DP and Shortest Paths

Frank

October 29, 2021

1 Bellman-Ford (single source)

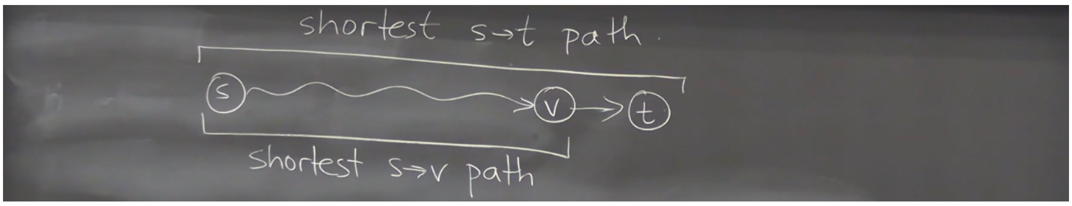
1.1 Problem

- **Input:**
  - $G = (V, E)$ directed graph, $n = |V|$ and $m = |E|$
  - $s \in V$ source node
  - $wt : E \to \mathbb{R}$

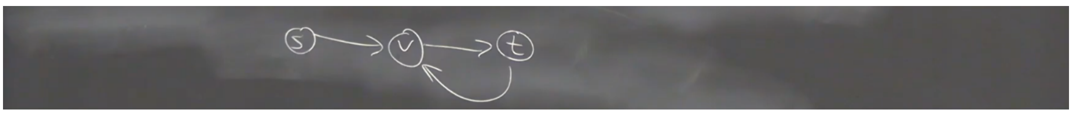
  Assume G has no negative weight cycles, since you can go around the cycle an infinite amount of times. However, a negative weighted edge is fine, as opposed to Dijkstra's
- **Output:**
  - wt of shortest $s \to u$ path $\forall u \in V$

1.2 Subproblem

Consider the shortest $s \to t$ path, what is the optimal substructure?



We need to find the shortest $s \to v$ path. However, there is a problem, if there are cycles, then v is the predecessor of t but t is also the predecessor of v.

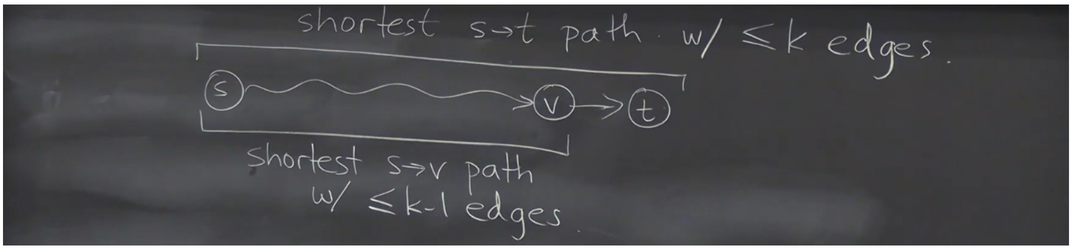


Thus, we need extra parameters (create extra subproblems).

- Thus, the definition of our subproblems is $L[u, k]$ = minimum weight of $s \to u$ path using at most k edges, $u \in V$, $k \geq 0$ $(\star)$

  Then note that $L[u, k] \leq L[u, k-1]$, because increasing the number of edges we can use will not decrease our minimum weight path

Now go back to our previous problem, we now consider the shortest $s \to t$ path with at most k edges.



Now the optimal substructure becomes restricted to those paths with at most $k-1$ edges. So compute $L(v, k-1)$ for all $v : (v, u) \in E$ and add the corresponding edge from $v \to t$ to find the minimum weighted path for a substructure.

Then our recursive formula is

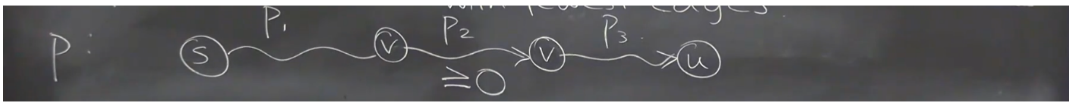
$$L[u,k] = \begin{cases} 0 & k = 0 \text{ and } u = s \\ \infty & k = 0 \text{ and } u \neq s \\ min_{v:(v,u)\in E}\{L[v,k-1] + wt(v,u)\} & k > 0 \end{cases} \qquad (\dagger)$$

Now we need an upper bound on k, because we could go on forever.

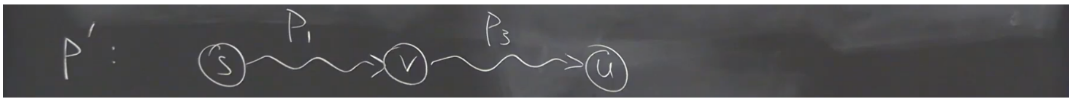
- **Claim 1:** If G has no negative-weighted cycles reachable from s, then $\forall u\ \exists$ a shortest $s \to u$ path that has at most $n-1$ edges

  Proof: let P = shortest $s \to u$ path with fewest edges, then no node repeats.

  If a node repeats, then we have a cycle, but since we have no negative-weighted cycles by assumption, then we have that $P_2 \geq 0$



Now consider P' without the cycle



We have that P' has weight less than or equal to P, but has fewer edges, which contradicts our original definition of P. Thus no node repeats for P.

Since P has at most n nodes (no node repeats for P), then we have that P has at most $n-1$ edges.

Thus, we can set $k \leq n-1$

- **Claim 2:** G has no neg-wt cycle reachable from $s \iff \forall u,\ L[u,n] = L[u,n-1]$

  Proof ($\implies$): Follows from Claim 1.

  Since claim 1 states that if there are no negative-weighted cycles in G, then the shortest path has at most $n-1$ edges. So $L[u,n] \geq L[u,n-1]$, but note that L cannot increase when we increase k, so we get equality.

  Proof ($\impliedby$): Follows from:

  $\forall u,\ L[u,k] = L[u,k-1] \implies \forall u,\ L[u,k+1] = L[u,k]$, or that if no node changes when we increase maximum path length, then no nodes will change after that.

  Note that $L[u,k+1] = min_{v:(v,u)\in E}\{L[v,k] + wt(v,u)\}$ by (†), but also note that $L[v,k] = L[v,k-1]$ by assumption, so

$$
\begin{aligned}
L[u,k+1] &= min_{v:(v,u)\in E}\{L[v,k] + wt(v,u)\} \\
&= min_{v:(v,u)\in E}\{L[v,k-1] + wt(v,u)\} \\
&= L[u,k]
\end{aligned}
$$

  Thus, it follows that L never changes after $k-1$ edges, and therefore G has no negative-weighted cycles, because if it does L would decrease eventually.

Now we can use claim 2 to verify whether G has a negative-weighted cycle. All we have to do is confirm that $\forall u,\ L[u,n] = L[u,n-1]$

This also means that at any point in our algorithm, if $L[u,k]$ does not change for any u, then we can simply stop there, since it will never change in the future by claim 2.

With claim 1 and claim 2, we can bound $k \leq n-1$, and we can also stop earlier, making the algorithm more efficient.

1.3 Algorithm

procedure BF(G, s, wt)
    $L[s, 0] \leftarrow 0$
    for $(u \in V - \{s\})$ **do**
        $L[u, 0] \leftarrow \infty$
    for $(k = 1 \rightarrow n)$ **do**
        for $(u \in V)$ **do**
            $L[u, k] \leftarrow L[u, k-1]$
            for (each $v \in V : (v, u) \in E$) **do**
                if $(L[u, k] > L[v, k-1] + wt(v, u))$ **then**
                    $L[u, k] = L[v, k-1] + wt(v, u)$
    if $(\exists u : L[u, n] \neq L[u, n-1])$ **then**
        return We have a negative-weighted cycle reachable from s
    else
        return $L[-, n]$ ▷ Return last column (for $k = n$)

1.4 Running Time

- **Adj matrix:** $O(n^3)$
- **Adj list:** $O(n \cdot m)$, better than $O(n^3)$ if graph is sparce, same if graph is dense.

1.5 Additional Problems

- Finding the actual negative-weighted cycle instead of its existence.

  Claim 3: If $L[u, n] \neq L[u, n-1]$, then if P is a minimum weight $s \rightarrow u$ path using at most n edges, then

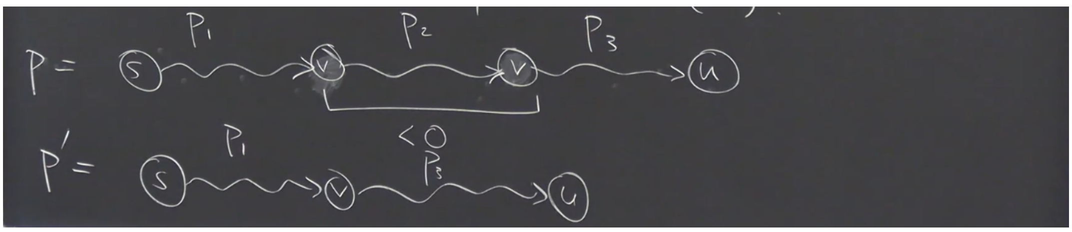
  1. P has a cycle
  2. Every cycle on P has $wt < 0$

     Proof:

     Let $P =$ any min-weight $s \rightarrow u$ path with at most n edges

     Since by assumption, $L[u, n-1]$ is the min-weight path with at most $n-1$ edges, and $L[u, n] < L[u, n-1]$ (it cannot increase), then we have that P must have exactly n nodes $\implies$ some node repeats (by claim 1) (proves 1).

     The cycle must then have negative weight,



Since if P_2 has weight greater or equal to 0, then P' would be a path with the same min-weight but with $n-1$ edges, but that contradicts the fact that $L[u, n] < L[u, n-1]$. (proves 2)

We can then use this claim to find the negative-weighted cycle.

- Finding whether there are negative-weighted cycles unreachable from s

  Create s' and connect it with weight 0 to every other node, and perform algorithm starting from s' to find negative-weighted cycles unreachable from s. However, we have screwed up the shortest paths, because you can reach everywhere from s' with weight 0.