Breadth First Search

Arnab Ganguly, Assistant Professor
Department of Computer Science, University of Wisconsin – Whitewater
Data Structures (CS 223)

Simulation Weblink

https://www.cs.usfca.edu/~galles/visualization/BFS.html

1 Algorithm

Although powerful, the topological sort based shortest path algorithm applies only to acyclic graphs, simply because other graphs do not have a topological order. So, we need to find other promising avenues. Let us simplify the setting – we are given an undirected graph and each edge weight is 1. Now, we can find the shortest path starting from a source vertex s using yet another classical algorithm, called *Breadth First Search* (BFS).

The idea is simple – starting from the source vertex s, we will process vertices in order of their distance from s, i.e., the closest vertex (breaking ties arbitrarily) is processed first, then the second-closest vertex, and so on.

Let $level[\,]$ be an integer array of length the number of vertices in the graph, which stores the BFS distances (same as the shortest path lengths) to every vertex from s.

BFS starting from vertex s

- Initialize all cells of $level[\,]$ to ∞.
- Enqueue s to a queue.
- Set $level[s] = 0$
- Repeat the following steps until the queue is empty
  - dequeue vertex v from the queue
  - for every outgoing edge e of v, do the following:
    - let w be the destination of e
    - if ($level[w]$ *equals* ∞) then
      - set $level[w]$ to $level[v] + 1$
      - enqueue w

1.1 Simulation

We simulate the BFS algorithm on the graph in Figure 1, starting from the vertex v_0. Table 1 shows the simulation details.

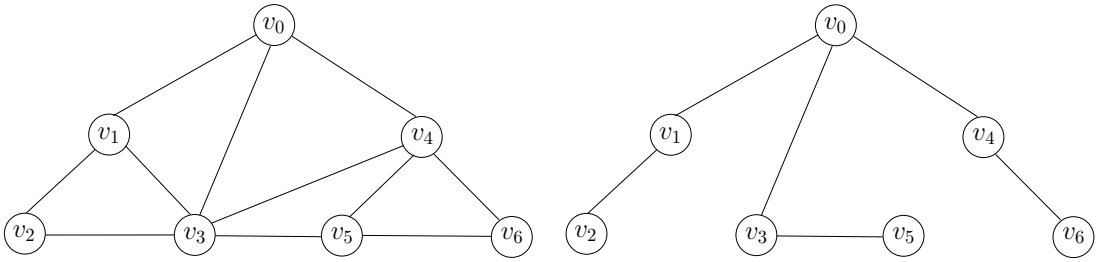

Figure 1: BFS from v_0 on the graph on left side leads to the BFS-tree on the right side

Table 1: Illustration of BFS

	level array							**queue**
	v_0	v_1	v_2	v_3	v_4	v_5	v_6	
At Start:	0	∞	∞	∞	∞	∞	∞	$[v_0]$
Dequeue (Relax v_0)	0	1	∞	1	1	∞	∞	$[v_1, v_3, v_4]$
Dequeue (Relax v_1)	0	1	2	1	1	∞	∞	$[v_3, v_4, v_2]$
Dequeue (Relax v_3)	0	1	2	1	1	2	∞	$[v_4, v_2, v_5]$
Dequeue (Relax v_4)	0	1	2	1	1	2	2	$[v_2, v_5, v_6]$
Dequeue (Relax v_2)	0	1	2	1	1	2	2	$[v_5, v_6]$
Dequeue (Relax v_5)	0	1	2	1	1	2	2	$[v_6]$
Dequeue (Relax v_6)	0	1	2	1	1	2	2	[ ]

1.2 BFS Tree

If we execute the BFS algorithm on a graph from a vertex s, then we obtain a tree on the same set of vertices that has the following nice property – *a path from s to a vertex v in the tree is the same as a shortest path from s to v in the graph.* This tree is called the BFS tree, also known as the shortest path tree. Obtaining the tree is achieved as follows:

BFS tree

- Start with only the set of vertices of $\mathcal{G}$, with no edge connecting any two vertices.
- Execute the BFS algorithm by making the following minor modification.
- Suppose a vertex v is dequeued from the queue, following which an unvisited vertex w is added to the queue. Then, add an edge from v to w in the tree.

Figure 1 right shows the resultant BFS tree due to the execution of BFS on the graph in left.

1.3 Extension to Directed Graphs

Extending BFS to directed graphs (including DAGs) is straightforward. When a vertex v is dequeued, process only the outgoing edges of v. The rest of the algorithm does not change.

1.4 Complexity

Note that an edge or a vertex is processed only a constant number of times. Hence the complexity on a graph with N vertices and M edges is $O(N + M)$.

2 Connectivity and Reachability

An undirected graph is connected if there is a path between any two pair of nodes. BFS can be used to check whether a graph is connected by using the following strategy:

Connectivity

- Start BFS at an arbitrary vertex.
- Upon termination, if there exists a vertex v for which $level[v] = \infty$, then the graph is not connected, otherwise it is connected.

Remark: Connectivity in directed graphs is different, and we are skipping that in this discussion.

In the reachability problem, given two vertices s and t, our task is to determine if there exists a path from s to t. The reachability problem can again be solved using BFS.

Reachability

- Start BFS at s.
- Then t is reachable if and only if $level[t] \neq \infty$ upon termination.

3 Shortest Path in Graphs with Positive Integral Weights

Split an edge (u, v) with weight W into a path having W edges. See Figure 2 for illustration. Now run BFS algorithm on this graph. More specifically,

Using BFS to obtain shortest path in weighted graphs

- Convert the weighted graph $\mathcal{G}$ into an unweighted graph $\mathcal{H}$ as follows: If edge weight is 1, leave it alone. Otherwise, split an edge (u, v) with positive integral weight $W \geq 2$ into a path having W edges.
- Run BFS from s on $\mathcal{H}$.
- Upon termination, the shortest s-t path is given by $level[t]$.

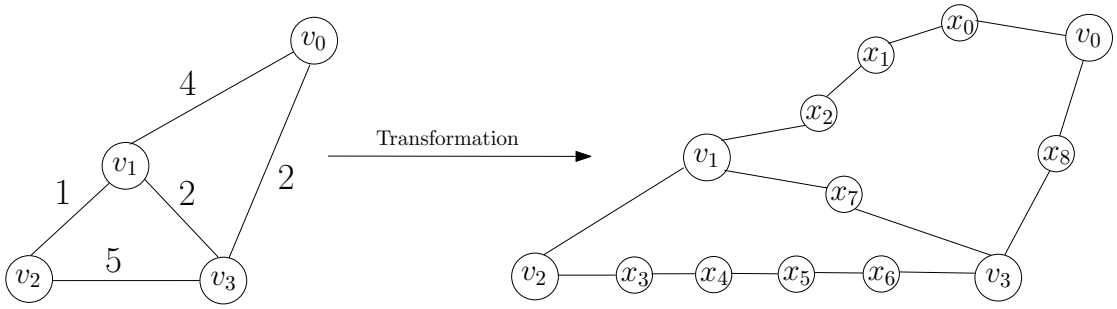

Figure 2: Reducing a weighted undirected graph to unweighted