Dijkstra's Algorithm: Runtime Analysis

Concept Challenge: Procedure

- **Pause** Try to solve the problem yourself
- **Discuss** with other learners (if you can)
- **Watch** the UC San Diego learners video
- **Answer** the question again
- **Confirm** your understanding with our explanation

Dijkstra: Algorithm

```
Dijkstra(S, G):
    Initialize: Priority queue (PQ), visited HashSet,
                parent HashMap, and distances to infinity
    Enqueue {S, 0} onto the PQ
    while PQ is not empty:
        dequeue node curr from front of queue
        if(curr is not visited)
            add curr to visited set
            If curr == G return parent map
            for each of curr's neighbors, n, not in visited set:
                if path through curr to n is shorter
                    update curr as n's parent in parent map
                    enqueue {n, distance} into the PQ
    // If we get here then there's no path
```

A. What is the running time of this algorithm in terms of |V| and |E|? (If more than one might be correct—which is tighter?)

A. $O(|V|^2)$

B. $O(|E| + |V|)$

C. $O(|E| * \log(|E|))$

D. $O(|E| * \log(|E|) + |V|)$

E. $O(|E|*|V|)$

Dijkstra: Algorithm

```
Dijkstra(S, G):
  Initialize: Priority queue (PQ), visited HashSet,
              parent HashMap, and distances to infinity
  Enqueue {S, 0} onto the PQ
  while PQ is not empty:
    dequeue node curr from front of queue
    if(curr is not visited)
      add curr to visited set
      If curr == G return parent map
      for each of curr's neighbors, n, not in visited set:
        if path through curr to n is shorter
          update curr as n's parent in parent map
          enqueue {n, distance} into the PQ
  // If we get here then there's no path
```

A. What is the running time of this algorithm in terms of |V| and |E|? (If more than one might be correct—which is tighter?)

A. O(|V|2)
B. O(|E| + |V|)
C. O(|E| * log(|E|))
D. O(|E| * log(|E|) + |V|)
E. O(|E|*|V|)

Walkthrough

$O(|V|)$

```
Dijkstra(S, G):
    Initialize: Priority queue (PQ), visited HashSet,
                parent HashMap, and distances to infinity
    Enqueue {S, 0} onto the PQ
    while PQ is not empty:
        dequeue node curr from front of queue
        if(curr is not visited)
            add curr to visited set
            If curr == G return parent map
            for each of curr's neighbors, n, not in visited set:
                if path through curr to n is shorter
                    update curr as n's parent in parent map
                    enqueue {n, distance} into the PQ
    // If we get here then there's no path
```

Walkthrough

```
Dijkstra(S, G):
    Initialize: Priority queue (PQ), visited HashSet,
                parent HashMap, and distances to infinity
    Enqueue {S, 0} onto the PQ
    while PQ is not empty:
        dequeue node curr from front of queue
        if(curr is not visited)
            add curr to visited set
            If curr == G return parent map
            for each of curr's neighbors, n, not in visited set:
                if path through curr to n is shorter
                    update curr as n's parent in parent map
                    enqueue {n, distance} into the PQ
    // If we get here then there's no path
```

O(|V|)
O(|1|)

Walkthrough

```
Dijkstra(S, G):
    Initialize: Priority queue (PQ), visited HashSet,
                parent HashMap, and distances to infinity
    Enqueue {S, 0} onto the PQ
    while PQ is not empty:
        dequeue node curr from front of queue
        if(curr is not visited)
            add curr to visited set
            If curr == G return parent map
            for each of curr's neighbors, n, not in visited set:
                if path through curr to n is shorter
                    update curr as n's parent in parent map
                    enqueue {n, distance} into the PQ
    // If we get here then there's no path
```

O(|V|)
~~O(|1|)~~

$O(|V|\cancel{+1})$

Walkthrough

```
Dijkstra(S, G):
    Initialize: Priority queue (PQ), visited HashSet,
                parent HashMap, and distances to infinity
    Enqueue {S, 0} onto the PQ
    while PQ is not empty:
        dequeue node curr from front of queue
        if(curr is not visited)
            add curr to visited set
            If curr == G return parent map
            for each of curr's neighbors, n, not in visited set:
                if path through curr to n is shorter
                    update curr as n's parent in parent map
                    enqueue {n, distance} into the PQ
    // If we get here then there's no path
```

O(|V|)

O(?)

Walkthrough

```
Dijkstra(S, G):
    Initialize: Priority queue (PQ), visited HashSet,
                parent HashMap, and distances to infinity
    Enqueue {S, 0} onto the PQ
    while PQ is not empty:
        dequeue node curr from front of queue
        if(curr is not visited)
            add curr to visited set
            If curr == G return parent map
            for each of curr's neighbors, n, not in visited set:
                if path through curr to n is shorter
                    update curr as n's parent in parent map
                    enqueue {n, distance} into the PQ
    // If we get here then there's no path
```

$\boxed{O(|V|)}$

$\boxed{O(|E|)}$

Walkthrough

```
Dijkstra(S, G):
    Initialize: Priority queue (PQ), visited HashSet,
                parent HashMap, and distances to infinity
    Enqueue {S, 0} onto the PQ
    while PQ is not empty:
        dequeue node curr from front of queue
        if(curr is not visited)
            add curr to visited set
            If curr == G return parent map
            for each of curr's neighbors, n, not in visited set:
                if path through curr to n is shorter
                    update curr as n's parent in parent map
                    enqueue {n, distance} into the PQ
    // If we get here then there's no path
```

O(|V|)

O(|E|)

Most lines in the loop are O(1)

Walkthrough

```
Dijkstra(S, G):
    Initialize: Priority queue (PQ), visited HashSet,
                parent HashMap, and distances to infinity
    Enqueue {S, 0} onto the PQ
    while PQ is not empty:
        dequeue node curr from front of queue
        if(curr is not visited)
            add curr to visited set
            If curr == G return parent map
            for each of curr's neighbors, n, not in visited set:
                if path through curr to n is shorter
                    update curr as n's parent in parent map
                    enqueue {n, distance} into the PQ
    // If we get here then there's no path
```

O(|V|)

O(|E|)

O(log |E|)

Walkthrough

```
Dijkstra(S, G):
    Initialize: Priority queue (PQ), visited HashSet,
                parent HashMap, and distances to infinity
    Enqueue {S, 0} onto the PQ
    while PQ is not empty:
        dequeue node curr from front of queue
        if(curr is not visited)
            add curr to visited set
            If curr == G return parent map
            for each of curr's neighbors, n, not in visited set:
                if path through curr to n is shorter
                    update curr as n's parent in parent map
                    enqueue {n, distance} into the PQ
    // If we get here then there's no path
```

O(|V|)

O(|E|)

O(log |E|)

O(log |E|)

Insert and remove from a Priority Queue with N elements is log(N)

Walkthrough

Dijkstra(S, G):
    Initialize: Priority queue (PQ), visited HashSet,
                parent HashMap, and distances to infinity
    Enqueue {S, 0} onto the PQ
    while PQ is not empty:
        dequeue node curr from front of queue
        if(curr is not visited)
            add curr to visited set
            If curr == G return parent map
            for each of curr's neighbors, n, not in visited set:
                if path through curr to n is shorter
                    update curr as n's parent in parent map
                    enqueue {n, distance} into the PQ
    // If we get here then there's no path

$O(|V|)$

$O(|E|)$

$O(\log |E|)$

But what about this for loop?

$O(\log |E|)$

Insert and remove from a Priority Queue with N elements is log(N)

Walkthrough

```
Dijkstra(S, G):
    Initialize: Priority queue (PQ), visited HashSet,
                parent HashMap, and distances to infinity      O(|V|)
    Enqueue {S, 0} onto the PQ
    while PQ is not empty:                                     O(|E|)
        dequeue node curr from front of queue                  O(log |E|)
        if(curr is not visited)
            add curr to visited set
            If curr == G return parent map
            for each of curr's neighbors, n, not in visited set:
                if path through curr to n is shorter
                    update curr as n's parent in parent map
                    enqueue {n, distance} into the PQ          O(log |E|)
    // If we get here then there's no path
```

It's already accounted for!

Insert and remove from a Priority Queue with N elements is log(N)

Walkthrough

```
Dijkstra(S, G):
    Initialize: Priority queue (PQ), visited HashSet,
                parent HashMap, and distances to infinity
    Enqueue {S, 0} onto the PQ
    while PQ is not empty:
        dequeue node curr from front of queue
        if(curr is not visited)
            add curr to visited set
            If curr == G return parent map
            for each of curr's neighbors, n, not in visited set:
                if path through curr to n is shorter
                    update curr as n's parent in parent map
                    enqueue {n, distance} into the PQ
    // If we get here then there's no path
```

$\boxed{O(|V|)}$

$\boxed{O(|E| \log |E|)}$

Walkthrough

```
Dijkstra(S, G):
    Initialize: Priority queue (PQ), visited HashSet,
                parent HashMap, and distances to infinity
    Enqueue {S, 0} onto the PQ
    while PQ is not empty:
        dequeue node curr from front of queue
        if(curr is not visited)
            add curr to visited set
            If curr == G return parent map
            for each of curr's neighbors, n, not in visited set:
                if path through curr to n is shorter
                    update curr as n's parent in parent map
                    enqueue {n, distance} into the PQ
    // If we get here then there's no path
```

O(|V|)

O(|E| log |E|)

Overall:
O(|E| log |E| + |V|)

Walkthrough

```
Dijkstra(S, G):
    Initialize: Priority queue (PQ), visited HashSet,
                parent HashMap, and distances to infinity
    Enqueue {S, 0} onto the PQ
    while PQ is not empty:
        dequeue node curr from front of queue
        if(curr is not visited)
            add curr to visited set
            If curr == G return parent map
            for each of curr's neighbors, n, not in visited set:
                if path through curr to n is shorter
                    update curr as n's parent in parent map
                    enqueue {n, distance} into the PQ
    // If we get here then there's no path
```

`O(|V|)`

`O(|E| log |E|)`

Overall:
O(|E| log |E| + |V|)

Because E <= $|V|^2$ and $\log(|V|^2)$ is just $O(\log(|V|))$ we could tighten to:
O(|E| log |V| + |V|)