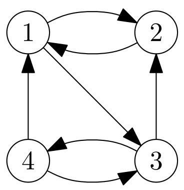
1.
- an adjacency matrix (a matrix whose rows and columns are indexed by the vertices, with a 1 in row i and column j if there is an edge from vertex i to vertex j),
- an adjacency list (in a form described by Goodrich and Tamassia in which there is an object for each vertex and edge, each edge object points to its two vertex endpoints, and each vertex object points to two collections of edge objects for its incoming and outgoing edges), and
- a Python representation described by van Rossum in which a graph is a dictionary whose keys are vertices and whose values are lists of the outgoing neighbors of each vertex (see the link for examples).

(a) Draw the graph whose adjacency matrix is below, using circles for the vertices and arrows for the edges. Label each vertex with the number of its row and column.

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Solution:



(b) How many objects of each type would be needed to represent the same graph in the Goodrich and Tamassia representation?

Solution: 4 vertices, 7 edges.

(c) Write down a Python expression that produces the van Rossum representation for the same graph. In this expression, use the number i to represent the vertex whose adjacencies are given by the ith row and column of the matrix.

Solution: `G = { 1:[2, 3], 2:[1], 3:[2, 3], 4:[1, 3] }`

2. In the low-attention-span web surfer model used to define the pagerank algorithm, suppose that the probability of getting bored and going to a uniformly random page is 0.1, and the probability of following a random link from the current page is 0.9. Given the graph represented by the same matrix above, starting from a state in which each vertex is equally likely, what would be the probabilities of being on each of the four vertices after a single time step?

Solution: Let I be the initial vertex position and F be the final vertex position. We're using the formula:

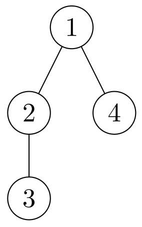
$$\mathbb{P}\left[F = j\right] = \frac{\mathbb{P}\left[\text{bored}\right]}{\#\text{ pages}} + \mathbb{P}\left[\text{not bored}\right] \sum_{i|i\to j} \frac{\mathbb{P}\left[I = i\right]}{\#\text{ links going out of i}}$$

For our graph,

$$\begin{aligned}
\mathbb{P}\left[F = 1\right] &= \frac{0.1}{4} + 0.9\left(\frac{1/4}{1} + \frac{1/4}{2}\right) = 0.3625\\
\mathbb{P}\left[F = 2\right] &= \frac{0.1}{4} + 0.9\left(\frac{1/4}{2} + \frac{1/4}{2}\right) = 0.25\\
\mathbb{P}\left[F = 3\right] &= \frac{0.1}{4} + 0.9\left(\frac{1/4}{2} + \frac{1/4}{2}\right) = 0.25\\
\mathbb{P}\left[F = 4\right] &= \frac{0.1}{4} + 0.9\left(\frac{1/4}{2}\right) = 0.1375
\end{aligned}$$

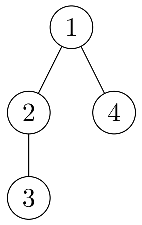
3. (a) Draw a graph, a designated starting vertex in your graph, and a sequence of the vertices that can be reached from the starting vertex, with the property that the sequence you give could have been generated by a depth first search, but not by a breadth first search.

Solution: In the following graph, if 1 is the starting vertex, DFS could generate the sequence 1-2-3-4, but BFS could not.



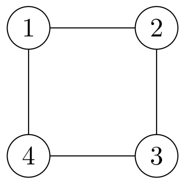
(b) Draw a graph, a designated starting vertex in your graph, and a sequence of the vertices that can be reached from the starting vertex, with the property that the sequence you give could have been generated by a breadth first search, but not by a depth first search.

Solution: In the same graph, if 1 is the starting vertex, BFS could generate the sequence 1-2-4-3, but DFS could not.



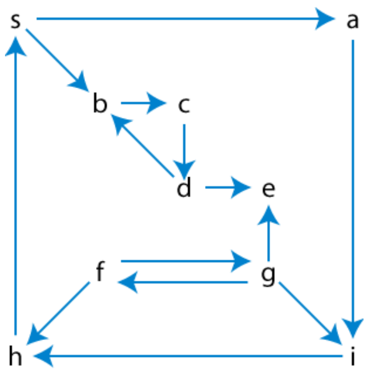
(c) (For 265 students only.) Draw a graph in which it is impossible for depth first search and breadth first search to generate the same sequence as each other, no matter which starting vertex is chosen and no matter which ordering is chosen for the list of edges out of each vertex. Explain why the two search algorithms must always generate different sequences for your graph.

Solution:

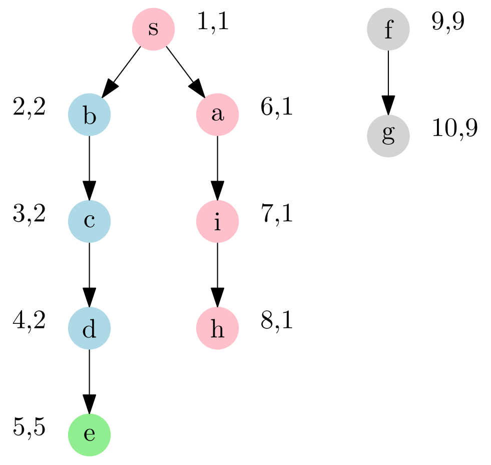


In a 4-cycle, there is always exactly one vertex between vertices 1 and 3 in any sequence generated by a DFS. But there is either zero or two vertices between 1 and 3 in any sequence generated by a BFS. So the two search algorithms always generate different sequences.

4. For the graph below, draw a possible depth-first search forest. Additionally, label each vertex with the lowlink number that would be computed by Tarjan's strongly connected components algorithm, and list the strongly connected components. Don't forget to include any single-vertex components, if they exist.



Solution:



The numbers next to each vertex correspond to the index, followed by the lowlink.

The components are:
s, a, i, h
b, c, d
e
f, g