Dijkstra's Algorithm

Algorithm Dijkstra

input: A weighted directed graph $G = (V, E)$

output: Distances array $\lambda[1..n]$ where $\lambda[y]$ is the distance from 1 to y.

$X \leftarrow \{1\}$; $Y \leftarrow V - \{1\}$; $\lambda[1] \leftarrow 0$;
for $y \leftarrow 2$ to n **do**
    if y is adjacent to 1 **then**
        $\lambda[y] \leftarrow length[1, y]$
    else
        $\lambda[y] \leftarrow \infty$
    end if
end for
for $j \leftarrow 1$ to $n - 1$ **do**
    let $y \in Y$ be the vertex with the min λ
    $X \leftarrow X \cup \{y\}$; $Y \leftarrow Y - \{y\}$
    for each edge (y, w) **do**
        $z \leftarrow \lambda[y] + length[y, w]$
        if $w \in Y$ and $z < \lambda[w]$ **then** $\lambda[w] \leftarrow z$
    end for
end for

Remarks

1. **Running Time** $= \Theta(m + n^2) = \Theta(n^2)$ where $m = |E|$. This is because finding the $\min_{y \in Y} \lambda[y]$ costs $\Theta(n^2)$ in total. **Extra space** $= \Theta(n)$ ..why?

2. **Implementation**: The graph G can be saved as adjacency list which costs $\Theta(m + n)$ space. For the sets X and Y we can use only one binary array $X[1..n]$ where initially X =[1 0 0 0 0 0 ... 0]. The operation $X \leftarrow X\ \{y\}$ can be implemented by setting $X[y] = 1$. The set Y can be obtained from X as it has the opposite content.

3. **Improving Dijkstra's:** The running time can be improved if $m = o(n^2)$ by using min-heap to maintain the values $\lambda[y]$ and extract the min in constant time. Updating the heap takes $O(\log n)$ and there could be at most m updates (because when a y is moved to X, the λ-values of its neighbors in Y have to be updated.)