COL758: Advanced Algorithms **Spring 2019**

Lecture 13: February 21

Lecturer: Naveen Garg *Scribe: Anant Chhajwani*

13.1 Shortest Paths

In this lecture, we first revised combinatorial algorithms for shortest path problem on weighted directed graphs with negative edges, this included Bellman-Ford for single source shortest path (SSSP) and Floyd-Warshall for all pairs shortest path (APSP). Later we studied Seidel's algorithm, which is an algebraic algorithm to solve all pairs shortest path problem on unweighted, undirected graphs.

The table below shows standard algorithms with their running time for variants of shortest path problem:

	$l\colon E \to \mathbb{R}^+$	$l\colon E \to \mathbb{R}$
Single Source	Dijkstra $(m + n \log n)$	Bellman-Ford (mn)
All Pairs	$n\times$ Dijkstra $(mn + n^2 \log n)$	Floyd-Warshall (n^3)

13.2 Bellman-Ford (SSSP with negative edges)

We use an array $d^i[v]$ to store an upper bound on the distance of v from s after i^{th} round.

Input: A directed graph $G = (V, E)$, a length function $l\colon E \to \mathbb{R}$, and a source vertex $s \in V$.
Result: $\forall v \in V, d^{n-1}[v]$ stores distance of v from s
$d^0[s] \leftarrow 0$; $d^0[v] \leftarrow \infty$;
for $i \leftarrow 1$ **to** $n-1$ **do**
  for *all* $u \in V$ **do**
    $d^i[u] \leftarrow \min(d^{i-1}[u], \min_{(v,u)\in E}(d^{i-1}[v] + l(v,u))$;
  end
end

Algorithm 1: Bellman-Ford

Note: We assume there are no negative weight cycles in graph, as it can cause shortest path length to be $-\infty$ for some vertex v.

Invariant: $d^i[v]$ is the length of the shortest path from s to v having at most i edges.

Proof by Induction: Since $d^0[s] = 0$ and $d^0[v] = \infty$, the invariant holds for $i = 0$. To verify for $i + 1$, suppose the shortest $s - v$ path containing at most $i + 1$ edges has

1. **exactly** $i + 1$ **edges**, then there is a neighbour w of v that lies on $s - v$ path and has i edges on its $s - w$ path. Since the length of shortest $s - w$ path with at most i edges is given by $d^i[w]$, the length of shortest $s - v$ path with $i + 1$ edges is $d^{i+1}[v] = d^i[w] + l(w, v)$.

2. **less than** $i+1$ **edges**, then the length of shortest $s-v$ path will be same as previous iteration, hence $d^{i+1}[v] = d^i[v]$.

Since any shortest path can use at most $n-1$ edges, $d^{n-1}[v]$ stores the length of shortest $s-v$ path.

Running Time: Every iteration of inner for loop takes time equal to in-degree of u, therefore each iteration of outer for loop takes time equal to sum of in-degree over all vertices $u \in V$, which equals $O(m)$. Therefore, total time taken is $O(mn)$.

13.3 Floyd-Warshall (APSP with negative edges)

We use an array $d^i[u,v]$ to store an upper bound on the distance of v from u after i^{th} round.

Input: A directed graph $G=(V,E)$ and a length function $l\colon E \to \mathbb{R}$.
Result: $\forall (u,v) \in V \times V, d^n[u,v]$ stores distance of v from u
$d^0[u,u] \leftarrow 0$;
if $(u,v) \in E$ then $d^0[u,v] \leftarrow l(u,v)$, else $d^0[u,v] \leftarrow \infty$;
$i \leftarrow 0$;
for *all* $w \in V$ **do**
  $i \leftarrow i+1$;
  for *all* $(u,v) \in V \times V$ **do**
    $d^i[u,v] \leftarrow \min(d^{i-1}[u,v], d^{i-1}[u,w] + d^{i-1}[w,v])$;
  end
end

Algorithm 2: Floyd-Warshall

Let $w_1, w_2, \ldots, w_n$ be the vertices in the order of execution of outer for-loop in above algorithm.

Invariant: $d^i[u,v]$ is the length of the shortest path from u to v which is only allowed to use vertices $\{w_1, w_2, \ldots, w_i\}$ as internal vertices in a path.

Proof by Induction: For $i=0$, no vertex is allowed to be an internal vertex in a path, therefore the shortest path can only be an edge. Hence, $d^0[u,u]=0$, $d^0[u,v]=l(u,v)$ if $(u,v) \in E$, otherwise $d^0[u,v]=\infty$. To check for $i+1$, suppose the shortest path from u to v which is allowed to use only $w_1, w_2, \ldots, w_{i+1}$ as internal vertices

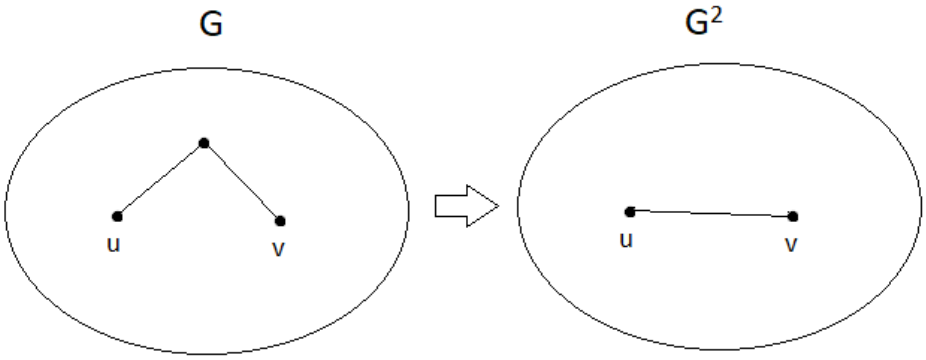
1. **uses w_{i+1} as internal vertex**, then consider two parts of this shortest $u-v$ path, one path is $u-w_{i+1}$ and other is $w_{i+1}-v$. Both these paths use only $w_1, \ldots, w_i$ as internal vertices, therefore length of these paths are $d^i[u,w]$ and $d^i[w,v]$. So, total length of shortest $u-v$ path is $d^{i+1}[u,v] = d^i[u,w_{i+1}] + d^i[w_{i+1},v]$.
2. **does not use w_{i+1} as internal vertex**, then it uses only $w_1, \ldots, w_i$ as internal vertices, therefore $d^{i+1}[u,v] = d^i[u,v]$.

Running Time: Since outer for loop performs n iterations and inner for loop performs n^2 iterations, total time taken is $O(n^3)$.

13.4 Seidel's Algorithm (APSP on undirected, unweighted graphs)

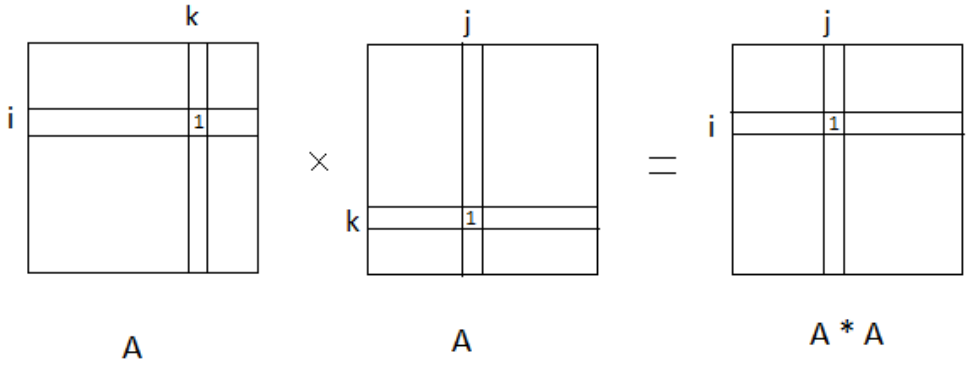
Seidel's algorithm is an algebraic algorithm to solve all pairs shortest path problem in undirected, unweighted graphs in time $O(n^\omega \log n)$, where $O(n^\omega)$ is time for multiply two $n \times n$ square matrices. The current best known value for ω is 2.373. Also, naive matrix multiplication has $\omega = 3$, and using Strassen's algorithm we can get $\omega = \log_2 7 = 2.81$.

Definition: The square of a graph $G = (V, E)$ is the graph $G^2 = (V, E^2)$ such that $(u, v) \in E^2$ if and only if there is a path of length ≤ 2 between u and v in graph G.



Lemma: The adjacency matrix of G^2 is $A * A + A$, where A is the adjacency matrix of G, $*$ is boolean AND operation and $+$ is boolean OR operation.

Proof: The adjacency matrix A ($n \times n$ square matrix) of a graph $G = (V, E)$ contains a 1 at (i, j) iff $(i, j) \in E$. Also, the (i, j) entry of $A * A$ contains a 1 iff there exists k such that $A[i, k] = 1$ and $A[k, j] = 1$. Therefore, $A * A$ has a 1 at (i, j) if there is a 2-length path between i and j. Thus, $A * A + A$ has a 1 at (i, j) if there is a 2-length or 1-length path between i and j.



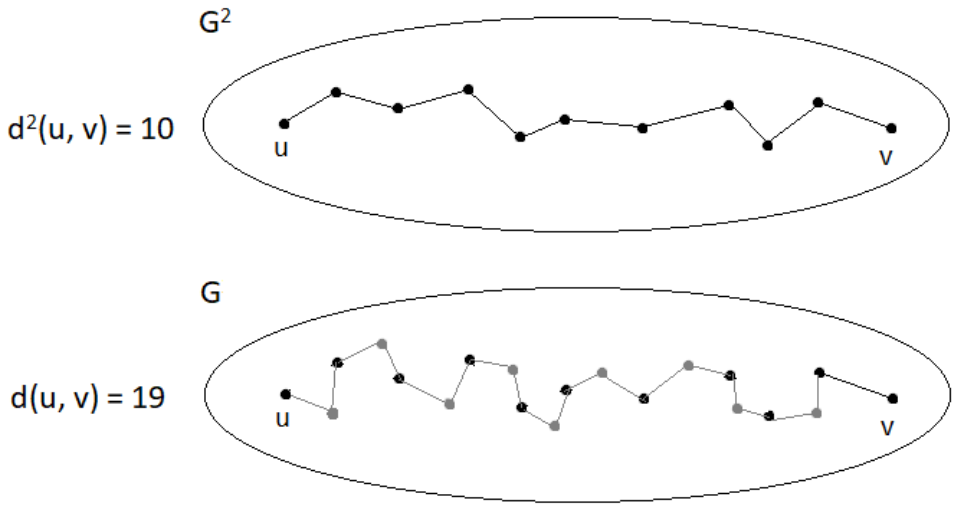
Question: Given $d^2(u, v)$ (length of shortest path between u and v in G^2) can we compute $d(u, v)$ (length of shortest path between u and v in G)?

Answer: Lets take an example, say $d^2(u, v) = 10$. Then we can say that $d(u, v) \leq 20$ because for each edge in G^2 there is a corresponding path of length ≤ 2 in G. Now, is it possible to have $d(u, v) = 18$? No, since this would imply $d^2(u, v) = 9$ because for every two consecutive edges in $u - v$ path of length 18 in G we have an edge in G^2. Thus, we conclude that $d(u, v) = 19$ or 20.

Generalizing this argument we get a relation between $d(u, v)$ and $d^2(u, v)$ as

$$d^2(u, v) = \left\lceil \frac{d(u, v)}{2} \right\rceil$$

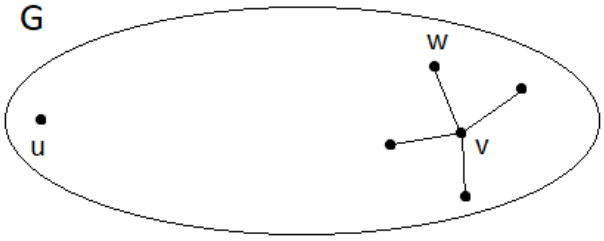
Therefore, $d(u, v) = 2d^2(u, v)$ or $2d^2(u, v) - 1$.



Question: How to decide whether $d(u,v)$ equals $2d^2(u,v)$ or $2d^2(u,v)-1$?

Answer: Let w be neighbor of v in graph G. Also, let us suppose we are given $d^2(u,v)=6$. Then we know that $d(u,v)=11$ or 12. Now, if $d^2(u,w)=5$ then we must have $d(u,v)=11$ since $d(u,w)\leq 10$ and $(w,v)\in E$. Also, if $d(u,v)=11$ then there exists a neighbor w of v in G such that $d^2(u,w)=5$, because in the shortest $u-v$ path of length 11 in G there is a neighbor w which is 10 units far from u, $d(u,w)=10$. Thus, we can generalize this statement in the following lemma.

Lemma 1: $d(u,v)=2d^2(u,v)-1$ iff there exists a neighbor w of v in G such that $d^2(u,w)=d^2(u,v)-1$.



Lemma 2: $d(u,v)=2d^2(u,v)-1$ iff $\sum_{(w,v)\in E} d^2(u,w) < d^2(u,v)\times \deg(v)$.

Proof: Lets take same example where $d^2(u,v)=6$ and $d(u,v)=11$. We also know a neighbor w of v in G having $d^2(u,w)=5$. Then we can not have $d^2(u,w')=7$ for some neighbor w' of v, because there is a length 2 path from w' to w passing through v and so we must have $d^2(u,w')\leq d^2(u,w)+1=6$. This proves lemma 2.

Let D^2 be a $n\times n$ matrix with entries $D^2[u,v]=d^2(u,v)$. Then, multiplying D^2 with adjacency matrix A of G we get a matrix with entries $D^2A[u,v]=\sum_{(w,v)\in E} d^2(u,w)$. Now, we can obtain matrix D with entries as $D[u,v]=2D^2[u,v]-\mathbb{1}[D^2A[u,v]<D^2[u,v]\times\deg(v)]$.

Notes:

1. This is a recursive algorithm since it computes $d(u,v)$, distances in G from $d^2(u,v)$, distance is G^2. We construct adjacency matrix A' of G^2 from adjacency matrix A of G using boolean matrix operations given by $A'=A*A+A$.

2. At each recursive step, we use the formula $D[u,v] = 2D^2[u,v] - \mathbb{1}[D^2A[u,v] < D^2[u,v] \times \deg(v)]$ which requires multiplying matrices D^2 and A in $O(n^\omega)$ time and then perform some matrix operations in $O(n^2)$ time. Thus, each recursive step takes $O(n^\omega)$ time.

3. The base case of our recursion is $D^{n-1}[u,v] = 1$, $u \neq v$ and $D^{n-1}[u,u] = 0$, since any shortest path is of length at most $n-1$. Since we square the graph at each recursive step, the recursion depth is $O(\log n)$ and total running time of Seidel's algorithm is $O(n^\omega \log n)$.