과제2) 지하철 최단거리 보고서

자료구조

담당교수	이승수 교수님
작성일	2022. 05. 31.
학과	컴퓨터공학부
학번	201901594
이름	장근하

1. 프로젝트 개요

본 보고서는 출발역, 도착역으로 도달하는 최단거리와 최소환승을 계산해주는 예를 보여준다.
출발역과 도착역을 입력하면 최단거리 또는 최소환승을 선택해 계산하는 프로그램을 작성한다.
- 출발역에서 도착역까지 최단거리 또는 최소환승 이동 과정을 계산한다.

2. 설계

1) 프로그램명: SubwayCalc

2) 요구정의 및 분석

2-1) 요구정의 입·출력 정의

가. 입력은 무엇인가?

- 출발역, 도착역 입력
- 최단거리, 최소환승 여부 입력

나. 출력은 무엇인가?

- 최단거리과 최소환승중 선택한 경로
- 도달하는데 걸린 시간과, 지나친 정거장 수

다. 세부 기능 및 기능별 요구 조건

- 행렬 파일과 지하철 정보 읽기를 위한 파일 입출력 필요
- 구조체 인접행렬과 지하철 구조체 배열 구현
- 다익스트라 알고리즘 구현
- 경로 분석시 최소환승 기능 구현

2-2) 분 석

가. 종이와 연필로 계산한다고 가정하고 절차를 정리

1. 구조체 인접행렬, 지하철 정보 배열 구현
2. 파일 입출력으로 행렬, 지하철 정보 입력
3. 출발역, 도착역과 최단거리, 최소환승 여부 입력
   - 출발역과 도착역이 동일할시 반복
   - 출발역 또는 도착역이 역 목록에 존재하지 않을시 반복
   - 최단경로, 최소환승 이외의 번호가 입력되었을 시 반복
4. 출발역이 여러 개인지 (환승인지) 확인
   - 한 개일 경우 환승역이 아니므로 바로 다음 단계로
   - 여러 개일 경우 그중에서 가장 짧은 출발역을 다익스트라로 계산해서 선택
   - 최소환승일 경우 모든 환승역 가중치에 1000씩 더해서 최소환승 유도
5. 선택된 출발역을 시작점으로 다익스트라로 계산 후 출력
   - 최소환승일 경우 모든 환승역 가중치에 1000씩 더해서 최소환승 유도
   - 다익스트라로 계산하면서 이동 인덱스를 저장
   - 이동 인덱스를 바탕으로 거꾸로 추적하면서 조건에 따라 계산 및 출력

나. 문제 해결 과정을 정리

main		자 료 명	자 료 형	자 료 수	자 료 범 위	매 체
입 력	0	-	-	-	-	-
출 력	1	프로그램 종료 문자	char	-	-	화면
처리 절차 분석		1. 구조체 배열 구현 및 초기화 2. 역명 구조체 선언 및 초기화 3. 구조체 배열 및 역명 구조체 입력 함수 호출 4. sub_find 함수 호출 5. 구조체 배열 할당 해제 6. 프로그램 종료 문자 출력				

makeArray		자 료 명	자 료 형	자 료 수	자 료 범 위	매 체
입 력	0	-	-	-	-	-
출 력	1	구조체 행렬	element**	R*R	R*R	저장
처리 절차 분석		구조체 행렬을 이중 for문으로 동적 할당				

initArray		자 료 명	자 료 형	자 료 수	자 료 범 위	매 체
입 력	1	구조체 행렬	element**	R*R	R*R	전달
출 력	1	구조체 행렬	element**	R*R	R*R	저장
처리 절차 분석		구조체 행렬의 초기 값을 설정 초기 값(from="", to="", data=9999, ic=FALSE)				

killArray		자 료 명	자 료 형	자 료 수	자 료 범 위	매 체
입 력	1	구조체 행렬	element**	R*R	R*R	전달
출 력	1	할당 해제	-	-	-	저장
처리 절차 분석		구조체 행렬을 할당 해제				

readCSV		자 료 명	자 료 형	자 료 수	자 료 범 위	매 체
입 력	1	인덱스	int	0-19	0-19	전달
출 력	1	파일 내용	FILE*	-	-	저장
처리 절차 분석		i번째 인덱스의 파일명과 경로를 합치고 파일을 읽어서 파일의 내용을 리턴 ex) .data/1호선.csv				

readSubInfo		자 료 명	자 료 형	자 료 수	자 료 범 위	매 체
입 력	1	역명 구조체	sublist	0-551	0-551	전달
출 력	1	역명 구조체	sublist	0-551	0-551	저장
처리 절차 분석		역명 구조체에 순서대로 역명과 역코드를 삽입				

readSubArray		자 료 명	자 료 형	자 료 수	자 료 범 위	매 체
입 력	1	구조체 행렬	element**	R*R	R*R	전달
입 력	1	역명 구조체	sublist	0-551	0-551	전달
출 력	1	해당 구조체들에 저장	-	-	-	저장
처리 절차 분석		1. 모든 호선의 인덱스를 불러옴 2. 해당 호선의 인접행렬 가중치를 입력하고 다음 범위로 확장 - 입력하면서 역명 구조체에 I번째 호선 인덱스도 저장 3. 역명 구조체에서 출발지와 목적지 정보를 입력 4. 환승 정보를 불러오고 존재하는 역코드의 인덱스를 저장 5. 역코드의 인덱스를 바탕으로 인접행렬에 환승 가중치를 저장, 환승여부를 변경				

subChk		자 료 명	자 료 형	자 료 수	자 료 범 위	매 체
입 력	1	역명 구조체	sublist	0-551	0-551	전달
입 력	1	역명	char	0-99	0-99	전달
출 력	1	인덱스	int	0-551	0-551	저장
처리 절차 분석		해당 역명과 동일한 이름의 역 인덱스를 리턴				

choose		자 료 명	자 료 형	자 료 수	자 료 범 위	매 체
입 력	1	거리 배열	int	0-551	0-551	전달
입 력	1	범위	int	552	552	전달
입 력	1	방문 여부 배열	int	0-551	0-551	전달
출 력	1	인덱스	int	0-551	0-551	저장
처리 절차 분석		범위중 방문하지 않았고 거리가 가장 작은 인덱스를 리턴				

shortest_path		자 료 명	자 료 형	자 료 수	자 료 범 위	매 체
입 력	1	구조체 행렬	element**	R*R	R*R	전달
입 력	1	시작 인덱스	int	0-551	0-551	전달
출 력	1	path에 인덱스 저장	-	-	-	저장
처리 절차 분석		1. 거리 배열, 방문여부 배열, 이동기록 배열 초기화 2. 최소환승 옵션일 경우 한번에 한해 모든 환승-환승역에 가중치 부여 3. 거리 배열과 이동기록 배열의 시작지점을 설정 4. 가장 거리가 작은 인덱스를 호출 (choose) 5. 해당 인덱스의 방문 여부를 방문으로 변경 6. 방문한적 없는 모든 인덱스들 중에서 기존 거리 배열보다 최단거리가 있는지 확인 7. 최단거리가 있다면 해당 인덱스를 이동 기록에 저장하고 거리 배열을 갱신				

print_path		자 료 명	자 료 형	자 료 수	자 료 범 위	매 체
입 력	1	구조체 행렬	element**	R*R	R*R	전달
입 력	1	역명 구조체	sublist	0-551	0-551	전달
입 력	2	출발, 도착 인덱스	int	0-551	0-551	전달
출 력	1	이동 경로 및 소요시간, 정거장 수 출력				화면
처리 절차 분석		1. 이동 시간, 환승 시간, 정거장 수 초기화 2. 이동 기록을 거꾸로 임시 배열에 저장 3. 임시 배열을 읽어서 이동 경로를 읽으면서 추적 - 이동 중 환승 여부를 체크하고 환승이 아닐 경우 a. 시간과 정거장 카운트는 그대로 갱신 및 증가 b. 다음역이 환승역이 아닐 경우 경로 출력 - 이동 중 환승 여부를 체크하고 환승이 일경우 a. 다음역이 환승역이 아닐 경우 환승 시간 갱신, 환승 출력 b. 최소환승일 경우 추가되었던 가중치를 빼고 계산 4. 최소환승, 최단거리 여부와 소요시간과 정거장 수 출력				

calc_path		자 료 명	자 료 형	자 료 수	자 료 범 위	매 체
입 력	1	구조체 행렬	element**	R*R	R*R	전달
입 력	1	역명 구조체	sublist	0-551	0-551	전달
입 력	2	출발, 도착 인덱스	int	0-551	0-551	전달
출 력	1	이동 시간+환승 시간	int	-	-	저장
처리 절차 분석		기본적으로 print_path와 동일하나, 화면의 출력은 없고 이동 시간+환승 시간만 출력				

sub_find		자 료 명	자 료 형	자 료 수	자 료 범 위	매 체
입 력	1	구조체 행렬	element**	R*R	R*R	전달
입 력	1	역명 구조체	sublist	0-551	0-551	전달
출 력	1	프로그램 종료 출력	-	-	-	화면
처리 절차 분석		1. 출발역과 도착역을 입력 - 서로 동일할시 1번으로 반복 - 역명 구조체에 존재하지 않을시 1번으로 반복 2. 최단거리, 최소환승 여부를 입력 - 1 또는 2가 아닐시 2번으로 반복 3. 출발역이 여러 개인지 (환승인지) 확인 - 한 개일 경우 환승역이 아니므로 바로 다음 단계로 - 여러 개일 경우 그중에서 가장 짧은 출발역을 다익스트라로 계산해서 선택 (calc_path 사용) 4. 선택된 출발역을 시작점으로 다익스트라로 계산 후 출력(print_path)				

3) 입·출력 설계

가. 입 력

(1) 입력방법(매체)는 어떻게 하겠는가? → 화면

(2) 입력에 필요한 변수는 무엇이 필요한가 ? → sub1, sub2, Chk_num

(3) 입력에 필요한 자료형은 어떻게 하겠는가? → char

나. 출 력

(1) 출력방법(매체)는 어떻게 하겠는가? → 화면

(2) 출력에 필요한 변수는 무엇이 필요한가 ?→ csvLists, subinfo, Sub_time, IC_time 등

(3) 출력에 필요한 자료형은 어떻게 하겠는가? → char, int

1. 입·출력

2. 매 체 : 화면

	10	20	30	40	50	60	70	80	90
	출발역을 입력해주세요: xxx								
	도착역을 입력해주세요: xxxx								
	방식? 1. 최단경로 2. 최소환승								
	: x								
	<출발>								
	-><x호선> xxx								
1	-><x호선> xxx								
2	-><x호선> xxxx								
3	-><x호선> xxx								
4	-><x호선> xxx								
5	-><환승 : 소요시간 x 분> xx								
6	-><x호선> xxx								
7	-><x호선> xxxxx								
8	-><x호선> xx								
9	.....								
10	-><x호선> xxxxx								
11	-><x호선> xxxxxx								
12	-><x호선> xxxx								
	x로 도착								
	소요시간 : x (x + 환승 소요시간 x) 분								
	정거장 수 : x 개								

3. 구현

1) 프로그램 설계 (각 함수별 기능)

함수명		main			
		자료명	자료형	변수명	매체
입 력	0	-	-	-	전 달
출 력	1	-	int	-	화 면
작업변수	2	구조체 행렬, 지하철 구조체	element**, sublist	subarray, subinfo	저 장

처리 과정 코딩		{ element** subarray= makeArray();
		initArray(subarray);
		
		sublist subinfo[R];
		readSubInfo(subinfo);
		readSubArray(subarray,subinfo);
		
		debug_print(subarray,subinfo,0);
		sub_find(subarray,subinfo);
		
		killArray(subarray); //인접행렬 free
		printf("프로그램이 종료되었습니다 ₩n");
		
		return 0;
		}			

함수명		makeArray			
		자료명	자료형	변수명	매체
입 력	0	-	-	-	-
출 력	1	구조체 행렬	element **	arr	저 장

처리 과정 코딩		{
		element **arr = malloc(sizeof(element *) * R); // 이중 포인터에 (element 포인터 크기 * row)만큼 동적 메모리 할당. 배열의 세로
		for (int i = 0; i < R; i++){ // 세로 크기만큼 반복
		arr[i] = malloc(sizeof(element) * C); // (element의 크기 * col)만큼 동적 메모리 할당. 배열의 가로
		}
		return arr;
		}			

함수명		initArray			
		자료명	자료형	변수명	매체
입 력	1	구조체 행렬	element **	arr	전 달
출 력	1	구조체 행렬	element **	arr	저 장

처리 과정 코딩		{for (int i = 0; i < R; i++){
		for (int j = 0; j < C; j++){
		strcpy(arr[i][j].from,""); // 출발지 비움
		strcpy(arr[i][j].to,""); // 목적지 비움
		arr[i][j].data=9999; // 가중치 9999로 초기화
		arr[i][j].ic=FALSE; // 환승 X로 초기화
		}
		}
		}			

함수명		killArray			
		자료명	자료형	변수명	매체
입 력	1	구조체 행렬	element**	arr	전 달
출 력	0	-	-	-	저 장
작업변수	1	for문 매개변수	int	i	생 성

처리 과정 코딩		{ for (int i = 0; i < R; i++) {
		free(arr[i]);
		}
		free(arr);}			

함수명		readCSV			
		자료명	자료형	변수명	매체
입 력	1	파일 인덱스	int	i	전 달
출 력	1	파일 데이터	File*	stream	저 장
작업변수	1	파일 경로	char	temp	생 성

처리 과정 코딩		{ char fileDir[50] = "./data/"; // 하위 경로
		strcat(fileDir, csvLists[i]); // i번째 파일명을 하위 경로에 붙임
		strcat(fileDir, ".csv"); // .csv를 하위 경로에 붙임
		FILE *stream = fopen(fileDir, "r"); // 하위 경로의 파일을 불러옴
		if (stream != NULL){ //파일이 비어있지 않으면
		return stream;
		}
		else{ //파일이 비어있거나 문제가 있다면
		printf("Failed to load %s !!!", fileDir); //오류 출력
		exit(1); //종료
		}}			

함수명		subChk			
		자료명	자료형	변수명	매체
입 력	2	역명 구조체, 역명	sublist, char	subinfo[], chk[]	전 달
출 력	1	동일 인덱스	int	i	저 장

처리 과정 코딩		{ for (int i=0;i<R;i++){
		if (strcmp(chk,subinfo[i].name)==0) //해당하는 문자열을 찾으면
		return i; //해당 인덱스 출력
		}
		return -1; //목록에 없으면 -1 출력
		}			

함수명		choose			
		자료명	자료형	변수명	매체
입 력	3	거리, 범위, 방문여부	int	distance[], n, found[]	전 달
출 력	1	최소 인덱스	int	minpos	저 장
작업변수	1	for문 매개변수	int	i	생 성
작업변수	1	최소 거리값	int	min	생 성

처리 과정 코딩		{ int i, min, minpos;
		min = INT_MAX; // 최대로 설정
		minpos = -1;
		for (i = 0; i<n; i++)
		if (distance[i]< min && !found[i]) { //방문하지 않았고 최소보다 더 작으면
		min = distance[i]; // 거리 입력
		minpos = i; // 인덱스 입력
		}
		return minpos;}			

함수명		readSubInfo			
		자료명	자료형	변수명	매체
입 력	1	역명 구조체	sublist	subinfo[]	전 달
출 력	1	역명 구조체	sublist	subinfo[]	저 장

처리 과정 코딩		{ char buf[2048];
		FILE *stream = readCSV(18);
		char *line, *tmp;
		line = fgets(buf, 2048, stream); // 한줄 미리 읽음 (안쓰는 칸 거르기)
		int idx=0;
		while ((line = fgets(buf, 2048, stream))!=NULL) {
		int i=0;
		tmp = strtok(buf, ","); //
		while (tmp != NULL){
		if (i == 0){
		strcpy(subinfo[idx].code, tmp);
		}
		else if (i == 1){
		if(idx!=R) // 마지막줄이 아니면
		tmp[strlen(tmp) - 1] = '₩0'; // ₩n을 제거해준다.
		strcpy(subinfo[idx].name, tmp);
		}
		i++;
		tmp = strtok(NULL, "");
		}
		idx++;
		}
		
		fclose(stream);
		}			

함수명		shortest_path			
		자료명	자료형	변수명	매체
입 력	1		element**	arr	전 달
입 력	1	시작점	int	start	
출 력	1	경로	int	path	저 장
작업변수	1	for문 매개변수	int	i	생 성
작업변수	1	for문 매개변수	int	j	생 성
작업변수	1	임시 정점	int	u, w	생 성

처리 과정 코딩		{
		int i, u, w;
		static int trans_done=0; // 정적변수
		for (i = 0; i<R; i++){ /* 초기화 */
		distance[i] = arr[start][i].data;
		found[i] = FALSE;
		path[i] = start;
		}
		path[start] = -1;
		if(option==2&&trans_done==0){ //최소환승 옵션일 경우
		for (i = 0; i<R; i++){
		for (int j = 0; j<R; j++){
		if(arr[i][j].ic==1)// 환승역이면
		arr[i][j].data+=1000; //가중치를 크게 증가시킴			

```
                    }
                }
            trans_done=1;
    }

            found[start] = TRUE;    /* 시작 정점 방문 표시 */
            distance[start] = 0;
            for (i = 0; i<R-1; i++) {
                    u = choose(distance, R, found);
                    found[u] = TRUE; // 해당 정점을 방문 표시
                    for (w = 0; w<R; w++)
                            if (!found[w]) // 방문한적이 없다면
                                    if (distance[u] + arr[u][w].data < distance[w]){
                                            path[w]=u; //인덱스 저장
                                            distance[w] = distance[u] +
arr[u][w].data;
                                    }
            }
}
```

함수명		readSubArray			
		자료명	자료형	변수명	매체
입 력	1	구조체 행렬	element**,	arr	전 달
입 력	1	역명 구조체	sublist	subinfo[]	전 달
출 력	2	구조체 행렬과 역명 구조체에 저장됨			저 장
작업변수	7	반복문 매개변수	int	i, j, m, n, tmpCnt, tmpR, tmpC	생 성
작업변수	1	임시 정수 배열	int	tmpIC[]	생 성
작업변수	1	임시 문자열	char *	line, tmp	생 성

처리 과정 코딩

```
{   int cRow=0; // 현재 행열 값을 저장하는 벼수
    char *tmp, *line; //줄과 단어 변수 선언
    char buf[2048]; // 버퍼 선언
    srand(time(NULL)); //완전 랜덤화

    //인접행렬 정보 입력
    for (int i = 0; i < 18; i++){
        FILE *stream = readCSV(i);
        char *line, *tmp;
        line = fgets(buf, 2048, stream); // 한줄 미리 읽음 (안쓰는 칸 거르기)
        int tmpR = 0, tmpC = 0; //임시 행 열
        while ((line = fgets(buf, 2048, stream)) != NULL){ //한줄씩 읽음
            tmpC = 0; //임시 열을 0으로 초기화
            tmp = strtok(line, ","); // ,로 나눠진 line의 첫 단어를 읽음
            tmp = strtok(NULL, ","); //다음 단어를 읽음 ( 첫번째 역 코드 뛰어넘기 )
            while (tmp != NULL){ // 읽은 단어가 NULL이 아니면
                subinfo[cRow + tmpR].num=i; // ~호선 인덱스를 역명 구조체에 저장
                arr[cRow + tmpR][cRow + tmpC].data = atoi(tmp); // 구조체 행렬에 cRow
+ tmpR,C의 인덱스에 해당 값 저장
                tmp = strtok(NULL, ","); //다음 단어를 읽음
                tmpC++; // 임시 열 추가
            }
            tmpR++; // 임시행 추가
```

```
        }
        cRow = cRow + tmpR; // 현재 열에 임시 행만큼 더해줘서 저장
        fclose(stream); //파일 닫기
    }

    //여기서부터는 구조체의 from to 저장
    for (int i = 0; i < R; i++){
        for (int j = 0; j < R; j++){
            strcpy(arr[i][j].from,subinfo[i].code); //arr의 i행에 출발 역코드 저장
            strcpy(arr[i][j].to,subinfo[j].code); //arr의 j행에 도착 역코드 저장
        }
    }

    //여기서부터는 환승하는 코드의 인덱스 저장
    FILE *stream = readCSV(19); // 환승파일 읽기
    int tmpCnt = 0;
    int tmpIC[IC];
    line = fgets(buf, 2048, stream);
    tmp = strtok(line, ",");
    //tmp = strtok(NULL, ","); //환승 단어 스킵
    while (tmp != NULL){
        if(tmpCnt==IC-1) // 마지막 단어면
                tmp[strlen(tmp) - 1] = '₩0'; // ₩n을 제거해준다.
        for (int i = 0; i < R; i++){
            if (strcmp(tmp, subinfo[i].code) == 0){ // 역 코드가 동일하다면
                tmpIC[tmpCnt] = i;// 인덱스 저장
                //printf("%d) %s %d 인덱스 [%s]₩n", tmpCnt,tmp,i,subinfo[i].name);
                tmpCnt++; // 다음
            }
        }
        tmp = strtok(NULL, ",");//다음 읽기
    }

    //여기서부터는 환승정보를 인접행렬에 저장
    tmpCnt=0;
    int i=0;
    int j=0;
    int m;
    int n;
    while ((line = fgets(buf, 2048, stream)) != NULL){ //다음 줄 읽기
        j=0;
        tmp = strtok(line, ",");
        tmp = strtok(NULL, ",");
        while (tmp != NULL){
            if(j==IC-1) // 마지막 단어면
                tmp[strlen(tmp) - 1] = '₩0'; // ₩n을 제거해준다.
            m = tmpIC[i]; // 행 인덱스 불러오기
            n = tmpIC[j]; // 열 인덱스 불러오기
            if (atoi(tmp) != 9999&&atoi(tmp) != 0){ // 해당 가중치가 방문할 수 있는 값이면
                //printf("%d(%s), ",n,subinfo[n].name);
                arr[m][n].data = (rand() % atoi(tmp)) + 1; // 1 ~ 해당 값까지 랜덤 난수로
```

```
저장
                    arr[m][n].ic = TRUE; // 해당 지점은 환승이므로 TRUE
                }
                if (j<IC)
                    j++; // 다음 열
                tmp = strtok(NULL, ",");
            }
            //printf("[%d:%s]₩n",m,subinfo[m].name);
            if (i<IC-2)
                i++; // 다음 행
        }
        fclose(stream);
}
```

함수명		print_path			
		자료명	자료형	변수명	매체
입 력	1	구조체 행렬	element**,	arr	전 달
입 력	1	역명 구조체	sublist	subinfo[]	전 달
입 력	2	시작점, 끝점	int	start, end	전 달
출 력	1	이동 경로, 소요시간, 정거장 수 출력			화 면
작업변수	4	반복문 매개변수	int	k, q, i, limit	생 성
작업변수	1	임시 정수 배열	int	way[]	생 성

처리 과정 코딩

```
{   //초기화 단계
    Sub_Time=0;
    IC_Time=0;
    Sub_Cnt=1;
    int i = end;
    int k = 0;
    int limit = 0;
    //초기화 단계


        int way[R];
    int Now;
        while (path[i] != -1){ // -1에 도달할때까지 인덱스들을 저장
                way[k++]=i; // 인덱스 저장
                way[k++]=path[i]; // i 다음으로 이어지는 인덱스 저장
                i = path[i]; // 다음
        }
    //printf("(%d 인덱스)", k-1);
    if(strcmp(subinfo[way[0]].name,subinfo[way[1]].name)==0) limit+=1; // 도착역과 이전역의 이름이 동일하다면 = 환승이면 한개를 덜 읽게 만든다.
    printf("₩n<출발>₩n");
    //printf("(%d분)", Sub_Time);
    printf("-><%s> %s₩n", csvLists[subinfo[way[k-1]].num],subinfo[way[k-1]].name);
        for(int q = k-1; q > limit; q=q-2){
    if(arr[way[q]][way[q-1]].ic==0){
        // 환승이 아닐경우
        Now=arr[way[q]][way[q-1]].data; //시간은 그대로 저장
        Sub_Time+=Now;
        Sub_Cnt++;
        if(q<4||arr[way[q-2]][way[q-3]].ic==0){ // 다음역이 환승역이 아니면 진행, 쇼
```

```
트서킷으로 잘못된 참조 방지
                    //printf("(%d분)", Sub_Time);
                    printf("-><%s>                                                %s₩n",
csvLists[subinfo[way[q-1]].num],subinfo[way[q-1]].name);
                }
            }
            else{
                // 환승일 경우
                if(q<4||arr[way[q-2]][way[q-3]].ic==0){ // 다음역이 환승역(중복)이 아니면 진
행, 쇼트서킷으로 잘못된 참조 방지
                    Now=arr[way[q]][way[q-1]].data;
                    if(option==2) Now=arr[way[q]][way[q-1]].data-1000; // 최소환승일경우 가
중치가 더해져 있으므로 1000분 제거
                    IC_Time+=Now;
                    printf("-><환승 : 소요시간 %d 분> %s₩n", Now,subinfo[way[q-1]].name);
                }
            }
              }
              printf("₩n");
        if(option==1) printf("최단거리로 도착₩n");
        if(option==2) printf("최소환승으로 도착₩n");
        printf("소요시간 : %d (%d + 환승 소요시간 %d) 분
₩n",Sub_Time+IC_Time,Sub_Time,IC_Time);
        printf("정거장 수 : %d 개₩n₩n",Sub_Cnt);
}
```

함수명		calc_path			
		자료명	자료형	변수명	매체
입 력	1	구조체 행렬	element**,	arr	전 달
입 력	1	역명 구조체	sublist	subinfo[]	전 달
입 력	2	시작점, 끝점	int	start, end	전 달
출 력	1	총 이동시간	int	Sub_Time+IC_Time	화 면
작업변수	4	반복문 매개변수	int	k, q, i, limit	생 성
작업변수	1	임시 정수 배열	int	way[]	생 성

처리 과정 코딩

```
{
    //초기화 단계
    shortest_path(arr, start); //해당 인덱스의 다익스트라를 돌림
    Sub_Time=0;
    IC_Time=0;
    Sub_Cnt=1;
        int i = end;
        int k = 0;
    int limit = 0;
    //초기화 단계

        int way[R];
    int Now;
        while (path[i] != -1){ // -1에 도달할때까지 인덱스들을 저장
                way[k++]=i; // 인덱스 저장
                way[k++]=path[i]; // i 다음으로 이어지는 인덱스 저장
                i = path[i]; // 다음
```

```
            }
    if(strcmp(subinfo[way[0]].name,subinfo[way[1]].name)==0) limit+=1; // 도착역과 이전역의 이름이 동일하다면 = 환승이면 한개를 덜 읽게 만든다.
              for(int q = k-1; q > limit; q=q-2){
          if(arr[way[q]][way[q-1]].ic==0){
              // 환승이 아닐경우
              // 가중치를 더하고 정거장 카운트 올림
              Now=arr[way[q]][way[q-1]].data;
              Sub_Time+=Now;
              Sub_Cnt++;
          }
          else{
              // 환승일 경우
              // 가중치를 더하고 정거장 카운트 올림
              if(q<4||arr[way[q-2]][way[q-3]].ic==0){ // 다음역이 환승역(중복)이 아니면 진행, 쇼트서킷으로 잘못된 참조 방지
                  Now=arr[way[q]][way[q-1]].data;
                  if(option==2) Now=arr[way[q]][way[q-1]].data-1000; // 최소환승일경우 가중치가 더해져 있으므로 1000분 제거?
                  IC_Time+=Now;
              }
          }
            }
    return Sub_Time+IC_Time;
}
```

함수명		sub_find			
		자료명	자료형	변수명	매체
입 력	1	구조체 행렬	element**,	arr	전 달
입 력	1	역명 구조체	sublist	subinfo[]	전 달
출 력	1	총 이동시간	int	Sub_Time+IC_Time	화 면
작업변수	1	반복문 매개변수	int	i	생 성
작업변수	1	최단 시간	int	min_time	생 성
작업변수	4	인덱스 및 인덱스 배열 등	int	sub1_idx, sub2_idx, ov_idx[R], ov_time[R],curIdx	생 성
작업변수	3	문자열 입력	char	char sub1[],char sub2[],Chk_num[]	생 성

처리 과정 코딩

```
{   char sub1[100]; // 출발역 입력
    char sub2[100]; // 도착역 입력
    int sub1_idx; // 출발역 인덱스
    int sub2_idx; // 도착역 인덱스
    int ov_idx[R]; // 출발역 환승 중복 인덱스 저장 배열
    int ov_time[R]; // 출발역 환승 중복 시간 저장 배열
    int curIdx=0; // 출발역 환승 중복 인덱스 배열 인덱스
    int min_time; // 출발역이 환승일경우 환승 역중 가장 최단인 시간

    while(1){ // 역 이름을 입력하는 부분
        printf("출발역을 입력해주세요: ");
        fgets(sub1,sizeof(sub1),stdin); //출발역 입력
        //sub1이 비어있지 않고 0번째가 ₩0이 아니면서 끝부분이 ₩n일경우
        if(sub1 !=NULL && sub1[0]!='₩0' && sub1[strlen(sub1)-1]=='₩n')
            sub1[strlen(sub1)-1]='₩0';
```

```
        printf("도착역을 입력해주세요: ");
        fgets(sub2,sizeof(sub2),stdin); //도착역 입력
        //sub2가 비어있지 않고 0번째가 ₩0이 아니면서 끝부분이 ₩n일경우
        if(sub2 !=NULL && sub2[0]!='₩0' && sub2[strlen(sub2)-1]=='₩n')
            sub2[strlen(sub2)-1]='₩0';

        if(strcmp(sub1,sub2)==0){ // 출발역과 도착역이 동일하면 에러 1
            printf("에러1 : 출발역과 도착역이 동일합니다! ₩n");
            printf("다시 입력해주세요! ₩n");
            continue;
        }
        sub1_idx=subChk(subinfo,sub1);
        sub2_idx=subChk(subinfo,sub2);
        if (sub1_idx==-1||sub2_idx==-1){ // 인덱스가 존재하지 않으면 에러 2
            printf("에러2 : 출발역또는 도착역이 정확하지 않습니다! ₩n");
            printf("다시 입력해주세요! ₩n");
            continue;
        }
        break;
    }
    while(1){ // 우선 방식을 입력하는 부분
        printf("방식? 1. 최단경로 2. 최소환승₩n");
        printf(": ");
        char Chk_num[5]; // 우선방식 여부
        fgets(Chk_num,sizeof(Chk_num),stdin); //출발역 입력
        //sub1이 비어있지 않고 0번째가 ₩0이 아니면서 끝부분이 ₩n일경우
        if(Chk_num !=NULL && Chk_num[0]!='₩0' &&
Chk_num[strlen(Chk_num)-1]=='₩n')
            Chk_num[strlen(Chk_num)-1]='₩0';

        if(atoi(Chk_num)!=1&&atoi(Chk_num)!=2){ // 값이 정해진 값이 아니면 에러 3
            printf("에러3 : 올바르지 않은 값입니다! ₩n");
            printf("다시 입력해주세요! ₩n");
            continue;
        }
        else if(atoi(Chk_num)==1){
            option=1;
        }
        else{
            option=2;
        }
        break;
    }

    for(int i=0;i<R;i++){
        if(strcmp(sub1,subinfo[i].name)==0){ // 이름이 동일하다면
            ov_idx[curIdx]=i; // 해당 출발역 인덱스를 저장
            ov_time[curIdx++]=calc_path(i, sub2_idx,subinfo,subarray); // 해당 출발역 소요
시간 저장
        }
    }
```

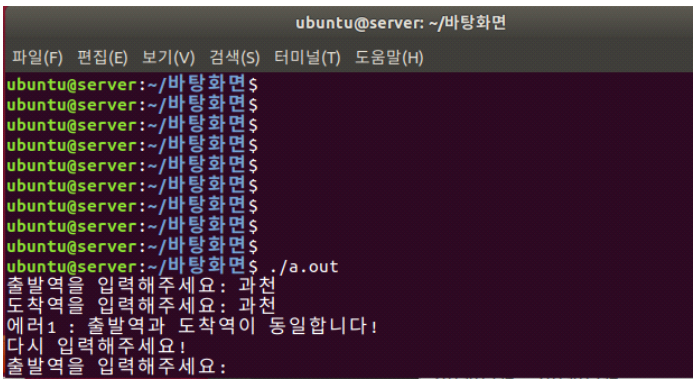
```
    min_time = ov_time[0]; // 0번째 배열값 저장
    sub1_idx = ov_idx[0]; // 0번째 인덱스 저장
    if(curIdx>1){
        // 환승일 경우 (출발역이 여러개)
        for (int i = 1; i < curIdx; i++){
            if (ov_time[i] < min_time){
                // 최소일 경우
                min_time = ov_time[i]; // i번째 시간 저장
                sub1_idx = ov_idx[i]; // i번째 인덱스 저장
            }
        }
    }
    shortest_path(subarray, sub1_idx); // 다익스트라
          print_path(sub1_idx, sub2_idx, subinfo, subarray); // 최종 출력 함수
}
```

4. 테스트 및 결과

1) 에러검사

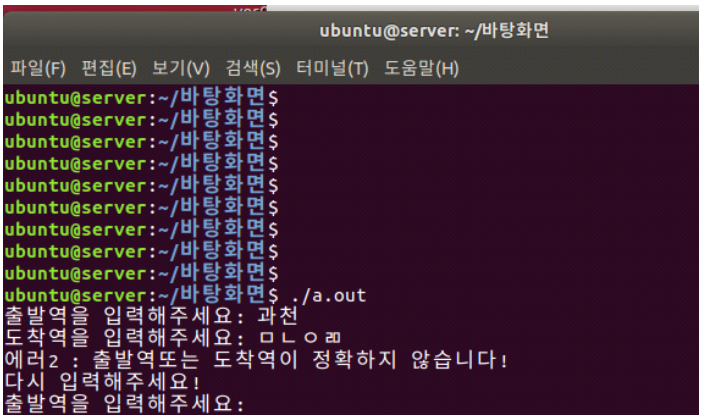
에러 1 출발역 도착역 동일에러

- 출발역과 도착역의 입력이 동일할시 에러



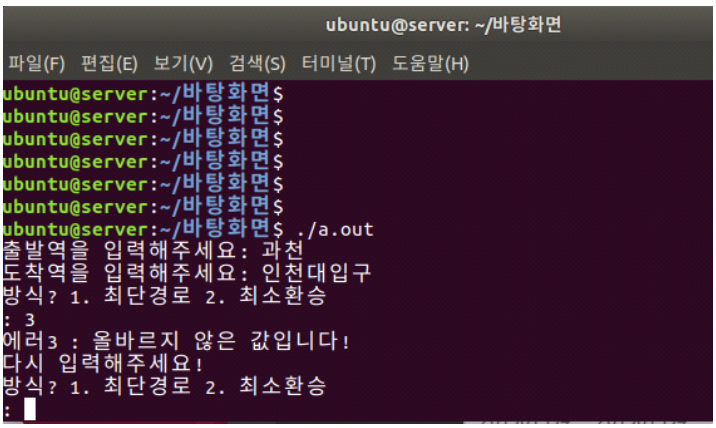
에러 2 출발역 도착역 입력에러

- 출발역 또는 도착역이 존재하지 않으면 에러



에러 3 올바르지 않은 값 에러

- 최단경로, 최소환승 외에 다른 것을 선택하면 에러



2) 출발역, 도착역 입력시

```
출발역을 입력해주세요: 과천
도착역을 입력해주세요: 인천대입구
방식? 1. 최단경로 2. 최소환승
: 1

<출발>
-><4호선> 과천
-><4호선> 대공원
-><4호선> 경마공원
-><4호선> 선바위
-><4호선> 남태령
-><환승 : 소요시간 7 분> 사당
-><2호선> 낙성대
-><2호선> 서울대입구
-><2호선> 봉천
-><2호선> 신림
-><2호선> 신대방
-><2호선> 구로디지털단지
-><2호선> 대림
-><환승 : 소요시간 3 분> 신도림
-><1호선> 구로
-><1호선> 구일
-><1호선> 개봉
-><1호선> 오류동
-><1호선> 온수
-><1호선> 역곡
-><1호선> 소사
-><1호선> 부천
-><1호선> 중동
-><1호선> 송내
-><1호선> 부개
-><환승 : 소요시간 2 분> 부평
-><인천1선> 동수
-><인천1선> 부평삼거리
-><인천1선> 간석오거리
-><인천1선> 인천시청
-><인천1선> 예술회관
-><인천1선> 인천터미널
-><인천1선> 문학경기장
-><인천1선> 선학
-><인천1선> 신연수
-><인천1선> 원인재
-><인천1선> 동춘
-><인천1선> 동막
-><인천1선> 캠퍼스타운
-><인천1선> 테크노파크
-><인천1선> 지식정보단지
-><인천1선> 인천대입구

최단거리로 도착
소요시간 : 101 (89 + 환승 소요시간 12) 분
정거장 수 : 42 개

프로그램이 종료되었습니다
```

```
출발역을 입력해주세요: 과천
도착역을 입력해주세요: 인천대입구
방식? 1. 최단경로 2. 최소환승
: 2

<출발>
-><4호선> 과천
-><4호선> 대공원
-><4호선> 경마공원
-><4호선> 선바위
-><4호선> 남태령
-><4호선> 사당
-><4호선> 이수
-><4호선> 동작
-><4호선> 이촌
-><4호선> 신용산
-><4호선> 삼각지
-><4호선> 숙대입구
-><환승 : 소요시간 3 분> 서울역
-><1호선> 남영
-><1호선> 용산
-><1호선> 노량진
-><1호선> 대방
-><1호선> 신길
-><1호선> 영등포
-><1호선> 신도림
-><1호선> 구로
-><1호선> 구일
-><1호선> 개봉
-><1호선> 오류동
-><1호선> 온수
-><1호선> 역곡
-><1호선> 소사
-><1호선> 부천
-><1호선> 중동
-><1호선> 송내
-><1호선> 부개
-><환승 : 소요시간 6 분> 부평
-><인천1선> 동수
-><인천1선> 부평삼거리
-><인천1선> 간석오거리
-><인천1선> 인천시청
-><인천1선> 예술회관
-><인천1선> 인천터미널
-><인천1선> 문학경기장
-><인천1선> 선학
-><인천1선> 신연수
-><인천1선> 원인재
-><인천1선> 동춘
-><인천1선> 동막
-><인천1선> 캠퍼스타운
-><인천1선> 테크노파크
-><인천1선> 지식정보단지
-><인천1선> 인천대입구

최소환승으로 도착
소요시간 : 115 (106 + 환승 소요시간 9) 분
정거장 수 : 48 개

프로그램이 종료되었습니다
```

- 올바르게 출발역과 도착역을 입력하면 최단거리와 최소환승을 고를 수 있고, 고른 것에 따라 최적의 경로와 이동 시간이 출력됨

5. 소스코드 (주석포함)

```
/* 프로그램명: SubwayCalc.c 스택함수를 이용한 스택 계산기 프로그램*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <limits.h>

#define TRUE 1
#define FALSE 0
#define INF          9999
#define MAX_VERTICES 552 //정점(역)의 수

int Sub_Time=0; // 이동 시간 전역변수 (환승 시간 제외)
int Sub_Cnt=0; // 정거장 횟수 전역변수 (환승 시간 제외)
int IC_Time=0; // 환승 시간 전역변수
int option=1; // 우선방식 여부 전역변수 (1=최단거리, 2=최소환승)

const int R=MAX_VERTICES; // 행렬 상수
const int C=MAX_VERTICES; // 행렬 상수
const int IC=150; // 환승역 개수 상수

typedef struct sublist{ // 역명 구조체
    char name[30]; // 역명
    char code[10]; // 역 코드
    int num; // 호선 번호
} sublist;

typedef struct element{ // 인접행렬 구조체
    char from[10]; // 출발지
    char to[10]; // 목적지
    int data; // 가는 시간
    int ic; // 환승 여부
} element;

int distance[MAX_VERTICES]; // 시작 역으로부터의 최단 경로 거리
int found[MAX_VERTICES]; // 방문한 역 표시
int path[MAX_VERTICES]; // 최단거리 역까지 거치는 노드들을 저장
int check[MAX_VERTICES];// 한 역으로 가는 역을 표시


char *csvLists[20]={"1호선","1지선","2호선","2지선","3호선","4호선",
                    "5호선","5지선","6호선","7호선","8호선","9호선",
                    "분당선","인천1선","중앙선","경춘선","경의선","공항철도","역이름","환승정보"}; //역 파일명 모음



/*
* 함수명: makeArray()
* 인 자 : 없음
```

```
* 리턴형: element**
* 설 명 : 구조체 행렬을 동적 생성하는 함수
*/
element** makeArray(){
    element **arr = malloc(sizeof(element *) * R);   // 이중 포인터에 (element 포인터 크기 * row)만큼 동적 메모리 할당. 배열의 세로
    for (int i = 0; i < R; i++){              // 세로 크기만큼 반복
        arr[i] = malloc(sizeof(element) * C);     // (element의 크기 * col)만큼 동적 메모리 할당. 배열의 가로
    }
    return arr;
}



/*
* 함수명: initArray(arr)
* 인 자 : element**
* 리턴형: void
* 설 명 : 구조체 행렬 내부를 초기화하는 함수
*/
void initArray(element** arr){
    for (int i = 0; i < R; i++){
        for (int j = 0; j < C; j++){
                strcpy(arr[i][j].from,""); // 출발지 비움
                strcpy(arr[i][j].to,""); // 목적지 비움
                arr[i][j].data=9999; // 가중치 9999로 초기화
                arr[i][j].ic=FALSE; // 환승 X로 초기화
            }
    }
}
/*
* 함수명: killArray(arr)
* 인 자 : element**
* 리턴형: void
* 설 명 : 구조체 행렬을 할당 해제하는 함수
*/
void killArray(element** arr){
    for (int i = 0; i < R; i++) {
        free(arr[i]);
    }
    free(arr);
}

/*
* 함수명: readCSV(i)
* 인 자 : int
* 리턴형: FILE*
* 설 명 : i번째 파일을 읽어서 FILE*로 리턴하는 함수
*/
FILE* readCSV(int i){
        char fileDir[50] = "./data/"; // 하위 경로
        strcat(fileDir, csvLists[i]); // i번째 파일명을 하위 경로에 붙임
```

```
        strcat(fileDir, ".csv"); // .csv를 하위 경로에 붙임
        FILE *stream = fopen(fileDir, "r"); // 하위 경로의 파일을 불러옴
        if (stream != NULL){ //파일이 비어있지 않으면
            return stream;
        }
        else{ //파일이 비어있거나 문제가 있다면
            printf("Failed to load %s !!!", fileDir); //오류 출력
            exit(1); //종료
        }
}

/*
* 함수명: readSubInfo(subinfo[])
* 인 자 : sublist
* 리턴형: void
* 설 명 : 지하철 역명 구조체 정보의 역명과 코드를 채우는 함수
*/
void readSubInfo(sublist subinfo[]){
        char buf[2048];
        FILE *stream = readCSV(18);
        char *line, *tmp;
        line = fgets(buf, 2048, stream); // 한줄 미리 읽음 (안쓰는 칸 거르기)
        int idx=0;
        while ((line = fgets(buf, 2048, stream))!=NULL) {
            int i=0;
            tmp = strtok(buf, ","); //
            while (tmp != NULL){
                if (i == 0){
                    strcpy(subinfo[idx].code, tmp);
                }
                else if (i == 1){
                    if(idx!=R) // 마지막줄이 아니면
                        tmp[strlen(tmp) - 1] = '₩0'; // ₩n을 제거해준다.
                    strcpy(subinfo[idx].name, tmp);
                }
                i++;
                tmp = strtok(NULL, "");
            }
            idx++;
        }

        fclose(stream);
}

/*
* 함수명: readSubArray(arr,subinfo[])
* 인 자 : element** , sublist
* 리턴형: void
* 설 명 : 구조체 행렬의 정보와 지하철 역명 구조체 정보의 호선 인덱스를 채우는 함수
*/
void readSubArray(element** arr,sublist subinfo[]){
```

```
    int cRow=0; // 현재 행열 값을 저장하는 벼수
    char *tmp, *line; //줄과 단어 변수 선언
    char buf[2048]; // 버퍼 선언
    srand(time(NULL)); //완전 랜덤화

    //인접행렬 정보 입력
    for (int i = 0; i < 18; i++){
        FILE *stream = readCSV(i);
        char *line, *tmp;
        line = fgets(buf, 2048, stream); // 한줄 미리 읽음 (안쓰는 칸 거르기)
        int tmpR = 0, tmpC = 0; //임시 행 열
        while ((line = fgets(buf, 2048, stream)) != NULL){ //한줄씩 읽음
            tmpC = 0; //임시 열을 0으로 초기화
            tmp = strtok(line, ","); // ,로 나눠진 line의 첫 단어를 읽음
            tmp = strtok(NULL, ","); //다음 단어를 읽음 ( 첫번째 역 코드 뛰어넘기 )
            while (tmp != NULL){ // 읽은 단어가 NULL이 아니면
                subinfo[cRow + tmpR].num=i; // ~호선 인덱스를 역명 구조체에 저장
                arr[cRow + tmpR][cRow + tmpC].data = atoi(tmp); // 구조체 행렬에 cRow + tmpR,C의 인덱스
에 해당 값 저장
                tmp = strtok(NULL, ","); //다음 단어를 읽음
                tmpC++; // 임시 열 추가
            }
            tmpR++; // 임시행 추가
        }
        cRow = cRow + tmpR; // 현재 열에 임시 행만큼 더해줘서 저장
        fclose(stream); //파일 닫기
    }

    //여기서부터는 구조체의 from to 저장
    for (int i = 0; i < R; i++){
        for (int j = 0; j < R; j++){
            strcpy(arr[i][j].from,subinfo[i].code); //arr의 i행에 출발 역코드 저장
            strcpy(arr[i][j].to,subinfo[j].code); //arr의 j행에 도착 역코드 저장
        }
    }

    //여기서부터는 환승하는 코드의 인덱스 저장
    FILE *stream = readCSV(19); // 환승파일 읽기
    int tmpCnt = 0;
    int tmpIC[IC];
    line = fgets(buf, 2048, stream);
    tmp = strtok(line, ",");
    //tmp = strtok(NULL, ","); //환승 단어 스킵
    while (tmp != NULL){
        if(tmpCnt==IC-1) // 마지막 단어면
                tmp[strlen(tmp) - 1] = '₩0'; // ₩n을 제거해준다.
        for (int i = 0; i < R; i++){
            if (strcmp(tmp, subinfo[i].code) == 0){ // 역 코드가 동일하다면
                tmpIC[tmpCnt] = i;// 인덱스 저장
                //printf("%d) %s %d 인덱스 [%s]₩n", tmpCnt,tmp,i,subinfo[i].name);
                tmpCnt++; // 다음
```

```
            }
        }
        tmp = strtok(NULL, ",");//다음 읽기
    }

    //여기서부터는 환승정보를 인접행렬에 저장
    tmpCnt=0;
    int i=0;
    int j=0;
    int m;
    int n;
    while ((line = fgets(buf, 2048, stream)) != NULL){ //다음 줄 읽기
        j=0;
        tmp = strtok(line, ",");
        tmp = strtok(NULL, ",");
        while (tmp != NULL){
            if(j==IC-1) // 마지막 단어면
                tmp[strlen(tmp) - 1] = '₩0'; // ₩n을 제거해준다.
            m = tmpIC[i]; // 행 인덱스 불러오기
            n = tmpIC[j]; // 열 인덱스 불러오기
            if (atoi(tmp) != 9999&&atoi(tmp) != 0){ // 해당 가중치가 방문할 수 있는 값이면
                //printf("%d(%s), ",n,subinfo[n].name);
                arr[m][n].data = (rand() % atoi(tmp)) + 1; // 1 ~ 해당 값까지 랜덤 난수로 저장
                arr[m][n].ic = TRUE; // 해당 지점은 환승이므로 TRUE
            }
            if (j<IC)
                j++; // 다음 열
            tmp = strtok(NULL, ",");
        }
        //printf("[%d:%s]₩n",m,subinfo[m].name);
        if (i<IC-2)
            i++; // 다음 행
    }
    fclose(stream);
}

/*
* 함수명: subChk(subinfo[],chk[])
* 인 자 : sublist,char
* 리턴형: int
* 설 명 : 역명 인덱스를 탐색하는 함수
*/
int subChk(sublist subinfo[],char chk[]){
    for (int i=0;i<R;i++){
        if (strcmp(chk,subinfo[i].name)==0) //해당하는 문자열을 찾으면
            return i; //해당 인덱스 출력
    }
    return -1; //목록에 없으면 -1 출력
}

/*
```

```
* 함수명: choose(distance[], n, found[])
* 인 자 : int 3개
* 리턴형: int
* 설 명 : 시간이 가장 작은 인덱스를 리턴하는 함수
*/
int choose(int distance[], int n, int found[]){
          int i, min, minpos;
          min = INT_MAX; // 최대로 설정
          minpos = -1;
          for (i = 0; i<n; i++)
                    if (distance[i]< min && !found[i]) { //방문하지 않았고 최소보다 더 작으면
                              min = distance[i]; // 거리 입력
                              minpos = i; // 인덱스 입력
                    }
          return minpos;
}

/*
* 함수명: shortest_path(arr, start)
* 인 자 : element**, int
* 리턴형: void
* 설 명 : 최단거리를 탐색하는 함수
*/
void shortest_path(element** arr, int start){
          int i, u, w;
    static int trans_done=0; // 정적변수
          for (i = 0; i<R; i++){  /* 초기화 */
                    distance[i] = arr[start][i].data;
                    found[i] = FALSE;
                    path[i] = start;
          }
          path[start] = -1;
    if(option==2&&trans_done==0){ //최소환승 옵션일 경우
        for (i = 0; i<R; i++){
                        for (int j = 0; j<R; j++){
                if(arr[i][j].ic==1)// 환승역이면
                    arr[i][j].data+=1000; //가중치를 크게 증가시킴
                  }
              }
        trans_done=1;
    }

          found[start] = TRUE;     /* 시작 정점 방문 표시 */
          distance[start] = 0;
          for (i = 0; i<R-1; i++) {
                    u = choose(distance, R, found);
                    found[u] = TRUE; // 해당 정점을 방문 표시
                    for (w = 0; w<R; w++)
                              if (!found[w]) // 방문한적이 없다면
                                        if (distance[u] + arr[u][w].data < distance[w]){
                                                  path[w]=u; //인덱스 저장
```

```
                                                distance[w] = distance[u] + arr[u][w].data;
                                            }
                    }
}

/*
* 함수명: print_path(int start, int end,sublist subinfo[],element** arr)
* 인 자 : int, int, sublist, element**
* 리턴형: void
* 설 명 : path[]를 탐색하면서 정거장과 이동 시간을 출력하는 함수
*/
void print_path(int start, int end,sublist subinfo[],element** arr){
    //초기화 단계
    Sub_Time=0;
    IC_Time=0;
    Sub_Cnt=1;
        int i = end;
        int k = 0;
    int limit = 0;
    //초기화 단계

        int way[R];
    int Now;
        while (path[i] != -1){ // -1에 도달할때까지 인덱스들을 저장
                way[k++]=i; // 인덱스 저장
                way[k++]=path[i]; // i 다음으로 이어지는 인덱스 저장
                i = path[i]; // 다음
        }
    //printf("(%d 인덱스)", k-1);
    if(strcmp(subinfo[way[0]].name,subinfo[way[1]].name)==0) limit+=1; // 도착역과 이전역의 이름이 동일하다면 = 환승이면 한개를 덜 읽게 만든다.
    printf("₩n<출발>₩n");
    //printf("(%d분)", Sub_Time);
    printf("->%s> %s₩n", csvLists[subinfo[way[k-1]].num],subinfo[way[k-1]].name);
        for(int q = k-1; q > limit; q=q-2){
        if(arr[way[q]][way[q-1]].ic==0){
            // 환승이 아닐경우
            Now=arr[way[q]][way[q-1]].data; //시간은 그대로 저장
            Sub_Time+=Now;
            Sub_Cnt++;
            if(q<4||arr[way[q-2]][way[q-3]].ic==0){ // 다음역이 환승역이 아니면 진행, 쇼트서킷으로 잘못된 참조 방지
                //printf("(%d분)", Sub_Time);
                printf("->%s> %s₩n", csvLists[subinfo[way[q-1]].num],subinfo[way[q-1]].name);
            }
        }
        else{
            // 환승일 경우
            if(q<4||arr[way[q-2]][way[q-3]].ic==0){ // 다음역이 환승역(중복)이 아니면 진행, 쇼트서킷으로 잘못된 참조 방지
                Now=arr[way[q]][way[q-1]].data;
```

```
                if(option==2) Now=arr[way[q]][way[q-1]].data-1000; // 최소환승일경우 가중치가 더해져 있으므로 1000분 제거
                IC_Time+=Now;
                printf("->(환승 : 소요시간 %d 분) %s₩n", Now,subinfo[way[q-1]].name);
            }
        }
          }
          printf("₩n");
    if(option==1) printf("최단거리로 도착₩n");
    if(option==2) printf("최소환승으로 도착₩n");
    printf("소요시간 : %d (%d + 환승 소요시간 %d) 분₩n",Sub_Time+IC_Time,Sub_Time,IC_Time);
    printf("정거장 수 : %d 개₩n₩n",Sub_Cnt);
}

/*
* 함수명: calc_path(int start, int end,sublist subinfo[],element** arr)
* 인 자 : int, int, sublist, element**
* 리턴형: int
* 설 명 : path[]를 탐색하면서 총 이동 시간을 리턴하는 함수
*/
int calc_path(int start, int end,sublist subinfo[],element** arr){
    //초기화 단계
    shortest_path(arr, start); //해당 인덱스의 다익스트라를 돌림
    Sub_Time=0;
    IC_Time=0;
    Sub_Cnt=1;
        int i = end;
        int k = 0;
    int limit = 0;
    //초기화 단계

        int way[R];
    int Now;
        while (path[i] != -1){ // -1에 도달할때까지 인덱스들을 저장
                way[k++]=i; // 인덱스 저장
                way[k++]=path[i]; // i 다음으로 이어지는 인덱스 저장
                i = path[i]; // 다음
        }
    if(strcmp(subinfo[way[0]].name,subinfo[way[1]].name)==0) limit+=1; // 도착역과 이전역의 이름이 동일하다면 = 환승이면 한개를 덜 읽게 만든다.
        for(int q = k-1; q > limit; q=q-2){
        if(arr[way[q]][way[q-1]].ic==0){
            // 환승이 아닐경우
            // 가중치를 더하고 정거장 카운트 올림
            Now=arr[way[q]][way[q-1]].data;
            Sub_Time+=Now;
            Sub_Cnt++;
        }
        else{
            // 환승일 경우
            // 가중치를 더하고 정거장 카운트 올림
```

```
            if(q<4||arr[way[q-2]][way[q-3]].ic==0){ // 다음역이 환승역(중복)이 아니면 진행, 쇼트서킷으로 잘못된 참조 방지
                Now=arr[way[q]][way[q-1]].data;
                if(option==2) Now=arr[way[q]][way[q-1]].data-1000; // 최소환승일경우 가중치가 더해져 있으므로 1000분 제거?
                IC_Time+=Now;
            }
        }
          }
    return Sub_Time+IC_Time;
}

void debug_print(element** subarray,sublist subinfo[],int debug){ //디버그용
    if (debug==1){
        for(int i=0;i<R;i++){
            for (int j = 0; j < R; j++){
                if(subarray[i][j].data!=9999&&subarray[i][j].data!=0){
                    if(subarray[i][j].ic==TRUE){
                        printf("[%d -> %d] : %d ", i,j,subarray[i][j].data);
                        printf("[%s (%s) -> %s (%s)] 환승₩n", subarray[i][j].from,csvLists[subinfo[i].num],subarray[i][j].to,csvLists[subinfo[j].num]);
                    }
                    else{
                        printf("[%d -> %d] : %d ", i,j,subarray[i][j].data);
                        printf("[%s -> %s]₩n", subarray[i][j].from,subarray[i][j].to);
                    }
                }
            }
        }
        /*
        for(int i=0;i<R;i++){
            printf("%d) %s - %s (%s)₩n",i,subinfo[i].code,subinfo[i].name,csvLists[subinfo[i].num]);
        }*/
    }
}

/*
* 함수명: sub_find(element** subarray, sublist subinfo[])
* 인 자 : element**, sublist
* 리턴형: void
* 설 명 : 출발역, 도착역과 최단거리, 최소환승을 입력받고 관련 함수들을 돌리는 함수 (사실상 main)
*/
void sub_find(element** subarray,sublist subinfo[]){
    char sub1[100]; // 출발역 입력
    char sub2[100]; // 도착역 입력
    int sub1_idx; // 출발역 인덱스
    int sub2_idx; // 도착역 인덱스
    int ov_idx[R]; // 출발역 환승 중복 인덱스 저장 배열
    int ov_time[R]; // 출발역 환승 중복 시간 저장 배열
    int curIdx=0; // 출발역 환승 중복 인덱스 배열 인덱스
    int min_time; // 출발역이 환승일경우 환승 역중 가장 최단인 시간
```

```
while(1){ // 역 이름을 입력하는 부분
    printf("출발역을 입력해주세요: ");
    fgets(sub1,sizeof(sub1),stdin); //출발역 입력
    //sub1이 비어있지 않고 0번째가 ₩0이 아니면서 끝부분이 ₩n일경우
    if(sub1 !=NULL && sub1[0]!='₩0' && sub1[strlen(sub1)-1]=='₩n')
        sub1[strlen(sub1)-1]='₩0';

    printf("도착역을 입력해주세요: ");
    fgets(sub2,sizeof(sub2),stdin); //도착역 입력
    //sub2가 비어있지 않고 0번째가 ₩0이 아니면서 끝부분이 ₩n일경우
    if(sub2 !=NULL && sub2[0]!='₩0' && sub2[strlen(sub2)-1]=='₩n')
        sub2[strlen(sub2)-1]='₩0';

    if(strcmp(sub1,sub2)==0){ // 출발역과 도착역이 동일하면 에러 1
        printf("에러1 : 출발역과 도착역이 동일합니다! ₩n");
        printf("다시 입력해주세요! ₩n");
        continue;
    }
    sub1_idx=subChk(subinfo,sub1);
    sub2_idx=subChk(subinfo,sub2);
    if (sub1_idx==-1||sub2_idx==-1){ // 인덱스가 존재하지 않으면 에러 2
        printf("에러2 : 출발역또는 도착역이 정확하지 않습니다! ₩n");
        printf("다시 입력해주세요! ₩n");
        continue;
    }
    break;
}
while(1){ // 우선 방식을 입력하는 부분
    printf("방식? 1. 최단경로 2. 최소환승₩n");
    printf(": ");
    char Chk_num[5]; // 우선방식 여부
    fgets(Chk_num,sizeof(Chk_num),stdin); //출발역 입력
    //sub1이 비어있지 않고 0번째가 ₩0이 아니면서 끝부분이 ₩n일경우
    if(Chk_num !=NULL && Chk_num[0]!='₩0' && Chk_num[strlen(Chk_num)-1]=='₩n')
        Chk_num[strlen(Chk_num)-1]='₩0';

    if(atoi(Chk_num)!=1&&atoi(Chk_num)!=2){ // 값이 정해진 값이 아니면 에러 3
        printf("에러3 : 올바르지 않은 값입니다! ₩n");
        printf("다시 입력해주세요! ₩n");
        continue;
    }
    else if(atoi(Chk_num)==1){
        option=1;
    }
    else{
        option=2;
    }
    break;
}
```

```
    for(int i=0;i<R;i++){
        if(strcmp(sub1,subinfo[i].name)==0){ // 이름이 동일하다면
            ov_idx[curIdx]=i; // 해당 출발역 인덱스를 저장
            ov_time[curIdx++]=calc_path(i, sub2_idx,subinfo,subarray); // 해당 출발역 소요시간 저장
        }
    }
    min_time = ov_time[0]; // 0번째 배열값 저장
    sub1_idx = ov_idx[0]; // 0번째 인덱스 저장
    if(curIdx>1){
        // 환승일 경우 (출발역이 여러개)
        for (int i = 1; i < curIdx; i++){
            if (ov_time[i] < min_time){
                // 최소일 경우
                min_time = ov_time[i]; // i번째 시간 저장
                sub1_idx = ov_idx[i]; // i번째 인덱스 저장
            }
        }
    }
    shortest_path(subarray, sub1_idx); // 다익스트라
          print_path(sub1_idx, sub2_idx, subinfo, subarray); // 최종 출력 함수
}

int main(){
    element** subarray= makeArray();
    initArray(subarray);

    sublist subinfo[R];
    readSubInfo(subinfo);
    readSubArray(subarray,subinfo);

    debug_print(subarray,subinfo,0);
    sub_find(subarray,subinfo);

    killArray(subarray); //인접행렬 free
    printf("프로그램이 종료되었습니다 ₩n");

    return 0;
}
```