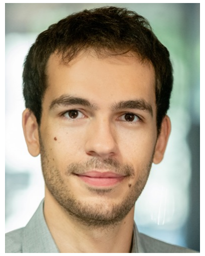
ICDE 2022 tutorial



Toward Responsive DBMS: Optimal Join Algorithms, Enumeration, Factorization, Ranking, and Dynamic Programming

Nikolaos Tziavelis, Wolfgang Gatterbauer, Mirek Riedewald

Northeastern University, Boston

Part 5: Dynamic programming & Semirings

Northeastern University
Khoury College of Computer Sciences

Slides: https://northeastern-datalab.github.io/responsive-dbms-tutorial

DOI: https://doi.org/10.1109/ICDE53745.2022.00299

Data Lab: https://db.khoury.northeastern.edu

DATALAB @Northeastern

This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 4.0 International License.
See https://creativecommons.org/licenses/by-nc-sa/4.0/ for details

Outline tutorial

1: Introduction (Nikos) ~40min

2: Tree Decompositions (Mirek) ~20min

3: Acyclic Queries & Enumeration (Wolfgang) ~25min

BREAK

4: Factorization (Nikos) ~15min

5: Dynamic Programming & Semirings (Wolfgang) ~20min

6: Any-k or Ranked Enumeration (Nikos) ~35min

7. Decomposition of Comparison Predicates (Mirek) ~10min

8. Conclusion (Mirek) ~5min

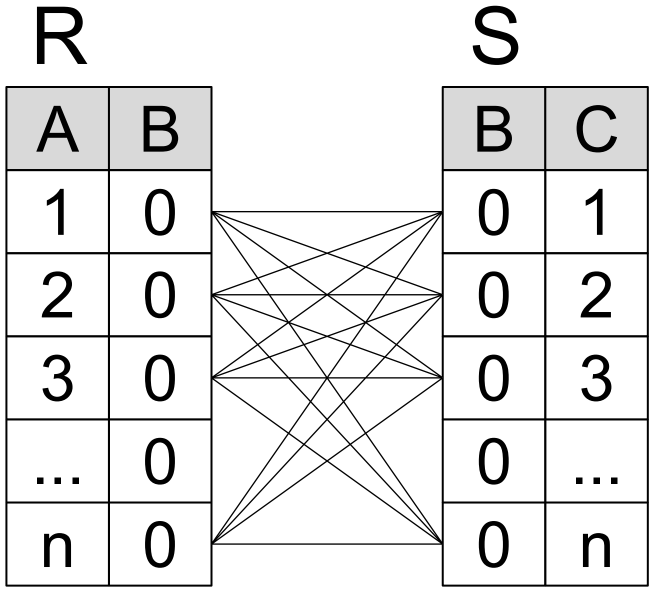
Outline Part 5

Part 5: Dynamic Programming & Semirings (Wolfgang) ~20min

- Top-1 = Dynamic Programming (DP)
- Top-1 Yannakakis as variant of Tree-DP
- Algebra: Totally Ordered Commutative Monoids

Evaluating Top-1 result with a modern DBMS

$$P_2(x, y, z) :- R(x, y), S(y, z)$$



R

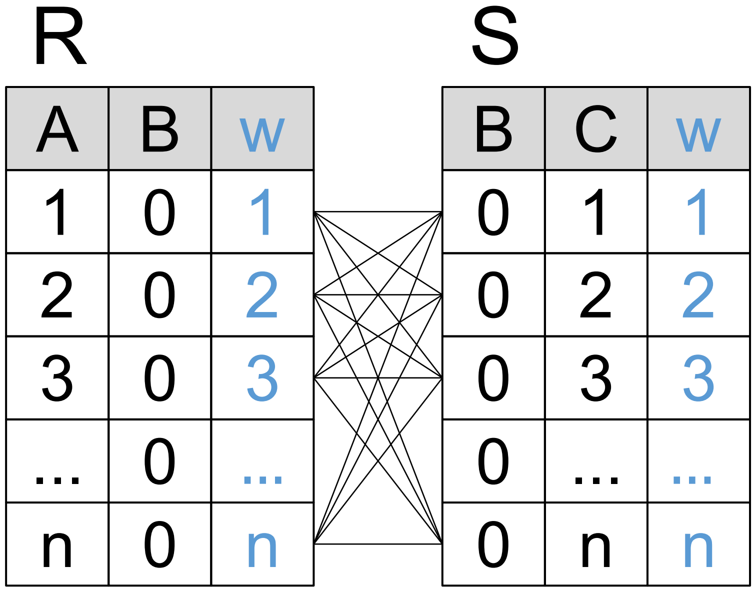
A	B
1	0
2	0
3	0
...	0
n	0

S

B	C
0	1
0	2
0	3
0	...
0	n

Evaluating Top-1 result with a modern DBMS

$$P_2(x, y, z) :- R(x, y), S(y, z)$$



R

A	B	w
1	0	1
2	0	2
3	0	3
…	0	…
n	0	n

S

B	C	w
0	1	1
0	2	2
0	3	3
0	…	…
0	n	n

Adding weights to each tuple

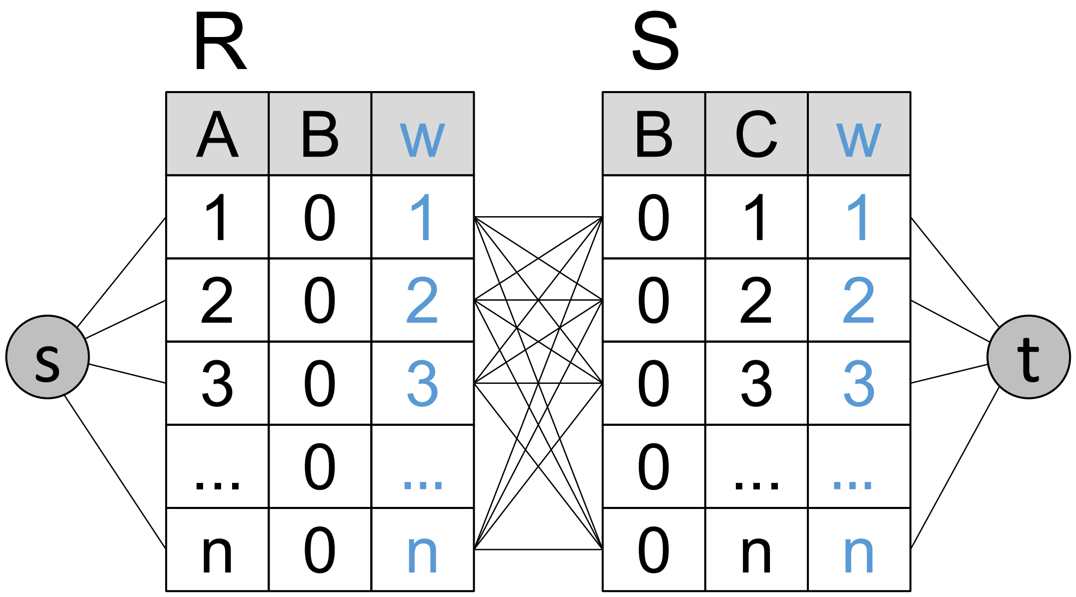
```
select R.A, R.B, S.C,
       R.w + S.w as weight
from   R, S
where  R.B=S.B
order by weight asc
limit  1
```

Q: Find the lightest joining pair

?

Evaluating Top-1 result with a modern DBMS

$$P_2(x,y,z) :- R(x,y), S(y,z)$$



A	B	w
1	0	1
2	0	2
3	0	3
…	0	…
n	0	n

B	C	w
0	1	1
0	2	2
0	3	3
0	…	…
0	n	n

```
select R.A, R.B, S.C,
       R.w + S.w as weight
from   R, S
where  R.B=S.B
order by weight asc
limit  1
```

Q: Find the lightest joining pair

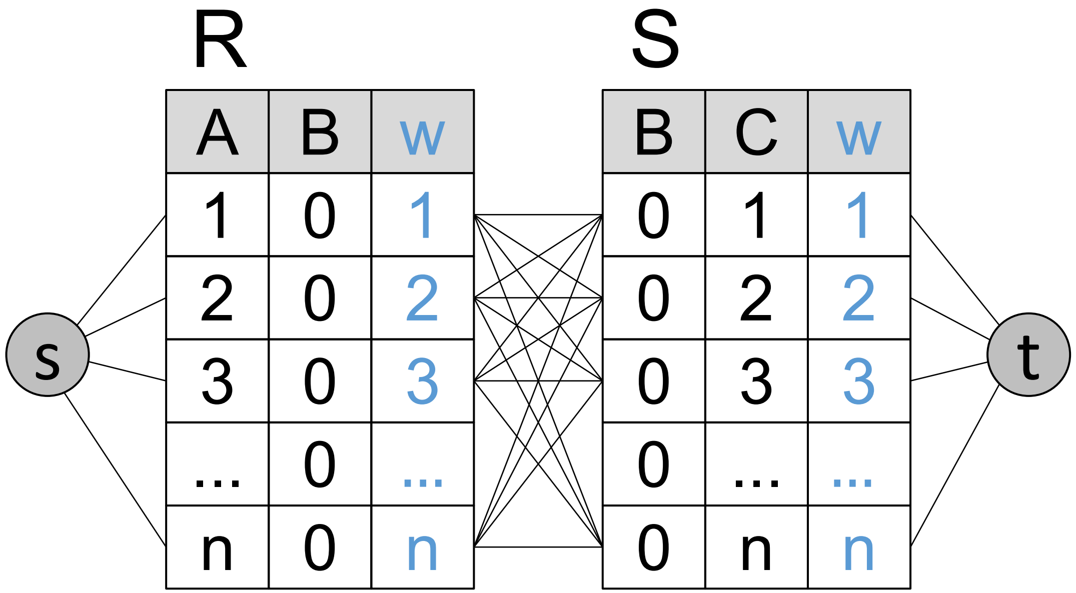
?

Adding weights to each tuple

Basically a shortest path calculation
(from source s to target t)

Evaluating Top-1 result with a modern DBMS

$$P_2(x,y,z) :- R(x,y), S(y,z)$$



R

A	B	w
1	0	1
2	0	2
3	0	3
…	0	…
n	0	n

S

B	C	w
0	1	1
0	2	2
0	3	3
0	…	…
0	n	n

Adding weights to each tuple

Basically a shortest path calculation (from source s to target t)

```
select  R.A, R.B, S.C,
        R.w + S.w as weight
from    R, S
where   R.B=S.B
order by weight asc
limit   1
```

Q: Find the lightest joining pair

O(n^2) possible pairs, but only 1 lightest.

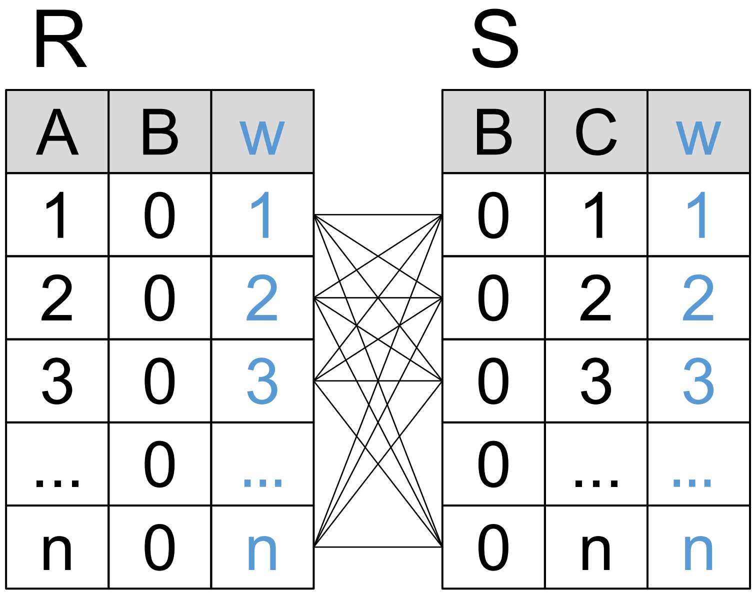
Problem: DBMS first calculates all n^2 pairs, then picks the top-1 ☹

Result

A	B	C	weight
1	0	1	2
1	0	2	3
2	0	1	3
3	0	1	4
2	0	2	4
1	0	3	4
…	…	…	…
n	0	n	n+n

Evaluating Top-1 result with Postgres

$$P_2(x,y,z) :- R(x,y), S(y,z)$$



R

A	B	w
1	0	1
2	0	2
3	0	3
…	0	…
n	0	n

S

B	C	w
0	1	1
0	2	2
0	3	3
0	…	…
0	n	n

```
---------------------------
-- Query 1
---------------------------
SELECT    A, R.B, S.C,
             R.W + S.W as weight
FROM      R, S
WHERE    R.B=S.B
ORDER BY weight ASC
LIMIT    1;
```

$n= 1{,}000$: $t_{Q1}= 0.22$ sec

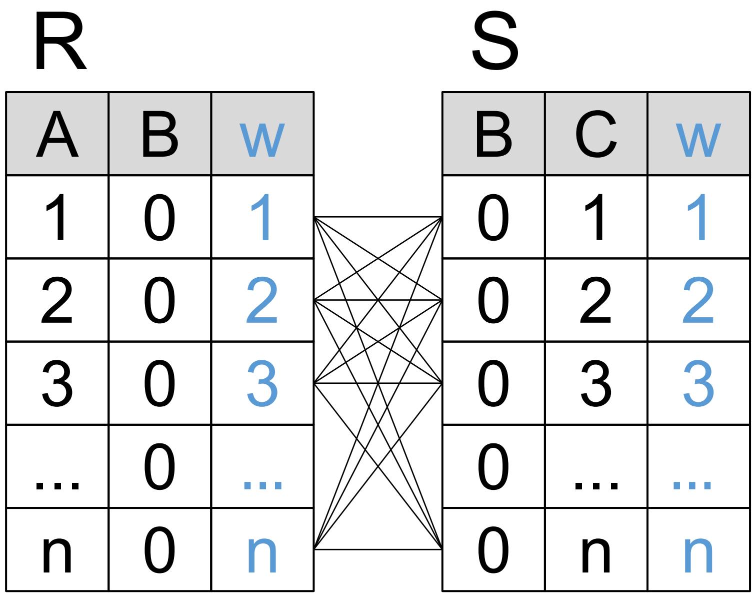
$n=10{,}000$: $t_{Q1}=$ 22 sec $O(n^2)$ ☹

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Towards Responsive DBMS. ICDE 2022 tutorial: https://northeastern-datalab.github.io/responsive-dbms-tutorial

Evaluating Top-1 result with Postgres

$$P_2(x,y,z) :- R(x,y), S(y,z)$$



R

A	B	w
1	0	1
2	0	2
3	0	3
...	0	...
n	0	n

S

B	C	w
0	1	1
0	2	2
0	3	3
0	...	...
0	n	n

Maximal intermediate result size is $O(n)$ ☺

Dynamic programming!

```
---------------------------
-- Query 1
---------------------------
SELECT   A, R.B, S.C,
            R.W + S.W as weight
FROM     R, S
WHERE   R.B=S.B
ORDER BY weight ASC
LIMIT   1;
```

```
---------------------------
-- Query 2
---------------------------
SELECT R.A, X.B, S.C, X.W as weight
FROM R, S,
    (SELECT T1.B, W1, W2, W1+W2 W
    FROM
        (SELECT B, MIN(W) W1
        FROM R
        GROUP BY B) T1,
        (SELECT B, MIN(W) W2
        FROM S
        GROUP BY B) T2
    WHERE T1.B = T2.B
    ORDER BY W ASC
    LIMIT 1) X
WHERE X.B = R.B
AND X.W1 = R.W
AND X.B = S.B
AND X.W2 = S.W
LIMIT   1;
```

n= 1,000: t_{Q1}= 0.22 sec t_{Q2}= 1 msec

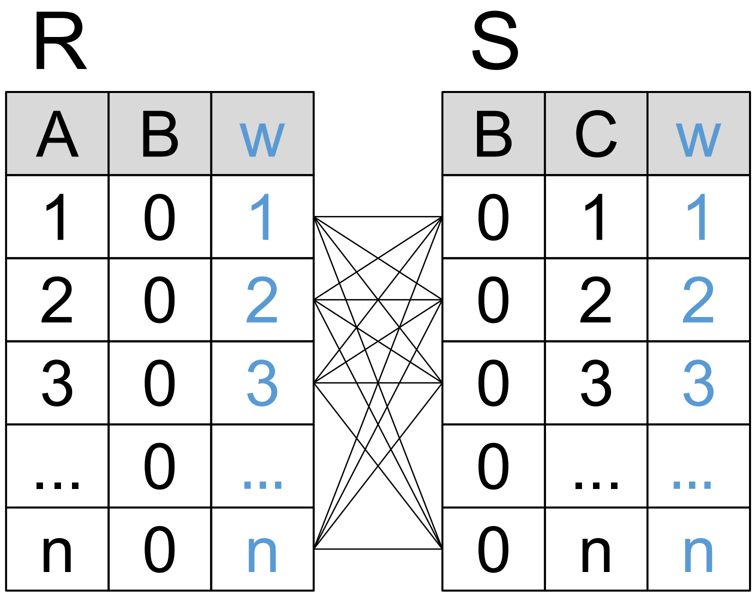
n=10,000: t_{Q1}= 22 sec $O(n^2)$ ☹ t_{Q2}= 4 msec $O(n)$ ☺

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Towards Responsive DBMS. ICDE 2022 tutorial: https://northeastern-datalab.github.io/responsive-dbms-tutorial

Evaluating Top-1 value with Postgres

$$P_2(x,y,z) :- R(x,y), S(y,z)$$



R

A	B	w
1	0	1
2	0	2
3	0	3
…	0	…
n	0	n

S

B	C	w
0	1	1
0	2	2
0	3	3
0	…	…
0	n	n

Maximal intermediate result size is O(n) ☺

Dynamic programming!

```
---------------------------
-- Query 1
---------------------------
SELECT min(R.W + S.W) as weight
INTO record1
FROM R, S
WHERE R.B=S.B;
```

```
----------------------------
-- Query 2
----------------------------
SELECT min(W1+W2) as weight
INTO record2
FROM
    (SELECT B, MIN(W) W1
    FROM R
    GROUP BY B) T1,
    (SELECT B, MIN(W) W2
    FROM S
    GROUP BY B) T2
WHERE T1.B = T2.B;
```

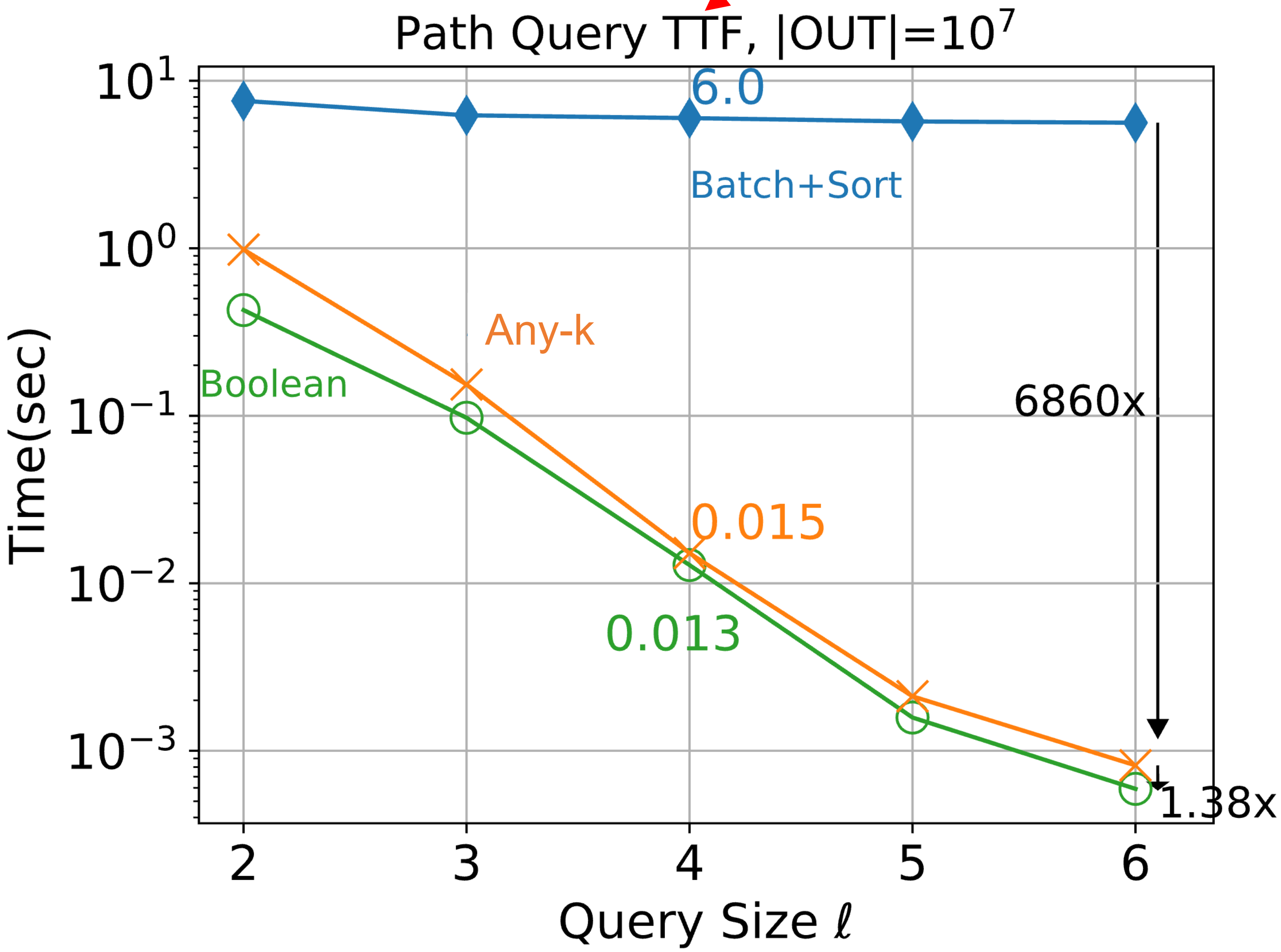
n= 1,000: t_{Q1}= 0.1 sec t_{Q2}< 1 msec

n=10,000: t_{Q1}= 9.4 sec $O(n^2)$ ☹ t_{Q2}= 3 msec $O(n)$ ☺

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql
Towards Responsive DBMS. ICDE 2022 tutorial: https://northeastern-datalab.github.io/responsive-dbms-tutorial

Any-*k*: Faster and more versatile than Top-*k* today

TTF = Time-To-First = Top-1



Path query with constant size output and increasing query size

Source: Tziavelis, Ajwani, Gatterbauer, Riedewald, Yang. Optimal Algorithms for Ranked Enumeration of Answers to Full Conjunctive Queries. PVLDB 2020. http://www.vldb.org/pvldb/vol13/p1582-tziavelis.pdf. Code to reproduce experiments linked from project page: https://northeastern-datalab.github.io/anyk/
Towards Responsive DBMS. ICDE 2022 tutorial: https://northeastern-datalab.github.io/responsive-dbms-tutorial

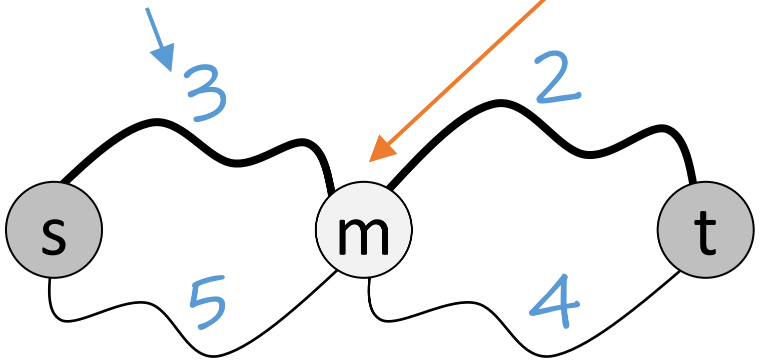
Two slightly different definitions of Dynamic Programming

1. Principle of optimality* in mathematical optimization: "*There is an optimal policy s.t. irrespective of the initial state and decision, the remaining decisions are chosen optimally.*"

2. A more general algorithm that solves problems with optimal substructure: "*Break a complicated problem into simpler sub-problems, and then solve them recursively bottom-up.*"

first decision (edge) with cost 3

intermediate state (node)



Bellman equation: Edges or transitions mean that state j can be reached from state i

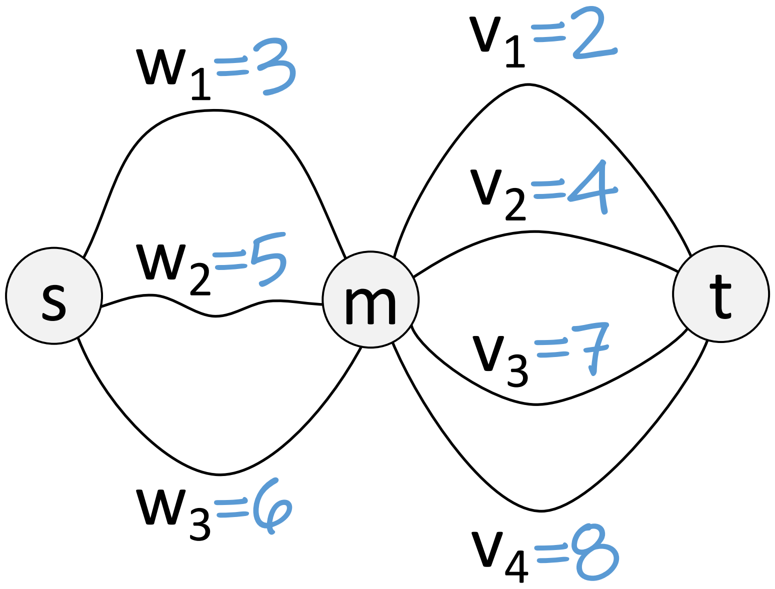
$$J^*(i) = \min\{w(i,j) + J^*(j) \mid (i,j) \in E\}$$

backwards reasoning or "bottom-up"

* Definition slightly modified from originally "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision" by "Bellman. Dynamic Programming. Princeton University Press 1957, https://dl.acm.org/doi/10.5555/862270" due to details discussed in part 6 and by "Morin. Monotonicity and the Principle of Optimality. JMAA 1982, https://doi.org/10.1016/0022-247X(82)90223-2"

1. Dynamic programming for mathematical optimization

weights or cost

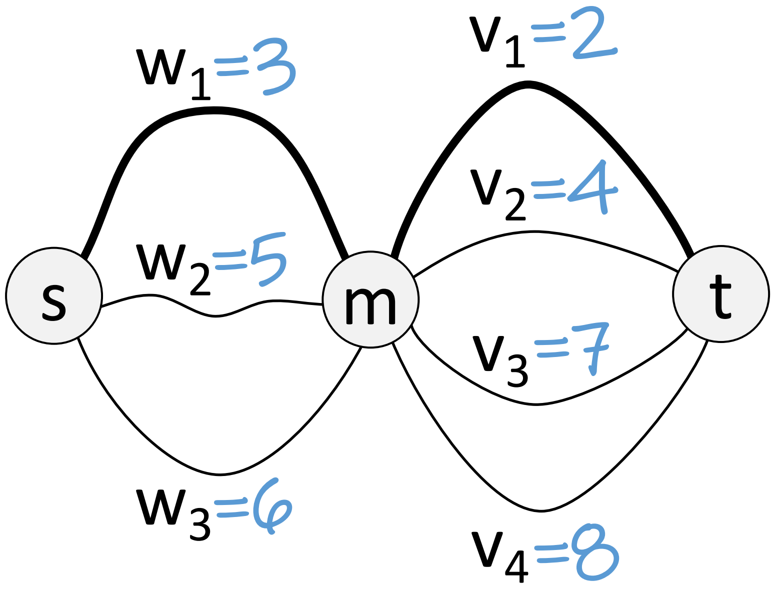


What is the shortest path from s to t?

?

1. Dynamic programming for mathematical optimization

weights or cost



Principle of optimality from Dynamic Programming: *irrespective of the initial state and decision, an optimal solution continues optimally from the resulting state*

$$\min\{w_1+v_1,\ w_1+v_2,\ w_1+v_3,\ w_1+v_4,\ \ldots,\ w_3+v_4\}$$

$$\min\{3+2,\ 3+4,\ 3+7,\ 3+8,\ \ldots,\ 6+8\}$$

$$= \min\{w_1, w_2, w_3\} + \min\{v_1, v_2, v_3, v_4\}$$

$$\min\{3, 5, 6\} + \min\{2, 4, 7, 8\}$$

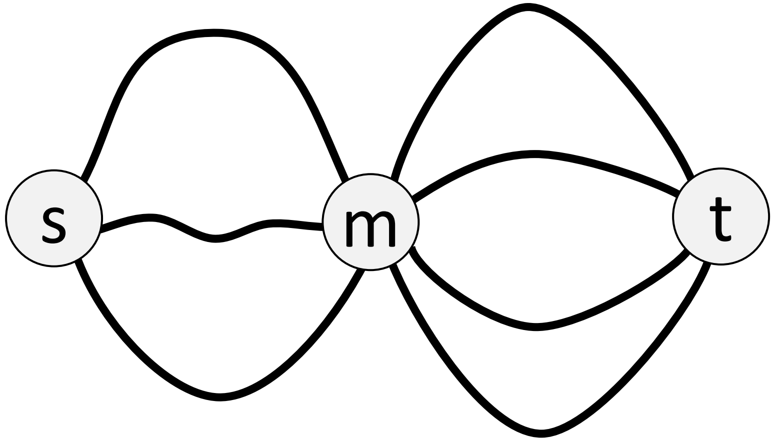
What is the shortest path from s to t?

Answer: 5 = 3 + 2

Distributivity law: + distributes over min

$\min\{x,y\}+z = \min\{x+z, y+z\}$

2. Dynamic programming as using optimal substructure

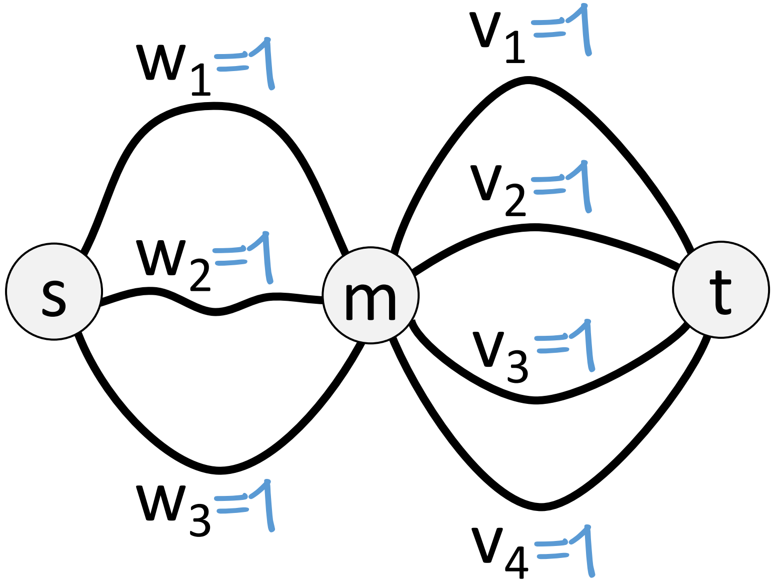


How many paths are there from s to t?

?

2. Dynamic programming as using optimal substructure

The more general algebraic structure behind these two examples are "semirings" (more on that later). On a high-level, we just apply the distributivity law.



$$\text{sum}\{w_1 \cdot v_1, w_1 \cdot v_2, w_1 \cdot v_3, w_1 \cdot v_4, \ldots, w_3 \cdot v_4\}$$

$$\text{sum}\{1 \cdot 1,\ 1 \cdot 1,\ 1 \cdot 1,\ 1 \cdot 1, \ldots,\ 1 \cdot 1\}$$

$$= \text{sum}\{w_1, w_2, w_3\} \cdot \text{sum}\{v_1, v_2, v_3, v_4\}$$

$$\text{sum}\{1,\ 1,\ 1\} \cdot \text{sum}\{1,\ 1,\ 1,\ 1\}$$

How many paths are there from s to t?

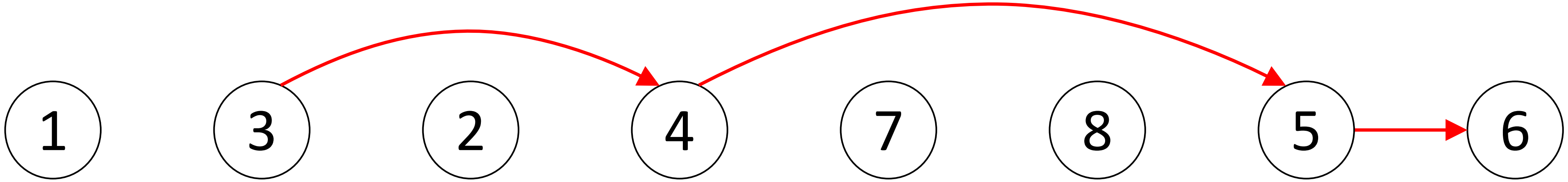
Answer: 12 = 3 · 4

Distributivity law: ⊗ distributes over ⊕

$(x \oplus y) \otimes z = x \otimes z \oplus y \otimes z$

3. Find a "Longest Increasing Subsequence" with DP

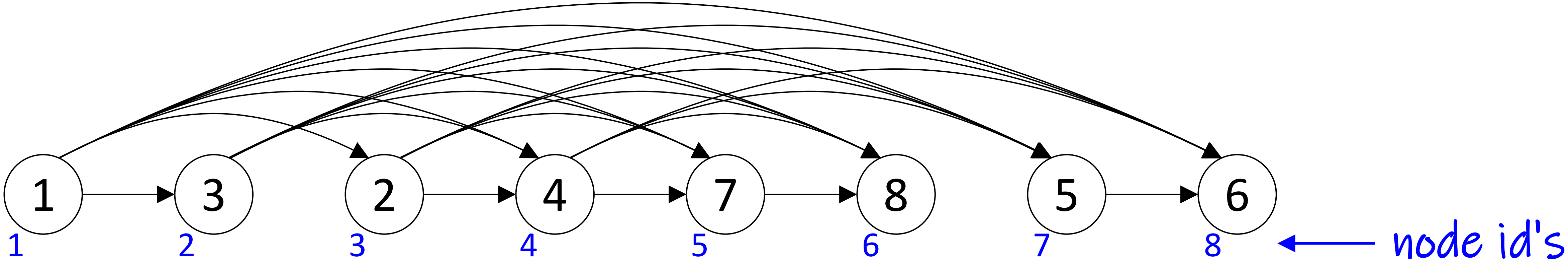
In an increasing subsequence the numbers are getting strictly larger.



How do we find a longest increasing subsequence ?

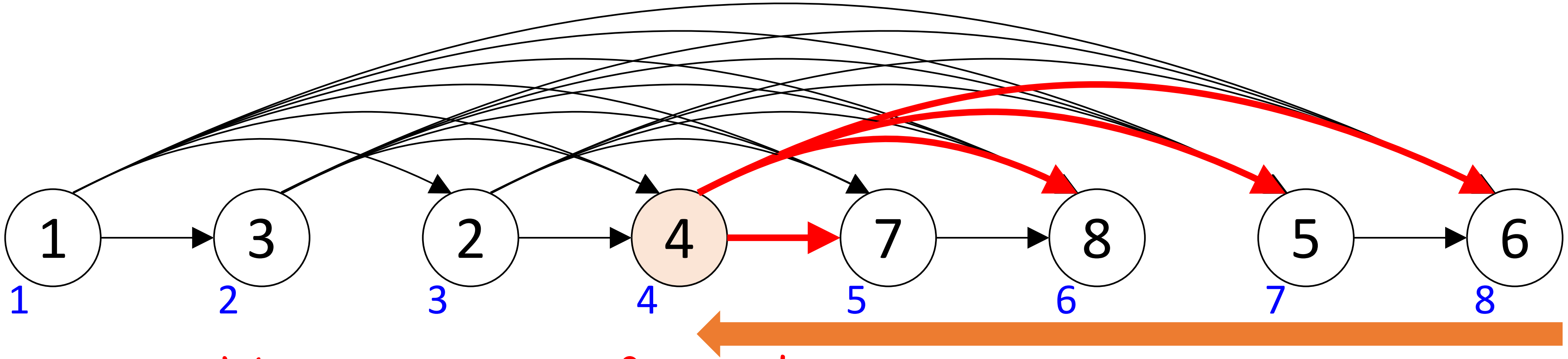
3. Find a "Longest Increasing Subsequence" with DP

Edges show permissible transitions



3. Find a "Longest Increasing Subsequence" with DP

Edges show permissible transitions



Permissible transitions for node 4: {5:2, 7:2, 6:1, 8:1, ⊥:0}

next state id: additional length

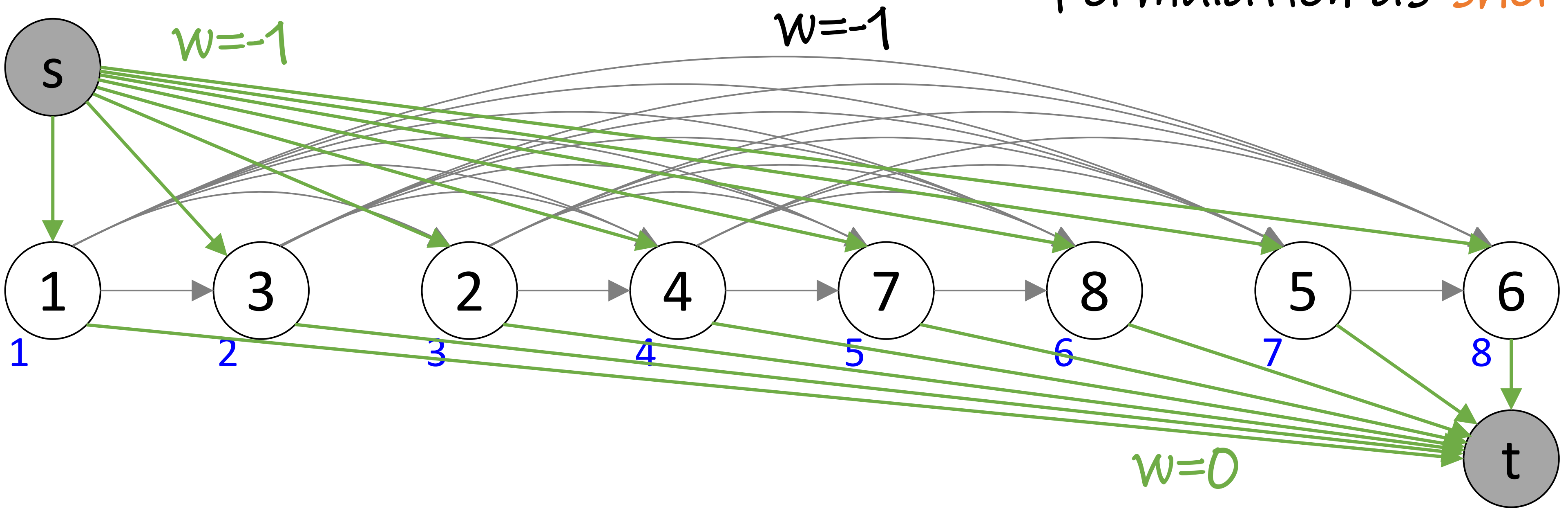
DP formulation:

```
for $i = n, \dots, 2, 1$:
  $L(i) = 1 + \max\{L(j) \mid (i, j) \in E\}$
return $\max\{L(i) \mid i \in [n]\}$
```

Can we also formulate it as shortest path problem ?

3. Find a "Longest Increasing Subsequence" with DP

Formulation as shortest path problem



DP formulation:

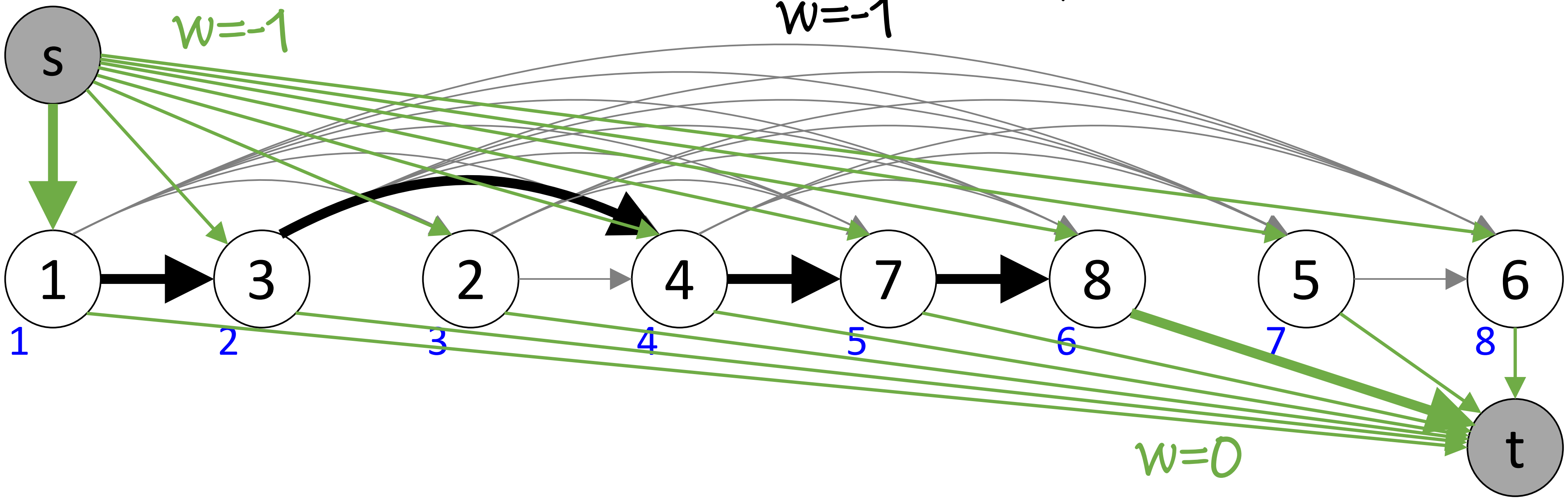
```
for $i = n, \dots, 2, 1$:
  $L(i) = 1 + \max\{L(j) \mid (i,j) \in E\}$
return $\max\{L(i) \mid i \in [n]\}$
```

As shortest path problem:

```
$d(t) = 0$
for $i = n, \dots, 2, 1, s$:
  $d(i) = \min\{w(i,j) + d(j) \mid (i,j) \in E\}$
return $-d(s)$
```

3. Find a "Longest Increasing Subsequence" with DP

Formulation as shortest path problem



$$L(1) = -d(s) = 5$$

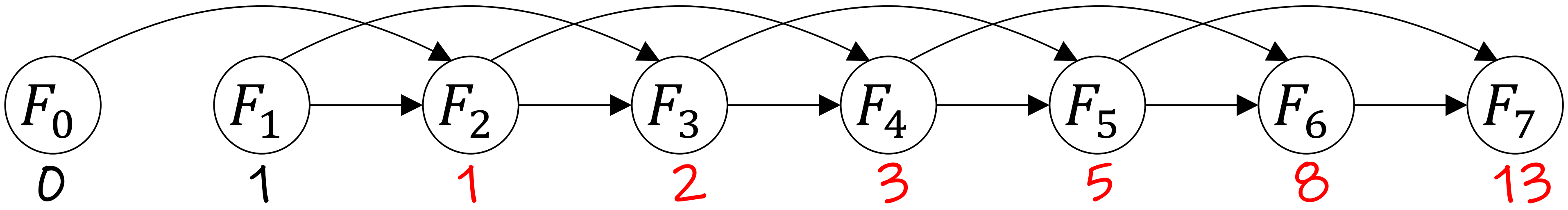
DP formulation:

```
for $i = n, \dots, 2, 1$:
  $L(i) = 1 + \max\{L(j) \mid (i, j) \in E\}$
return $\max\{L(i) \mid i \in [n]\}$
```

As shortest path problem:

```
$d(t) = 0$
for $i = n, \dots, 2, 1, s$:
  $d(i) = \min\{w(i, j) + d(j) \mid (i, j) \in E\}$
return $-d(s)$
```

4. Calculating Fibonacci numbers with DP



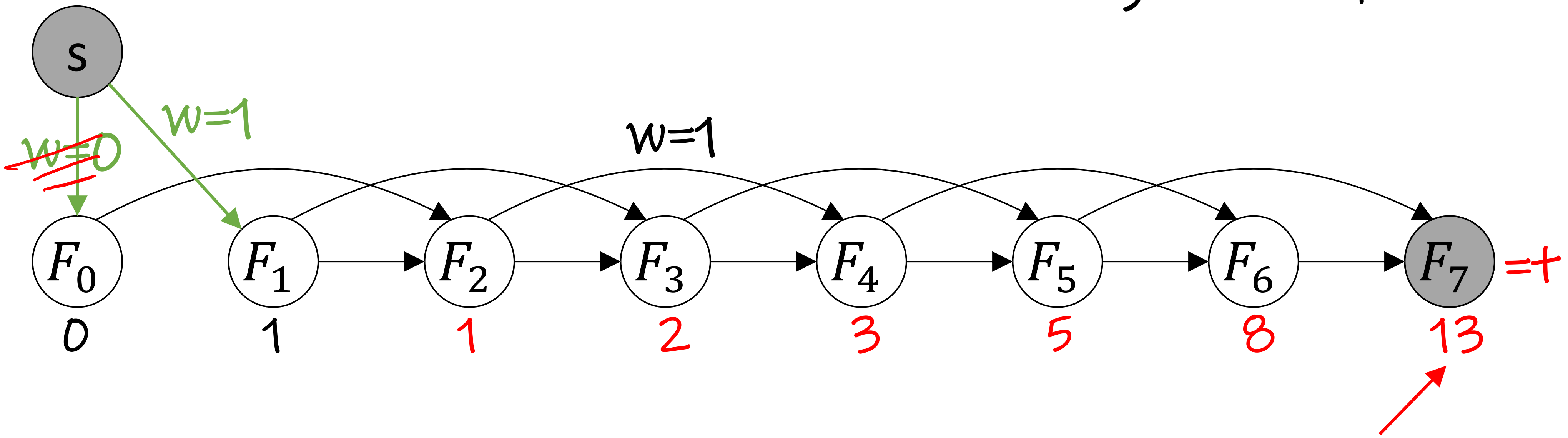
DP formulation:

```
$F_0 = 0; F_1 = 1$
for $i = 2, 3, \ldots, n$:
  $F_i = F_{i-1} + F_{i-2}$
return $F_n$
```

Can we also formulate it as path counting problem ?

4. Calculating Fibonacci numbers with DP

Edges show permissible transitions



13 paths from s to t=F_7

DP formulation:

```
$F_0 = 0; F_1 = 1$
for $i = 2, 3, \ldots, n$:
  $F_i = F_{i-1} + F_{i-2}$
return $F_n$
```

As path counting problem:

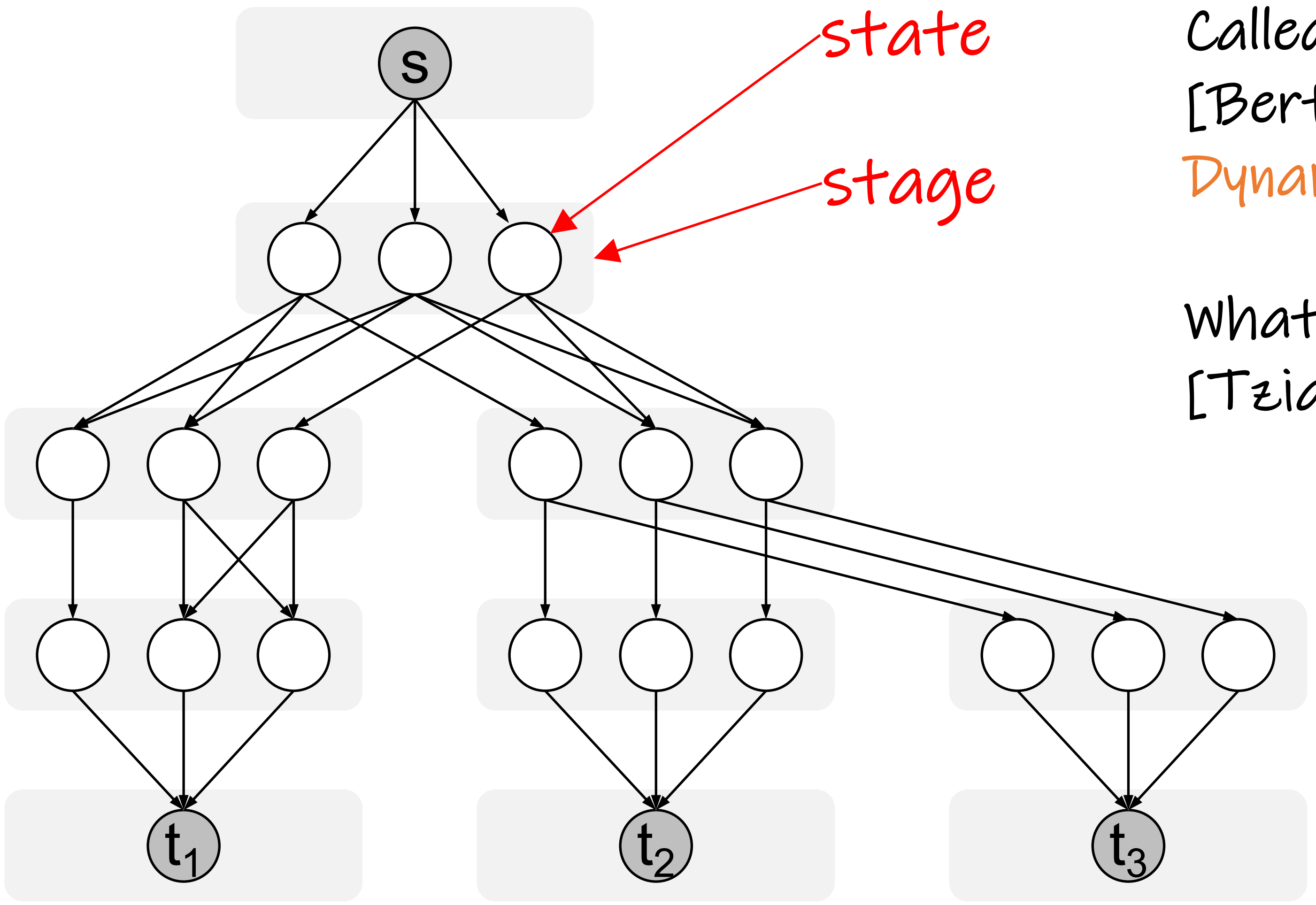
```
$F_0 = 0; F_1 = 1$
for $j = 2, 3, \ldots, n$:
  $F_j = \text{sum}\{w(i, j) \cdot F(i) \mid (i, j) \in E\}$
return $F_n$
```

Outline Part 5

Part 5: Dynamic Programming & Semirings (Wolfgang) ~20min

- Top-1 = Dynamic Programming (DP)
- Top-1 Yannakakis as variant of Tree-DP
- Algebra: Totally Ordered Commutative Monoids

Tree-DP = instance of Non-Serial DP (NSDP)



Called "diverging branch structure" by [Bertele'72] as one instance of Non-Serial Dynamic Programming (NSDP).

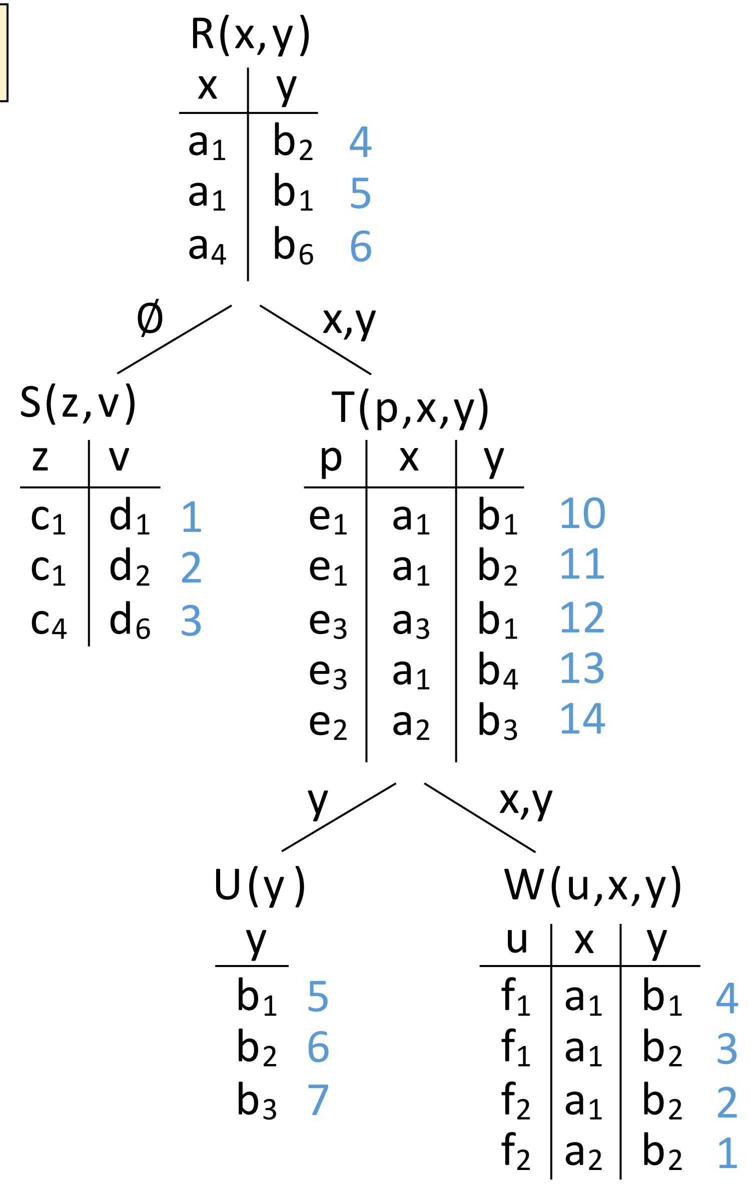
What we will refer to as "Tree-DP" [Tziavelis'20]

[Bertele'72] Bertele, Brioschi. Nonserial Dynamic Programming. Academic Press. https://dl.acm.org/doi/10.5555/578817. [Tziavelis'20] Tziavelis, Ajwani, Gatterbauer, Riedewald, Yang. Optimal Algorithms for Ranked Enumeration of Answers to Full Conjunctive Queries. PVLDB 2020. http://www.vldb.org/pvldb/vol13/p1582-tziavelis.pdf

Yannakakis Algorithm – Example

631

Q(x,y,z,v,p,u) :- R(x,y), S(z,v), T(p,x,y), U(y), W(u,x,y).



R(x,y)

x	y	
a_1	b_2	4
a_1	b_1	5
a_4	b_6	6

S(z,v)

z	v	
c_1	d_1	1
c_1	d_2	2
c_4	d_6	3

T(p,x,y)

p	x	y	
e_1	a_1	b_1	10
e_1	a_1	b_2	11
e_3	a_3	b_1	12
e_3	a_1	b_4	13
e_2	a_2	b_3	14

U(y)

y	
b_1	5
b_2	6
b_3	7

W(u,x,y)

u	x	y	
f_1	a_1	b_1	4
f_1	a_1	b_2	3
f_2	a_1	b_2	2
f_2	a_2	b_2	1

Each tuple now has a weight.

New goal: Find the "lightest join" (minimum sum of weights)

?

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Towards Responsive DBMS. ICDE 2022 tutorial: https://northeastern-datalab.github.io/responsive-dbms-tutorial

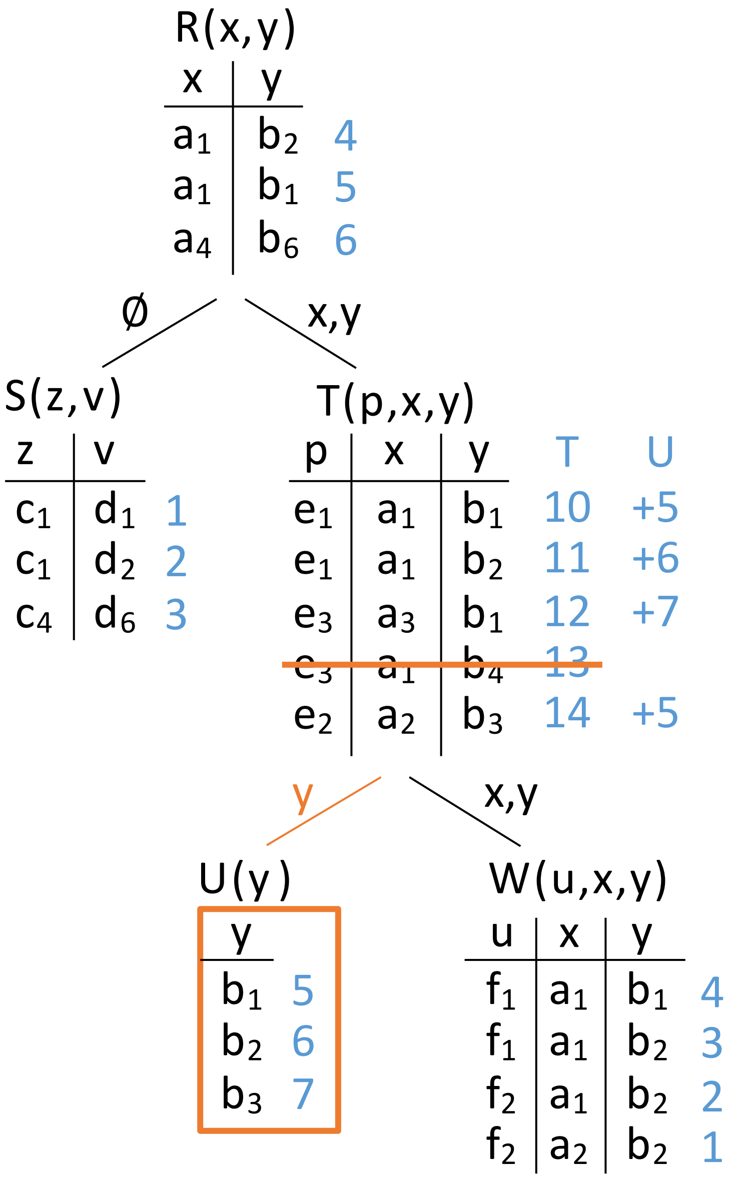
Yannakakis Algorithm – Example

Q(x,y,z,v,p,u) :- R(x,y), S(z,v), T(p,x,y), U(y), W(u,x,y).

1. Adapted bottom-up semi-join propagation in $O(|\text{input}|)$.
   Finds the minimum value

Each tuple now has a weight.

New goal: Find the "lightest join"
(minimum sum of weights)



R(x,y)

x	y	
a_1	b_2	4
a_1	b_1	5
a_4	b_6	6

S(z,v)

z	v	
c_1	d_1	1
c_1	d_2	2
c_4	d_6	3

T(p,x,y)

p	x	y	T	U
e_1	a_1	b_1	10	+5
e_1	a_1	b_2	11	+6
e_3	a_3	b_1	12	+7
~~e_3~~	~~a_1~~	~~b_4~~	~~13~~	
e_2	a_2	b_3	14	+5

U(y)

y	
b_1	5
b_2	6
b_3	7

W(u,x,y)

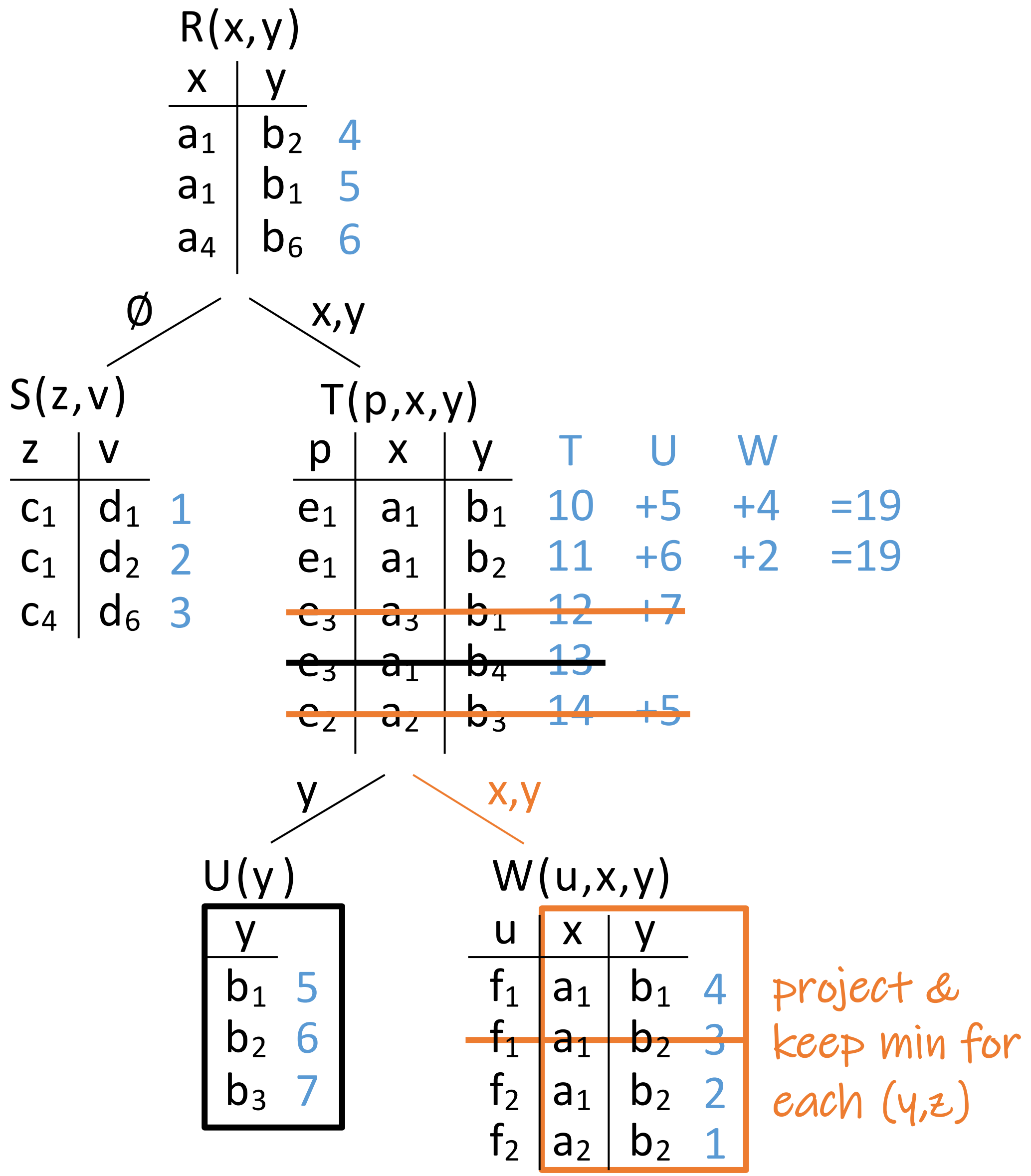
u	x	y	
f_1	a_1	b_1	4
f_1	a_1	b_2	3
f_2	a_1	b_2	2
f_2	a_2	b_2	1

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql
Towards Responsive DBMS. ICDE 2022 tutorial: https://northeastern-datalab.github.io/responsive-dbms-tutorial

Yannakakis Algorithm – Example

Q(x,y,z,v,p,u) :- R(x,y), S(z,v), T(p,x,y), U(y), W(u,x,y).

1. Adapted bottom-up semi-join propagation in $O(|\text{input}|)$.
   Finds the minimum value



R(x,y)

x	y	
a_1	b_2	4
a_1	b_1	5
a_4	b_6	6

S(z,v)

z	v	
c_1	d_1	1
c_1	d_2	2
c_4	d_6	3

T(p,x,y)

p	x	y	T	U	W	
e_1	a_1	b_1	10	+5	+4	=19
e_1	a_1	b_2	11	+6	+2	=19
~~e_3~~	~~a_3~~	~~b_1~~	~~12~~	~~+7~~		
~~e_3~~	~~a_1~~	~~b_4~~	~~13~~			
~~e_2~~	~~a_2~~	~~b_3~~	~~14~~	~~+5~~		

U(y)

y	
b_1	5
b_2	6
b_3	7

W(u,x,y)

u	x	y	
f_1	a_1	b_1	4
~~f_1~~	~~a_1~~	~~b_2~~	~~3~~
f_2	a_1	b_2	2
f_2	a_2	b_2	1

project & keep min for each (y,z)

Each tuple now has a weight.

New goal: Find the "lightest join" (minimum sum of weights)

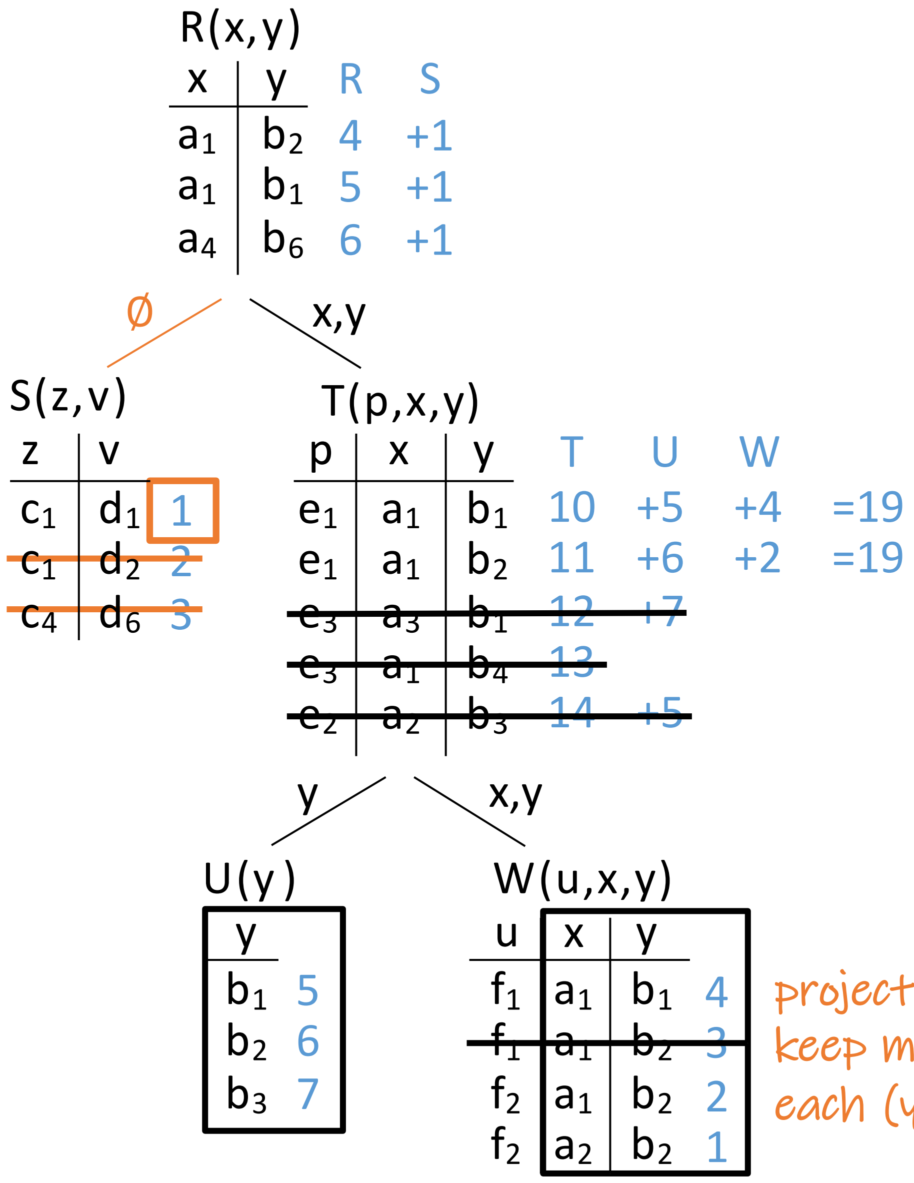
SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql
Towards Responsive DBMS. ICDE 2022 tutorial: https://northeastern-datalab.github.io/responsive-dbms-tutorial

Yannakakis Algorithm – Example

631

Q(x,y,z,v,p,u) :- R(x,y), S(z,v), T(p,x,y), U(y), W(u,x,y).

1. Adapted bottom-up semi-join propagation in $O(|\text{input}|)$.
   Finds the minimum value



R(x,y)

x	y	R	S
a_1	b_2	4	+1
a_1	b_1	5	+1
a_4	b_6	6	+1

no common variables, thus keep min across all tuples

S(z,v)

z	v	
c_1	d_1	1
~~c_1~~	~~d_2~~	~~2~~
~~c_4~~	~~d_6~~	~~3~~

T(p,x,y)

p	x	y	T	U	W	
e_1	a_1	b_1	10	+5	+4	=19
e_1	a_1	b_2	11	+6	+2	=19
~~e_3~~	~~a_3~~	~~b_1~~	~~12~~	~~+7~~		
~~e_3~~	~~a_1~~	~~b_4~~	~~13~~			
~~e_2~~	~~a_2~~	~~b_3~~	~~14~~	~~+5~~		

U(y)

y	
b_1	5
b_2	6
b_3	7

W(u,x,y)

u	x	y	
f_1	a_1	b_1	4
~~f_1~~	~~a_1~~	~~b_2~~	~~3~~
f_2	a_1	b_2	2
f_2	a_2	b_2	1

project & keep min for each (y,z)

Each tuple now has a weight.

New goal: Find the "lightest join" (minimum sum of weights)

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Towards Responsive DBMS. ICDE 2022 tutorial: https://northeastern-datalab.github.io/responsive-dbms-tutorial

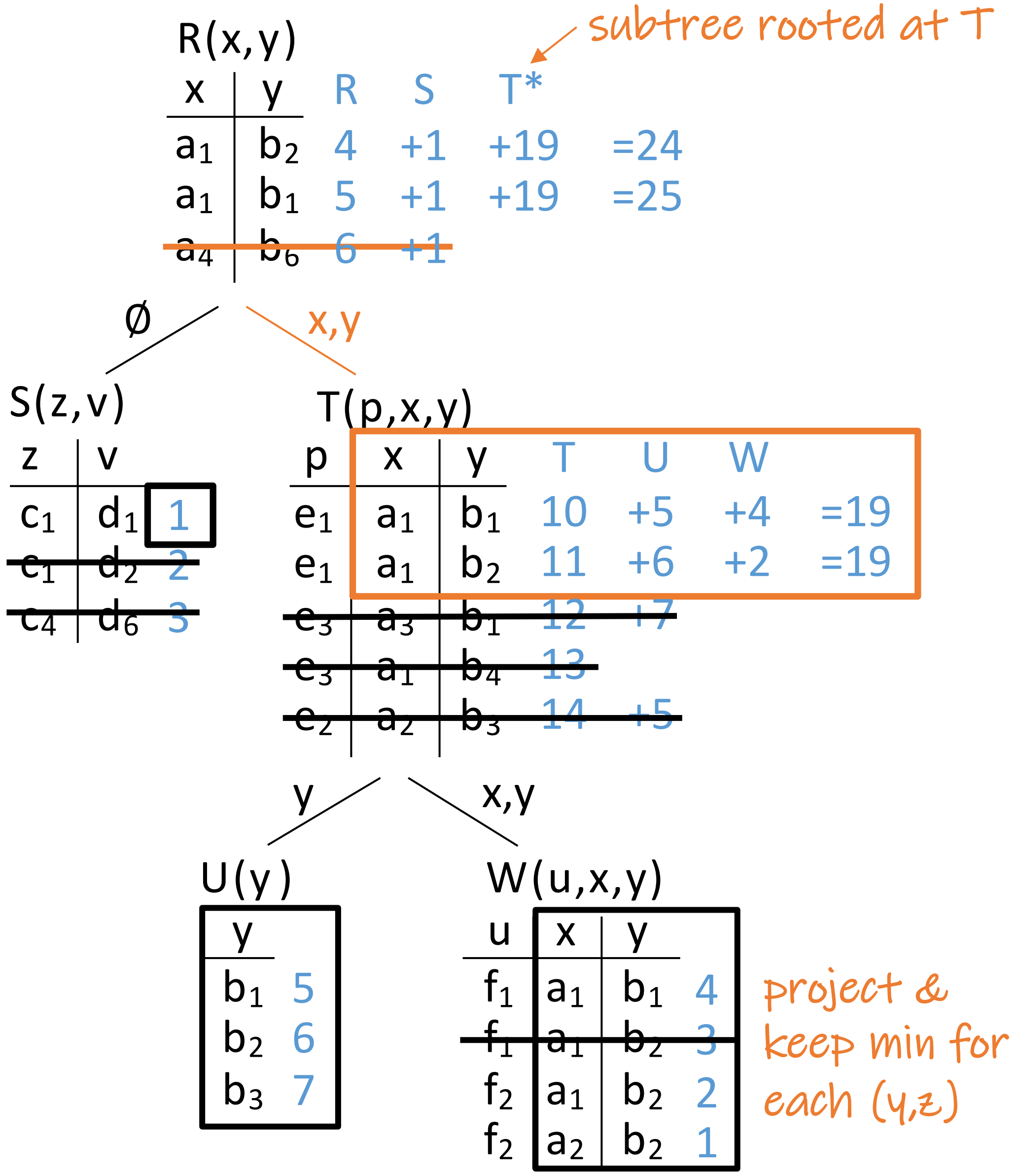
Yannakakis Algorithm – Example

Q(x,y,z,v,p,u) :- R(x,y), S(z,v), T(p,x,y), U(y), W(u,x,y).

1. Adapted bottom-up semi-join propagation in $O(|\text{input}|)$.
   Finds the minimum value

Each tuple now has a weight.

New goal: Find the "lightest join" (minimum sum of weights)



subtree rooted at T

R(x,y)

x	y	R	S	T*	
a_1	b_2	4	+1	+19	=24
a_1	b_1	5	+1	+19	=25
~~a_4~~	~~b_6~~	~~6~~	~~+1~~		

S(z,v)

z	v	
c_1	d_1	$\boxed{1}$
~~c_1~~	~~d_2~~	~~2~~
~~c_4~~	~~d_6~~	~~3~~

T(p,x,y)

p	x	y	T	U	W	
e_1	a_1	b_1	10	+5	+4	=19
e_1	a_1	b_2	11	+6	+2	=19
~~e_3~~	~~a_3~~	~~b_1~~	~~12~~	~~+7~~		
~~e_3~~	~~a_1~~	~~b_4~~	~~13~~			
~~e_2~~	~~a_2~~	~~b_3~~	~~14~~	~~+5~~		

U(y)

y	
b_1	5
b_2	6
b_3	7

W(u,x,y)

u	x	y	
f_1	a_1	b_1	4
~~f_1~~	~~a_1~~	~~b_2~~	~~3~~
f_2	a_1	b_2	2
f_2	a_2	b_2	1

project & keep min for each (y,z)

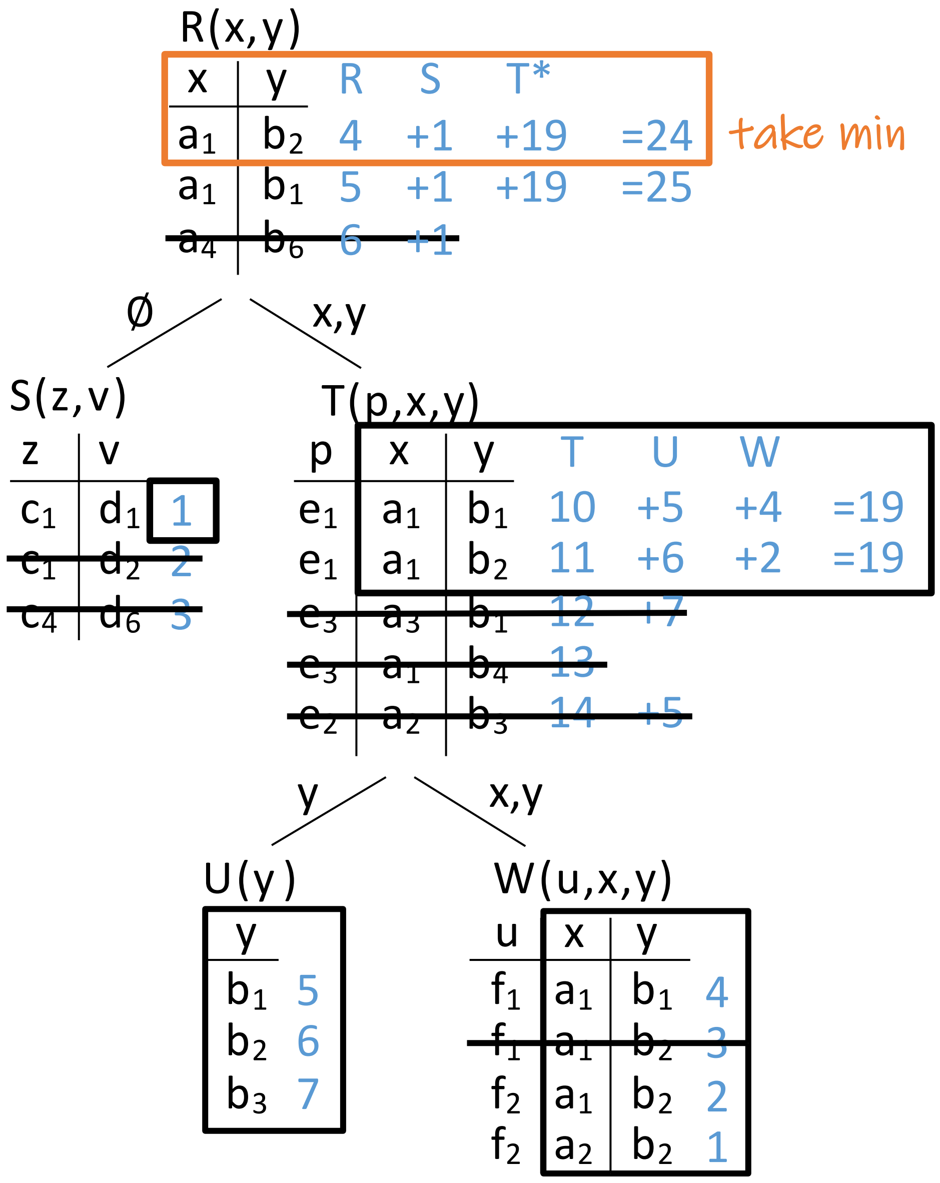
SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Towards Responsive DBMS. ICDE 2022 tutorial: https://northeastern-datalab.github.io/responsive-dbms-tutorial

Yannakakis Algorithm – Example

Q(x,y,z,v,p,u) :- R(x,y), S(z,v), T(p,x,y), U(y), W(u,x,y).

1. Adapted bottom-up semi-join propagation in $O(|\text{input}|)$.
   Finds the minimum value = 24



R(x,y)

x	y	R	S	T*	
a_1	b_2	4	+1	+19	=24
a_1	b_1	5	+1	+19	=25
~~a_4~~	~~b_6~~	~~6~~	~~+1~~		

S(z,v)

z	v	
c_1	d_1	1
~~c_1~~	~~d_2~~	~~2~~
~~c_4~~	~~d_6~~	~~3~~

T(p,x,y)

p	x	y	T	U	W	
e_1	a_1	b_1	10	+5	+4	=19
e_1	a_1	b_2	11	+6	+2	=19
~~e_3~~	~~a_3~~	~~b_1~~	~~12~~	~~+7~~		
~~e_3~~	~~a_1~~	~~b_4~~	~~13~~			
~~e_2~~	~~a_2~~	~~b_3~~	~~14~~	~~+5~~		

U(y)

y	
b_1	5
b_2	6
b_3	7

W(u,x,y)

u	x	y	
f_1	a_1	b_1	4
~~f_1~~	~~a_1~~	~~b_2~~	~~3~~
f_2	a_1	b_2	2
f_2	a_2	b_2	1

Each tuple now has a weight.

New goal: Find the "lightest join"
(minimum sum of weights)

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql
Towards Responsive DBMS. ICDE 2022 tutorial: https://northeastern-datalab.github.io/responsive-dbms-tutorial

Yannakakis Algorithm – Example

Q(x,y,z,v,p,u) :- R(x,y), S(z,v), T(p,x,y), U(y), W(u,x,y).

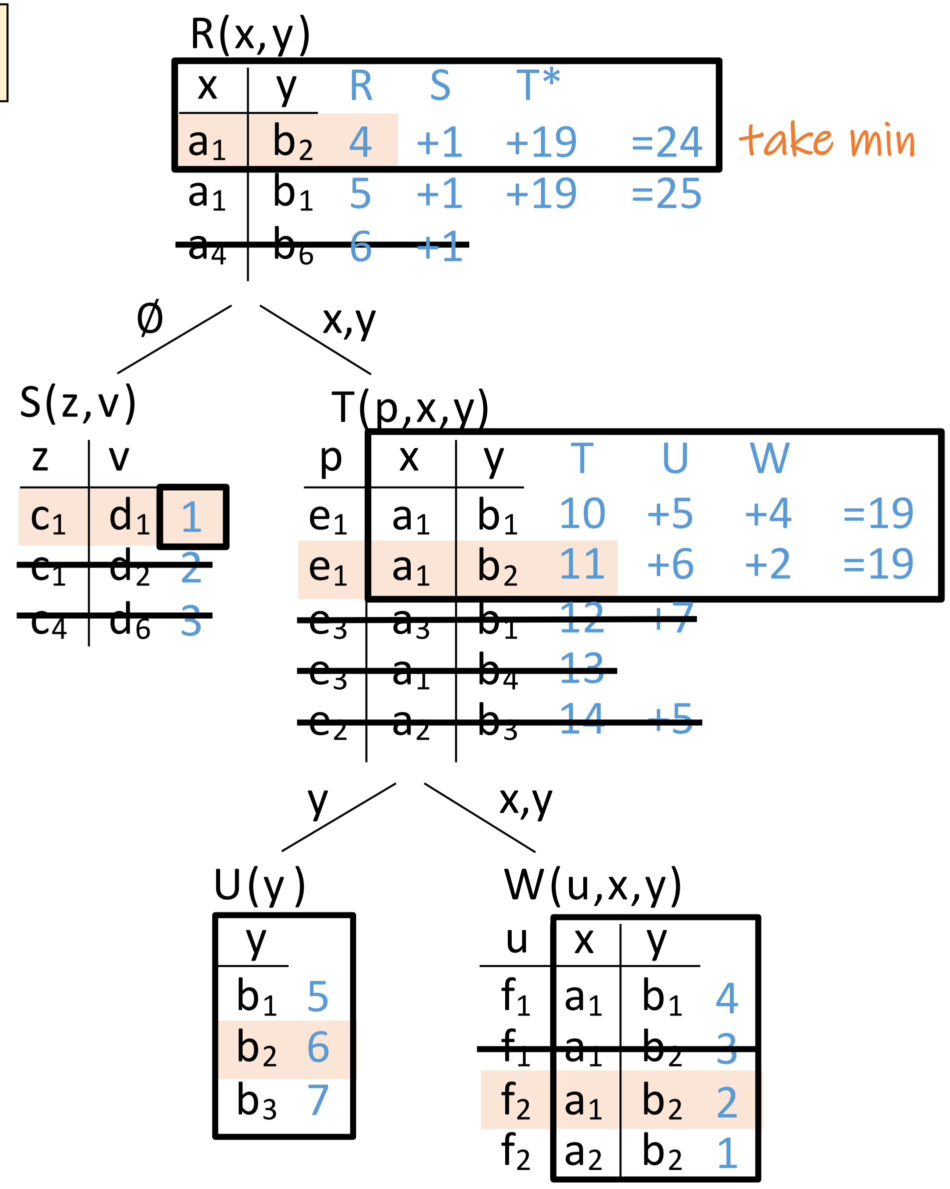
1. Adapted bottom-up semi-join propagation in $O(|\text{input}|)$.
   Finds the minimum value = 24
2. Top-down traversal (equi-join) in $O(1)$.
   Builds an optimal solution: = (a_1,b_1,c_2,d_1,e_1)

Tree Dynamic Programming in time $O(n)$!

Fractional edge cover for Q is 3. Thus a standard DBMS plan will need $O(n^3)$

Each tuple now has a weight.

New goal: Find the "lightest join" (minimum sum of weights)



R(x,y)

x	y	R	S	T*	
a_1	b_2	4	+1	+19	=24
a_1	b_1	5	+1	+19	=25
~~a_4~~	~~b_6~~	~~6~~	~~+1~~		

S(z,v)

z	v	
c_1	d_1	1
~~c_1~~	~~d_2~~	~~2~~
~~c_4~~	~~d_6~~	~~3~~

T(p,x,y)

p	x	y	T	U	W	
e_1	a_1	b_1	10	+5	+4	=19
e_1	a_1	b_2	11	+6	+2	=19
~~e_3~~	~~a_3~~	~~b_1~~	~~12~~	~~+7~~		
~~e_3~~	~~a_1~~	~~b_4~~	~~13~~			
~~e_2~~	~~a_2~~	~~b_3~~	~~14~~	~~+5~~		

U(y)

y	
b_1	5
b_2	6
b_3	7

W(u,x,y)

u	x	y	
f_1	a_1	b_1	4
~~f_1~~	~~a_1~~	~~b_2~~	~~3~~
f_2	a_1	b_2	2
f_2	a_2	b_2	1

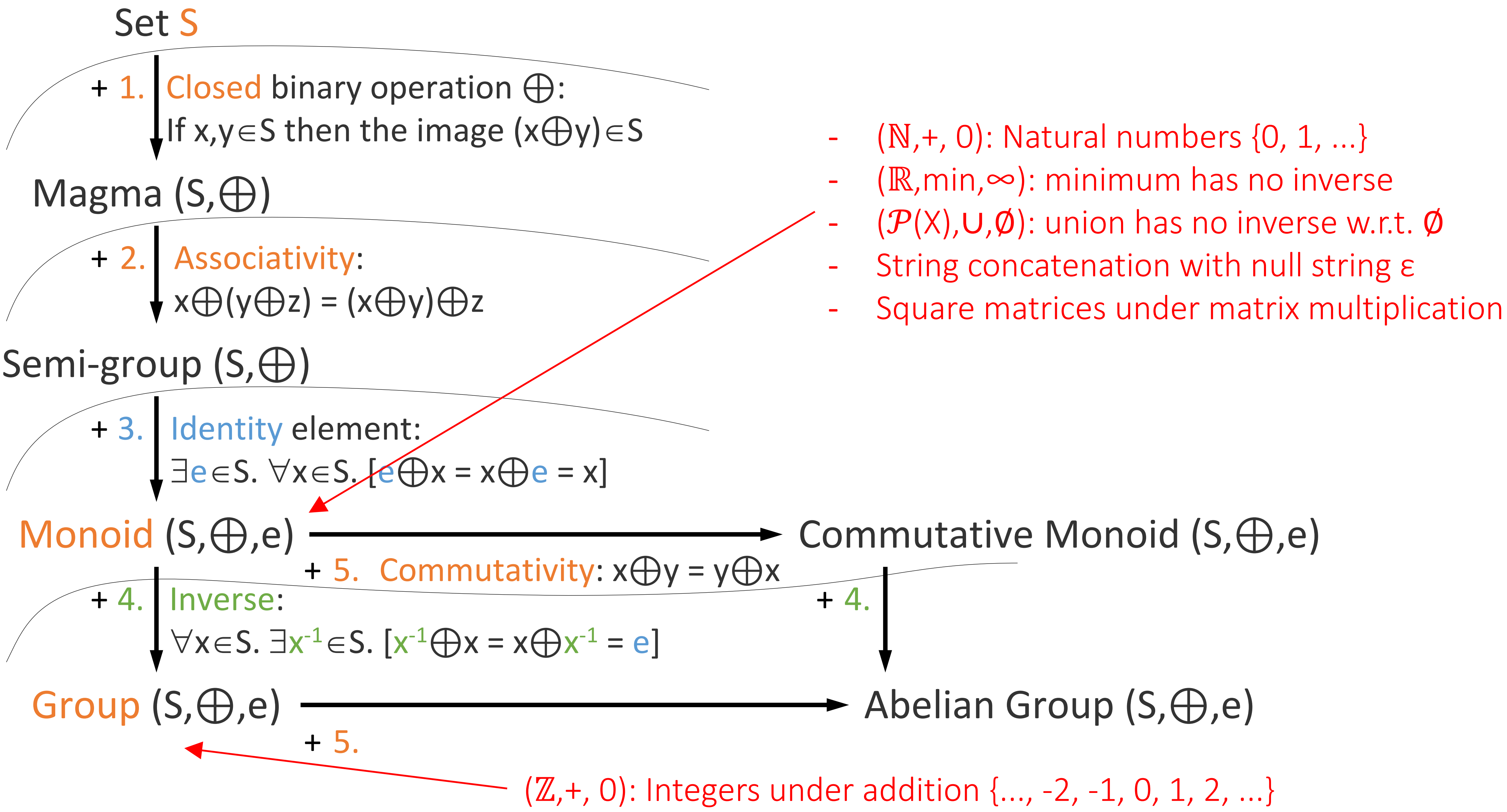
SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql
Towards Responsive DBMS. ICDE 2022 tutorial: https://northeastern-datalab.github.io/responsive-dbms-tutorial

Outline Part 5

Part 5: Dynamic Programming & Semirings (Wolfgang) ~20min

- Top-1 = Dynamic Programming (DP)
- Top-1 Yannakakis as variant of Tree-DP
- Algebra: Totally Ordered Commutative Monoids

Group-like structures: a set & one binary operation



Set S

+ 1. Closed binary operation $\oplus$:
If $x,y \in S$ then the image $(x \oplus y) \in S$

Magma $(S,\oplus)$

+ 2. Associativity:
$x \oplus (y \oplus z) = (x \oplus y) \oplus z$

Semi-group $(S,\oplus)$

+ 3. Identity element:
$\exists e \in S.\ \forall x \in S.\ [e \oplus x = x \oplus e = x]$

Monoid $(S,\oplus,e)$

+ 5. Commutativity: $x \oplus y = y \oplus x$

Commutative Monoid $(S,\oplus,e)$

+ 4. Inverse:
$\forall x \in S.\ \exists x^{-1} \in S.\ [x^{-1} \oplus x = x \oplus x^{-1} = e]$

+ 4.

Group $(S,\oplus,e)$

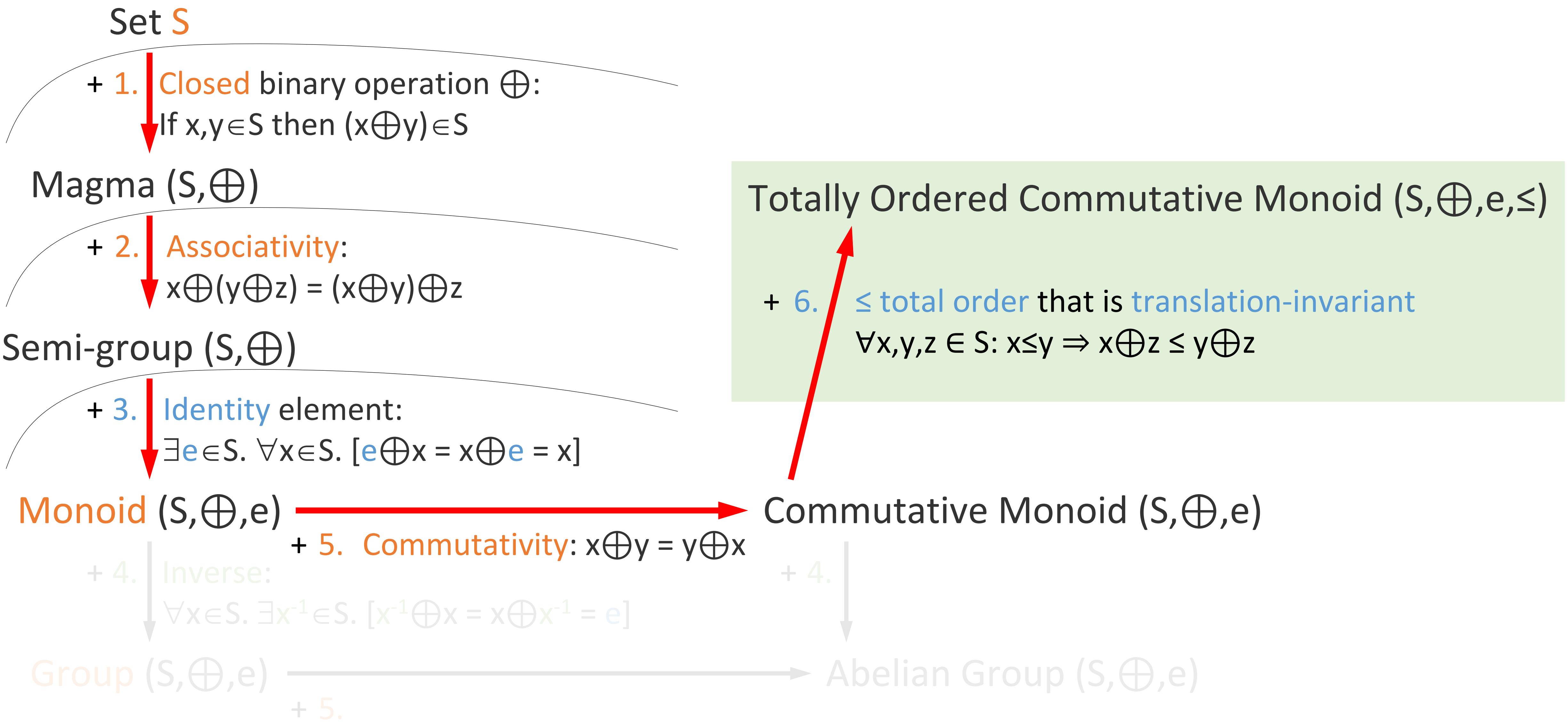
+ 5.

Abelian Group $(S,\oplus,e)$

- $(\mathbb{N},+,0)$: Natural numbers {0, 1, ...}
- $(\mathbb{R},\min,\infty)$: minimum has no inverse
- $(\mathcal{P}(X),\cup,\emptyset)$: union has no inverse w.r.t. $\emptyset$
- String concatenation with null string ε
- Square matrices under matrix multiplication

$(\mathbb{Z},+,0)$: Integers under addition {..., -2, -1, 0, 1, 2, ...}

Group-like structures: a set & one binary operation



Set S

+ 1. Closed binary operation $\oplus$: If $x,y \in S$ then $(x \oplus y) \in S$

Magma $(S,\oplus)$

+ 2. Associativity: $x \oplus (y \oplus z) = (x \oplus y) \oplus z$

Semi-group $(S,\oplus)$

+ 3. Identity element: $\exists e \in S.\ \forall x \in S.\ [e \oplus x = x \oplus e = x]$

Monoid $(S,\oplus,e)$

+ 5. Commutativity: $x \oplus y = y \oplus x$

Commutative Monoid $(S,\oplus,e)$

+ 6. $\leq$ total order that is translation-invariant $\forall x,y,z \in S$: $x \leq y \Rightarrow x \oplus z \leq y \oplus z$

Totally Ordered Commutative Monoid $(S,\oplus,e,\leq)$

+ 4. Inverse: $\forall x \in S.\ \exists x^{-1} \in S.\ [x^{-1} \oplus x = x \oplus x^{-1} = e]$

Group $(S,\oplus,e)$

+ 5.

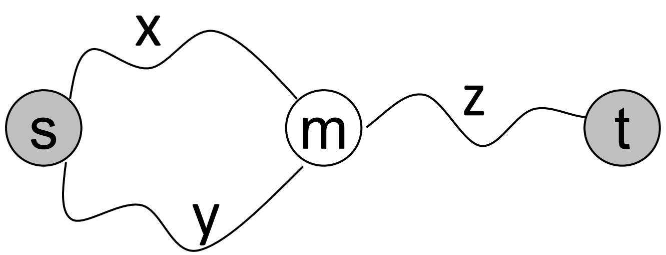
+ 4.

Abelian Group $(S,\oplus,e)$

Totally ordered commutative monoid

- Totally ordered commutative monoid $(S,\oplus,e,\leq)$
  6. $\leq$ total order that is translation-invariant (sometimes called "compatible" with $\oplus$, or monotonic), i.e. $\forall x,y,z \in S$: $x\leq y \Rightarrow x\oplus z \leq y\oplus z$
  - equivalent to "optimal substructure" in DP

when the solution to an optimization problem can be constructed from optimal solutions to its subproblems.



$x\leq y \quad \oplus \quad z$

- Let's generalize
  - $\min[(x\oplus z), (y\oplus z)] = \min[x,y] \oplus z$
  - $(x\oplus z) \min (y\oplus z) = (x \min y) \oplus z$ (+ distributes over min)
  - $(x \cdot z) \quad + \quad (y \cdot z) = (x \quad + \quad y) \quad \cdot z$ (multipl. distributes over add.)

Semirings: two binary operators

semirings are rings
w/o the additive inverse

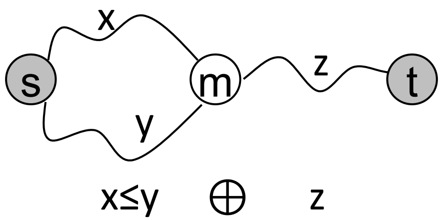
- Semiring $(S,\oplus,\otimes,0,1)$
  1. $(S,\oplus,0)$ is commutative monoid
  2. $(S,\otimes,1)$ is monoid
  3. $\otimes$ distributes over $\oplus$: $(x\oplus y) \otimes z = (x\otimes z) \oplus (y\otimes z)$
  4. 0 annihilates $\otimes$: $0 \otimes x = 0$ (0 is an absorbing element for $\otimes$)

e.g.: Natural numbers
under addition $(\mathbb{N},+, 0)$

- Examples
  1. $\mathbb{T}=(\mathbb{R}_+^{\infty},\min,+,\infty,0)$ Shortest-distance: $\min[x,y] + z = \min[(x+z),(y+z)]$

     min-sum semiring, also called tropical semiring: sum distributes over min

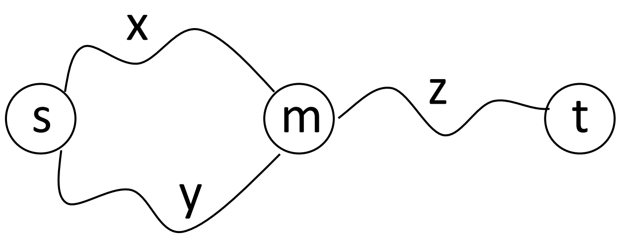
     not the other way: $\min[x+y,z] \neq \min[x,z] + \min[y,z]$; e.g. $\min[3+4,5] = 5 \neq 7 = \min[3,5] + \min[4,5]$
  2. $\mathbb{N}=(\mathbb{N},+,\cdot,0,1)$ Number of paths (bag semantics)
  3. $\mathbb{R}=(\mathbb{R},+,\cdot,0,1)$ Ring of real numbers
  4. $\mathbb{B}=(\{0,1\},\vee,\wedge,0,1)$ Boolean (set semantics)
  5. $\mathbb{V}=([0,1],\max,\cdot,0,1)$ Probability of best derivation (Viterbi)



Two equivalent algebraic perspectives of DP

Monoid perspective

- Totally ordered commutative monoid $(S,\otimes,e,\leq)$
  - $\leq$ total order that is translation-invariant, i.e. $\forall x,y,z \in S$: $x \leq y \Rightarrow x \otimes z \leq y \otimes z$
  - implies the distributivity law ($\otimes$ distributes over min): $(x \otimes z) \min (y \otimes z) = (x \min y) \otimes z$
  - equivalent to "optimal substructure" in DP



$x \leq y \quad \oplus \quad z$

Semiring perspective

- Selective commutative dioid $(S,\oplus,\otimes,e_{\oplus},e_{\otimes})$
  - semiring, thus $\otimes$ distributes over $\oplus$
  - semiring, thus $\oplus$ is commutative
  - commutative semiring, thus $\otimes$ is also commutative
  - additionally, $\oplus$ is selective: $x \oplus y = x$ or $y \quad \forall x,y \in S$
  - selectivity & commutativity implies: total order $\leq$ on S

$(\mathbb{R}_{+}^{\infty},+,0)$: totally ordered comm. monoid

$(\mathbb{R}_{+}^{\infty},\min,\infty)$: selective monoid

$(\mathbb{R}_{+}^{\infty},\min,+,\infty,0)$: tropical semiring

Summary Dynamic Programming & Semirings

1. DP works on trees and can be seen as a variant of message passing
2. DP is often the same problem as shortest path finding
3. Semirings allow us to further abstract what DP does:
   - Shortest Paths problems are instances of the tropical semiring $\mathbb{T}=(\mathbb{R}^{\infty},\min,+,\infty,0)$
   - Path counting on the $\mathbb{N}=(\mathbb{N},+,\cdot,0,1)$ semiring
4. Yannakakis is DP on the Boolean semiring $\mathbb{B}=(\{0,1\},\vee,\wedge,0,1)$

Outline tutorial

1: Introduction (Nikos) ~40min

2: Tree Decompositions (Mirek) ~20min

3: Acyclic Queries & Enumeration (Wolfgang) ~25min

BREAK

4: Factorization (Nikos) ~15min

5: Dynamic Programming & Semirings (Wolfgang) ~20min

6: Any-k or Ranked Enumeration (Nikos) ~35min

7. Decomposition of Comparison Predicates (Mirek) ~10min

8. Conclusion (Mirek) ~5min