

CS170– Spring 2022— Homework 2

CurMack

August 5, 2022

Breadth-First Search

Breadth-first search finds shortest paths in any graph whose edges have unit length.

Algorithm 1: BFS(G, s)

Input: Graph $G = (V, E)$, directed or undirected; vertex $s \in V$
Output: For all vertices u reachable from s , $dist(u)$ is set to the distance from s to u .

```
1 for all  $u \in V$  do
2    $dist(u) \leftarrow \infty$ 
3  $dist(s) \leftarrow 0$ 
4  $Q \leftarrow [s]$  (queue containing just  $s$ )
5 while  $Q$  is not empty do
6    $u \leftarrow \text{eject}(Q)$ 
7   for all edges  $(u, v) \in E$  do
8     if  $dist(v) == \infty$  then
9        $\text{inject}(Q, v)$ 
10       $dist(v) \leftarrow dist(u) + 1$ 
```

Running Time: $\mathcal{O}(|V| + |E|)$

Dijkstra's Algorithm

In summary, we can think of Dijkstra's algorithm as just BFS, except it uses a priority queue instead of a regular queue, so as to prioritize nodes in a way that takes edge lengths into account.

Algorithm 2: $\text{dijkstra}(G, l, s)$

Input: Graph $G = (V, E)$, directed or undirected; positive edge lengths $\{l_e : e \in E\}$; vertex $s \in V$ **Output:** For all vertices u reachable from s , $\text{dist}(u)$ is set to the distance from s to u

```

1 for all  $u \in V$  do
2    $\text{dist}(u) \leftarrow \infty$ 
3    $\text{prev}(u) \leftarrow \text{nil}$ 
4  $H \leftarrow \text{makequeue}(V)$  (using dist-values as keys)
5 while  $H$  is not empty do
6    $u \leftarrow \text{deletemin}(H)$ 
7   for all edges  $(u, v) \in E$  do
8     if  $\text{dist}(v) > \text{dist}(u) + l(u, v)$  then
9        $\text{dist}(v) \leftarrow \text{dist}(u) + l(u, v)$ 
10       $\text{prev}(v) \leftarrow u$ 
11       $\text{decreasekey}(H, v)$ 

```

Running Time: Since makequeue takes at most as long as $|V|$ **insert** operations, we get a total of $|V|$ **deletemin** and $|V| + |E|$ **insert/decreasekey** operations. The time needed for these varies by implementation; for instance, a binary heap gives an overall running time of $\mathcal{O}((|V| + |E|) \log |V|)$.

Bellman-Ford Algorithm

The Bellman-Ford algorithm for single-source shortest paths in general graphs:

Algorithm 3: $\text{update}((u, v) \in E)$

```

1  $\text{dist}(v) \leftarrow \min\{\text{dist}(v), \text{dist}(u) + l(u, v)\}$ 

```

Algorithm 4: $\text{shortest-paths}(G, l, s)$

Input: Directed graph $G = (V, E)$; edge lengths $\{l_e : e \in E\}$ with no negative cycles; vertex $s \in V$ **Output:** For all vertices u reachable from s , $\text{dist}(u)$ is set to the distance from s to u

```

1 for all  $u \in V$  do
2    $\text{dist}(u) \leftarrow \infty$ 
3    $\text{prev}(u) \leftarrow \text{nil}$ 
4  $\text{dist}(s) \leftarrow 0$ 
5 repeat
6   for all  $e \in E$  do
7      $\text{update}(e)$ 
8 until  $|V| - 1$  times;

```

Running Time: $\mathcal{O}(|V| \cdot |E|)$

2 Preorder, Postorder

For all v that $pre(r) < pre(v) < post(v) < post(r)$ (v is reachable from r), set $pre'(v) = pre(v) - 1$, $post'(v) = post(v) - 1$.

For all v that $pre(r) < post(r) < pre(v) < post(v)$ (v is in different connected component from r), set $pre'(v) = pre(v) - 2$, $post'(v) = post(v) - 2$.

3 Where's the Graph?

- (a) run BFS, for each number, there are 5 edges: $(+1, -1, +y, -y, /y)$

Algorithm 5: BFS2021(x, y)

```

1 for all  $u \in V$  do
2    $dist(u) \leftarrow \infty$ 
3  $dist(x) \leftarrow 0$ 
4  $Q \leftarrow [x]$  (queue containing just  $s$ )
5 while  $Q$  is not empty do
6    $u \leftarrow \text{eject}(Q)$ 
7   for all  $v \in \{u + 1, u - 1, u + y, u - y, u/y \text{ (if possible)}\}$  do
8     if  $dist(v) == \infty$  then
9        $\text{inject}(Q, v)$ 
10       $dist(v) \leftarrow dist(u) + 1$ 
11      if  $v == 2021$  then
12         $\text{return } dist(v)$ 
```

- (b) Construct the species as a graph: each specy is a vertex and an edge from x to y means y directly descended from x . Run DFS on a directed graph G , and have the pre-visit and post-visit numbers $pre(v)$, $post(v)$ for every vertex. When the program is queried, it checks whether the edge (a, b) is a back edge (a is descended from b), a tree or forward edge (b is descended from a), or a cross edge (a and b share a common ancestor but are not descended from each other)
- (c) We can view each box as a node in a directed graph, with an edge from a to b indicating that a fits in b . (see in reference) Run BFS from the smallest box x to find the shortest path to any other box in linear time on the reversed DAG.

4 The Greatest Roads in America

(See in reference) We want to build a new graph G' such that we can apply Dijkstra's algorithm on G to solve the problem.

Create G' which consists of $G_0, G_1, G_2, \dots, G_k$.

For each $v \in V$, add v_0, v_1, \dots, v_k to V' .

For each $(u, v) \in E$, add $(u_0, v_0), (u_1, v_1), \dots, (u_k, v_k)$ to E' .

For each $(v, w) \in R$, add $(v_0, w_1), (v_1, w_2), \dots, (v_{k-1}, w_k)$ to E'

call Dijkstra to find shortest $h_0 \rightarrow h_k$ path.

Runtime: $\mathcal{O}(k(m + n) \log n)$

5 Pattern Matching

(a) $\mathcal{O}(mn)$ time algorithm for this problem:

```

1 for  $i \in \{0, 1, \dots, m - n\}$  do
2    $\lfloor$  check if  $s[i : i + n - 1]$  differs from  $g$  in at most  $k$  positions.

```

Since there are $\mathcal{O}(m)$ iterations and each check takes $\mathcal{O}(n)$ time, the time complexity is $\mathcal{O}(mn)$

(b) (See in reference) use the mapping $\Phi : \{0, 1\} \rightarrow \{-1, 1\}$ on g, s and get g', s' . So $p_1(x) = a_0 + a_1x + \dots + a_{n-1}x^{n-1}$ where $a_d = g'(n - d - 1)$ for all $d \in \{0, 1, \dots, n - 1\}$. Similarly, let $p_2(x) = b_0 + b_1x + \dots + b_{m-1}x^{m-1}$ where $b_d = s'(d)$ for all $d \in \{0, 1, \dots, m - 1\}$. Noe consider $p_3(x) = p_1(x) \times p_2(x)$, the coefficients will be:

$$c_{n-1+j} = \sum_{i=0}^{n-1} a_{n-1-i} b_{j+i} = \sum_{i=0}^{n-1} g'(i) s'(j+i)$$

for any $j \in \{0, 1, \dots, m - n\}$. If these strings differ in at most k positions, then this dot product will be at least $n - 2k$. Thus we need to output all the j 's between 0 and $m - n$ such that $c_{n-1+j} \geq n - 2k$.

Runtime: $\mathcal{O}(m \log m)$