Dijkstra’s Algorithm

https://cs.pomona.edu/classes/cs140/
Dijkstra’s Single-Source Shortest Path Algorithm
Outline

**Topics and Learning Objectives**
- Discuss graphs with edge weights
- Discuss shortest paths
- Discuss Dijkstra’s algorithm including a proof

**Exercise**
- Dijkstra’s Algorithm
Extra Resources

• Introduction to Algorithms, 3rd, chapter 24
• Algorithms Illuminated Part 2: Chapter 9
Dijkstra’s Algorithm

Find the shortest path between a start vertex $s$ and every other vertex in the graph $G$.

Can halt the algorithm if you only want to find shortest path to a specific vertex (for example, a destination city).

Uses:
- Network routing
- Path planning
- Etc.
Dijkstra’s Algorithm

Find the shortest path between a start vertex $s$ and every other vertex in the graph $G$

Can halt the algorithm if you only want to find shortest path to a specific vertex (for example, a destination city)

Uses:
• Network routing
• Path planning
• Etc.
Dijkstra’s Algorithm

Input
• A weighted graph $G = (V,E)$ and
• A source vertex $s$

Output
• for all $v$ in $V$ we output the length of the shortest path from $s \rightarrow v$
• you can also output the actual path, but we’ll just worry about length for now

Assumptions
• A path exists from $s$ to every other node (how can we check this property?)
• All edge weights are non-negative
What is the shortest path from S to all other vertices?
How did we do shortest path before?

• BFS

• How can we modify that process to work for graphs with weighted edges?

• Why would we not want to do that?
FUNCTION Dijkstra(G, start_vertex)
    found = {}
    lengths = {v: INFINITY FOR v IN G.vertices}

    found.add(start_vertex)
    lengths[start_vertex] = 0

    WHILE found.length != G.vertices.length
        FOR v IN found
            FOR vOther, weight IN G.edges[v]
                IF vOther NOT IN found
                    vOther_length = lengths[v] + weight
                    IF vOther_length < min_length
                        min_length = vOther_length
                        vMin = vOther

            found.add(vMin)
            lengths[vMin] = min_length

    RETURN lengths
FUNCTION Dijkstra(G, start_vertex)
    found = {}
    lengths = {}
    FOR v IN G.vertices:
        lengths[v] = INFINITY
    found.add(start_vertex)
    lengths[start_vertex] = 0
    WHILE found.length != G.vertices.length:
        FOR v IN found:
            FOR vOther, weight IN G.edges[v]:
                IF vOther NOT IN found:
                    vOther_length = lengths[v] + weight
                    IF vOther_length < min_length:
                        min_length = vOther_length
                        vMin = vOther
        found.add(vMin)
        lengths[vMin] = min_length
    RETURN lengths

V - found (set)

found (set)
FUNCTION Dijkstra(G, start_vertex)

found = {}
lenghts = {v: INFINITY FOR v IN G.vertices}

found.add(start_vertex)
lenghts[start_vertex] = 0

WHILE found.length != G.vertices.length

FOR v IN found
  FOR vOther, weight IN G.edges[v]
    IF vOther NOT IN found
      vOther_length = lengths[v] + weight
      IF vOther_length < min_length
        min_length = vOther_length
        vMin = vOther

found.add(vMin)
lenghts[vMin] = min_length

RETURN lenghts
FUNCTION Dijkstra(G, start_vertex)
    found = {}
    lengths = {v: INFINITY FOR v IN G.vertices}
    found.add(start_vertex)
    lengths[start_vertex] = 0

    WHILE found.length != G.vertices.length
        FOR v IN found
            FOR vOther, weight IN G.edges[v]
                IF vOther NOT IN found
                    vOther_length = lengths[v] + weight
                    IF vOther_length < min_length
                        min_length = vOther_length
                        vMin = vOther
        found.add(vMin)
        lengths[vMin] = min_length

    RETURN lengths
Exercise
Dijkstra’s Algorithm with negative edges

• How might you deal with negative edges?
• How about adding some value to every edge?

What is the shortest path from s to t?
Dijkstra’s Algorithm with negative edges

• How might you deal with negative edges?
• How about adding some value to every edge?

What is the shortest path from s to t?
Dijkstra’s Algorithm with negative edges

• How might you deal with negative edges?
• How about adding some value to every edge?

What is the shortest path from s to t?

We would add a different amount to each path!
Dijkstra’s Algorithm

• What have we done so far?
  • We’ve only shown that it works for the given example.
  • This is not enough to prove correctness.

• In general, examples are good for:
  • Demonstration
  • Contradictions

• They are not good for proving correctness.
Proof by Induction Cheat-sheet

Proof by induction that \( P(n) \) holds for all \( n \)

1. \( P(1) \) holds because <something about the code/problem>
2. Let’s assume that \( P(k) \) (where \( k < n \)) holds.
3. \( P(n) \) holds because of \( P(k) \) and <something about the code>
4. Thus, by induction, \( P(n) \) holds for all \( n \)
Correctness

**Theorem** for Dijkstra’s algorithm:

For every graph with non-negative edge lengths, Dijkstra’s algorithm computes all shortest path distances from `start_vertex` to every other vertex.

**Base Case:**

- `lengths[start_vertex] = 0`

Proof by induction that $P(n)$ holds for all $n$

- $P(1)$ holds because ...
- Let’s assume that $P(k)$ (where $k < n$) holds.
- $P(n)$ holds because of $P(k)$ and ...
- Thus, by induction, $P(n)$ holds for all $n$
Correctness

**Theorem** for Dijkstra’s algorithm:
For every graph with non-negative edge lengths, Dijkstra’s algorithm computes all shortest path distances from `start_vertex` to every other vertex

**Inductive Hypothesis:**
• Assume all previous iterations produce correct shortest paths
• For all `v` in `found`, `lengths[v] = shortest path length from start_vertex to v`
FUNCTION Dijkstra(G, start_vertex)
found = {}
lenghts = {v: INFINITY FOR v IN G.vertices}
found.add(start_vertex)
lenghts[start_vertex] = 0

WHILE found.length != G.vertices.length
    FOR v IN found
        FOR vOther, weight IN G.edges[v]
            IF vOther NOT IN found
                vOther_length = lengths[v] + weight
                IF vOther_length < min_length
                    min_length = vOther_length
                    vMin = vOther
            found.add(vMin)
lenghts[vMin] = min_length

RETURN lengths

Proof by induction that P(n) holds for all n
• P(1) holds because ...
• Let’s assume that P(k) (where k < n) holds.
  • P(n) holds because of P(k) and ...
• Thus, by induction, P(n) holds for all n
Inductive Step

In the current iteration:

• We pick an edge \((v^*, v_{\text{Min}})\) based on Dijkstra’s greedy criterion
• add \(v_{\text{Min}}\) to \(\text{found} \)
• Set the path length of \(v_{\text{Min}} \rightarrow \text{lengths}[v_{\text{Min}}] = \text{lengths}[v^*] + \text{weight}_{v^*,v_{\text{Min}}}\)

What do we know about \(\text{lengths}[v^*]\)?

• Optimal path to \(v^*\), and we won’t find a better path to \(v_{\text{Min}}\)

Our inductive hypothesis states that it is the minimal path length

How do we prove this? Loop Invariant

Proof by induction that \(P(n)\) holds for all \(n\)

• \(P(1)\) holds because ...
• Let’s assume that \(P(k)\) (where \(k < n\)) holds.
• \(P(n)\) holds because of \(P(k)\) and ...
• Thus, by induction, \(P(n)\) holds for all \(n\)
Inductive Step

In the current iteration:

• We pick an edge \((v^*, v_{\text{Min}})\) based on Dijkstra’s greedy criterion
• add \(v_{\text{Min}}\) to found
• Set the path length of \(v_{\text{Min}} \rightarrow \text{lengths}[v_{\text{Min}}] = \text{lengths}[v^*] + \text{weight}_{v^*, v_{\text{Min}}}\)

What do we know about \(\text{lengths}[v^*]\)?

• Optimal path to \(v^*\), and we won’t find a better path to \(v_{\text{Min}}\)

How do we prove this? Loop Invariant

By our inductive hypothesis, our theorem for Dijkstra’s is correct
Correctness

How many different types of paths do we consider each iteration?
Correctness

Dijkstra’s says that this is the best available path.

some positive path length
Correctness

How do we know that the path from $v^*$ to $v_{\text{Min}}$ is better than the path from $v^*$ to $y$?

Both include the path from $s$ to $v^*$, and Dijkstra’s Algorithm always picks the minimal path length.
Correctness

How do we know that the path from $v^*$ to $y$ to $v_{\text{Min}}$ is not even better than the path from $v^*$ to $v_{\text{Min}}$?

Dijkstra’s Algorithm only operates on graphs with positive edge weights. Thus, this new path must be greater than or equal to the $(v^*, v_{\text{Min}})$ edge.
How do we know that the path from $v^*$ to $v_{\text{Min}}$ is better than the path from $x$ to $v_{\text{Min}}$?

Dijkstra’s Algorithm compares these two options and picks the minimal path length.
How do we know that the path from \( x \) to \( y \) to \( v_{\text{Min}} \) is not even better than the path from \( v^* \) to \( v_{\text{Min}} \)?

Dijkstra's Algorithm only operates on graphs with positive edge weights. Thus, this new path must be greater than or equal to the \((v^*, v_{\text{Min}})\) edge.
**Not taking the shortest edge. We are taking the shortest path!**

Some positive path length

Not taking the shortest edge. We are taking the shortest path!
Sometimes the shortest edge is on the shortest path.
Why doesn’t Dijkstra’s work on graphs with negative edges?
Correctness (summary)

• Given our assumption that we do not have negative edges
• And our inductive hypothesis that our path to $v^*$ is the shortest
• And our analysis of Dijkstra’s greedy criterion

• We have shown that

$$\text{lengths}[v_{\text{Min}}] = \text{lengths}[v^*] + \text{weight}_{v^*,v_{\text{Min}}}$$

is the best available path length.
FUNCTION Dijkstra(G, start_vertex)

found = {}
lenghts = {v: INFINITY FOR v IN G.vertices}

found.add(start_vertex)
lenghts[start_vertex] = 0

WHILE found.length != G.vertices.length

FOR v IN found

    FOR vOther, weight IN G.edges[v]

        IF vOther NOT IN found

            vOther_length = lengths[v] + weight

            IF vOther_length < min_length

                min_length = vOther_length

                vMin = vOther

            found.add(vMin)

        lengths[vMin] = min_length

RETURN lengths

What is the running time?
FUNCTION Dijkstra(G, start_vertex)
    found = {}
    lengths = {v: INFINITY FOR v IN G.vertices}

    found.add(start_vertex)
    lengths[start_vertex] = 0

    WHILE found.length != G.vertices.length
        FOR v IN found
            FOR vOther, weight IN G.edges[v]
                IF vOther NOT IN found
                    vOther_length = lengths[v] + weight
                    IF vOther_length < min_length
                        min_length = vOther_length
                        vMin = vOther
                    found.add(vMin)
                    lengths[vMin] = min_length

    RETURN lengths

What is the running time?
O(n)

How many times does the outer loop run?
O(n)

How many times do the inner two loops run?
O(m)
FUNCTION Dijkstra(G, start_vertex)
    found = {}
    lengths = {v: INFINITY FOR v IN G.vertices}
    found.add(start_vertex)
    lengths[start_vertex] = 0

    WHILE found.length != G.vertices.length
        FOR v IN found
            FOR vOther, weight IN G.edges[v]
                IF vOther NOT IN found
                    vOther_length = lengths[v] + weight
                    IF vOther_length < min_length
                        min_length = vOther_length
                        vMin = vOther
            found.add(vMin)
            lengths[vMin] = min_length

    RETURN lengths

What is the running time?

How many times does the outer loop run?
\(O(n)\)

How many times do the inner two loops run?
\(O(m)\)

\(O(nm)\)