BFS, DFS 구현 보고서

201504280
컴퓨터공학과 신윤호

1. 구현 내용 :: BFS

1) Vertex 클래스 및 초기화

Vertex 클래스는 각 정점에 대한 정보를 가지는 클래스이다. 이 클래스는 id, color, distance, parent 필드를 가지도록 구현했다. 생성된 Vertex 객체들은 vertices 배열에 저장되어 관리된다. 가령 verices[0]에는 0번 Vertex객체가 저장되도록 했다.

그래프는 인접 행렬 방식으로 표현했다. 과제에 주어진 노드들 중 시작정점 s를 0번으로 하여 7번 노드까지, 총 8*8크기의 2차원 배열을 사용했다. 한편, 큐는 우선순위를 고려하지 않는 큐 이므로 그냥 자바에서 제공하는 Linked List를 큐로 사용했다.

2) runBFS메소드

BFS알고리즘을 수행하는 메소드다. BFS 알고리즘은 시작 정점에서 가까운 정점을 먼저 탐색하고 멀리 떨어진 정점을 나중에 탐색하는 방식으로 탐색이 진행된다. 즉, 먼저 거리가 d만큼 떨어진 정점들을 모두 탐색한 뒤, (d+1)만큼 거리가 뜰어진 정점들을 탐색하는 방식이다.

runBFS 메소드는 먼저 시작 정점을 초기화하고 큐에 삽입하면서 시작된다. 즉, 시작 정점 s의 탐색여부를 GRAY로 설정해 거리 d는 0, parent는 null로 초기화한 뒤 큐에 포함시킨다. 여기까지를 초기화 과정으로 볼 수 있다. 이 이후에는 다음과 같은 방식으로 알고리즘이 진행되도록 했다.

① 큐에서 정점 하나를 출력한다.
② 해당 정점을 기준으로, 인접한 정점의 탐색 여부를 확인한다.
   만약, 아직 탐색되지 않은 정점을 만난 경우 해당 정점의 탐색 여부를 GRAY로 하고 큐에 삽입한다.
③ 인접 정점의 탐색을 마쳤으면 해당 정점을 BLACK으로 한다.
④ ①~③을 반복한다.

위의 과정에서 ②번 과정을 수행할 때, 인접 정점인지 확인하는 과정은 isAdj메소드를 정의해서 구현했다. isAdj 메소드는 정점 u와 v를 인자로 받은 뒤, 인접 행렬을 조사해 두 정점이 인접한 정점이면 true를 반환하고 그렇지 않으면 false를 반환하는 메소드다. 즉, isAdj(u,v) && v.color.equals("WHITE")는 인접한 정점이면서 아직 탐색되지 않은 정점일 때를 구현한 것이다.

한편, while문 내부의 for문이 모두 수행되면 정점 u에 대해 인접한 정점이 모두 탐색되었음이 보장된다. 따라서 u의 색깔을 BLACK으로 해 주고, 큐에서 정점 하나를 출력해 다시 반복을 수행한다. 이 반복은 큐가 텅 빌 때 까지 진행된다.

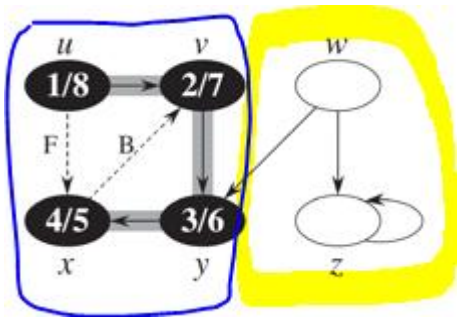
2. 구현 내용 :: DFS

1) Vertex 클래스 및 초기화

BFS의 Vertex클래스와 유사하며 추가적으로 d와 f필드를 가지고 있다. 과제에서 제시된대로 d 필드는 정점 u가 처음 발견된 시점을 갖게 되며 f필드는 정점 u가 탐색을 마친 시점을 갖도록 구현했다. 그 외에 그래프 표현에 인접 행렬을 사용했다는 점 등은 BFS와 같다.

2) dfs메소드

dfs메소드는 존재하는 각 정점에 대해, 만약 해당 정점이 WHITE인 경우 해당 정점을 기준을 DFS탐색을 시작하도록 한다. 어떤 정점 s를 기준으로 DFS를 수행한 뒤에도 여전히 WHITE인 정점은 s에서 시작할 때에는 도달할 수 없는 정점이다. 즉 다음과 같은 상황이라고 할 수 있다.



이 경우 u에서 출발했을 때 w나 z는 도달할 수 없는 정점이다. dfs메소드는 위 상황일 때 w를 기준으로 DFS를 수행하도록 하기 때문에 w와 z를 발견할 수 있게 해 준다.

3) dfsVisit메소드

실질적인 DFS가 진행되는 메소드다. BFS가 큐와 반복문을 주로 이용한 반면 DFS는 recursion을 이용해 알고리즘이 전개된다. 먼저 시작 정점인 u는 현재 탐색한 것이므로 색을 GRAY로 만들어준다. 그리고 현재 탐색 시간을 카운트해 준다. 다음으로는 u에 인접한 정점들 중 아직 방문하지 않은 정점, 즉 인접 정점중 색이 WHITE인 정점에 dfsVisit 메소드를 재귀 호출한다. 메소드가 재귀 호출되면 해당 정점을 기준으로 DFS가 재귀적으로 수행된다. 가령 u를 기준으로 시작하여 v에 대해 dfsVisit메소드가 호출된다면 v를 시작점으로 DFS가 시작되는 것이다.

이런 방식으로 도달할 수 있는 모든 정점을 탐색한 뒤, 더 이상 탐색할 수 있는 정점이 없다면 자기 자신을 호출한 메소드로 백 트래킹 한다. 백 트래킹이 시작되었다는 것은 해당 정점을 기준으로 인접한 정점을 모두 탐색했다는 뜻이므로 해당 정점의 색을 BLACK으로 바꾸어준다. 그리고 time을 카운트한 뒤, 해당 정점의 f필드에 이를 저장해준다. 이런 식으로 맨 처음 호출되었던 정점 u까지 백트래킹이 이루어지면 한 단위의 dfsVisit이 종료되었다고 할 수 있다.

만약 아직 탐색되지 않은 정점이 있다면 해당 노드는 u를 시작점으로는 도달할 수 없는 정점이다. 해당 정점은 dfs에서 그 정점을 기준으로 다시 dfsVisit을 시작하도록 함으로써 탐색된다.

3. 결과 화면

1) BFS

```
정점    부모정점    비용
--------------------------------
s   s   0
r   s   1
t   w   2
u   t   3
v   r   2
w   s   1
x   w   2
y   x   3

Process finished with exit code 0
```

2) DFS

```
정점    부모정점    발견시간    탐색완료시간
--------------------------------
u   u   1   8
v   u   2   7
w   u   9   12
x   y   4   5
y   v   3   6
z   w   10  11

Process finished with exit code 0
```