University of London
Computing and Information Systems/ Creative Computing
CO3310 Artificial Intelligence
Coursework assignment 1 2020-2021

Question 1 - Search

a) Compare the following search strategies in terms of their time and space complexity. **[**8 marks**]**

i) Breadth-first
ii) Depth-first
iii) Depth-first with iterative deepening
iv) Greedy search

Time complexity - How long does the search method take to reach a solution.
Space complexity - How much memory is used to store states and information during the search.

Recap : In general, algorithms evaluated in O notation. Fastest to slowest ('best' to 'worst') : O(1) O(log(n)) O(n) O(n(log(n)) O(n^m) O(m^n) O(n!)

(note. for the following, *b* = branching factor, *m* = maximum depth (path length), *d* = minimum solution depth in the search tree).

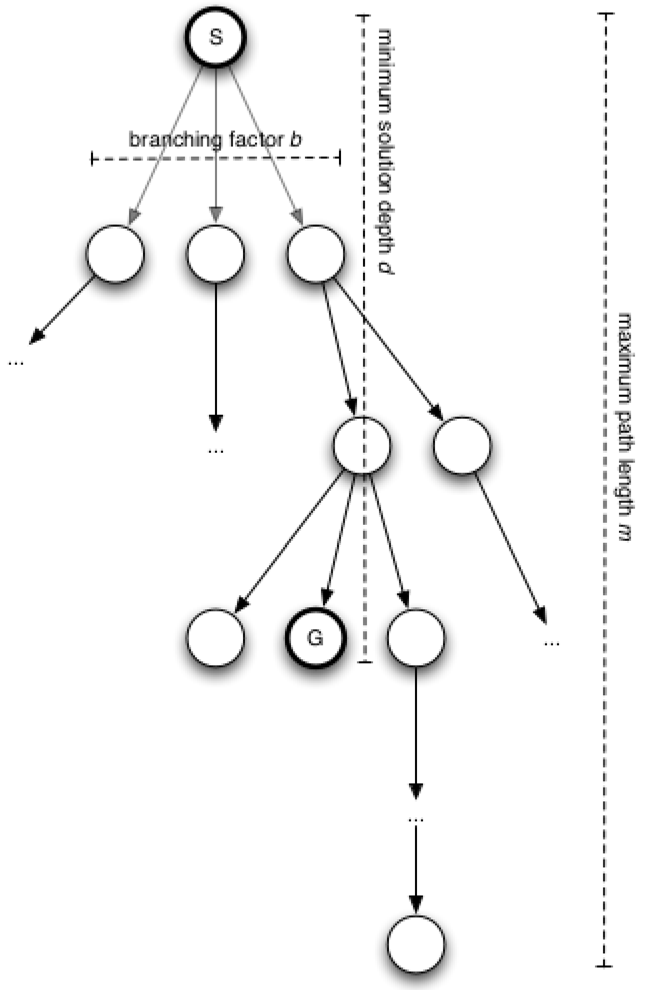


Figure 3.2: Quantities useful in analysing the complexity of searches. *S* is the start state, and *G* is the goal state. Omitted branches indicated by '...' in the diagram.

Figure 1. (Konidaris, 2013, p.11).

Breadth-first search algorithm has a **time** and **space complexity** of O(b^d+1) in the worst case.
Both **breadth-first** and **depth-first** start by adding the start node '*S'* to a queue. Then take that node, check if it's children are the solution and add the children as nodes to the queue if they are not the goal node '*G',* and so on.
Breadth-first will then select the node from the queue with the **smallest** depth first. This will ensure all nodes at each level '*k'* are expanded before nodes at the next level '*k+1'* are expanded.

This approach will find the **shortest path** to the solution, but will store a lot of the tree in memory to do so as well as increasing in duration exponentially. This is because in the worst case that the last node at level *'d'* is the goal node '*G'* taking O(b^d+1) amount of **time** to find (+1 for the root note), and every node up to that node must be kept in memory if the search is to continue, also taking up O(b^d+1) amount of **space** in memory.

Breadth-first's time and space complexity is exponential for finding solutions, solutions that are deep in trees take a lot of time compared to other search algorithms and cannot discover longer paths to a goal at a smaller cost.

Depth-first, in comparison, expands nodes from the queue with the deepest depth. That node's children are all checked to be the solution before the nodes adjacent 'sibling' children are. The full depth of a node is expanded before another branch.
This approach has the improvement over **depth-first** that less of the tree is stored in memory at a time as after the path is expanded it does not need to be stored. Giving its **space complexity** as O(bm).
However, exploring a path to it's depth for the solution may not be as fast as searching shorter paths to a solution first. Giving the worst case **time complexity** as an exponential O(b^m) '*m'* the maximum depth over **breadth-first's** *'d'* minimum depth.
Depth-first will be 'complete' if '*m'* and *'b'* are finite where sometimes they aren't, but is never 'optimal'.

Depth-first with **iterative deepening**, performs a **depth-first** search to a maximum depth of 1 for all nodes in the queue and if the solution is not found in the children at that level the maximum depth is increased by 1, and so on. This approach has a **time complexity** of O(b^d) (faster than breadth-first and depth-first) and a **space complexity** of O(bd) (lower than both breadth and depth-first searches).

It is a hybrid approach of **breadth-first** and **depth-first** to combine the small memory footprint of **depth-first** with the **speed** of **breadth-first** as well as other non time/ space complexity advantages.

Greedy search algorithm is similar to **Depth-first** search that follows a single path to be expanded for the solution before searching another path, but uses a **heuristic** function to select the next node in the path that is likely to lead to the goal with a smaller path cost.

Heuristic functions output an estimate, so that the “quality of our search [and therefore time and space complexities] depends on the quality of [the] heuristic [function]” (Konidaris, 2013, p.14) making their variety difficult for comparison.
Because of similarity to **depth-first** search large sections of the tree are kept in memory to be unexpanded later but the time complexity can be improved.

“The worst-case **time** and **space complexity** is $O(b^m)$ … [and] can be substantially reduced with a good heuristic function” (Russell and Norvig, 2003, p.97) as heuristic functions can be more or less successful in their estimates of likely steps in a path likely to reach a solution reducing the amount of time and tree needed to be kept in memory.

Because of the variety of greedy algorithms and their heuristics they can out perform breadth-first, depth-first and depth-first iterative deepening search algorithms in certain settings but worst case $O(b^m)$ complexities place in comparison to the O() mentioned earlier in this answer.

b) Explain in your own words the difference between **tree search** and **graph search**. **[**6 marks**]**

The difference between **tree-search** and **graph-search** is the way that they traverse a graph to discover the path to the solution, the **traversal pattern,** that is applied to the state space.
Their difference does not come from the type of graph they search, ie. a tree (that by definition contains no cycles or a ‘graph’ that can).
They can both be applied to general graphs or tree graphs.

Tree-search’s traversal pattern may revisit states many times, whereas **graph-search’s** traversal pattern maintains a ‘closed list’ to keep track of visited states so that the approach can be more efficient.
The **closed list** is a list of nodes that have been visited and expanded so the algorithm can check the list for information to be more efficient by not revisiting nodes or repeating their expansion. This saves time.
However, the memory footprint of this extra list is greater “because the graph-search algorithm keeps every node in memory” than tree-searches (Russell and Norvig, 2003, p.112).

Because of the structure of graphs, this memory requirement will be exponential but can be made more efficient with a heuristic function or is only practical for small depth, simple problems.

c) Calculate the route that would be chosen from Mehadia to Bucharest by :
[16 marks**]**

i) Uniform cost search
ii) A* search.

Explain your answer and show your working

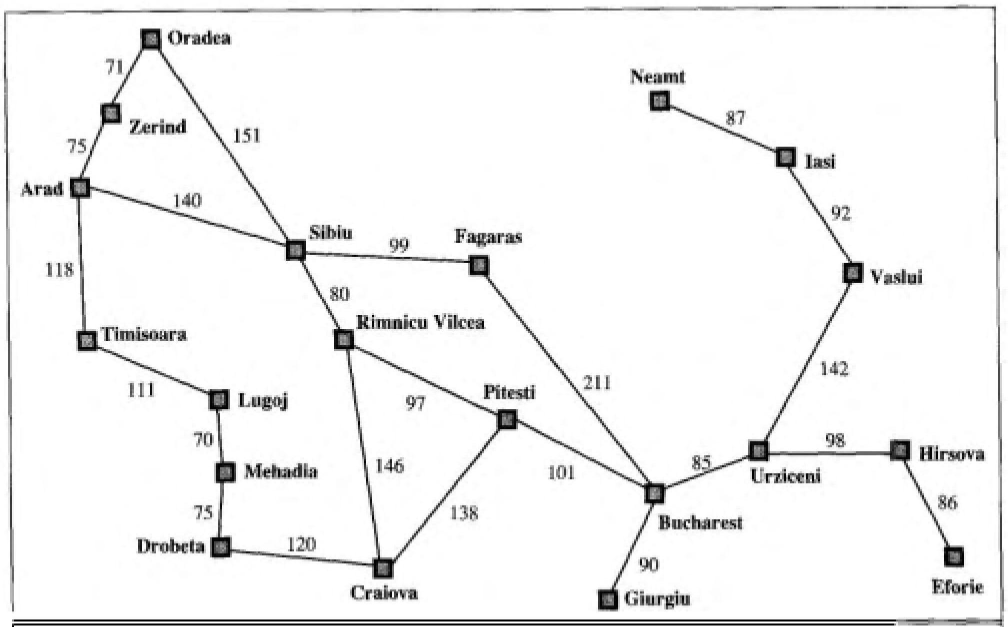


Figure 3.2 A simplified road map of part of Romania.

Figure 2. (Russell and Norvig, 2003, p.97).

i) Uniform cost search

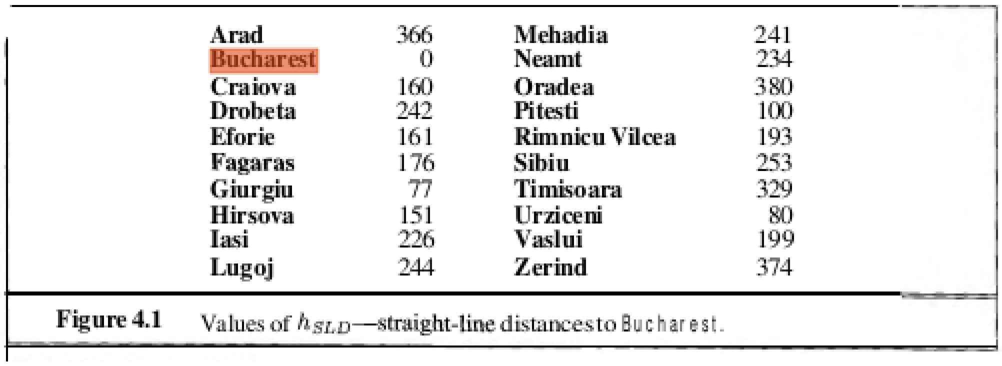
Uniform-cost search uses a priority queue to decide which member of the fringe it will search next at any time, choosing the item summing the smaller total path cost to reach to goal from the start. At each stage in the process the algorithm will follow the cheapest total path cost option currently available. This continuous speculation at each step, compares and revises the optimal lowest total path cost between nodes to the goal state. Visited nodes are not re-added to the fringe. This approach will produce an optimal path, but it's time complexity is not the most efficient.

The process of the algorithm applied to the route from Mehadia to Bucharest.

	Current Node	**Current Fringe**	**Action**	**Action Cost**	**Total cost of path from start**	**Speculation and comparison of potential path costs / notes.**	**Added to Close list (searched) (visited states)**
0		Mehadia	Go to the only state in the fringe	-	-	Start State '*S*'.	Mehadia
1	Mehadia	Lugoj, Drobeta	Mehadia → Lugoj	70	70	70 < 75 to Drobeta	Lugoj
2	Lugoj	Timișoara, Drobeta	Mehadia → Drobeta	75	75	A smaller total path cost than Lugoj → Timișoara which = 111 + 70 = 181 > 75	Drobeta
3	Drobeta	Craiova, Timișoara	Lugoj → Timișoara	111	70 + 111 = 181	181 < 195 (Mehadia → Drobeta → Craiova) 75 + 120	Timișoara
4	Timișoara	Arad, Craiova	Drobeta → Craiova	120	75 + 120 = 195	195 < 70 +111 + 118 = 299 (to Arad)	Craiova
5	Craiova	Arad, Rimnicu Vilcea, Pitesti	Timișoara → Arad	118	70 + 111 +	299 < 333 to Pitesti	Arad

					118 = 299	299 < 341 to Rimnicu Vilcea	
6	Arad	Rimnicu Vilcea, Pitești, Zerind, Sibiu	Craiova → Pitești	138	75 + 120 + 138 = 333	333 < 374 to Zerind 333 < 439 to Sibiu 333 < 341 to Rimnicu Vilcea	Pitești ok
7	Pitesti	Rimnicu Vilcea, Zerind, Sibiu, Bucharest	Craiova → Rimnicu Vilcea	146	75 + 120 + 146 = 341	341 < 434 to Bucharest 341 < 374 to Zerind 341 < 439 to Sibiu 341 < 439 to Rimnicu Vilcea (alt)	Rimnicu Vilcea
8	Rimnicu Vilcea	Zerind, Sibiu, Bucharest	Arad → Zerind	75	70 + 111 + 118 + 75 = 374	374 < 439 to Sibiu 374 < 434 to Bucharest	Zerind
9	Zerind	Sibiu, Bucharest, Oradea	Pitesti → Bucharest	101	75 + 120 + 138 + 101 = 434	434 < 439 to Sibiu 434 < 445 to Oradea **Goal State *'G'***	

Optimal Path :	Mehadia → Drobeta → Craiova → Pitești → Bucharest
Total Cost :	434



Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Figure 4.1 Values of h_{SLD}—straight-line distances to Bucharest.

Figure 3 (Russell and Norvig, 2003, p.95).

ii) A* search.

A* search optimize the Dijkstra search algorithm with the addition of a heuristic function that estimates which nodes are closer to the goal state than others. This additional information improves the selection from the priority queue for nodes to expand before others.

“A* [will] expand states in order of least estimated total path cost” (Konidaris, 2020, p.14) using the following approach :

g(n) = total path cost.
h(n) = heuristic estimate
f(n) = estimates total path cost
f(n) = h(n) + g(n), the output for the total path cost of each node ‘*n*’ in the queue + the heuristic estimate for the smaller total that will determine the next ‘*n*' that will be selected next in the path.

The algorithms process applied to the same map.

	Current Node	**Fringe**	**g(n)**	**h(n)**	**f(n)**	**Action**	**Closed List**
0		Mehadia				Select the only node in the open	Mehadia

						list/ queue.	
1	Mehadia	Lugoj Drobeta	70 75	244 242	314 317	Mehadia→ Lugoj	Lugoj
2	Lugoj	Drobeta Timișoara	75 181	242 329	317 510	Mehadia→ Drobeta	Drobeta
3	Drobeta	Timișoara Craiova	181 120	329 160	510 280	Drobeta → Craiova	Craiova
4	Craiova	Pitesti Rimnicu Vilcea Timișoara	333 341 181	100 193 329	433 534 510	Craiova → Pitești	Pitesti
5	Pitesti	Rimnicu Vilcea Timișoara Bucharest	341 181 434	193 329 0	534 510 434	Pitesti → Bucharest **Goal State '*G*'**	

Optimal Path : Mehadia → Drobeta → Craiova → Pitești → Bucharest
Total Cost : 434

The same as uniform-cost search but in fewer stages, fewer calculations with less of the graph stored in memory.
The application of the heuristic here that gives the straight line distance to the goal substantially reduces the steps in comparison to uniform cost search algorithm approach but with the same total path cost and the same path !

Question 2 - Formal logic, knowledge representation and symbolic reasoning

a) Explain **in your own words** what is meant by the following terms, in the context of formal logic: **[6 marks]**

- **i) Soundness**
- **ii) Completeness**
- **iii) Consistency**
- **iv) Validity**

i) Soundness - The results of the reasoning process applied to a knowledge base and its sentences cannot output that something is '*true*' that the knowledge base does not detail.

"[that] the resulting sentences are actually entailed by the knowledge base" (Konidaris, 2013, p.20).

Un-sound inference will output an answer that is not supported or entailed by the knowledge base. It is one of the two important properties of the reasoning process. Sound inference algorithms only output sentences that are entailed, otherwise they are "essentially [making] things up as it goes along " (Russell and Norvig, 2003, p.203).

ii) Completeness - The sentences in the knowledge base can be used by an inference algorithm to derive any truthful output that can be a result of those sentences.
That is it is able to output any sentence that is entailed by the sentences in the knowledge base. The other of the two important properties for a reasoning process, particularly when the knowledge base is infinite. This completeness means :

"Any sentence entailed by the knowledge base can be inferred" (Konidaris, 2013, p.20) and output by the inference algorithm.

iii) Consistency - Where truth values are assigned to the variables in the sentences in the knowledge base, the set of sentences is **consistent** with a model if they all have the

same ‘true’ assignment as each other for that model. This means these assignments show something together with that model.
This means the assignment of truth values to the set of sentences ‘entail’ |= the value of another sentence, another something or ’fact’.
The consistency of the set shows that sentence to be.
There can be many models consistent with a single set of sentences. Meaning the set of sentences in the knowledge base can be used to entail many things.

iv) **Validity -** If a sentence is valid in all models, then it is known as valid.

“P v ¬P is [a] valid [sentance]” (Russell and Norvig, 2003, p.210).

They can be seen to say the same thing in different ways, or different things in the same way. Because they are always ‘true’ other statements that are ‘true’ in the same models will be logically equivalent to the valid sentence, meaning they can be used to assist legitimate inference and entailment of sentences.
This allows for reasoning from sentences in a knowledge base.

b) Show which literals can be interred from the following knowledge bases, using both reasoning patterns and truth tables. Show all steps in your reasoning and explain your answers.

For each answer the logical equivalences are determined with the set of logical equivalences and the reasoning patterns on pages 20-22 as well as truth tables to illustrate these equivalences and that they hold.

Where the whole knowledge base is displayed on a truth table it is to show the additional literals that can be inferred by them collectively with ^ between the literals in the base.

Truth table columns in **bold** show literals that can be inferred.

i)

Knowledge Base

P ^ Q

Q → (R → S)
P → ¬S

P ^ Q =

P	Q	**P^Q**	**¬(¬P ^ ¬Q)**	¬(P^Q)	**(¬P v ¬Q)**
1	1	1	1	0	1
1	0	0	0	1	0
0	1	0	0	1	0
0	0	0	0	1	0

P ^ Q ==	Q ^ P	Commutativity of ^
	¬(¬P ^ ¬Q)	Double negative addition
	¬P V ¬Q	De Morgan

Q → (R → S) =

Q	R	S	(R → S)	**Q → (R → S)**	**¬Q v ¬R v S**	¬Q	**¬Q v (R → S)**	¬R v S
1	1	1	1	1	1	0	1	1
1	1	0	0	0	0	0	0	0
1	0	1	1	1	1	0	1	1
1	0	0	1	1	1	0	1	1
0	1	1	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0
0	0	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1

Q → (R → S) ==	¬(Q ^ R ^ ¬ S)	De Morgan
	(¬Q v ¬R v S)	De Morgan
	Q → (¬S → ¬R)	Contraposition
	Q → (¬R v S)	Implication elimination
	¬Q v (R → S)	Implication elimination

P → ¬S ==

P	S	¬S	**P→(¬S)**	P^S	**¬(P^S)**	¬P	**S → ¬P**	**¬P v ¬ S**
1	1	0	0	1	0	0	0	0
1	0	1	1	0	1	0	1	1
0	1	0	1	0	1	1	1	1
0	0	1	1	0	1	1	1	1

P → ¬S ==	¬(P^S)	
	¬(¬S) → ¬P	
	S → ¬P	Contraposition
	¬P v ¬S	Implication elimination
	(¬P v ¬ S)	De Morgan

P	**Q**	**R**	**S**	**P^Q**	R → S	**Q → (R → S)**	(P^Q) ^ (Q → (R → S)	**P→ ¬S**	**((P^Q) ^ (Q → (R → S)) ^ (P → ¬S)**
1	1	1	1	1	1	1	1	0	0
1	1	1	0	1	0	0	0	1	0
1	1	0	1	1	1	1	1	0	0
1	1	0	0	1	1	1	1	1	**1**
1	0	1	1	0	1	1	0	0	0
1	0	1	0	0	0	1	0	1	0
1	0	0	1	0	1	1	0	0	0
1	0	0	0	0	1	1	0	1	0
0	1	1	1	0	1	1	0	1	0
0	1	1	0	0	0	0	0	1	0
0	1	0	1	0	1	1	0	1	0
0	1	0	0	0	1	1	0	1	0
0	0	1	1	0	1	1	0	1	0
0	0	1	0	0	0	1	0	1	0
0	0	0	1	0	1	1	0	1	0
0	0	0	0	0	1	1	0	1	0

Showing literals infrared : **P ^ Q ^ ¬R ^ ¬S**

P, Q, ¬R, ¬S

ii) Knowledge Base

¬(P ^ Q)
P v Q
Q → R
¬R

¬(P ^ Q) ==

P	Q	(P ^ Q)	**¬(P^Q)**	**¬P v ¬Q**
1	1	1	0	0
1	0	0	1	1
0	1	0	1	1
0	0	0	1	1

¬(P ^ Q) == (¬P v ¬Q) De Morgan

P v Q ==

P	Q	**P v Q**	¬P ^ ¬Q	**¬(¬P ^ ¬Q)**
1	1	1	0	1
1	0	1	0	1
0	1	1	0	1
0	0	0	1	0

P v Q == ¬(¬P ^ Q) Double negative addition

Q → R ==

Q	R	**Q → R**	**¬R → ¬Q**	**¬Q v R**	¬Q	¬R	**¬(Q ^ ¬R)**	Q ^ ¬R
1	1	1	1	1	0	0	1	0
1	0	0	0	0	0	1	0	1
0	1	1	1	1	1	0	1	0
0	0	1	1	1	1	1	1	0

Q → R ==	¬R → ¬Q	Contraposition
	¬Q v R	Implication elimination
	¬(Q ^ ¬R)	De Morgan

¬R ==

¬R	R
1	0
0	1

¬R v R = ‘true’ = valid sentence.

¬Q v R, ¬R Unit resolution.
---------------------,
¬Q **New inferred literal.**

P v Q, ¬Q Unit resolution with above sentence .
---------------------,
P **New infrared literal.**

P ^ ¬Q ^ ¬R Additional literals inferred.

iii) Knowledge Base

(A v B) ^ C
¬ (B ^ C)

(A v B) ^ C ==

A	B	C	A v B	**(A V B) ^ C**	C ^ B	C ^ A	**((C ^ B) v (C ^ A))**
1	1	1	1	1	1	1	1
1	1	0	1	0	0	0	0
1	0	1	1	1	0	1	1
1	0	0	1	0	0	0	0
0	1	1	1	1	1	0	1
0	1	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0

(A v B) ^ C == ((C ^ B) v (C ^ A)) Distribution of ^ over v

¬(B ^ C) ==

B	C	¬B	¬C	B ^ C	**¬(B ^ C)**	**(¬B v ¬C)**
1	1	0	0	1	0	0
1	0	0	1	0	1	1
0	1	1	0	0	1	1
0	0	1	1	0	1	1

¬(B ^ C) == (¬B v ¬C) De Morgan

A	B	C	A v B	(AvB)^C	(B^C)	¬(B^C)	**((A v B) ^ C) ^ ¬(B^C)**

1	1	1	1	1	1	0	0
1	1	0	1	0	0	1	0
1	0	1	1	1	0	1	**1**
1	0	0	1	0	0	1	0
0	1	1	1	1	1	0	0
0	1	0	1	0	0	1	0
0	0	1	0	0	0	1	0
0	0	0	0	0	0	1	0

New literals inferred : **A, ¬B, C**
A ^ ¬B ^ C

c) Represent the following statements as formulas in Predicate Calculus.
Give two versions of each formula with ∃ and ∀.

i) ∀ x, Musician(x) → PlaysAMusicalInstrument(x) v Sings(x),
∃ x, PlaysAMusicalInsturment(x) v Sings(x) → Musician(x),

ii) ∀ x, Piano(x) v Trumpet(x) v Guitar(x) v Saxophone(x) v Trombone(x) →
MusicalInstruments(x),
∃ x, MusicalInstrument(x) → Piano(x) v Trumpet(x) v Guitar(x) v Saxophone(x)
v
Trombone(x),

iii) ∀ x, Trumpets(x) v Saxophones(x) v Trombones(x) → Horn(x),
∃ x, Horns(x) → Trumpets(x) v Saxophones(x) v Trombones(x),

iv) ∀ x, MusicalInstruments(x) → Horn(x) v ¬ Horn(x)
∃ x, MusicalInstruments(x) → ¬Horn(x)

v) ∀ x, Singers(x) → PlaysTrumpet(x) v ¬PlaysTrumpet(x),
∃ x, Singer(x) → PlaysTrumpet(x),

d) Encode the following facts about jazz and blues performers as Prolog clauses.

SWI - Prolog was used to encode and evaluate the answers for this question as the following code and outputs.

1) Billie Holiday, Bessie Smith, Chet Baker and Louis Armstrong were all singers.

```
singer("Billie Holiday").
singer("Bessie Smith").
singer("Chet Baker").
singer("Louis Armstrong").
```

2) Chet Baker, Louis Armstrong and Dizzy Gillespie played trumpet.

```
trumpeter("Chet Baker").
trumpeter("Louis Armstrong").
trumpeter("Dizzy Gillespie").
```

3) Count Basie, Thelonios Monk, Duke Ellington and Mary Lou Williams played the piano.

```
pianist("Count Basie").
pianist("Thelonios Monk").
pianist("Duke Ellington").
pianist("Mary Lou Williams").
```

4) Count Basie, Duke Ellington and Chick Webb were big-band leaders.

```
bigbandleader("Count Basie").
bigbandleader("Duke Ellington").
bigbandleader("Chick Webb").
```

5) Charlie Parker, John Coltrane and Lester Young played the saxophone.

```
saxophonist("Charlie Parker").
saxophonist("John Coltrane").
saxophonist("Lester Young").
```

6) Chick Webb was a drummer.

```
drummer("Chick Webb").
```

7) Django Reinhardt, Wes Montgomery, Robert Johnson and BB King played guitar.

```
guitarist("Django Reinhardt").
guitarist("Wes Montgomery,").
guitarist("Robert Johnson").
guitarist("BB King").
```

8) Robert Johnson, BB King, Bessie Smith and Billie Holiday were blues artistes.

```
bluesplayer("Robert Johnson").
bluesplayer("BB King").
bluesplayer("Bessie Smith").
bluesplayer("Billie Holiday").
```

9) Charlie Parker, Dizzy Gillespie, Thelonious Monk and Mary Lou Williams played in the ‘bebop’ style.

```
bebopplayer("Charlie Parker").
bebopplayer("Dizzy Gillespie").
bebopplayer("Thelonious Monk").
bebopplayer("Mary Lou Williams").
```

Additional common sense inferences and/ or other reasonable general statements :

```
musician(X) :- singer(X) ; trumpeter(X) ; pianist(X) ; bigbandleader(X) ;
               saxophonist(X) ; drummer(X) ; guitarist(X) ; bluesplayer(X) ;
               bebopplayer(X).

/* alternate unused attempt to remove duplicates.
musician(X) :- (singer(X)) ; ((not(singer(X))) , (trumpeter(X) ; pianist(X) ;
bigbandleader(X) ; saxophonist(X) ; drummer(X) ; guitarist(X))).
*/
```

i)

```
?- musician(X).
X = "Billie Holiday" ;
X = "Bessie Smith" ;
X = "Chet Baker" ;
X = "Louis Armstrong" ;
X = "Chet Baker" ;
X = "Louis Armstrong" ;
X = "Dizzy Gillespie" ;
X = "Count Basie" ;
X = "Thelonios Monk" ;
X = "Duke Ellington" ;
X = "Mary Lou Williams" ;
X = "Count Basie" ;
X = "Duke Ellington" ;
X = "Chick Webb" ;
X = "Charlie Parker" ;
X = "John Coltrane" ;
X = "Lester Young" ;
X = "Chick Webb" ;
X = "Django Reinhardt" ;
X = "Wes Montgomery," ;
X = "Robert Johnson" ;
X = "BB King" ;
X = "Robert Johnson" ;
```

```
X = "BB King" ;
X = "Bessie Smith" ;
X = "Billie Holiday" ;
X = "Charlie Parker" ;
X = "Dizzy Gillespie" ;
X = "Thelonious Monk" ;
X = "Mary Lou Williams".
```

ii)

```
?- bebopplayer(X) , trumpeter(X).
X = "Dizzy Gillespie"
```

iii)

```
?- trumpeter(“Count Basie”).
false
```

iv)

```
?- saxophonist(X) , (not(bebopplayer(X))).
X = "John Coltrane" ;
X = "Lester Young".
```

v)

```
?- bigbandleader(X) , pianist(X).
X = "Count Basie" ;
X = "Duke Ellington" ;
```

vi)

```
?- singer(X), musician(X).
X = "Billie Holiday" ;
X = "Billie Holiday" ;
X = "Bessie Smith" ;
X = "Bessie Smith" ;
X = "Chet Baker" ;
X = "Chet Baker" ;
X = "Louis Armstrong" ;
X = "Louis Armstrong" ;
```

Question 3 - Probabilistic reasoning

a) Explain the following terms in your own words.

i) Probability distribution, like a 'bell curve' plotted on a graph, a mathematical function used in probability theory to describe the probabilities of the different values being assigned to variables within an experiment or sample space.
It contains all the probabilities of the different outcomes (or values, that the random variables can be assigned) happening.
The set of all possible outcomes and their probability to be assigned at each variable in an experiment.

ii) Boolean, discrete and continuous random variables are the three types of random variables that can be assigned values. They each have a domain (a set of values it can be assigned) and a probability between 0 and 1 for its assignment to each value in the domain. The boolean domain is [true, false] for example and a continuous random variable domain can be a segment of the real number line.

iii) Prior probability is the probability assignment to a variable before new data or an experiment or other information is taken into account. It is a way of stating the current understood or perceived probability something has, before something affecting that probability takes place.

iv) Posterior probability is the 'after' probability when an outcome of an experiment or other information is taken into consideration when evaluating the probability value assigned to a variable.
This is the 'new' probability of the 'prior' probability now the new data is considered.
It is used in belief revision.

v) Marginalisation can be used to discover the probability of a single variable by 'summing over' all possible values of another in a joint probability so that it can be considered without reference as an individual variable, because in the joint probability the variables 'depend' on each other.

$$P(A) = P(A,B) + P(A,\neg B)$$ (Konidaris, 2013, p.28)

vi) **The Chain Rule for probability** can be used to compute the joint probabilities of more than one event. It can be used to calculator for example :

P(A and B) by P(A) P(B|A) (probability of A multiplied by the probability of B given that A is true).

And the joint probability of more than one event such as P(A and B and C).

In the same way that the conditioning formula gives the joint probability without a table the chain rule gives the joint probability without a table for an ‘*n*’ amount of variables.

b) This question concerns the probabilities of outcomes of a roll of two fair, standard six-faced dice, one red and one blue.

i) **1)**

P	**A**	**¬A**
B	5/36	10/36
¬B	1/36	20/36

2)

P	**A**	**¬A**
C	1/36	5/36
¬C	5/36	25/36

ii)

1) **P(A ^ B)** = 5/36

2) **P(A ^ C) =** 1/36

3) **P(A|B) =** 1/3 = 11/36

P(A|B) = P(A,B) / P(A, B) + P(¬A, B)
5/36 / 5/36 + 10/36 = 5/36 / 15/36
= 1/3 = 11/36

4) **P(B|A) =** 5/6 = 30/36

P(B|A) = P(B, A) / P(B, A) + P(¬B, A)
5/36 / 5/36 + 1/36 = 5/36 / 6/36
= 5/6 = 30/36

5) **P(A|C) =** 1/6 = 6/36

P(A|C) = P(A,C) / P(A, C) + P(¬A, C)
1/36 / 1/36 + 5/36 = 1/36 / 6/36
= 1/6 = 6/36

6) **P(C|A)** = 1/6 = 6/36

P(C|A) = P(C, A) / P(C ,A) + P(¬C, A)
1/36 / 1/36 + 5/36 = 1/36 / 6/36
= 1/6 = 6/36

iii) **Independent variables** in probability are where the probability of a variable is unaffected by the probability of other variables, as opposed to joint probabilities that are affected by the probability of the others occuring.
"P(A, B) = P(A, ¬B)" (Konidaris, 2013, p.28). A is unaffected by B.

Iv, v, vi)
P(A|B) = P(A)

P(B|A) = P(B)
P(A^B) = P(A)P(B)

P(Y|X) = P(X|Y)P(Y) taken from (essential reading, p.480) (Equation 13.11)

P(A|B) = P(B|A)P(A)
P(B|A) = P(A|B)P(B)
P(A^B) = P(A)P(B) = P(A|B) x P(B|A) = P(A|B)P(A) x P(A|B)P(B)

If the variables are independent A will be unaffected by the probability of B and P(A|B) = P(A) = P(B|A)P(A), as B would have no effect on A being true or untrue as, shown in iii.

vii)

Statements for Dice A and B are **not** independent as their probability of occurring are affected by the probability of the other, as shown by the joint probability table and answers in ii.

P(A)P(B) != P(A^B), (15/36 x 1/36 = 5/432) != 5/36 = P(A^B)
and P(A|B) != P(B|A)P(A) , P(A|B) != P(A)
Calculated with P(B) = 15/36 = P(A,B) + P(¬A,B)
P(A) = 1/36 = P(A,B) + P(A,¬B)

viii)

Statements for Dice A and C each have a probability of 1/6 , that is the same as P(A|C) and P(C|A), which are both 1/6 as well, indicating with the above definitions that they **are** independent. P(A^C) = P(A)P(C)
P(C) = P(A,C) + P(¬A,C) = 1/6 = 6/36
P(A) = P(A,C) + P(A,¬C) = 1/6 = 6/36
P(A)P(C) = P(A^C) = 1/36 = 6/36 x 6/36
Which is shown in the joint probability table.

c)

P	
ABC	540/46656
AB¬C	2700/46656
A¬BC	756/46656
A¬B¬C	3780/46656
¬ABC	2700/46656
¬AB¬C	13500/46656
¬A¬BC	3780/46656
¬A¬B¬C	18900/46656

= 1.

1) **P(A ^ B) = 5/36 (the same)**

   P(A) = P(A,B) + P(A,¬B) (Konidaris, 2013, p.28).
   P(A^B) = P(A^B^C) + P(A^B^¬C) = 540/46656 + 2700/46656 = 3240/46656
   = 5/36

2) **P(A ^ C) = 1/36 (the same)**

   P(A^C) = P(A^B^C) + P(A^¬B^C) = 540/46656 + 756/46656 = 1296/46656
   = 1/36

3) **P(A|B) = 2/7 (not the same)**

   P(A|B) = P(A,B) / P(A, B) + P(¬A, B)
   p(¬A,B) = P(¬A,B,C) + P(¬A,B,¬C) = 2700/46656 + 13500/46656 = 16200/46656 = 25/ 72.
   P(A|B) = 5/36 / 5/36 + 25/72 = 2/7

4) **P(B|A) = 10/17 (not the same)**

P(B|A) = P(B,A) / P(B,A) + P(¬B,A)
P(¬B,A) = P(¬B,A,C) + P(¬B,A,¬C) = 756/46656 + 3780/46656 = 4536/46656 = 7/72.
P(B|A) = 5/36 / 5/36 + 7/72 = 10/17

5) **P(A|C) = 1/6** **(the same)**

P(A|C) = P(A,C) / P(A,C) + P(¬A,C)
P(¬A,C) = P(¬A,B,C) + P(¬A,¬B,C) = 2700/46656 + 3780/46656 = 6480/46656 = 5/36
P(A|C) = 1/36 / 1/36 + 5/36 = 1/36 / 6/36 = 1/6

6) **P(C|A) = 1/6** **(the same)**

P(C|A) = P(C,A) / P(C,A) + P(¬C,A)
P(¬C,A) = P(¬C,A,B) + P(¬C,A,¬B)
P(C|A) = 1/36 / 1/36 + 5/36 = 1/36 / 6/36 = 1/6

Because some answers are the same and some are not as question 3, indicates that the joint probability of all 3 variables affects the outcome of certain events but not others as only some of the variables are independent, others are dependent.
Variables A and C are independent of B so they have the same result as the prior table.
Variable B is dependent on A and C so there are different answers.

References

- S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach. (London: Pearson, 2003).
- G.D. Konidaris, Artificial Intelligence (2013). University of London - Study Guide.