1 Conditions for Shortest Path Algorithms

Suppose we are asked to find the shortest path between two nodes in the following graph. Which algorithms can we use?

- BFS
- Dijkstra
- Bellman-Ford
- All of the above
- BFS and Dijkstra
- Dijkstra and Bellman-Ford

We have a graph with negative edge weights. Can we use Dijkstra to find shortest paths?

- Yes
- No

We have a directed graph with positive edge weights. Can we use Dijkstra to find shortest paths?

- Yes
- No

We have an undirected graph with positive edge weights. Can we use Dijkstra to find shortest paths?

- Yes
- No

2 Dijkstra's Algorithm

Suppose we are given a graph with node values A, B, C, D, E, F that have nonnegative (≥ 0) edge weights, starting from the node A. Is the middle of the algorithm our computer crashes. We look through the memory dump, and see that the state of the data structure at that time is:

```
```

Additionally from the memory dump we see that the current node when the crash happened was C.

What is the maximum possible length of the shortest path from node A to node D?

- 4
- 6
- 7
- 8
- 10

What is the maximum possible length of the shortest path from node A to node E?

- 5
- 6
- 7
- 8
- 10

What is the maximum possible length of the shortest path from node A to node F?

- 10
- 12

3 Runtime

Suppose we implement Dijkstra with a red-black tree. What is the asymptotically smallest upper bound on runtime in terms of \( n \) (the number of nodes) and \( m \) (the number of edges)?

- \( O(m \log n) \)
- \( O(m \log \log n) \)
- \( O(n \log n) \)
- \( O(n \log \log n) \)
- \( O(n \log m) \)

What if we implement Dijkstra with a Fibonacci heap? What is the asymptotically smallest upper bound on runtime in terms of \( n \) (the number of nodes) and \( m \) (the number of edges)?

- \( O(m \log n) \)
- \( O(m \log \log n) \)
- \( O(n \log n) \)
- \( O(n \log \log n) \)
- \( O(n \log m) \)

4 Implementations of Heaps in Various Languages

Our data structure keeps a collection of items, each of form \((\text{key}, \text{object})\). Implementations of heaps in various programming languages do not support the update key operation.

- Insert
- Remove
- Get minimum
- Key comparison (\(< \), \(= \), \(> \))

Suppose that we have a heap data structure that does not support updating the keys (many standard implementations of heaps in various programming languages do not support the update key operation).

- The ordering might differ in each run of Dijkstra.
- The ordering might be different in each run of Dijkstra.
- The ordering might differ in each run of Dijkstra.
- The ordering might be different in each run of Dijkstra.
- The ordering might differ in each run of Dijkstra.

If we run the Dijkstra algorithm on the graph of U.S. streets/roads/highways/etc., starting from the Stanford Oval, which of the following locations will become the current node first?

- Times Square in New York
- The Hollywood Sign
- Times Square in New York
- The Hollywood Sign
- Times Square in New York

Suppose we run Dijkstra on some graph with nodes A, B, C, D, E, F.

- What is the maximum possible length of the shortest path from node A to node B?
- What is the minimum possible length of the shortest path from node A to node B?
- What is the minimum possible length of the shortest path from node A to node C?
- What is the minimum possible length of the shortest path from node A to node D?
- What is the minimum possible length of the shortest path from node A to node E?
- What is the minimum possible length of the shortest path from node A to node F?

- 2
- 4
- 5
- 6
- 8

What is the time complexity of the shortest path algorithm in terms of \( n \) and \( m \)?

- \( O(m + n \log n) \)
- \( O(m \log n) \)
- \( O(n \log n) \)
- \( O(m \log \log n) \)
- \( O(m + n \log \log n) \)

If we run a modification of Dijkstra with the following pseudo-code:

```
while H is not empty do
    vertex v = remove the item with the smallest key from H
    for all neighbors w of v do
        if d[v] + w[v] < d[w] then
            d[w] = d[v] + w[v]
            insert vertex w into H
```

What is the asymptotically smallest upper bound on the runtime of the above code assuming that both the insert and remove operations on \( H \) take \( O(\log |\text{size of } H|) \) time? Assume that \( n = |V|, m = |E| \) (in particular \( \log \log n \approx \log \log m) \).

- \( O(n \log n) \)
- \( O(n \log \log n) \)
- \( O(n \log \log \log n) \)
- \( O(n \log \log \log \log n) \)
- \( O(n \log \log \log \log \log n) \)