# 1.프로젝트 목표 및 문제 소개

### 1-1 프로젝트 목표

문제 해결을 위한 적절한 알고리즘을 선택할수있는 능력 및 선택한 알고리즘과 다른 알고리즘사이의 성능을 비교 평가할 수 있는 분석 능력함양 의사소통을 통한 팀워크 증진 및 팀원 아이디어 평가할 수 있는 능력 배양

### 1-2 프로젝트 문제 소개

**미로 탈출 알고리즘 구현**

기존의 미로 탈출 알고리즘에 HP을 설정하여 길을 탐색할때마다 HP가 -1씩 되는 형식으 로 미로 곳곳에는 HP를 회복할 수 있는 수단(사탕)을 설정하여 사탕을 먹으면서 최단거리를 산출하여 미로를 탈출하는 알고리즘 구현

### 1-3 프로젝트 문제 분석

① 미로 탈출 알고리즘을 이용하여 생성된 미로를 탈출해야한다.

② 미로를 탐색하는데에는 -1 HP가 소모되며 HP가 1이상이어야만 미로를 탈출할 수 있다.

③ 미로 곳곳에는 HP를 채울 수 있는 수단(사탕)이있다.

# 2. 관련기술 분석

**2-1 알고리즘 이란?[1](알고리즘의 개념을 소개하고 프로젝트를 수행함있어 어떤 방식으로 문제 방안을 찾았는지에대한 근거를 제시하기위하여 알고리즘이 어떤것인와 알고리즘 성능평가 기준에 대한 내용을 추가하였습니다)**

**컴퓨터를 이용하여 문제를 풀기위한 방법을 과정이나 절차를 이용해 만들어 놓은 것.**

**컴퓨터가 이해할 수 있는 언어(C, Java...)를 이용해 실행할 내용을 컴퓨터가 할 수 있는 가장 작은 기본 작업의 형태로 만들고 순차적으로 나열한다. 컴퓨터는 지시 사항을 차례대로 하나씩 수행하면서 최종 결과를 얻게 되고 그 결과는 해결하고자 했던 문제의 해답이 된다**

. **2-2 알고리즘 성능평가 유형**

| 구분 | 유형 | 설명 |
|---|---|---|
| 성능 분석 | 알고리즘 복잡도 분석 | 알고리즘이 수행하는 연산의 횟수 측정비교<br>일반적으로 연산의 횟수는 n의함수<br>시간 복잡도 분석 : 수행시간 분석<br>공간 복잡도 분석 : 수행시 필요로 하는 메모리 공간 분석 |
| 성능 측정 | 수행 시간측정 | 알고리즘의실제 수행시간을 측정하는 것<br>알고리즘의 실제 구현물이 필요 |
| 안정성(정확성 증명) | 동일한 상황에서 같은 결과값 도출 | 얼마나 오류가없이 정확하게 알고리즘 동작하는가 |

**2-3 미로 탈출 알고리즘의 종류(사전 조사 에 해당 하는 부분이기때문에 알고리즘의 시간복잡도를 추가하였습니다.)**

**1.깊이 우선 탐색 알고리즘 DFS(Depth Fist Search)**

-백트래킹(Backtracking)

탐색 과정이 시작 노드에서 한없이 깊이 진행되는 것을 막기 위해 깊이 제한(depth bound)을 사용한다. 깊이 제한에 도달할 때까지 목표노드가 발견되지 않으면 최근에 첨가된 노드의 부모노드로 되돌아와서, 부모노드에 이전과는 다른 동작자를 적용하여 새로운 자식노드를 생성한다. 여기서 부모노드로 되돌아오는 과정을 백트래킹(backtracking)이라 한다.

**장점**

- 단지 현 경로상의 노드들만을 기억하면 되므로 저장공간의 수요가 비교적 적다.
- 목표노드가 깊은 단계에 있을 경우 해를 빨리 구할 수 있다.

**단점**

- 해가 없는 경로에 깊이 빠질 가능성이 있다. 따라서 실제의 경우 미리 지정한 임의의 깊이까지만 탐색하고 목표노드를 발견하지 못하면 다음의 경로를 따라 탐색하는 방법이 유용할 수 있다.
- 얻어진 해가 최단 경로가 된다는 보장이 없다. 이는 목표에 이르는 경로가 다수인 문제에 대해 깊이우선 탐색은 해에 다다르면 탐색을 끝내버리므로, 이때 얻어진 해는 최적이 아닐 수 있다는 의미이다.

**시간 복잡도**

인접 리스트를 사용해 그래프를 표현하는 경우, DFS()는 한 정점마다 한 번씩 호출되므로, 정확히 |V|번 호출됩니다. DFS() 한 번의 수행 시간은 모든 인접 간선을 검사하는 for문에 의해 지배되는데, 모든 정점에 대해 DFS()를 수행하고 나면 모든 간선을 정확히 한 번(방향 그래프의 경우) 혹은 두 번(무향 그래프의 경우) 확인함을 알 수 있지요. 따라서 깊이 우선 탐색의 시간 복잡도는 O(|V|+|E|)가 된다.

### 2. 너비 우선 탐색 알고리즘 BFS(Breadth Fist Search)

BFS는 현재 위치에 인접한 모든 위치의 노드를 방문하고, 그 이웃 노드들의 또 다른 이웃 노드들을 방문하는 것은 그 다음에 진행하는 것을 의미하다 . BFS를 가장 효과적으로 구현하는 방법은 큐를 이용해서 순환적 형태로 구현하는 것이 가장 깔끔하다.

**장점**

- 출발노드에서 목표노드까지의 최단 길이 경로를 보장한다.

**단점**

- 경로가 매우 길 경우에는 탐색 가지가 급격히 증가함에 따라 보다 많은 기억 공간을 필요로 하게 된다.
- 해가 존재하지 않는다면 유한 그래프(finite graph)의 경우에는 모든 그래프를 탐색한 후에 실패로 끝난다.
- 무한 그래프(infinite graph)의 경우에는 결코 해를 찾지도 못하고, 끝내지도 못한다.

**시간 복잡도**

**너비 우선 탐색의 시간 복잡도는 깊이 우선 탐색과 다를 것이 없습니다. 모든 정점을 한 번씩 방문하며, 정점을 방문할 때마다 인접한 모든 간선을 검사하기 때문이지요. 따라서, 인접 리스트로 구현된 경우에는 O(|V|+|E|), 인접 행렬로 구현했을 경우 O(|V|^2)의 시간 복잡도를 갖게 됩니다.**

### 3. 스택 알고리즘(Stack)

스 택은 마지막으로 입력된 데이터가 첫 번째로 출력되는 LIFO(Last In First Out)형의 자료구조이다. 스택에 입력되는 데이터는 여러 가지 형태가 있을 수 있다. 스택의 동작은 보통 3가지(push, pop, stack top)로 구분된다.

- push 동작은 스택의 top에 새로운 데이터를 입력(추가)한다.
- pop 동작은 스택의 top에 있는 데이터를 출력(제거)한다.
- stack top은 스택의 top에 있는 데이터를 출력하지만 제거하지는 않는다.

### 4. 큐 알고리즘(Queue)

Queue는 선행적인 자료구조로서 뒷(rear) 부분에서 입력된 데이터가 앞(front) 부분에서 출력되는 FIFO(First In First Out)로 동작한다. Queue의 FIFO에서는 첫 번째로 입력된 데이터가 첫 번째로 출력된다. Queue에 저장되는 데이터 노드들은 스택처럼 Linked list로 연결되지만 동작위치가 스택과는 다르다. 스택은 입력과 출력이 top 위치 한군데에서 발생하지만, Queue는 입력과 출력이 앞과 뒤 양쪽에서 발생한다. Queue는 처리하고자 하는 데이터들을 일렬로 줄지어 보관하는 대기열과 같은 자료구조로서 컴퓨팅에서 많이 활용된다. 예를 들면, 버스를 타기 위해서 정거장에 줄 서있는 사람들의 대기열과 같은 구조로서 제일 먼저 줄을 선 사람이 제일 처음으로 버스에 타는 동작 방식이 Queue이다.

### 5. 재귀호출 알고리즘(recursive)

함수 안에서 함수 자기자신을 호출하는 방식을 재귀호출(recursive call)이라고 한다. 재귀호출은 일반적인 상황에서는 잘 사용하지 않지만 알고리즘을 구현할 때 매우 유용한다. 보통 알고리즘에 따라서 반복문으로 구현한 코드보다 재귀호출로 구현한 코드가 좀 더 직관적이고 이해하기 쉬운 경우가 많다.

시간복잡도 : 반복해서 분할하는 기법을 사용하기 때문에

$$T(n)=\Theta(1)+2\cdot T(n2)+\Theta(n)$$

### 6. 다익스트라 알고리즘(Dijkstra)

컴퓨터 과학에서, 데이크스트라 알고리즘(다익스트라 알고리즘)(영어: Dijkstra algorithm)은 어떤 변도 음수 가중치를 갖지 않는 유향 그래프에서 주어진 출발점과 도착점 사이의 최단 경로 문제를 푸는 알고리즘이다.
데이크스트라 알고리즘은 각각의 꼭짓점 v에 대해 s에서 v까지의 최단 거리 d[v]를 저장하면서 작동한다. 알고

리즘의 시작 시에 d[s]=0이고, s가 아닌 다른 모든 꼭짓점 v에 대해서는 d[v]=∞로 놓아 다른 꼭짓점에 대해서는 아직 최단 경로를 모른다는 사실을 표시한다. 알고리즘이 종료되었을 때 d[v]는 s에서 v까지의 최단 경로의 거리를 나타내게 되고, 만약 경로가 존재하지 않으면 거리는 여전히 무한대로 남는다.

**시간복잡도**

**다익스트라 알고리즘의 시간 복잡도는 자료구조를 어떤 것을 쓰느냐에 따라 다르지만, 배열에서는 O(V^2), 이진 힙에서는 O(ElgV), 피보나치 힙에서는 O(E+VlgV)가 된다.**

**7. 동적 프로그레밍**

( dynamic programming)이란 복잡한 문제를 간단한 여러 개의 문제로 나누어 푸는 방법을 말한다. 이것은 부분 문제 반복과 최적 부분 구조를 가지고 있는 알고리즘을 일반적인 방법에 비해 더욱 적은 시간 내에 풀 때 사용한다.

일반적으로 주어진 문제를 풀기 위해서, 문제를 여러 개의 하위 문제(subproblem)로 나누어 푼 다음, 그것을 결합하여 최종적인 목적에 도달하는 것이다. 각 하위 문제의 해결을 계산한 뒤, 그 해결책을 저장하여 후에 같은 하위 문제가 나왔을 경우 그것을 간단하게 해결할 수 있다. 이러한 방법으로 동적 계획법은 계산 횟수를 줄일 수 있다. 특히 이 방법은 하위 문제의 수가 기하급수적으로 증가할 때 유용하다.

## 3. 전제 조건

미로의 출구는 존재한다는 가정 하에서 프로젝트를 설계

1. 백트래킹 기법 상하좌우 우선 방식으로 출구(*)에서 시작하여 도착지까지 미로를 탐색한다.
2. 출구를 탐색할 때 다시 되돌아오는 길이 생긴다면 다시 되돌아온 길은 사방이 막혀있거나 중복된 길을 갔다는 말이 되기 때문에 중복된 길을 제거하기 위하여 자료구조에서 배웠던 스택을 이용하여 가는 길에 대하여 PUSH 하고 되돌아온 길을 POP 해서 최단 경로 구현 및 스택에 캔디 정보를 구조체로 같이 넘겨 캔디의 정보를 함께 출력할 수 있게 구한다.

## 4. 업무 분담 및 일정 계획

### 4-1.업무 분담(업부 분담 내용에대한 추가사항 기술)

| | |
|---|---|
| 이우경 | 소스 구현(스택과 백트리킹 알고리즘 구현) |
| | |

### 4-2.일정 계획

**1주 백트래킹을 이용하여 미로 탈출 구현**

**2주~ 3주 문제 해결 방안 분석**

- 음의 가중치를 이용하여 문제를 해결하는 방안

( 벨만 - 포드 알고리즘을 사용하여 간선의 비용을 정접과 간선 사이의 비용을 캔디가 없을 경우에는 -1로 주고 있을 경우에는 HP(5) 만큼 증가시키는 알고리즘을 생각하였지만 미로를 생성하는데 필요한 정점이 많아질 것으로 생각하여 비용 문제 및 GUI를 생각해서 다른 방안을 고민하였습니다.)

- 스택을 이용하여 이동경로를 전부 집어넣고 중복되는 값에 대하여 제거하는 식의 알고리즘 해결 방안 고민 -> 선택

**4주 구현 및 테스트**

- 첫 번째로 스택의 구조를 분석하기 위하여 스택에 대하여 조사를 해보았고 스택의 LIFO 구조를 분석하였으며 두 번째로 백트래킹 한 정보들을 스택에 집어넣는 것(PUSH)들을 실행 하였고 중복되는 값(4)일 경우에 제거하는(POP)과정을 거친 뒤에 최단 거리를 출력하게 작성하였습니다.

**5주~ 6주 구현 및 테스트**

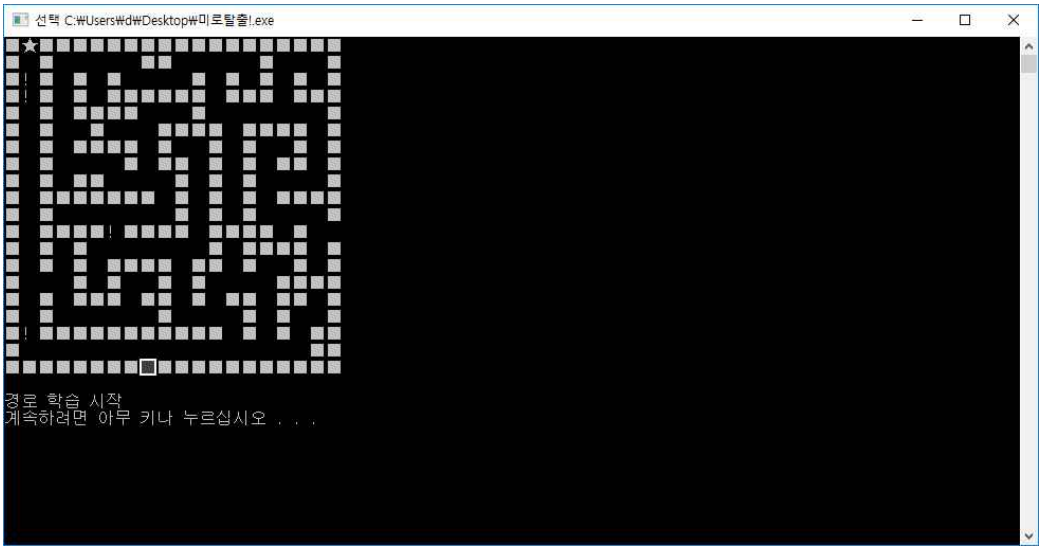
4주에서 만들어진 최단거리 구하는 알고리즘을 가지고 이제 미로에 HP를 가지고 있는 캔디들을 생성하고 움직이는 비용을 계산하고 캔디를 먹었을 시에 HP를 증가시키는 기능 및 이것을 또다시 4번과 결합하고 캔디를 먹으면서 최단 거리를 구하는 알고리즘을 구현

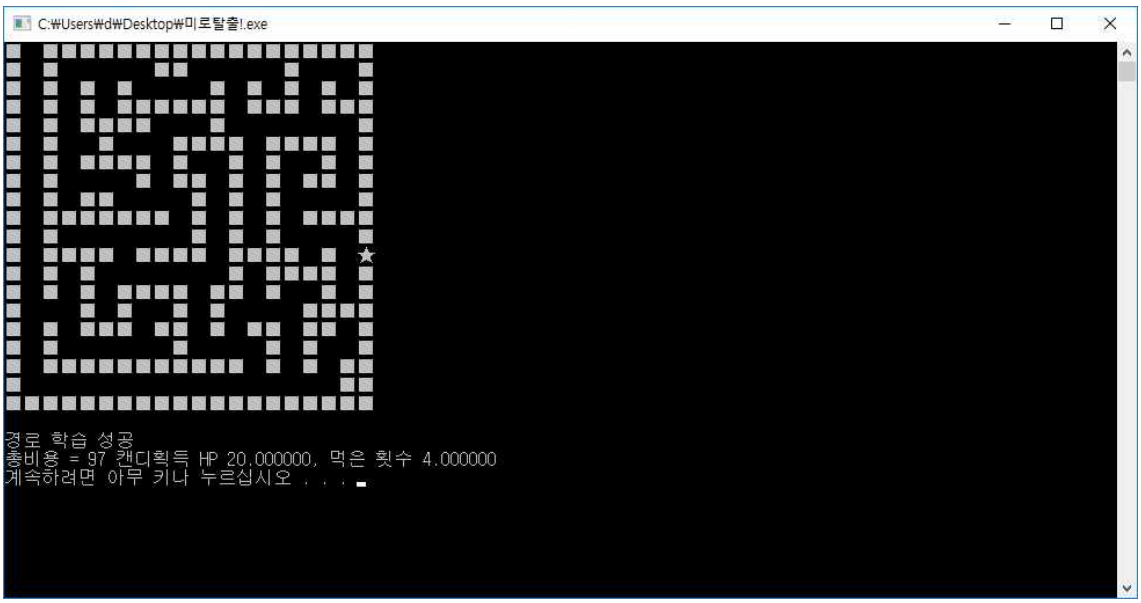
7주 최종 보고서 작성`

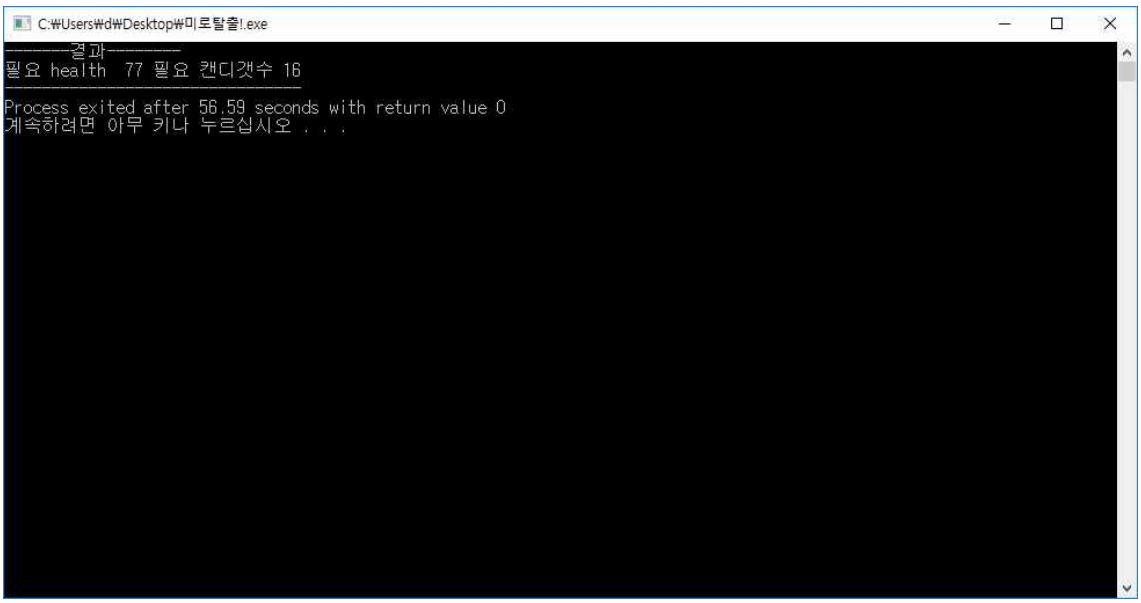
## 5. 선정이유 및 구현 (좀더 전공과 연관시킨 스토리를 가지고

앞서 위에서 알고리즘 성능평가에있어 구분되는 3가지 조건중에 안정성에 중점사항을 잡았습니다 왜냐하면 CPU와 메모리의 실행속도나 크기 비약적인 발전하고있는 현재에 더 이상 알고리즘 처리속도의 중요도가 떨어졌다고 생각합니다 예를 들어보자면 병원같은 데 사용할 제품의 알고리즘을 선택한고 하였을때 병원 아무리 속도가 빠르다고 하더라도 환자의 생명과 연관된 제품을 많이쓰기 때문에 동일 상황에대한 에러율이 많거나 다른 결과값을 출력하게된다면 위험한 상황이 올수도있습니다. 그래서 저희는 알고리즘의 안전성을 생각하여 다음과 같은 알고리즘을 선택하였습니다.

## 6. 실행 및 테스트




## 7. 프로젝트 자체 평가(팀원 포함) 및 느낀 점(개인적 경험을 바탕으로 프로젝트를 완료시 느낀점에 대하여 기술)

최단 단순하고 쉽게 프로그램을 만드는 것이 편해서 개발자 중심으로 프로그램을 작성해서 사용자(소비자)의 Needs를 분석할 수 있는 능력이 많이 부족했었다 그래서 항상 프로젝트를 했을 때 지적받았던 사항이 사전조사 부분과 문제정의부분에 많은 지적을 받았었는데 사용자(소비자)의 Needs를 분석하고 만든 프로그램에대한 피드백을 팀원끼리 의논하여 GUI 중심의 알고리즘 프로젝트를 진행하여서 다음번 프로젝트에서는 팀원들 과의 발전된 의사소통능력을 통하여 문제 정의 과정을 좀더 세분화 시킬 수 있는 능력을 함양한것같다.

## 8. 부록

부록 문서 참조(스택, 백트래킹)

**출처:**

https://ko.wikipedia.org/wiki/%EB%8D%B0%EC%9D%B4%ED%81%AC%EC%8A%A4%ED%8A%B8%EB%9D%BC_%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98
https://ko.wikipedia.org/wiki/%EA%B9%8A%EC%9D%B4_%EC%9A%B0%EC%84%A0_%ED%83%90%EC%83%89
http://monsieursongsong.tistory.com/4
http://manducku.tistory.com/24?category=683258 [Manducku`s Code]