그래프의 기본과 탐색

2020년 5월 16일 토요일 오전 12:04

간선의 배열 간선(시작정점, 끝 정점)을 배열에 연속적으로 저장

① 무향그래프
서로 대칭적인 관계로
연결되어 나타낸 그래프

인접행렬 두 정점을 연결하는 간선의 유무를 행렬로 표현
VxV 정방행렬
두 정점이 인접 1, 인접x 0

② 유향그래프 (대칭x)
간선을 화살표로 표현
방향성의 개념 포함됨

무향그래프 1의개수=2*간선수
i번째 행의 합=i번째 열의합=Vi의 차수
유향그래프
i번째 행의 합=Vi의 진출 차수
i번째 열의 합=Vi의 진입 차수

③ 가중치 그래프
이동하는데 드는 비용을 간선에 부여한
그래프

인접행렬의 단점
1. 입력의 크기가 커지면 n제곱 만큼의 메모리가 필요
2. 인접정점을 찾을 때 마다 1000개의 슬롯조사
3. 정점의 개수에 비례하여 시간복잡도 증가
   ->간선들의 정보를 나열하여 저장하면 메모리 사용을 줄일 수 있음

해결방법
인접리스트 사용
각 정점에 인접한 요소만 리스트에 넣음

0 [ ]→[6]→[2]
1 [2]→[5]→[4]
불필요한 메모리, 비용감소

무향 그래프 ① 노드수=간선수
② 각 정점의 노드수= 정점에 연결된 간선수

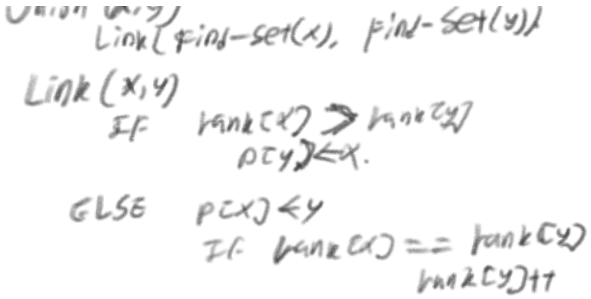
·그래프순회
비선형구조인 그래프로 표현된 모든 자료(정점)를
빠짐없이 탐색

① DFS (깊이 우선 탐색)
v 선택된 정점에서 아직 방문하지 않은 다음
정점을 방문
v 탐색시 방문했던 부모 노드로
돌아옴
v 재귀호출을 통하여 구현.
④ 시작점의 인접한 정점들을
먼저 모두 차례대로 방문
다시 그 시작점으로 부터 아직
방문하지 않은 정점을 방문

② BFS
v 선택된 정점에서 아직 방문하지
않은 다음 노드를 방문
방문했던 정점을 다시 시작점으로하여
계속해서 방문을 수행함
v 이미 방문한 정점은 재방문하지 않는다.
v 인접한 정점들에 대해 탐색후
차례로 너비우선탐색
→ FIFO 형태인 Queue를 활용

상호 배타 집합들 (서로소)
1. find-set
2. Make-set
3. Union
표현 방법 ① 연결리스트
② 트리 → 더 좋은 방법
인것 같다. ↗ 직관적이고
이해가 쉽다.

① Make-set (x) 연산
p[x] ← 0
rank[x] ← 0
② find-set (x)
If x ≠ p[x]
p[x] ← find-set(p[x])
return p[x]
③ Union 연산.
Union (x,y)
Link(find-set(x), find-set(y))
Link (x,y)



2. Find-set은 .find STL을 이용할 수 있다.
3. Union은 c++에서 set_union STL을 이용할 수 있다.

요약:
dfs는 재귀
bfs는 큐를 이용한다.
방문여부