CS 341 – Algorithms

Lecture 14 – Dynamic Programming on Graphs

7,9 July 2021

Today’s Plan

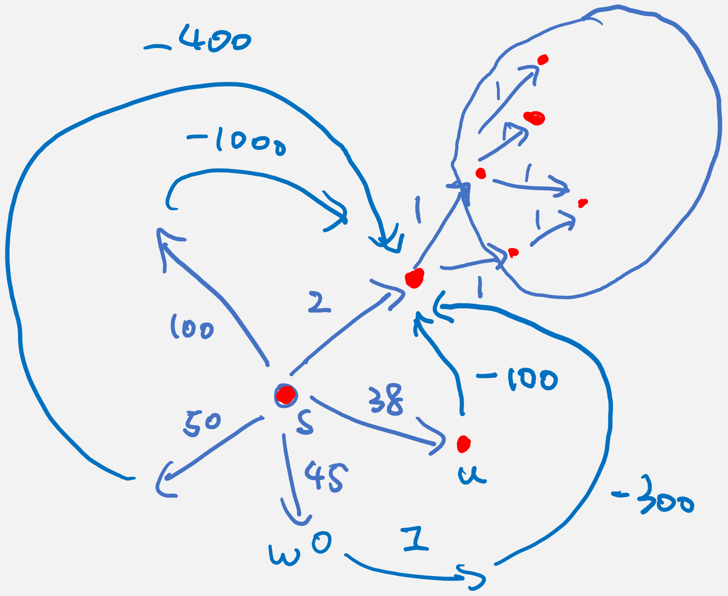
1. Shortest Paths with Negative Edges
2. Dynamic Programming and Bellman-Ford Algorithm
3. Negative Cycles
4. All-Pairs Shortest Paths and Floyd-Warshall Algorithm
5. Traveling Salesman Problem

Shortest Paths with Negative Edges

Input: A directed graph $G = (V, E)$, a (possibly ***negative***) length l_e on each edge $e \in E$, a vertex $s \in V$.

Output: The shortest path distance from s to every vertex $v \in V$.

What's wrong with Dijkstra's algorithm in this more general setting?



Negative Cycles

There could be negative cycles so that the shortest path distance is not well-defined.

We will study algorithms to solve the following two problems:

1. If G has no negative cycles, solve the single-source shortest paths problem.

2. Given a directed graph G, check if there is a negative cycle C, i.e. $\sum_{e \in C} l_e$.

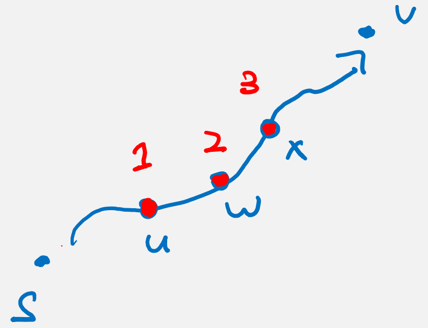
Today’s Plan

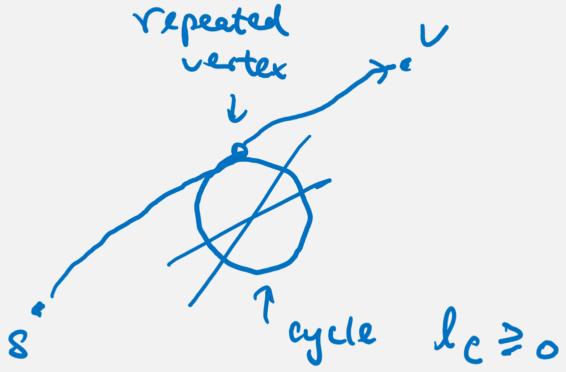
1. Shortest Paths with Negative Edges
2. Dynamic Programming and Bellman-Ford Algorithm
3. Negative Cycles
4. All-Pairs Shortest Paths and Floyd-Warshall Algorithm
5. Traveling Salesman Problem

Intuition

Although Dijkstra's algorithm may not compute all distances in one pass,

it will compute the distance to *some* vertices correctly, e.g. first vertex on a shortest path.





how many iterations ?

no negative cycles $\Rightarrow$ shortest path is a simple path

$\Rightarrow$ shortest path is of length $\leq n-1$

$\Rightarrow$ at most $n-1$ iterations

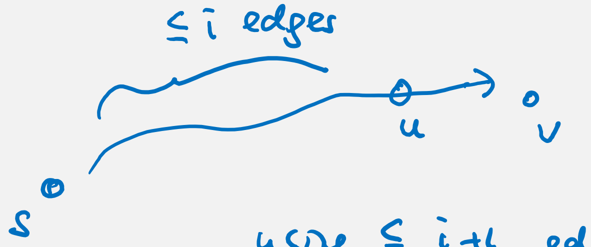
Dynamic Programming

Subproblems: Let $D(v, i)$ be the shortest path distance from s to v using at most i edges.

answer: $D(v, n-1)$ $\forall v$, because shortest paths are simple

base cases: $D(s,0) = 0$, $D(v,0) = \infty$ $\forall v \in V - s$.

recurrence: $D(v, i+1)$



using $\leq i+1$ edges

$uv \in E$ $\quad$ u in-neighbor of v

$$D(v, i+1) = \min \left\{ \begin{array}{l} D(v,i), \\ \min\limits_{u : uv \in E} \left\{ D(u,i) + l_{uv} \right\} \end{array} \right\}$$

Analysis

time complexity : computed $D(v,i)$ correctly $\forall v$

compute $D(w,i+1)$. time $O(\text{in-deg}(w))$

compute $D(w,i+1)\ \forall w$. time $O\left(\sum_w \text{in-deg}(w)\right) = O(m)$

compute up to $D(w,n-1)$

$\Rightarrow$ n iterations $\Rightarrow$ total time complexity $O(mn)$.

space complexity : $O(n^2)$

just compute distances $O(n)$

to compute $D(w,i+1)$, just need $D(v,i)$

Bellman-Ford Algorithm

The algorithm can made simpler, by using just one array instead of two.

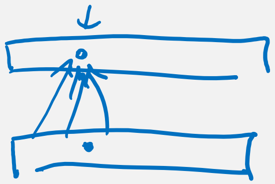
$dist[s] = 0, \quad dist[v] = \infty \quad \forall\, v \in V - s.$

for i from 1 to $n-1$ do

  for each edge $uv \in E$ do

    if $dist[u] + l_{uv} < dist[v]$ ← relaxation step

      $dist[v] = dist[u] + l_{uv}$ and $parent[v] = u$.



$D(v, i+1)$

$D(v, i)$

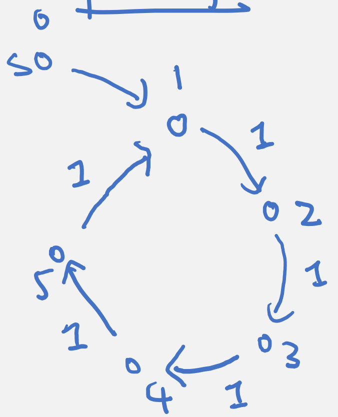
idea: keep a tighter upper bound on the shortest path distance

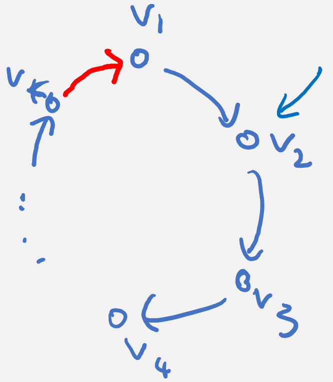
Shortest Path Tree

It is possible to have a cycle in the edges $(parent[v], v)$.

Lemma. If there a directed cycle C in the edges $(parent[v], v)$, then C must be a negative cycle.

proof.





$parent[v_i] = v_{i-1} \quad \forall\ 2 \le i \le k$

$d[v_i] \ge d[v_{i-1}] + l_{v_{i-1}v_i}. \quad \forall\ 2 \le i \le k$

$\not<$ otherwise, update the parent of v_i

$d[v_1] > d[v_k] + l_{v_k v_1}$ as we reset the parent of v_1

add all these inequalities

$$\sum_{j=1}^{k} d[v_j] > \sum_{j=1}^{k} d[v_j] + \sum_{e \in C} l_e \Rightarrow 0 > \sum_{e \in C} l_e \quad \square$$

Cor no negative cycle, shortest path tree. □

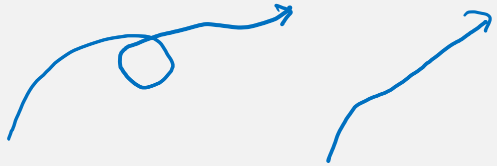
Today’s Plan

1. Shortest Paths with Negative Edges
2. Dynamic Programming and Bellman-Ford Algorithm
3. Negative Cycles
4. All-Pairs Shortest Paths and Floyd-Warshall Algorithm
5. Traveling Salesman Problem

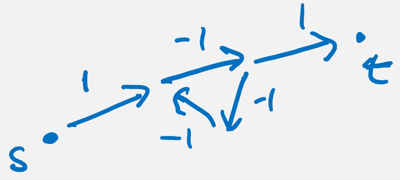
Ideas

Note that $D(v, i)$ is computed correctly even though the graph has negative cycles for any v and any $i \geq 0$.

used no negative cycles to conclude that we can stop at $D(v, n-1)$.



if $\exists$ negative cycle



$D(t, 3) = 1 \quad D(t, 6) = -2 \quad D(t, 9) = -5 \quad \ldots$

then $\exists\, v$ s.t $D(v, k) \to -\infty$ as $k \to \infty$

if $\nexists$ negative cycles, $D(v, n) = D(v, n-1) \quad \forall v$

$\Rightarrow D(v, k) = D(v, n-1) \quad \forall v \;\; \forall k \geq n-1$

$\Rightarrow D(v, k)$ finite $\forall v$ as $k \to \infty$

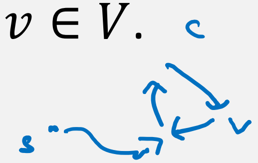
Assumption: Every vertex can be reached from vertex s.

This is without loss of generality for finding negative cycles, as the problem can be restricted to a SCC.

Observations

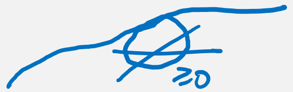
Claim 1. If the graph has a negative cycle, then $D(v,k) \to -\infty$ as $k \to \infty$ for some $v \in V$.

uses assumption that s can reach the negative cycle

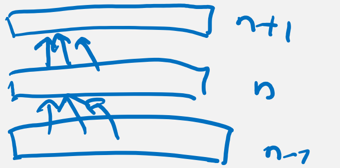


Claim 2. If the graph has no negative cycles, then $D(v,n) = D(v,n-1)$ for all $v \in V$.

shortest path must be simple



, optimal achieved at $D(v, n-1)$.



Claim 3. If $D(v,n) = D(v,n-1)$ for all $v \in V$, then the graph has no negative cycles.

proof

$$D(v,n+1) = \min\left\{ D(v,n),\ \min_{u:uv\in E} \{ D(u,n) + l_{uv} \} \right\}$$

$$= \min\left\{ D(v,n-1),\ \min_{u:uv\in E} \{ D(u,n-1) + l_{uv} \} \right\} \quad \text{by assumption}$$

$$= D(v,n)$$

by induction $\Rightarrow$ $D(v,k) = D(v,n-1)$ $\forall v$ $\forall k \geq n-1$

$\Rightarrow$ $D(v,k)$ finite $\forall v$ $\forall k \geq n-1$

$\overset{\text{Claim 1}}{\Rightarrow}$ no negative cycles. $\square$

Remark: Early termination rule is $D(v,k+1) = D(v,k)$ for all $v \in V$.

Algorithms

Checking: Claim 2+3 says that no negative cycles $\Leftrightarrow$ $D(v,n) = D(v,n-1)\ \forall v$

$D(v,n) < D(v,n-1)$ for some v

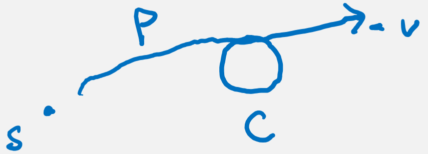
Finding: It would be easier to explain using the $\Theta(n^2)$ space dynamic programming algorithm.

compute $D(v,i)\ \forall v,\ \forall 1 \le i \le n$, $parent(v,i) = u$ if $D(v,i) = D(u,i-1) + l_{uv}$

now, if $D(v,n) < D(v,n-1)$,

then we know that shortest path using at most n edges to get to v must have exactly n edges, otherwise $D(v,n) = D(v,n-1)$.

$\Rightarrow$



path must have repeated vertice

$\Rightarrow \exists$ cycle C in the path

<u>Claim</u> C must be negative

$D(v,n-1) \le length(P') \le length(P) = D(v,n)$, contradiction

$\Rightarrow$ by tracing out P using parent information, we can find C. $\square$

<u>Problem</u>: Bellman-Ford

Today’s Plan

1. Shortest Paths with Negative Edges
2. Dynamic Programming and Bellman-Ford Algorithm
3. Negative Cycles
4. All-Pairs Shortest Paths and Floyd-Warshall Algorithm
5. Traveling Salesman Problem

All-Pairs Shortest Paths

Input: A directed graph $G = (V, E)$, a (possibly *negative*) length l_e on each edge $e \in E$.

Output: The shortest path distance from s to t for all $s, t \in V$.

apply Bellman-Ford for all s, time $O(nm \cdot n) = O(n^2 m)$

$\Omega(n^4)$ if $m = \Omega(n^2)$

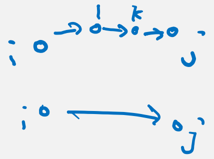
Floyd-Warshall $O(n^3)$

more subproblems : $D(u, v, i)$

Dynamic Programming

vertex set $\{1, 2, \ldots, n\}$

<u>Subproblems</u>: $D(i,j,k)$ is the shortest path distance from i to j using $\{1,\ldots,k\}$ as intermediate vertices.
^possible



<u>answers</u>: $D(i,j,n)$ $\forall\, i,j$

<u>base cases</u>: $D(i,j,0) = \ell_{ij}$ $\forall\, ij \in E$ $\qquad D(i,j,0) = \infty$ $\forall\, ij \notin E$.

<u>recurrence</u>: computed $D(i,j,k)$ $\forall\, i,j$

want to compute $D(i,j,k+1)$

optimal way

$$D(i,j,k+1) = \min\left\{ \begin{array}{l} D(i,j,k), \\ D(i,k+1,k) + D(k+1,j,k) \end{array} \right\}$$

use k+1 once, $\boxed{\text{because there are no negative cycles}}$

Floyd-Warshall Algorithm

$D(i,j,0) = \infty \quad \forall ij \notin E. \qquad D(i,j,0) = \ell_{ij} \quad \forall ij \in E.$ // base cases

for k from 1 to n do

  for i from 1 to n do

    for j from 1 to n do

$$D(i,j,k+1) = \min\{\underbrace{D(i,j,k)}_{\text{don't use } k+1},\ \underbrace{D(i,k+1,k) + D(k+1,j,k)}_{\text{use } k+1}\}.$$

Time Complexity: $\Theta(n^3)$

Open Problem: Is there an $O(n^{3-\epsilon})$ algorithm for all-pairs shortest paths? e.g. $O(n^{2.9999})$

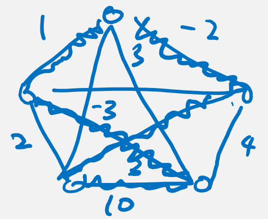
Today’s Plan

1. Shortest Paths with Negative Edges
2. Dynamic Programming and Bellman-Ford Algorithm
3. Negative Cycles
4. All-Pairs Shortest Paths and Floyd-Warshall Algorithm
5. Traveling Salesman Problem

Traveling Salesman Problem

Input: A directed graph $G = (V, E)$, a (possibly *negative*) length l_{ij} for all $i, j \in V$.

Output: A directed cycle C that visits every vertex exactly once that minimizes $\sum_{e \in C} l_e$.



It is one of the most famous problems in combinatorial optimization.

NP-complete

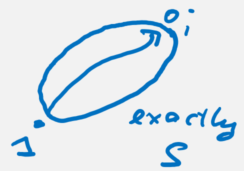
naive $O(n! \cdot n)$ impractical $n \approx 13$

DP $O(2^n \cdot n^2)$ $n \approx 30$

remember which nodes that visited.

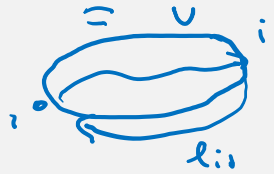
Dynamic Programming

Start from vertex 1



Subproblems: $C(i, S)$ be the shortest path distance from $\underline{1}$ to i with vertices in S on the path.

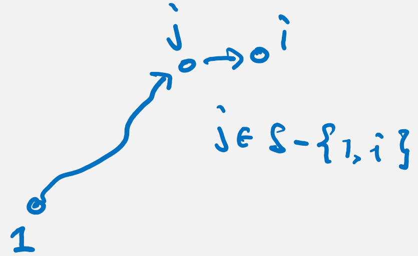
answer: $\min\limits_{1 \le i \le n} \{ C(i, V) + \ell_{i1} \}$



base cases: $C(i, \{1, i\}) = \ell_{1i} \quad \forall i$

computed $C(i, S) \quad \forall\, |S| \le k$.

want to compute $C(i, S)$ for $|S| = k+1$



idea: try all possible second last vertex of the path

$$C(i, S) = \min_{j \in S - \{1, i\}} \{ C(j, S - \{i\}) + \ell_{ji} \}$$

Analysis

Time: $O(2^n \cdot n)$ subproblems

each subproblem $O(n)$ time

total $O(2^n \cdot n^2)$

Space: $\Theta(2^n \cdot n)$

best: $O(2^n \cdot n)$ time $O(\text{poly}(n))$ space

Concluding Remarks

We have seen many examples and structures to design dynamic programming algorithms,

from lines to trees to graphs.

I hope that you will be familiar with this technique, and be able to solve new problems with ease!