University of Central Lancashire
UCLan

CO2412
Computational Thinking

Travelling salesman (tutorial 3.5)
Parse trees (tutorial 4.1)
Propositional logic expressivity (tutorial 4.2)
Enumeration of truth tables (tutorial 4.3)

Where opportunity creates success

Travelling salesman: Tutorial 3.5 problem

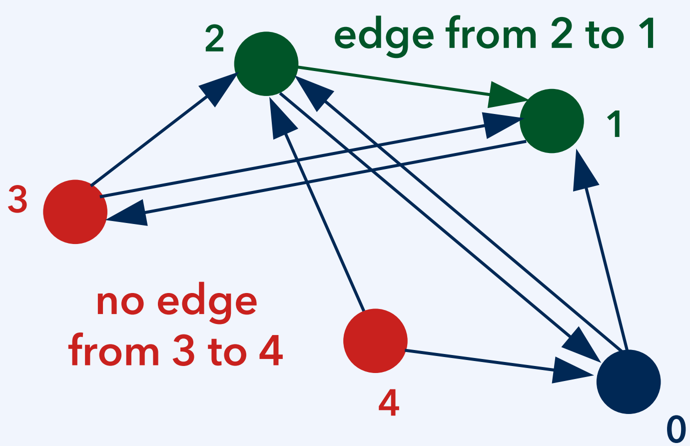
University of Central Lancashire
UCLan

CO2412

8th March 2022

Adjacency matrix data structure

Matrix-like data structures in Python include lists of lists (*i.e.*, 2D dynamic arrays), if the numpy library is used, two-dimensional static arrays. For graphs, the most relevant data structure of this type is the **adjacency matrix**.




```
self._adj_matrix = [ [0, 1, 1, 0, 0],
                     [0, 0, 0, 1, 0],
                     [1, 1, 0, 0, 0],
                     [0, 1, 1, 0, 0],
                     [1, 0, 1, 0, 0] ]
```

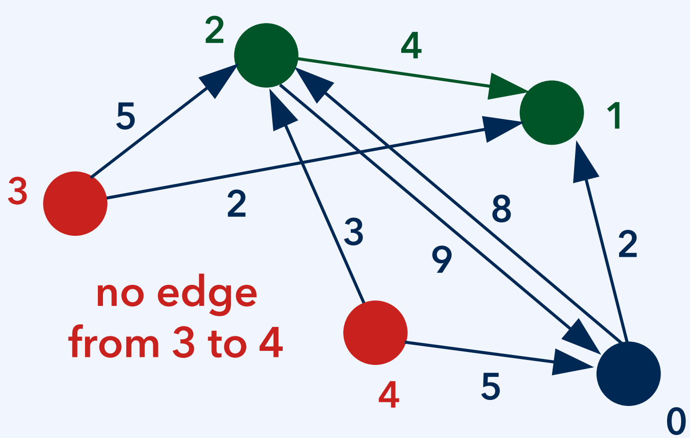
For a sparse graph, the majority of entries in the 2D array/matrix is zero. Adjacency matrices are commonly used when expecting a **dense graph**.

self._adj_matrix[2][1] = **1**, or **True**

self._adj_matrix[3][4] = **0**, or **False**

Adjacency matrix data structure

For a graph with labelled edges, the adjacency matrix contains edge labels.
In the case of a weighted graph, the labels represent the length of the edges.
If these edges are travel distances, the diagonal entries should be zero.




```
self._adj_matrix = [ [0, 2, 8, ∞, ∞],
                     [∞, 0, ∞, ∞, ∞],
                     [9, 4, 0, ∞, ∞],
                     [∞, 2, 5, 0, ∞],
                     [5, ∞, 3, ∞, 0] ]
```

For a sparse graph, the majority of entries in the 2D array/matrix is infinity. Adjacency matrices are commonly used when expecting a **dense graph**.

self._adj_matrix[1][1] = **0**

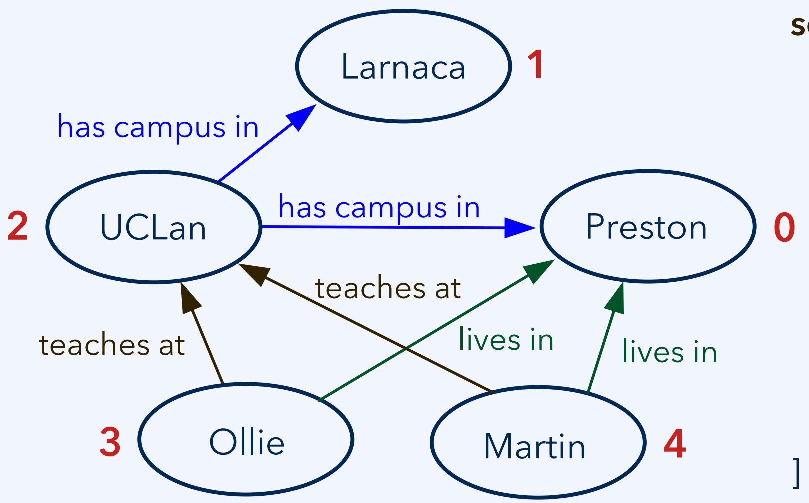
self._adj_matrix[2][1] = **1**

self._adj_matrix[3][4] = **∞**

Adjacency matrix data structure

For a graph with labelled edges, the adjacency matrix contains edge labels.

Task 3.5.1c: Adaptation to the assessment problem.




```
self._adj_matrix = [
    [None,            None,          None,           None,     None],
    [None,            None,          None,           None,     None],
    ["has campus in", "has campus in", None,         None,     None],
    ["lives in",      None,          "teaches at",   None,     None],
    ["lives in",      None,          "teaches at",   None,     None]
]
```

```
self._node_labels = [
    "Preston",
    "Larnaca",
    "UCLan",
    "Ollie",
    "Martin"
]
```

For a sparse graph, the majority of entries in the 2D array/matrix is None. Adjacency matrices are commonly used when expecting a dense graph.

Adjacency matrix data structure

Task 3.5.1b: Implement calculation of the length of a path.

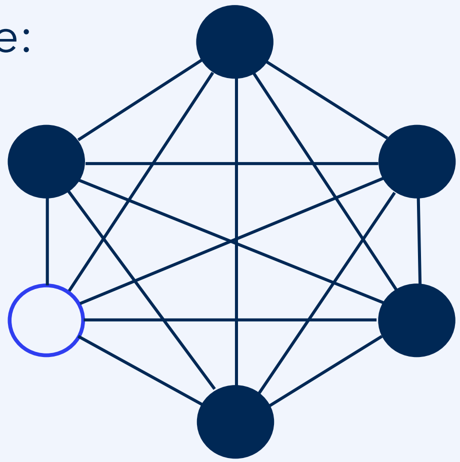
```
# returns weight of the edge between i and j
#
def get_weight(self, i, j):
    return self._adj_matrix[i][j]

# the path is a list of node IDs
#
def get_length_of_path(self, path):
    length = 0
    for i in range(len(path) - 1):
        length += self.get_weight(path[i], path[i+1])
    return length
```

Path given by a list of traversed nodes. The length of the path is:

```
_adj_matrix[ path[0] ][ path[1] ]
    + _adj_matrix[ path[1] ][ path[2] ]
    + _adj_matrix[ path[2] ][ path[3] ]
    ...
    + _adj_matrix[ path[len(path) - 2] ]
            ][ path[len(path) - 1] ]
```

Travelling salesman problem (TSP)

Example:
$n = 6$



How many cycles covering all nodes are there in a **complete graph** with n nodes, that is a graph where every node is adjacent to every other node?

Task 3.5.2a: No. of Hamilton cycles

A travelling salesman needs to visit all the cities, by a path that ends at the same city where it starts (a **cycle**).

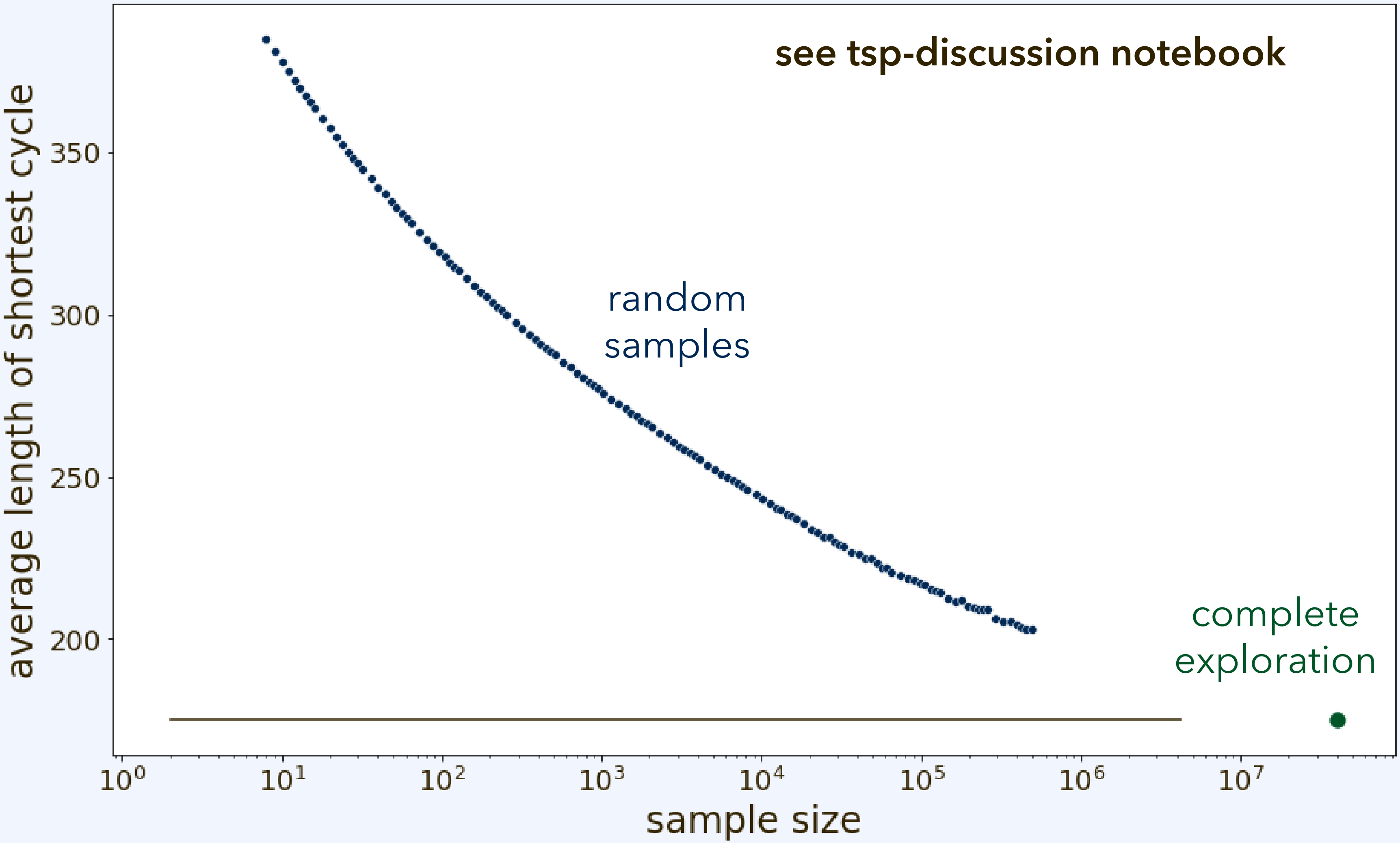
No city may be visited twice. Every city must be visited exactly once. (Except for returning to the start.) These cycles are **Hamilton cycles**.

Assume that the initial/final node is fixed; let it be the node no. 0.

No. of Hamilton cycles in a complete graph: $(n - 1) \cdot (n - 2) \cdot \ldots \cdot 2 \cdot 1 = (n - 1)!$

$n = 12$ nodes: $11! = 39.9$ million cycles; $n = 15$ nodes: $14! = 87.2$ billion cycles.

TSP: Randomized approximation algorithm



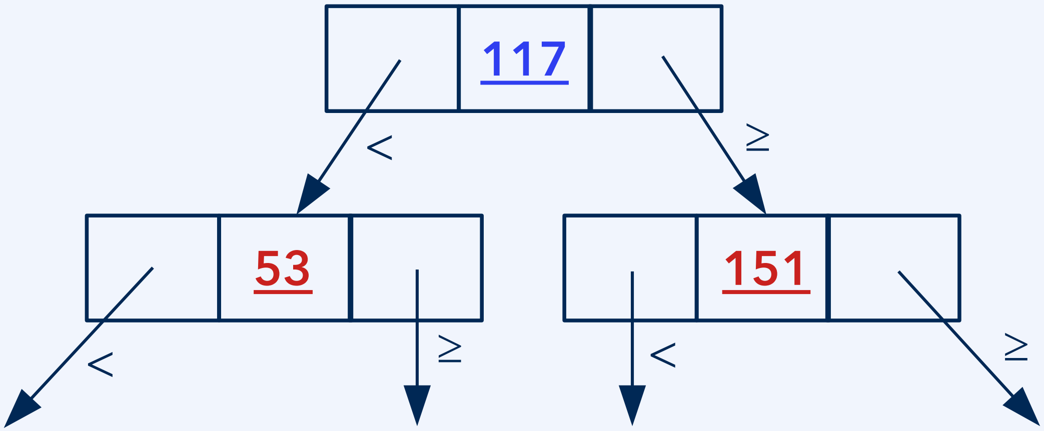
Parse trees: Tutorial 4.1 problem

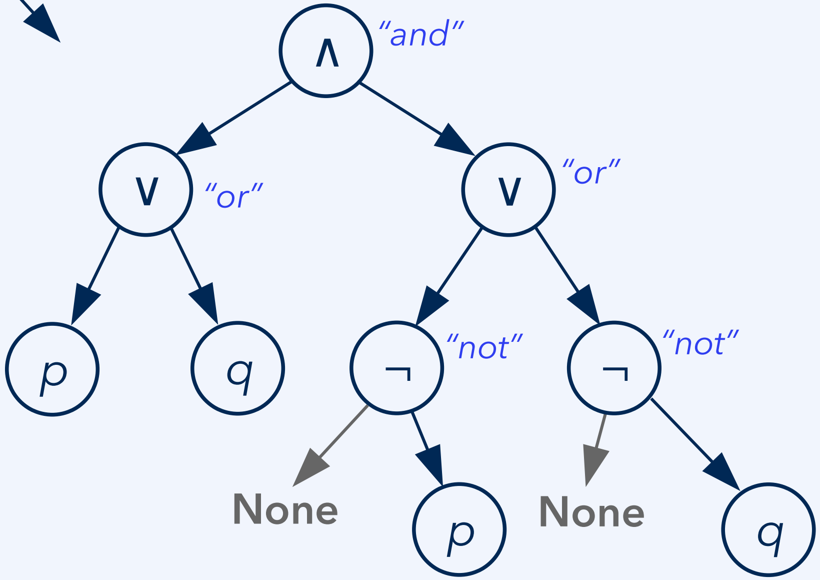
University of Central Lancashire
UCLan

CO2412

8th March 2022

Parse trees for propositional logic statements





see parse-tree-discussion notebook

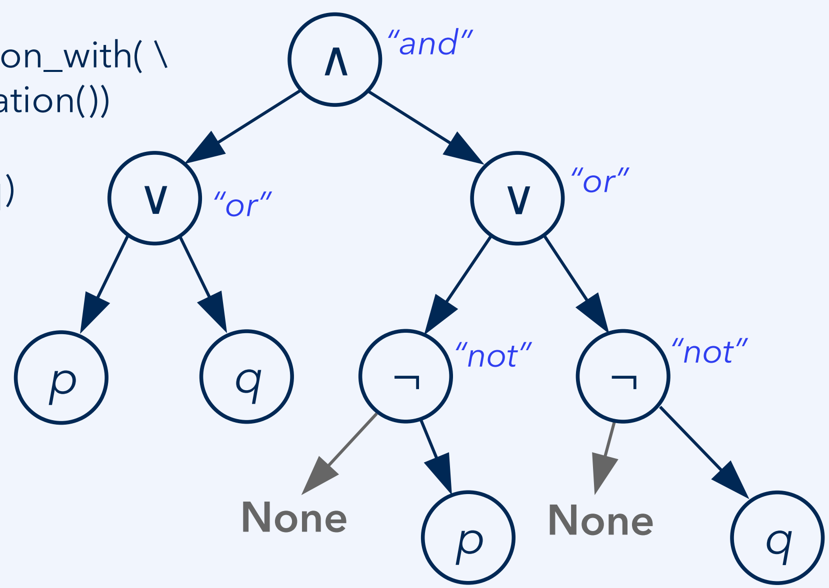
Parse trees for propositional logic statements

Task 4.1b: **Create a Propositional object for the statement $((p \vee q) \wedge (\neg p \vee \neg q))$ and print its truth table.**

"(p or q) and (not-p or not-q)"

$$(p \vee q) \wedge (\neg p \vee \neg q)$$

```
p_or_q = Propositional("p").disjunction_with( \
                              Propositional("q"))
not_p_or_not_q = Propositional("p").negation().disjunction_with( \
                              Propositional("q").negation())

statement_S = p_or_q.conjunction_with(not_p_or_not_q)
```




```
Truth table of ((p or q) and (not p or not q))

False(p)        False(q)        <>        False
False(p)         True(q)        <>        True
 True(p)        False(q)        <>        True
 True(p)         True(q)        <>        False

True 2 times, false 2 times, undefined 0 times
```

Parse trees for propositional logic statements

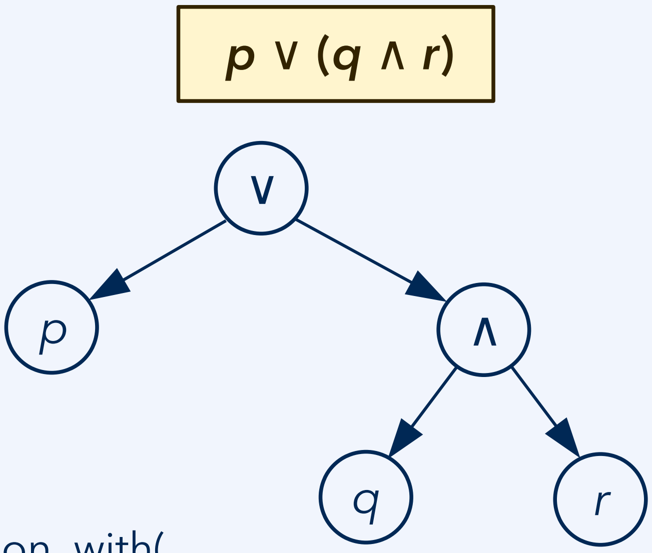
Task 4.1e: Create a Propositional object that generates the truth table …

p	*q*	*r*	**specification**	$q \wedge r$	$p \vee (q \wedge r)$
False	False	False	False	False	False
False	False	True	False	False	False
False	True	False	False	False	False
False	True	True	True	**True**	True
True	False	False	True	False	True
True	False	True	True	False	True
True	True	False	True	False	True
True	True	True	True	**True**	True

Parse trees for propositional logic statements

Task 4.1e: Create a Propositional object that generates the truth table …

p	q	r	$p \lor (q \land r)$
False	False	False	False
False	False	True	False
False	True	False	False
False	True	True	True
True	False	False	True
True	False	True	True
True	True	False	True
True	True	True	True




```
statement_T = Propositional("p").disjunction_with(
    Propositional("q").conjunction_with(Propositional("r"))
)
```

Parse trees for propositional logic statements

Task 4.1c: What is the truth table for the statement ($p \wedge q$) ↔ ($q \wedge r$)?

p	q	r	$p \wedge q$	$q \wedge r$	$(p \wedge q) \leftrightarrow (q \wedge r)$
False	False	False	False	False	**True**
False	False	True	False	False	**True**
False	True	False	False	False	**True**
False	True	True	False	True	False
True	False	False	False	False	**True**
True	False	True	False	False	**True**
True	True	False	True	False	False
True	True	True	True	True	**True**

Parse trees for propositional logic statements

Task 4.1d: Implement the logical equivalence (biconditionality) operator.

```python

        ############
        # NEW CODE #
        ############
        #
        elif self._item == "<->":
            if left_evaluation == "undefined" or right_evaluation == "undefined":
                return "undefined"
            else:
                return (left_evaluation == right_evaluation)

```

In [3]:

```python
# Task 4.1c/d: What is the truth table for the statement (p ∧ q) ↔ (q ∧ r)?
#
p_and_q = Propositional("p").conjunction_with(Propositional("q"))
q_and_r = Propositional("q").conjunction_with(Propositional("r"))
statement_R = p_and_q.biconditional_with(q_and_r)

print("Truth table of", statement_R.to_string(), "\n")
(t, f, u) = statement_R.print_whole_truth_table()
print("\nTrue", t, "times, false", f, "times, undefined", u, "times")
```

```
Truth table of ((p and q) <-> (q and r))

False(p)        False(q)        False(r)        <>      True
False(p)        False(q)         True(r)        <>      True
False(p)         True(q)        False(r)        <>      True
False(p)         True(q)         True(r)        <>      False
 True(p)        False(q)        False(r)        <>      True
 True(p)        False(q)         True(r)        <>      True
 True(p)         True(q)        False(r)        <>      False
 True(p)         True(q)         True(r)        <>      True

True 6 times, false 2 times, undefined 0 times
```

Logic expressivity: Tutorial 4.2 problem

CO2412

8th March 2022

False for *n* valuations: Complexity of the problem

Problem 4.2: Create a statement that becomes False for *n* valuations.

What is the **complexity of the problem**, *i.e.*, the best possible asymptotic efficiency of an algorithm that solves it? Let us establish some **lower bounds**:

- For n False entries in the truth table, the size of the truth table must be at least n. Therefore, $\boldsymbol{m = O(\log n)}$ **atomic statements** are needed.

  Example: $n = 37$; truth table size: 64; no. of atomic statements: $m = 6$.

- Space for **encoding one atomic statement: O(log *m*)** = O(log log *n*).

  Example: $n = 2^{10,000}$; $m = 10,000$; atomic statements $p_0, \ldots, p_{9998}, p_{9999}$.

  Remark: Each atomic statement must occur (be written) at least once …

False for n valuations: Complexity of the problem

Problem 4.2: Create a statement that becomes False for n valuations.

What is the **complexity of the problem**, *i.e.*, the best possible asymptotic efficiency of an algorithm that solves it? Let us establish some **lower bounds**:

- For n False entries in the truth table, the size of the truth table must be at least n. Therefore, **m = O(log n) atomic statements** are needed.

  Example: $n = 37$; truth table size: 64; no. of atomic statements: $m = 6$.

- Space for **encoding one atomic statement: O(log m)** = O(log log n).

  Example: $n = 2^{10{,}000}$; $m = 10{,}000$; atomic statements $p_0, \ldots, p_{9998}, p_{9999}$.

- Space & time **for the whole statement: O(m log m)** = O(log $n \cdot$ log log n).

False for n valuations: Complexity of the problem

Problem 4.2: Create a statement that becomes False for n valuations.

What is the **complexity of the problem**, *i.e.*, the best possible asymptotic efficiency of an algorithm that solves it? $m = O(\log n)$ atomic statements needed.

Lower bound: Requires at least $O(m \log m) = O(\log n \cdot \log \log n)$ space & time.

Is there an algorithm that solves the problem in $O(\log n \cdot \log \log n)$ time? **Yes.**

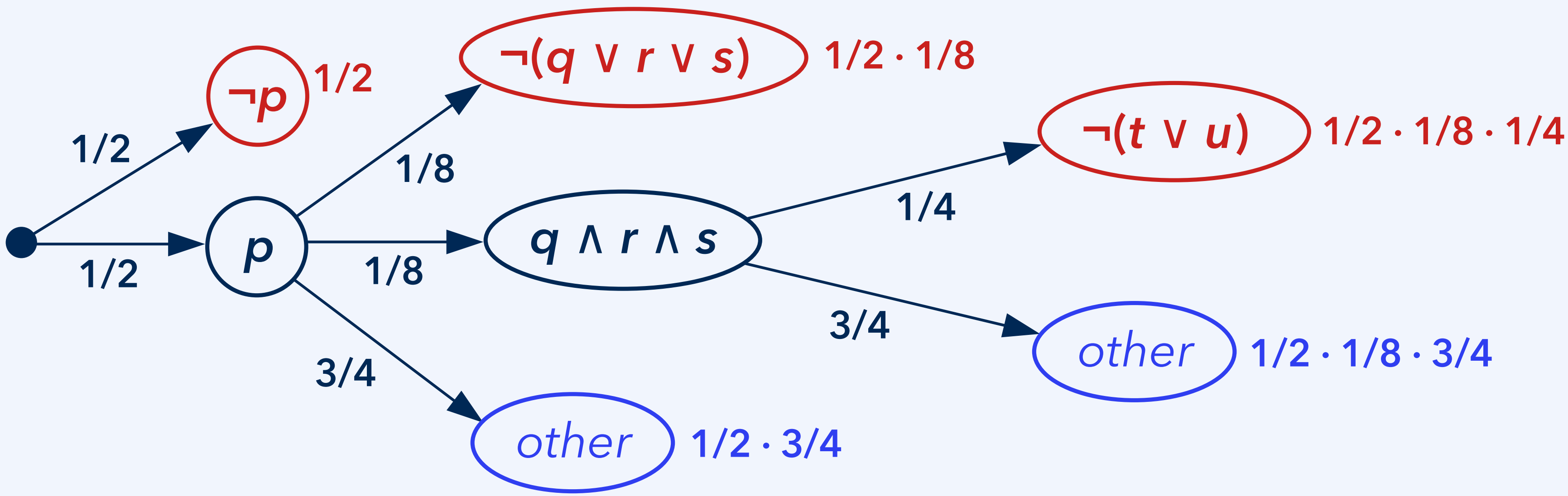
Example: $n = 37$; statement must be False for 37 out of 64 valuations:

$$\frac{37}{64} = \frac{32}{64} + \frac{4}{64} + \frac{1}{64} = \frac{1}{2} + \left(\frac{1}{2} \cdot \frac{1}{8}\right) + \left(\frac{1}{2} \cdot \frac{1}{8} \cdot \frac{1}{4}\right)$$

$$37 = (\underset{32}{\mathbf{1}}\ 0\ 0\ \underset{4}{\mathbf{1}}\ 0\ \underset{1}{\mathbf{1}})_2$$

False for n valuations: Example, $n = 37$

Problem 4.2: Create a statement that becomes False for *n* valuations.



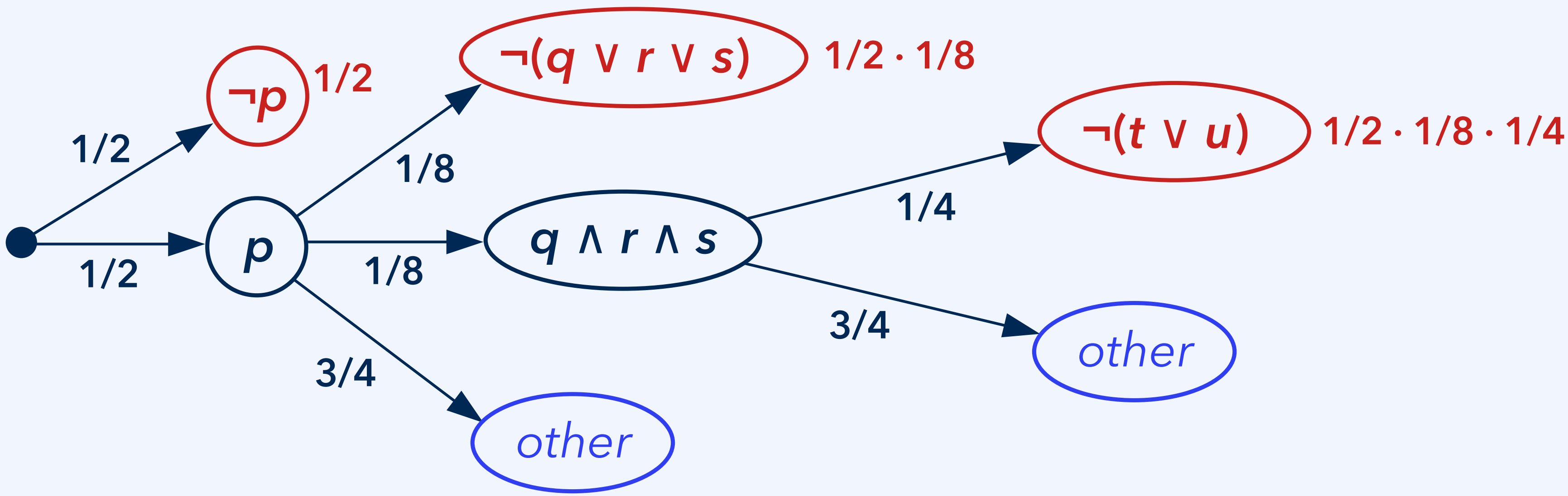
Example: $n = 37$; statement must be **False for 37** out of 64 valuations:

$$\frac{37}{64} = \frac{32}{64}+\frac{4}{64}+\frac{1}{64} = \frac{1}{2} + \left(\frac{1}{2}\cdot\frac{1}{8}\right) + \left(\frac{1}{2}\cdot\frac{1}{8}\cdot\frac{1}{4}\right)$$

$$37 = (\underset{32}{1}\;\;0\;\;0\;\;\underset{4}{1}\;\;0\;\;\underset{1}{1})_2$$

False for n valuations: Example, n = 37

Problem 4.2: Create a statement that becomes False for n valuations.



$$p \;\wedge\; (q \vee r \vee s) \;\wedge\; ((q \wedge r \wedge s) \rightarrow (t \vee u))$$

$$\boxed{p_0 \;\wedge\; (p_1 \vee p_2 \vee p_3) \;\wedge\; ((p_1 \wedge p_2 \wedge p_3) \rightarrow (p_4 \vee p_5))}$$

$$37 = (\,\mathbf{1}\;\;0\;\;0\;\;\mathbf{1}\;\;0\;\;\mathbf{1}\,)_2$$

$$\mathbf{0}\quad 1\text{–}2\text{–}\mathbf{3}\quad 4\text{–}\mathbf{5}$$

Number of truth tables: Tutorial 4.3 problem

CO2412

8th March 2022

Satisfiability of propositional logic statements

Question 4.3.1: Are the following propositional logic statements satisfiable?

$S_a \quad = \quad ((p \wedge r) \rightarrow (q \vee r)) \leftrightarrow (s \wedge \neg s)$

False

$S_b \quad = \quad (p \vee \neg q \vee \neg r \vee \neg s) \wedge ((p \wedge r) \rightarrow (q \vee r)) \wedge \neg p \wedge q \wedge r \wedge s$

$\boldsymbol{p \vee \neg q \vee \neg r \vee \neg s \equiv \neg(\neg p \wedge q \wedge r \wedge s)}$

$S_c \quad = \quad (p \leftrightarrow \neg p) \rightarrow ((p \leftrightarrow \neg q) \wedge (q \leftrightarrow \neg r) \wedge (r \leftrightarrow \neg s) \wedge (s \leftrightarrow \neg t) \wedge (t \leftrightarrow \neg p))$

False

$S_d \quad = \quad \neg S_c$

Satisfiability of propositional logic statements

Question 4.3.1: Are the following propositional logic statements satisfiable?

$S_a \;=\; ((p \wedge r) \rightarrow (q \vee r)) \leftrightarrow (s \wedge \neg s)$

True **False**

If $p \wedge r$ is True, r is True, and hence $q \vee r$ is also True.

$(p \wedge r) \rightarrow (q \vee r)$ is a tautology
$(s \wedge \neg s)$ is a contradiction

S_a is a contradiction

$S_b \;=\; (p \vee \neg q \vee \neg r \vee \neg s) \wedge ((p \wedge r) \rightarrow (q \vee r)) \wedge \neg p \wedge q \wedge r \wedge s$

There is no valuation for which $(p \vee \neg q \vee \neg r \vee \neg s)$ and $(\neg p \wedge q \wedge r \wedge s)$ are both True. Therefore, **S_b is a contradiction**.

$S_c \;=\; (p \leftrightarrow \neg p) \rightarrow ((p \leftrightarrow \neg q) \wedge (q \leftrightarrow \neg r) \wedge (r \leftrightarrow \neg s) \wedge (s \leftrightarrow \neg t) \wedge (t \leftrightarrow \neg p))$

False

An implication can only become False if the left-hand side is True. That is impossible here. **S_c is a tautology** (and, hence, **satisfiable**).

$S_d \;=\; \neg S_c$

Since S_c is a tautology, **$\neg S_c$ is a contradiction**.

Enumeration of truth tables

Question 4.3.2a: Are there ≥1000 truth tables with four atomic statements?

Let us consider the case where there are $n = 2$ atomic statements, p and q:

p	q	T_0	T_1	T_2	T_3	…	T_{14}	T_{15}
False	False	False	False	False	False	…	True	True
False	True	False	False	False	False	…	True	True
True	False	False	False	True	True	…	True	True
True	True	False	True	False	True	…	False	True

For n atomic statements, there are 2^n different valuations; therefore, there are 2^n entries – each of which may be True or False – in the truth tables.

The total number of different truth tables is therefore $2^{(2^n)}$.

$n = 2$: there are $2^4 = 16$ truth tables; $n = 4$: there are $2^{16} = 65{,}536$ truth tables.

Enumeration of truth tables

Question 4.3.2a: Are there ≥1000 truth tables with four atomic statements? Yes!

p	q	r	s	T_0	T_1	T_2	…	T_{65535}
False	False	False	False	False	False	False	…	True
False	False	False	True	False	False	False	…	True
False	False	True	False	False	False	False	…	True
False	False	True	True	False	False	False	…	True
False	True	False	False	False	False	False	…	True
False	True	False	True	False	False	False	…	True
False	True	True	False	False	False	False	…	True
…	…	…	…	…	…	…	…	…
True	True	False	True	False	False	False	…	True
True	True	True	False	False	False	True	…	True
True	True	True	True	False	True	False	…	True

In general, there are $2^{(2^n)}$ truth tables; $n = 4$: there are $2^{16} = 65{,}536$ truth tables.

Enumeration of truth tables

Task 4.3.2b: Find two (semantically) different statements both entailed by *R*:

There, the premise R was given by $\neg((p \rightarrow q) \rightarrow (\neg q \wedge \neg r))$.

Entailment

$$\boldsymbol{R \vDash S}$$

("The premise R entails the conclusion S")

All models of *R* are also models of *S*.

S may still be True where R is False, *i.e.*, S may have more models than R.

The statement $R \rightarrow S$ is a tautology.

Two statements entailed by R:

$$\boldsymbol{R \vDash R}$$

$$\boldsymbol{R \vDash \textbf{True}}$$

Enumeration of truth tables

Question 4.3.2c: How many propositional logic statements S, using p, q, and r only, with different truth tables exist, such that $R \vDash S$?

There, the premise R was given by $\neg((p \rightarrow q) \rightarrow (\neg q \land \neg r))$.

p	q	r	$(p \rightarrow q)$	$(\neg q \land \neg r)$	$((p \rightarrow q) \rightarrow (\neg q \land \neg r))$	R
False	**False**	**False**	True	True	True	False
False	False	True	**True**	**False**	False	True
False	True	False	**True**	**False**	False	True
False	True	True	**True**	**False**	False	True
True	**False**	**False**	False	True	True	False
True	**False**	True	False	False	True	False
True	True	False	**True**	**False**	False	True
True	True	True	**True**	**False**	False	True

All the True entries in the truth table of R must also be True for S.

Enumeration of truth tables

Question 4.3.2c: How many propositional logic statements S, using p, q, and r only, with different truth tables exist, such that $R \vDash S$?

There, the premise R was given by $\neg((p \to q) \to (\neg q \wedge \neg r))$.

p	q	r	$(p \to q)$	$(\neg q \wedge \neg r)$	$((p \to q) \to (\neg q \wedge \neg r))$	R
False	**False**	**False**	True	True	**True**	**False**
False	False	True	**True**	**False**	False	True
False	True	False	**True**	**False**	False	True
False	True	True	**True**	**False**	False	True
True	**False**	**False**	False	True	**True**	**False**
True	**False**	True	False	False	**True**	**False**
True	True	False	**True**	**False**	False	True
True	True	True	**True**	**False**	False	True

All the True entries in the truth table of R must also be True for S.
There are **3 False entries** in the truth table of R; they may be True or False for S.

Therefore, there are $2^3 = 8$ semantically different $S_0, \ldots, S_7$ entailed by R.

University of Central Lancashire
UCLan

CO2412
Computational Thinking

Travelling salesman (tutorial 3.5)
Parse trees (tutorial 4.1)
Propositional logic expressivity (tutorial 4.2)
Enumeration of truth tables (tutorial 4.3)

Where opportunity creates success