CSE544: Principles of Database Systems

Review of MapReduce
Datalog (cont'd)

Announcements

- Review 7 (Datalog) was due yesterday
- Project Milestone due next Friday
- HW3 is due the following Monday

MapReduce Review

- What is the map function?
- What is the reduce function?
- What is a map task?
- What is a reduce task?
- What is a mapreduce job?

MapReduce Review

- What is the map function?
  - Takes (k,v) returns {(k1,v1),(k2,v2),…}
- What is the reduce function?
  - Takes (k,{v1,v2,…}) returns any result
- What is a map task?
  - A set of (k,v) pairs that are scheduled as a unit
- What is a reduce task?
  - A set of (k,{v1,…}) pairs scheduled as a unit
- What is a mapreduce job?
  - The entire map reduce program

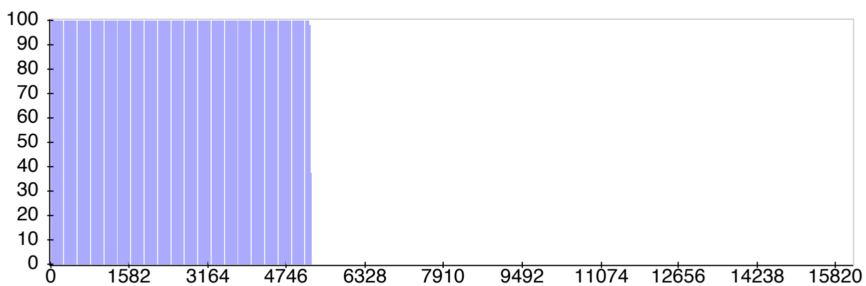
Anatomy of a Query Execution

- Running Part B of HW2
- 20 nodes = 1 master + 19 workers
- Using PARALLEL 50
- Let’s see what happened

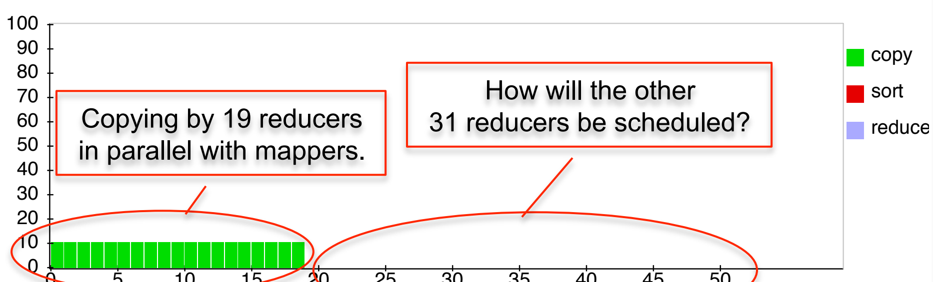
1h 16min

Only 19 reducers active, out of 50. Why?

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	33.17%	15816	10549	38	5229	0	0 / 0
reduce	4.17%	50	31	19	0	0	0 / 0



luce Completion Graph - close

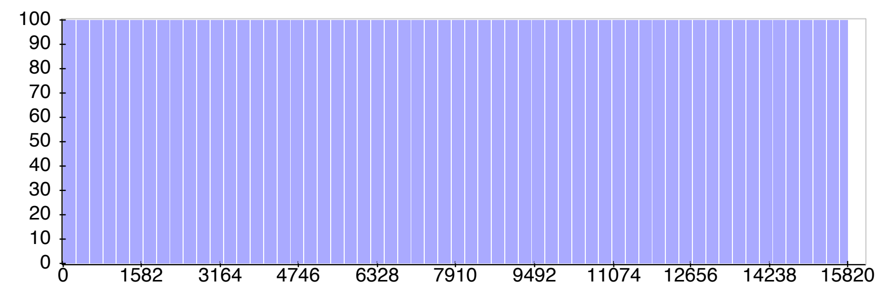


3h 50min

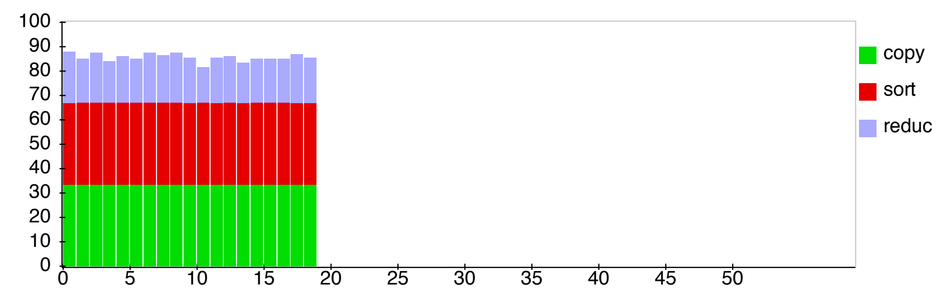
Some errors start to occur. Watch this…

Completed. Sorting, and the rest of Reduce may proceed now

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	15816	0	0	15816	0	0 / 18
reduce	32.42%	50	31	19	0	0	0 / 0

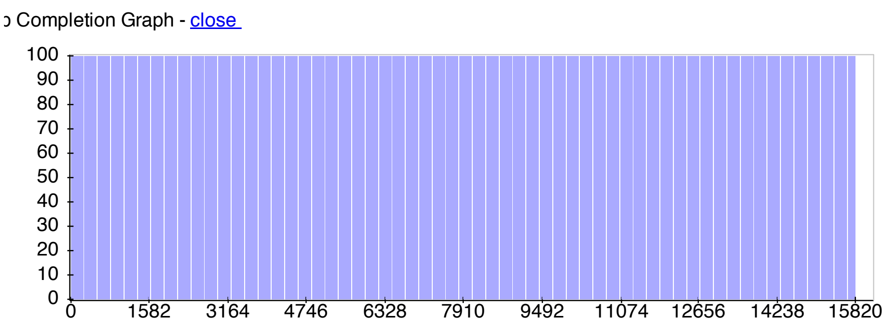


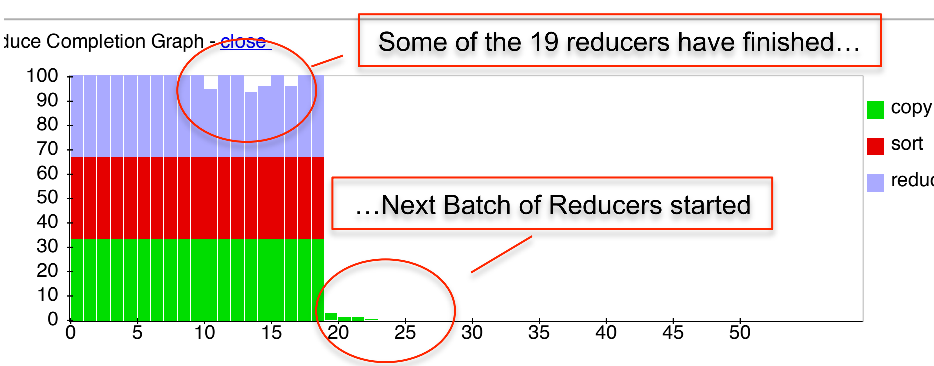
ıce Completion Graph - close



3h 51min

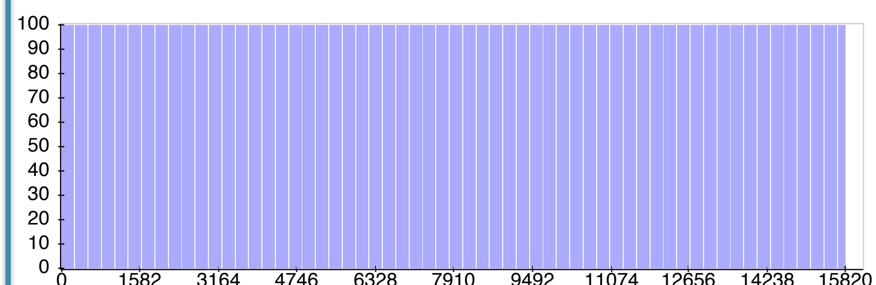
Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	15816	0	0	15816	0	0 / 18
reduce	37.72%	50	19	22	9	0	0 / 0

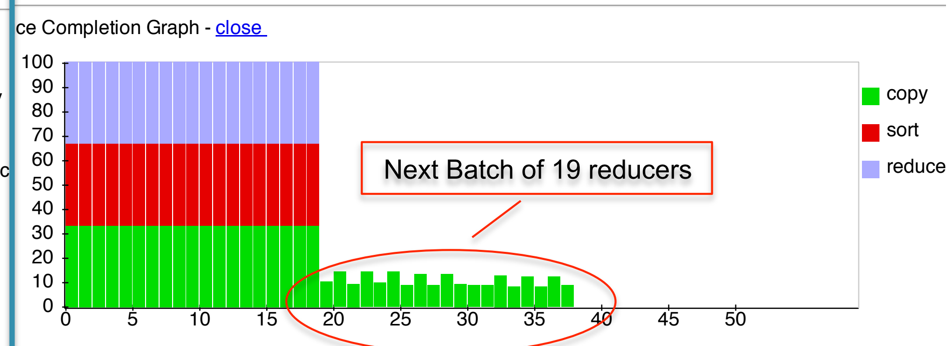




3h 52min

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	15816	0	0	15816	0	0 / 18
reduce	42.35%	50	11	20	19	0	0 / 0

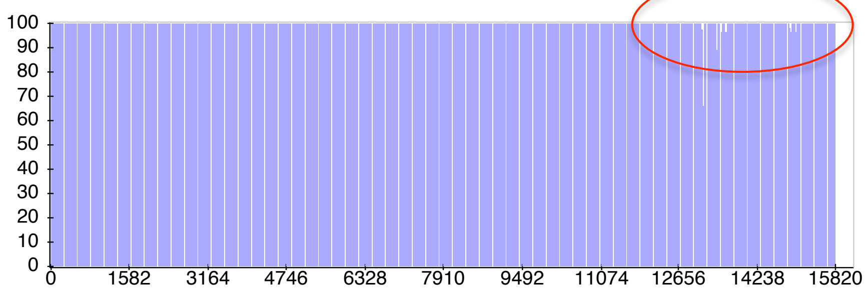




4h 18min

Several servers failed: “fetch error”. Their map tasks need to be rerun. All reducers are waiting….

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	99.88%	15816	2638	30	13148	0	15 / 3337
reduce	48.42%	50	15	16	19	0	0 / 0

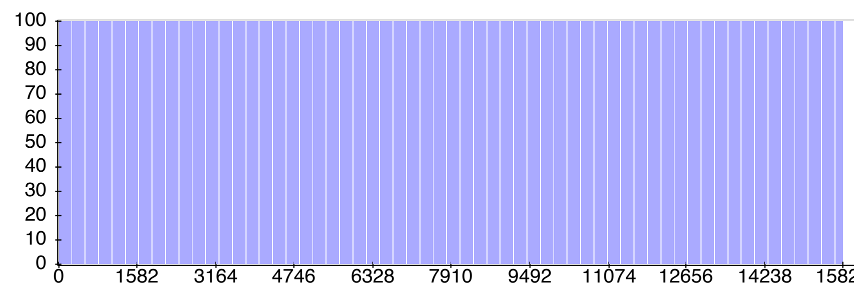


uce Completion Graph - close

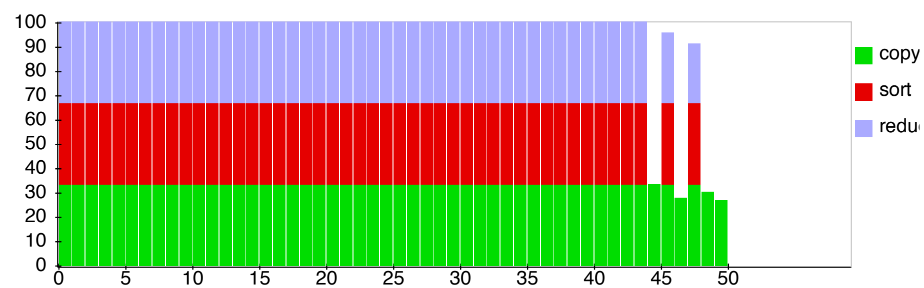
7h 10min

Mappers finished, reducers resumed.

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	15816	0	0	15816	0	26 / 5968
reduce	94.15%	50	0	6	44	0	0 / 8



uce Completion Graph - close



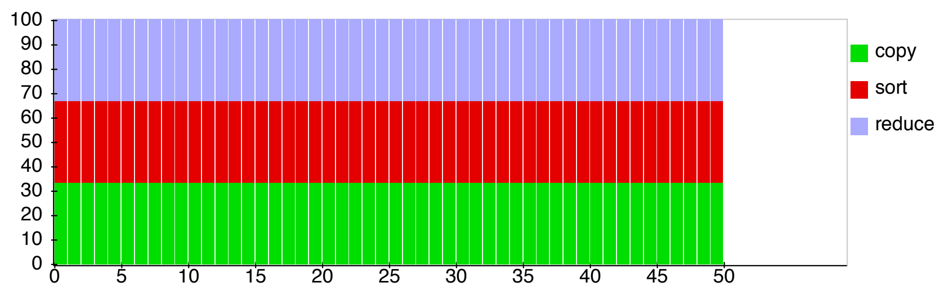
7h 20min

Success! 7hrs, 20mins.

Hadoop job_201203041905_0001 on ip-10-203-30-146

User: hadoop
Job Name: PigLatin:DefaultJobName
Job File: hdfs://10.203.30.146:9000/mnt/var/lib/hadoop/tmp/mapred/staging/hadoop/.staging/job_201203041905_0001/job.xml
Submit Host: ip-10-203-30-146.ec2.internal
Submit Host Address: 10.203.30.146
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Succeeded
Started at: Sun Mar 04 19:08:29 UTC 2012
Finished at: Mon Mar 05 02:28:39 UTC 2012
Finished in: 7hrs, 20mins, 10sec
Job Cleanup: Successful
Black-listed TaskTrackers: 3

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	15816	0	0	15816	0	26 / 5968
reduce	100.00%	50	0	0	50	0	0 / 14



Degree Sequence

Famous Example of Big Data Analysis

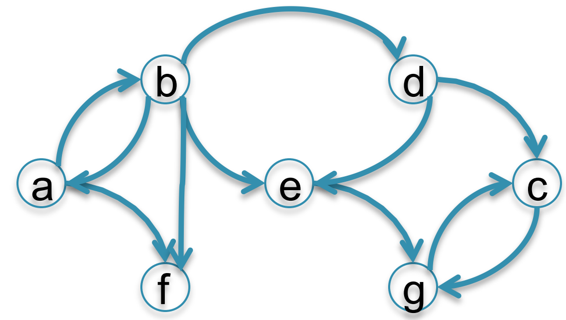
Kumar et al., *The Web as a Graph*

- Question 1: is the Web like a “random graph”?
- Question 2: how does the Web graph look like?

Graph as Databases

Many large databases are graphs

- The Web
- The Internet
- Social Networks
- Flights btw. Airports
- Etc,etc,etc



Source	Target
a	b
b	a
a	f
b	f
b	e
b	d
d	e
d	c
e	g
g	c
c	g

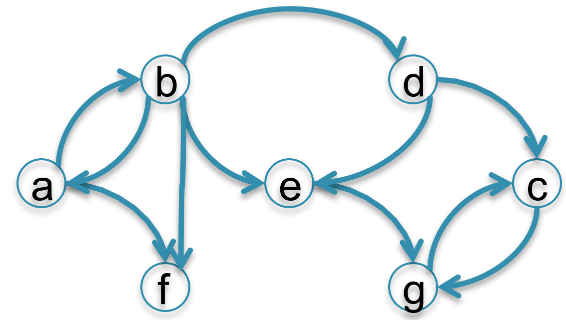
Data Analytics on Big Graphs

Queries expressible in SQL:

- How many nodes (edges)?
- How many nodes have > 4 neighbors?
- Which are the "most connected nodes"?

Queries requiring recursion:

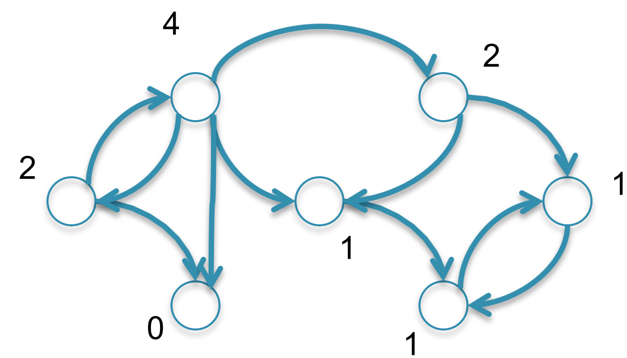
- Is the graph connected?
- What is the diameter of the graph?
- Compute *PageRank*
- Compute the *Centrality* of each node



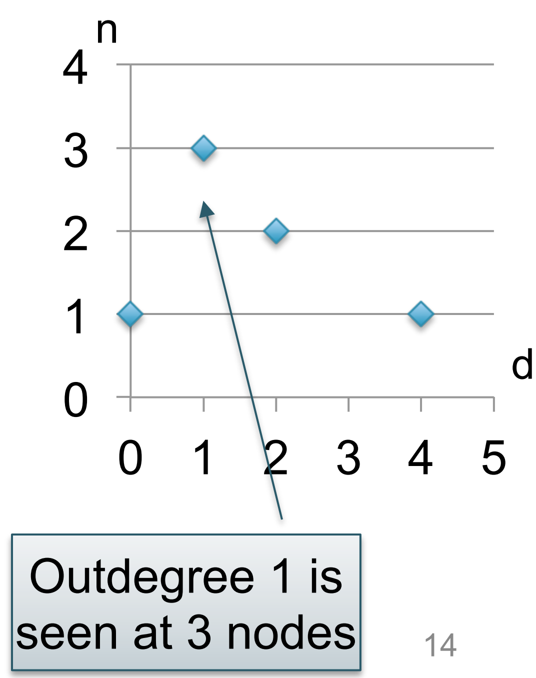
Source	Target
a	b
b	a
a	f
b	f
b	e
b	d
d	e
d	c
e	g
g	c
c	g

Histogram of a Graph
a.k.a. Degree Sequence

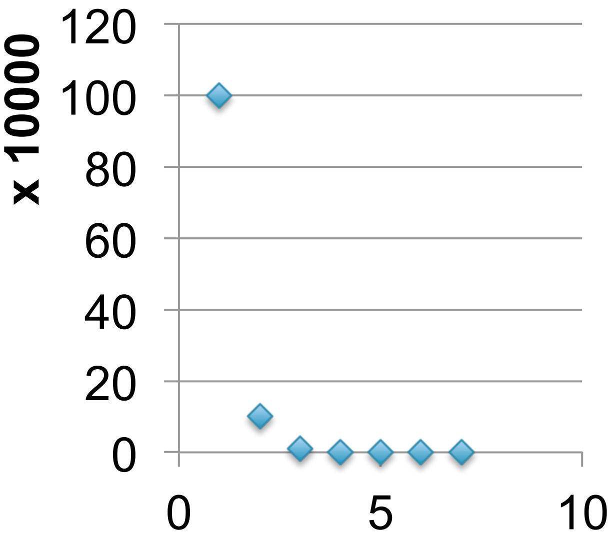
- Outdegree of a node = number of outgoing edges
- For each d, let n(d) = number of nodes with oudegree d
- The outdegree histogram of a graph = the scatterplot (d, n(d))

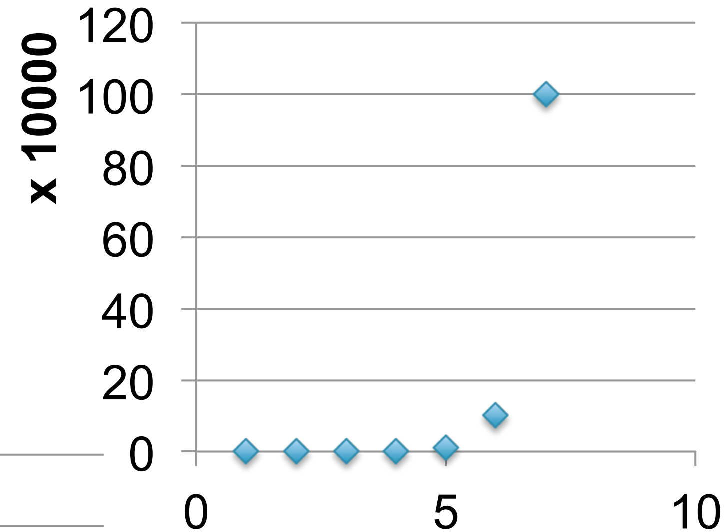


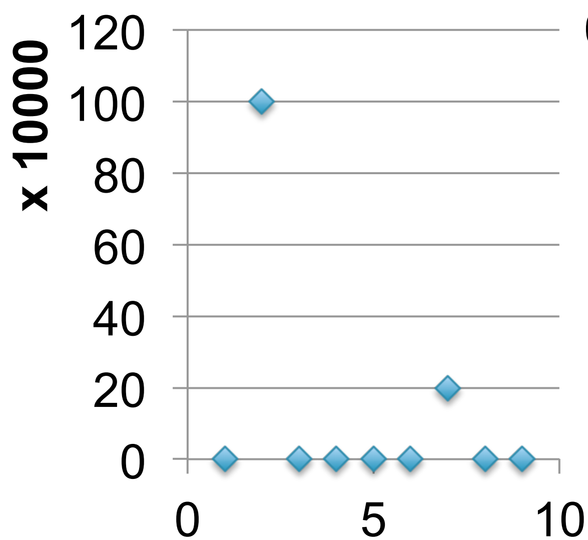
d	n(d)
0	1
1	3
2	2
3	0
4	1



Histograms Tell Us Something About the Graph



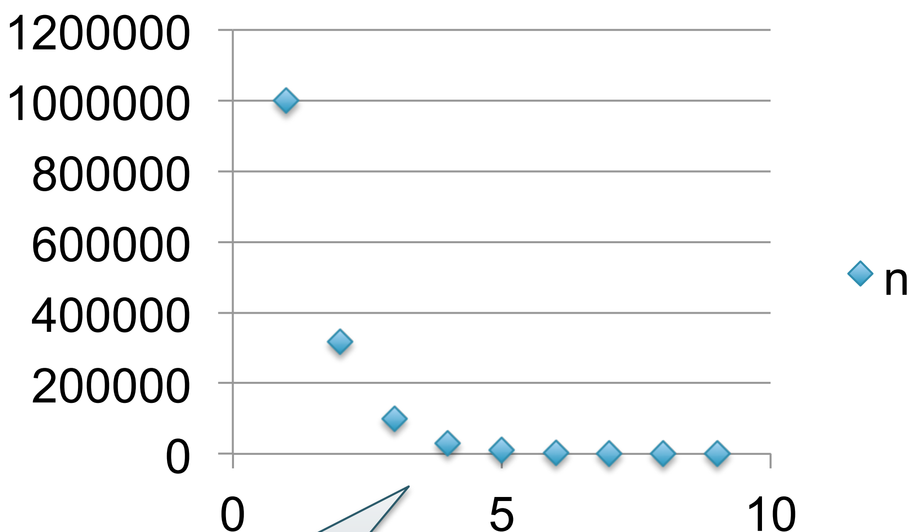


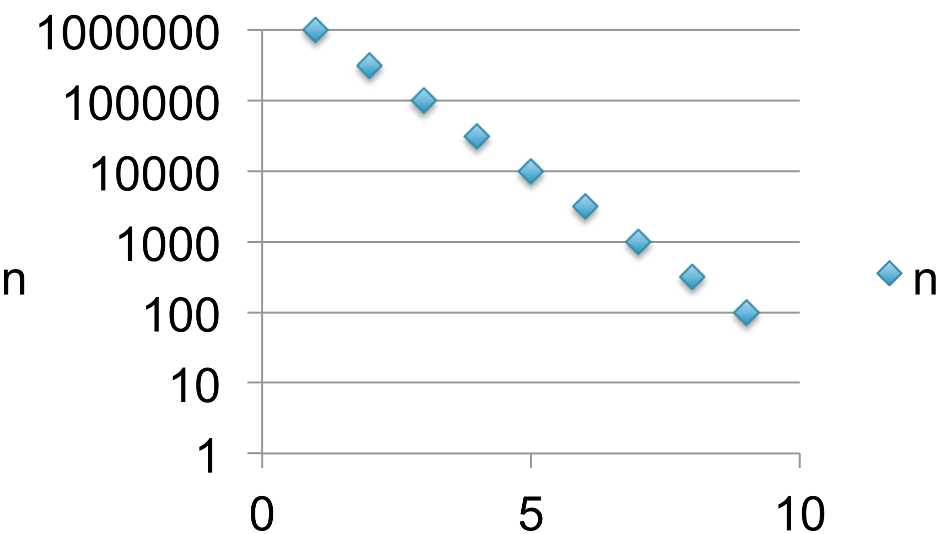


What can you say about these graphs?

Exponential Distribution

- n(d) ≅ $c/2^d$ (generally, cx^d, for some $x < 1$)
- A *random graph* has exponential distribution
- Best seen when n is on a log scale

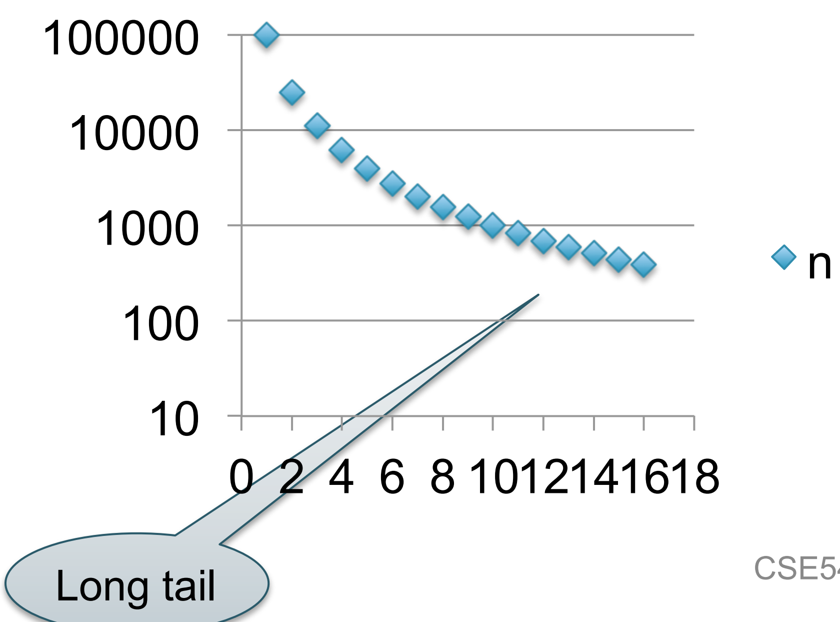


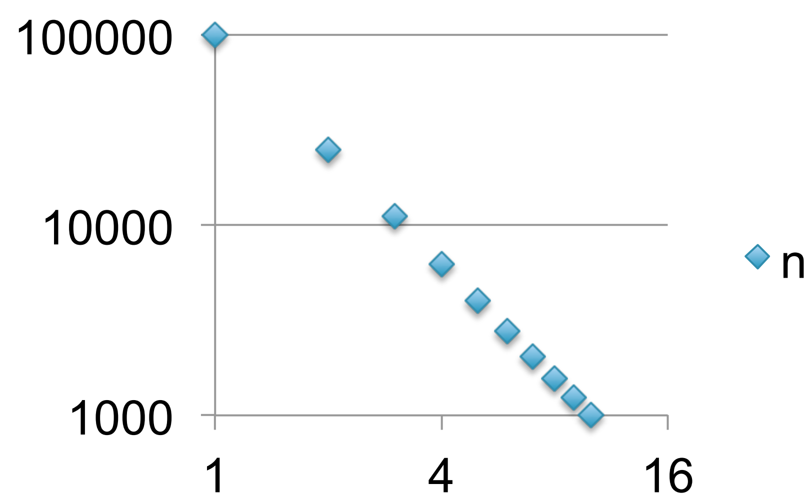


Quickly vanishing

Zipf Distribution

- $n(d) \cong 1/d^x$, for some value $x>0$
- Human-generated data has Zipf distribution: letters in alphabet, words in vocabulary, etc.
- Best seen in a log-log scale





The Histogram of the Web

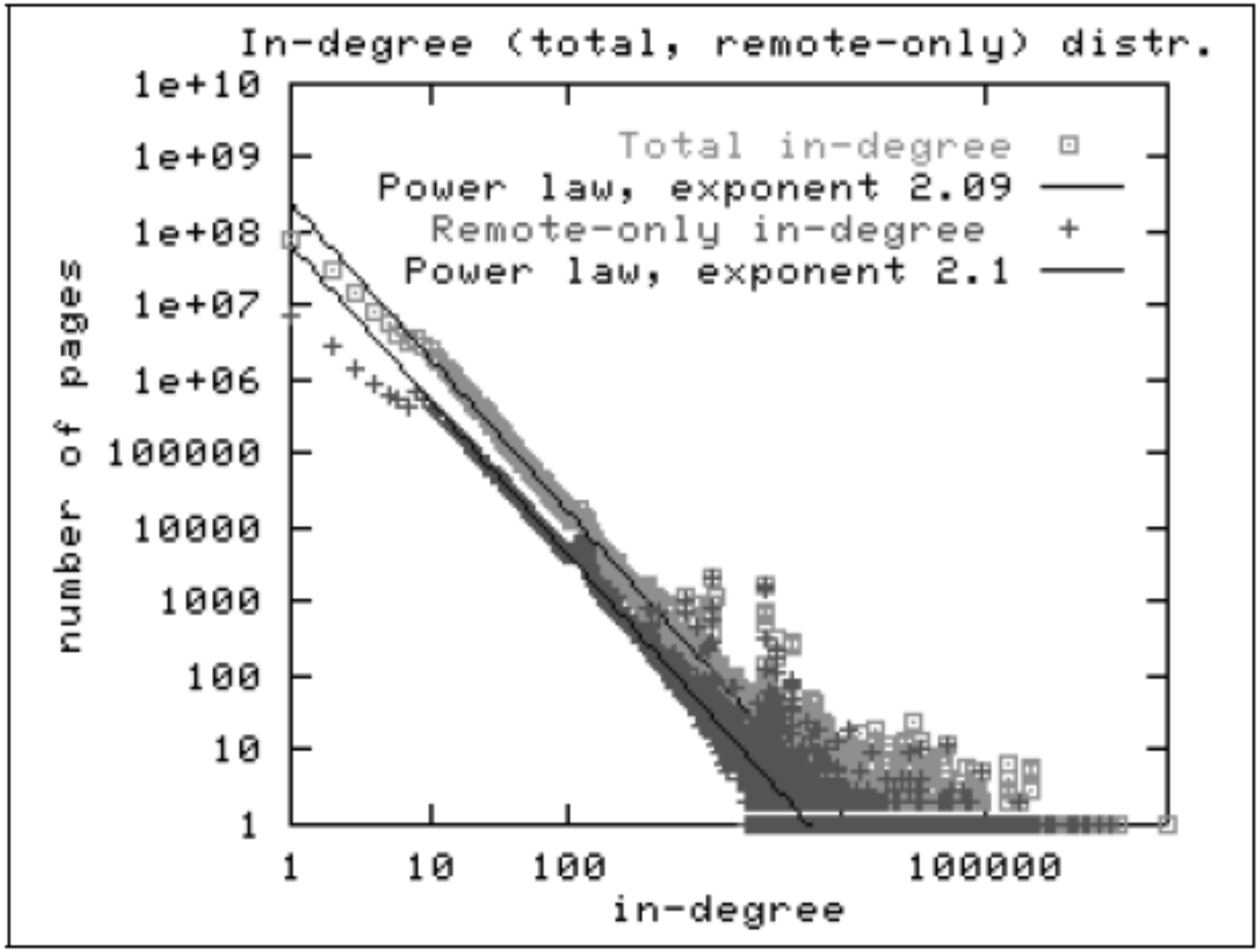


Figure 2: In-degree distribution.

Late 1990's
200M Webpages

Exponential ?

Zipf ?

The Bowtie Structure of the Web

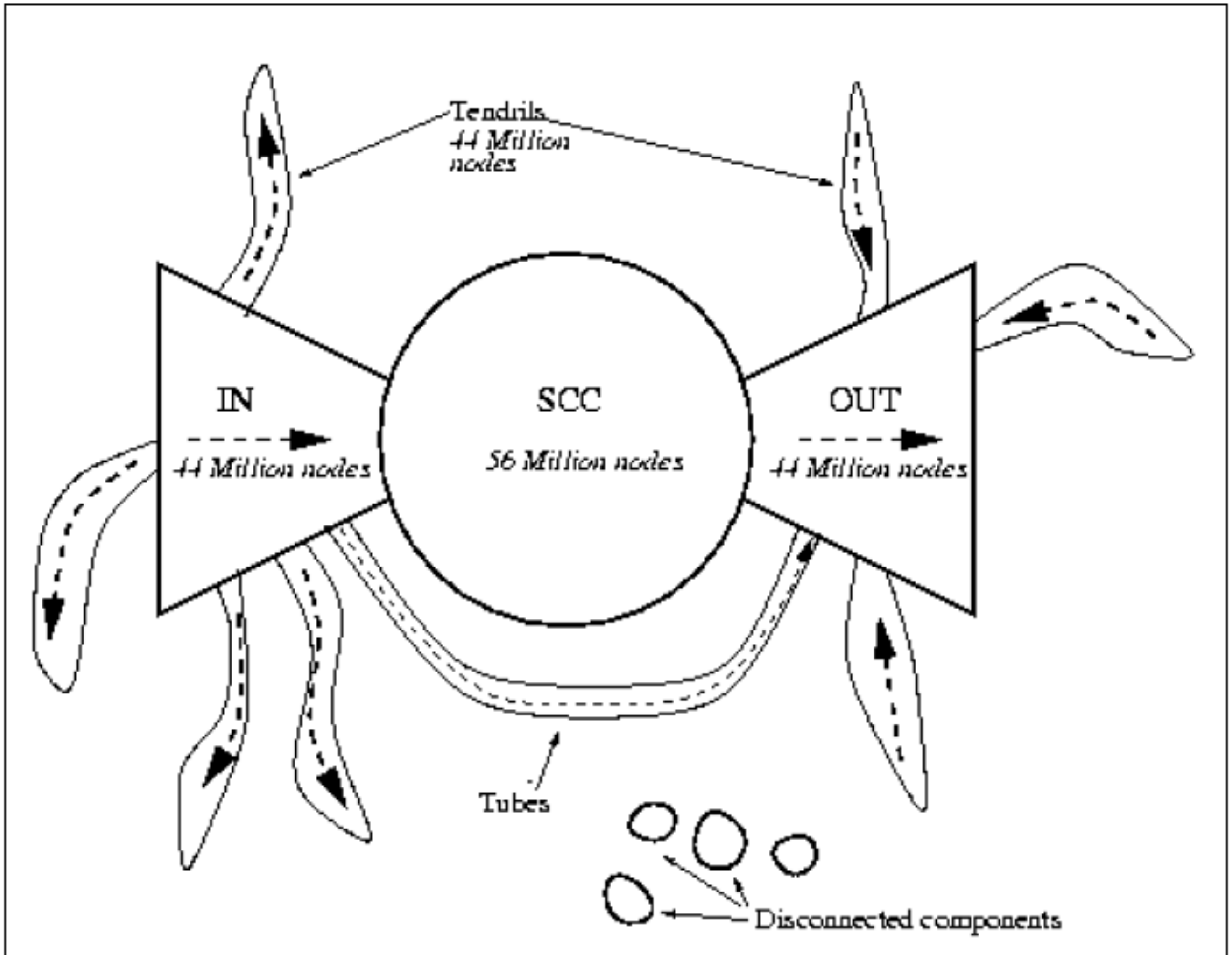


Figure 4: The web as a bowtie. SCC is a giant strongly connected component. IN consists of pages with paths to SCC, but no path from SCC. OUT consists of pages with paths from SCC, but no path to SCC. TENDRILS consists of pages that cannot surf to SCC, and which cannot be reached by surfing from SCC.

Review

- What is datalog?
- What is the naïve evaluation algorithm?
- What is the seminaive algorithm?

Discussion in Class

Assume a linear graph:

$0 \longrightarrow 1 \longrightarrow 2 \longrightarrow 3 \longrightarrow \ldots \longrightarrow n$

Right linear TC

```
T(x,y) :- R(x,y)
T(x,y) :- R(x,z), T(z,y)
```

Non-linear TC

```
T(x,y) :- R(x,y)
T(x,y) :- T(x,z), T(z,y)
```

- How many iterations are needed by each algorithm?
- How many times is the tuple T(0,n) discovered by each algorithm?

Discussion in Class

The *Declarative Imperative* paper:

- What are the extensions to datalog in Dedalus?
- What is the main usage of Dedalus described in the paper? Discuss some applications, discuss what’s missing.

Semantics of a Datalog Program

Three different, equivalent semantics:

- Minimal model semantics
- Least fixpoint semantics
- Proof-theoretic semantics

Minimal Model Semantics

To each rule r: $\boxed{P(x_1 \ldots x_k) \text{ :- } R_1(\ldots), R_2(\ldots), \ldots}$

Minimal Model Semantics

To each rule r: $P(x_1 \ldots x_k)$:- $R_1(\ldots), R_2(\ldots), \ldots$

All variables in the rule

Associate the logical sentence Σ_r: $\forall z_1 \ldots \forall z_n. [(R_1(\ldots) \wedge R_2(\ldots) \wedge \ldots) \rightarrow P(\ldots)]$

Minimal Model Semantics

To each rule r: $\boxed{P(x_1 \ldots x_k) \text{ :- } R_1(\ldots), R_2(\ldots), \ldots}$

All variables in the rule

Associate the logical sentence Σ_r: $\boxed{\forall z_1 \ldots \forall z_n . [(R_1(\ldots) \wedge R_2(\ldots) \wedge \ldots) \rightarrow P(\ldots)]}$

Same as: $\forall x_1 \ldots \forall x_k . [\exists y_1 \ldots \exists y_m . (R_1(\ldots) \wedge R_2(\ldots) \wedge \ldots) \rightarrow P(\ldots)]$

Head variables

Existential variables

Minimal Model Semantics

To each rule r: $P(x_1 \ldots x_k)$:- $R_1(\ldots), R_2(\ldots), \ldots$

All variables in the rule

Associate the logical sentence Σ_r: $\forall z_1 \ldots \forall z_n. [(R_1(\ldots) \wedge R_2(\ldots) \wedge \ldots) \rightarrow P(\ldots)]$

Same as: $\forall x_1 \ldots \forall x_k. [\exists y_1 \ldots \exists y_m.(R_1(\ldots) \wedge R_2(\ldots) \wedge \ldots) \rightarrow P(\ldots)]$

Head variables

Existential variables

Definition. If **P** is a datalog program,
$\Sigma_{\mathbf{P}}$ is the set of all logical sentences associated to its rules.

Minimal Model Semantics

To each rule r: $P(x_1 \ldots x_k) \text{ :- } R_1(\ldots), R_2(\ldots), \ldots$

All variables in the rule

Associate the logical sentence Σ_r: $\forall z_1 \ldots \forall z_n. [(R_1(\ldots) \wedge R_2(\ldots) \wedge \ldots) \rightarrow P(\ldots)]$

Same as: $\forall x_1 \ldots \forall x_k. [\exists y_1 \ldots \exists y_m.(R_1(\ldots) \wedge R_2(\ldots) \wedge \ldots) \rightarrow P(\ldots)]$

Head variables

Existential variables

<u>Definition</u>. If **P** is a datalog program,
$\Sigma_{\mathbf{P}}$ is the set of all logical sentences associated to its rules.

Example. Rule: $T(x,y) \text{ :- } R(x,z), T(z,y)$

Sentence: $\forall x. \forall y. \forall z.(R(x,z) \wedge T(z,y) \rightarrow T(x,y))$
$\equiv \forall x. \forall y.(\exists z.R(x,z) \wedge T(z,y) \rightarrow T(x,y))$

Minimal Model Semantics

Definition. A pair (I,J) where I is an EDB and J is an IDB is a *model* for P, if $(I,J) \models \Sigma_{\mathbf{P}}$

Definition. Given an EDB database instance I and a datalog program **P**, the minimal model, denoted J = **P**(I) is a minimal database instance J s.t. $(I,J) \models \Sigma_{\mathbf{P}}$

Theorem. The minimal model always exists, and is unique.

Minimal Model Semantics

Definition. A pair (I,J) where I is an EDB and J is an IDB is a *model* for P, if (I,J) $\models \Sigma_P$

Definition. Given an EDB database instance I and a datalog program **P**, the minimal model, denoted J = **P**(I) is a minimal database instance J s.t. (I,J) $\models \Sigma_P$

Theorem. The minimal model always exists, and is unique.

Example:

1 → 2 → 3 → 4 → 5

```
T(x,y) :- R(x,y)
T(x,y) :- R(x,z), T(z,y)
```

Which of these IDBs are *models*?
Which are *minimal models*?

R=

1	2
2	3
3	4
4	5

T=

1	2
2	3
3	4
4	5
1	3
2	4
3	5

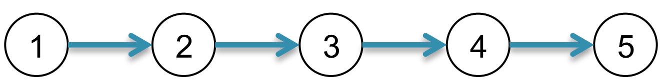
Minimal Model Semantics

Definition. A pair (I,J) where I is an EDB and J is an IDB is a *model* for P, if (I,J) $\models \Sigma_P$

Definition. Given an EDB database instance I and a datalog program **P**, the minimal model, denoted J = **P**(I) is a minimal database instance J s.t. (I,J) $\models \Sigma_P$

Theorem. The minimal model always exists, and is unique.

Example:




```
T(x,y) :- R(x,y)
T(x,y) :- R(x,z), T(z,y)
```

Which of these IDBs are *models*?
Which are *minimal models*?

R=

1	2
2	3
3	4
4	5

T=

1	2
2	3
3	4
4	5
1	3
2	4
3	5

T=

1	2
2	3
3	4
4	5
1	3
2	4
3	5
1	4
2	5
1	5

Minimal Model Semantics

Definition. A pair (I,J) where I is an EDB and J is an IDB is a *model* for P, if (I,J) $\models \Sigma_P$

Definition. Given an EDB database instance I and a datalog program **P**, the minimal model, denoted J = **P**(I) is a minimal database instance J s.t. (I,J) $\models \Sigma_P$

Theorem. The minimal model always exists, and is unique.

Example:

1 → 2 → 3 → 4 → 5

Which of these IDBs are *models*?
Which are *minimal models*?

R=

1	2
2	3
3	4
4	5

```
T(x,y) :- R(x,y)
T(x,y) :- R(x,z), T(z,y)
```

T=

1	2
2	3
3	4
4	5
1	3
2	4
3	5

T=

1	2
2	3
3	4
4	5
1	3
2	4
3	5
1	4
2	5
1	5

T=

1	1
1	2
1	3
1	4
1	5
…	…
…	…
5	4
5	5

All 25 pairs of nodes

Minimal Fixpoint Semantics

Definition. Fix an EDB I, and a datalog program **P**.
The *immediate consequence* operator $T_{\mathbf{P}}$ is defined as follows.
For any IDB J:
$T_{\mathbf{P}}(J)$ = all IDB facts that are immediate consequences from I and J.

Fact. For any datalog program P, the immediate consequence operator is monotone. In other words, if $J_1 \subseteq J_2$ then $T_{\mathbf{P}}(J_1) \subseteq T_{\mathbf{P}}(J_2)$.

Minimal Fixpoint Semantics

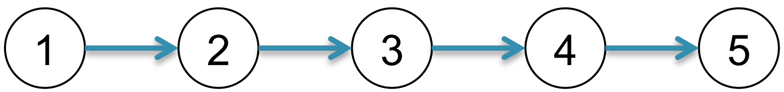
Definition. Fix an EDB I, and a datalog program **P**.
The *immediate consequence* operator $T_{\mathbf{P}}$ is defined as follows.
For any IDB J:
$T_{\mathbf{P}}(J)$ = all IDB facts that are immediate consequences from I and J.

Fact. For any datalog program P, the immediate consequence operator is monotone. In other words, if $J_1 \subseteq J_2$ then $T_{\mathbf{P}}(J_1) \subseteq T_{\mathbf{P}}(J_2)$.

Theorem. The immediate consequence operator has a unique, minimal fixpoint J: $\text{fix}(T_{\mathbf{P}}) = J$, where J is the minimal instance with the property $T_{\mathbf{P}}(J) = J$.

Proof: using Knaster-Tarski's theorem for monotone functions.
The fixpoint is given by:
$\text{fix}(T_{\mathbf{P}}) = J_0 \cup J_1 \cup J_2 \cup \ldots$ where $J_0 = \varnothing$, $J_{k+1} = T_{\mathbf{P}}(J_k)$

Minimal Fixpoint Semantics



T(x,y) :- R(x,y)
T(x,y) :- R(x,z), T(z,y)

R=

1	2
2	3
3	4
4	5

$J_0 = \varnothing$

T =

$J_1 = T_{\mathbf{P}}(J_0)$

1	2
2	3
3	4
4	5

$J_2 = T_{\mathbf{P}}(J_1)$

1	2
2	3
3	4
4	5
1	3
2	4
3	5

$J_3 = T_{\mathbf{P}}(J_2)$

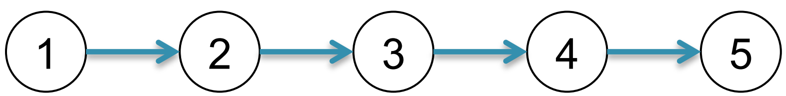
1	2
2	3
3	4
4	5
1	3
2	4
3	5
1	4
2	5

$J_4 = T_{\mathbf{P}}(J_3)$

1	2
2	3
3	4
4	5
1	3
2	4
3	5
1	4
2	5
1	5

Proof Theoretic Semantics

Every fact in the IDB has a *derivation tree*, or *proof tree* justifying its existence.

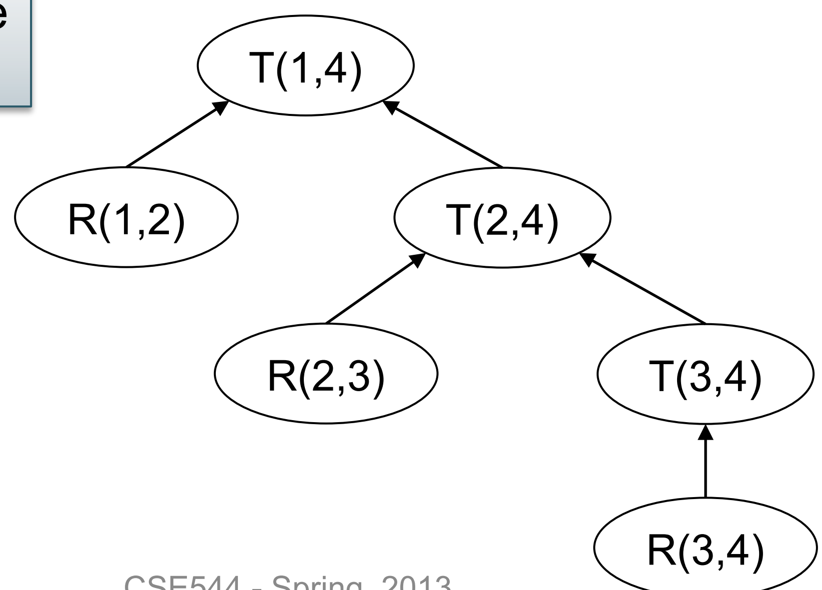


T(x,y) :- R(x,y)
T(x,y) :- R(x,z), T(z,y)

R=

1	2
2	3
3	4
4	5

Derivation tree of T(1,4)



Adding Negation: Datalog$^{\neg}$

Example: compute the complement of the transitive closure

```
T(x,y) :- R(x,y)
T(x,y) :- T(x,z), R(z,y)
CT(x,y) :- Node(x), Node(y), not T(x,y)
```

What does this mean??

Recursion and Negation Don't Like Each Other

EDB: I = { R(a) }

S(x) :- R(x), not T(x)
T(x) :- R(x), not S(x)

Which IDBs are models of **P**?

J_1 = { } J_2 = {S(a)} J_3 = {T(a)} J_4 = {S(a), T(a) }

Recursion and Negation Don't Like Each Other

EDB: $I = \{ R(a) \}$

$$\boxed{\begin{array}{l} S(x) \text{ :- } R(x), \text{ not } T(x) \\ T(x) \text{ :- } R(x), \text{ not } S(x) \end{array}}$$

Which IDBs are models of **P**?

$J_1 = \{ \}$ — No: both rules fail

$J_2 = \{S(a)\}$ — Yes: the facts in J_2 are $R(a), S(a), \neg T(a)$ and both rules are *true*.

$J_3 = \{T(a)\}$ — Yes

$J_4 = \{S(a), T(a)\}$ — Yes

There is no *minimal* model!

Recursion and Negation Don't Like Each Other

EDB: I = { R(a) }

S(x) :- R(x), not T(x)
T(x) :- R(x), not S(x)

Which IDBs are models of **P**?

J_1 = { } — No: both rules fail

J_2 = {S(a)} — Yes: the facts in J_2 are R(a), S(a), ¬T(a) and both rules are *true*.

J_3 = {T(a)} — Yes

J_4 = {S(a), T(a) } — Yes

There is no ***minimal*** model!

There is no minimal fixpoint! (Why does Knaster-Tarski's theorem fail?)

Adding Negation: datalog$^{\neg}$

- **Solution 1: Stratified Datalog$^{\neg}$**
  – Insist that the program be *stratified*: rules are partitioned into strata, and an IDB predicate that occurs only in strata ≤ k may be negated in strata ≥ k+1
- **Solution 2: Inflationary-fixpoint Datalog$^{\neg}$**
  – Compute the fixpoint of $J \cup T_{\mathbf{P}}(J)$
  – Always terminates (why ?)
- **Solution 3: Partial-fixpoint Datalog$^{\neg,*}$**
  – Compute the fixpoint of $T_{\mathbf{P}}(J)$
  – May not terminate

Stratified datalog$^\neg$

A datalog$^\neg$ program is *stratified* if its rules can be partitioned into k strata, such that:

- If an IDB predicate P appears negated in a rule in stratum i, then it can only appear in the head of a rule in strata 1, 2, …, i-1

P:

P_1 :- $body_1$
P_2 :- $body_2$
….

….

P_j :- $body_j$
….

Stratum i

….

….
P_n :- $body_n$

Note: a datalog$^\neg$ program either is stratified or it ain't!

Which programs are stratified?

T(x,y) :- R(x,y)
T(x,y) :- T(x,z), R(z,y)
CT(x,y) :- Node(x), Node(y), not T(x,y)

S(x) :- R(x), not T(x)
T(x) :- R(x), not S(x)

Stratified datalog$^\neg$

- Evaluation algorithm for stratified datalog$^\neg$:

- For each stratum i = 1, 2, …, do:
  - Treat all IDB's defined in prior strata as EBS
  - Evaluate the IDB's defined in stratum i, using either the naïve or the semi-naïve algorithm

Does this compute a minimal model?

```
T(x,y) :- R(x,y)
T(x,y) :- T(x,z), R(z,y)
--------------------------------------------
CT(x,y) :- Node(x), Node(y), not T(x,y)
```

Stratified datalog¬

- Evaluation algorithm for stratified datalog¬:

- For each stratum i = 1, 2, …, do:
  - Treat all IDB's defined in prior strata as EBS
  - Evaluate the IDB's defined in stratum i, using either the naïve or the semi-naïve algorithm

Does this compute a minimal model?

NO:
J_1 = { T = transitive closure, CT = its complement}
J_2 = { T = all pairs of nodes, CT = empty}

T(x,y) :- R(x,y)
T(x,y) :- T(x,z), R(z,y)

CT(x,y) :- Node(x), Node(y), not T(x,y)

Inflationary-fixpoint datalog$^\neg$

Let **P** be any datalog$^\neg$ program, and I an EDB.
Let $T_P(J)$ be the *immediate consequence* operator.
Let $F(J) = J \cup T_P(J)$ be the *inflationary immediate consequence* operator.

Define the sequence: $J_0 = \varnothing$, $J_{n+1} = F(J_n)$, for $n \geq 0$.

Definition. The inflationary fixpoint semantics of **P** is $J = J_n$ where n is such that $J_{n+1} = J_n$

Why does there always exists an n such that $J_n = F(J_n)$?

Find the inflationary semantics for:

```
T(x,y) :- R(x,y)
T(x,y) :- T(x,z), R(z,y)
CT(x,y) :- Node(x), Node(y), not T(x,y)
```

```
S(x) :- R(x), not T(x)
T(x) :- R(x), not S(x)
```

Inflationary-fixpoint datalog¬

- Evaluation for Inflationary-fixpoint datalog¬
- Use the naïve, of the semi-naïve algorithm
- Inhibit any optimization that rely on monotonicity (e.g. out of order execution)

Partial-fixpoint datalog$^{\neg,*}$

Let **P** be any datalog$^{\neg}$ program, and I an EDB.
Let $T_{\mathbf{P}}(J)$ be the *immediate consequence* operator.

Define the sequence: $J_0 = \varnothing$, $J_{n+1} = T_{\mathbf{P}}(J_n)$, for $n \geq 0$.

Definition. The partial fixpoint semantics of **P** is $J = J_n$ where n is such that $J_{n+1} = J_n$, if such an n exists, undefined otherwise.

Find the partial fixpoint semantics for:

T(x,y) :- R(x,y)
T(x,y) :- T(x,z), R(z,y)
CT(x,y) :- Node(x), Node(y), not T(x,y)

S(x) :- R(x), not T(x)
T(x) :- R(x), not S(x)

Note: there may not exists an n such that $J_n = F(J_n)$

Discussion

- Which semantics does Daedalus adopt?

Discussion

Comparing datalog$^\neg$

- Compute the complement of the transitive closure in inflationary datalog$^\neg$
- Compare the expressive power of:
  - Stratified datalog$^\neg$
  - Inflationary fixpoint datalog$^\neg$
  - Partial fixpoint datalog$^\neg$