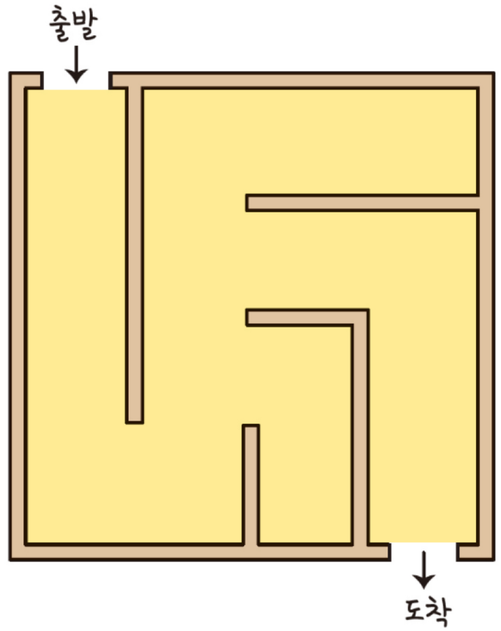
미로 찾기 알고리즘

모두의 알고리즘 with 파이썬 - 문제 16

20.06.14 장예훈

문제 분석과 모델링

다음 그림과 같이 미로의 형태와 출발점과 도착점이 주어졌을 때 출발점에서 도착점까지 가기 위한 최단 경로를 찾는 알고리즘을 만들어 보세요.



🤷 사람에게는 정말 쉬운 문제이지만 컴퓨터에게 풀게 하려면 어떻게 해야 할까?

문제 분석과 모델링

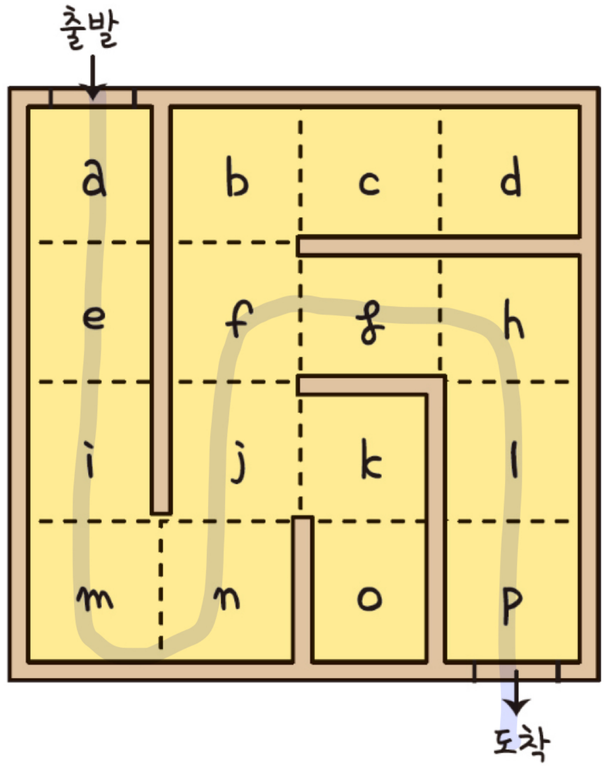
모델링(모형화)

- 주어진 현실의 문제를 정형화하거나 단순화하여 수학이나 컴퓨터 프로그램으로 쉽게 설명 할 수 있도록 다시 표현한 것
- 수학식이나 프로그래밍 언어로 번역하는 절차

문제 분석과 모델링

1단계

출발점 a에서 시작하여 벽으로 막히지 않은 위치로 차례로 이동하여 도착점 p에 이르는 가장 짧은 경로를 구하고, 그 과정에서 지나간 위치의 이름을 출력해보세요

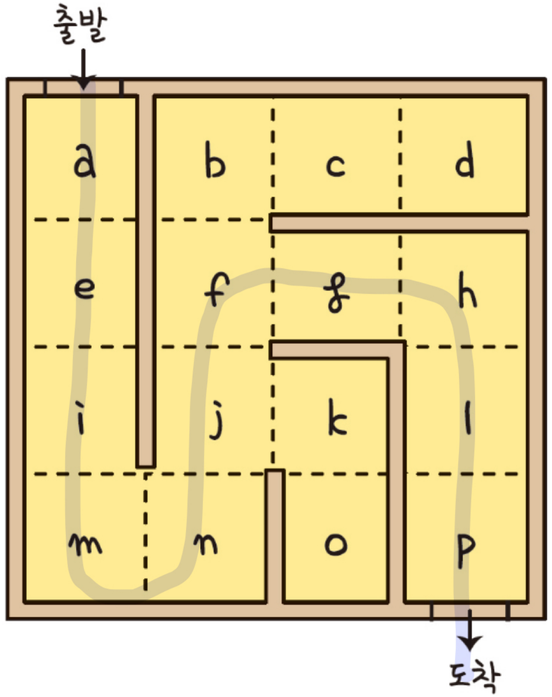


문제 분석과 모델링

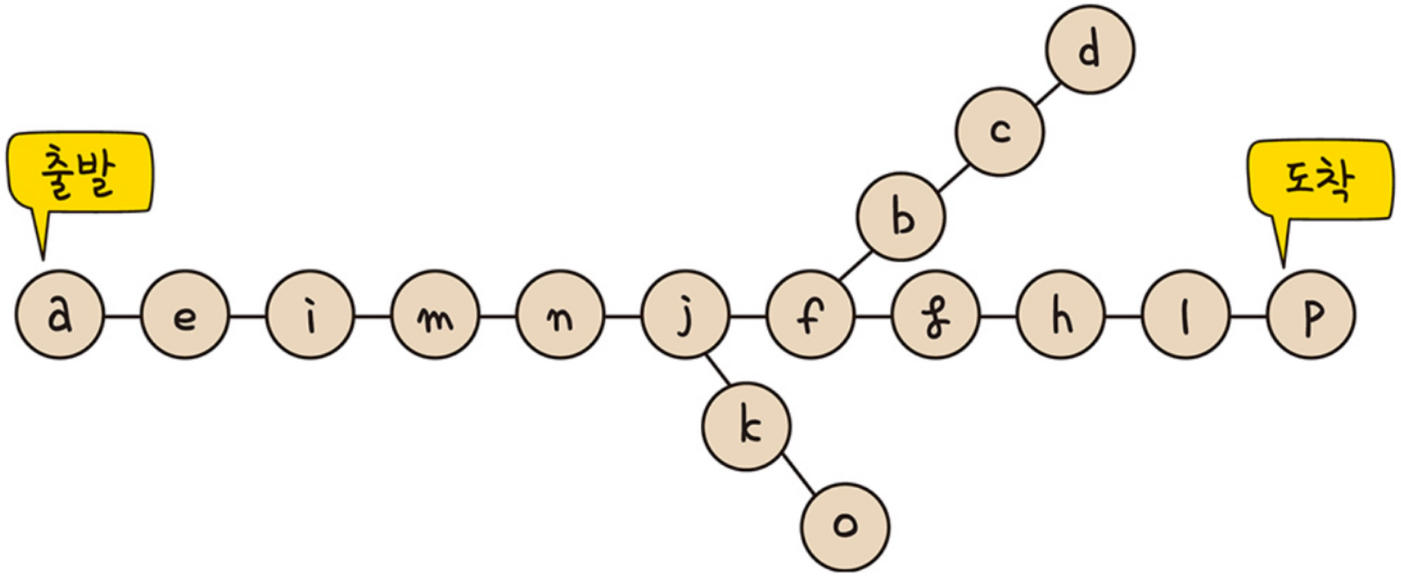
2단계

각 위치 사이의 관계 정의

➡️ 위치 열여섯개를 각각 꼭짓점으로 지정, 이동할 수 있는 이웃한 위치를 모두 선으로 연결



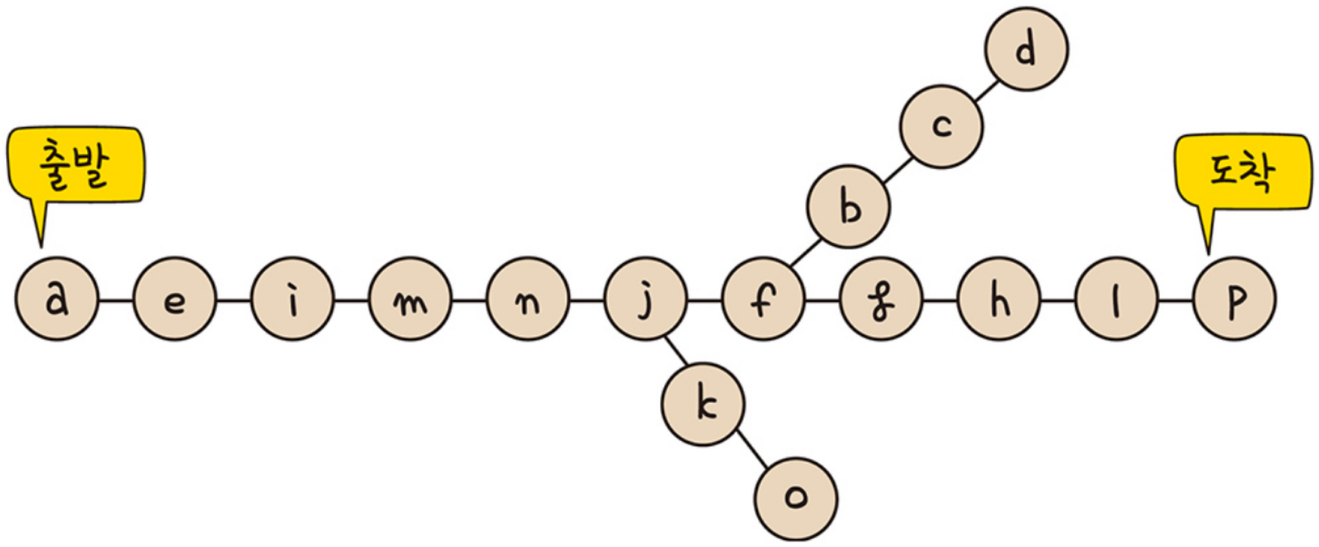
➡️



문제 분석과 모델링

3단계

그래프를 딕셔너리로 전환



➡️

```
maze = {
    'a': ['e'],
    'b': ['c', 'f'],
    'c': ['b', 'd'],
    'd': ['c'],
    'e': ['a', 'i'],
    'f': ['b', 'g', 'j'],
    'g': ['f', 'h'],
    'h': ['g', 'l'],
    'i': ['e', 'm'],
    'j': ['f', 'k', 'n'],
    'k': ['j', 'o'],
    'l': ['h', 'p'],
    'm': ['i', 'n'],
    'n': ['m', 'j'],
    'o': ['k'],
    'p': ['l']
}
```

미로 찾기 알고리즘

입력: 미로 정보 g, 출발점 start, 도착점 end / 출력: 이동경로 or 나갈 수 없으면 “?”

```
1  def solve_maze(g, start, end):
2      qu = []
3      done = set()
4
5      qu.append(start)
6      done.add(start)
7
8      while qu:
9          p = qu.pop(0)
10         v = p[-1]
11         if v==end:
12             return p
13         for x in g[v]:
14             if x not in done:
15                 qu.append(p+x)
16                 done.add(x)
17     return "?"
```

```
solve_maze(maze, 'a', 'p')
```

```
'aeimnjfghlp'
```

line 2: 기억장소 #1 - 앞으로 처리해야 할 이동 경로를 큐에 저장

line 3: 기억장소 #2 - 이미 큐에 추가한 꼭짓점들을 집합에 기록(중복 방지)

line 5: 출발점을 큐에 넣고 시작

line 6: 집합에도 추가

line 10: 큐에 저장된 이동 경로의 마지막 문자가 현재 처리해야 할 꼭짓점

line 11-12: 처리해야할 꼭짓점이 도착점이면 이동경로 출력하고 종료

line 13: 대상 꼭짓점에 연결된 꼭짓점들 중에

line 14: 아직 큐에 추가된 적이 없는 꼭짓점을

line 15: 이동 경로에 새 꼭짓점으로 추가하여 큐에 저장하고

line 16: 집합에도 추가

line 17: 탐색을 마칠 때까지 도착점이 나오지 않으면 나갈 수 없는 미로

미로 찾기 알고리즘

입력: 미로 정보 g, 출발점 start, 도착점 end / 출력: 이동경로 or 나갈 수 없으면 “?”

```python
def solve_maze(g, start, end):
    qu = []
    done = set()

    qu.append(start)
    done.add(start)

    while qu:
        p = qu.pop(0)
        v = p[-1]
        if v==end:
            return p
        for x in g[v]:
            if x not in done:
                qu.append(p+x)
                done.add(x)
    return "?"
```

```python
maze = {
    'a': ['e'],
    'b': ['c', 'f'],
    'c': ['b', 'd'],
    'd': ['c'],
    'e': ['a', 'i'],
    'f': ['b', 'g', 'j'],
    'g': ['f', 'h'],
    'h': ['g', 'l'],
    'i': ['e', 'm'],
    'j': ['f', 'k', 'n'],
    'k': ['j', 'o'],
    'l': ['h', 'p'],
    'm': ['i', 'n'],
    'n': ['m', 'j'],
    'o': ['k'],
    'p': ['l']
}
```

```python
solve_maze(maze, 'a', 'p')
```

```
'aeimnjfghlp'
```

```
qu: ['a']
done: {'a'}
p: a
v: a
------------
x: e
qu: ['ae']
done: {'e', 'a'}
========================
qu: ['ae']
done: {'e', 'a'}
p: ae
v: e
------------
x: a
------------
x: i
qu: ['aei']
done: {'i', 'e', 'a'}
========================
qu: ['aei']
done: {'i', 'e', 'a'}
p: aei
v: i
------------
x: e
------------
x: m
qu: ['aeim']
done: {'m', 'i', 'e', 'a'}
```

감사합니다 😊