# LAB on OJ

– Exact Inference

- http://10.88.3.60/
- 1003 Exact Inference

# Review on exact inference algorithm

- Example: MaryCall, JohnCall in Ch14

- Enumeration algorithm:
  - Step 1: Select the entries consistent with the evidence
  - Step 2: Sum out *hidden vars* to get joint of Query and evidence
  - Step 3: Normalize

- Elimination algorithm:
  - Make factors
  - Join all *factors* and eliminate all *hidden vars*.

# Review on exact inference algorithm

- Example: MaryCall, JohnCall in Ch14

- Enumeration algorithm:
  - Def enumeration_ask(X, e, bn):
  - Def enumeration_all(X, e, bn):

- Elimination algorithm:
  - Def elimination_ask(X, e, bn):

# Problem description – sample in

P(Earthquake = -)

P(Burglary = + | John = +, Mary = +)

******

Burglary

0.001

***

Earthquake

0.002

***

Alarm | Burglary Earthquake

0.95 + +

0.94 + -

0.29 - +

0.001 - -

***

John | Alarm

0.9 +

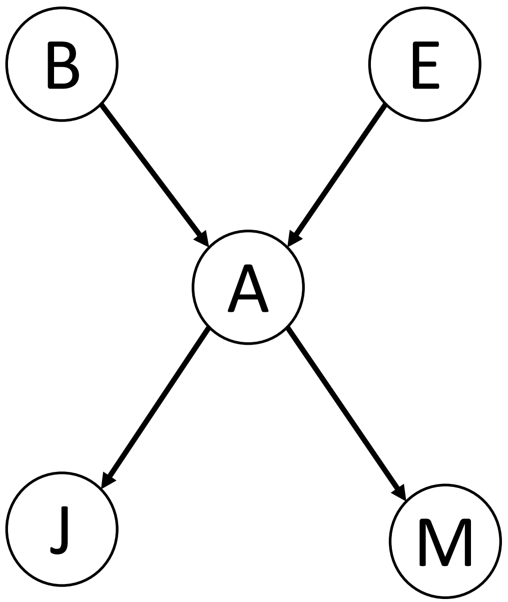
0.05 -

***

Mary | Alarm

0.7 +

0.01 –



# Problem description – sample out

probability by enumeration:  1.0

probability by elimination:  1.0

**********

probability by enumeration:  0.28

probability by elimination:  0.28

**********

# Classes

```
class BayesNet:
    def __init__(self, node_specs=[]):
    def add(self, node_spec):
    def variable_node(self, var):
    def variable_values(self, vars):
```

*used in building the BayesNet*

```
class BayesNode:
    """A conditional probability distribution for a boolean variable, P(X | parents).
Part of a BayesNet."""
    def __init__(self, x, parents, cpt):
    def p(self, value, event):
```

*used in building the BayesNet*

# Classes

```
class ProbDist:                    used in the computation for probability distribution
    def __init__(self, varname='?', freqs=None):
    def normalize(self):

class Factor:                      used in elimination algorithm
    def __init__(self, variables, cpt):
    def pointwise_product(self, other):
    def sum_out(self, var):
    def p(self, e):
    def normalize(self):
```

# Procedure

- First read the input and *bulid bn = BayesNet()*

  by *bn.add((node, parents, cpt))*
- Discuss (1) joint/ marginal distribution

  (2) conditional distribution

- (1): def joint_probability():

  Call *enumerate_all*() and *elimination_ask*() respectively

- (2): def conditional_probability():

  Call *enumerate_ask*() and *elimination_ask*() respectively

# pseudocode

- Enumeration algorithm

**function** ENUMERATION-ASK($X$, $\mathbf{e}$, $bn$) **returns** a distribution over $X$
  **inputs**: $X$, the query variable
    $\mathbf{e}$, observed values for variables $\mathbf{E}$
    $bn$, a Bayes net with variables $\{X\} \cup \mathbf{E} \cup \mathbf{Y}$  /* $\mathbf{Y}$ = *hidden variables* */

  $\mathbf{Q}(X) \leftarrow$ a distribution over $X$, initially empty
  **for each** value $x_i$ of $X$ **do**
    $\mathbf{Q}(x_i) \leftarrow$ ENUMERATE-ALL($bn$.VARS, $\mathbf{e}_{x_i}$)
      where $\mathbf{e}_{x_i}$ is $\mathbf{e}$ extended with $X = x_i$
  **return** NORMALIZE($\mathbf{Q}(X)$)

---

**function** ENUMERATE-ALL($vars$, $\mathbf{e}$) **returns** a real number
  **if** EMPTY?($vars$) **then return** 1.0
  $Y \leftarrow$ FIRST($vars$)
  **if** $Y$ has value $y$ in $\mathbf{e}$
    **then return** $P(y \mid parents(Y)) \times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}$)
    **else return** $\sum_y P(y \mid parents(Y)) \times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}_y$)
      where $\mathbf{e}_y$ is $\mathbf{e}$ extended with $Y = y$

**Figure 14.9** The enumeration algorithm for answering queries on Bayesian networks.

# pseudocode

- Elimination algorithm

```
function ELIMINATION-ASK(X, e, bn) returns a distribution over X
  inputs: X, the query variable
          e, observed values for variables E
          bn, a Bayesian network specifying joint distribution P(X_1, ..., X_n)

  factors ← [ ]
  for each var in ORDER(bn.VARS) do
     factors ← [MAKE-FACTOR(var, e)|factors]
     if var is a hidden variable then factors ← SUM-OUT(var, factors)
  return NORMALIZE(POINTWISE-PRODUCT(factors))
```

**Figure 14.10** The variable elimination algorithm for inference in Bayesian networks.