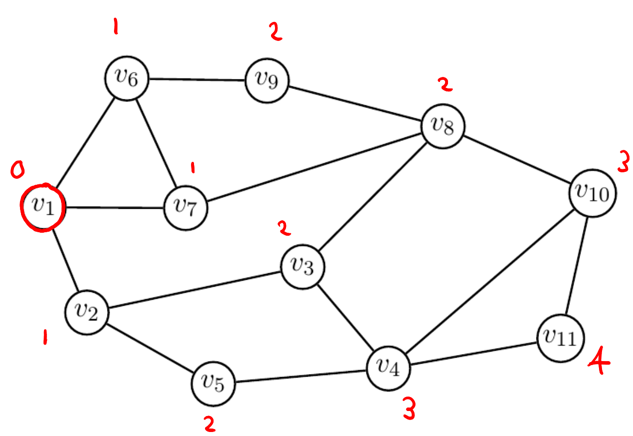
3.4 Breadth-first search (BFS): shortest paths and more

Let $G = (V, E)$ be an undirected graph. We define the distance function $d : V \times V \to \mathbb{Z}_{\geq 0} \cup \{\infty\}$ as follows

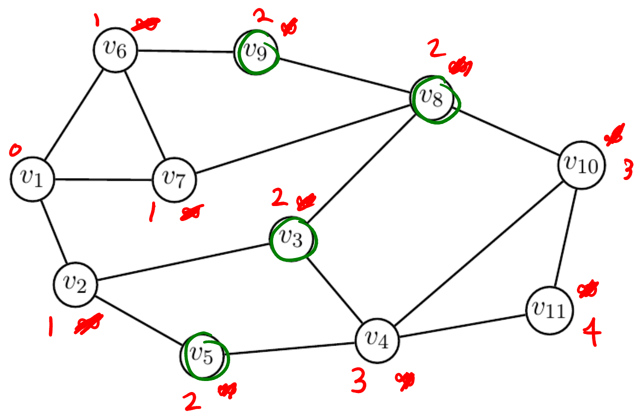
$$d(u,v) = \min\{ |P| : P \subseteq E,\ P \text{ is a } u\text{-}v \text{ walk} \} \quad \forall u,v \in V.$$

BFS determines, for a given vertex $s \in V$, all distances $d(s,v)$ for $v \in V$.

$d(v_1, \cdot)$



Pseudocode for BFS and example run



Algorithm 2: Breadth-first search: computation of distances from a fixed vertex v_1

Input: $G = (V, E)$, $v_1 \in V$.
Output: $d(v_1, v)$ for all $v \in V$.

1. **Initialization:**

   $$d(v_1, v) = \begin{cases} \infty & \text{if } v \in V \setminus \{v_1\}, \\ 0 & \text{if } v = v_1. \end{cases}$$

   $L = \{v_1\}$. `// vertices to be processed`
   $k = 1$. `// shortest possible assignable distance`

2. **while** ($L \neq \emptyset$) **do:**
   $L_{\text{new}} = \emptyset$.
   for $v \in N(L) := \{u \in V : \exists w \in L \text{ with } \{w, u\} \in E\}$ **do:**
   if $d(v_1, v) = \infty$ **then:**
   $d(v_1, v) = k$.
   $L_{\text{new}} = L_{\text{new}} \cup \{v\}$.
   $k = k + 1$.
   $L = L_{\text{new}}$.

3. **return** $d(v_1, v)$ for all $v \in V$.

---

k	L	N(L)
1	$\{v_1\}$	$\{v_2, v_7, v_6\}$
2	$\{v_2, v_7, v_6\}$	$\{v_1, v_6, v_7, v_9, v_8, v_3, v_5\}$
3	$\{v_5, v_3, v_9, v_8\}$	$\{v_6, v_7, v_2, v_4, v_3, v_8, v_9, v_{10}\}$
4	$\{v_4, v_{10}\}$	$\{v_5, v_3, v_8, v_4, v_{10}, v_{11}\}$
5	$\{v_{11}\}$	$\{v_4, v_{10}\}$
6	$\{\ \}$	

BFS for directed graphs

Only modification: Replace $N(L)$ by

$$N^+(L) := \{u \in V : \exists\, v \in L \text{ with } (v,u) \in A\}.$$

Theorem 3.13: Correctness of breadth-first search

For every graph $G = (V, E)$ and every vertex $v_1 \in V$, Algorithm 2 calculates the values $d(v_1, v)$ correctly for all $v \in V$.

<u>Proof</u>

$\rightarrow$ see script.

Theorem 3.14: Running time of breadth-first search

For every graph $G = (V, E)$ and every vertex $v_1 \in V$, Algorithm 2 has a running time bounded by $O(m + n)$.

Algorithm 2: Breadth-first search: computation of distances from a fixed vertex v_1

Input: $G = (V, E)$, $v_1 \in V$.
Output: $d(v_1, v)$ for all $v \in V$.

$O(n)$ { 1. **Initialization:**

$$d(v_1, v) = \begin{cases} \infty & \text{if } v \in V \setminus \{v_1\}, \\ 0 & \text{if } v = v_1. \end{cases}$$

$L = \{v_1\}$. `// vertices to be processed`

$k = 1$. `// shortest possible assignable distance` }

$O(n)$ iterations → 2. **while** ($L \neq \emptyset$) **do:**

$O(1)$ { $L_{\text{new}} = \emptyset$. }

$O(m+n)$ { **for** $v \in N(L) := \{u \in V : \exists w \in L \text{ with } \{w, u\} \in E\}$ **do:**

  if $d(v_1, v) = \infty$ **then:**

    $d(v_1, v) = k$.

    $L_{\text{new}} = L_{\text{new}} \cup \{v\}$. }

$O(1)$ { $k = k + 1$.

$L = L_{\text{new}}$. }

3. **return** $d(v_1, v)$ for all $v \in V$.

$\Rightarrow$ running time $= O(n(n+m))$

It takes $O\left(|L| + \sum_{v \in L} \deg(v)\right)$ time to compute $N(L)$

$L_0 = \{v_1\}, L_1, \ldots, L_q$

Time to compute all $N(L_i)$ $O\left(\sum_{i=1}^{q} \left(|L_i| + \sum_{v \in L_i} \deg(v_i)\right)\right) = O\left(|V| + \sum_{i=1}^{q} \sum_{v \in L_i} \deg(v)\right)$

$= O\left(|V| + \sum_{v \in V} \deg v\right) = O(|V| + |E|)$.