CS 341: Algorithms

Lecture 10: Graphs, breadth first search

Éric Schost

based on lecture notes by many other CS341 instructors

David R. Cheriton School of Computer Science, University of Waterloo

Fall 2024

Definitions

Undirected graphs

Definition, notation: a graph $\boldsymbol{G}$ is pair $\boldsymbol{(V, E)}$:

- V is a finite set, whose elements are called **vertices**
  (we often take $V = \{1, \ldots, n\}$)
- E is a finite set, whose elements are **sets of two (distinct) vertices**, and are called **edges**.

Convention: $\boldsymbol{n}$ is the number of vertices, $\boldsymbol{m}$ is the number of edges.

Undirected graphs

Definition, notation: a graph $\boldsymbol{G}$ is pair $\boldsymbol{(V, E)}$:

- V is a finite set, whose elements are called **vertices**
  (we often take $V = \{1, \ldots, n\}$)
- E is a finite set, whose elements are **sets of two (distinct) vertices**, and are called **edges**.

Convention: $\boldsymbol{n}$ is the number of vertices, $\boldsymbol{m}$ is the number of edges.

Data structures:

- **adjacency lists:** an array $A[1..n]$ s.t. $A[v]$ is the **linked list** of all edges connected to v.
  $\boldsymbol{2m}$ list cells, total size $\boldsymbol{\Theta(n + m)}$, but testing if an edge exists is not $O(1)$

- **adjacency matrix:** a $(0, 1)$ matrix M of size $n \times n$, with $M[v, w] = 1$ iff $\{v, w\}$ is an edge.
  size $\boldsymbol{\Theta(n^2)}$, but testing if an edge exists is $O(1)$

Connected graphs, path, cycles, trees

Definition:

- **walk:** a sequence $\boldsymbol{v_0, \ldots, v_k}$ of vertices, with $\{v_i, v_{i+1}\}$ in E for $i = 0, \ldots, k-1$.
  length = number of steps = k ($k = 0$ is OK)

- **path:** a walk with distinct vertices

- **connected graph:** $G = (V, E)$ such that for all v, w in V, there is a path/walk $v \rightsquigarrow w$

- **cycle:** a walk $\boldsymbol{v_0, \ldots, v_k, v_0}$ with $\boldsymbol{k \geq 2}$ and v_i's distinct
  ($k = 1$ would be v_0, v_1, v_0, this is not a cycle)

- **tree:** a connected graph with no cycle
  (equiv: a connected graph with $\boldsymbol{m = n - 1}$)
  (equiv: a graph with $m = n - 1$ and no cycle)

- **rooted tree:** a tree with a special vertex called **root**

Breadth-first search

Breadth-first exploration of a graph

BFS(G, s)

G: a graph with n vertices, given by adjacency lists

s: a vertex from G

```
let Q be an empty queue
let visited be an array of size n, with all entries set to false
enqueue(s, Q)
visited[s] ← true
while Q not empty do
    v ← dequeue(Q)
    for all w neighbours of v do
        if visited[w] is false
            enqueue(w, Q)
            visited[w] ← true
```

Runtime

Anaysis:

- each vertex is enqueued at most once
- so each vertex is dequeued at most once $\quad$ $\boldsymbol{O(n)}$ for steps 5-6
- so each adjacency list is read at most once

For all v, write d_v = number of neighbours of v = length of $A[v]$ = **degree** of v.

Then total cost at step 7 is

$$O\left(\sum_v d_v\right) = O(m)$$

cf. the adjacency array A has $2m$ cells

Total: $\boldsymbol{O(n+m)}$

Correctness 1

Claim

For all vertices v, if visited$[v]$ is true at the end, there is a walk $s \rightsquigarrow v$ in G

Proof. Let $s = v_0, \ldots, v_K$ be the vertices for which visited is set to true, in this order. We prove: **for all $\boldsymbol{i}$, there is a walk $\boldsymbol{s \rightsquigarrow v_i}$** by induction.

- OK for $i = 0$
- suppose true for $v_0, \ldots, v_{i-1}$.

  when visited$[v_i]$ is set to true, we are examining the neighbours of a certain v_j, $\boldsymbol{j < i}$.

  by assumption, there is a walk $s \rightsquigarrow v_j$

  because $\{v_j, v_i\}$ is in E, there is a walk $s \rightsquigarrow v_i$

Correctness 2

Claim

For all vertices v, if there is a walk $s \rightsquigarrow v$ in G, $\mathsf{visited}[v]$ is true at the end

Proof. Let $v_0 = s, \ldots, v_k = v$ be a walk $s \rightsquigarrow v$. We prove $\mathsf{visited}[v_i]$ is true for all i by induction.

- $\mathsf{visited}[v_0]$ is true
- if $\mathsf{visited}[v_i]$ is true, we will examine all neighbours v of v_i

  so at the end of Step 7, all $\mathsf{visited}[v]$ will be true, for v neighbour of v_i

  in particular, $\mathsf{visited}[v_{i+1}]$ will be true

Correctness

Summary

For all vertices v, there is a walk $s \rightsquigarrow v$ in G **if and only if** visited$[v]$ is true at the end

Applications

- testing if there is a walk $s \rightsquigarrow v$
- testing if G is connected
- a spanning tree, if G is connected (tree that covers all vertices)

in $\boldsymbol{O(n+m)}$.

Correctness

Summary

For all vertices v, there is a walk $s \rightsquigarrow v$ in G **if and only if** visited$[v]$ is true at the end

Applications

- testing if there is a walk $s \rightsquigarrow v$
- testing if G is connected
- a spanning tree, if G is connected (tree that covers all vertices)

in $\boldsymbol{O(n+m)}$.

Exercise

For a connected graph, $n - 1 \leq m$, so $O(n+m) = O(m)$.

Keeping track of parents and levels

```
BFS(G, s)
    let Q be an empty queue
    let parent be an array of size n, with all entries set to NIL
    let level be an array of size n, with all entries set to ∞
    enqueue(s, Q)
    parent[s] ← s
    level[s] ← 0
    while Q not empty do
        v ← dequeue(Q)
        for all w neighbours of v do
            if parent[w] is NIL
                enqueue(w, Q)
                parent[w] ← v
                level[w] ← level[v] + 1
```

BFS tree

Definition: the **BFS tree** T is the subgraph made of:

- all w such that $\mathsf{parent}[w] \neq \mathbf{NIL}$.
- all edges $\{w, \mathsf{parent}[w]\}$, for w as above (except $w = s$)

Claim

The BFS tree T is a tree

Proof: T connected, n_s vertices, $n_s - 1$ edges
(n_s is the number of vertices reachable from s)

Remark: we make it a **rooted** tree by choosing s as root

Shortest paths from the BFS tree

Claim

If there is a walk $s \rightsquigarrow v$ in G then $\mathsf{level}[v] = \mathrm{dist}(s, v)$

Observation 1:

- $\mathrm{dist}(s, v) =$ length of the shortest path $s \rightsquigarrow v$
- so $\mathrm{dist}(s, v) \leq \mathsf{level}[v]$
- want: $\boldsymbol{\mathsf{level}[v] \leq \mathrm{dist}(s, v)}$

Observation 2: the levels in the queue are always of the form

$$[\boldsymbol{\ell}, \ldots, \boldsymbol{\ell}] \text{ or } [\boldsymbol{\ell}, \ldots, \boldsymbol{\ell}, \boldsymbol{\ell} + \mathbf{1}, \ldots, \boldsymbol{\ell} + \mathbf{1}]$$

so **if we dequeue v before w, $\mathsf{level}[v] \leq \mathsf{level}[w]$**

Shortest paths from the BFS tree

Claim

If there is a walk $s \rightsquigarrow v$ in G then $\mathsf{level}[v] = \mathrm{dist}(s, v)$

Proof

- take a shortest path $\boldsymbol{s = v_0 \to v_1 \to v_2 \to \cdots \to v_k = v}$ $\qquad k = \mathrm{dist}(s, v)$
- prove $\mathsf{level}[v_i] \leq i$ for all i by induction $\qquad i = k$ gives $\mathsf{level}[v] \leq \mathrm{dist}(s, v)$
- OK for $i = 0$

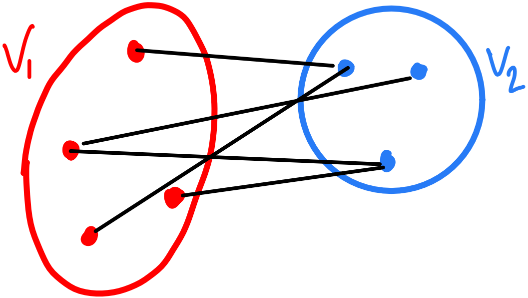
Induction step: suppose OK for $i - 1$

- the parent of v_i is either v_{i-1}, or a vertex we processed before v_{i-1}
- so in any case, $\mathsf{level}[\mathsf{parent}(v_i)] \leq \mathsf{level}(v_{i-1})$ $\qquad$ previous slide
- lhs is $\mathsf{level}(v_i) - 1$, rhs is at most $i - 1$ (induction assumption)
- done

Bipartite graphs

Definition

- a graph $G = (V, E)$ is **bipartite** if there is a partition $V = \boldsymbol{V_1} \cup \boldsymbol{V_2}$ such that all edges have **one end in** V_1 and **one end in** V_2.



Using BFS to test bipartite-ness

Claim.

Suppose G connected, run BFS from any s, and set

- V_1 = vertices with odd level
- V_2 = vertices with even level.

Then G is bipartite if and only all edges have one end in V_1 and one end in V_2 (testable in $\boldsymbol{O(n+m)}$)

Using BFS to test bipartite-ness

Claim.

Suppose G connected, run BFS from any s, and set

- V_1 = vertices with odd level
- V_2 = vertices with even level.

Then G is bipartite if and only all edges have one end in V_1 and one end in V_2 (testable in $\boldsymbol{O(m)}$)

Using BFS to test bipartite-ness

Claim.

Suppose G connected, run BFS from any s, and set

- V_1 = vertices with odd level
- V_2 = vertices with even level.

Then G is bipartite if and only all edges have one end in V_1 and one end in V_2 (testable in $\boldsymbol{O(m)}$)

Proof. $\Longleftarrow$ obvious.

For $\Longrightarrow$, let W_1, W_2 be a bipartition. Because paths alternate between W_1, W_2:

- V_1 (= vertices with odd level) is included in W_1 (say)
- V_2 (= vertices with even level) is included in W_2

So $\boldsymbol{V_1 = W_1}$ and $\boldsymbol{V_2 = W_2}$.

Subgraphs, connected components

Definition:

- **subgraph** of $G = (V, E)$: a graph $G' = (V', E')$, where
  - $V' \subset V$
  - $E' \subset E$, with all edges E' joining vertices from V'
- **connected component** of $G = (V, E)$
  - a connected subgraph of G
  - that is not contained in a larger connected subgraph of G

Let $G_i = (V_i, E_i)$, $i = 1, \ldots, s$ be the connected components of $G = (V, E)$.

- the V_i's are a partition of V, with $\sum_i n_i = n$ $\quad\quad$ $\boldsymbol{n_i = |V_i|}$
- the E_i's are a partition of E, with $\sum_i m_i = m$ $\quad\quad$ $\boldsymbol{m_i = |E_i|}$

Computing the connected components

Idea: add an outer loop that runs BFS on successive vertices

Exercise

Fill in the details.

Complexity:

- each pass of BFS $\boldsymbol{O(n_i + m_i)}$, if $G_i(V_i, E_i)$ is the ith connected component
- total $\boldsymbol{O(n + m)}$