W, 10/02/19

Fall'19 CSCE 629

# Analysis of Algorithms

Fang Song
Texas A&M U

## Lecture 13

- Bellman-Ford algorithm: Shortest path with negative weight

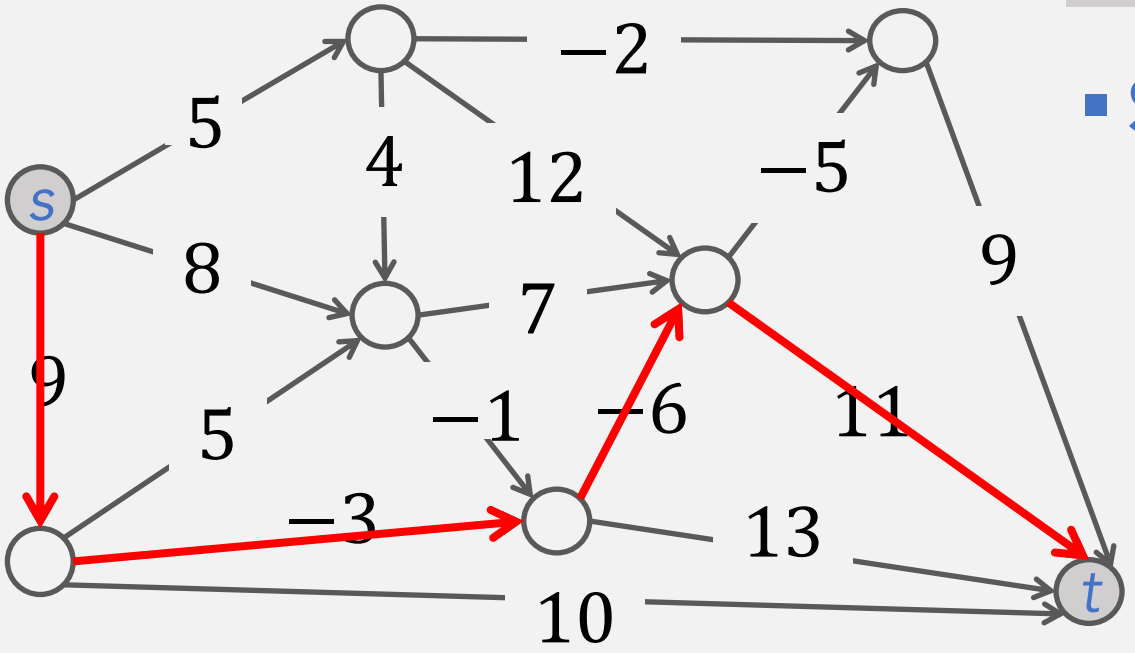
Credit: based on slides by A.Smith and K.Wayne

# Logistics

- Write up your algorithm
  - Start with "Idea": short description of the key to your algorithm. [Your peer should be able to take it and come up with an algorithm with *a bit*, if any, thought]
  - Pseudocode: write in a nice format as you'd do in a real program, even if you are using plain-English descriptions [Your peer should be able to implement your algorithm without thinking other than implementation details]
  - Comments in your pseudocode are helpful
- When studying
  - Identify what you don't understand
  - Test yourself via (a) rederiving algorithms, proofs from class and (b) exercises in book.
  - Read alternative explanations (e.g. KT,E,DPV)

# Recall: shortest path problem

- Input. graph $G$, node $s$ and $t$
- Output. $dist(s,t)$

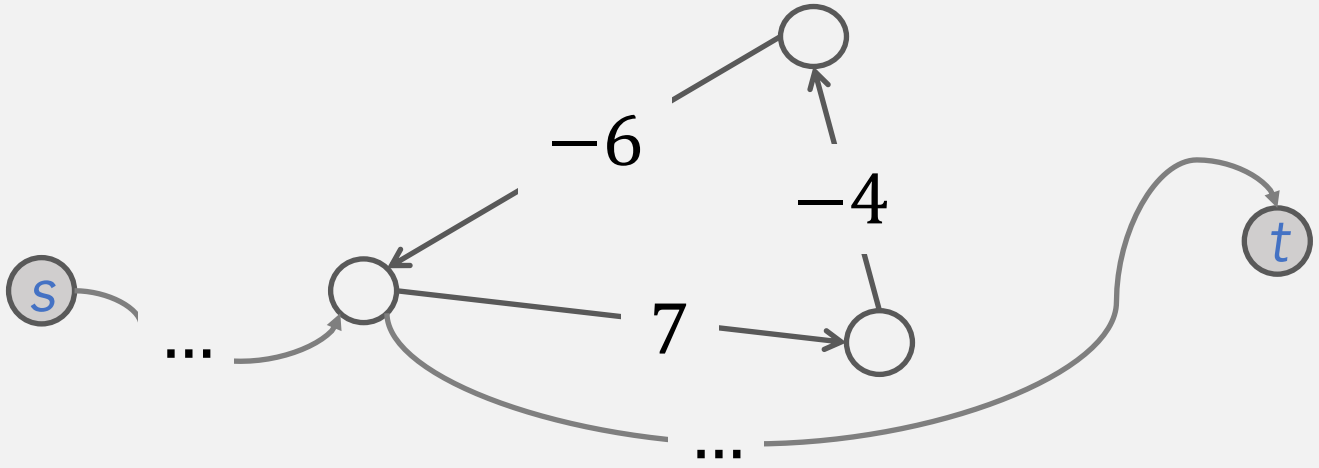



- Every edge has a length $l_e$ (can be negative)
- Length of a path $l(P) = \sum_{e \in P} l_e$
- Distance $dist(u,v) = \min_{P:u \rightsquigarrow v} l(P)$

- Special cases
  - All edge of equal length: BFS $O(m+n)$
  - DAG: DP in topological order $O(m+n)$

Length of shortest path: $dist(s,t) = 9 - 3 - 6 + 11 = 11$

# A technical issue: negative length cycles




- Observation.
  - If some $s \rightsquigarrow t$ path contains a negative length cycle, there does not exist a shortest $s \rightsquigarrow t$ path;
  - Otherwise there exists a simple (i.e., no repetition node) path $\leq n - 1$ edges
- For simplicity: assuming $G$ has no NegativeLengthCycle
  - can be detected with little overhead

# DP1: develop a recursion

Def. for all $i = 0, \ldots, n-1, v \in V$

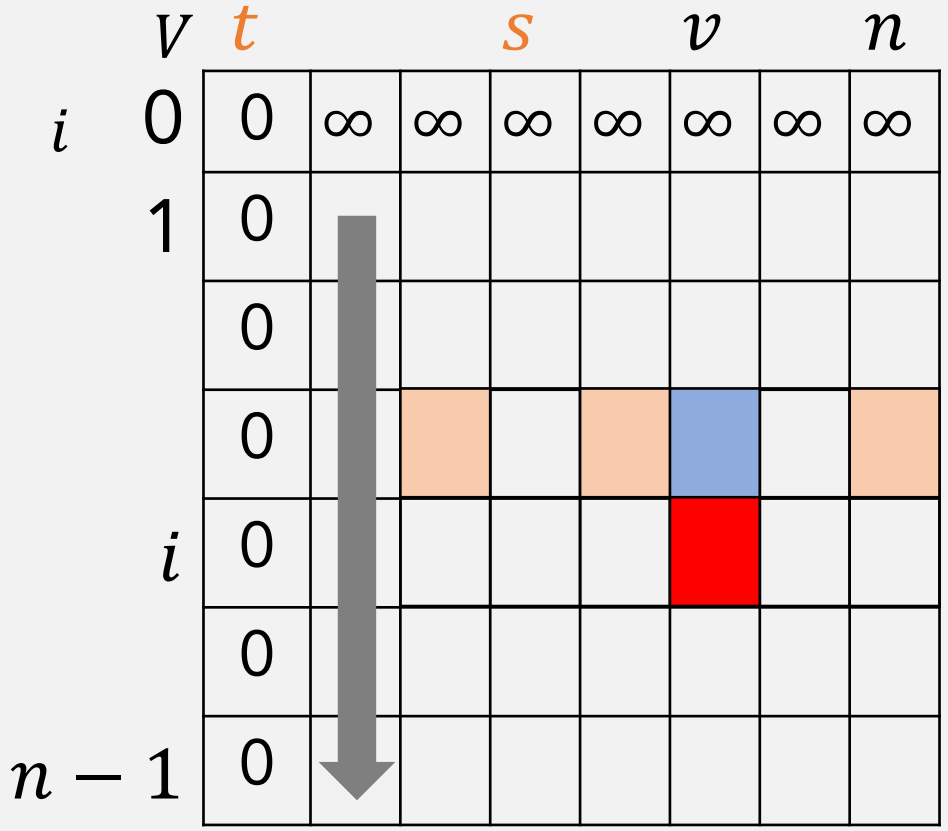
$\mathrm{OPT}(i, v) :=$ length of shortest $v \rightsquigarrow t$ path P using $\leq i$ edges

- Case 1. P uses at most $i - 1$ edges
  - $\mathrm{OPT}(i, v) := \mathrm{OPT}(i-1, v)$
- Case 2. P uses exactly $i$ edges
  - If $(v, w)$ the first edge, then OPT uses $(v, w)$ and then selects best $w \rightsquigarrow t$ path using $\leq i - 1$ edges

- Goal. Find $\mathrm{OPT}(n-1, s)$
- Basis: $\mathrm{OPT}(i, v) = 0$ or $\infty$ if $i = 0$
- Recursion: how to define $\mathrm{OPT}(i, v)$ recursively?

$$\mathrm{OPT}(i, v) = \begin{cases} 0 \text{ if } v = t;\ \infty \text{ if } i = 0 \\ \min\left\{\mathrm{OPT}(i-1, v),\ \min_{v \to w \in E}\{\mathrm{OPT}(i-1, w) + l_{v \to w}\}\right\} \text{ otherwise} \end{cases}$$

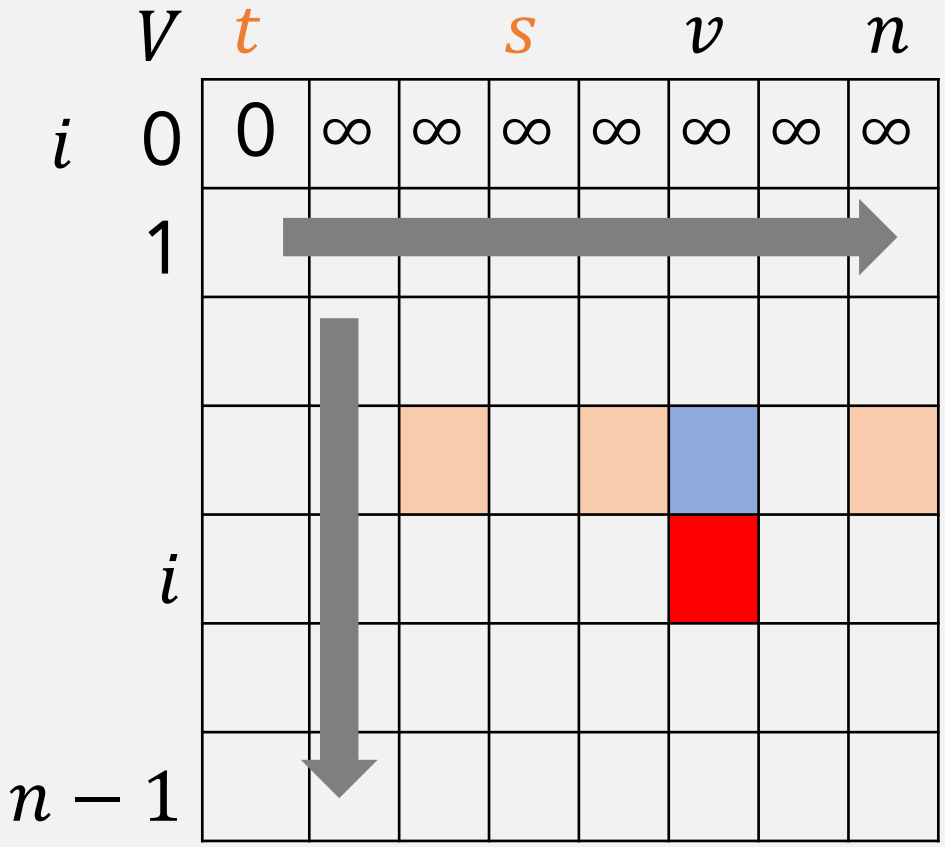
# DP2: build up solutions




- Subproblems: $O(n^2)$
- Memoization data struture
  - 2-D array $M[0, \dots n-1, v_1, \dots, v_n]$
- Dependencies
  - Each $\mathrm{OPT}(i, v)$ depends on subproblems on the row above
- Evaluation order
  - Row by row, each row arbitrary order

$$\mathrm{OPT}(i,v) = \begin{cases} 0 \; if \; v = t; \; \infty \text{ if } i = 0 \\ \min\left\{\mathrm{OPT}(i-1,v), \min_{v \to w \in E}\{\mathrm{OPT}(i-1,w) + l_{v \to w}\}\right\} \text{otherwise} \end{cases}$$

# DP2: build up solutions




***SPLen(G, t)***

**// $M[i, v]$** memoize subproblem values

// $M[0, t] = 0$ and $M[0, v] = \infty$ otherwise

For $i = 1, \ldots, n-1$ // row by row

  For $v \in V$ // any order

    $M[i, v] \leftarrow M[i-1, v]$ // case 1

    For edge $(v \to w) \in E$ // case 2

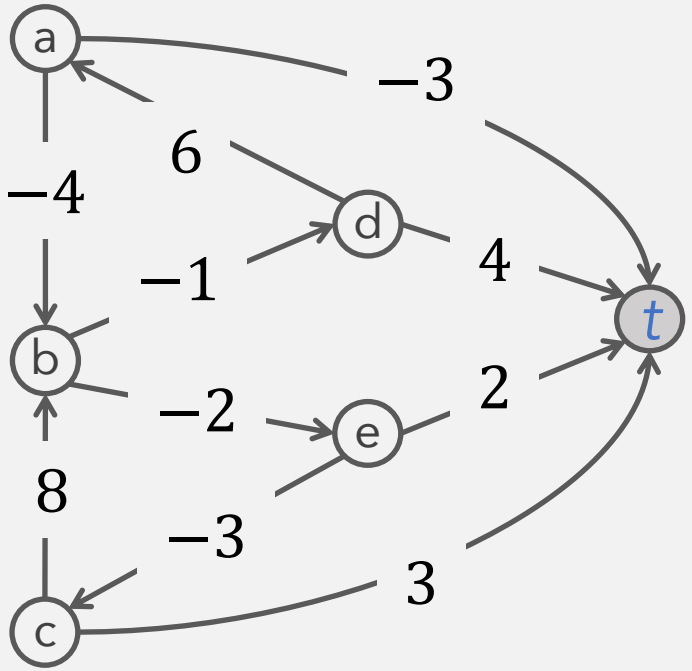
      $M[i, v]$

      $\leftarrow \min\{M[i, v], M[i-1, w] + l_{vw}\}$

This actually gives us $dist(v, t)$ for all $v \in V$

- Analysis: $O(n^2)$ space; $O(mn)$ time [visit all edges for each $i$]
- Finding a shortest path: maintain a "successor" for each entry

# Example




| $i$ | $v$: $t$ | $a$ | $b$ | $c$ | $d$ | $e$ |
|---|---|---|---|---|---|---|
| 0 | 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 1 | 0 | −3 | ∞ | 3 | | |
| 2 | 0 | | | | | |
| 3 | 0 | | | | | |
| 4 | 0 | | | | | |
| 5 | 0 | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 0 | −3 | ∞ | 3 | 4 | 2 |
| 0 | −3 | 0 | 3 | 3 | 0 |
| 0 | −4 | −2 | 3 | 3 | 0 |
| 0 | −6 | −2 | 3 | 2 | 0 |
| 0 | −6 | −2 | 3 | 0 | 0 |

For $i = 1, \ldots, n-1$ // row by row
  For $v \in V$ // any order
    $M[i, v] \leftarrow M[i-1, v]$ // case 1
    For edge $(v \rightarrow w) \in E$ // case 2
      $M[i, v] \leftarrow \min\{M[i, v], M[i-1, w] + l_{vw}\}$

# A simple but impactful improvement

Maintain only one array $M[v]$ = length of shortest $v \rightsquigarrow t$ path found so far.
No need to check edge $(v, w)$ unless $M[w]$ changed in previous iteration.

Theorem. Throughout the algorithm, $M[v]$ is length of some $v \rightsquigarrow t$ path, and after $i$ rounds of updates, the value $M[v]$ is no larger than the length of shortest $v \rightsquigarrow t$ path using $\leq i$ edges.

- Memory: $O(m + n)$.
- Running time: $O(mn)$ worst case, but faster in practice.

➔ Bellman-Ford algorithm: efficient implementation

- Application: (distance-vector) routing protocol on Internet [more to come]

# Single-source shortest paths with negative weights

| Year | Worst case | Discovered by |
|---|---|---|
| 1955 | $O(n^4)$ | Shimbel |
| 1956 | $O(mn^2W)$ | Ford |
| 1958 | $O(mn)$ | Bellman, Moore |
| 1983 | $O(n^{3/4}m\log W)$ | Gabow |
| 1989 | $O(mn^{1/2}\log(nW))$ | Gabow-Tarjan |
| 1993 | $O(mn^{1/2}\log W)$ | Goldberg |
| 2005 | $O(n^{2.38}W)$ | Sankowsi, Yuster-Zwich |
| 2016 | $O(n^{10/7}\log W)$ | Cohen-Madry-Sankowski-Vladu |
| 20XX | ??? | You??? |

weights between $[-W, W]$