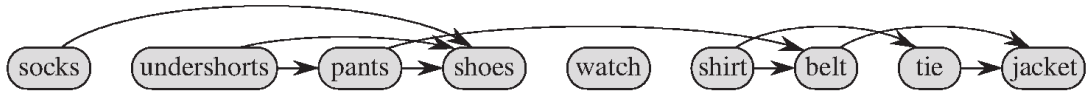
Exercises, chapter 6, solutions

---

1. Many different solutions are possible. Here is one, from:

   - T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein: *Introduction to Algorithms (Third Edition)*, The MIT Press, 2009. [Figure 22.7]



2. (a) The difference between `DFS` in Chapter 2 and `DFSTopologicalSort` in Chapter 6 is that the latter uses some additional information in the form of a list named *sorted* for storing vertices of V. More precisely, whenever `DFSTopologicalSort` reaches a dead end at some vertex u during the depth-first search (i.e., when all neighbors of u have been taken care of in the **foreach**-loop on lines 2–4 and it's time to backtrack), it will insert u at the current head of *sorted* (line 5 of `DFSTopologicalSort`) before returning from the recursive call.

   (b) It is useful because it lets a critical path in G be computed efficiently with the "relaxation" technique. For relaxation to work, after we have used the estimated length ℓ of a longest path to some vertex u to update the estimated lengths of longest paths to the vertices in u's adjacency list, the value of ℓ is not allowed to change. Considering the vertices in the order obtained by a topological sort of G guarantees that all vertices that have paths leading to u will have been handled before u, so the final value of ℓ is available once it is time to use ℓ.

3. **Algorithm for checking if an undirected graph $G = (V, E)$ contains a cycle:**

   (i) If $|E| \geq |V|$ then return TRUE.

   (ii) Otherwise ($|E| \leq |V| - 1$), run `HasCycle`(G) below, which does a depth-first search of G using a procedure `Modified-DFS`. For any unvisited vertex i, `Modified-DFS` returns TRUE if and only if the connected component in G that i belongs to contains a cycle. The parameter *parent* specifies the vertex from which the search came; it is excluded when handling the neighbors of u so that $\langle parent, u, parent \rangle$ will not be considered to be a cycle.

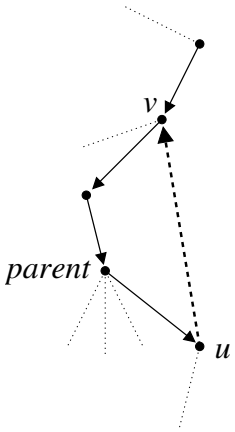
```
HasCycle(G)
1  Create a Boolean array visited of length |V| and initialize all of its entries to FALSE.
2  has_cycle ← FALSE
3  for i ← 1 to |V| do
4      if not visited[i] then
5          has_cycle ← has_cycle or Modified-DFS(i, null)
6  return has_cycle

Modified-DFS(u, parent)
1  visited[u] ← TRUE
2  tmp ← FALSE
3  foreach v in AdjacencyList(G, u) do
4      if v ≠ parent then
5          if visited[v] then        /* A cycle has been found. */
6              tmp ← TRUE
7          else
8              tmp ← tmp or Modified-DFS(v, u)
9  return tmp
```

Explanation:

First observe that the number of edges in any undirected graph $G = (V, E)$ without cycles is at most $|V| - 1$. Consequently, if the input graph satisfies $|E| \geq |V|$ then it must have a cycle, in which case the algorithm can return TRUE immediately in step (i).

On the other hand, if $|E| \leq |V| - 1$ then G may or may not have a cycle. To find the answer, the algorithm will perform a depth-first search of G in step (ii). There are two possibilities:

- If G contains a cycle, there is some edge that connects a vertex u to another vertex v such that: (i) v was visited by the depth-first search before visiting u; and (ii) v is not the parent of u in the depth-first search. This will be discovered by the algorithm on line 5 of `Modified-DFS` during the visit to u because then v is in `AdjacencyList`(G, u) and: (i) *visited*$[v]$ will already have been set to TRUE; and (ii) v is different from *parent*. See the following figure for an illustration:



  When this situation occurs, line 6 of `Modified-DFS` sets *tmp* equal to TRUE. Next, since the logical **or**-operation between TRUE and anything else gives TRUE, the value TRUE will be returned to the calling function and all the way back up to `HasCycle`, where it is assigned to the variable *has_cycle*. The value of *has_cycle* can never be changed back to FALSE, so the value TRUE is returned at the end.

- If G doesn't contain any cycles, *visited*$[v]$ will never be TRUE for any neighbor v of the currently visited vertex during the depth-first search (except for the parent of u in the depth-first search). Therefore, line 6 of `Modified-DFS` will never be executed. Since the logical **or**-operation between FALSE and FALSE is FALSE, the value of *has_cycle* will not change from its initial value FALSE, and the algorithm will correctly return FALSE at the end.

Time complexity analysis:

Checking if $|E| \geq |V|$ holds in step (i) can be done in $O(|V|)$ time by counting the edges one by one until all edges have been counted or we reach $|V|$. Step (ii) takes $O(|V| + |E|)$ time according to the analysis of the depth-first search algorithm, but here we also have $O(|V| + |E|) = O(|V|)$ because this step is only executed when $|E| \leq |V| - 1$. In total, the algorithm's time complexity is $O(|V|)$.