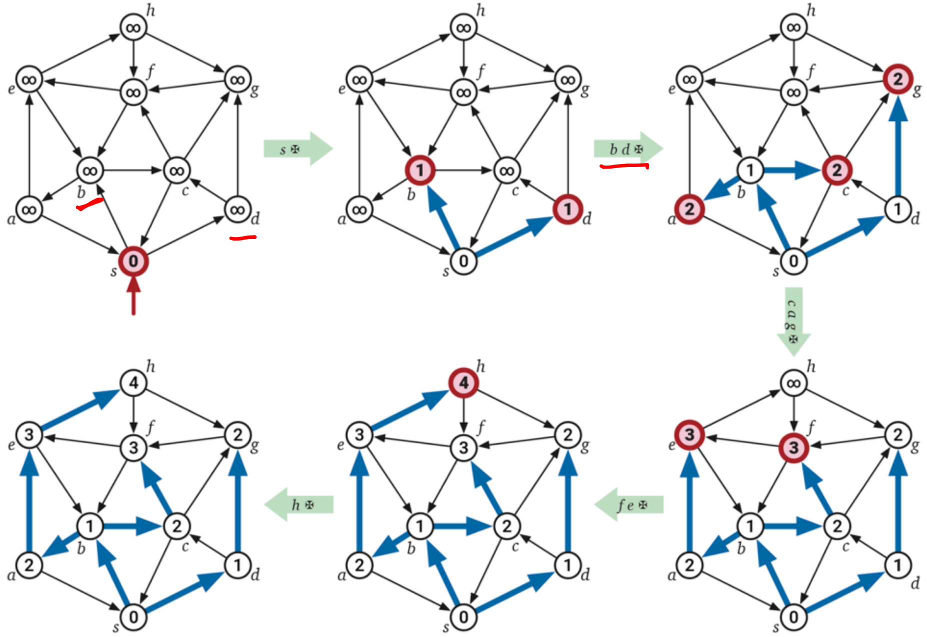
Algorithm Design & Analysis (CSE222)

Lecture-18

Outline

- Union Find data Structure

- Shortest Path
  - Using BFS & token




```
BFSWithToken(s):
    InitSSSP(s)
    Push(s)
    Push(✠)                     ⟨⟨start the first phase⟩⟩
    while the queue contains at least one vertex
        u ← Pull( )
        if u = ✠
            Push(✠)             ⟨⟨start the next phase⟩⟩
        else
            for all edges u→v
                if dist(v) > dist(u) + 1        ⟨⟨if u→v is tense⟩⟩
                    dist(v) ← dist(u) + 1
                    pred(v) ← u                 ⟨⟨relax u→v⟩⟩
                    Push(v)
```

Outline

- Shortest Path
  - Dijkstra's algorithm (single source)

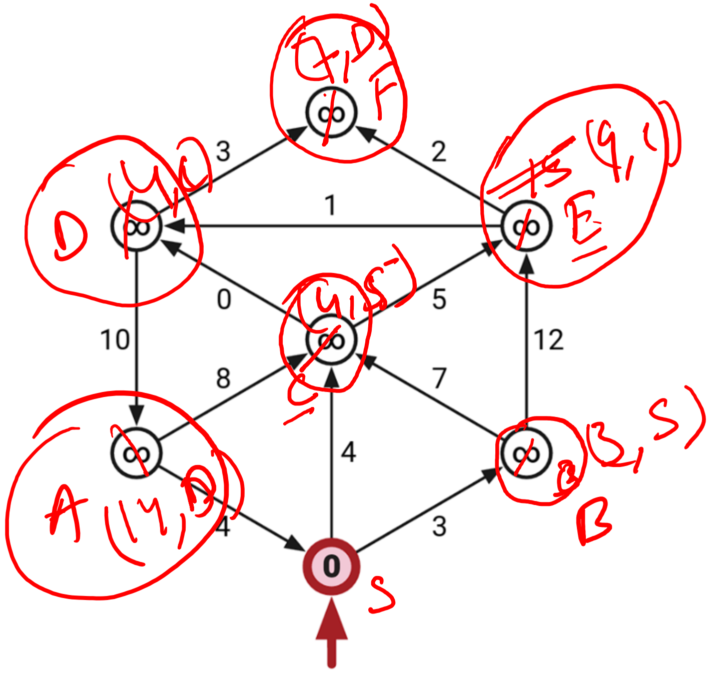
Dijkstra's Algorithm

For an edge u→v
Tense: If dist(v) > dist(u) + w(u→v)
Relaxation: dist(v) = dist(u) + w(u→v)

Dijkstra's Algorithm

1. Start with a source.
2. Check tense edge from visited vertex set.
3. Relax such edge.
4. Include the vertex with smallest distance.
5. Repeat from 2 to 4.

Running time: $O(|V|^2)$



S, C, D, A

Dijkstra's Algorithm

```
DIJKSTRA(s):
    INITSSSP(s)
    INSERT(s, 0)
    while the priority queue is not empty
        u ← EXTRACTMIN()
        for all edges u→v
            if u→v is tense
                RELAX(u→v)
                if v is in the priority queue
                    DECREASEKEY(v, dist(v))
                else
                    INSERT(v, dist(v))
```

Dijkstra's Algorithm

<u>Claim:</u> If G has no negative-weighted edge then for all u_i and u_j, such that $i < j$, we have $dist(u_i) < dist(u_j)$.

<u>Proof:</u> Fixing an arbitrary index i we have following two cases

- If G have the edge $u_i \rightarrow u_{i+1}$, and this edge has been relaxed in the ith iteration. Then we have $dist(u_{i+1}) = dist(u_i) + w(u_i \rightarrow u_{i+1}) > dist(u_i)$.
- Else either G does not have the edge $u_i \rightarrow u_{i+1}$ or the edge has not been relaxed in the ith iteration. In this iteration, the priority queue must have, $dist(u_{i+1}) > dist(u_i)$.

Dijkstra’s Algorithm

Claim: If G has no negative-weighted edge then each vertex will be visited at most once.

Proof: For a vertex v, prove that for any i and j iteration, where i < j, it is impossible to get dist(v) at ith iteration is smaller than dist(v) in jth iteration.

Dijkstra's Algorithm

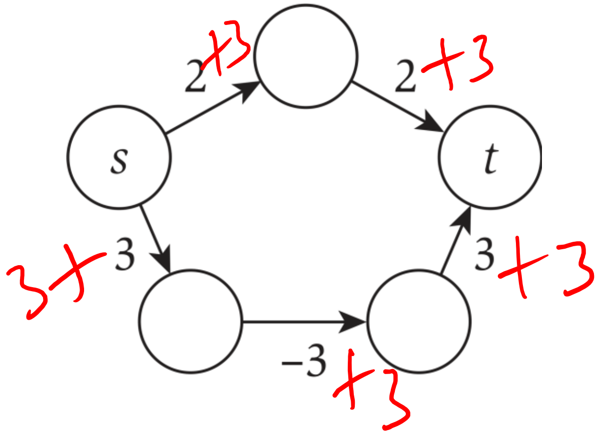
Claim: If G has no negative-weighted edge, then after the end of Dijkstra, dist(u) is the length of the shortest path in G from s to u for every vertex u.

Proof: Consider an arbitrary path in G, $u_0 \rightarrow u_1 \rightarrow \ldots \rightarrow u_t$, where $s = u_0$ and $u = u_t$.

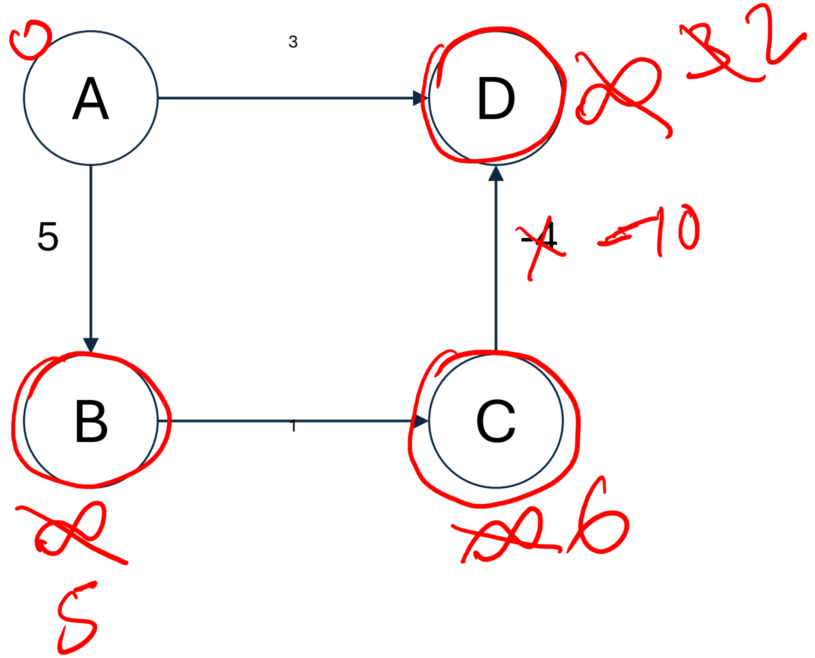
- Base: At t = 0, $dist(u_0) = dist(s) = 0$.
- For $j > 0$, the induction hypothesis is $dist(u_j) \leq L_j$
- After visiting u_j either $dist(u_{j+1}) \leq dist(u_j)+w(u_j \rightarrow u_{j+1})$ or we set $dist(u_{j+1}) = dist(u_j)+w(u_j \rightarrow u_{j+1})$
- In either way $dist(u_{j+1}) \leq dist(u_j)+w(u_j \rightarrow u_{j+1}) = L_j + w(u_j \rightarrow u_{j+1}) = L_{j+1}$.

Negative Edges

Add smallest weight to all the edges.



Negative Edges



Reference

Slides

Jeff Erickson Chp-8