Algorithm Design and Implementation

Principle of Algorithms IX

Dynamic Programming II

Guoqiang Li

School of Software, Shanghai Jiao Tong University

Sequence Alignment

String similarity

Q. How similar are two strings?

Example. **ocurrance** and **occurrence**.

o	c	u	r	r	a	n	c	e	–
o	c	c	u	r	r	e	n	c	e

6 mismatches, 1 gap

o	c	–	u	r	r	a	n	c	e
o	c	c	u	r	r	e	n	c	e

1 mismatch, 1 gap

o	c	–	u	r	r	–	a	n	c	e
o	c	c	u	r	r	e	–	n	c	e

0 mismatches, 3 gaps

Edit distance

Edit distance. [Levenshtein 1966, Needleman-Wunsch 1970]

- Gap penalty δ; mismatch penalty α_{pq}.
- Cost = sum of gap and mismatch penalties.

C	T	-	G	A	C	C	T	A	C	G
C	T	G	G	A	C	G	A	A	C	G

cost=$\delta + \alpha_{CG} + \alpha_{TA}$

Applications. Bioinformatics, spell correction, machine translation, speech recognition, information extraction, ...

Example.

Spokesperson	*confirms*		*senior*	*government*	*adviser*	*was*	*found*
Spokesperson	*said*	*the*	*senior*		*adviser*	*was*	*found*

BLOSUM matrix for proteins

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	7	-3	-3	-3	-1	-2	-2	0	-3	-3	-3	-1	-2	-4	-1	2	0	-5	-4	-1
R	-3	9	-1	-3	-6	1	-1	-4	0	-5	-4	3	-3	-5	-3	-2	-2	-5	-4	-4
N	-3	-1	9	2	-5	0	-1	-1	1	-6	-6	0	-4	-6	-4	1	0	-7	-4	-5
D	-3	-3	2	10	-7	-1	2	-3	-2	-7	-7	-2	-6	-6	-3	-1	-2	-8	-6	-6
C	-1	-6	-5	-7	13	-5	-7	-6	-7	-2	-3	-6	-3	-4	-6	-2	-2	-5	-5	-2
Q	-2	1	0	-1	-5	9	3	-4	1	-5	-4	2	-1	-5	-3	-1	-1	-4	-3	-4
E	-2	-1	-1	2	-7	3	8	-4	0	-6	-6	1	-4	-6	-2	-1	-2	-6	-5	-4
G	0	-4	-1	-3	-6	-4	-4	9	-4	-7	-7	-3	-5	-6	-5	-1	-3	-6	-6	-6
H	-3	0	1	-2	-7	1	0	-4	12	-6	-5	-1	-4	-2	-4	-2	-3	-4	3	-5
I	-3	-5	-6	-7	-2	-5	-6	-7	-6	7	2	-5	2	-1	-5	-4	-2	-5	-3	4
L	-3	-4	-6	-7	-3	-4	-6	-7	-5	2	6	-4	3	0	-5	-4	-3	-4	-2	1
K	-1	3	0	-2	-6	2	1	-3	-1	-5	-4	8	-3	-5	-2	-1	-1	-6	-4	-4
M	-2	-3	-4	-6	-3	-1	-4	-5	-4	2	3	-3	9	0	-4	-3	-1	-3	-3	1
F	-4	-5	-6	-6	-4	-5	-6	-6	-2	-1	0	-5	0	10	-6	-4	-4	0	4	-2
P	-1	-3	-4	-3	-6	-3	-2	-5	-4	-5	-5	-2	-4	-6	12	-2	-3	-7	-6	-4
S	2	-2	1	-1	-2	-1	-1	-1	-2	-4	-4	-1	-3	-4	-2	7	2	-6	-3	-3
T	0	-2	0	-2	-2	-1	-2	-3	-3	-2	-3	-1	-1	-4	-3	2	8	-5	-3	0
W	-5	-5	-7	-8	-5	-4	-6	-6	-4	-5	-4	-6	-3	0	-7	-6	-5	16	3	-5
Y	-4	-4	-4	-6	-5	-3	-5	-6	3	-3	-2	-4	-3	4	-6	-3	-3	3	11	-3
V	-1	-4	-5	-6	-2	-4	-4	-6	-5	4	1	-4	1	-2	-4	-3	0	-5	-3	7

Quiz 1

What is edit distance between these two strings?

P A L E T T E P A L A T E

Assume gap penalty = 2 and mismatch penalty = 1.

A. 1
B. 2
C. 3
D. 4
E. 5

Merging

Goal. Given two strings $x_1 x_2 \ldots x_m$ and $y_1 y_2 \ldots y_n$, find a min-cost alignment.

Definition. An alignment M is a set of ordered pairs $x_i - y_j$ such that each character appears in at most one pair and no crossings.

Definition The cost of an alignment M is:

$$\text{cost}(M) = \underbrace{\sum_{(x_i, y_j) \in M} \alpha_{x_i y_j}}_{\text{mismatch}} + \underbrace{\sum_{i: x_i \text{ unmached}} \delta + \sum_{j: y_j \text{ unmatched}} \delta}_{\text{gap}}$$

x_1	x_2	x_3	x_4	x_5		x_6
C	T	A	C	C	-	G
-	T	A	C	A	T	G
	y_1	y_2	y_3	y_4	y_5	y_6

an alignment of CTACCG and TACATG

$M = \{x_2 - y_1, x_3 - y_2, x_4 - y_3, x_5 - y_4, x_6 - y_6\}$

Sequence alignment: problem structure

Def. $OPT(i, j)$: min cost of aligning prefix strings $x_1 x_2 \ldots x_i$ and $y_1 y_2 \ldots y_j$.
Goal. $OPT(m, n)$.

Case 1. $OPT(i, j)$ matches $x_i - y_j$.
Pay mismatch for $x_i - y_j$ + min cost of aligning $x_1 x_2 \ldots x_{i-1}$ and $y_1 y_2 ... y_{j-1}$.

Case 2a. $OPT(i, j)$ leaves x_i unmatched.
Pay gap for x_i + min cost of aligning $x_1 x_2 ... x_{i-1}$ and $y_1 y_2 \ldots y_j$.

Case 2b. $OPT(i, j)$ leaves y_j unmatched.
Pay gap for y_j+ min cost of aligning $x_1 x_2 ... x_i$ and $y_1 y_2 \ldots y_{j-1}$.

Bellman equation.

$$OPT(i,j) = \begin{cases} j\delta & \text{if i = 0} \\ i\delta & \text{if j = 0} \\ \min \begin{cases} \alpha_{x_i y_j} + OPT(i-1, j-1) \\ \delta + OPT(i-1, j) \\ \delta + OPT(i, j-1) \end{cases} & \text{otherwise} \end{cases}$$

Sequence alignment: bottom-up algorithm

SequenceAlignment($m, n, x_1, \ldots, x_m, y_1, \ldots, y_n, \delta, \alpha$)

for $i = 0$ *to* m **do**
    $M[i, 0] \leftarrow i\delta;$
end
for $j = 0$ *to* n **do**
    $M[0, j] \leftarrow j\delta;$
end
for $i = 1$ *to* m **do**
    for $j = 1$ *to* n **do**
        $M[i, j] \leftarrow$
        $\min\{\alpha_{x_i y_j} + M[i-1, j-1], \delta + M[i-1, j], \delta + M[i, j-1]\};$
    end
end
Return $M[m, n]$;

Sequence alignment: traceback

		S	I	M	I	L	A	R	I	T	Y
	0	2	4	6	8	10	12	14	16	18	20
I	2	4	1	3	2	4	6	8	7	9	11
D	4	6	3	3	4	4	6	8	9	9	11
E	6	8	5	5	6	6	6	8	10	11	11
N	8	10	7	7	8	8	8	8	10	12	13
T	10	12	9	9	9	10	10	10	10	9	11
I	12	14	8	10	8	10	12	12	9	11	11
T	14	16	10	10	10	10	12	14	11	8	11
Y	16	18	12	12	12	12	12	14	13	10	7

Theorem

The DP algorithm computes the edit distance (and an optimal alignment) of two strings of lengths m and n in $\Theta(mn)$ time and space.

Proof.

- Algorithm computes edit distance.
- Can trace back to extract optimal alignment itself.

Theorem (Backurs-Indyk 2015)

If can compute edit distance of two strings of length n in $O(n^{2-\varepsilon})$ time for some constant $\varepsilon > 0$, then can solve SAT with n variables and m clauses in $poly(m)2^{(1-\delta)n}$ time for some constant $\delta > 0$.

Quiz 3

It is easy to modify the DP algorithm for edit distance to ...

A. Compute edit distance in $O(mn)$ time and $O(m+n)$ space.

B. Compute an optimal alignment in $O(mn)$ time and $O(m+n)$ space.

C. Both A and B.

D. Neither A nor B.

$$OPT(i,j) = \begin{cases} j\delta & \text{if i = 0} \\ i\delta & \text{if j = 0} \\ \min \begin{cases} \alpha_{x_i y_j} + OPT(i-1, j-1) \\ \delta + OPT(i-1, j) \\ \delta + OPT(i, j-1) \end{cases} & \text{otherwise} \end{cases}$$

Hirschberg’s Algorithm

Sequence alignment in linear space

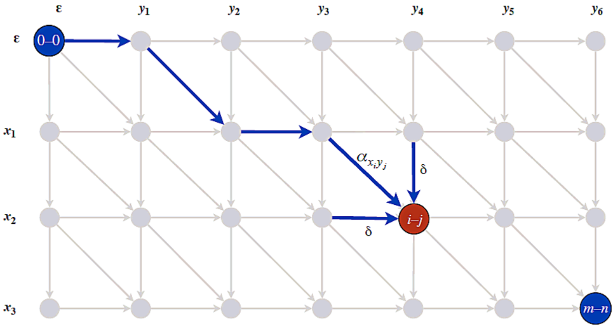
[Hirschberg] There exists an algorithm to find an optimal alignment in $O(mn)$ time and $O(m + n)$ space.

- Clever combination of divide-and-conquer and dynamic programming.
- Inspired by idea of Savitch from complexity theory.

Hirschberg's algorithm

Edit distance graph.

- Let $f(i, j)$ denote length of shortest path from $(0, 0)$ to (i, j).
- Lemma: $f(i, j) = OPT(i, j)$ for all i and j.



Edit distance graph.

- Let $f(i, j)$ denote length of shortest path from $(0, 0)$ to (i, j).
- Lemma: $f(i, j) = OPT(i, j)$ for all i and j.

Proof. [by strong induction on $i + j$]

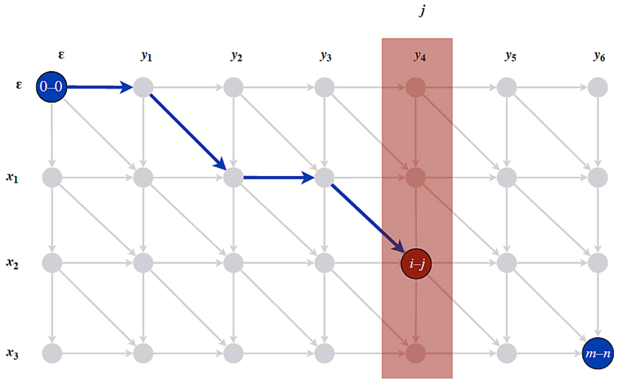
- Base case: $f(0, 0) = OPT(0, 0) = 0$.
- Inductive hypothesis: assume true for all (i', j') with $i' + j' < i + j$.
- Last edge on shortest path to (i, j) is from $(i - 1, j - 1)$, $(i - 1, j)$, or $(i, j - 1)$.
- Thus,

$$
\begin{aligned}
f(i, j) &= \min\left\{\alpha_{x_i y_j} + f(i-1, j-1), \delta + f(i-1, j), \delta + f(i, j-1)\right\} \\
&= \min\left\{\alpha_{x_i y_j} + OPT(i-1, j-1), \delta + OPT(i-1, j), \delta + OPT(i, j-1)\right\} \\
&= OPT(i, j)
\end{aligned}
$$

Hirschberg's algorithm

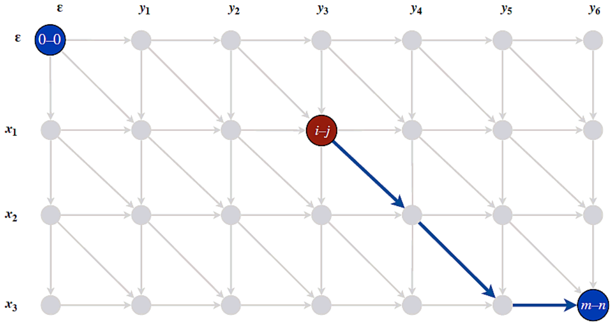
Edit distance graph.

- Let $f(i, j)$ denote length of shortest path from $(0, 0)$ to (i, j).
- Lemma: $f(i, j) = OPT(i, j)$ for all i and j.
- Can compute $f(\cdot, j)$ for any j in $O(mn)$ time and $O(m)$ space.



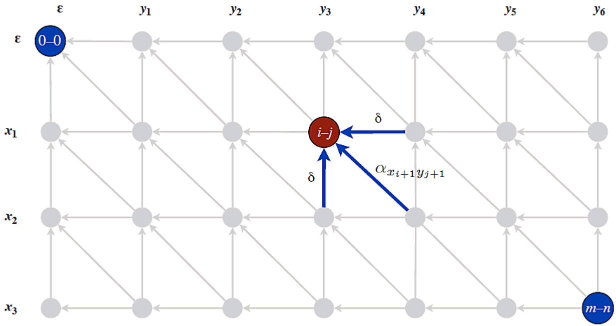
Edit distance graph.

- Let $g(i, j)$ denote length of shortest path from (i, j) to (m, n).



Edit distance graph.

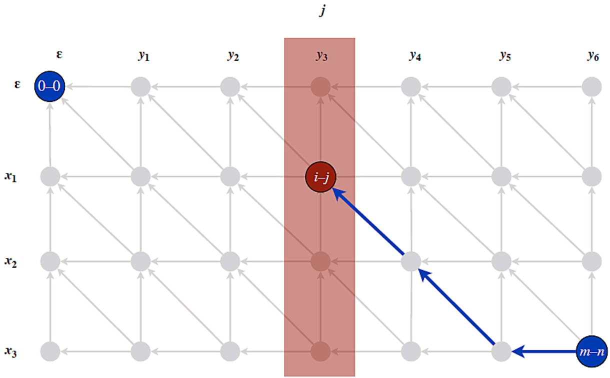
- Let $g(i, j)$ denote length of shortest path from (i, j) to (m, n).
- Can compute $g(i, j)$ by reversing the edge orientations and inverting the roles of $(0, 0)$ and (m, n).



Hirschberg's algorithm

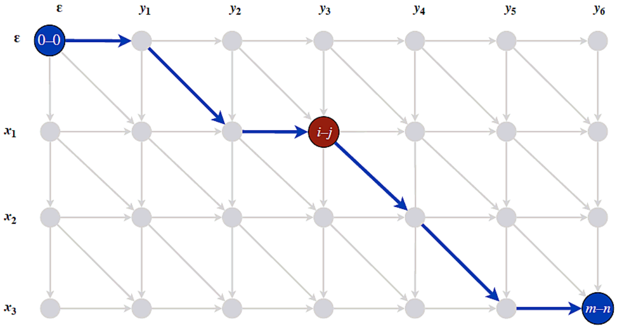
Edit distance graph.

- Let $g(i, j)$ denote length of shortest path from (i, j) to (m, n).
- Can compute $g(\cdot, j)$ for any j in $O(mn)$ time and $O(m)$ space.



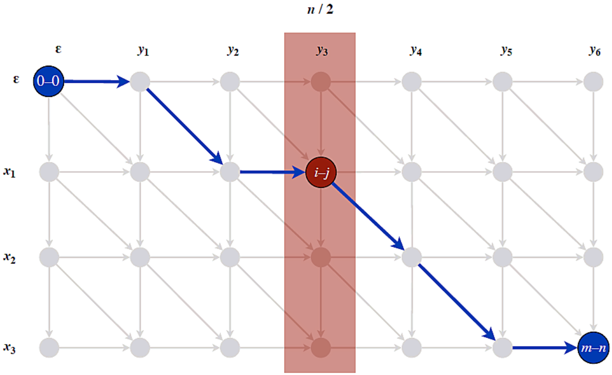
Hirschberg's algorithm

Observation 1. The length of a shortest path that uses (i, j) is $f(i, j) + g(i, j)$.



Hirschberg's algorithm

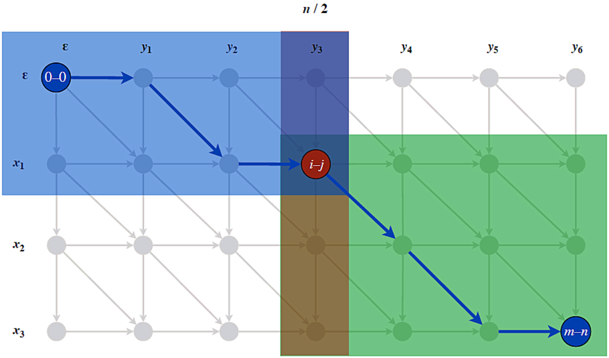
Observation 2. let q be an index that minimizes $f(q, n/2) + g(q, n/2)$. Then, there exists a shortest path from $(0, 0)$ to (m, n) that uses $(q, n/2)$.



Hirschberg's algorithm

Divide. Find index q that minimizes $f(q, n/2) + g(q, n/2)$; save node i-j as part of solution.

Conquer. Recursively compute optimal alignment in each piece.



Hirschberg's algorithm: space analysis

Theorem

Hirschberg's algorithm uses $\Theta(m+n)$ space.

Proof.

- Each recursive call uses $\Theta(m)$ space to compute $f(\cdot, n/2)$ and $g(\cdot, n/2)$.
- Only $\Theta(1)$ space needs to be maintained per recursive call.
- Number of recursive calls $\leq n$.

Quiz 4

What is the worst-case running time of Hirschberg's algorithm?

A. $O(mn)$
B. $O(mn \log m)$
C. $O(mn \log n)$
D. $O(mn \log m \log n)$

Hirschberg's algorithm: running time analysis warmup

Theorem

Let $T(m,n)$ be max running time of Hirschberg's algorithm on strings of lengths at most m and n. Then, $T(m,n) = O(mn \log n)$.

Proof.

- $T(m,n)$ is monotone nondecreasing in both m and n.
-
$$\begin{aligned} T(m,n) &\le 2T(m,n/2) + O(mn) \\ &\Rightarrow T(m,n) = O(mn \log n) \end{aligned}$$

Remark. Analysis is not tight because two subproblems are of size $(q, n/2)$ and $(m-q, n/2)$. Next, we prove $T(m,n) = O(mn)$.

Hirschberg's algorithm: running time analysis

Theorem

Let $T(m, n)$ be max running time of Hirschberg's algorithm on strings of lengths at most m and n. Then, $T(m, n) = O(mn)$.

Proof. [by strong induction on $m + n$]

- $O(mn)$ time to compute $f(\cdot, n/2)$ and $g(\cdot, n/2)$ and find index q.
- $T(q, n/2) + T(m - q, n/2)$ time for two recursive calls.
- Choose constant c so that:

$$T(m, 2) \leq cm$$
$$T(2, n) \leq cn$$
$$T(m, n) \leq cmn + T(q, n/2) + T(m - q, n/2)$$

Hirschberg's algorithm: running time analysis

Claim

$T(m, n) \leq 2cmn$

- Base cases: $m = 2$ and $n = 2$.
- Inductive hypothesis: $T(m, n) \leq 2cmn$ for all (m', n') with $m' + n' < m + n$.

$$\begin{aligned} T(m, n) &\leq T(q, n/2) + T(m - q, n/2) + cmn \\ &\leq 2cqn/2 + 2c(m - q)n/2 + cmn \\ &= cqn + cmn - cqn + cmn \\ &= 2cmn \end{aligned}$$

Longest common subsequence

Problem. Given two strings $x_1 x_2 \ldots x_m$ and $y_1 y_2 \ldots y_n$, find a common subsequence that is as long as possible.

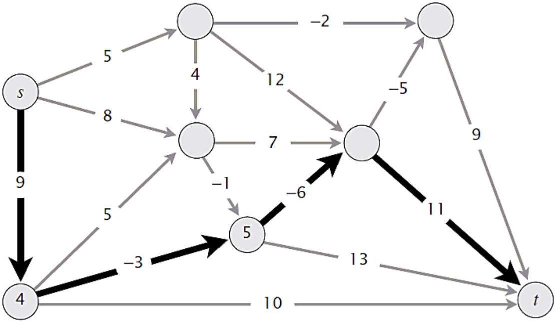
Alternative viewpoint. Delete some characters from x; delete some character from y; a common subsequence if it results in the same string.

Example. **LCS(GGCACCACG, ACGGCGGATACG) = GGCAACG**.

Applications. Unix diff, git, bioinformatics.

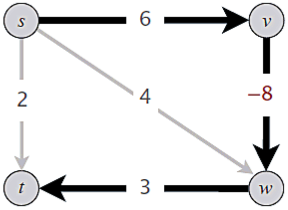
Bellman-Ford-Moore Algorithm

Shortest-path problem. Given a digraph $G = (V, E)$, with arbitrary edge lengths ℓ_{vw}, find shortest path from source node s to destination node t.



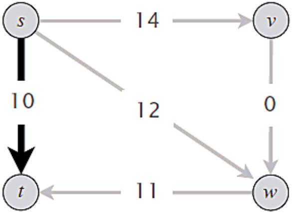
Shortest paths with negative weights: failed attempts

Dijkstra. May not produce shortest paths when edge lengths are negative.



Dijkstra selects the vertices in the orders, t, w, v
But shortest path from s to t is $s \to v \to w \to t$.

Reweighting. Adding a constant to every edge length does not necessarily make Dijkstra's algorithm produce shortest paths.

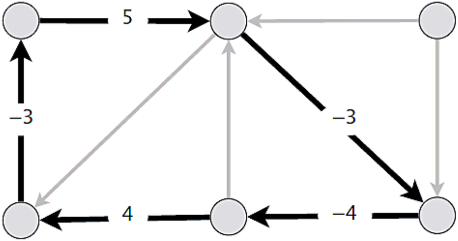


Adding 8 to each edge weight changes the shortest path from $s \to v \to w \to t$ to $s \to t$.

Negative cycles

Definition

A negative cycle is a directed cycle for which the sum of its edge lengths is negative.

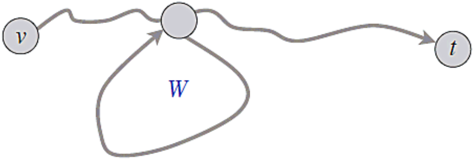


Lemma 1

If some $v \rightsquigarrow t$ path contains a negative cycle, then there does not exist a shortest $v \rightsquigarrow t$ path.

Proof.

If there exists such a cycle W, then can build a $v \rightsquigarrow t$ path of arbitrarily negative length by detouring around W as many times as desired.

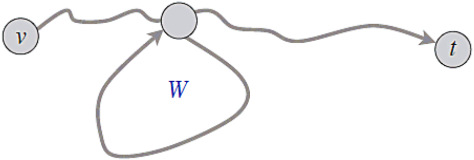


Lemma 2

If G has no negative cycles, then there exists a shortest $v \rightsquigarrow t$ path that is simple (and has $\leq n-1$ edges).

Proof.

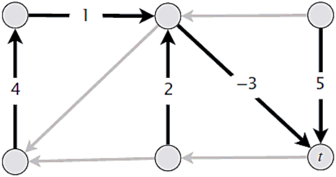
- Among all shortest $v \rightsquigarrow t$ paths, consider one that uses the fewest edges.
- If that path P contains a directed cycle W, can remove the portion of P corresponding to W without increasing its length.

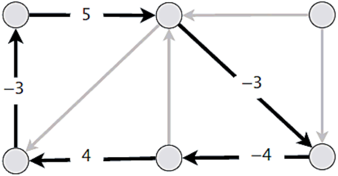


Shortest-paths and negative-cycle problems

Single-destination shortest-paths problem. Given a digraph $G = (V, E)$ with edge lengths ℓ_{vw} (but no negative cycles) and a distinguished node t, find a shortest $v \rightsquigarrow t$ path for every node v.

Negative-cycle problem. Given a digraph $G = (V, E)$ with edge lengths ℓ_{vw}, find a negative cycle (if one exists).





Quiz 5

Which subproblems to find shortest $v \rightsquigarrow t$ paths for every node v?

A. $OPT(i, v)$: length of shortest $v \rightsquigarrow t$ path that uses exactly i edges.

B. $OPT(i, v)$: length of shortest $v \rightsquigarrow t$ path that uses at most edges.

C. Neither A nor B.

Shortest paths with negative weights: dynamic programming

Definition. $OPT(i, v)$: length of shortest $v \rightsquigarrow t$ path that uses $\leq i$ edges.
Goal $OPT(n-1, v)$ for each v.

Case 1. Shortest $v \rightsquigarrow t$ path uses $\leq i-1$ edges.

- $OPT(i, v) = OPT(i-1, v)$

Case 2. Shortest $v \rightsquigarrow t$ path uses exactly i edges.

- if (v, w) is first edge in shortest such $v \rightsquigarrow t$ path, incur a cost of ℓ_{vw}.
- Then, select best $w \rightsquigarrow t$ path using $\leq i-1$ edges.

Bellman equation.

$$OPT(i, v) = \begin{cases} 0 & \text{if } i = 0 \text{ and } v = t \\ \infty & \text{if } i = 0 \text{ and } v \neq t \\ \min \left\{ OPT(i-1, v), \min\limits_{(v,w) \in E} \{ OPT(i-1, w) + \ell_{vw} \} \right\} & \text{if } i > 0 \end{cases}$$

Shortest paths with negative weights: implementation

```
ShortestPaths(V, E, ℓ, t)

for each node (v ∈ V) do
    M[0, v] ← ∞;
end
M[0, t] ← 0;
for i = 1 to n − 1 do
    for each node v ∈ V do
        M[i, v] ← M[i − 1, v];
        for each edge (v, w) ∈ E do
            M[i, v] ← min {M[i, v], M[i − 1, w] + ℓ_vw};
        end
    end
end
```

Shortest paths with negative weights: implementation

Theorem

Given a digraph $G = (V, E)$ with no negative cycles, the DP algorithm computes the length of a shortest $v \rightsquigarrow t$ path for every node v in $\Theta(mn)$ time and $\Theta(n^2)$ space.

Proof.

- Table requires $\Theta(n^2)$ space.
- Each iteration i takes $\Theta(m)$ time since we examine each edge once.

Finding the shortest paths.

- Approach 1: Maintain *successor* $[i, v]$ that points to next node on a shortest $v \rightsquigarrow t$ path using $\leq i$ edges.
- Approach 2: Compute optimal lengths $M[i, v]$ and consider only edges with $M[i, v] = M[i-1, w] + \ell_{vw}$. Any directed path in this subgraph is a shortest path.

Quiz 6

It is easy to modify the DP algorithm for shortest paths to . . .

A. Compute lengths of shortest paths in $O(mn)$ time and $O(m+n)$ space.
B. Compute shortest paths in $O(mn)$ time and $O(m+n)$ space.
C. Both A and B.
D. Neither A nor B.

Shortest paths with negative weights: practical improvements

Space optimization. Maintain two 1D arrays (instead of 2D array).

- $d[v]$: length of a shortest $v \rightsquigarrow t$ path that we have found so far.
- $successor[v]$: next node on a $v \rightsquigarrow t$ path.

Performance optimization. If $d[w]$ was not updated in iteration $i-1$, then no reason to consider edges entering w in iteration i.

Bellman-Ford-Moore: efficient implementation

```
Bellman-Ford-Moore(V, E, c, t)

for each node v ∈ V do
    d[v] ← ∞;
    successor[v] ← null;
end
d ← 0;
for i = 1 to n − 1 do
    for  each node w ∈ V do
        if d[w] was updated in previous pass then
            for each edge (v, w) ∈ E do
                if (d[v] > d[w] + ℓ_vw) then
                    d[v] ← d[w] + ℓ_ww;
                    successor[v] ← w;
                end
            end
        end
    end
    if no d[·] value changed in pass i then  Break;
end
```

Quiz 7

Which properties must hold after pass i of Bellman-Ford-Moore?

A. $d[v]$: length of a shortest $v \rightsquigarrow t$ path using $\leq i$ edges.

B. $d[v]$: length of a shortest $v \rightsquigarrow t$ path using exactly i edges.

C. Both A and B.

D. Neither A nor B.

Bellman-Ford-Moore: analysis

Theorem

Assuming no negative cycles, Bellman-Ford-Moore computes the lengths of the shortest $v \rightsquigarrow t$ paths in $O(mn)$ time and $\Theta(n)$ extra space.

Remark

Bellman-Ford-Moore is typically faster in practice.

- Edge (v, w) considered in pass $i + 1$ only if $d[w]$ updated in pass i.
- If shortest path has k edges, then algorithm finds it after $\leq k$ passes.

Quiz 8

Assuming no negative cycles, which properties must hold throughout Bellman-Ford-Moore?

A. Following $successor[v]$ pointers gives a directed $v \rightsquigarrow t$ path.

B. If following $successor[v]$ pointers gives a directed $v \rightsquigarrow t$ path, then the length of that $v \rightsquigarrow t$ path is $d[v]$.

C. Both A and B.

D. Neither A nor B.

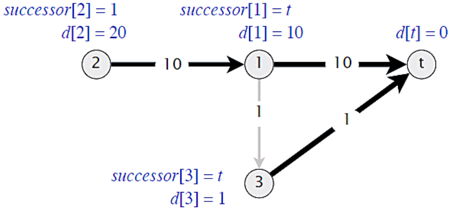
Claim

Throughout Bellman-Ford-Moore, following the $successor[v]$ pointers gives a directed path from v to t of length $d[v]$.

Counterexample. Claim is false!

- Length of successor $v \rightsquigarrow t$ path may be strictly shorter than $d[v]$.

consider nodes in order: t, 1, 2, 3



Bellman-Ford-Moore: analysis

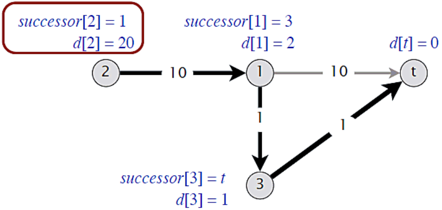
Claim

Throughout Bellman-Ford-Moore, following the $successor[v]$ pointers gives a directed path from v to t of length $d[v]$.

Counterexample. Claim is false!

- Length of successor $v \rightsquigarrow t$ path may be strictly shorter than $d[v]$.

consider nodes in order: t, 1, 2, 3



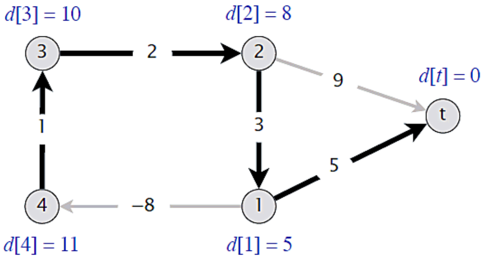
Claim

Throughout Bellman-Ford-Moore, following the successor[v] pointers gives a directed path from v to t of length $d[v]$.

Counterexample. Claim is false!

- Length of successor $v \rightsquigarrow t$ path may be strictly shorter than $d[v]$.
- If negative cycle, successor graph may have directed cycles.

consider nodes in order: $t, 1, 2, 3, 4$



Bellman-Ford-Moore: analysis

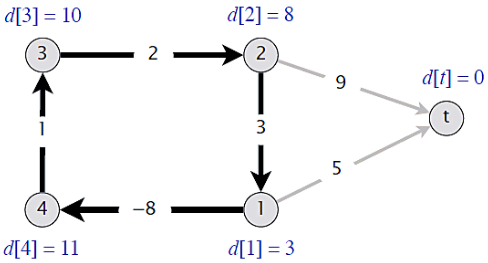
Claim

Throughout Bellman-Ford-Moore, following the successor$[v]$ pointers gives a directed path from v to t of length $d[v]$.

Counterexample. Claim is false!

- Length of successor $v \rightsquigarrow t$ path may be strictly shorter than $d[v]$.
- If negative cycle, successor graph may have directed cycles.

consider nodes in order: $t, 1, 2, 3, 4$



Bellman-Ford-Moore: finding the shortest paths

Lemma 6

Any directed cycle W in the successor graph is a negative cycle.

Proof.

- If $successor[v] = w$, we must have $d[v] \geq d[w] + \ell_{vw}$.
  (LHS and RHS are equal when $successor[v]$ is set; $d[w]$ can only decrease; $d[v]$ decreases only when $successor[v]$ is reset)
- Let $v_1 \to v_2 \to \ldots \to v_k \to v_1$ be the sequence of nodes in a directed cycle W.
- Assume that (v_k, v_1) is the last edge in W added to the successor graph.
- Just prior to that:

$$
\begin{array}{lll}
d\,[v_1] & \geq d\,[v_2] & +\ell\,(v_1, v_2) \\
d\,[v_2] & \geq d\,[v_3] & +\ell\,(v_2, v_3) \\
\vdots & \vdots & \vdots \\
d\,[v_{k-1}] & \geq d\,[v_k] & +\ell\,(v_{k-1}, v_k) \\
d\,[v_k] & > d\,[v_1] & +\ell\,(v_{k-1}, v_1)
\end{array}
$$

- Adding inequalities yields $\ell\,(v_1, v_2) + \ell\,(v_2, v_3) + \ldots + \ell\,(v_{k-1}, v_k) + \ell\,(v_k, v_1) < 0$

Bellman-Ford-Moore: finding the shortest paths

Theorem

Assuming no negative cycles, Bellman-Ford-Moore finds shortest $v \rightsquigarrow t$ paths for every node v in $O(mn)$ time and $\Theta(n)$ extra space.

Proof.

- The successor graph cannot have a directed cycle. [Lemma 6]
- Thus, following the successor pointers from v yields a directed path to t.
- Let $v = v_1 \to v_2 \to \ldots \to v_k = t$ be the nodes along this path P.
- Upon termination, if $\mathit{successor}[v] = w$, we must have $d[v] = d[w] + \ell_{vw}$.
  (LHS and RHS are equal when $\mathit{successor}[v]$ is set; $d[\cdot]$ did not change)
- Thus,

$$
\begin{array}{lll}
d\left[v_1\right] & = d\left[v_2\right] & +\ell\left(v_1, v_2\right) \\
d\left[v_2\right] & = d\left[v_3\right] & +\ell\left(v_2, v_3\right) \\
\vdots & \vdots & \vdots \\
d\left[v_{k-1}\right] & = d\left[v_k\right] & +\ell\left(v_{k-1}, v_k\right)
\end{array}
$$

- Adding equations yields $d[v] = d + \ell\left(v_1, v_2\right) + \ell\left(v_2, v_3\right) + \ldots + \ell\left(v_{k-1}, v_k\right)$

Single-source shortest paths with negative weights

year	worst case	discovered by
1955	$O\left(n^4\right)$	Shimbel
1956	$O\left(mn^2 W\right)$	Ford
1958	$O(mn)$	Bellman, Moore
1983	$O\left(n^{3/4} m \log W\right)$	Gabow
1989	$O\left(mn^{1/2} \log(nW)\right)$	Gabow-Tarjan
1993	$O\left(mn^{1/2} \log W\right)$	Goldberg
2005	$O\left(n^{2.38} W\right)$	Sankowsi, Yuster-Zwick
2016	$\tilde{O}\left(n^{10/7} \log W\right)$	Cohen-Madry-Sankowski-Vladu
20xx	???	

series single-source shortest paths with weights between $-W$ and W