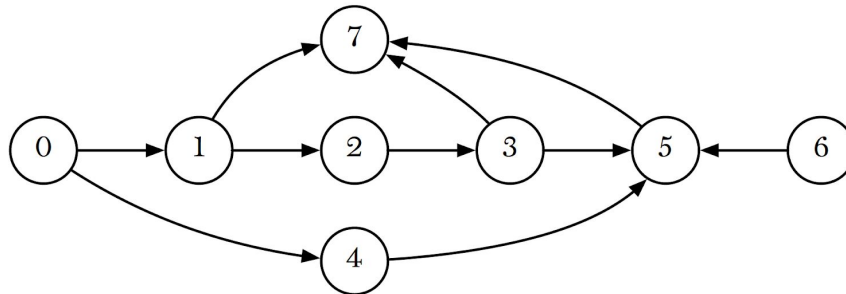


EXERCISE 1: Shortest Common Ancestor

In a directed graph, a vertex x is an ancestor of v if there exists a (directed) path from v to x . Given two vertices v and w in a rooted directed acyclic graph (DAG), a shortest common ancestor $sca(v, w)$ is a vertex x which:

- is an ancestor to both v and w ;
- minimizes the sum of the distances from v to x and w to x (this path, which goes from v to x to w , is the shortest ancestral path between v and w).

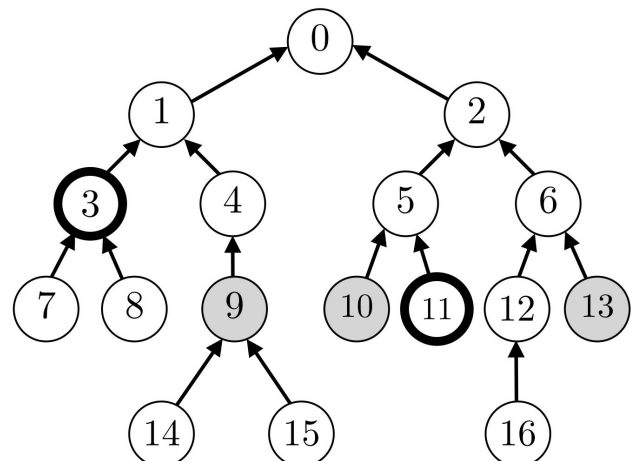
A. In the following digraph, find the shortest common ancestor of vertices 1 and 4, and give the sum of the path lengths from these vertices to all common ancestors, and then circle the shortest.



B. Describe an algorithm for calculating the shortest common ancestor of two vertices v and w . Your algorithm should run in linear time (proportional to $V + E$).

C. How would your algorithm differ if we are interested in the shortest ancestral path between two **sets** of vertices A and B instead of two vertices? I.e. between any vertex v in A and any vertex w in B .

In the example, $A = \{3, 11\}$ and $B = \{9, 10, 13\}$. The shortest common ancestor is 5 (between 10 and 11).



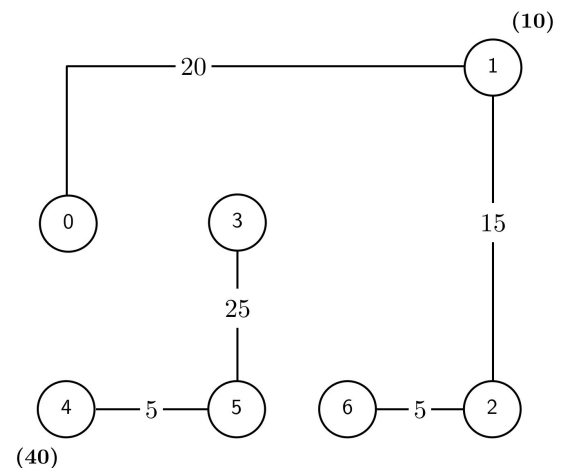
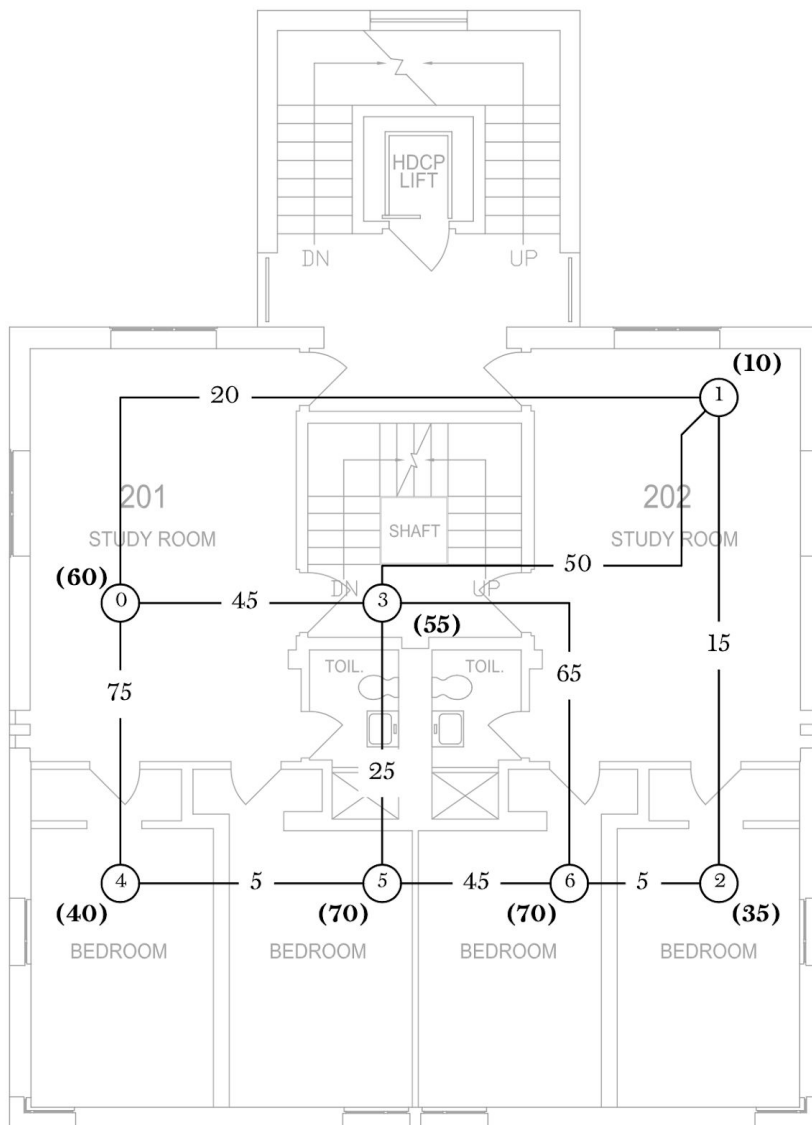
EXERCISE 2: Dorm Room and Routers (Design Question)

There are N rooms, each of which needs an internet connection. A room i has internet access if either of the following is true:

- There is a router installed in room i (this costs $r_i > 0$).
- The room i is connected by some fiber path to another room j which itself has internet access (putting down fiber between room i and j costs $f_{ij} > 0$).

The goal of this problem is to determine in which rooms to install a router, and in which pair of rooms to connect together with fiber, so as to minimize the total cost.

Formulate the problem as a *minimum spanning tree* problem, given a graph $G = (V, E)$ with vertices $V = \{v_1, \dots, v_n\}$ and the previously mentioned costs, r_i and f_{ij} . You may use the example below to test your formulation.



Solution

For example, this instance contains 7 dorm rooms and 10 possible connections. The router installation costs are indicated in bold and parentheses; the fiber costs are given on the edges. The optimal solution, which costs 120, installs a router in rooms 1 and 4 (for a cost of $10 + 40$) and builds the fiber connections shown above.

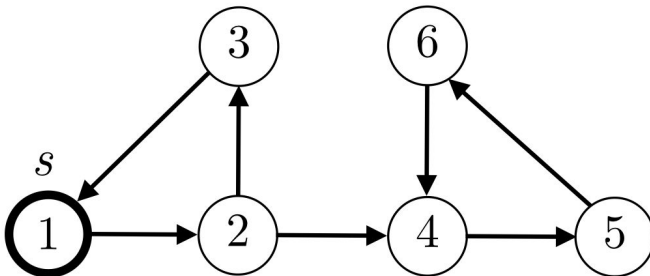
EXERCISE 3: Detecting Directed Cycles

An online version of this exercise is available at: <https://stepik.org/lesson/219467>

The online version provides instant feedback and has an extra optional part.

A. Consider the graph G given below and the marked vertex s . Show in the given box what the output would be if `depthFirstSearch` is called on G and s .

```
1 private boolean[] marked;  
2  
3 public void depthFirstSearch(Digraph G, int s) {  
4     marked = new boolean[G.V()];  
5     dfs(G, s);  
6 }  
7  
8 private void dfs(Digraph G, int v) {  
9     marked[v] = true;  
10    StdOut.println("Starting " v);  
11    for (int w : G.adj(v)) {  
12        if (!marked[w])  
13            dfs(G, w);  
14    }  
15    StdOut.println("Finished " + v);  
16 }
```



B. Consider the following modified version of the `dfs` method. Explain with the simplest counterexample why this code is not a correct cycle detection code.

```
1 private void dfs(Digraph G, int v) {  
2     marked[v] = true;  
3  
4     for (int w : G.adj(v)) {  
5         if (!marked[w])  
6             dfs(G, w);  
7         else StdOut.print("Cycle found!");  
8     }  
9 }
```

C. Briefly describe how depth-first search could be modified to detect cycles in a digraph.

D. Fill the blank lines in the following DFS code so that it prints "Cycle found!" if and only if there is cycle in the graph. Assume that the graph is connected.

```
1 private boolean[] marked;
2 private boolean[] onStack;
3
4 public void checkCycles(Digraph G, int s) {
5     marked = new boolean[G.V()];
6     _____
7     dfs(G, s);
8 }
9
10 private void dfs(Graph G, int v) {
11     marked[v] = true;
12     _____
13     for (int w : G.adj(v)) {
14         if (!marked[w])
15             dfs(G, w);
16         else if (_____)
17             StdOut.print("Cycle found!");
18     }
19     _____
20 }
```