DS 320 Algorithms for Data Science **Lecture #5 Worksheet**
Spring 2025 Prof. Kira Goldner and Jeffrey Considine

Application of DFS: Topological Sort

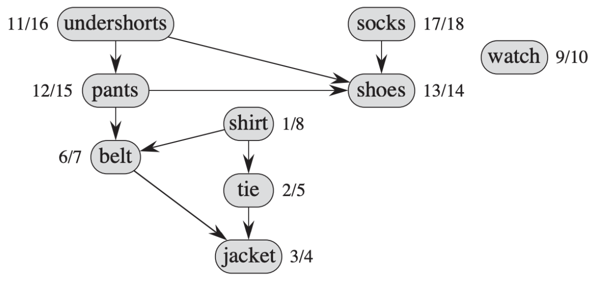

Figure 1: Top sort example graph from CLRS.

Theorem 1. *G has a topological order* $\iff$ *G is a DAG.*

Topological Sort Algorithm:

Theorem 2. *If the tasks are scheduled by decreasing postorder number, then all precedence constraints are satisfied.*

Breadth-First Search

Algorithm 1 BFS(G, s)

```
**Input:** Graph $G = (V, E)$ and starting vertex $s$.
initialize: (1) array $dist$ of length $n$, (2) queue $q$, (3) linked list $L$ of sets, (4) tree $T = (\{s\}, \emptyset)$
dist$[s] = 0$
$L[0] = \{s\}$
enqueue $s$ to $q$
mark $s$ as discovered and all other $v$ as undiscovered
**while** size$(q) > 0$ **do**
    $v$ = dequeue$(q)$
    **for** $(v, w) \in E$ **do**
        **if** $w$ is undiscovered **then**
            enqueue $w$ in $q$
            mark $w$ as discovered
            dist$(w)$ = dist$(v)$ + 1
            add $w$ to $L[dist(w)]$
            add $(v, w)$ to $T$
        **end if**
    **end for**
**end while**
**return** $T, L$
```

What happens when we run BFS(G, 1) where G is the graph below?

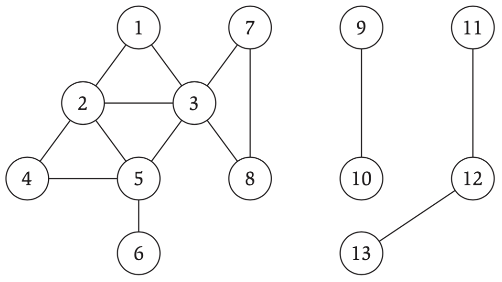

Figure 2: Example graph G. From Kleinberg Tardos.

What is BFS doing? BFS labels each vertex with the distance from s, or the number of edges in the shortest path from s to the vertex. (**Exercise:** Prove this!)

Runtime:

Claim 1. Let T be a breadth-first search tree, let x and y be nodes in T belonging to layers L_i and L_j respectively, and let (x, y) be an edge of G. Then i and j differ by at most 1.

Proof.

Definition 1. We say a graph $G = (V, E)$ is *bipartite* if the vertices can be partitioned into disjoint sets $V = A \sqcup B$ such that for every edge in E, one endpoint is in A and the other endpoint is in B. That is, there are no edges within A or within B.

We'll use another idea from graph theory: vertex coloring. We try to color the vertices with the *smallest* number of colors possible, but a vertex cannot be colored the same color as any of its neighbors. Then a bipartite graph is a graph that can be colored with two colors—all vertices in A can be colored red and all vertices in B can be colored blue.

First convince yourself of the following:

Claim 2. If a graph is bipartite, it cannot contain an odd cycle.

Then, use BFS to come up with a graph-coloring algorithm to determine whether the graph is bipartite.

We now prove correctness of this algorithm, and start the proof as follows. Let G be a connected graph, and let $L_1, L_2, \ldots$ be the *layers*, or sets of vertices of distance 1, 2, and so on, produced by BFS starting at node s. Then exactly one of the following two things must hold.

a. Suppose G has no edges with both endpoints in the same layer. Argue that G must be a bipartite graph.

Hint: It may be useful to first prove claim below, and then add the idea of vertex coloring.

b. Suppose G has an edge with both endpoints in the same layer. Argue that in this case, G must contain an odd-length cycle and therefore not be bipartite.

Hint: It may again be useful to consider the BFS tree produced by the algorithm.