대구대학교 DAEGU UNIVERSITY

일상에서 찾아보는

자료구조의 원리

04주차 순서대로 처리하기 - 큐

3차시

파이썬 코딩:
원형 큐를 이용한 미로 탐색

학습목표

» 너비우선탐색을 구현하여 큐의 사용법을 설명할 수 있다.

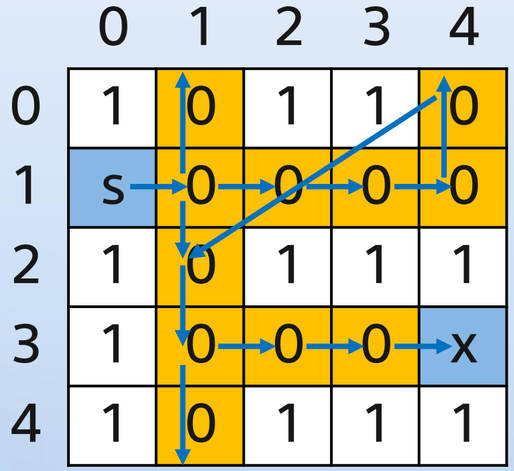
학습내용

- 큐의 응용: 너비 우선 탐색

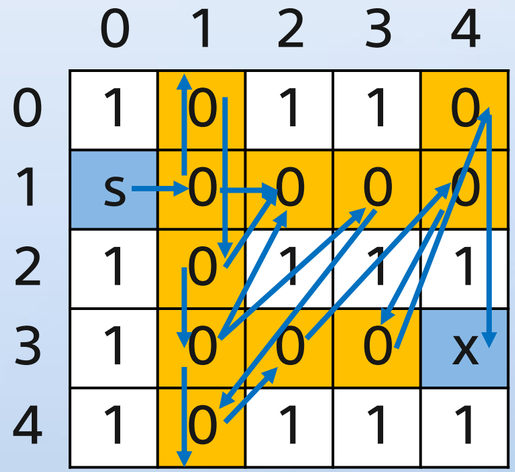
큐의 응용: 너비 우선 탐색

» 탐색 방법

- 깊이 우선 탐색(DFS, Depth First Search) – 스택을 이용
- 너비 우선 탐색(BFS, Breadth First Search) – 큐를 이용



스택은 반대의 순서로 길을 따라
나와서 미로를 탐색: LIFO



큐는 현재 위치에서 주변의 가까운 길을
모두 먼저 탐색: FIFO

큐의 응용: 너비 우선 탐색

» 미로 탐색

- 큐를 이용한 너비 우선 탐색(BFS, Breadth First Search)
- 현재 위치에서 인접한 위치들을 모두 탐색함

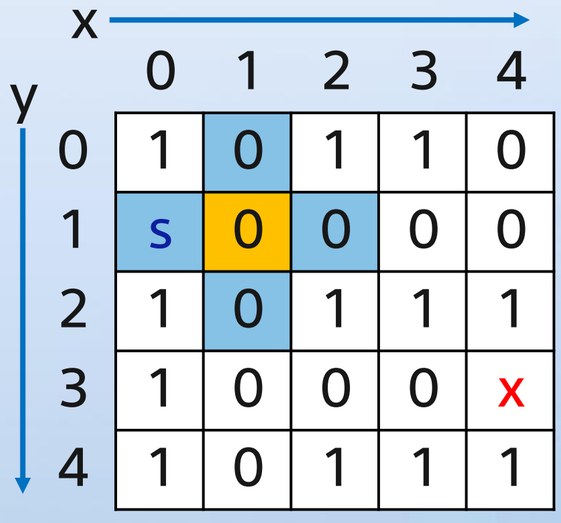
출발 점을 큐에 삽입

↓

큐가 비어 있지 않으면
삭제하여 위치로 이동

↓

현재 위치에서 이동할 수
있는 모든 지점 삽입



큐의 응용: 너비 우선 탐색

BFS 함수

- start 위치에서 시작함
- CircularQueue 큐 객체를 생성함
- 큐가 비어 있지 않은 동안 반복문을 수행
  - dequeue를 사용하여 큐의 front에서 현재 위치를 가져옴
  - 현재 위치가 도착점('x')인지 확인, 도착점이면 "탈출 성공"을 반환함
  - 현재 위치를 탐색했음을 '*'로 표시함
  - 네 방향의 주변으로 이동 가능한 위치를 큐에 삽입함

큐의 응용: 너비 우선 탐색

» BFS 함수

```
def BFS(start, maze):
    queue = CircularQueue()  # 원형 큐 객체
    queue.enqueue(start)  # 출발 위치 삽입
    # 네 방향 (상, 하, 좌, 우)
    directions = [(0, -1), (0, 1), (-1, 0), (1, 0)]
    print("출발", start)

    while not queue.isEmpty():
        position = queue.dequeue()  # 큐에서 현재 위치 확인
        print("현재 위치", position, end=' -> ')
        if position is None:
            continue
        x, y = position
        if maze[y][x] == 'x':  # 현재 위치가 도착점이면
            return "탈출 성공"
        maze[y][x] = '*'  # 방문 여부를 표시
        for direction in directions:  # 네 방향으로 이동 가능 지점 확인
            next_x, next_y = x + direction[0], y + direction[1]
            if (0 <= next_x < len(maze[0])) and
                                (0 <= next_y < len(maze)) ₩
                    and maze[next_y][next_x] in ['0', 'x']:
                queue.enqueue((next_x, next_y))  # 이동할 수 있는
지점이면 큐에 추가
        # 큐 출력해 보기
        queue.display()
        # 미로 출력해 보기
        for row in maze:
            for cell in row:
                print(cell, end=' ')
            print()
        print()
    return "탈출 실패"
```

큐의 응용: 너비 우선 탐색

» 테스트

```
# 미로 설정 (0: 경로, 1: 벽, x: 도착점)
maze = [
    ['1', '0', '1', '1', '0'],
    ['0', '0', '0', '0', '0'],
    ['1', '0', '1', '1', '1'],
    ['1', '0', '0', '0', 'x'],
    ['1', '0', '1', '1', '1']
]

start = (0, 1)  # 출발 지점

# 미로 탐색 실행
result = BFS(start, maze)

print(result)
```

학습정리

너비 우선 탐색

- 큐는 선입선출의 원리로 동작하는 자료구조임
  - 운영체제의 스케줄링 큐는 CPU와 기타 자원의 효율적인 관리를 사용함
- 원형 큐는 큐의 크기를 고정하여 메모리를 효율적으로 사용함
- 너비 우선 탐색(BFS: Breadth-First Search)을 위해 큐를 사용함
  - 경로를 찾는데 유용하게 사용함