17 Shortest Paths in Graphs with Negative Weights

- Shortest paths in graphs with negative weights
- Next: network flow

17.1 Shortest Paths in Graphs with Negative Weights

Problem: Given a digraph $G = (V, E)$ with edge weights c_{uv} for each edge $(u, v) \in E$ and vertices $s, t \in V$, find a minimum cost path from s to t, where the cost of a path is the sum of the edge weights.

Assumption: G does not contain any negative cost cycles. If there are negative cost cycles, then our lowest possible cost would be arbitrary low.

We can't necessarily use Dijkstra's algorithm, because it greedily selects the lowest weight edge that is adjacent to the nodes that we have visited, which only works when there are no negative edges.

17.1.1 Bellman-Ford Algorithm

Main idea: let $\text{opt}(i, v)$ = minimum cost of a path from v to t using at most i edges.

Why is the memo not too big?

- only n vertices v.
- the minimum cost path is simple, so we only need to consider paths of length $\leq n - 1$.

Does the minimum cost path for $\text{opt}(i, v)$ use all i edges?

- no: $\text{opt}(i, v) = \text{opt}(i - 1, v)$
- yes: $\text{opt}(i, v) = \min_{u \in V}(\text{opt}(i - 1, u) + c_{uv})$, where $c_{uv} = \infty$ if $(u, v) \notin E$.

Initial conditions: $\text{opt}(0, t) = 0$, $\text{opt}(0, v) = \infty$ for all $v \neq t$.

Our goal is to compute $\text{opt}(n - 1, s)$.

```
BF(G, c, s, t):
    let n = number of vertices of G
    let M be an nxn array
    let M[0, t] = 0 and M[0, v] = infty for all v != t
    for i=1 to n-1 do
        for v in V do
            let M[i, v] = min{M[i-1, v],
                              min{M[i-1, u] + c_uv for u in V}}
        endfor
    endfor
    return M[n-1, s]
```

The total running time of this algorithm is $O(n \cdot \sum_{v \in V}(d_v + 1)) = O(n \cdot (m + n))$, where d_v is the outdegree of v.