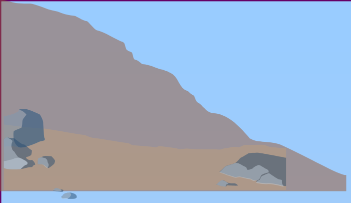
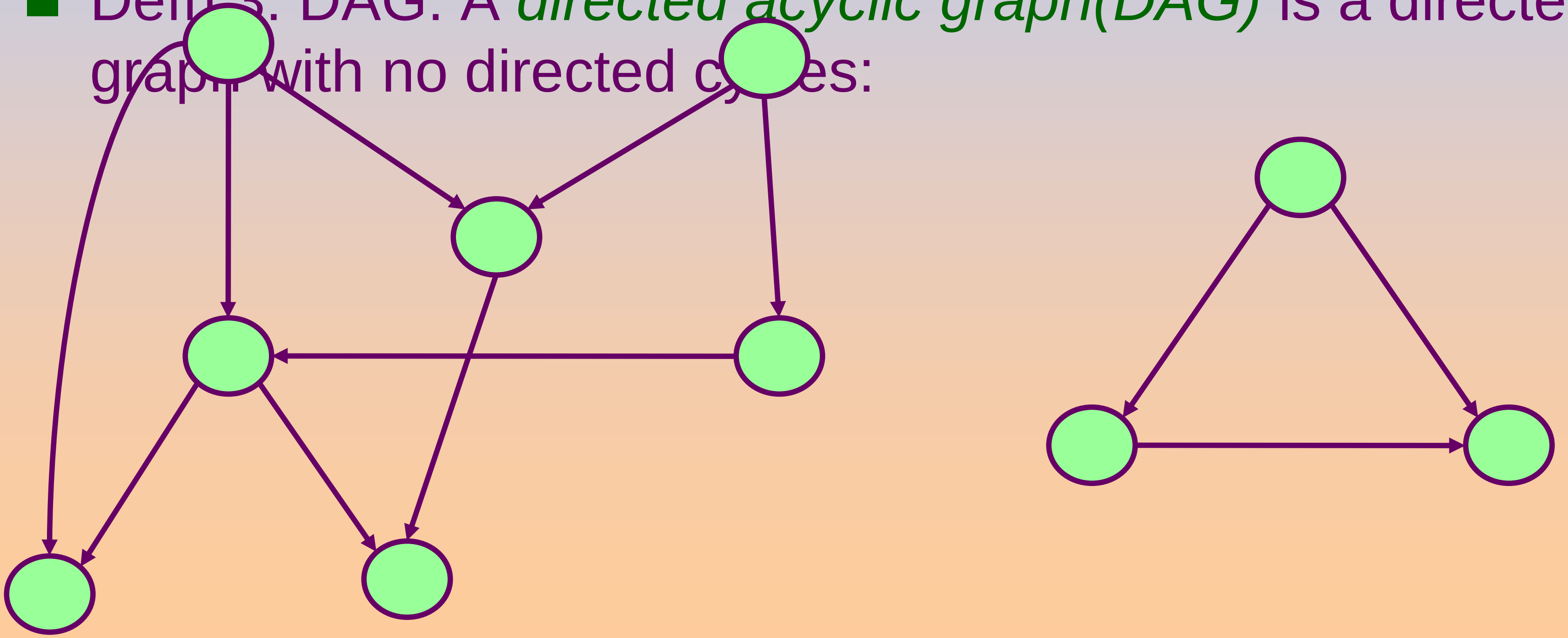
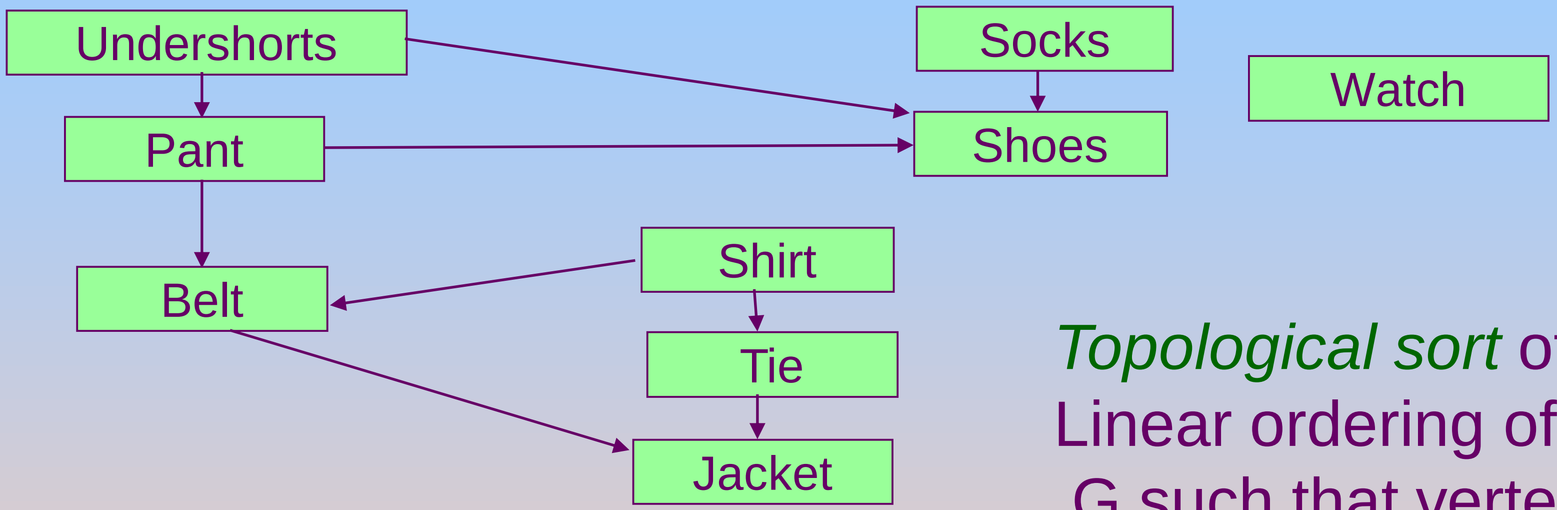


# Definitions

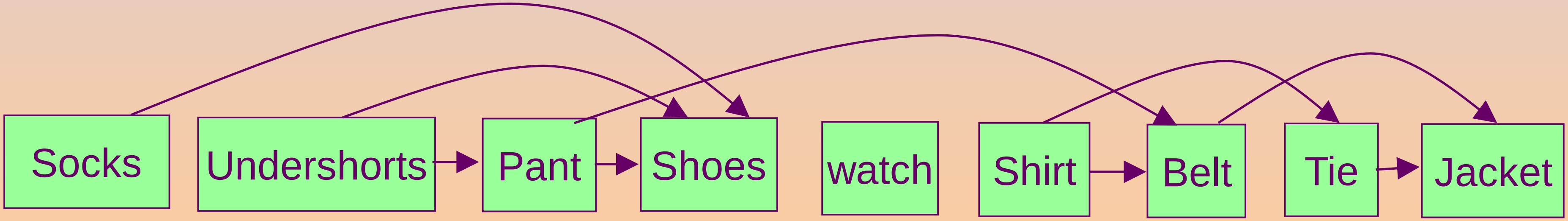
- Defn 1: Indegree : The number of incoming edges into a node
- Defn 2: Outdegree: The number of outgoing edges from a node
- Defn 3: DAG: A *directed acyclic graph(DAG)* is a directed graph with no directed cycles:



# Topological Sort



*Topological sort* of a DAG:
Linear ordering of all vertice
G such that vertex *u* come
vertex *v* if edge (*u*, *v*) ∈ G



- Topological sort was defined as “Linear ordering of all vertices in graph G such that vertex *u* comes before vertex *v* if edge (*u*, *v*) ∈ G”

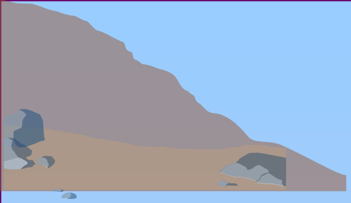
Propose an algorithm that can produce a topological sorted sequence of the nodes in a DAG. (Hint:Use  the statement in question 3 which said “If there is a path from a node u to another node v in a DAG, then f(u) > f(v)”.)

Idea: Print all nodes in decreasing order of finish times

Do a DFS and write out the start and finish times.

While putting down a finish time, push the node into a stack as well.

After the DFS is complete, pop from stack and print



# Kahn’s algorithms

- There could be many other alternate ways to find the topological sorted form of a DAG. Another popular alternative uses a BFS like appraoch called the Kahns algorithm. The idea is to first output the nodes with indegree 0 in any order. Remove them and their outgoing edges. We now have a new set of nodes with indegree 0 which are the children of the earlier

# Kahns Algorithm

- You maintain an AdjList along with the current indegree of each node.
- Keep a queue which is the set of nodes with indegree 0.
- Each time you pick a node whose indegree is 0, dequeue it, reduce indegree of all nodes adjancent to it. While reducing the indegree if any node indegree turns 0 add it to the queue.
- O(V+E)