

6. DYNAMIC PROGRAMMING II

- ▶ *sequence alignment*
- ▶ *Hirschberg's algorithm*

Assignment → Project Exam Help
→ *Bellman–Ford–Moore algorithm*

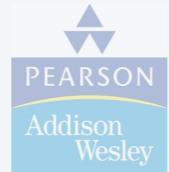
ctor protocols
<https://edulassistpro.github.io/>

Add WeChat *edu_assist_pro*
▶ neg es

Lecture slides by Kevin Wayne

Copyright © 2005 Pearson–Addison Wesley

<http://www.cs.princeton.edu/~wayne/kleinberg-tardos>



6. DYNAMIC PROGRAMMING II

- ▶ *sequence alignment*
- ▶ *Hirschberg's algorithm*

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

SECTION 6.6

String similarity

Q. How similar are two strings?

Ex. occurrence and occurrence.

o	c	u	r	r	a	n	c	e	Assignment	Project	Exam	Help	u	r	r	a	n	c	e
---	---	---	---	---	---	---	---	---	------------	---------	------	------	---	---	---	---	---	---	---

o	c	c	u	r	r	e	https://eduassistpro.github.io/	u	r	r	e	n	c	e
---	---	---	---	---	---	---	---------------------------------	---	---	---	---	---	---	---

6 mismatches, 1 gap Add WeChat edu_assist_pro 1 mismatch, 1 gap

o	c	-	u	r	r	-	a	n	c	e
---	---	---	---	---	---	---	---	---	---	---

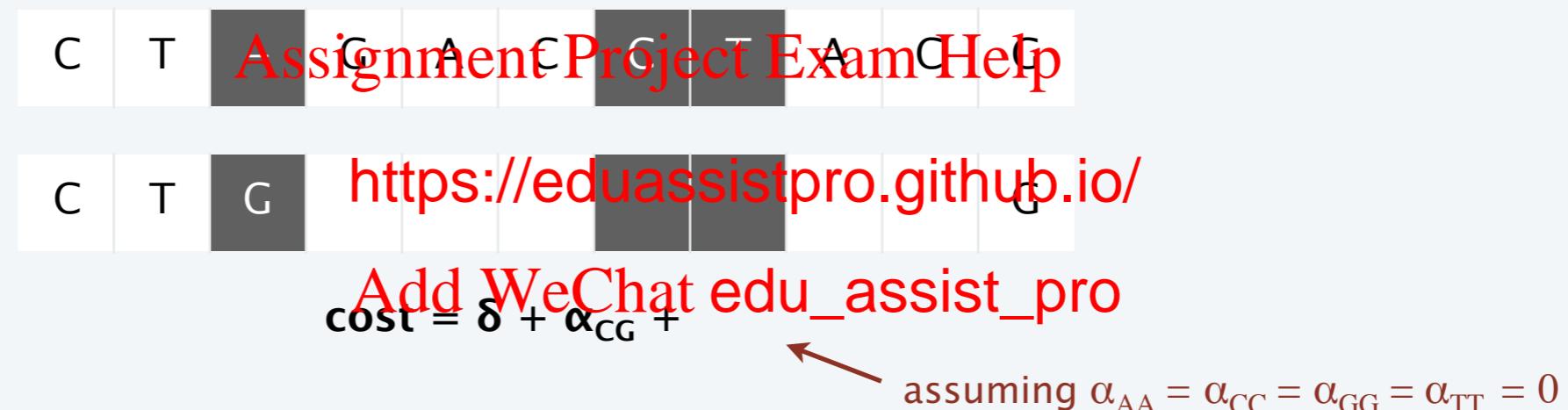
o	c	c	u	r	r	e	-	n	c	e
---	---	---	---	---	---	---	---	---	---	---

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.



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

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

BLOSUM matrix for proteins

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



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 Assignment Project Exam Help
- B. 2 <https://eduassistpro.github.io/>
- C. 3 Add WeChat edu_assist_pro
- D. 4
- E. 5

Sequence alignment

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

Def. 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.

$x_i - y_j$ and $x_{i'} - y_{j'}$ cross if $i < i'$, but $j > j'$

Def. The **cost** of an alignment M is:

Assignment Project Exam Help

$$\text{cost}(M) = \sum_{\substack{(x_i, y_j) \in M \\ \text{mismatch}}} \alpha_{x_i y_j} \delta$$

<https://eduassistpro.github.io/>
Add WeChat ^g [edu_assist_pro](#)

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

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 $x_1 x_2 \dots x_i$ and $y_1 y_2 \dots y_j$.

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 $x_1 x_2 \dots x_{i-1}$ and $y_1 y_2 \dots y_{j-1}$.

Assignment Project Exam Help

Case 2a. $OPT(i, j)$ leaves x_i

Pay gap for x_i + min cost of <https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Case 2b. $OPT(i, j)$ leaves y_j unmatched.

Pay gap for y_j + min cost of aligning $x_1 x_2 \dots x_i$ and $y_1 y_2 \dots y_{j-1}$.

optimal substructure property
(proof via exchange argument)

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

SEQUENCE-ALIGNMENT($m, n, x_1, \dots, x_m, y_1, \dots, y_n, \delta, \alpha$)

FOR $i = 0$ TO m

$M[i, 0] \leftarrow i \delta.$

FOR $j = 0$ TO n Assignment Project Exam Help

$M[0, j] \leftarrow j \delta.$

<https://eduassistpro.github.io/>

FOR $i = 1$ TO m Add WeChat edu_assist_pro

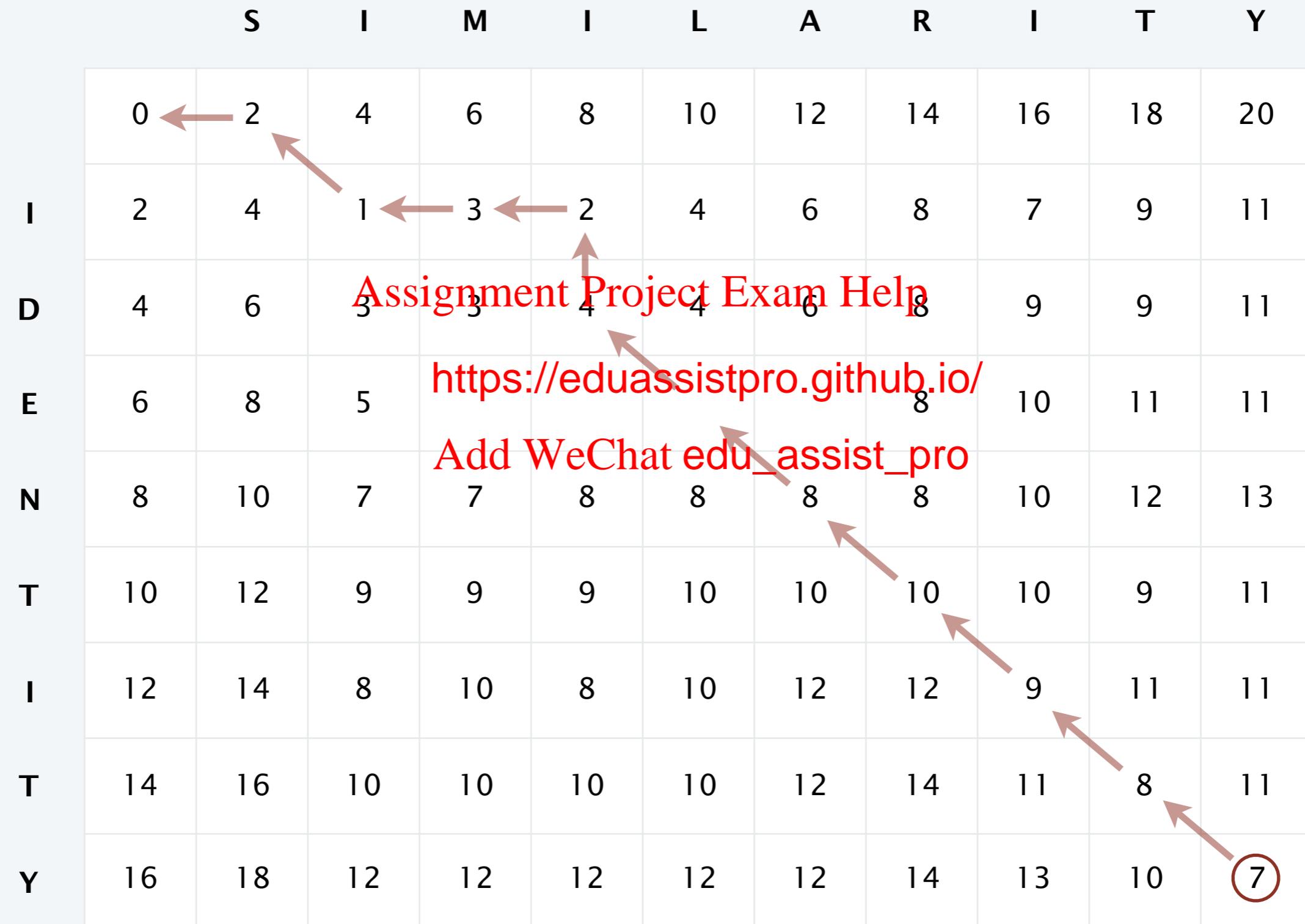
FOR $j = 1$ TO n

$M[i, j] \leftarrow \min \{ \alpha_{x_i y_j} + M[i - 1, j - 1],$
 $\delta + M[i - 1, j],$ 
 $\delta + M[i, j - 1] \}.$

already
computed

RETURN $M[m, n].$

Sequence alignment: traceback



Sequence alignment: analysis

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.

Pf.

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

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Theorem. [Backurs–Indyk 2015] If we can compute the 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 $\text{poly}(m) 2^{(1-\delta)n}$ time for some constant $\delta > 0$.

Edit Distance Cannot Be Computed
in Strongly Subquadratic Time
(unless SETH is false)*

Arturs Backurs[†]
MIT

Piotr Indyk[‡]
MIT

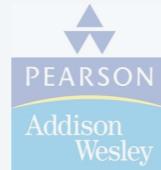
which would disprove SETH
(strong exponential time hypothesis)



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. [Assignment](#) [Project](#) [Exam](#) [Help](#)
- D. Neither A nor B. <https://eduassistpro.github.io/>
[Add WeChat edu_assist_pro](#)

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



6. DYNAMIC PROGRAMMING II

- ▶ *sequence alignment*
- ▶ *Hirschberg's algorithm*

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

SECTION 6.7

Sequence alignment in linear space

Theorem. [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.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

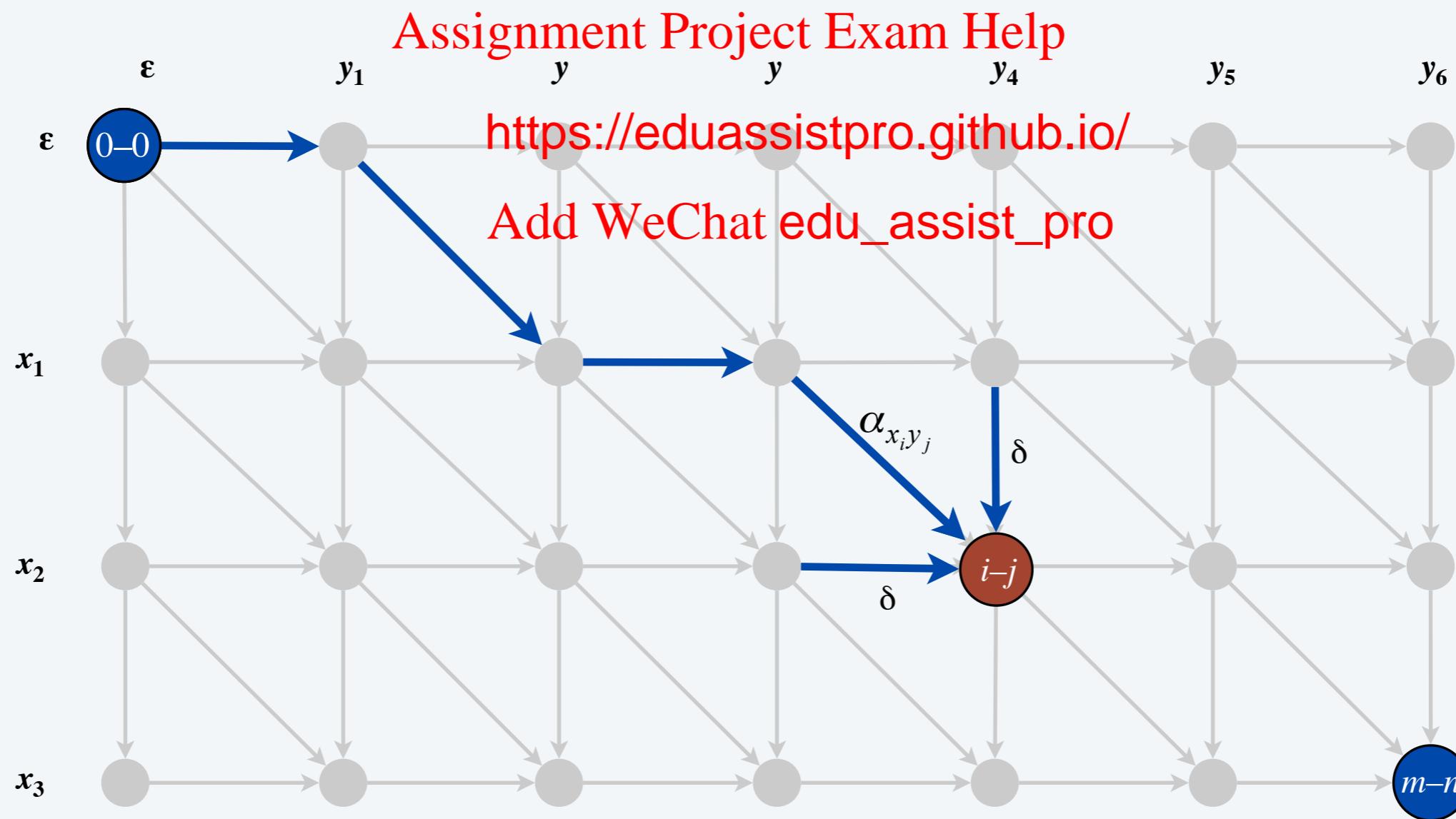
Add WeChat edu_assist_pro



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 .



Hirschberg's algorithm

Edit distance graph.

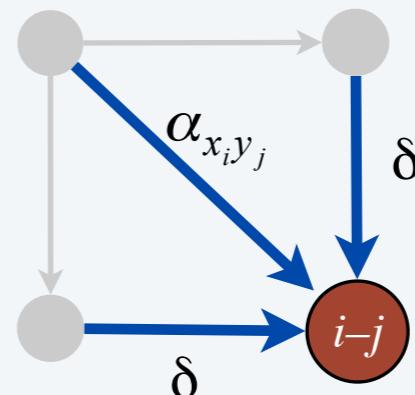
- Let $f(i, j)$ denote length of shortest path from $(0, 0)$ to (i, j) .
- Lemma: $f(i, j) = OPT(i, j)$ for all i and j .

Pf of Lemma. [by strong induction on $i + j$]

- Base case: $f(0, 0) = OPT(0, 0) = 0$
- Inductive hypothesis: $f(i', j')$ with $i' + j' < i + j$.
- Last edge on shortest <https://eduassistpro.github.io/>, $(i - 1, j - 1)$, $(i - 1, j)$, or $(i, j - 1)$.
- Thus, [Add WeChat edu_assist_pro](#)

$$f(i, j) = \min\{\alpha_{x_i y_j} + f(i - 1, j - 1), \delta + f(i - 1, j), \delta + f(i, j - 1)\}$$

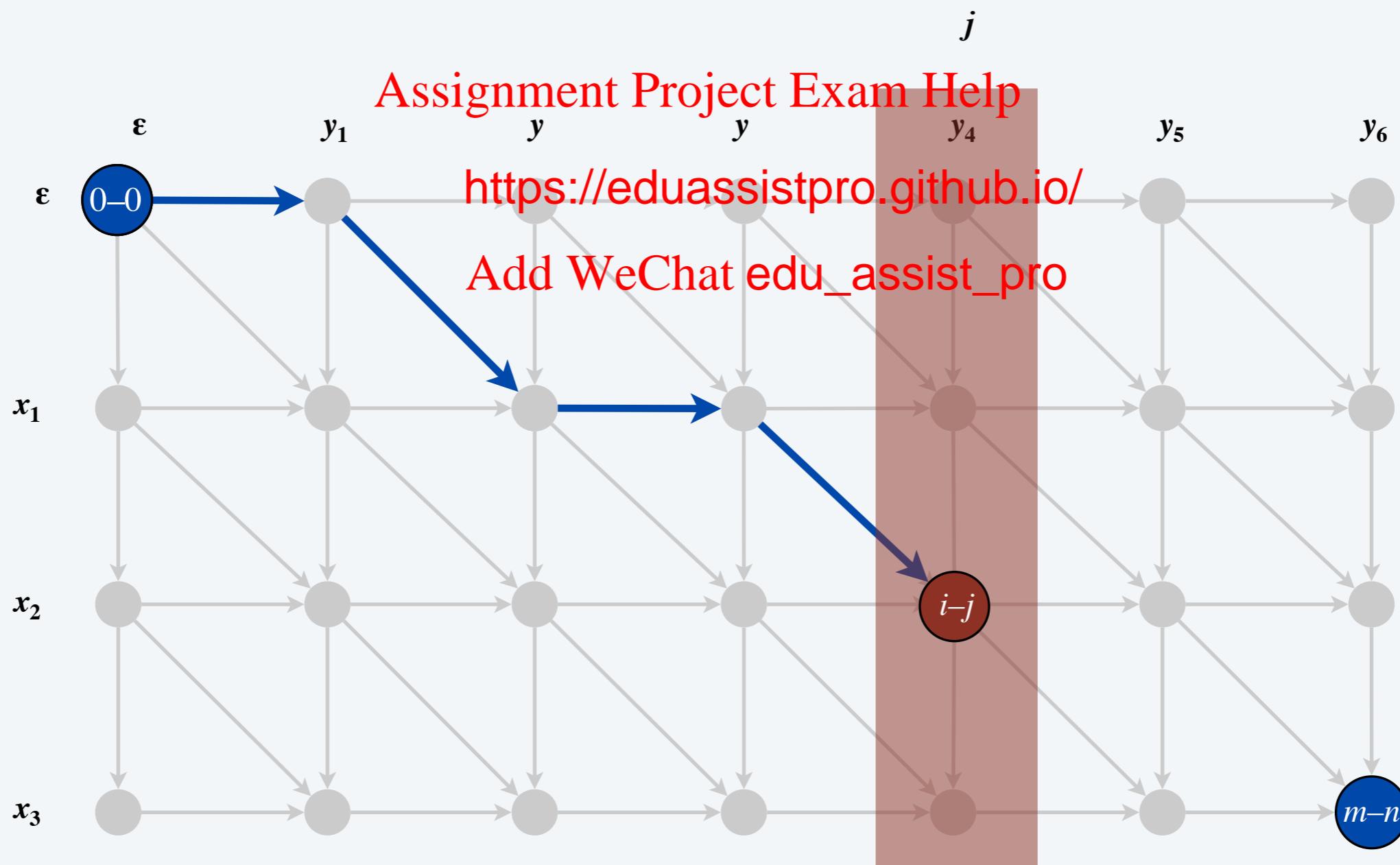
$$\begin{aligned} &= \min\{\alpha_{x_i y_j} + OPT(i - 1, j - 1), \delta + OPT(i - 1, j), \delta + OPT(i, j - 1)\} \\ \text{inductive hypothesis} \nearrow &= OPT(i, j) \blacksquare \\ \text{Bellman equation} \nearrow & \end{aligned}$$



Hirschberg's algorithm

Edit distance graph.

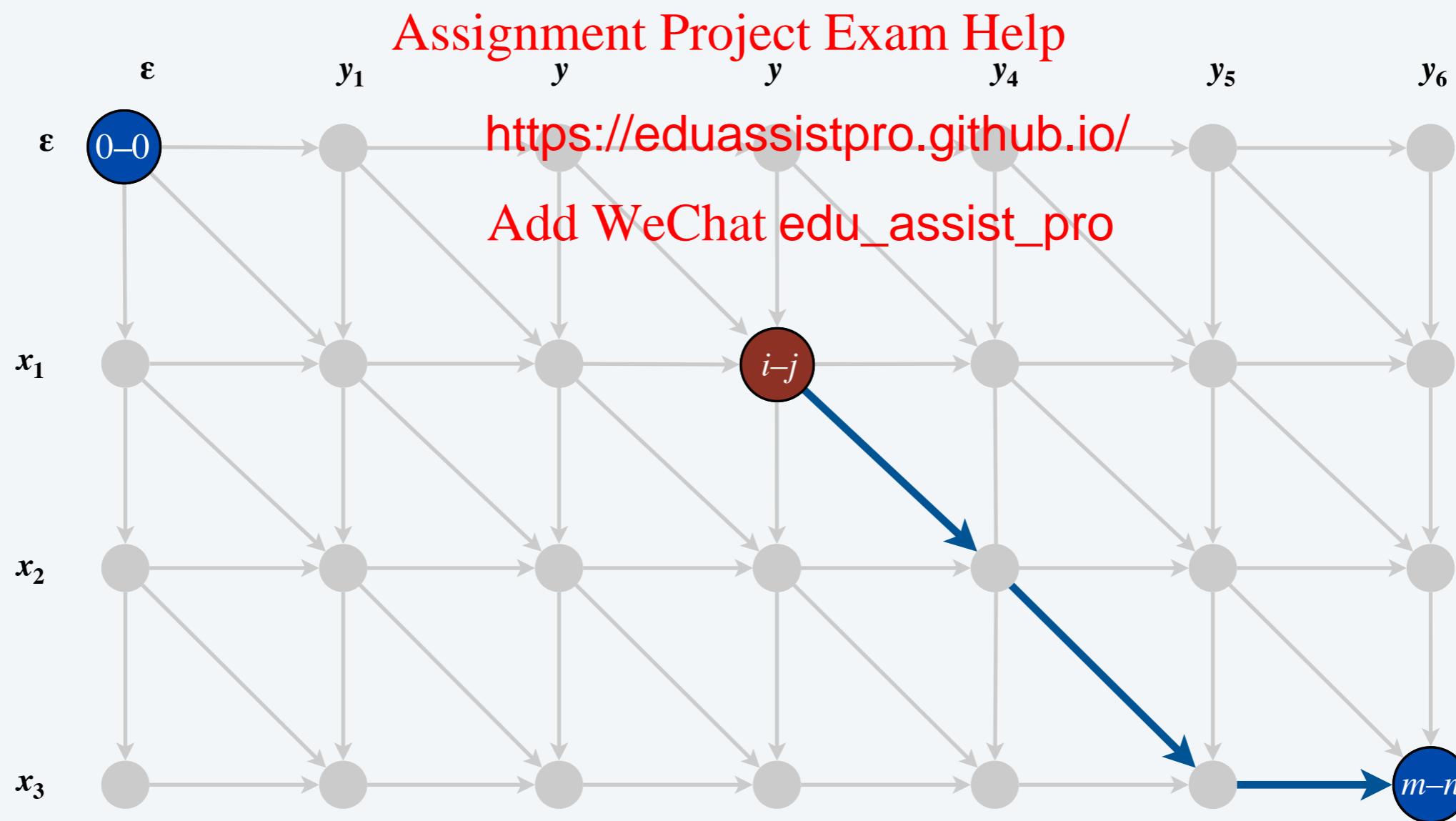
- Let $f(i, j)$ denote length of shortest path from $(0, 0)$ to (i, j) .
- Lemma: $f(i, j) = OPT(i, j)$ for all i and j .
- Can compute $f(\cdot, j)$ for any j in $O(mn)$ time and $O(m + n)$ space.



Hirschberg's algorithm

Edit distance graph.

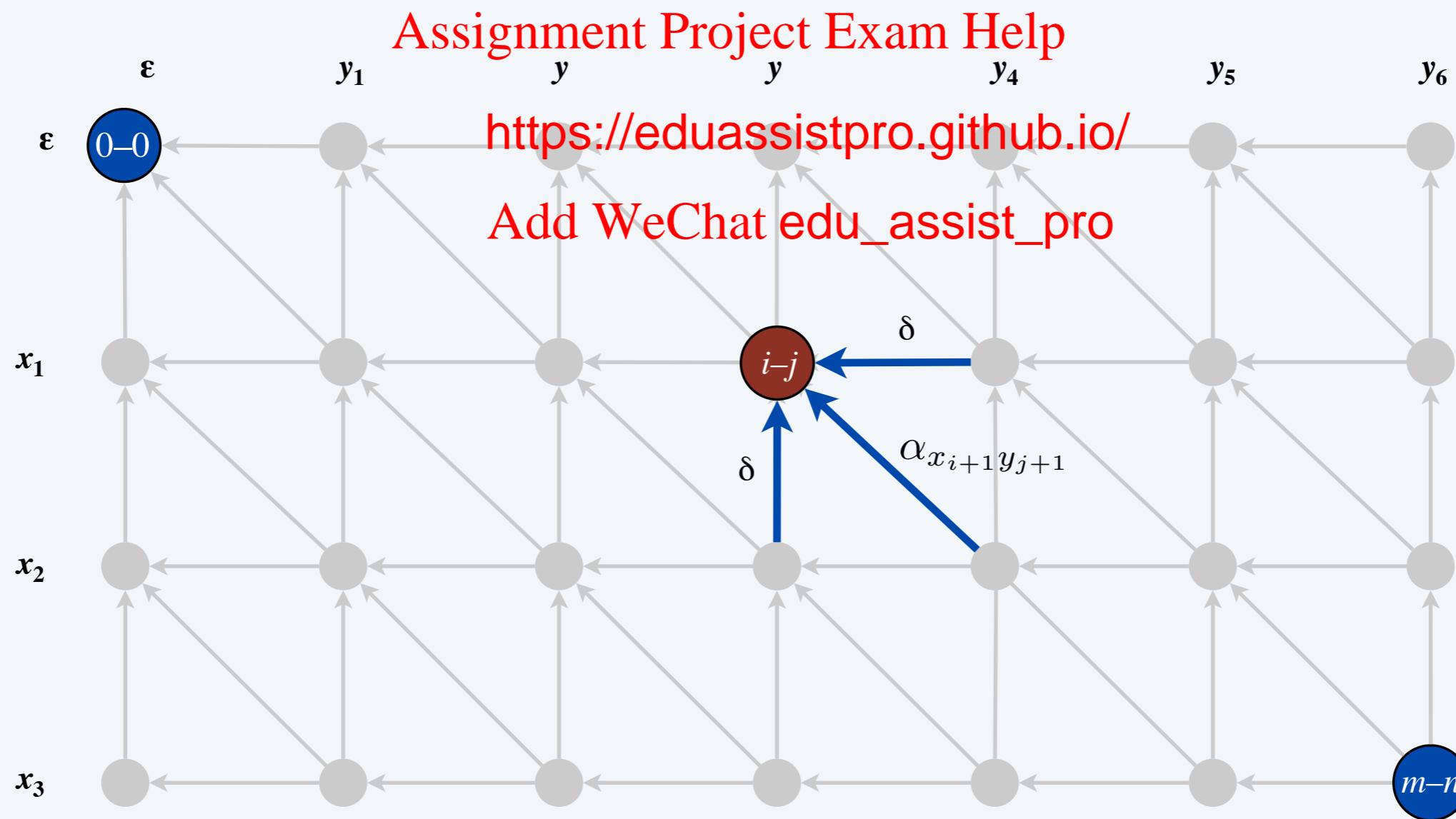
- Let $g(i, j)$ denote length of shortest path from (i, j) to (m, n) .



Hirschberg's algorithm

Edit distance graph.

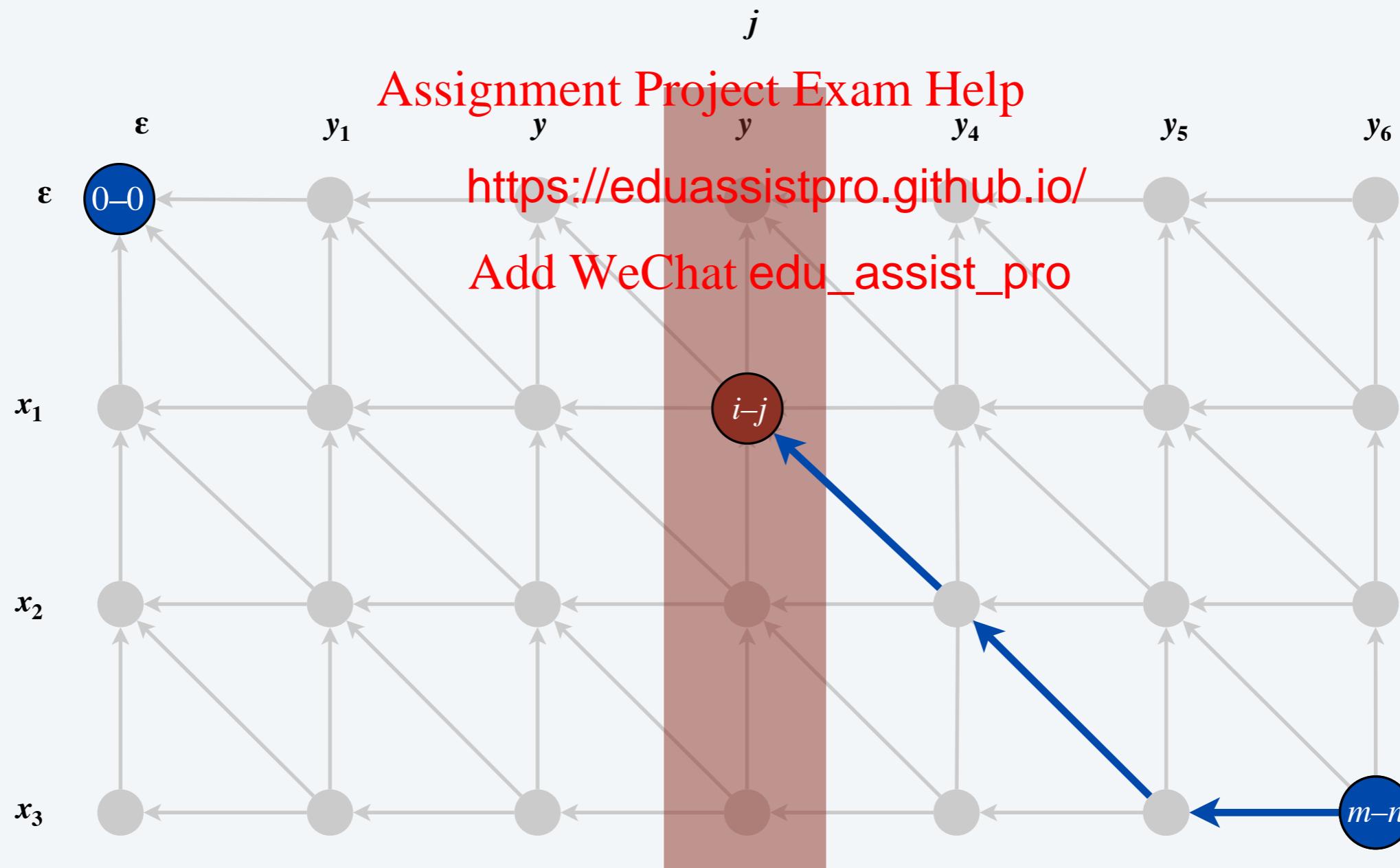
- Let $g(i, j)$ denote length of shortest path from (i, j) to (m, n) .
- Can compute $g(i, j)$ by reversing the edge orientations and inverting the roles of $(0, 0)$ and (m, n) .



Hirschberg's algorithm

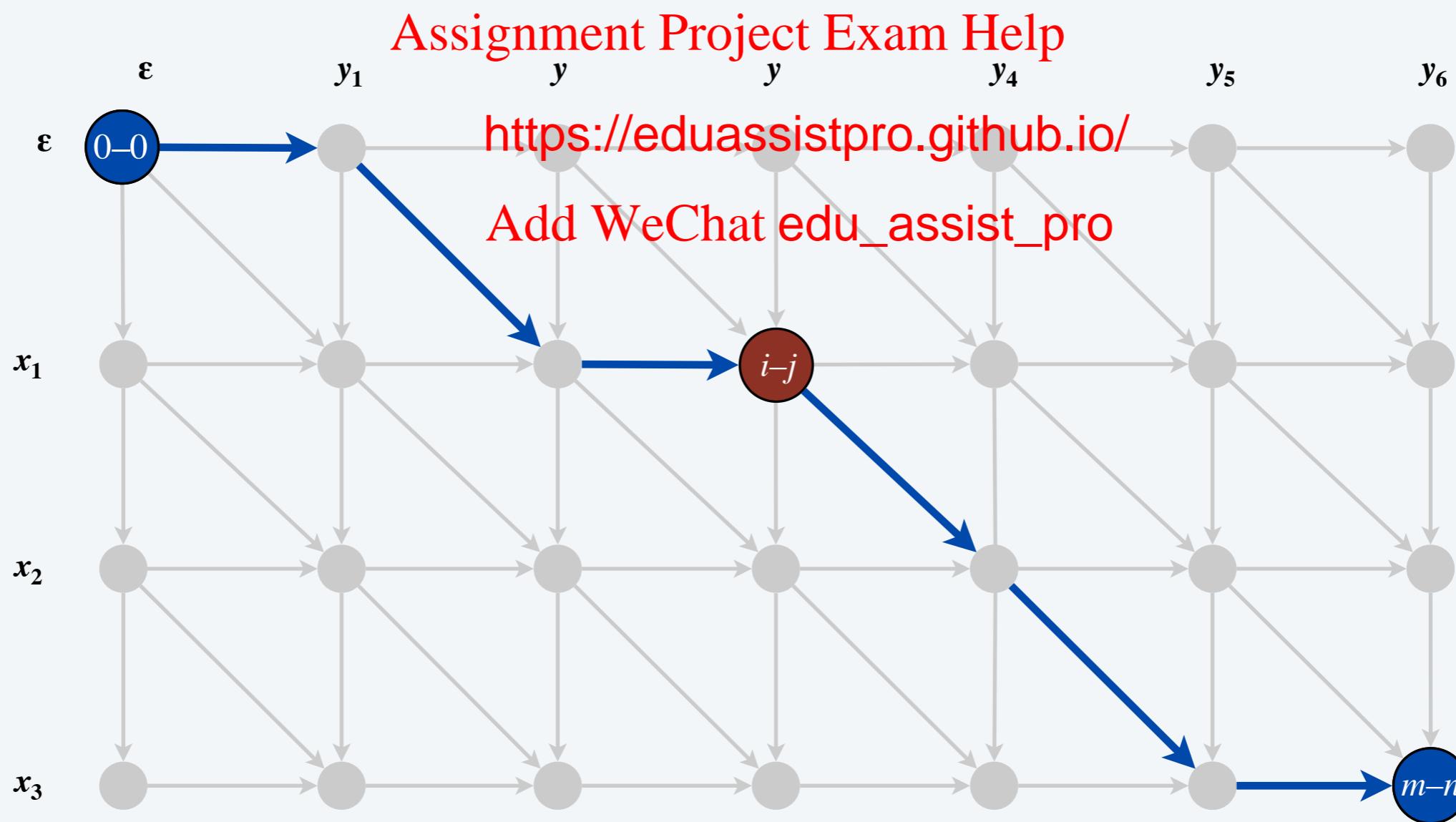
Edit distance graph.

- 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 $O(m + n)$ space.



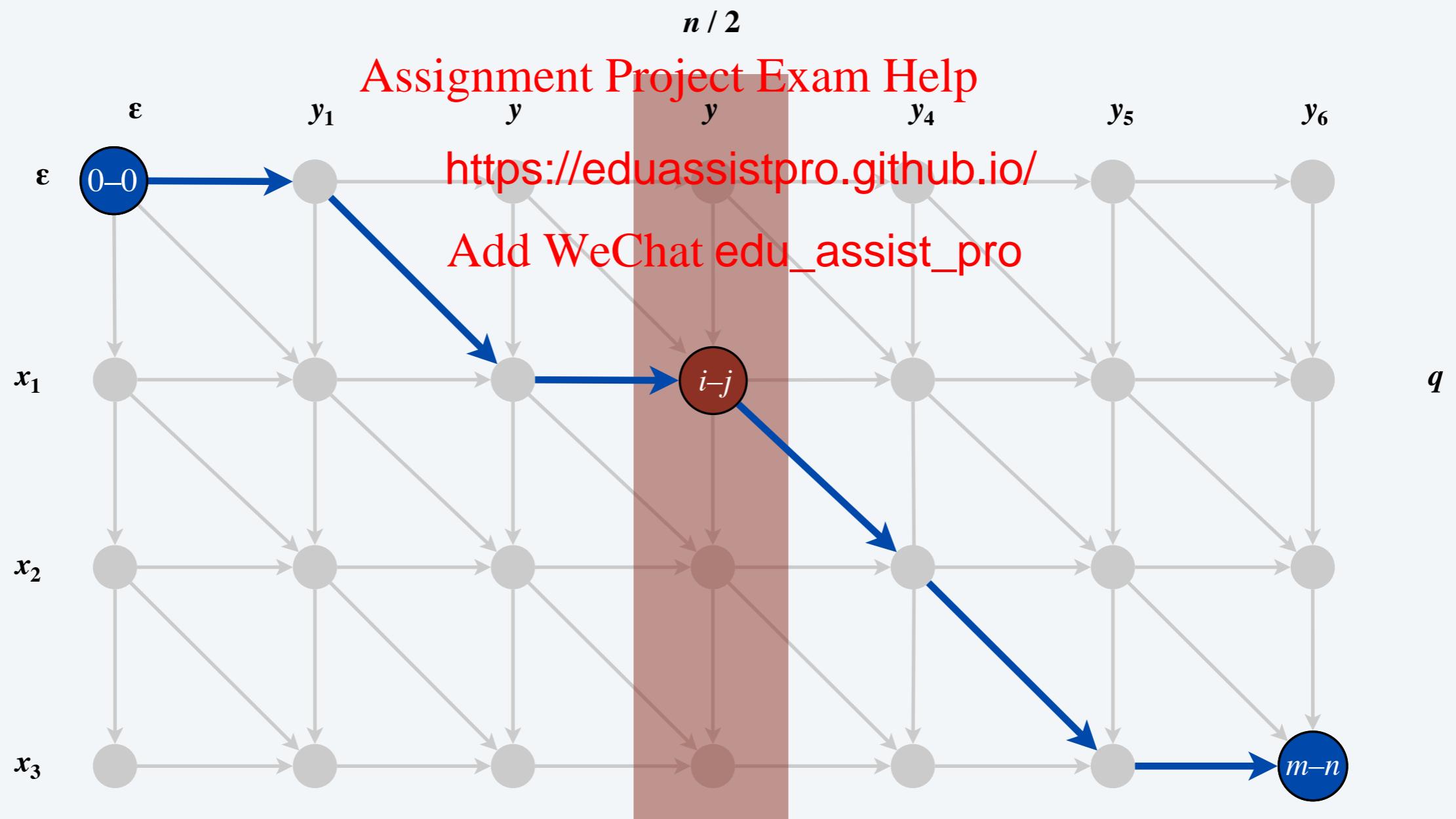
Hirschberg's algorithm

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



Hirschberg's algorithm

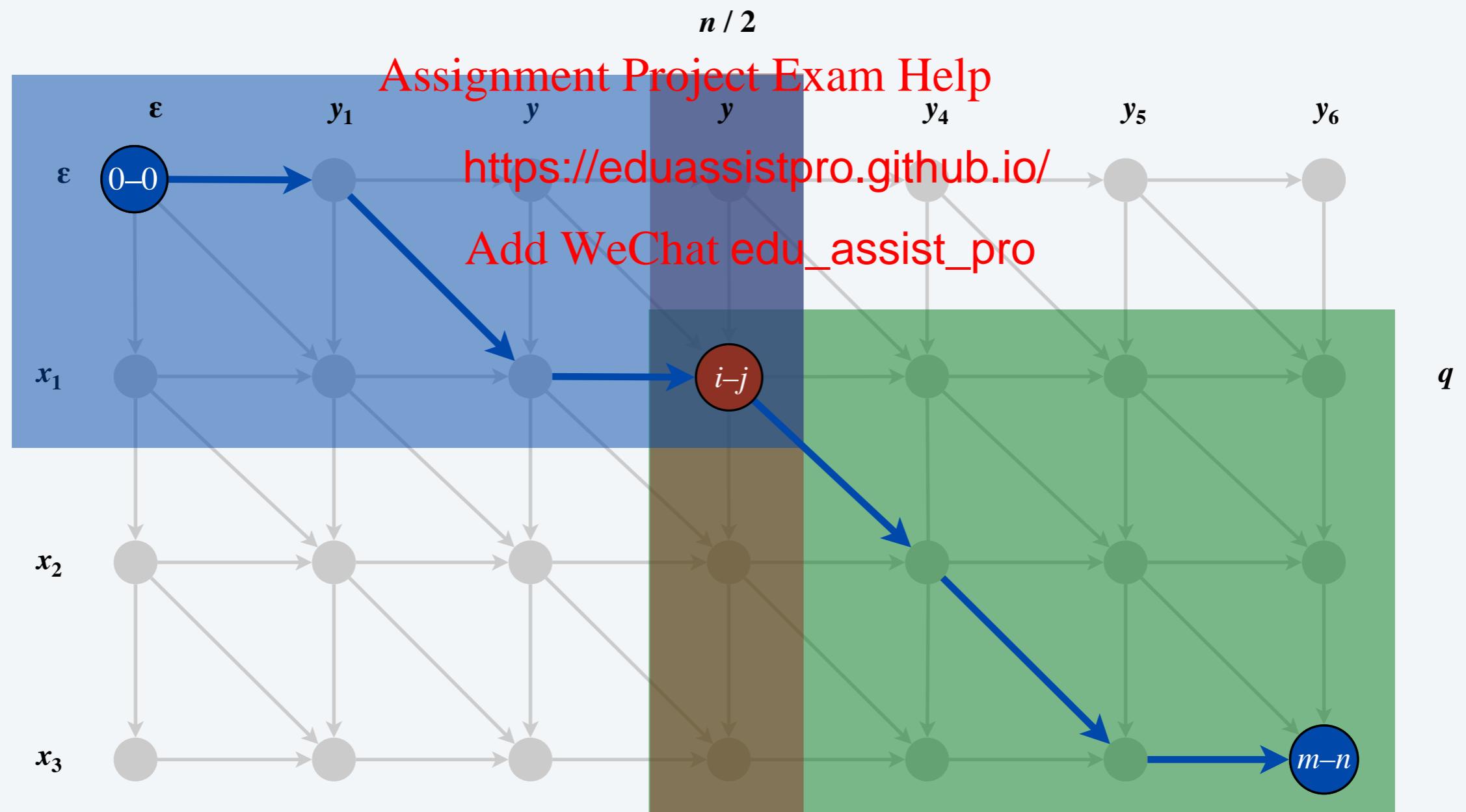
Observation 2. let q be an index that minimizes $f(q, n/2) + g(q, n/2)$.
Then, there exists a shortest path from $(0, 0)$ to (m, n) that uses $(q, n/2)$.



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.

Pf.

- 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 $\leq n$. ■

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



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. $O(mn \log m \log n)$

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Hirschberg's algorithm: running time analysis warmup

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

Pf.

- $T(m, n)$ is monotone nondecreasing in both m and n .
 - $T(m, n) \leq 2 T(m, n/2) + O(m n)$
 $\Rightarrow T(m, n) = O(m n \log n)$
- Assignment Project Exam Help
<https://eduassistpro.github.io/>
Add WeChat edu_assist_pro

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

Hirschberg's algorithm: running time analysis

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

Pf. [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 t

<https://eduassistpro.github.io/>

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

- 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 $m' + n' < m + n$.

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

inductive hypothesis

LONGEST COMMON SUBSEQUENCE

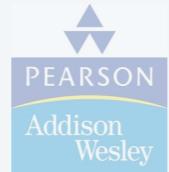


Problem. Given two strings $x_1 x_2 \dots x_m$ and $y_1 y_2 \dots y_n$, find a common subsequence that is as long as possible.

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

Ex. LCS(GGCA~~CC~~ACG, AC
<https://eduassistpro.github.io/>
 AACG.
 Add WeChat edu_assist_pro

Applications. Unix diff, git, bioinformatics.



6. DYNAMIC PROGRAMMING II

- ▶ *sequence alignment*
- ▶ *Hirschberg's algorithm*

Assignment Project Exam Help
▶ *Bellman–Ford–Moore algorithm*

<https://eduassistpro.github.io/>

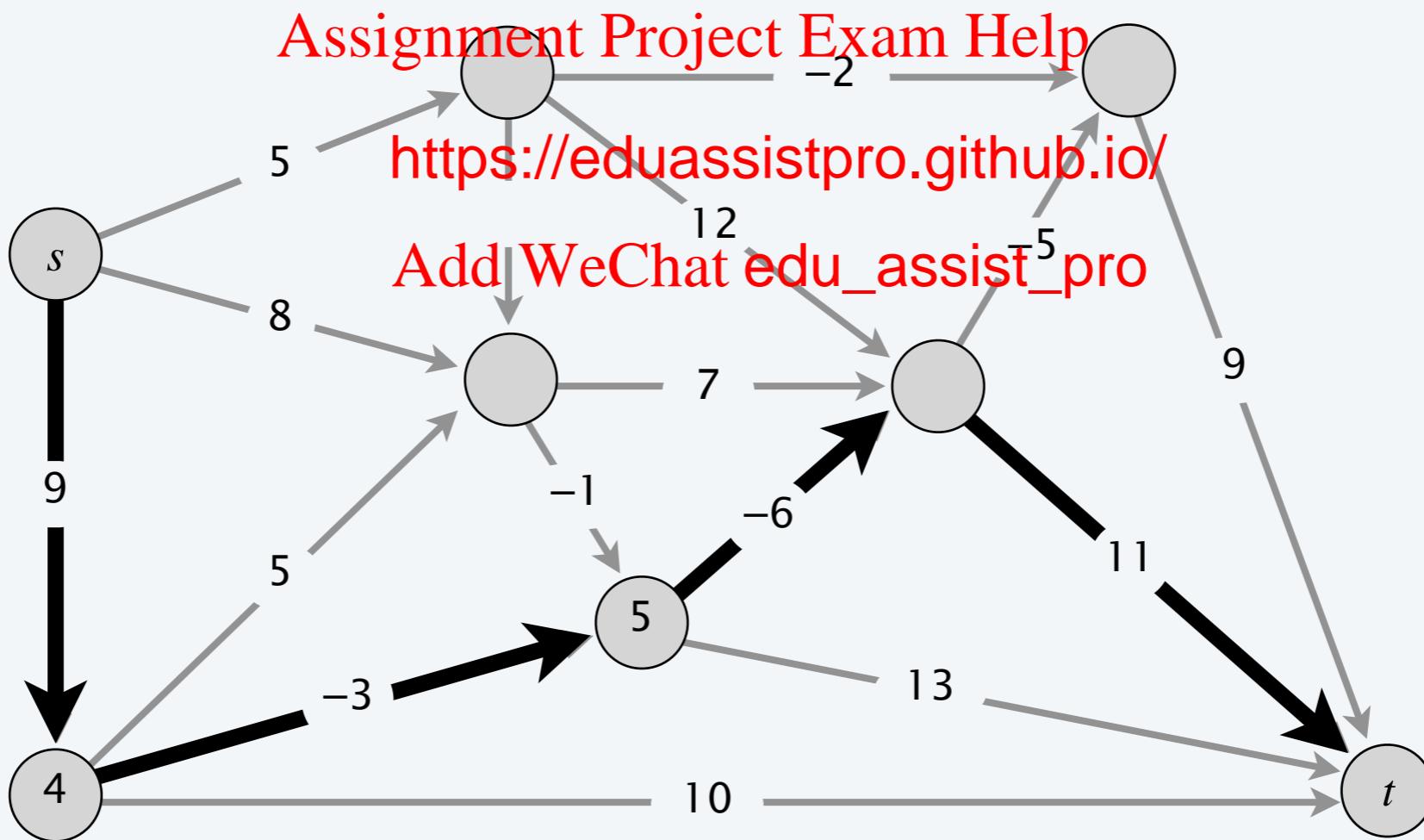
Add WeChat edu_assist_pro

SECTION 6.8

Shortest paths with negative weights

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 .

assume there exists a path
from every node to t



length of shortest path from s to t = $9 - 3 - 6 + 11 = 11$

Shortest paths with negative weights: failed attempts

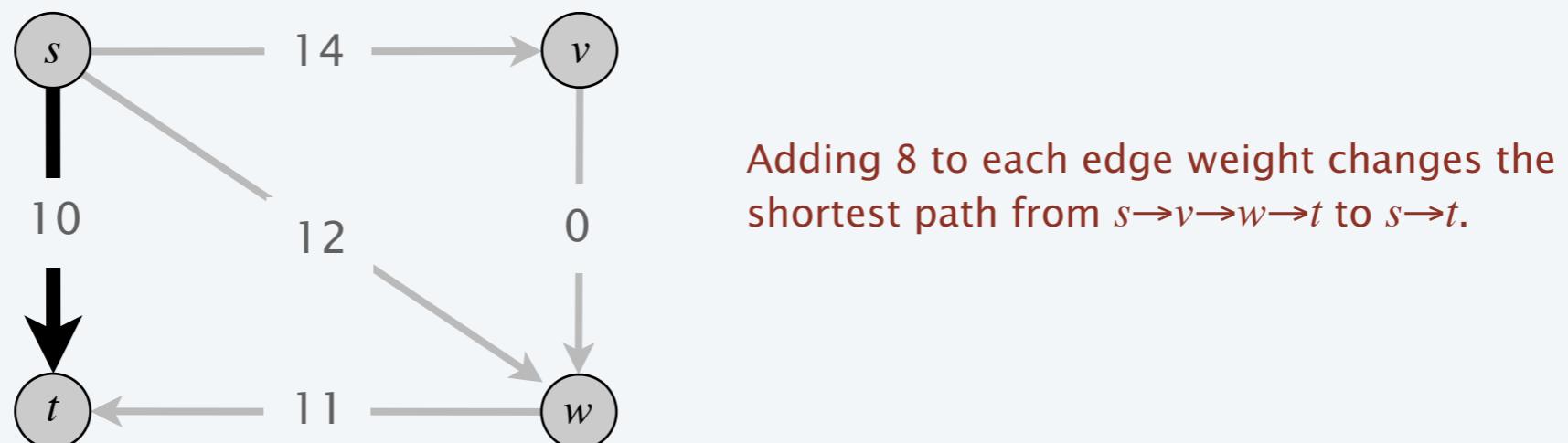
Dijkstra. May not produce shortest paths when edge lengths are negative.



<https://eduassistpro.github.io/>

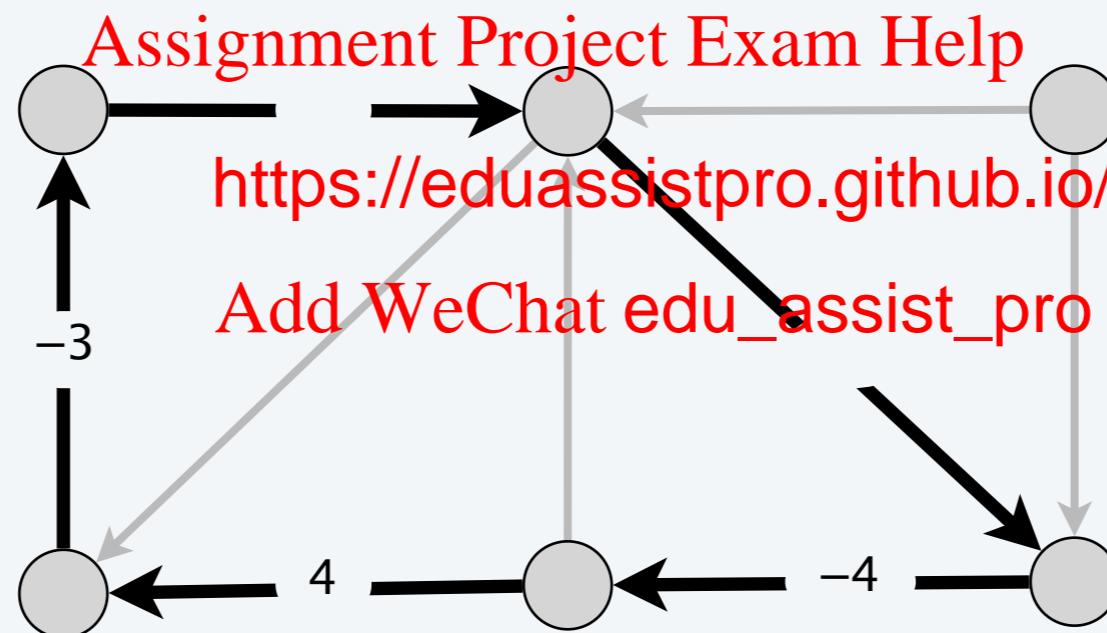
Add WeChat edu_assist_pro

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



Negative cycles

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



a **negative cycle** W : $\ell(W) = \sum_{e \in W} \ell_e < 0$

Shortest paths and negative cycles

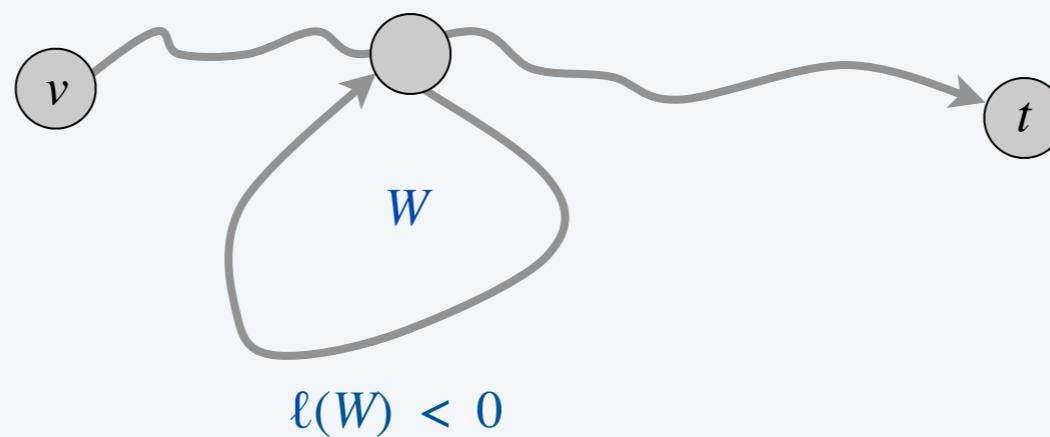
Lemma 1. If some $v \rightarrow t$ path contains a negative cycle, then there does not exist a shortest $v \rightarrow t$ path.

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

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Shortest paths and negative cycles

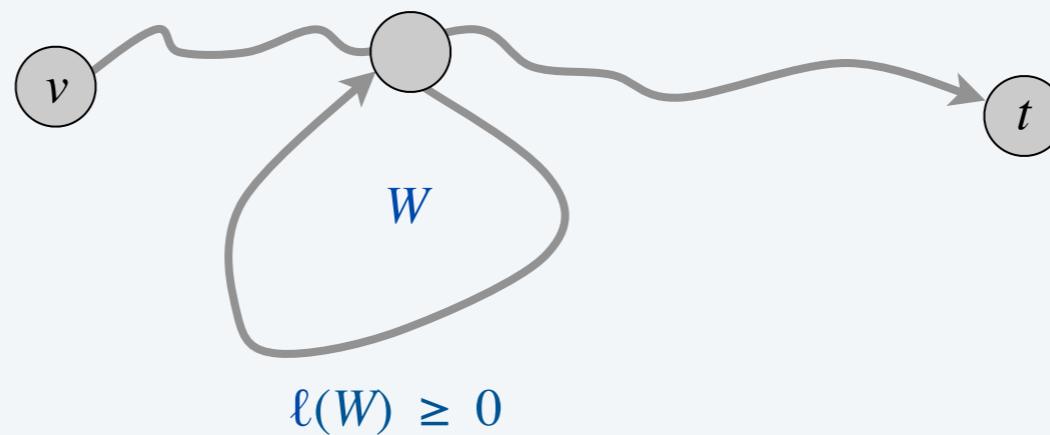
Lemma 2. If G has no negative cycles, then there exists a shortest $v \rightarrow t$ path that is simple (and has $\leq n - 1$ edges).

Pf.

- Among all shortest $v \rightarrow t$ 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 with \blacksquare

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Shortest-paths and negative-cycle problems

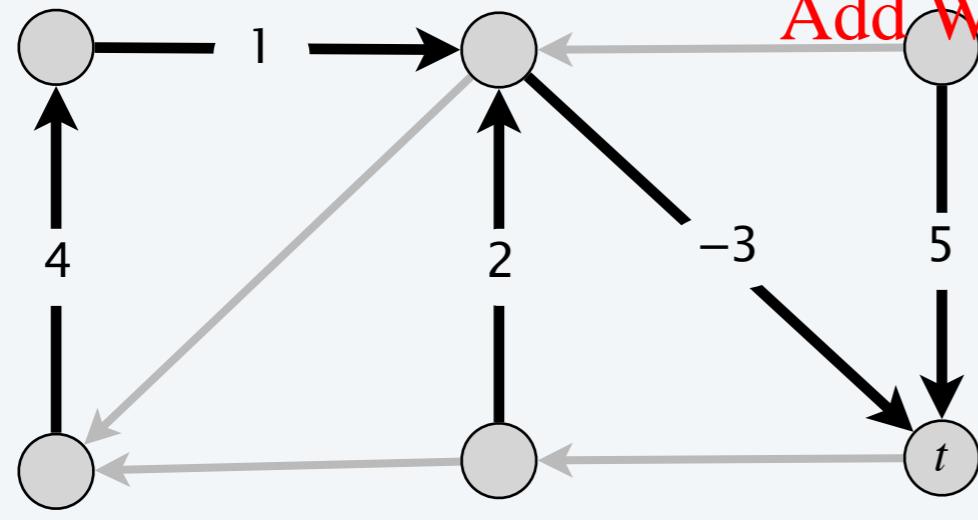
Single-destination shortest-paths problem. Given a digraph $G = (V, E)$ with edge lengths ℓ_{vw} (but no negative cycles) and a distinguished node t , find a shortest $v \rightarrow t$ path for every node v .

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

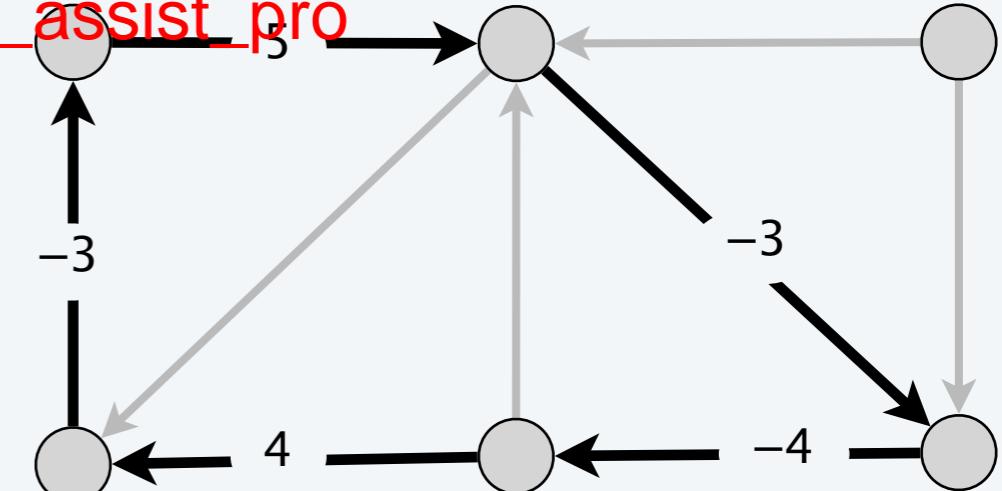
Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



shortest-paths tree



negative cycle



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

- A. $OPT(i, v) =$ length of shortest $v \rightarrow t$ path that uses exactly i edges.
- B. $OPT(i, v) =$ length of shortest $v \rightarrow t$ path that uses at most edges.
- C. Neither A nor B.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Shortest paths with negative weights: dynamic programming

Def. $OPT(i, v)$ = length of shortest $v \rightarrow t$ path that uses $\leq i$ edges.

Goal. $OPT(n - 1, v)$ for each v .

by Lemma 2, if no negative cycles,
there exists a shortest $v \rightarrow t$ path that is simple

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

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

Assignment Project Exam Help

optimal substructure property
(proof via exchange argument)

Case 2. Shortest $v \rightarrow t$ path <https://eduassistpro.github.io/>

- if (v, w) is first edge in shortest path such that $w \neq t$,
then $OPT(i, v) = \min_{w \in V} \{OPT(i - 1, w) + \ell_{vw}\}$.
- Then, select best $w \rightarrow t$ path using $\leq i - 1$ edges.

Bellman equation.

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

Shortest paths with negative weights: implementation

SHORTEST-PATHS(V, E, ℓ, t)

FOREACH node $v \in V$:

$M[0, v] \leftarrow \infty$. Assignment Project Exam Help

$M[0, t] \leftarrow 0$. <https://eduassistpro.github.io/>

FOR $i = 1$ TO $n - 1$ Add WeChat edu_assist_pro

FOREACH node $v \in V$:

$M[i, v] \leftarrow M[i - 1, v]$.

FOREACH edge $(v, w) \in E$:

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

Shortest paths with negative weights: implementation

Theorem 1. Given a digraph $G = (V, E)$ with no negative cycles, the DP algorithm computes the length of a shortest $v \rightarrow t$ path for every node v in $\Theta(mn)$ time and $\Theta(n^2)$ space.

Pf.

- Table requires $\Theta(n^2)$ space.
- Each item in the table is reached once.

<https://eduassistpro.github.io/>

■

Finding the shortest paths. Add WeChat edu_assist_pro

- Approach 1: Maintain $\text{successor}[i, v]$ that points to next node on a shortest $v \rightarrow t$ 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.



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. [Assignment](#) [Project](#) [Exam](#) [Help](#)
- D. Neither A nor B. <https://eduassistpro.github.io/>
[Add WeChat edu_assist_pro](#)

Shortest paths with negative weights: practical improvements

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

- $d[v]$ = length of a shortest $v \rightarrow t$ path that we have found so far.
- $\text{successor}[v]$ = next node on a $v \rightarrow t$ path.

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

[Assignment](#) [Project](#) [Exam](#) [Help](#)

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Bellman–Ford–Moore: efficient implementation

BELLMAN–FORD–MOORE(V, E, c, t)

FOREACH node $v \in V$:

$d[v] \leftarrow \infty.$

$successor[v] \leftarrow null.$

Assignment Project Exam Help

$d[t] \leftarrow 0.$

FOR $i = 1$ TO $n - 1$ <https://eduassistpro.github.io/>

FOREACH node $w \in V$: Add WeChat edu_assist_pro

IF ($d[w]$ was updated in previous pass)

FOREACH edge $(v, w) \in E$:

IF ($d[v] > d[w] + \ell_{vw}$)

$d[v] \leftarrow d[w] + \ell_{vw}.$

$successor[v] \leftarrow w.$

pass i
 $O(m)$ time

IF (no $d[\cdot]$ value changed in pass i) STOP.



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

- A. $d[v] = \text{length of a shortest } v \rightarrow t \text{ path using } \leq i \text{ edges.}$
- B. $d[v] = \text{length of a shortest } v \rightarrow t \text{ path using exactly } i \text{ edges.}$
- C. Both A and B.
- D. Neither A nor B.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Bellman–Ford–Moore: analysis

Lemma 3. For each node v : $d[v]$ is the length of some $v \rightarrow t$ path.

Lemma 4. For each node v : $d[v]$ is monotone non-increasing.

Lemma 5. After pass i , $d[v] \leq$ length of a shortest $v \rightarrow t$ path using $\leq i$ edges.

Pf. [by induction on i]

- Base case: $i = 0$. Assignment Project Exam Help
- Assume true after pass $i - 1$.
- Let P be any $v \rightarrow t$ path. <https://eduassistpro.github.io/>
- Let (v, w) be first edge in P and let P' be the path from w to t .
- By inductive hypothesis, at the end of pass i , $d[w] \leq c(P')$ because P' is a $w \rightarrow t$ path with $\leq i$ edges.
- After considering edge (v, w) in pass $i + 1$:

and by Lemma 4,
 $d[w]$ does not increase

$$\begin{aligned} d[v] &\leq \ell_{vw} + d[w] \\ &\leq \ell_{vw} + c(P') \\ &= \ell(P) \quad \blacksquare \end{aligned}$$

*and by Lemma 4,
 $d[v]$ does not increase*

Bellman–Ford–Moore: analysis

Theorem 2. Assuming no negative cycles, Bellman–Ford–Moore computes the lengths of the shortest $v \rightarrow t$ paths in $O(mn)$ time and $\Theta(n)$ extra space.

Pf. Lemma 2 + Lemma 5. ▀


shortest path exists and
has at most $n-1$ edges


after i passes,
 $d[v] \leq$ length of shortest path
that uses $\leq i$ edges

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Remark. Bellman–Ford–Moore AddWeChat ~~typo~~ edu_assist~~p~~ 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.



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

- A. Following $\text{successor}[v]$ pointers gives a directed $v \rightarrow t$ path.
- B. If following $\text{successor}[v]$ pointers gives a directed $v \rightarrow t$ path,
then the length of that $v \rightarrow t$ path is $d[v]$.
- C. Both A and B. <https://eduassistpro.github.io/>
- D. Neither A nor B. Add WeChat edu_assist_pro

Bellman–Ford–Moore: analysis

Claim. Throughout Bellman–Ford–Moore, following the $\text{successor}[v]$ pointers gives a directed path from v to t of length $d[v]$.

Counterexample. Claim is false!

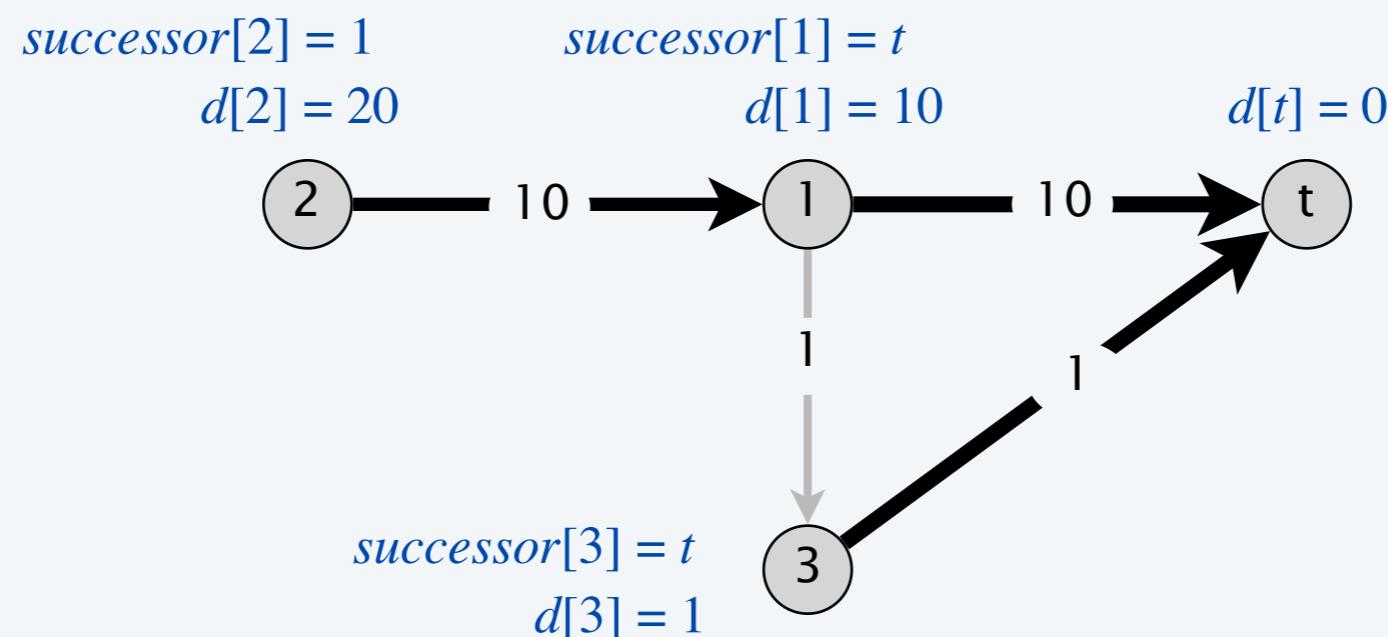
- Length of successor $v \rightarrow t$ path may be strictly shorter than $d[v]$.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

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



Bellman–Ford–Moore: analysis

Claim. Throughout Bellman–Ford–Moore, following the $\text{successor}[v]$ pointers gives a directed path from v to t of length $d[v]$.

Counterexample. Claim is false!

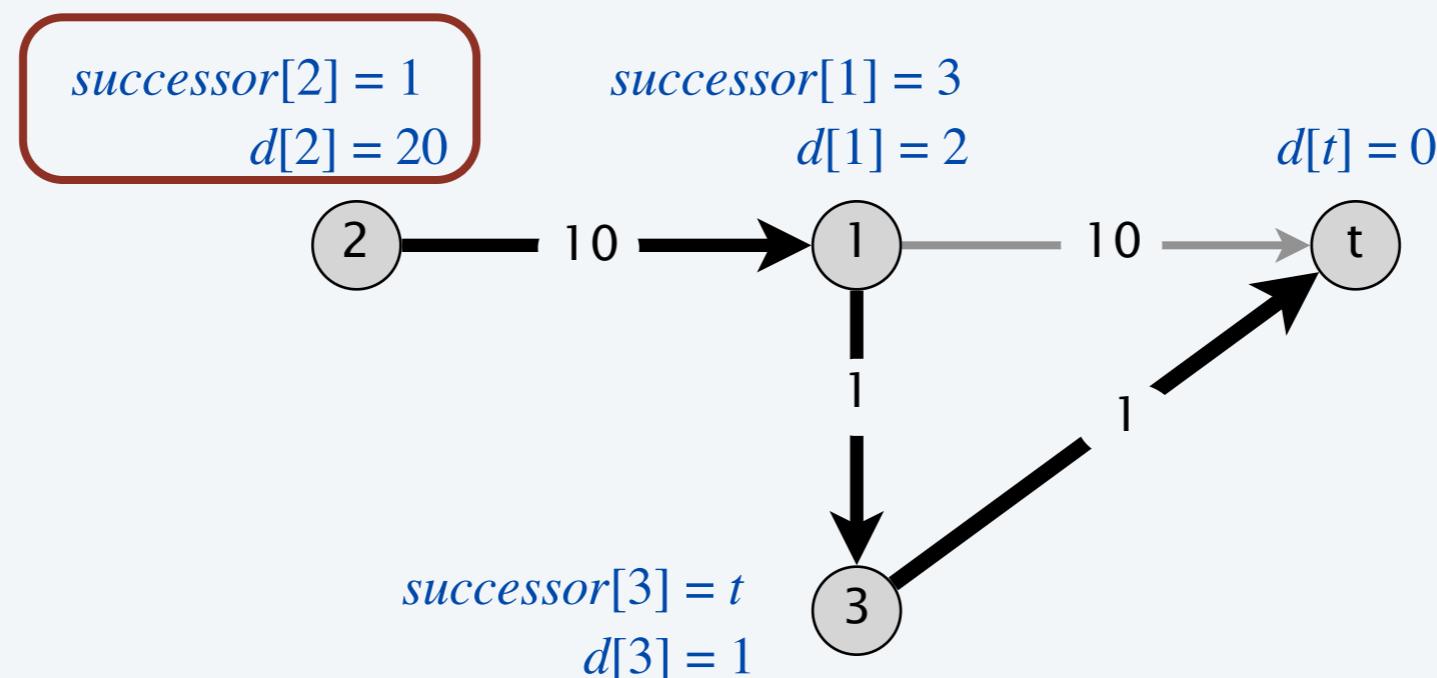
- Length of successor $v \rightarrow t$ path may be strictly shorter than $d[v]$.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

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



Bellman–Ford–Moore: analysis

Claim. Throughout Bellman–Ford–Moore, following the $\text{successor}[v]$ pointers gives a directed path from v to t of length $d[v]$.

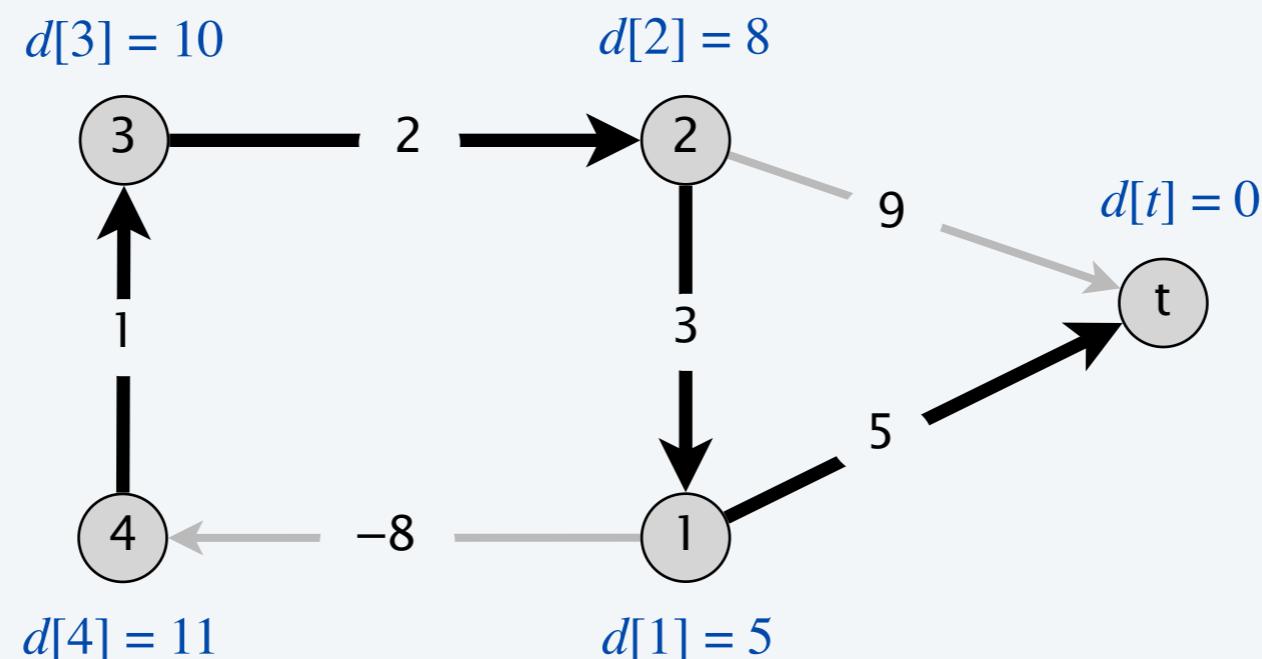
Counterexample. Claim is false!

- Length of successor $v \rightarrow t$ path may be strictly shorter than $d[v]$.
- If negative cycles succeed

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

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



Bellman–Ford–Moore: analysis

Claim. Throughout Bellman–Ford–Moore, following the $\text{successor}[v]$ pointers gives a directed path from v to t of length $d[v]$.

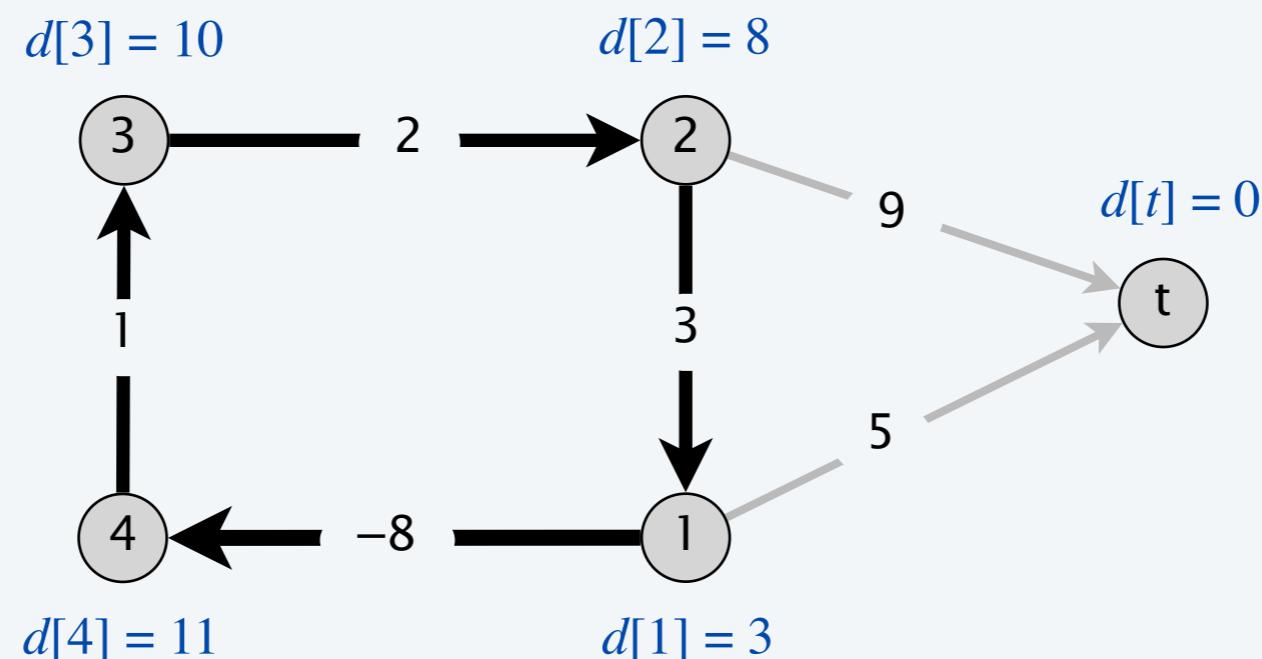
Counterexample. Claim is false!

- Length of successor $v \rightarrow t$ path may be strictly shorter than $d[v]$.
- If negative cycles succeed

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

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.

Pf.

- If $\text{successor}[v] = w$, we must have $d[v] \geq d[w] + \ell_{vw}$.
(LHS and RHS are equal when $\text{successor}[v]$ is set; $d[w]$ can only decrease;
 $d[v]$ decreases only when $\text{successor}[v]$ is reset)
- Let $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_k \rightarrow v_1$ be the sequence of nodes in a directed cycle W .

Assignment Project Exam Help

- Assume that (v_k, v_1) is the last edge added to the successor graph.

- Just prior to that: $d[v_1] \leq d[v_2]$

$$d[v_2] \geq d[v_3]$$

$$\vdots \quad \vdots \quad \vdots$$

$$d[v_{k-1}] \geq d[v_k] + \ell(v_{k-1}, v_k)$$

$$d[v_k] > d[v_1] + \ell(v_k, v_1) \quad \leftarrow \begin{array}{l} \text{holds with strict inequality} \\ \text{since we are updating } d[v_k] \end{array}$$

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

$\overbrace{\hspace{490pt}}$

W is a negative cycle

Bellman–Ford–Moore: finding the shortest paths

Theorem 3. Assuming no negative cycles, Bellman–Ford–Moore finds shortest $v \rightarrow t$ paths for every node v in $O(mn)$ time and $\Theta(n)$ extra space.

Pf.

- The successor graph cannot have a directed cycle. [Lemma 6]
- Thus, following the successor pointers from v yields a directed path to t .
- Let $v = v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_k = t$ be the nodes along this path P .
Assignment Project Exam Help
<https://eduassistpro.github.io/>
- Upon termination, if su have $d[v] = d[w] + \ell_{vw}$.
(LHS and RHS are equal et, $d[\cdot]$ did not change)
- Thus,
$$\begin{aligned} d[v_1] &= d[v_2] + \ell(v_1, v_2) \\ d[v_2] &= d[v_3] + \ell(v_2, v_3) \\ &\vdots && \vdots \\ d[v_{k-1}] &= d[v_k] + \ell(v_{k-1}, v_k) \end{aligned}$$

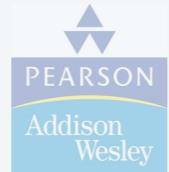
Add WeChat edu_assist_pro
since algorithm terminated
- Adding equations yields $d[v] = d[t] + \ell(v_1, v_2) + \ell(v_2, v_3) + \dots + \ell(v_{k-1}, v_k)$. ▀



Single-source shortest paths with negative weights

year	worst case	discovered by
1955	$O(n^4)$	Shimbel
1956	$O(m n^2 W)$	Ford
1958	$O(m n)$ Assignment Project Exam Help	Bellman, Moore
1983	$O(n \log(nW))$ https://eduassistpro.github.io/	Gabow
1989	$O(m n^{1/2} \log(nW))$ Add WeChat edu_assist_pro	bow–Tarjan
1993	$O(m n^{1/2} \log W)$	Goldberg
2005	$O(n^{2.38} W)$	Sankowski, Yuster–Zwick
2016	$\tilde{O}(n^{10/7} \log W)$	Cohen–Mądry–Sankowski–Vladu
20xx	???	

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



6. DYNAMIC PROGRAMMING II

- ▶ *sequence alignment*
- ▶ *Hirschberg's algorithm*

Assignment Project Exam Help
→ *Delman Ford Moore algorithm*

ctor protocols
<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

SECTION 6.9

Distance-vector routing protocols

Communication network.

- Node \approx router.
- Edge \approx direct communication link.
- Length of edge \approx latency of link. \leftarrow non-negative, but
Bellman–Ford–Moore used anyway!

Dijkstra's algorithm. Requires global information of network.
[Assignment](#) [Project](#) [Exam](#) [Help](#)

Bellman–Ford–Moore. [https://eduassistpro.github.io/
Bellman–Ford–Moore.nb.html](https://eduassistpro.github.io/Bellman-Ford-Moore.nb.html)

Add WeChat edu_assist_pro

Synchronization. We don't expect routers to run in lockstep. The order in which each edges are processed in Bellman–Ford–Moore is not important. Moreover, algorithm converges even if updates are asynchronous.

Distance-vector routing protocols

Distance-vector routing protocols. [“routing by rumor”]

- Each router maintains a vector of shortest-path lengths to every other node (distances) and the first hop on each path (directions).
- Algorithm: each router performs n separate computations, one for each potential destination node.

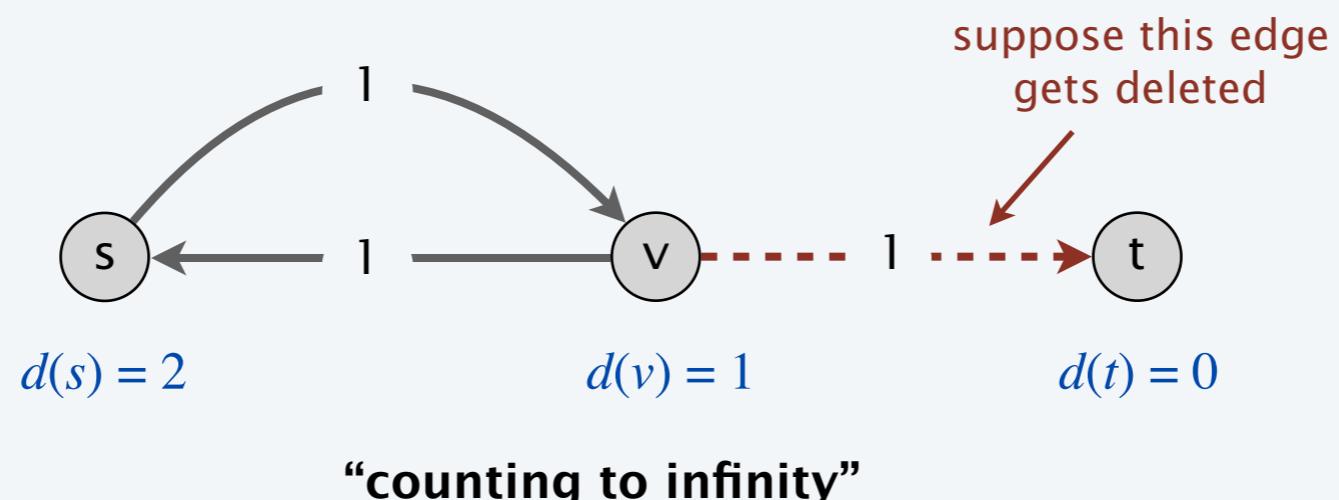
Ex. RIP, RIPv2, DEC'XNDRP, Node IV,
AppleTalk's RTMP.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Caveat. Edge lengths may **change** during algorithm (or fail completely).



Path-vector routing protocols

Link-state routing protocols.

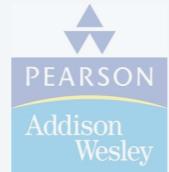
- Each router stores the whole network topology.
- Based on Dijkstra's algorithm.
- Avoids “counting-to-infinity” problem and related difficulties.
- Requires significantly more storage.

not just the distance
and first hop

Assignment Project Exam Help

Ex. BestPathGateways(OSPF) toc <https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



6. DYNAMIC PROGRAMMING II

- ▶ *sequence alignment*
- ▶ *Hirschberg's algorithm*

Assignment Project Exam Help
→ *Delman Ford Moore algorithm*

tor protocol <https://eduassistpro.github.io/>

Add WeChat edu_assist_pro
▶ neg es

SECTION 6.10

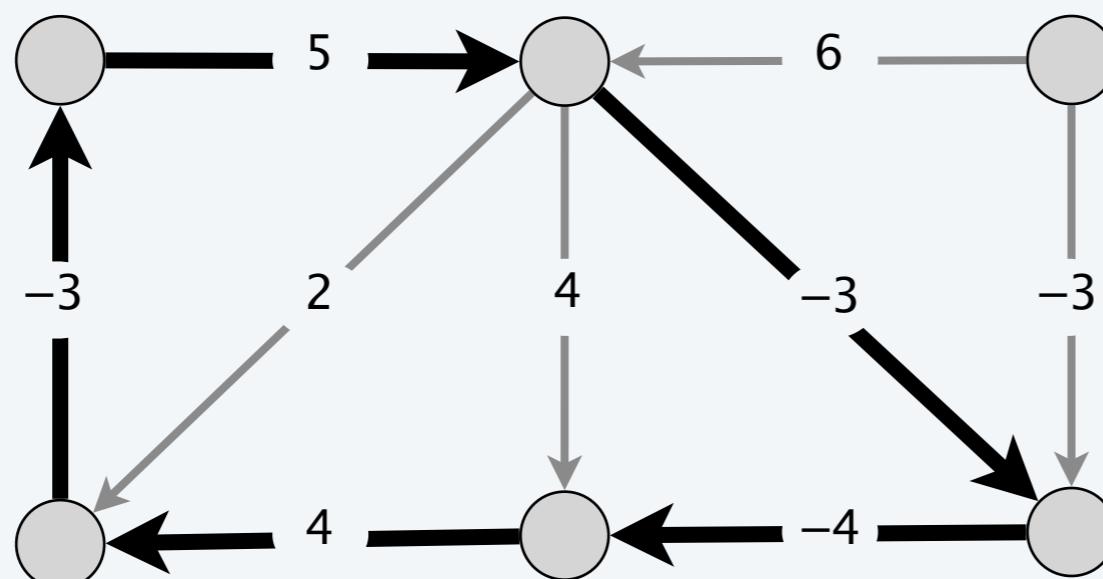
Detecting negative cycles

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

Assignment Project Exam Help

<https://eduassistpro.github.io/>

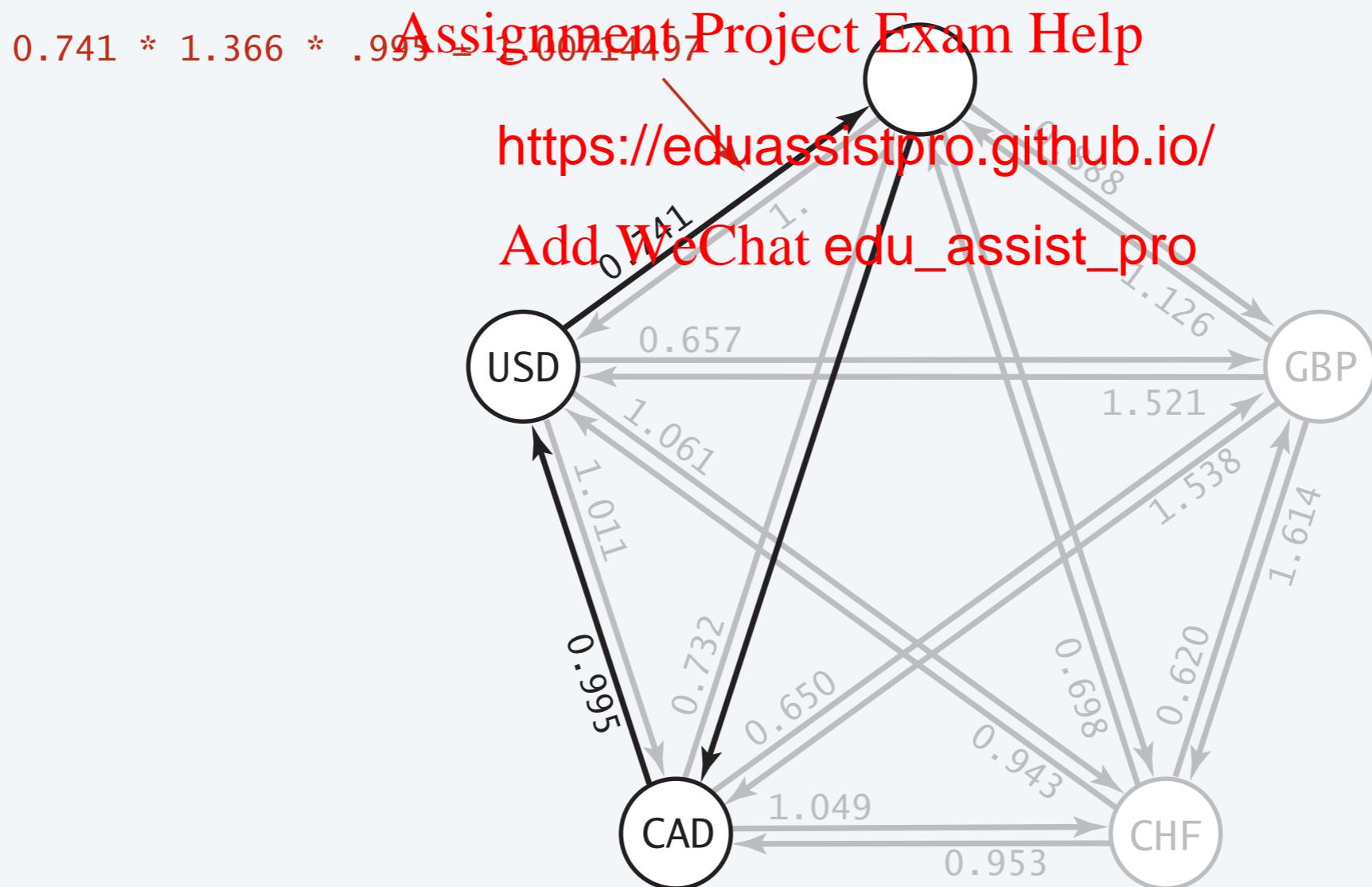
Add WeChat edu_assist_pro



Detecting negative cycles: application

Currency conversion. Given n currencies and exchange rates between pairs of currencies, is there an arbitrage opportunity?

Remark. Fastest algorithm very valuable!



Detecting negative cycles

Lemma 7. If $OPT(n, v) = OPT(n - 1, v)$ for every node v , then no negative cycles.

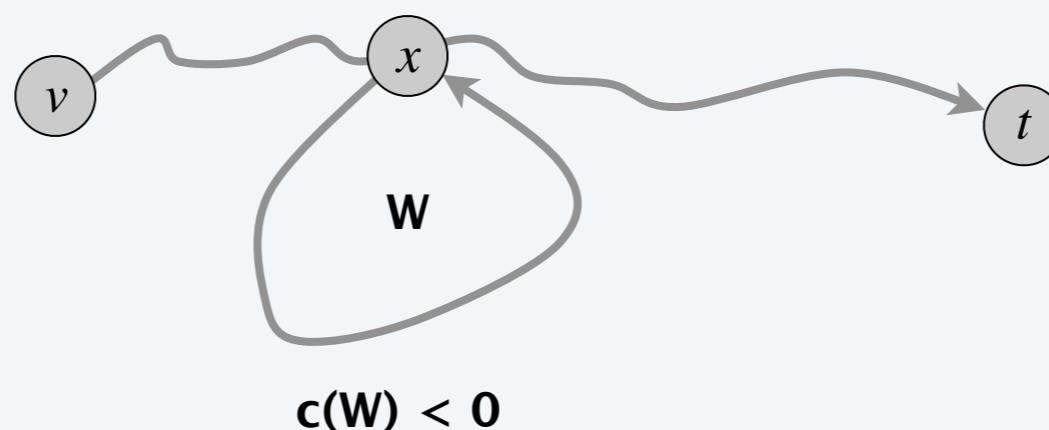
Pf. The $OPT(n, v)$ values have converged \Rightarrow shortest $v \rightarrow t$ path exists. ▀

Lemma 8. If $OPT(n, v) < OPT(n - 1, v)$ for some node v , then (any) shortest $v \rightarrow t$ path of length $\leq n$ contains a cycle W . Moreover W is a negative cycle.

Assignment Project Exam Help

Pf. [by contradiction]

- Since $OPT(n, v) < OPT(n - 1, v)$ <https://eduassistpro.github.io/> \Rightarrow shortest $v \rightarrow t$ path P has exactly n edges. Add WeChat edu_assist_pro
- By pigeonhole principle, the path P must contain a repeated node x .
- Let W be any cycle in P .
- Deleting W yields a $v \rightarrow t$ path with $< n$ edges $\Rightarrow W$ is a negative cycle. ▀



Detecting negative cycles

Theorem 4. Can find a negative cycle in $\Theta(mn)$ time and $\Theta(n^2)$ space.

Pf.

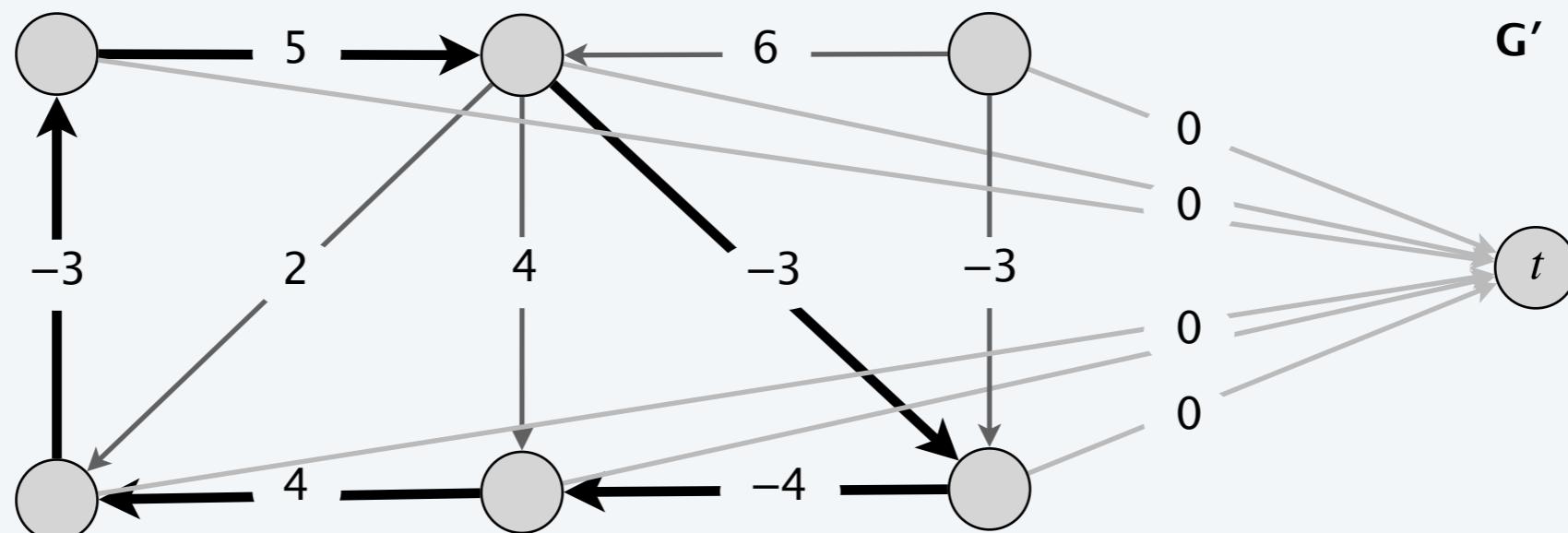
- Add new sink node t and connect all nodes to t with 0-length edge.
- G has a negative cycle iff G' has a negative cycle.
- Case 1. [$OPT(n, v) = OPT(n - 1, v)$ for every node v]
By Lemma 7, no negative cycles

Assignment Project Exam Help

- Case 2. [$OPT(n, v) < OPT(n - 1, v)$ for some node v]

Using proof of Lemma <https://eduassistpro.github.io/> $v \rightarrow t$ path.

(cycle cannot contain t since we do not add t)



Detecting negative cycles

Theorem 5. Can find a negative cycle in $O(mn)$ time and $O(n)$ extra space.

Pf.

- Run Bellman–Ford–Moore on G' for $n' = n + 1$ passes (instead of $n' - 1$).
- If no $d[v]$ values updated in pass n' , then no negative cycles.
- Otherwise, suppose $d[s]$ updated in pass n' .
- Define $\text{pass}(v) = \text{last pass in which } d[v] \text{ was updated}$.
Assignment Project Exam Help
- Observe $\text{pass}(s) = n'$ and $s(v) - 1$ for each v .
- Follow <https://eduassistpro.github.io/>
- Lemma 6 \Rightarrow the corresponding [Add WeChat edu_assist@pro](#) cycle. ■

Remark. See p. 304 for improved version and early termination rule.

(Tarjan's subtree disassembly trick)



How difficult to find a negative cycle in an undirected graph?

- A. $O(m \log n)$
- B. $O(mn)$
- C. $O(mn + n^2 \log n)$ Assignment Project Exam Help
- D. $O(n^{2.38})$ <https://eduassistpro.github.io/>
- E. No poly-time algorithm is known Add WeChat edu_assist_pro