<Discrete Mathematics>

작성자_2018320161_송대선

작성일_05_28

abstract recursive theorem

1. ∑ is closed under composition (∑ is a set of functions: $F(x, y): Z^+ \times Z^+ \rightarrow Z^+$)
2. F(x, x)∈∑
3. $\forall_{f \in \Sigma} \exists_{a \in Z^+} \forall_{x \in Z^+} E_{F(a, x)} = E_{f(x)}$

---

4. $\forall_{f \in \Sigma} \exists_{i \in Z^+} E_i = E_{f(x)}$

proof)

$\forall_{f \in \Sigma} \exists_{a \in Z^+} \forall_{x \in Z^+} E_{F(a, x)} = E_{f(x)}$

the function f is arbitrary

-> $E_{F(a, x)} = E_{f(F(x, x))}$

-> $E_{F(a, a)} = E_{f(F(a, a))}$

-> $E_i = E_{f(i)}$

abstract recursive theorem은 sound한 argument이다.

두 집합 A, ∑에 대하여,

A를 the set of all programs written in Java(C, Python... (Turing Complete Programs))라고 하고,

∑를 the set of all computable functions $Z^+ \rightarrow Z^+$라고 하면,

즉, 두 집합 A, ∑를 그렇게 잡아버리면 Sound해진다.

-> $E_{F(a, x)} = E_{f(x)}$이 무조건 발생

-> 정확히 하는 일이 같은 크기가 서로 다른 두 프로그램이 존재한다. -> 이걸 피할 수 없다.

-a quine- :self-pointing program

a quine과 abstract recursive theorem의 관계는 무엇인가

<Chapter 10> : Graphs

G=(V, E)

a graph:

1. a directed graph
2. a undirected graph

V: a finite set of nodes

E: a subdet of VxV

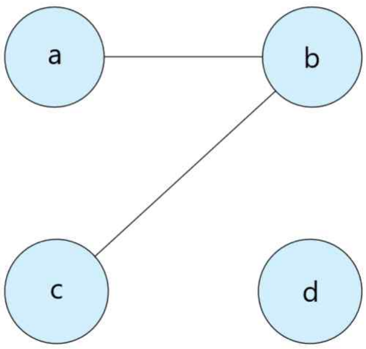
-> a binary relation on V

(undirected graph의 경우, edge는 a subset of unordered pairs of graph)

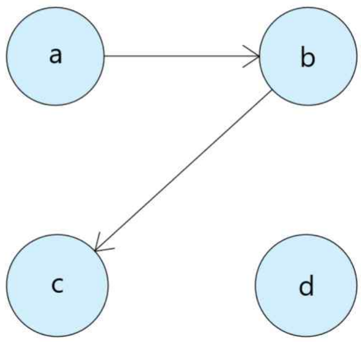
-transitive closure-

1. relation
2. directed graph

ex) V={a, b, c, d}, E={(a, b), (b, c)}



undirected graph

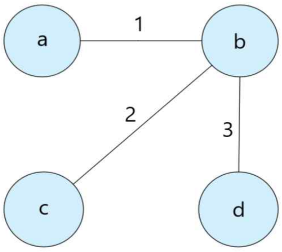


directed graph

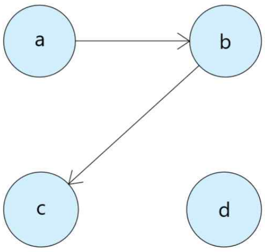
-adjacency/incidence-

Adjacency/Adjacent -> both directed & undirected에 사용

Incidence/incident -> only for undirected에 사용



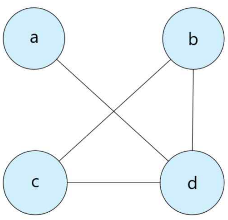
edge 1 is incident on(upon) a and b
edge 3 is incident on(upon) b and d
a and c are adjacent



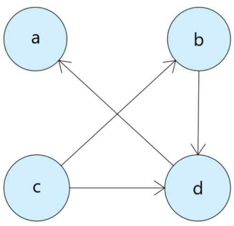
b is adjacent to a
c is adjacent to b

-Representation of graph-
1. Adjacency matrix
2. Adjacency lists

(1) Adjacency matrix



-> $\begin{bmatrix} 0\,0\,0\,1 \\ 0\,0\,1\,1 \\ 0\,1\,0\,1 \\ 1\,1\,1\,0 \end{bmatrix}$ (a boolean matrix), (no self loop)



-> $\begin{bmatrix} 0\,0\,0\,0 \\ 0\,0\,0\,1 \\ 0\,1\,0\,1 \\ 0\,0\,0\,0 \end{bmatrix}$ (a boolean matrix), (no self loop)