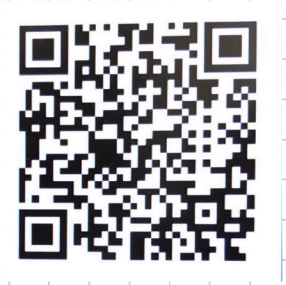
CS 170

Lecture 12

- use iClicker

Sanjam Garg.

join.iclicker.com/RGWR



DP Approach

1) Define appropriate subproblems
2) Write a recurrence relation
3) Determine order of computation.
   ↳ induces a DAG structure

Shortest paths in a Graph

→ edges with +ve weights

→ edges with -ve weights
- DAG
- arbitrary connections but no -ve cycles.

Dijkstra	$O((n+m)\log n)$	+ve
Bellman-Ford	$O(nm)$	-ve
DAG-SSSP (DP)	$O(n+m)$	-ve

Today: SSSP (Single Source Shortest Path)

Input: -Weighted graph $G=(V,E)$ weights w_e for edge e $(w_e \in \mathbb{R})$

-Source s

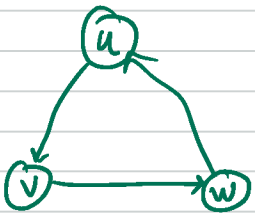
Output: For every v, dist(v) = shortest path from s to v $s \rightsquigarrow v$

Step ① dist(u) = shortest path from $s \rightsquigarrow u$

Step ② $$dist(u) = \min_{(u,v)\in E} \{dist(s,u) + w_{u,v}\} \quad \forall v \neq s$$

$$dist(s) = 0 \quad \text{(base case)}$$

Step ③



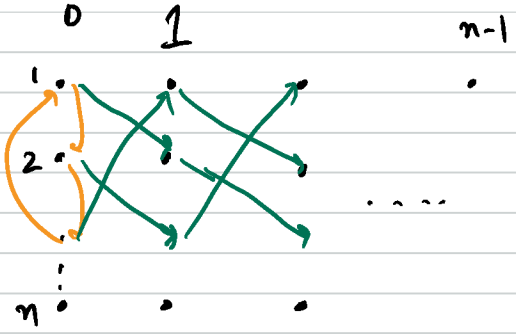
Step 1: $dist(v, k)$ = shortest distance from $s \rightsquigarrow v$ taking at most k edges.

Step 2:

Base Case: $dist(s, 0) = 0$; $dist(v, 0) = \infty$ $\forall v \neq s$

$$dist(v, k) \rightarrow \min \begin{cases} dist(v, k-1) \\ \min_{(u,v) \in E} dist(u, k-1) + w_{u,v} \end{cases}$$

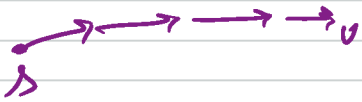
Step 3: $\forall v$ $dist(v, 0)$; $\forall v$ $dist(v, 1)$ . . . . $\forall v$ $dist(v, n-1)$



Algorithm

```
O(n)  ← For v ∈ V   dist(v, 0) = ∞
        dist(s, 0) = 0
O(m)  ← For all k = 1 .... n-1
O(n)  ←   For all v ∈ V
            dist(v, k) = dist(v, k-1)
O(m)  ←   For all (u, v) ∈ E
            if dist(v, k) > dist(u, k-1) + w_{u,v}
            then dist(v, k) = dist(u, k-1) + w_{u,v}
Output ∀ v ∈ V  dist(v, n-1)
```

$dist_{BF}(v)$ → $dist_{DP}(v, 0)$, $dist_{DP}(v, k-1)$



$O(n(n+m))$

↓

$O(nm)$ if the graph is connected.

Single Source Reliable Shortest Path.

Input: -Weighted graph $G=(V,E)$ weights w_e for edge e $(w_e \in R)$
-Source s and a bound B

Output: For every v, $dist(v)$ = shortest path from s to v $s \rightsquigarrow v$
taking atmost B edges

$$dist(v, B)$$

All pairs shortest path

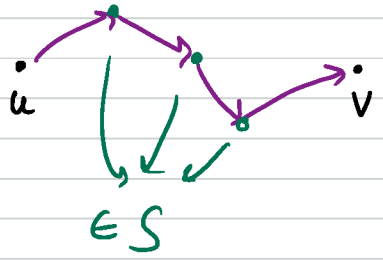
Input: -Weighted graph $G=(V,E)$ weights w_e for edge e $(w_e \in R)$
~~-Source s~~

Output: For every u,v, $dist(u,v)$ = shortest path from u to v $u \rightsquigarrow v$

* Run BF (or DP variant) for u
↳ $O(n \times nm) = O(n^2 m)$ ⟶ $O(n^4)$

DP solution direct
*
$dist(u, v, h)$ = shortest path from $u \rightsquigarrow v$ taking atmost h edges.
↳ $O(n^2 m)$

Step①: $dist(u, v, k)$ = shortest path from $u \rightsquigarrow v$ that only takes vertices in $\{1, \dots, k\}$



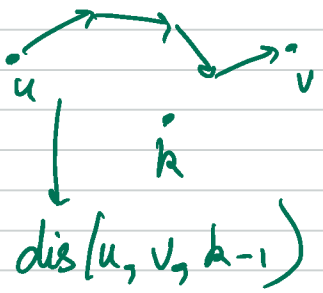
Step②: Base $dist(u, v, 0) = w_{uv}$

Property: On the shortest path from $u \rightsquigarrow v$ no vertex w happens twice.

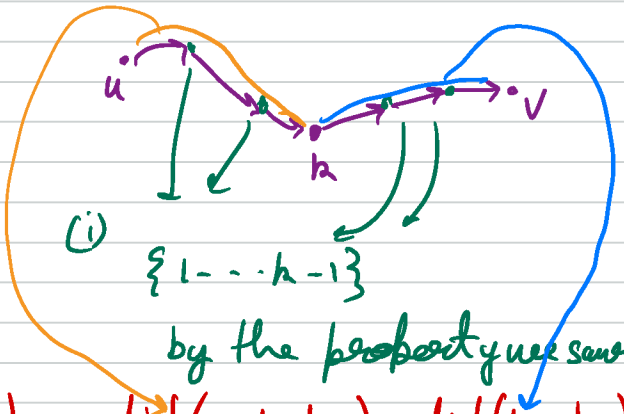
Proof: Assume w is visited twice.

$dist(u, v, k)$ → $\{1 \dots k\}$

Case I:



Case II:

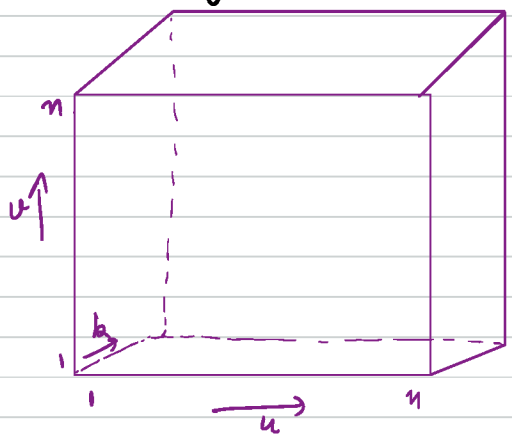


by the property we saw

$$dist(u, v, k) = \min \{ \underbrace{dist(u, v, k-1)}_{\text{Case I}}, \underbrace{dist(u, k, k-1) + dist(k, v, k-1)}_{\text{Case II}} \}$$

③ Order of computation.

$d(u, v, k)$ only depends on $d(\cdot, \cdot, k-1)$



$O(n^3)$

Algorithm (Floyd-Warshall)

For $i, j = 1 \dots n$ $\quad dist(i, j, 0) = \infty$

For $(i, j) \in E$ $\quad dist(i, j, 0) = w_{ij}$

For $i = 1 \dots n$ $\quad dist(i, i, 0) = 0$

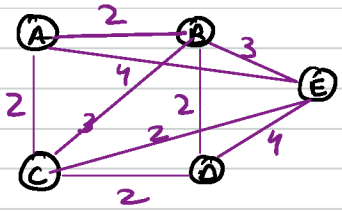
For $k = 1 \dots n$
  For $i = 1 \dots n$
    For $j = 1 \dots n$

$$dist(i, j, k) = \min\{ dist(i, j, k-1),\ dist(i, k, k-1) + dist(k, j, k-1) \}$$

Output $dist(i, j, n)$ for all $i, j \in V$

Travelling Salesman Problem.

Input: n cities & distances d_{ij} $(i \neq j)$
Goal: Find minimum distance path from city 1 back to city 1 visiting each city exactly once.



Brute-force: (i) $1 \to 2 \to 3 \cdots \to n$
(ii) $1 \to 3 \to 2 \cdots \to n$
$\vdots$
$n! \to n^n$ (costly)
↳ $1 \times \cdots \times \frac{n}{2} \cdots \times n$
$> \left(\frac{n}{2}\right)^{\frac{n}{2}}$

$2^n << n^{\frac{n}{2}}$

$C(j)$

$\min_j \{ C(j) + w_{j1} \}$

$C(j)$ = cost of the min path from $1 \to j$ that visits all nodes exactly once

$C(j, k)$ = cost of the min path from $1 \to j$ that visits all nodes $\{1 \cdots k\}$ exactly once