PS#2 Graphs

Çiçek Çavdar

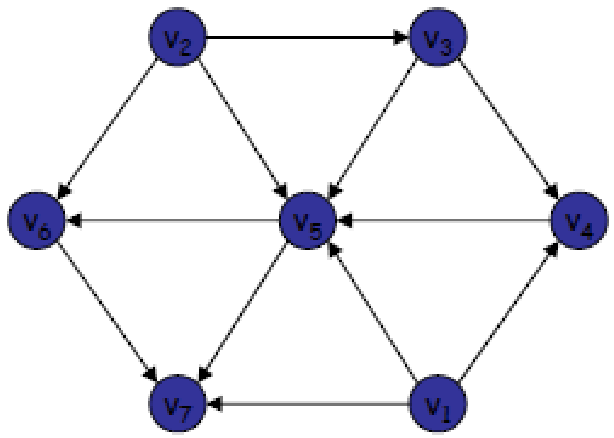
March 12th 2010

Graphs- Exercise 3.3

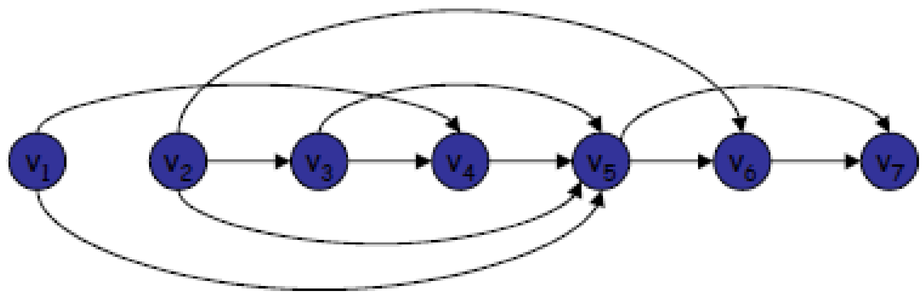
- The algorithm described in Section 3.6 for computing a topological ordering of a DAG repeatedly finds a node with no incoming edges and deletes it
- This will eventually produce a topological ordering, provided that the input graph really is a DAG.
- But suppose that we are given an arbitrary graph that **may or may not be a DAG.**
- Extend the topological ordering algorithm so that, given an input directed graph *G*, it outputs one of the two things:
  - (a) a topological ordering, thus establishing that *G* is a DAG;
  - (b) a cycle in *G*, thus establishing that *G* is not a DAG
  - The running time of your algorithm should be $O(m+n)$ for a directed graph with n nodes and m edges

Directed Acyclic Graphs

- A DAG is a directed graph that contains no directed cycles.
- A topological order of a directed graph $G = (V, E)$ is an ordering of its nodes as $v_1, v_2, \ldots, v_n$ so that for every edge (v_i, v_j) we have $i < j$.



a DAG



a topological ordering

Computing Topological Ordering

- In every DAG G, there is a node with no incoming edges

To compute a topological ordering of G:

Find a node v with no incoming edges and order it first

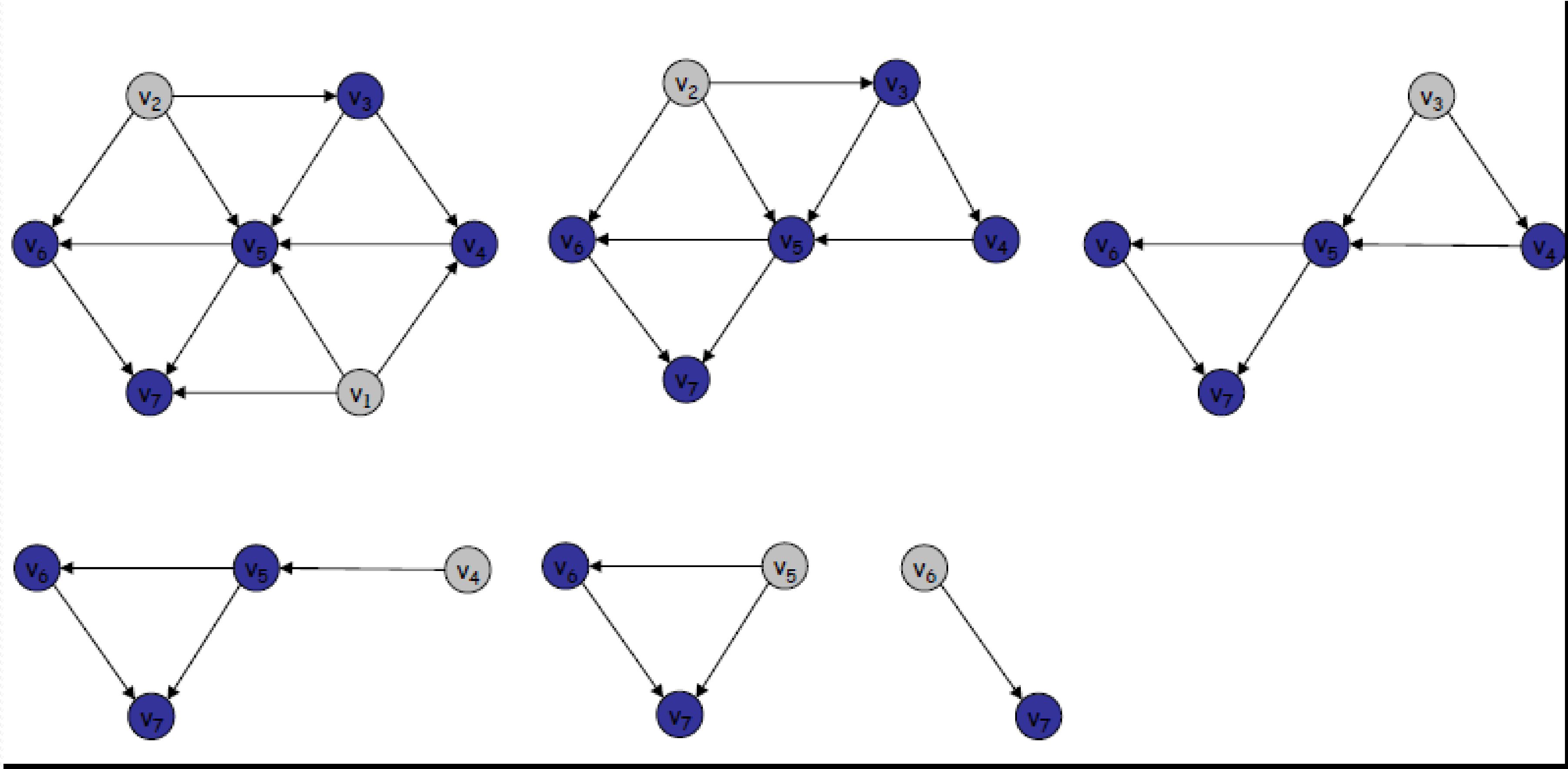
Delete v from G

Recursively compute a topological ordering of $G-\{v\}$

  and append this order after v

Computing Topological Ord.

- Look for nodes with no incoming edges



Question

- The algorithm described here will produce a topological ordering if the input graph is a DAG. But suppose that we are given an arbitrary graph that **may or may not be a DAG.**

- Extend the topological ordering algorithm so that, given an input directed graph *G*, it outputs one of the two things:
  - (a) a topological ordering, thus establishing that *G* is a DAG;
  - (b) a cycle in *G*, thus establishing that *G* is not a DAG
  - The running time of your algorithm should be $O(m+n)$ for a directed graph with n nodes and m edges

Solution

- If in each iteration, we find a node with no incoming edges then we will keep the algorithm

---

To compute a topological ordering of G:

Find a node v with no incoming edges and order it first

Delete v from G

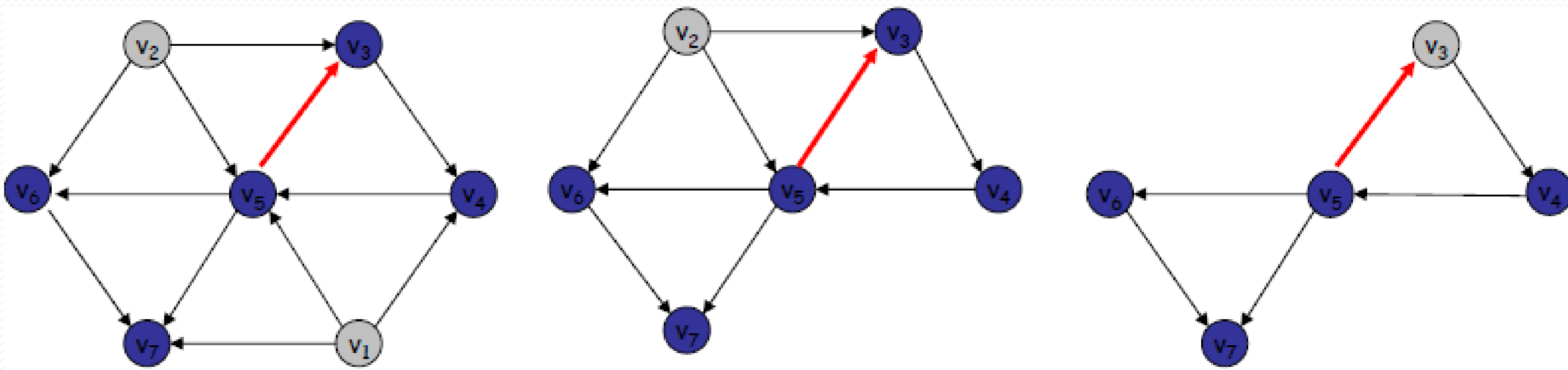
Recursively compute a topological ordering of $G-\{v\}$

and append this order after v

- If in some iteration, every node has at least one incoming edge, then this means *G* must contain a cycle
- We need to modify the above algorithm to include this condition:
- We backtrack the edges, i.e. we follow the first edge back until we reach a node that was visited before
- Taking the first edge on the adjacency list of incoming edges assures O(n), we don't make a search
- Since every node has an incoming edge, we will revisit a node v
- The nodes between two consecutive visits will be the nodes in the cycle C

Solution cont'd

- In the third iteration, all the nodes have at least one incoming edge
- Choose a node
- Choose the first edge on the adjacency list of incoming edges so that this can run in constant time per node
- Repeat until you find a cycle



Homework

- THIS IS **NOT** A HAND IN ASSIGNMENT

- Write a pseudo code for the algorithm described in the previous slides

Graphs- Execise 3.7

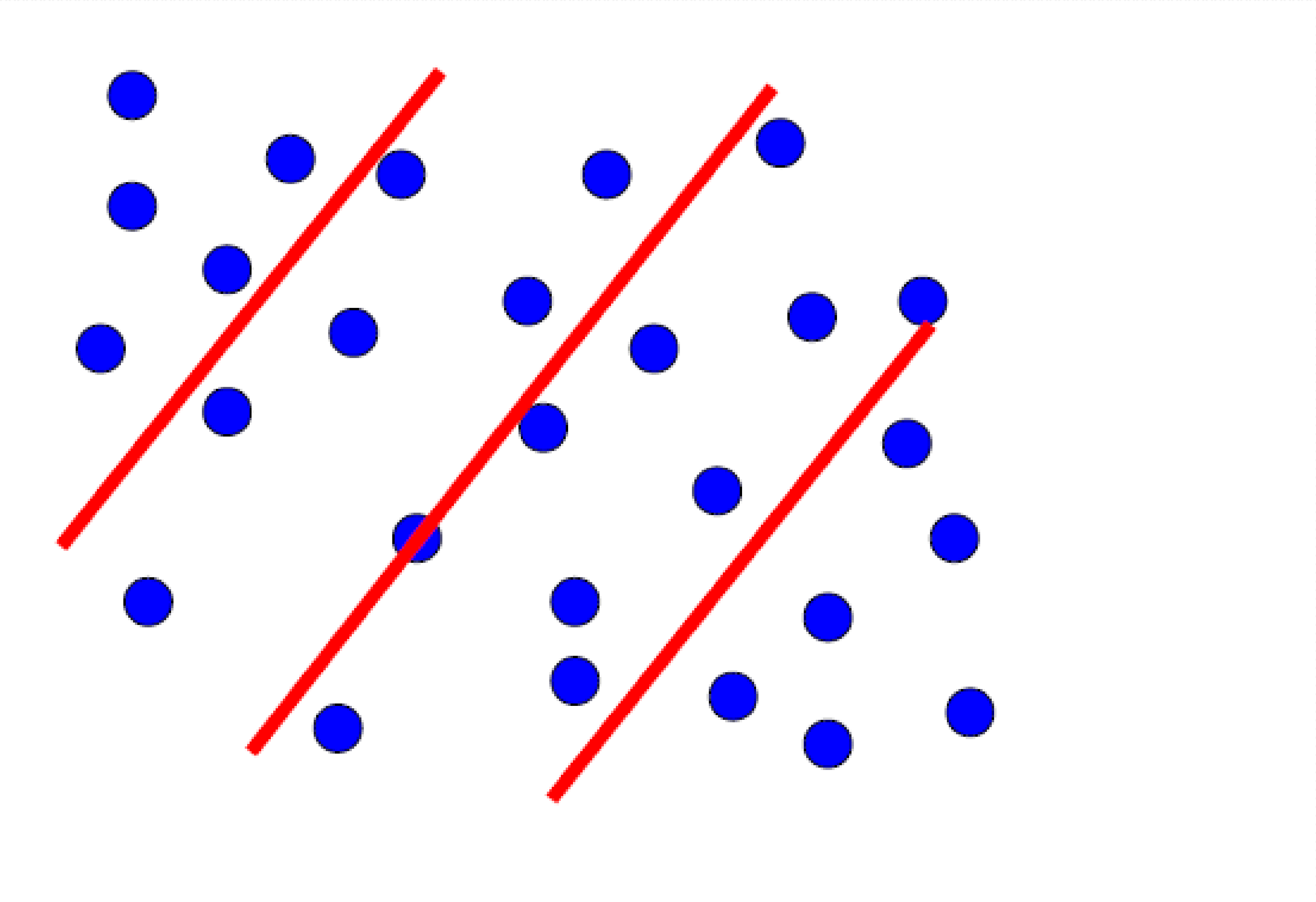
- Some friends of yours work on wireless networks and they are studying the properties of a network of *n* mobile devices.
- They define a graph at any point in time as: there is a node representing each of the *n* devices and there is an edge between device *i* and device *j* if the physical locations of *i* and *j* are no more than 500m apart.
- The guys like to have the devices connected at all times, so they've constrained the motion of the devices to satisfy the following property:
  - at all times, each device is within 500m of at least n/2 of the other devices. (assume n is even)
- Does this property by itself guarantee that the network will remain connected?

Translate the question

- Let G be a graph on n nodes where n is even. If every node of G has degree at least n/2 then G is connected.
- Is this statement true or false and give proof of your answer

Solution

- Proof by contradiction
- Let G be a graph on n nodes where n is even.
- Suppose every node of G has degree at least $n/2$ and G is disconnected.
- Let S be the smallest connected component.
- Since there are at least two connected components, we have $|S| <= n/2$



Solution cont’d

- Consider a node *u* that belongs to set S
- Its neighbors must lie in S, so its degree can be at most |S|-1, which is less than or equal to n/2 -1. And this is smaller than n/2.
- In this case, the graph is disconnected
- This means, if every node of G has degree at least n/2 then G is connected