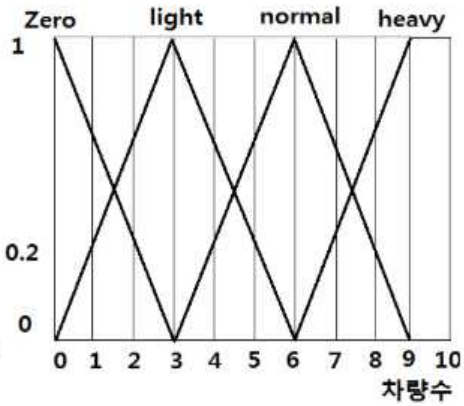
```
//삼각형 퍼지수 함수
double triangular_func(double r, double x) {
    r *= GRAPHSIZE; // Zero = 0, Light = 3, Normal = 6, Heavy = 9
    double p = r - GRAPHSIZE;
    double q = r + GRAPHSIZE;
    double result;

    if (p < x && x <= r) {
        result = (1 / (r - p)) * (x - r) + 1;
    }
    else if (r < x && x <= q) {
        result = -1 * (1 / (q - r)) * (x - r) + 1;
    }

    else { // x>=p or x<=p
        result = 0;
    }

    return ceil(result);
}
```

모든 삼각형퍼지 그래프의 형태는 같으므로 각각의 [변수*3 = r]이라는 규칙성을 이용해서

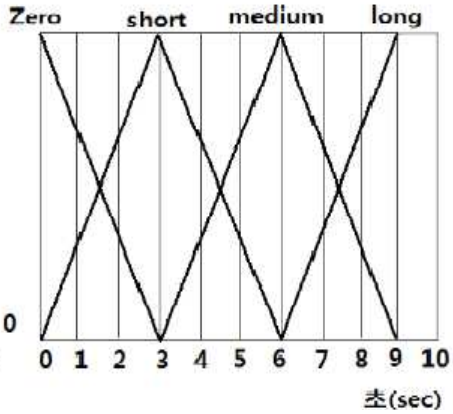
r *= GRAPHSIZE; 이라는 계산으로 처리하였다. (GRAPGHSIZE는 양옆의 x축길이에 해당한다.)



<녹색 신호 진입 차량>

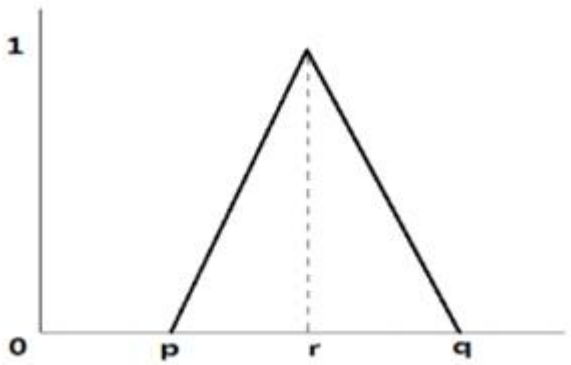


<적색 신호 대기 차량>



<녹색 신호 연장 시간>

진입중인 차량수가 7대라면 약간은 heavy하고 더 normal 한 정도를 나타내고, 대기중인 차량수가 5대라면 약간은 light하고 더 normal한 것을 나타낸다. 이 정도를 계산해서 확장시간을 구한 것이 최종 결과 값이다.



<삼각형 퍼지 수>

만약 녹색 신호 진입 차량이 light인 경우 r = 3, p = 0, q = 6이 된다.

Step 3 : apply_func 정의(triangular_func을 적용시키는 함수)

```
//삼각형 퍼지수 함수 적용 함수
void apply_func(int A, int B) {
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            for (int k = 0; k < 3; k++) {
                if (k == 0) {
                    result_arr[i][j][k] = triangular_func(i, A);
                }
                if (k == 1) {
                    result_arr[i][j][k] = triangular_func(j, B);
                }
                if (k == 2) {
                    result_arr[i][j][k] = arr[i][j] * ((double)3);
                }
            }
        }
    }
}
```

z가 0일 때 즉, 녹색 신호 진입 차량의 계산순서 일 때 삼각형퍼지수 계산을 이용해 triangular_func를 호출하면서 인자로 for문의 i값과 녹색 신호 진입 차량 수를 넘겨 준다

z가 1일 때 즉, 적색 신호 대기 차량의 계산순서 일 때 삼각형퍼지수 계산을 이용해 triangular_func를 호출하면서

인자로 for문의 j값과 적색 신호 대기 차량 수를 넘겨 준다
z가 2일 때 즉, 녹색 신호 연장 시간의 계산순서 일 때 교통제어 규칙인 arr[i][j]에 *3을 하여 연장 시간을 계산하여 넘겨준다.

Step 4 : 무게중심법(center_gravity)

```
//무게 중심법
double center_gravity(){
    double extension;
    double temp;
    double numerator = 0; //분자부분
    double denominator = 0; //분모부분

    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            temp = min(result_arr[i][j][0], result_arr[i][j][1]);
            numerator += temp * result_arr[i][j][2];
            denominator += temp;
        }
    }

    extension = numerator / denominator;

    return extension;
}
```

$$\text{Extension} = \frac{u[i] * u(i, \text{Extension})}{u[i]}$$

$$u[i] = \sum_{i=1}^{16} \min(\text{Ri-Arrival}, \text{Ri-Queue})$$

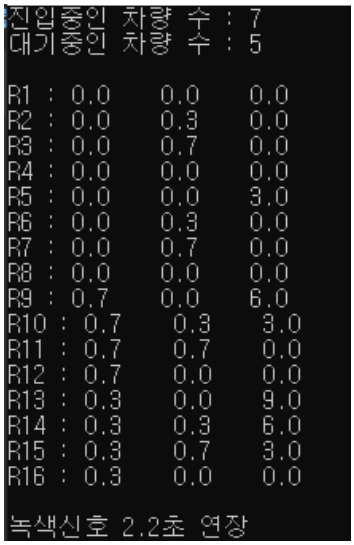
u(i, Extension) = the extension_unit of i-th Rule

Step 5 : 결과 출력 함수

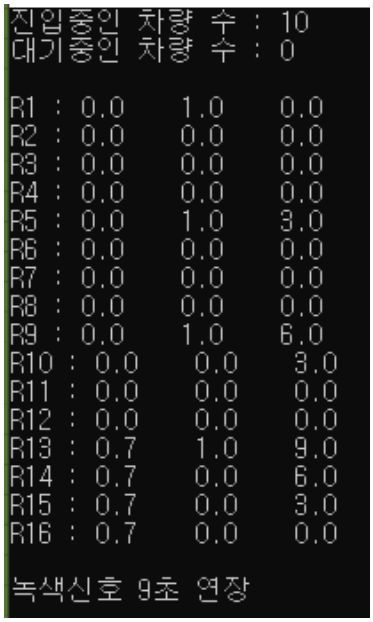
```
//진행 결과 출력 함수
void print_result(int A, int B) {
    int cnt = 0;
    apply_func(A, B);

    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            cnt++;
            printf("R%d : ", cnt);
            for (int k = 0; k < 3; k++) {
                printf("%.1f    ", result_arr[i][j][k]);
            }
            cout << endl;
        }
    }
}

//소숫점 1자리까지 반올림
double ceil(double x) {
    x = round(x * 10) / ((double)10);
    return x;
}

//메인 함수
int main() {
    double result;

    int go, wait;
    cout << "진입중인 차량 수 : ";
    cin >> go;
    cout << "대기중인 차량 수 : ";
    cin >> wait;
    cout << endl;

    print_result(go, wait);
    result = ceil(center_gravity());

    cout << endl;
    cout << "녹색신호 " << result << "초 연장" << endl;

    return 0;
}
```

② 실행 결과

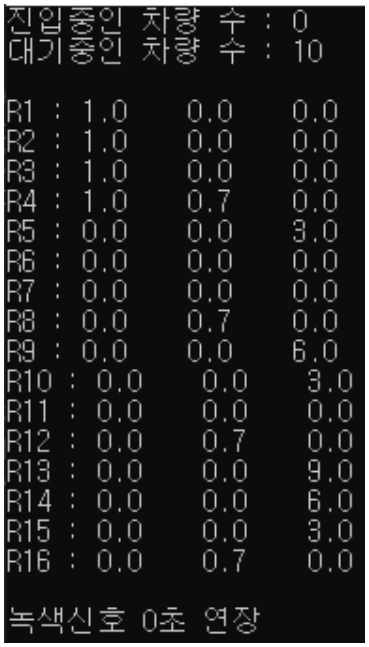
녹색 신호 진입 차량 : 7대, 적색 신호 대기 차량 : 5대 일 경우



녹색 신호 진입 차량 : 10대, 적색 신호 대기 차량 : 0대 일 경우



녹색 신호 진입 차량 : 0대, 적색 신호 대기 차량 : 10대 일 경우



③ 전체 코드

```
#include <iostream>
#include <stdio.h>
#include <algorithm>
#include <math.h>

#define GRAPHSIZE 3 //그래프 양옆 사이즈

using namespace std;

//녹색 신호 진입 차량
int Azero = 0; // 0대
int Alight = 1; // 3대
int Anormal = 2; // 6대
int Aheavy = 3; // 9대

//적색 신호 대기 차량
int Bzero = 0; // 0대
int Blight = 1; // 3대
int Bnormal = 2; // 6대
int Bheavy = 3; // 9대

//녹색 신호 연장 시간
int Czero = 0; // 0초
int Cshort = 1; // 3초
int Cmedium = 2; // 6초
int Clong = 3; // 9초

//계산 규칙 적용 결과값
double result_arr[4][4][3];

//진입 차량과 대기차량에 따른 녹색신호 확장시간 계산을 위한 규칙
static int arr[4][4] = {
        { Czero,Czero,Czero,Czero }, //Zero
        { Cshort, Czero,Czero,Czero }, //Light
        { Cmedium,Cshort,Czero,Czero }, //Normal
        { Clong,Cmedium,Cshort,Czero } //Heavy
};


//삼각형 퍼지수 함수
double triangular_func(double r, double x) {
        r *= GRAPHSIZE; // Zero = 0, Light = 3. Normal = 6, Heavy = 9
        double p = r - GRAPHSIZE;
        double q = r + GRAPHSIZE;
        double result;

        if (p < x && x <= r) {
                result = (1 / (r - p)) * (x - r) + 1;
        }
        else if (r < x && x <= q) {
                result = -1 * (1 / (q - r)) * (x - r) + 1;
        }

        else { // x>=p or x<=p
                result = 0;
        }

        return ceil(result);
}

//삼각형 퍼지수 함수 적용 함수
void apply_func(int A, int B) {
        for (int i = 0; i < 4; i++) {
                for (int j = 0; j < 4; j++) {
                        for (int k = 0; k < 3; k++) {
                                if (k == 0) {
                                        result_arr[i][j][k] = triangular_func(i, A);
                                }
                                if (k == 1) {
                                        result_arr[i][j][k] = triangular_func(j, B);
                                }
                                if (k == 2) {
                                        result_arr[i][j][k] = arr[i][j] * ((double)3);
                                }
                        }
                }
        }
}
```

```
//무게 중심법
double center_gravity(){
	double extension;
	double temp;
	double numerator = 0; //분자부분
	double denominator = 0; //분모부분

	for (int i = 0; i < 4; i++) {
		for (int j = 0; j < 4; j++) {
			temp = min(result_arr[i][j][0], result_arr[i][j][1]);
			numerator += temp * result_arr[i][j][2];
			denominator += temp;
		}
	}

	extension = numerator / denominator;

	return extension;
}

//진행 결과 출력 함수
void print_result(int A, int B) {
	int cnt = 0;
	apply_func(A, B);

	for (int i = 0; i < 4; i++) {
		for (int j = 0; j < 4; j++) {
			cnt++;
			printf("R%d : ", cnt);
			for (int k = 0; k < 3; k++) {
				printf("%.1f    ", result_arr[i][j][k]);
			}
			cout << endl;
		}
	}
}

//소숫점 1자리까지 반올림
double ceil(double x) {
	x = round(x * 10) / ((double)10);
	return x;
}

//메인 함수
int main() {
	double result;

	int go, wait;
	cout << "진입중인 차량 수 : ";
	cin >> go;
	cout << "대기중인 차량 수 : ";
	cin >> wait;
	cout << endl;

	print_result(go, wait);
	result = ceil(center_gravity());

	cout << endl;
	cout << "녹색신호 " << result << "초 연장" << endl;

	return 0;
}
```

④ 소감문

우리가 오늘 비가 온다라고 말하는 것은 결과적으로 비가오거나(숫자 1로 표현(TRUE)) 비가 오지 않아(숫자 0으로 표현(FALSE)) 참과 거짓이 명확해지는 Crisp Logic인데 비가 조금 온다라고 표현 하면 조금이라는 단어가 갖는 모호함 때문에 퍼지논리가 된다. 퍼지 이론을 공부하면서 오늘날 과학연구계는 물론 일상생활에 쓰이는 많은 제품을 만들 때도 많이 사용하고 있다. 세탁기를 운용할 때 세제량이나 물양을 조절하는 문제에서부터 자동차의 속도를 제어하는 알고리즘, 엘리베이터의 그룹 관리 등 광범위한 분야에 새로운 편의성을 제공하는 용도로 계속 이용되고 발전하고 있다는 것을 알게 되었다.

퍼지이론이 처음 나왔을 때에는 반응이 냉담하였다고한다. 확률에서는 일어날지도 모르는 불규칙성에 기초한 가능성으로 일어날 수 있는 모든 사건들의 확률합은 1이 되지만 퍼지이론에서 불확실성은 실제로 발생하는 사건에 대해서도 명확한 판단을 내릴 수 없는 명제를 대상으로 한다. 퍼지이론은 주관성에 바탕을 둔 학문이라고 봐도 무방하다고 생각하는데 인공지능이 발전하면서 더욱 더 사람의 생각처럼 만들고자 하면서 퍼지이론이 많이 활용되는 것 같다.

이번 퍼지이론은 지금까지 과제로 제출했었던 다층 퍼셉트론, 유전자 알고리즘에 비해 코드 구현 부분에 있어서는 가장 쉽고 빠른 시간안에 마칠 수 있었습니다. 단순 계산식만 구현하면 되었기 때문에 그랬던 것 같다. 한 학기 동안 다양한 것들을 직접 날코딩 해보는 수업은 처음 해본다. 물론 수업 외로 따로 공부를 더 해야 할 수 있는 것들이 많았지만 내가 직접 찾아보면서 궁금한 것을 알아내면서 문제를 풀어나가는 것에 매우 재미있었고 단순 암기식 아니라서 정말 내 실력이 향상된 것 같아 매우 유익하였다. 쉽지만은 않았지만 내가 한만큼 발전이 되는 수업이였고 이번 학기에서 가장 기억에 많이 남은 수업이였다.