Graph Traversal

Any-Order Traversal

시작노드 S를 백스에 넣는다.

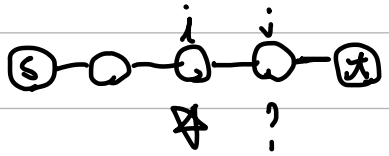
백스가 비어있지 않을 때 까지 ...

- 백스에 있는 한 노드를 꺼낸다.

백스가 - Stack : DFS

- Queue : BFS
- Priority Queue : edge weight : Prim
  S로부터의 거리 : Dijkstra
  Arbitary Function : A*

- Reachability



S부터 t까지 길이 있다면, t는 마킹된다.

i에 인접한 노드 j는 백스에 들어가므로 j도 마킹되었다.

i는 마킹되어있고 j가 마킹 안되어있을 경우는 없다.

O(n+m)

- Connected Component

  reachable한 노드의 집합

- 파라메틱 서치

모든 edge로 접근하고, 중간값 이하로만 갈 수 있는 길이 있나?

yes면 답을 올려본다.

no면 답을 줄게 해본다.

$O((n+m)\log m)$

Event Queue

마지막으로 몇 개가 감염될 것인가?

감염이 되는 칸들을 Queue에 넣고 하나씩 꺼내서 감염시킴.

↓

주변 칸에만 새로 감염되어야 할 칸이 생김. 이걸 Queue에 넣는다.

Queue에 최대 $n \times m$ 개가 들어갈 수 있으므로 $O(n \cdot m)$ 걸림.

Recursive DFS

임의의 노드 S

노드 S를 마킹.

set Preorder # Pre(S) = 1 (위치이 적음)

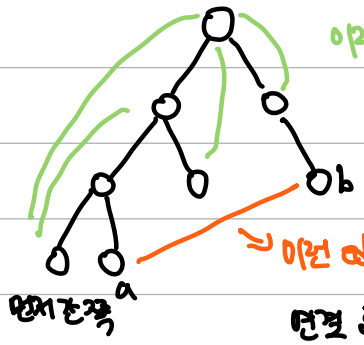
모든 인접한 노드 u에 대해서 마킹 안된 애들에 대해 RDFS(u)를 호출.

set postorder #

- DFS Tree

DFS를 하면서 전진한 edge만 추가해서 만든 tree

안들어간 edge를 추가할 때



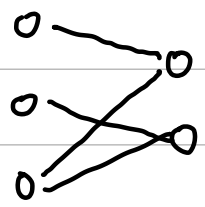
이런 식이라 애들만 가능함. (후진해서 갈 수 있는 곳 까지만)

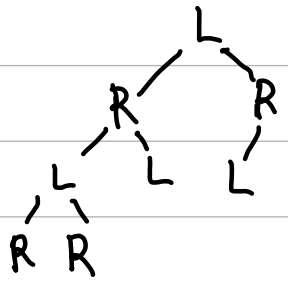
back edge (tree에 있는 edge 말고 나머지)

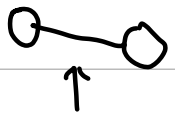
↘ 이런 edge는 안됨.

만약 되어있다면 a 갔을 때 b도 추가했어야 함.

Bipartite Graph Detection





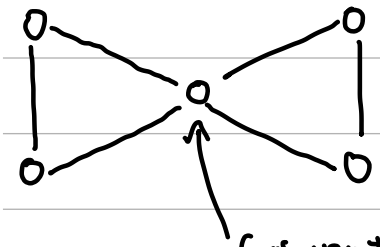


이 두 노드는 같은 쪽에 있을수 X.

이런 애들 중에 back edge 확인.

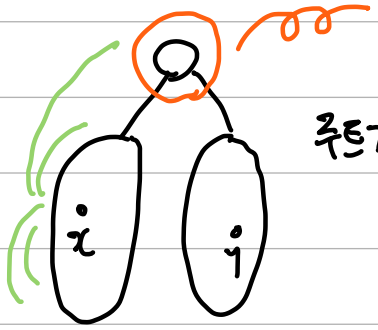
Cut Vertex

임의의 x 부터 y까지의 모든 패스에 노드 s가 있다면 s는 Cut Vertex.



Cut vertex가 leaf가 될 수 없다. tree만 봐도 Cut Vertex가 아님.

DFS Tree를 만들고, 루트의 child가 1개 초과면 iff Cut Vertex이다.



루트가 제거되었을 때 x, y 사이 길이 없어짐.

가능한 back edge.

루트가 아닌애들은?

모든 노드 x 에 대해 $A(x)$ 값을 구한다. (A값은 작을수록 위쪽)

$A(x) = \min$ {
- $Pre(x)$
- min of $Pre(s)$ — (x,s)가 backedge인 모든 s에 대해서
- min of $A(u)$ — x의 모든 자식노드들에 대해서

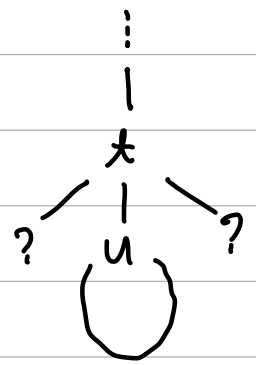
}

루트가 x인 서브트리에서 back edge를 타고 갈 수 있는 제일 높은곳.

x는 Cut Vertex iff $L(u) \geq pre(x)$ (하나라도)

↓

back edge를 타고 최대 x까지밖에 못간다.



L 값을 자연스럽게 밑에서부터 계산해올라오면 됨.

Cut edge ?

→ 모든 edge에 빨간도스를 넣고 Cut Vertex를 풀면 그냥 풀림.