Dijkstra's Algorithm

Dijkstra's algorithm computes the shortest path to all other vertices in a directed graph from a given source vertex, given that each edge has a non-negative length.

1 Algorithm

```
def Dijkstra(G, s):
    for x in V:
        d[x] = ∞
        p[x] = None
    d[s] = 0
    Q = V

    while Q not empty:
        Pick x in Q to minimize d[x]
        Q -= {x}

        for (x, y) in E:
            if d[x] + w(x, y) < d[y]:
                d[y] = d[x] + w(x, y)
                p[y] = *x

    return d, p
```

All shortest path lengths are in `d` when the algorithm finishes, and `p` contains a pointer to the parent vertex in the shortest path for each vertex. To get the shortest path to `v` as a series of vertices, run a DFS on `p` starting at `p[v]` and reverse the result.

2 Analysis

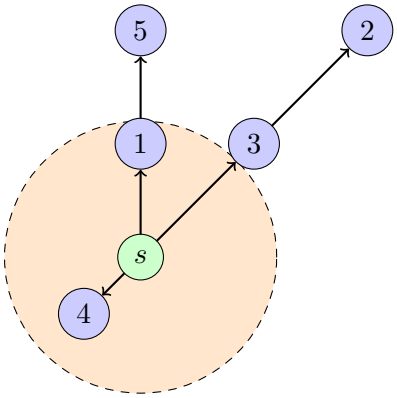
If `d` is kept as a standard array, then the algorithm runs in $\Theta(n^2)$ (obvious). However, the way `d` is used in the algorithm naturally lends itself to a min-heap representation. If we use a min-heap for `d`, then the runtime for the algorithm is

$$\Theta(n \lg n + \sum_{x \in V} |\text{adj}(x)| \lg n) \tag{1}$$

Because $\sum |\text{adj}(x)| \leq |E|$, the expression simplifies to $\Theta(n \lg n + m \lg n)$, which for a dense graph ($m >> n$) is $\Theta(m \lg n)$.

3 Proof

To illustrate the concept behind the proof, imagine G as a radial graph:



The main ideais that at any given iteration, `d` contains the shortest possible paths to all vertices within the circle of some radius. On the next iteration, the radius is expanded just enough to include one more vertex in the circle.

3.1 Proof by Induction

Base Case Denoting $L[x]$ as the actual shortest-path distance to x, the base case is trivial: $d[s] = L[s] = 0$.

Inductive Hypothesis The inductive hypothesis is that all previous iterations were correct. That is, $\forall x \in V - Q, d[x] = L[x]$.

For any path from s to w not computed already, it must cross the frontier defined by the circle in the diagram. This means that for any $s \to w$ path, it must visit at least one vertex outside the frontier. Call this first-outside vertex z, which is not w (otherwise path will be the same as computed path).

If the last vertex inside the frontier visited by the path is y, then we know that

$$len(P) \geq L[y] + c(y, z) \tag{2}$$

In the best case, all edges are 0 to w after crossing to z. Because all edges were defined to be non-negative and the algorithm picks a direct $y \to w$ path to minimize $L[y] + c(y, z)$ where $z = w$, the computed distance $d[y]$ is indeed the shortest $s \to z$ path possible, and by induction, Dijkstra's algorithm correctly solves the single-source shortest paths problem.