Mathematics of Computer Science. - M.I.T. opencourseware

Abstract:
This lecture contains the basic graph theory's component. walk & path, cycle, spanning tree is the base of graph theory. these component lead us up to MST that include weight system.

For the English:
have는 with으로 대체 가능, distinct, acycle, isomorphic, thirst for, choose A over B, lead up to #be_used_to_synonyms

Lec 8. GRAPH THEORY II: MINIMUM SPANNING TREES:
walk & path / connectivity / cycles & closed walk / spanning tree / MST

- Walk & Path
DEF: a walk is a sequence of vertices connected by edges // length = k
DEF: a path is a walk where all vertices are different (distinct)

LEMMA1: idf ∃walk from u to v, then ∃path from u to v
PF1: by well ordering principle: walk of minimal length
- u = v0 – v1 ... vk = v
PF2: by contradiction
- case k = 0 v
- case k = 1 u –v single edge
- case k >= 2 suppose walk is not a path, ∃i != j vi= vj
- u = v0 ... vi = vj ... vk => is not shorter walk

- Connectivity
DEF: u and v are connected if there is a path from u to v
DEF: a graph is connected when every pair of vertices are connected

- Cycles & closed walk
DEF: a closed walk which starts and ends at the same vertex
if k =< 3 and v0 ~ vk-1 are all different, then it is called cycle

- Trees
DEF: a connected and acycle(no cycle) graph is called a tree
DEF: a leaf is a node with degree 1 in a tree

LEMMA: any connected subgraph of a tree is a tree
PF: by contradiction. // suppose the connected sufgraph is not a tree: have cycle
which graph(tree) has this cycle

LEMMA: a tree with n vertices has n-1 edges
PF: by induction // on n =>P(n) = there are n-1 edges in any n vrtices tree

base case = P(1)
inductive step = suppose P(n) let T be a tree with n+1 vertices
let V be a leaf of the tree, delete V: this created a connected subgraph -> graph
By P(n): it has n-1 edges
re-attach V: T has (n-1) +1 = n edges

DEF: a spanning tree(ST) of a connected graph is a subgraph that is tree with the same vertices as the graph
THM: every connected graph has a ST
PF: by contradiction // assume a connected graph G has no ST
let T be a connected subgraph of G with the same Vertices as G and with the smallest # of edges possible
T is not a ST => it is a cycle

case1 = x - ... - y => still connected // does not contain e(check the orignal draw)
case2 = all vertices in G are still connected after removing e from T
there are more smaller tree exist => contradiction

- Weight & MST
Algorithm to create
DEF: the MST of an edge-weighted graph G is the ST of G with the smallest possible sum of edge weights
ALG: grow a subgraph one edge at a time at each step. Add the minimum weight edge that keeps the subgraph acycle
THM: for any connected weighted graph G, the Alg produces a MST
LEMMA: let S consist of the first m edges selected by the Alg. then ∃MST
T = (v, e) for G such that S ⊆ E
PF(THM): #V = n
1) if < n-1 edges are picked, then ∃edges in E- S that added without creating acycle
2) once m = n-1, we know S is a MST
PF(LEM): by induction on m => P(n) ∀G ∀S consist of the first m selected edges,
T = (v, e) ∃MST of G such that S ⊆ E
base case: m = 0 // empty set(∅) S is subset of all set / for any MST T = (v, e)
inductive step: assume P(m)
let e denote the (m+1)st selected edge, let S denote the selected edges
by P(m): let T' = (v, e') be a MST such that S⊆E
case e ∈ E' : S U {e} ⊆ E' -> P(m+1) v
case e !∈ E' : swap e and e' in T:
let T'@ = (v, e''), E'' = (E' - {e}) U {e}
T'' is acycle because removed e' from the only cycle in e' U {e} // T'' is connected since e' was n acycle T'' contains all vertices in G
=> T'' is a ST of G // weight T'' =< weight T' = MST => T'' is MST
(check the orignal memo-post-it)