강의 13: 그래프 I: 너비 우선 탐색

강의 개요

- 그래프 탐색의 응용
- 그래프 표현
- 너비 우선 탐색

기억해야 할 점:

그래프 $G = (V,E)$

- V = 정점의 집합 (임의의 레이블)
- E = 간선의 집합 i.e. 정점의 쌍 (v,w)
  - 순서쌍 ⇒ 그래프의 방향이 있는 간선
  - 비순서쌍 ⇒ 무방향

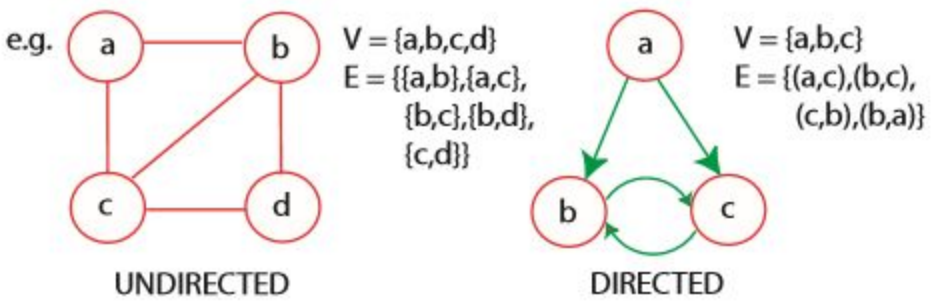


그림 1: 그래프 용어를 설명하기 위한 예제

그래프 탐색

"그래프를 탐험하다", 예시:

- 시작 점 s에서 원하는 정점으로의 경로를 찾는다
- 그래프의 또는 s에서 도달할 수 있는 모든 정점과 간선을 방문한다

응용:

여러 응용 사례들이 있습니다.

- 웹 크롤링 (구글이 페이지를 찾는 방법)
- 소셜 네트워킹 (페이스북이 친구 찾기를 사용하는 법)
- 네트워크 브로드캐스트 라우팅
- 가비지 컬렉션
- 모델 검사 (무한 상태 기계)
- 수학적 추측 확인하기
- 퍼즐이나 게임 풀기

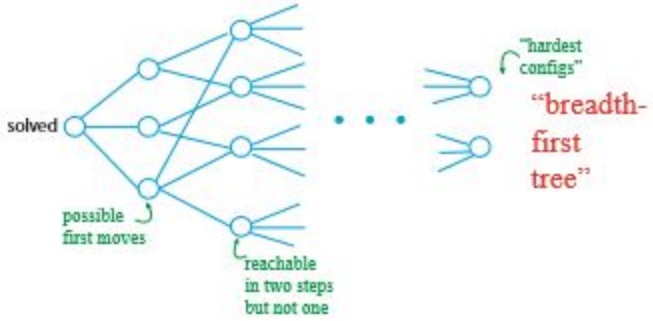
포켓 큐브:

2 x 2 x 2 루빅 큐브

짜임새 그래프:

- 가능한 각 상태에 대한 정점
- 한 상태에서 다른 상태로의 기본 이동에 대한 간선 (예: 90도 회전)
- 무방향: 거꾸로 움직일 수 있음

지름("신의 숫자")
2×2×2의 경우 11, 3×3×3의 경우 20, n×n×n의 경우 Θ(n2/lgn) [Demaine, Demaine, Eisenstat Lubiw Winslow 2011]



정점의 수 = $8! \cdot 38 = 264{,}539{,}520$ $8!$ 은 임의의 위치에서의 8개의 큐브릿을 의미하고 38은 각각의 큐브릿을 세 번 돌릴 수 있다는 것을 의미한다.

만약 큐브의 대칭성을 없애면 24로 나눌 수 있고 실제로 도달할 수 있는 곳을 생각하면 3으로 또 나눠줄 수 있다 (3개의 연결된 구성 요소가 있다).

그래프 표현: (자료구조)

인접 리스트:

$|V|$ 연결 리스트의 배열 Adj

- 각 정점 $u \in V$에 대해, $Adj[u]$ 는 u의 이웃들을 저장한다, 예: $\{v \in V \mid (u,v) \in E\}$. 방향 그래프에서 (u,v)는 나가는 간선이다. (그림 2 참조.)

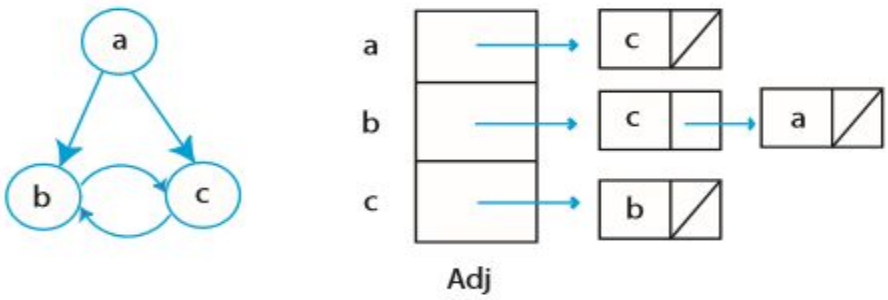


그림 2: 인접 리스트 표현: 공간 $\Theta(V + E)$

- 파이썬: Adj = 리스트/집합 값의 딕셔너리; 정점 = 해시 가능한 것 (예: int, tuple)
- 장점: 같은 정점에 여러 그래프가 있을 수 있음

묵시적으로 표현된 그래프:

$\text{Adj}(u)$ 는 함수이다 — 로컬 구조를 계산한다 (예: 루빅 큐브). "제로" 공간이 필요하다.

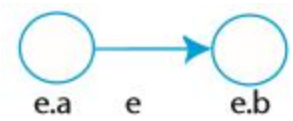
객체 지향 변형:

- 각 정점 u에 대한 객체
- u.neighbors = 이웃한 점들의 리스트 (예: $Adj[u]$)

즉, 이것은 묵시적으로 표현된 그래프를 위한 방법이다

입력 목록:

- 간선도 객체로 만들 수 있다



- u.edges = u에서 (나가는) 간선의 리스트
- 장점: 해싱 없이도 간선을 저장할 수 있다

너비 우선 탐색

s에서 레벨별로 그래프를 탐색한다

- level 0 = $\{s\}$
- level i = 최소 i개의 간선의 경로로 도달할 수 있는 정점

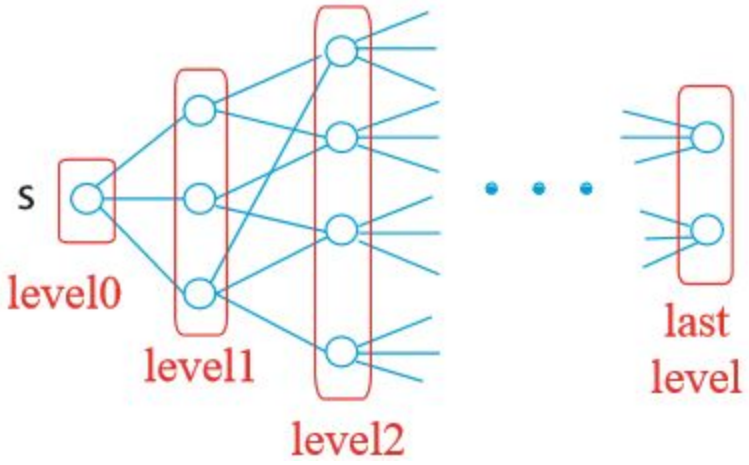


그림 3: 너비 우선 탐색의 설명

- 모든 나가는 간선을 사용해 level $i-1$ 에서 level $i > 0$를 만든다 하지만 이전

level의 정점은 무시한다

너비 우선 탐색의 알고리즘

```
BFS (V,Adj,s):                          See CLRS for queue-based implementation
    level = { s: 0 }
    parent = {s : None }
    i = 1
    frontier = [s]                       # previous level, i − 1
    while frontier:
        next = [ ]                       # next level, i
        for u in frontier:
            for v in Adj[u]:
                if v not in level:       # not yet seen
                    level[v] = i         # = level[u] + 1
                    parent[v] = u
                    next.append(v)
        frontier = next
        i += 1
```

예시

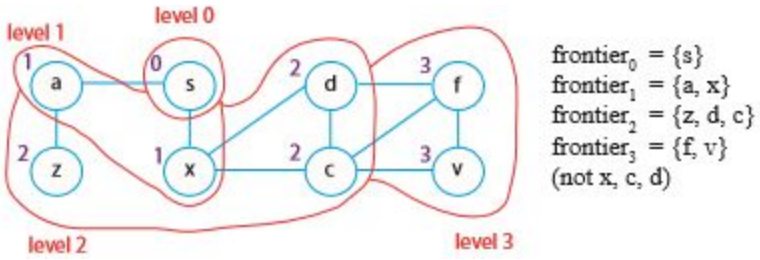


그림 4: 너비 우선 탐색의 Frontier

분석:

- 정점 V가 next를 (다음으로는 frontier를) 한 번만 방문한다
  (level[v] 가 정의되므로)
  기본 경우: $v = s$
- ⇒ Adj[v] 는 루프를 한 번만 통과한다

$$\text{time} = \sum_{v \in V} |Adj[V]| = \begin{cases} |E| \text{ for directed graphs} \\ 2|E| \text{ for undirected graphs} \end{cases}$$

- $\Rightarrow O(E)$ 시간

- $O(V+E)$ ("선형 시간") v에서 도달할 수 없는 정점을 리스트한다 (level이 지정되지 않은 정점)

최단 경로:

참조. L15-18

- 각 정점 v에 대해, s에서 v로 갈 수 있는 가장 적은 수의 간선은

$$\begin{cases} \text{level}[v] \text{ if } v \text{ assigned level} \\ \infty \quad \text{else (no path)} \end{cases}$$

- 부모 포인터는 최단 경로 트리를 형성한다 = 각 v에 대한 최단 경로의 조합
  $\Rightarrow$ 최단 경로를 찾으려면 v를 찾고, parent[v]를 찾고, parent[parent[v]]를 찾아 s에 도달하거나 끝날 때까지 찾는다