2016115572 권현수 sws.pdf

-입력처리와 동적할당

3 10 -1 3 2 6 9 -2 4 1 3 5 8 -2 2 7 9 -2 -3

위와 같은 입력이 들어왔을 때, 1번째 인자는 num_of_plate변수에 저장합니다. 그리고 array의 세로배열을 sum_of_plate만큼 동적할당 합니다. 2번째 인자는 plate의 길이를 나타내는 width 변수에 저장합니다. 3번째 인자는 option에 저장하고, 이것은 받아와야하는 plate의 개수나 slit의 개수가 이미 명시 되어있기 때문에 option에 의해 따로 수행을 할 내용이 없습니다.

이후 num_of_plate만큼 반복문을 돌려 여러 입력을 받습니다. 첫 번째로, slit의 개수를 받아서 num_of_slit변수에 저장합니다. 그리고 num_of_slit만큼 array의 가로배열을 동적할당 합니다. 이후 num_of_slit만큼 반복문을 돌려 slit의 위치를 받습니다. 이것은 array 2차원 배열에 저장됩니다(array[몇 번째 plate][몇 번째 slit]). array가로 배열의 끝부분을 명시해 주기 위해 끝부분에 0을 넣습니다. 그리고 -2를 받습니다.

num_of_plate만큼의 반복이 끝나면 -3을 받습니다.

-구조 및 함수설명

void simulate(int array, int width, int* selected)** - 주어진 array를 width변수를 이용하여 물의 최단경로 알고리즘을 수행합니다. 지나간 slit의 위치는 selected에 저장됩니다. plate의 개수만큼 반복문을 돌려서 find_shortest함수를 수행합니다. 이것을 distance_sum 변수에 계속 더하여 최단 거리를 계산합니다. 반복문을 모두 수행 후, distance_sum을 출력하고, start와 가장짧은 경로를 가진 slit이 저장된 selected배열을 모두 출력합니다.

double find_shortest(double* start, int* array, int* selected) - start로부터 가장 짧은 거리를 가진 slit의 위치를 찾아냅니다. array의 크기 즉, slit의 개수만큼 반복문을 돌려 distance가 가장 짧은 slit을 찾아내고 그것의 위치를 _index변수에 저장하여 selected 배열에 넣습니다. result_index는 selected 배열의 몇 번째에 넣을 것이냐를 나타내는데, 함수의 파라미터로 넣기에 복잡해지기 때문에 전역변수로 설정했습니다. 반복문이 끝나면 start변수에 가장 가까운 slit의 위치를 넣어서 다음 함수 수행 시에 바뀐 start값으로 다시 수행 할 수 있게 합니다.

만약 plate의 width가 홀수일 수도 있기 때문에, 중간에서 물이 떨어지면 그 물의 위치가 정수가 나오지 않으므로 특정 변수의 자료형을 int가 아닌 double로 설정했습니다(distance, shortest, start...).

double absol(double a) - a의 절댓값을 반환합니다. 양수면 그대로 반환, 음수면 그것의 반대 부호를 반환합니다.

int arrsize(int* array) - array 배열이 몇 개까지 채워져 있는지 그것의 개수를 반환합니다. 0이나 NULL이 나올때까지 I를 증가시키고, 나오면 I를 반환합니다.

void printarr(int arr, int n)** - 배열을 프린트합니다. - 디버깅용

끝