Lecture 14, Nov 13 2014

White-Path Lemma

- In (directed or undirected) graph G, node v is descendant of u iff at d(u) (time when u was discovered) there is a path from u to v using only currently white nodes (the "white path").
- Note that we are not claiming that DFS will follow this white path, only that its existence in the input graph guarantees that, by the end of the algorithm execution, v will become a descendant of u.

162

Proof of White-Path Lemma

- Assume v is descendant of u and thus there exists a u->v path using tree edges. Let ww' be an edge on this u->v path in the tree.

  If w' was not white at d(u), then ww' will not be tree edge. Thus, all nodes on the u->v path are white when u is discovered.

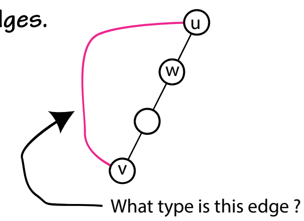
- Assume that at d(u) there is a white path from u to v. For contradiction, assume v is not a descendant of u.

  Let ww' be the first edge on this path where w is descendant of u but w' is not.
  - » We have f(u)> f(w) > d(w) > d(u).
  - » But we have to discover w' after starting u and before finishing w: d(u) < d(w') < f(w) < f(u)
  - » By parenthesis theorem, w' is also a descendant of u, contradiction.

163

Simple Lemma

- Lemma: if G undirected, then only tree and back edges.
  Proof: Consider uv edge, WLOG d(u)<d(v).
  Thus v must be discovered and finished
  before finishing u, since uv exists.
  If uv discovered from u, before v,
  it is tree edge
  if v was discovered before uv,
  uv becomes a back edge.

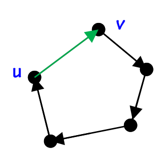


- HW: Why does the proof break down in the directed case ?

164

Discovering Cycles

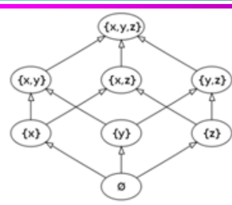
- Claim: G acyclic iff DFS yields no back edges.
- Proof:
  » Trivial to observe that back edge implies a cycle.
  » Assume there exists a cycle:
    – Let v be the node with smallest d on the cycle and let uv be edge of the cycle.
    – At d(v) all nodes on the cycle, including u, are white.
    – All these nodes, including u, become descendants of v.
    – Why ?
    – White-path lemma!
    – Thus, when u is scanned, we will discover uv edge and mark it as “back edge”.



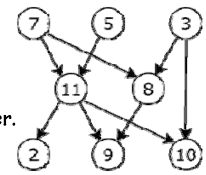
165

Partial Order

- Example: subsets ordered by inclusion
  - » Note implicit relationships, e.g. $x \in \{x,y,z\}$ but there is no edge between them.



- Acyclic directed graph, vertices ordered by reachability

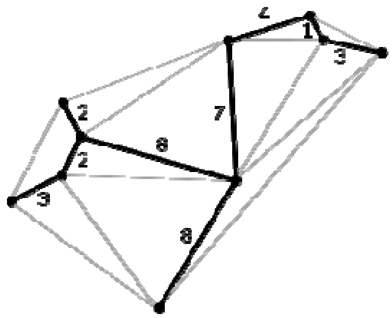


- Topological sort: total order compatible with partial order.
  - » 7,5,3,11,8,2,10,9
  - » 3,7,8,5,11,10,9,2
- Numerous applications, mostly in scheduling: instructions, compilations in makefile, etc.

Topological Sort

- Directed acyclic graph G.
- Algorithm:
  - » Call DFS to compute finishing times f[v] for each vertex v.
  - » As each v is finished, insert it onto the front of linked list
  - » Return the linked list.
- Claim: the output list is a legal topological sort.
  - » Sufficient to prove that, for every u and v s.t. (uv) is an edge, we have f[v] < f[u]. (Why ??)
  - » Consider edge (uv) explored by DFS.
    Observe that when (uv) is explored, v cannot be gray !
  - » (reason: back edge implies cycle)
  - » If v white, it becomes descendant of u, and thus f[v] < f[u].
  - » If v black, it finished before u started, so again f[v] < f[u].

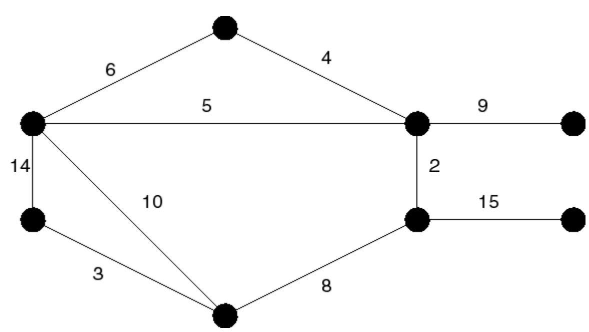
Min-Cost Spanning Tree



- Applications:
  - » Cable TV,
  - » Circuit layout,
  - » Basic task for many optimization algs (eg. flow).
- Formally:
  - » Undirected graph $G=(V,E)$.
  - » Weights $w: E \rightarrow R$
  - » Goal: find spanning tree of minimum weight.
    (spanning = connects all nodes in G)
    (tree weight = sum of weights of tree edges)
- For simplicity, we will assume single connected component. Straightforward extension to multiple components.

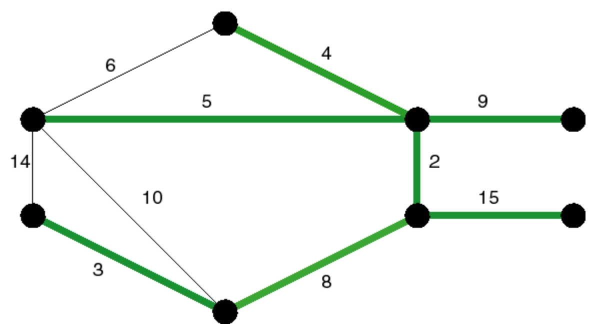
168

Example graph



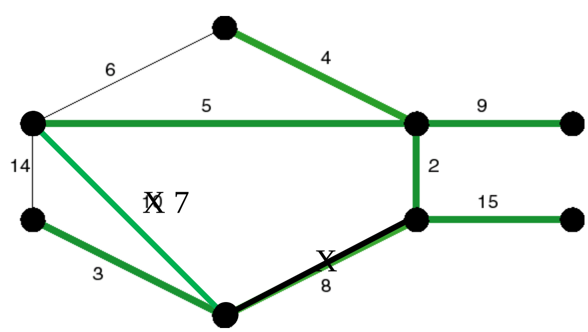
169

Example MST



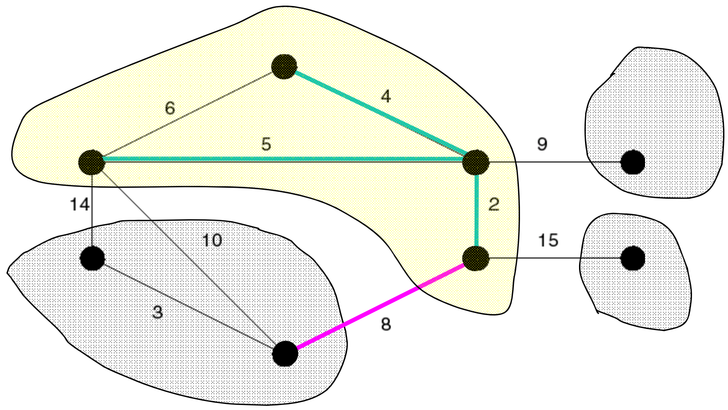
170

What if we modify edge weight ?



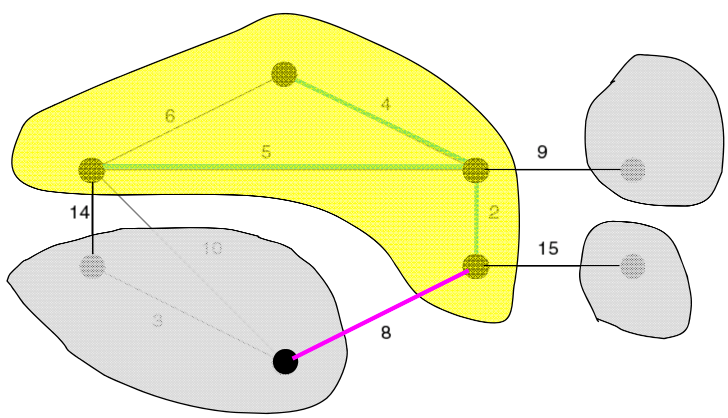
171

Greedy step



172

Greedy step



173

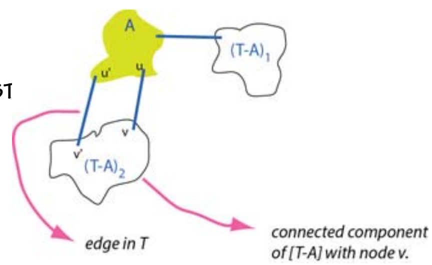
Optimum Substructure

- Assume T is MST of G, $A \subset T$ subtree of T
  let uv be min-weight edge connecting A to (T-A)

  Graph induced by edges in T but not in A

  $$\Rightarrow \exists \text{ MST } T' \text{s.t. } (A \cup uv) \subseteq T' \text{ (loose notation)}$$

- Proof: "Cut-and-Paste" approach"
  - » u'v' connects A to $(T\text{-}A)_2$ in T
  - » Replace u'v' by uv,
  - » Claim resulting T' is optimum MST



- Questions:
  - » Why no more edges parallel to u'v' in T ??
    - – T cannot have cycles
  - » Why u'v' exists at all ??
    - – walk in T until you hit $[T\text{-}A]_2$
    - – No edges in T between $[T\text{-}A]_1$ and $[T\text{-}A]_2$

174