

Contents

Contents	1
Table of Figures	1
1 Question 01 (CSP)	2
2 Question 02 (Ontology)	4
2.1 Overview.....	4
2.2 Part (a).....	4
2.3 Part (b)	6
2.4 Part (c).....	7
2.5 Part (d).....	18
3 Question 03 (Maze).....	24
3.1 Overview.....	24
3.2 Task 01 – Creating a random maze	24
3.3 Task 02 – DFS Search.....	26
3.4 Task 03 – Heuristic value function.....	28
3.5 Task 04 – A* Search.....	28
3.6 Task 05	31
4 Question 04 (Fuzzy-Logic)	33

Table of Figures

Figure 1: Constraints of CSP	2
Figure 2: Table before applying Solver.....	2
Figure 3:MIN optimization output.....	3
Figure 4: MAX optimization output	3
Figure 5 : Mechanical Engineering Taxonomy	6

1 Question 01 (CSP)

This question instructs to use MS-Excel solver to find the optimal value for each defined variable considering the constraint satisfaction. The problem talks about assigning shifts to 4 employees after considering some constraints, The constraints are.

Constraint 1	Minimum shifts assigned for an employee should be 2
Constraint 2	Maximum shifts assigned for an employee should be 10
Constraint 3	Atleast one staff should be assinged for each shift
Constraint 4	Two employees prefer Morning shits and other two prefer Evening shifts

Figure 1: Constraints of CSP

The final target of the question is to assign shifts to four employees for a seven-day week, by making sure that the constraints are not violated.

Implementation :

Firstly, assume that employees 1 and 2 prefer to work in the morning and the other two employees prefer to work in the evening. Then created a table as follow and entered 0 to all the variable columns,

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1																		
2			Day 1		Day 2		Day 3		Day 4		Day 5		Day 6		Day 7		Total shifts per Employee	
3	Employee	Preferred shift	Morning	Evening	Morning	Evening	Morning	Evening	Morning	Evening	Morning	Evening	Morning	Evening	Morning	Evening		
4	Employee 1	Morning	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
5	Employee 2	Morning	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	Employee 3	Evening	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	Employee 4	Evening	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
9																		

Figure 2: Table before applying Solver

Then to calculate the total employees assigned to each shift, used the 'sum' function in excel. As an example, to calculate the total employees assigned for morning shift of day 1, add the function.

=SUM(C4:C7)

Likewise, the same was done to calculate the total number of employees assigned for other shifts as well. And the same SUM function was used again to calculate the total shifts assigned to an employee.

Afterwards used the solver to define the constraints for the system. Firstly, defined the variable cells [C4:P7] and added a constraint specifying that the variable cell can only be binary values because the variable values can only take either 1 or 0, other values don't imply a meaning for the given context.

Then defined that the cell value C8 should be greater than or equal to one as for the constraint no3. Then applied the same to the cells from D8 to P8.

Defined a constraint in the solver to ensure that the total number of shifts assigned to each employee is between 2 and 10, inclusive of both 2 and 10.).

Afterwards as employees 1 and 2 prefer morning according to the assumptions made, added a constraint saying that for all evening shifts for employee 1 and 2 must '0'. The vice versa was done for the two employees who prefer evening shifts.

Finally added another constraint stating that the two employees who prefer the same shift must have either equal number of shifts or on guys has to have 1 shift higher than the other.

So, if the objective was set to **Min** the output is as follow:

Employee	Preferred shift	Day 1		Day 2		Day 3		Day 4		Day 5		Day 6		Day 7		Total shifts per Employee
		Morning	Evening	Morning	Evening	Morning	Evening	Morning	Evening	Morning	Evening	Morning	Evening	Morning	Evening	
Employee 1	Morning	1	0	1	0	0	0	0	0	0	0	0	0	1	0	3
Employee 2	Morning	0	0	0	0	1	0	1	0	1	0	1	0	0	0	4
Employee 3	Evening	0	0	0	0	0	0	0	1	0	0	0	1	0	1	3
Employee 4	Evening	0	1	0	1	0	1	0	0	0	1	0	0	0	0	4
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	

Figure 3: MIN optimization output

if the objective was set to **Max** the output is as follows:

Employee	Preferred shift	Day 1		Day 2		Day 3		Day 4		Day 5		Day 6		Day 7		Total shifts per Employee
		Morning	Evening	Morning	Evening	Morning	Evening	Morning	Evening	Morning	Evening	Morning	Evening	Morning	Evening	
Employee 1	Morning	1	0	1	0	1	0	1	0	1	0	1	0	1	0	7
Employee 2	Morning	1	0	1	0	1	0	1	0	1	0	1	0	1	0	7
Employee 3	Evening	0	1	0	1	0	1	0	1	0	1	0	1	0	1	7
Employee 4	Evening	0	1	0	1	0	1	0	1	0	1	0	1	0	1	7
		2	2	2	2	2	2	2	2	2	2	2	2	2	2	

Figure 4: MAX optimization output

Overall optimization is done to improve employee satisfaction, considered giving the minimum no.of shifts possible considering all the employees.

2 Question 02 (Ontology)

2.1 Overview

2.2 Part (a)

Scope of the Ontology

The scope of the ontology is to cover the main concepts and relations in mechanical engineering, focusing on the five aspects which are namely: engineering design, knowledge discovery, quality control, and education and training. The ontology will include terms and definitions related to mechanical engineering domains. The ontology will enable reasoning and inference over the knowledge base, as well as querying and searching for relevant information.

1. Engineering Design :

Creates efficient machines by turning ideas into practical plans. It recommends materials, identifies flaws, and generates alternative solutions.

Competency Questions:

- How can ontology suggest optimal materials and manufacturing processes for a given design concept?
- How can the ontology identify potential design flaws and suggest corrective actions?
- How can the ontology generate alternative design solutions based on specific constraints and objectives.

2. Knowledge Discovery :

Mines valuable insights from mechanical data like sensor readings and maintenance records. It predicts failures, develops new materials, and optimizes existing practices.

Competency Questions:

- How can the ontology link sensor data with design files to predict potential machine failures?
- How can users leverage the ontology to search for specific mechanical engineering knowledge using natural language?
- How can the ontology identify patterns in maintenance records to suggest preventative maintenance strategies?

3. Quality Control :

Ensures reliable machines through real-time monitoring, automated inspections, and risk assessments. It analyzes defects to identify root causes and improve quality continuously.

Competency Questions:

- How can the ontology recommend real-time adjustments to manufacturing processes based on sensor data?
- How can the ontology select the most appropriate inspection procedure for a specific component considering cost and accuracy?
- How can the ontology analyze defect data to identify root causes and suggest corrective actions?

4. Education and Training :

Makes learning mechanical engineering engaging and effective. It personalizes learning materials, provides hands-on simulations, and analyzes student data for personalized feedback.

Competency Questions:

- How can the ontology generate individualized learning pathways for students based on their skill levels and interests?
- How can the ontology utilize interactive simulations to provide students with hands-on learning experiences without physical equipment?
- How can the ontology analyze student performance data to provide personalized feedback and recommendations for improvement?

5. Mechatronics and Control Systems :

Integrates mechanical, electrical, and computer systems. It recommends sensors and actuators, generates control algorithms, and optimizes robot movements for efficiency and safety.

Competency Questions:

- How can the ontology recommend the most effective sensor and actuator combination for a specific control task?
- How can the ontology automatically generate control algorithms based on desired system behavior and real-time sensor data?
- How can the ontology optimize robot movements to ensure efficiency, accuracy, and safety in dynamic environments?

References

1. An ontology-based knowledge framework for engineering material selection: <https://www.sciencedirect.com/science/article/pii/S147403461500097X>
2. Ontology and its applications in mechanical engineering: https://www.researchgate.net/publication/288688809_Ontology_and_its_applications_in_mechanical_engineering

3. A function-based component ontology for systems design: <https://www.designsociety.org/download-publication/25620/A+Function-Based+Component+Ontology+for+Systems+Design>
4. Ontology-Based Knowledge Representation for Mechanical Products: <https://www.sciencedirect.com/science/article/am/pii/S016636151830808X>
5. Ontology engineering - Wikipedia: https://en.wikipedia.org/wiki/Ontology_engineering
6. Mechanical Engineering Ontology, by M. G. Negoita, A. G. Florea, and C. Badica, in Proceedings of the 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, 2008.
7. Ontology-Based Knowledge Management for Mechanical Engineering Design, by S. H. Kim, J. H. Lee, and S. H. Han, in International Journal of Advanced Manufacturing Technology, 2010.

2.3 Part (b)

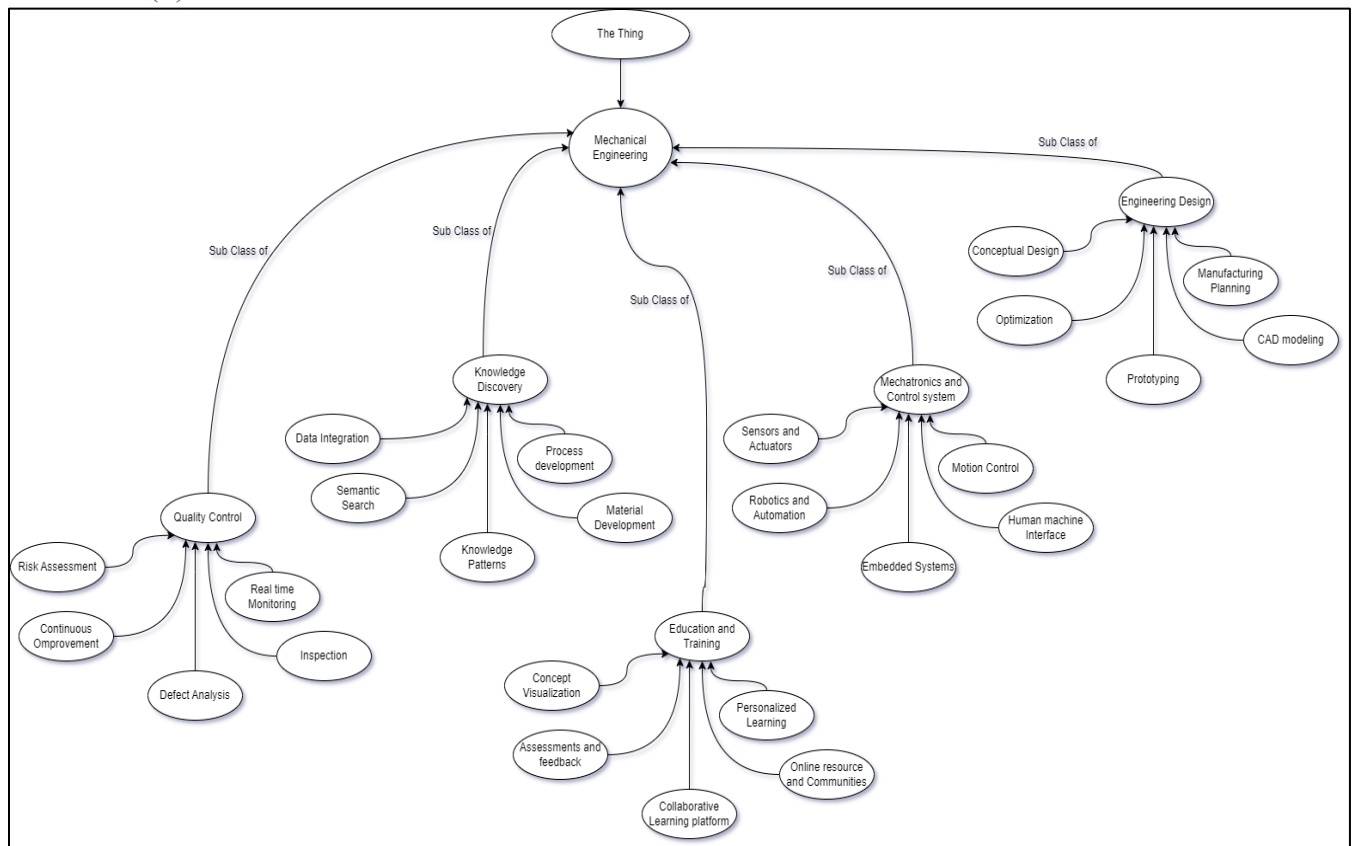


Figure 5 : Mechanical Engineering Taxonomy

The above image is an abstract taxonomy which describes how mechanical engineering is broken down according to the above-described subcategories and competency questions. Firstly, this contains the entity 'The Thing' which denotes that this is linked to another taxonomy, as it won't be a meaningful way of representation if a taxonomy/ concept graph is created for each domain or sub-domains. So, most of the time these concept graphs are linked together in order to get a meaningful understanding.

Afterwards the taxonomy shows that mechanical engineering has been further decomposed into five subcategories, namely:

Engineering design, Knowledge discovery, Quality control, Education and training, Mechatronics, and control systems. Each of these main subcategories are again divided into five more objects/ categories according to the above-mentioned competency questions.

- **Knowledge discovery** : This consists of data integration, semantic search, knowledge patterns, material, and process development. In the context of knowledge discovery for mechanical engineering data integration and semantic search gives a deeper understanding of system behaviors and potential failures. We push Mechanical Engineering into the future. Powerful materials like nanobots and biomaterials emerge from predictive models, while process development boosts production with machine learning and 3D printing. Imagine smarter, stronger machines shaping tomorrow.
- **Quality Control** :
This consists of Quality Control ensuring your machines stay top-notch. Sensors continuously scan for anomalies, automated eyes inspect everything from cracks to dimensions, while risk assessments predict potential problems. If flaws do slip through, advanced techniques like spectroscopy unveil their secrets, leading to continuous improvement through data-driven tweaks. This vigilant cycle keeps your machines running smoothly and safely, pushing the envelope of reliability.
- **Engineering Design**
Engineering Design is the foundational blueprint of innovation. It encompasses the meticulous process of conceptualizing, planning, and crafting mechanical solutions that are both functional and groundbreaking. Every design phase focuses on optimizing performance, durability, and efficiency, ensuring that the product not only meets but often exceeds industry standards.
- **Education and Training**
Education and Training are the pillars fostering expertise and proficiency in the mechanical engineering domain. Through rigorous academic curricula and hands-on training modules, aspiring engineers are equipped with the theoretical knowledge and practical skills essential for tackling real-world challenges. Continuous learning and upskilling ensure that professionals remain at the forefront of technological advancements, driving innovation and excellence in the field
- **Mechatronics and Control systems**
- echatronics and Control Systems represent the synergy of mechanical engineering, electronics, and computer science. This interdisciplinary field focuses on designing intelligent systems that seamlessly integrate hardware and software components. By harnessing advanced control algorithms and automation techniques, mechatronic systems enhance precision, efficiency, and adaptability, revolutionizing industries and shaping the future of automation.

2.4 Part (c)

<rdf:RDF

```

xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:relationship="http://example.org/MechanicalEngineering#"

<!-- OWL Class definition - The Thing -->
<owl:Class rdf:about="http://www.linkeddatatools.com/Thing">
  <rdfs:label>Thing</rdfs:label>
  <rdfs:comment>Defines The universal root category</rdfs:comment>
</owl:Class>

<!-- OWL Header -->
<owl:Ontology rdf:about="http://www.linkeddatatools.com/Thing">
  <dc:title>Mechanical Engineering</dc:title>
  <dc:description>An abstract ontology designed to give a brief
understanding into Mechanical Engineering and it's fundamentals.</dc:description>
</owl:Ontology>

<!-- OWL Subclass definition of Thing - Mechanical Engineering -->
<owl:Class
rdf:about="http://www.linkeddatatools.com/Thing#MechanicalEngineering">
  <rdfs:label>Mechanical Engineering</rdfs:label>
  <rdfs:comment>Represents most of the mechanical engineering fundamentals
in the ontology.</rdfs:comment>
</owl:Class>

<!-- Defining the main five subcategories under Mechanical engineering -->
<!-- Engineering Design subcategory definition -->
<owl:Class
rdf:about="http://www.linkeddatatools.com/Thing/MechanicalEngineering#Engineering
Design">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/Thing#MechanicalEngineering"/>
  <rdfs:label>Engineering Design</rdfs:label>
  <rdfs:comment>Creates efficient machines by turning ideas into practical
plans. It recommends materials, identifies flaws, and generates alternative
solutions.</rdfs:comment>
</owl:Class>

<!-- Knowledge Discovery subcategory definition -->

```



```

    <owl:Class
rdf:about="http://www.linkeddatatools.com/Thing/MechanicalEngineering#KnowledgeDiscovery">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/Thing#MechanicalEngineering"/>
    <rdfs:label>Knowledge Discovery</rdfs:label>
    <rdfs:comment>Mines valuable insights from mechanical data like sensor
readings and maintenance records. It predicts failures, develops new materials,
and optimizes existing practices.</rdfs:comment>
    </owl:Class>

    <!-- Quality Control subcategory definition -->
    <owl:Class
rdf:about="http://www.linkeddatatools.com/Thing/MechanicalEngineering#QualityControl">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/Thing#MechanicalEngineering"/>
    <rdfs:label>Quality Control</rdfs:label>
    <rdfs:comment>Ensures reliable machines through real-time monitoring,
automated inspections, and risk assessments. It analyzes defects to identify root
causes and improve quality continuously.</rdfs:comment>
    </owl:Class>

    <!-- Education and Training subcategory definition -->
    <owl:Class
rdf:about="http://www.linkeddatatools.com/Thing/MechanicalEngineering#EducationAndTraining">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/Thing#MechanicalEngineering"/>
    <rdfs:label>Education and Training</rdfs:label>
    <rdfs:comment>Makes learning mechanical engineering engaging and
effective. It personalizes learning materials, provides hands-on simulations, and
analyzes student data for personalized feedback.</rdfs:comment>
    </owl:Class>

    <!-- Mechatronics and Control Systems subcategory definition -->
    <owl:Class
rdf:about="http://www.linkeddatatools.com/Thing/MechanicalEngineering#MechatronicsAndControlSystems">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>

```

```

    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/Thing#MechanicalEngineering"/>
    <rdfs:label>Mechatronics and Control Systems</rdfs:label>
    <rdfs:comment>Integrates mechanical, electrical, and computer systems. It
recommends sensors and actuators, generates control algorithms, and optimizes
robot movements for efficiency and safety.</rdfs:comment>
  </owl:Class>

  <!-- Defining the subcategories under Engineering Design -->
  <!-- Prototyping Subclass-->
  <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/EngineeringDesign
#Prototyping">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#EngineeringDes
ign"/>
    <rdfs:label> Prototyping</rdfs:label>
    <rdfs:comment>tangible form of an engineering design.Allows engineers to
test theories, identify flaws, and refine designs before committing to costly
production.</rdfs:comment>
  </owl:Class>

  <!-- Optimization Subclass-->
  <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/EngineeringDesign
#Optimization">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#EngineeringDes
ign"/>
    <rdfs:label>Optimization</rdfs:label>
    <rdfs:comment>fine-tuning of mechanical systems to achieve the best
possible performance, often involving balancing multiple factors such as cost,
efficiency, durability, and functionality.</rdfs:comment>
  </owl:Class>

  <!-- Manufacturing and Planning Subclass-->
  <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/EngineeringDesign
#ManufacturingAndPlanning">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>

```

```

    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#EngineeringDes
ign"/>
    <rdfs:label>Manufacturing and Planning</rdfs:label>
    <rdfs:comment>Involves optimizing production processes, resource
allocation, and scheduling to efficiently turn designs into physical
products</rdfs:comment>
  </owl:Class>

  <!-- CAD Modeling Subclass-->
  <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/EngineeringDesign
#CADModeling">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#EngineeringDes
ign"/>
    <rdfs:label>CAD Modeling</rdfs:label>
    <rdfs:comment>A software which allows engineers to create and manipulate
3D models of their designs, enabling visualization, analysis, and
collaboration</rdfs:comment>
  </owl:Class>

  <!-- Conceptual Design Subclass-->
  <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/EngineeringDesign
#ConceptualDesign">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#EngineeringDes
ign"/>
    <rdfs:label>Conceptual Design</rdfs:label>
    <rdfs:comment>The phase where ideas are born and explored, laying the
foundation for detailed engineering solutions</rdfs:comment>
  </owl:Class>

  <!-- Defining the subcategories under Mechatronics and Control Systems -->
  <!-- Sensors and Actuators Subclass-->
  <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/MechatronicsAndCo
ntrolSystem#SensorsAndActuators">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>

```

```

    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#MechatronicsAndControlSystem"/>
    <rdfs:label> Sensors and Actuators</rdfs:label>
    <rdfs:comment>The eyes and ears of machines. Sensors gather data about a system's state, while actuators translate commands into physical actions, enabling intelligent control and interaction with the environment</rdfs:comment>
  </owl:Class>

  <!-- Motion Control Subclass-->
  <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/MechatronicsAndControlSystem#MotionControl">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#MechatronicsAndControlSystem"/>
    <rdfs:label>Motion Control</rdfs:label>
    <rdfs:comment>These systems precisely guide robots, vehicles, and other mechanisms, ensuring accurate and efficient movement</rdfs:comment>
  </owl:Class>

  <!-- Human Machine Interface(HMI) and Planning Subclass-->
  <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/MechatronicsAndControlSystem#HumanMachineInterface">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#MechatronicsAndControlSystem"/>
    <rdfs:label>Human Machine Interface(HMI)</rdfs:label>
    <rdfs:comment>bridge between humans and machines. HMIs provide intuitive interfaces for users to interact with machines, facilitating safe and efficient operation</rdfs:comment>
  </owl:Class>

  <!-- Robotics and Automation Subclass-->
  <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/MechatronicsAndControlSystem#RoboticsAndAutomation">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>

```

```

    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#MechatronicsAn
dControlSystem"/>
    <rdfs:label>Robotics and Automation</rdfs:label>
    <rdfs:comment></rdfs:comment>
</owl:Class>

<!-- Embedded Systems Subclass-->
<owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/MechatronicsAndCo
ntrolSystem#EmbeddedSystems">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#MechatronicsAn
dControlSystem"/>
    <rdfs:label>Embedded Systems</rdfs:label>
    <rdfs:comment>Brains of smart machines. Embedded systems are dedicated
computers embedded within devices, controlling their functions and interacting
with sensors and actuators in real-time</rdfs:comment>
</owl:Class>

<!-- Defining the subcategories under Quality Control -->
<!-- Real Time Monitoring Subclass-->
<owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/QualityControl#Re
alTimeMonitoring">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#QualityControl
"/>
    <rdfs:label>Real Time Monitoring</rdfs:label>
    <rdfs:comment>Allows engineers to monitor mechanical systems in real-
time, detect anomalies, and predict potential failures before they occur,
ensuring uptime and preventing costly breakdowns</rdfs:comment>
</owl:Class>

<!-- Risk Assesment Subclass-->
<owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/QualityControl#Ri
skAssesment">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>

```

```

        <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#QualityControl
"/>
        <rdfs:label>Risk Assessment</rdfs:label>
        <rdfs:comment>Identifies potential hazards and their likelihood in
mechanical systems, allowing engineers to implement safety measures and mitigate
risks before they materialize</rdfs:comment>
        </owl:Class>

        <!-- Continuous Improvement Subclass-->
        <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/QualityControl#Co
ntinuousImprovement">
            <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
            <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#QualityControl
"/>
            <rdfs:label>Continuous Improvement</rdfs:label>
            <rdfs:comment>Seeking ways to enhance the performance, efficiency, and
value of mechanical systems through iterative learning and
optimization</rdfs:comment>
            </owl:Class>

        <!-- Inspection Subclass-->
        <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/QualityControl#In
spection">
            <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
            <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#QualityControl
"/>
            <rdfs:label>Inspection</rdfs:label>
            <rdfs:comment>Inspections involve systematic examinations of mechanical
systems to identify defects, assess wear and tear, and ensure compliance with
safety regulations</rdfs:comment>
            </owl:Class>

        <!-- Defect Analysis Subclass-->
        <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/QualityControl#De
fectAnalysis">
            <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    
```

```

    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#QualityControl
"/>
    <rdfs:label>Defect Analysis</rdfs:label>
    <rdfs:comment>Defect analysis delves into the root causes of failures in
mechanical systems, providing valuable insights for preventing future occurrences
and improving design, materials, or manufacturing processes</rdfs:comment>
  </owl:Class>

  <!-- Defining the subcategories under Education and Training -->
  <!-- Concept Visualization Subclass-->
  <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/EducationAndTrain
ing#ConceptVisualization">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#EducationAndTr
aining"/>
    <rdfs:label>Concept Visualization</rdfs:label>
    <rdfs:comment></rdfs:comment>
  </owl:Class>

  <!-- Assesments and Feedback Subclass-->
  <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/EducationAndTrain
ing#AssesmentsAndFeedback">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#EducationAndTr
aining"/>
    <rdfs:label>Assesments and Feedback</rdfs:label>
    <rdfs:comment>Assessments and feedback loops provide constructive
criticism and performance benchmarks for both engineers and systems, fostering
continuous improvement and innovation</rdfs:comment>
  </owl:Class>

  <!-- Collaborative Learning Platform Subclass-->
  <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/EducationAndTrain
ing#CollaborativeLearningPlatform">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>

```

```

        <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#EducationAndTr
aining"/>
        <rdfs:label>Collaborative Learning Platform</rdfs:label>
        <rdfs:comment>Collaborative learning encourages engineers to share
expertise, brainstorm ideas, and solve problems together, leading to more
effective and innovative solutions</rdfs:comment>
    </owl:Class>

    <!--Personalized Learning Subclass-->
    <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/EducationAndTrain
ing#PersonalizedLearning">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
        <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#EducationAndTr
aining"/>
        <rdfs:label>Personalized Learning</rdfs:label>
        <rdfs:comment></rdfs:comment>
    </owl:Class>

    <!-- Online Resources and Communities Subclass-->
    <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/EducationAndTrain
ing#OnlineResourcesAndCommunities">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
        <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#EducationAndTr
aining"/>
        <rdfs:label>Online Resources and Communities</rdfs:label>
        <rdfs:comment> Online resources and communities offer engineers access to
vast repositories of information, tutorials, and discussions, fostering
continuous learning and collaboration</rdfs:comment>
    </owl:Class>

    <!-- Defining the subcategories under Knowledge Discovery -->
    <!-- Data Integration Subclass-->
    <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/KnowledgeDiscover
y#DataIntegration">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    
```



```

    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#KnowledgeDiscovery"/>
    <rdfs:label>Data Integration</rdfs:label>
    <rdfs:comment>combines information from various sources like sensors,
maintenance records, and design platforms, providing a holistic view of
mechanical systems and enabling data-driven insights for optimization and
predictive maintenance</rdfs:comment>
  </owl:Class>

  <!-- Semantic Search Subclass-->
  <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/KnowledgeDiscovery#SemanticSearch">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#KnowledgeDiscovery"/>
    <rdfs:label>Semantic Search</rdfs:label>
    <rdfs:comment>Semantic search techniques understand the meaning behind
engineering terms and concepts, allowing engineers to find relevant information
with greater precision and efficiency</rdfs:comment>
  </owl:Class>

  <!-- Knowledge Patterns Subclass-->
  <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/KnowledgeDiscovery#KnowledgePatterns">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#KnowledgeDiscovery"/>
    <rdfs:label>Knowledge Patterns</rdfs:label>
    <rdfs:comment>Knowledge patterns identify recurring relationships and
trends within mechanical data, offering valuable insights for design
optimization, predictive maintenance, and anomaly detection</rdfs:comment>
  </owl:Class>

  <!--Material Development Subclass-->
  <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/KnowledgeDiscovery#MaterialDevelopment">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>

```

```

    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#KnowledgeDisco
very"/>
    <rdfs:label>Material Development</rdfs:label>
    <rdfs:comment>Material development focuses on creating new materials with
superior properties like strength, lightweightness, and sustainability, enabling
engineers to design innovative and high-performance systems</rdfs:comment>
  </owl:Class>

  <!-- Process Development Subclass-->
  <owl:Class
rdf:about="http://www.linkeddatatools.com/mechanicalEngineering/KnowledgeDiscover
y#ProcessDevelopment">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <rdfs:subClassOf
rdf:resource="http://www.linkeddatatools.com/mechanicalEngineering#KnowledgeDisco
very"/>
    <rdfs:label>Process Development</rdfs:label>
    <rdfs:comment>Process development involves optimizing manufacturing and
assembly processes for efficiency, accuracy, and cost-effectiveness, ensuring
consistency and quality in the final product</rdfs:comment>
  </owl:Class>
</rdf:RDF>

```

2.5 Pard (d)

W3C validation was done

Validation Results			
Your RDF document validated successfully.			
Triples of the Data Model			
Number	Subject	Predicate	Object
1	http://www.linkeddatatools.com/Thing	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
2	http://www.linkeddatatools.com/Thing	http://www.w3.org/2000/01/rdf-schema#label	"Thing"
3	http://www.linkeddatatools.com/Thing	http://www.w3.org/2000/01/rdf-schema#comment	"Defines The universal root category"
4	http://www.linkeddatatools.com/Thing	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Ontology
5	http://www.linkeddatatools.com/Thing	http://purl.org/dc/elements/1.1/title	"Mechanical Engineering"
6	http://www.linkeddatatools.com/Thing	http://purl.org/dc/elements/1.1/description	"An abstract ontology designed to give a brief engineering and it's fundamentals."
7	http://www.linkeddatatools.com/Thing#MechanicalEngineering	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
8	http://www.linkeddatatools.com/Thing#MechanicalEngineering	http://www.w3.org/2000/01/rdf-schema#label	"Mechanical Engineering"
9	http://www.linkeddatatools.com/Thing#MechanicalEngineering	http://www.w3.org/2000/01/rdf-schema#comment	"Represents most of the mechanical engineering"
10	http://www.linkeddatatools.com/Thing#MechanicalEngineering#EngineeringDesign	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
11	http://www.linkeddatatools.com/Thing#MechanicalEngineering#EngineeringDesign	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
12	http://www.linkeddatatools.com/Thing#MechanicalEngineering#EngineeringDesign	http://www.w3.org/2000/01/rdf-schema#subClassOf	http://www.linkeddatatools.com/Thing#MechanicalEngineering
13	http://www.linkeddatatools.com/Thing#MechanicalEngineering#EngineeringDesign	http://www.w3.org/2000/01/rdf-schema#label	"Engineering Design"
14	http://www.linkeddatatools.com/Thing#MechanicalEngineering#EngineeringDesign	http://www.w3.org/2000/01/rdf-schema#comment	"Creates efficient machines by turning ideas into materials, identifies flaws, and generates a design."
15	http://www.linkeddatatools.com/Thing#MechanicalEngineering#KnowledgeDiscovery	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
16	http://www.linkeddatatools.com/Thing#MechanicalEngineering#KnowledgeDiscovery	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
17	http://www.linkeddatatools.com/Thing#MechanicalEngineering#KnowledgeDiscovery	http://www.w3.org/2000/01/rdf-schema#subClassOf	http://www.linkeddatatools.com/Thing#MechanicalEngineering
18	http://www.linkeddatatools.com/Thing#MechanicalEngineering#KnowledgeDiscovery	http://www.w3.org/2000/01/rdf-schema#label	"Knowledge Discovery"
19	http://www.linkeddatatools.com/Thing#MechanicalEngineering#KnowledgeDiscovery	http://www.w3.org/2000/01/rdf-schema#comment	"Gives valuable insights from mechanical data maintenance records. It predicts failures, and existing practices."
20	http://www.linkeddatatools.com/Thing#MechanicalEngineering#QualityControl	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
21	http://www.linkeddatatools.com/Thing#MechanicalEngineering#QualityControl	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
22	http://www.linkeddatatools.com/Thing#MechanicalEngineering#QualityControl	http://www.w3.org/2000/01/rdf-schema#subClassOf	http://www.linkeddatatools.com/Thing#MechanicalEngineering

SPARQL Query Process and Results Description

1. Querying Main Mechanical Engineering Categories

SPARQL Query : This query identifies all main subcategories under Mechanical Engineering.

PREFIX rdf: <<http://www.w3.org/1999/02/22-rdf-syntax-ns#>>

PREFIX rdfs: <<http://www.w3.org/2000/01/rdf-schema#>>

SELECT ?category ?label

WHERE {

?category rdf:type owl:Class .

?category rdfs:subClassOf <<http://www.linkeddatatools.com/Thing#MechanicalEngineering>> .

?category rdfs:label ?label .

}

SPARQL Endpoint

/Mechanical_Engineering/query

Content Type (SELECT)

JSON

Content Type (GRAPH)

Turtle

1

PREFIX owl: <http://www.w3.org/2002/07/owl#>

2

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

3

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

4

5

SELECT ?category ?label

6

WHERE {

7

?category rdf:type owl:Class.

8

?category rdfs:subClassOf <http://www.linkeddatatools.com/Thing#MechanicalEngineering> .

9

?category rdfs:label ?label .

10

}

Table

Response

5 results in 0.082 seconds

Simple view

Ellipse

Filter query results

Page size: 50

category	label
<http://www.linkeddatatools.com/Thing#MechanicalEngineering#EngineeringDesign>	Engineering Design
<http://www.linkeddatatools.com/Thing#MechanicalEngineering#KnowledgeDiscovery>	Knowledge Discovery
<http://www.linkeddatatools.com/Thing#MechanicalEngineering#QualityControl>	Quality Control
<http://www.linkeddatatools.com/Thing#MechanicalEngineering#EducationAndTraining>	Education and Training
<http://www.linkeddatatools.com/Thing#MechanicalEngineering#MechatronicsAndControlSystems>	Mechatronics and Control Systems

Showing 1 to 5 of 5 entries

<

1

>

Expected Result: The result will list down all main subcategories under the Mechanical Engineering umbrella, along with their respective labels.

2. Querying a Specific Subcategory's Definition

SPARQL Query: This query fetches the definition and comment for a specific subcategory, for instance, 'Engineering Design'.

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?comment

WHERE {

<http://www.linkeddatatools.com/Thing/MechanicalEngineering#EngineeringDesign> rdfs:comment
 ?comment .
 }

Expected Result: The result will display the definition and description (comment) associated with the 'Engineering Design' subcategory.



The screenshot shows a SPARQL query interface. The query is as follows:

```

1. PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2.
3. SELECT ?comment
4. WHERE {
5.   <http://www.linkeddatatools.com/Thing/MechanicalEngineering#EngineeringDesign> rdfs:comment ?comment .
6. }
  
```

The interface shows the query was executed successfully, returning 1 result in 0.019 seconds. The result is displayed in a table with the following content:

comment
1 Creates efficient machines by turning ideas into practical plans. It recommends materials, identifies flaws, and generates alternative solutions.

The interface also includes options for content type (JSON, Turtle), a filter for query results, and a page size of 50.

3. Querying Sub-subcategories Under a Specific Category

SPARQL Query: This query identifies the sub-subcategories under a specific category, such as 'Engineering Design'.

performance?

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?subcategory ?label

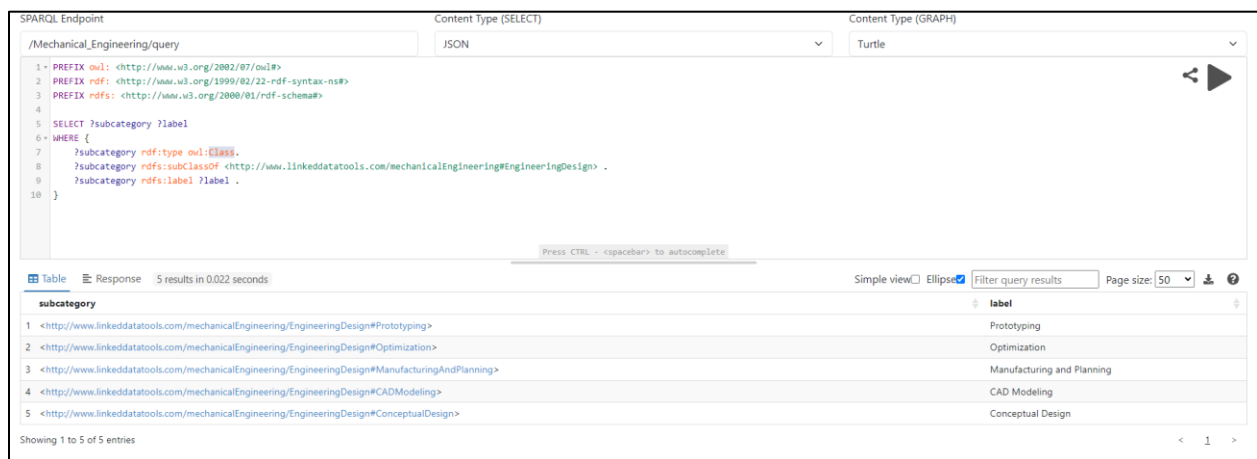
WHERE {

?subcategory rdfs:type owl:Class .

?subcategory rdfs:subClassOf

<http://www.linkeddatatools.com/mechanicalEngineering#EngineeringDesign> .

?subcategory rdfs:label ?label .



The screenshot shows a SPARQL query interface. The query is as follows:

```

PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?subcategory ?label
WHERE {
  ?subcategory rdfs:type owl:Class .
  ?subcategory rdfs:subClassOf <http://www.linkeddatatools.com/mechanicalEngineering#EngineeringDesign> .
  ?subcategory rdfs:label ?label .
}

```

The results are displayed in a table with 2 columns: **subcategory** and **label**. There are 5 results shown.

subcategory	label
<http://www.linkeddatatools.com/mechanicalEngineering/EngineeringDesign#Prototyping>	Prototyping
<http://www.linkeddatatools.com/mechanicalEngineering/EngineeringDesign#Optimization>	Optimization
<http://www.linkeddatatools.com/mechanicalEngineering/EngineeringDesign#ManufacturingAndPlanning>	Manufacturing and Planning
<http://www.linkeddatatools.com/mechanicalEngineering/EngineeringDesign#CADModeling>	CAD Modeling
<http://www.linkeddatatools.com/mechanicalEngineering/EngineeringDesign#ConceptualDesign>	Conceptual Design

Showing 1 to 5 of 5 entries

4. Querying Hierarchical Relationships

SPARQL Query: This query reveals the hierarchical relationships between categories and their subcategories.

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?subcategory ?parentCategory

WHERE {

?subcategory rdfs:type owl:Class .

?subcategory rdfs:subClassOf ?parentCategory .

}

SPARQL Endpoint: /Mechanical_Engineering/query

Content Type (SELECT): JSON

Content Type (GRAPH): Turtle

```

1 PREFIX owl: <http://www.w3.org/2002/07/owl#>
2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
4
5 SELECT ?subcategory ?parentCategory
6 WHERE {
7   ?subcategory rdf:type owl:Class.
8   ?subcategory rdfs:subClassOf ?parentCategory .
9 }

```

Table Response 30 results in 0.028 seconds

Simple view Ellipse Filter query results Page size: 50

subcategory	parentCategory
<http://www.linkeddatatools.com/Thing/MechanicalEngineering#EngineeringDesign>	<http://www.linkeddatatools.com/Thing#MechanicalEngineering>
<http://www.linkeddatatools.com/Thing/MechanicalEngineering#KnowledgeDiscovery>	<http://www.linkeddatatools.com/Thing#MechanicalEngineering>
<http://www.linkeddatatools.com/Thing/MechanicalEngineering#QualityControl>	<http://www.linkeddatatools.com/Thing#MechanicalEngineering>
<http://www.linkeddatatools.com/Thing/MechanicalEngineering#EducationAndTraining>	<http://www.linkeddatatools.com/Thing#MechanicalEngineering>
<http://www.linkeddatatools.com/Thing/MechanicalEngineering#MechatronicsAndControlSystems>	<http://www.linkeddatatools.com/Thing#MechanicalEngineering>
<http://www.linkeddatatools.com/mechanicalEngineering/EngineeringDesign#Prototyping>	<http://www.linkeddatatools.com/mechanicalEngineering#EngineeringDesign>
<http://www.linkeddatatools.com/mechanicalEngineering/EngineeringDesign#Optimization>	<http://www.linkeddatatools.com/mechanicalEngineering#EngineeringDesign>
<http://www.linkeddatatools.com/mechanicalEngineering/EngineeringDesign#ManufacturingAndPlanning>	<http://www.linkeddatatools.com/mechanicalEngineering#EngineeringDesign>
<http://www.linkeddatatools.com/mechanicalEngineering/EngineeringDesign#CADModeling>	<http://www.linkeddatatools.com/mechanicalEngineering#EngineeringDesign>
<http://www.linkeddatatools.com/mechanicalEngineering/EngineeringDesign#ConceptualDesign>	<http://www.linkeddatatools.com/mechanicalEngineering#EngineeringDesign>
<http://www.linkeddatatools.com/mechanicalEngineering/MechatronicsAndControlSystem#SensorsAndActuators>	<http://www.linkeddatatools.com/mechanicalEngineering#MechatronicsAndControlSystem>
<http://www.linkeddatatools.com/mechanicalEngineering/MechatronicsAndControlSystem#MotionControl>	<http://www.linkeddatatools.com/mechanicalEngineering#MechatronicsAndControlSystem>
<http://www.linkeddatatools.com/mechanicalEngineering/MechatronicsAndControlSystem#HumanMachineInterface>	<http://www.linkeddatatools.com/mechanicalEngineering#MechatronicsAndControlSystem>
<http://www.linkeddatatools.com/mechanicalEngineering/MechatronicsAndControlSystem#RoboticsAndAutomation>	<http://www.linkeddatatools.com/mechanicalEngineering#MechatronicsAndControlSystem>
<http://www.linkeddatatools.com/mechanicalEngineering/MechatronicsAndControlSystem#EmbeddedSystems>	<http://www.linkeddatatools.com/mechanicalEngineering#MechatronicsAndControlSystem>
<http://www.linkeddatatools.com/mechanicalEngineering/MechatronicsAndControlSystem#EmbeddedSystems>	<http://www.linkeddatatools.com/mechanicalEngineering#MechatronicsAndControlSystem>
<http://www.linkeddatatools.com/mechanicalEngineering/QualityControl#RealTimeMonitoring>	<http://www.linkeddatatools.com/mechanicalEngineering#QualityControl>
<http://www.linkeddatatools.com/mechanicalEngineering/QualityControl#RiskAssessment>	<http://www.linkeddatatools.com/mechanicalEngineering#QualityControl>
<http://www.linkeddatatools.com/mechanicalEngineering/QualityControl#ContinuousImprovement>	<http://www.linkeddatatools.com/mechanicalEngineering#QualityControl>
<http://www.linkeddatatools.com/mechanicalEngineering/QualityControl#Inspection>	<http://www.linkeddatatools.com/mechanicalEngineering#QualityControl>
<http://www.linkeddatatools.com/mechanicalEngineering/QualityControl#DefectAnalysis>	<http://www.linkeddatatools.com/mechanicalEngineering#QualityControl>
<http://www.linkeddatatools.com/mechanicalEngineering/EducationAndTraining#ConceptVisualization>	<http://www.linkeddatatools.com/mechanicalEngineering#EducationAndTraining>
<http://www.linkeddatatools.com/mechanicalEngineering/EducationAndTraining#AssessmentsAndFeedback>	<http://www.linkeddatatools.com/mechanicalEngineering#EducationAndTraining>
<http://www.linkeddatatools.com/mechanicalEngineering/EducationAndTraining#CollaborativeLearningPlatform>	<http://www.linkeddatatools.com/mechanicalEngineering#EducationAndTraining>
<http://www.linkeddatatools.com/mechanicalEngineering/EducationAndTraining#PersonalizedLearning>	<http://www.linkeddatatools.com/mechanicalEngineering#EducationAndTraining>
<http://www.linkeddatatools.com/mechanicalEngineering/EducationAndTraining#OnlineResourcesAndCommunities>	<http://www.linkeddatatools.com/mechanicalEngineering#EducationAndTraining>
<http://www.linkeddatatools.com/mechanicalEngineering/KnowledgeDiscovery#DataIntegration>	<http://www.linkeddatatools.com/mechanicalEngineering#KnowledgeDiscovery>
<http://www.linkeddatatools.com/mechanicalEngineering/KnowledgeDiscovery#SemanticSearch>	<http://www.linkeddatatools.com/mechanicalEngineering#KnowledgeDiscovery>
<http://www.linkeddatatools.com/mechanicalEngineering/KnowledgeDiscovery#KnowledgePatterns>	<http://www.linkeddatatools.com/mechanicalEngineering#KnowledgeDiscovery>
<http://www.linkeddatatools.com/mechanicalEngineering/KnowledgeDiscovery#MaterialDevelopment>	<http://www.linkeddatatools.com/mechanicalEngineering#KnowledgeDiscovery>
<http://www.linkeddatatools.com/mechanicalEngineering/KnowledgeDiscovery#ProcessDevelopment>	<http://www.linkeddatatools.com/mechanicalEngineering#KnowledgeDiscovery>

Showing 1 to 30 of 30 entries

www.linkeddatatools.com/mechanicalEngineering#EducationAndTraining

5. Querying Related Subclasses

SPARQL Query: This query identifies subcategories related to both 'Quality Control' and 'Education and Training'.

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?subcategory ?label

WHERE {

 ?subcategory rdf:type owl:Class .

 { ?subcategory rdfs:subClassOf
 <http://www.linkeddatatools.com/mechanicalEngineering#QualityControl> . }

UNION

```
{ ?subcategory rdfs:subClassOf
<http://www.linkeddatatools.com/mechanicalEngineering#EducationAndTraining> . }

?subcategory rdfs:label ?label .

}
```

Expected Result: The result will list subcategories that have a relationship with both 'Quality Control' and 'Education and Training'.

SPARQL Endpoint: /Mechanical_Engineering/query

Content Type (SELECT): JSON

Content Type (GRAPH): Turtle

```
1 PREFIX owl: <http://www.w3.org/2002/07/owl#>
2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
4
5 SELECT DISTINCT ?subcategory ?label
6 WHERE {
7   ?subcategory rdf:type owl:Class.
8   { ?subcategory rdfs:subClassOf <http://www.linkeddatatools.com/mechanicalEngineering#QualityControl> . }
9   UNION
10  { ?subcategory rdfs:subClassOf <http://www.linkeddatatools.com/mechanicalEngineering#EducationAndTraining> . }
11  ?subcategory rdfs:label ?label .
12 }
13
```

Table Response 10 results in 0.039 seconds

Simple view Ellipse Filter query results Page size: 50

subcategory	label
<http://www.linkeddatatools.com/mechanicalEngineering/QualityControl#RealTimeMonitoring>	Real Time Monitoring
<http://www.linkeddatatools.com/mechanicalEngineering/QualityControl#RiskAssessment>	Risk Assessment
<http://www.linkeddatatools.com/mechanicalEngineering/QualityControl#ContinuousImprovement>	Continuous Improvement
<http://www.linkeddatatools.com/mechanicalEngineering/QualityControl#Inspection>	Inspection
<http://www.linkeddatatools.com/mechanicalEngineering/QualityControl#DefectAnalysis>	Defect Analysis
<http://www.linkeddatatools.com/mechanicalEngineering/EducationAndTraining#ConceptVisualization>	Concept Visualization
<http://www.linkeddatatools.com/mechanicalEngineering/EducationAndTraining#AssessmentsAndFeedback>	Assessments and Feedback
<http://www.linkeddatatools.com/mechanicalEngineering/EducationAndTraining#CollaborativeLearningPlatform>	Collaborative Learning Platform
<http://www.linkeddatatools.com/mechanicalEngineering/EducationAndTraining#PersonalizedLearning>	Personalized Learning
<http://www.linkeddatatools.com/mechanicalEngineering/EducationAndTraining#OnlineResourcesAndCommunities>	Online Resources and Communities

Showing 1 to 10 of 10 entries

3 Question 03 (Maze)

3.1 Overview

The question states to implement a 6*6 random maze with 4 barriers and find the shortest path using A* and DFS algorithms, Python was used as the programming language to cater this problem. Because it has a rich set of libraries and modules that can be used to create and manipulate mazes far more efficiently such as numpy, matplotlib etc.

3.2 Task 01 – Creating a random maze

Under the main.py file two numpy arrays were created 1 to store the node values from 0 - 35 as shown in the specification maze image, and the other to store the random maze layout.

```
# Initializing a np.array to store the maz layout
maze_base = np.array([
    [0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0]
])
# contains the maze node values from 0 to 35
maze_nodes = np.array([[0, 6, 12, 18, 24, 30],
    [1, 7, 13, 19, 25, 31],
    [2, 8, 14, 20, 26, 32],
    [3, 9, 15, 21, 27, 33],
    [4, 10, 16, 22, 28, 34],
    [5, 11, 17, 23, 29, 35]
])
```

Afterwards a list is defined to store the randomly generated barrier node values. 4 barrier nodes are created. When creating the first barrier node it can take any node value from 0 – 35, then when initializing the second barrier node it can also take any node value other than the previous initialized barrier node value. Likewise, four barrier nodes will be created and added to a list.

```
# generate the barriers
barrier_nodes = []
while len(barrier_nodes) < 4: # 4 barriers need to be generated
    barrier_node = rand.randint(0, 35) # total nodes in the maze is 36
    if barrier_node not in barrier_nodes:
        barrier_nodes.append(barrier_node)
    else:
        continue
```

After initializing the barrier nodes, the program will create the start and the end node in order. As mentioned in the specification the program initializes a start node for the maze. It can take a value form 0 – 11 and cannot be a barrier node.

```
while True: # generating a start node
    start_node = rand.randint(0, 11) # TODO create a method to get a
```



```

random node value : improve code quality
# print(start_node)
if start_node not in barrier_nodes and not check_if_possible_path_exist:
    node_index_row = start_node % 6
    node_index_column = start_node // 6
    # start node is denoted as '1' in the 2d maze base layout
    maze_base[(node_index_row, node_index_column)] = 1 # inserting
the start node to the maze base layout

    # node_index = np.where(maze_nodes == start_node)
    # maze_base[node_index[0], node_index[1]] = 1
    break
else:
    continue

```

Additionally, the system checks if the start node is in the top-left or bottom-left corner, if so it will then check whether the start node is surrounded by barrier nodes and has no path to the goal node. If that is the case, then the system will get another node value randomly between 0 - 11 and initialize it as the start node.

```

def check_if_possible_path_exist(barrier_nodes, node): #checks for the
special case : if the start node is fully surrounded by barriers
# Checking the surrounding of StartNode
if node == 5:
    if node + 6 in barrier_nodes and node - 1 in barrier_nodes and node +
5 in barrier_nodes:
        return False
if node == 0:
    if node + 6 in barrier_nodes and node + 1 in barrier_nodes and node +
7 in barrier_nodes:
        return False
if node == 35:
    if node - 6 in barrier_nodes and node - 1 in barrier_nodes and node -
7 in barrier_nodes:
        return False
if node == 30:
    if node - 6 in barrier_nodes and node - 1 in barrier_nodes and node -
7 in barrier_nodes:
        return False

```

The same conditions are applied when a random node between 24 – 35 is initialized as the end node. An additional rule was taken into consideration so that searching could be done for the randomly generated maze.

After considering all of the rules the program will generate a random 6 * 6 maze with a start, end and 4 barrier nodes which has a either 1 or many paths from start to goal. The 6 * 6 numpy array which represents the maze is printed in the PyCharm terminal. The nodes are denoted as follows empty nodes are as '0's, Start node as '1', Goal node as '2' and barrier nodes as '9'.

3.3 Task 02 – DFS Search

The dfs.py file contains the basic core implementation for a DFS search using a list. A in-built or any other data types were not used as the Python data type 'list' was sufficient enough to implement the DFS searching algorithm.

The specified rules were considered when implementing the code for DFS searching algorithm.

- Valid moves are horizontal, vertical, and diagonal only.
- Cannot traverse through barrier nodes.
- The neighbor nodes must be processed in increasing order.
- Time to explore a node is one minute.
- The cost of traversal for all edges are equal.

dfs.py file contains the core implementation of the DFS search algorithm. It has a method 'dfs_search_maze' which takes 2 numpy array as inputs, one contains the random maze layout (variable name : maze_base) and the other is the maze node values. (0 – 35 , maze_nodes).

At first an empty list is assigned to a variable named 'visited_nodes', this will contain all the nodes that is traveled by the agent(in this case the algorithm). Then another empty list is assigned to node_stack, which will contain the to be processed nodes. That means when a node is explored then the system will check its neighbour nodes and insert the nodes to the node_stack list in descending order. If a neighbor node is already in the 'node_stack' or 'visited_node' list then that node will not be added again to the system to avoid recursive infinite loop of searching for path.

```
def dfs_search_maze(maze_base, maze_nodes):
    visited_nodes = []
    node_stack = []
    start_node_maze = getting_node_number(maze_base, 1)
    goal_node_maze = getting_node_number(maze_base, 2)

    barrier_nodes = [] # getting the barrier node values as in the sample
    maze.
    barrier_node_indexes = np.where(maze_base == 9)
    if len(barrier_node_indexes[0]) > 1:
        # Handle the case where there are multiple indices
        for i in range(len(barrier_node_indexes[0])):
            row, col = barrier_node_indexes[0][i], barrier_node_indexes[1][i]
            node_number = col * 6 + row
            barrier_nodes.append(node_number)
    else:
        row, col = barrier_node_indexes[0][0], barrier_node_indexes[1][0]
        node_number = col * 6 + row
        barrier_nodes.append(node_number)
```

as the dfs_search_maze only takes the 2d arrays as arguments, the system may have to find the row, column index of barrier nodes. The following code gives a set of values: the relevant nodes row, column values. As the maze has 4 nodes which has the value 9 assigned to it. The npwhere() function will return a

2d array : 1st row contains all the barrier nodes row index values and the second contains the column index values.

```
barrier_node_indexes = np.where(maze_base == 9)
```

Then the dfs search algorithm begins. Firstly, the program adds the start node to the `stack_node` list and until the stack node is empty or until the goal is found the program will repeat the following functionalities. The last indexed node value is removed from the stack and added to the visited nodes list. Once the node is added to the system then it is considered as the processing node. for this processing node there might be neighbor nodes from 3 to 8. Depending on the position of the node in the maze. So all these neighbor nodes are added to the system according to the descending order. Because the task mentions that the processing must be in the node value increasing order. When checking the neighbor nodes if a node is found to be a barrier, It will not be added to the 'node_stack' list as no traversal could be done through a barrier node. This process will be repeated till the goal is found.

The following code shows how the program enters the nodes to the 'node_stack' list in the specified order.

```
node_stack.append(start_node_maze)
while len(node_stack) > 0: # add more conditions if needed
    visited_nodes.append(node_stack.pop()) # adding the removed node
    value to
    # 'visited nodes' list
    processing_node = visited_nodes[len(visited_nodes) - 1]

    # checking whether processing node is the goal
    if processing_node == goal_node_maze:
        break

    neighbour_nodes = []

    # getting processing nodes row and column values
    row, col = np.where(maze_nodes == processing_node)

    # Check left, right, up, and down and diagonal neighbors
    if row > 0 and col > 0:
        neighbour_nodes.append(maze_nodes[row - 1, col - 1]) # Up-left
    if col > 0:
        neighbour_nodes.append(maze_nodes[row, col - 1]) # Left
    if row < maze_base.shape[0] - 1 and col > 0: # maze.shape[0] return 6
    for this 6*6 maze
        neighbour_nodes.append(maze_nodes[row + 1, col - 1]) # Down-left
    if row > 0:
        neighbour_nodes.append(maze_nodes[row - 1, col]) # Up
    if row < maze_base.shape[0] - 1:
        neighbour_nodes.append(maze_nodes[row + 1, col]) # Down
    if row > 0 and col < maze_base.shape[1] - 1:
        neighbour_nodes.append(maze_nodes[row - 1, col + 1]) # Up-right
    if col < maze_base.shape[1] - 1:
        neighbour_nodes.append(maze_nodes[row, col + 1]) # Right
    if row < maze_base.shape[0] - 1 and col < maze_base.shape[1] - 1:
        neighbour_nodes.append(maze_nodes[row + 1, col + 1]) # Down-right
```

Once the goal is found the system will output the visited node list and calculate the total time taken for dfs search by considering that it took exactly 1 minute to exploring one node. at the beginning of the code total_time_spent is initailaized as 0.

```
# calculated the time taken to find the goal
total_time_spent += 1
```

Once the goal is found then the visited node list is printed in the terminal. Also, the total time spent is also displayed in the terminal. For each iteration of processing nodes + 1 was added to the total time spent. Which is finally equal to len(visited_node) -1, because the program doesn't process nodes around the goal. It stops execution once the goal is found.

3.4 Task 03 – Heuristic value function

```
def get_heuristic_value(maze_nodes, neighbour_node, goal_node):
    p_row, p_column = getting_node_index(maze_nodes, neighbour_node)
    g_row, g_column = getting_node_index(maze_nodes, goal_node)
    return abs(g_row - p_row) + abs(g_column - p_column)
```

The above function gets the maze nodes and considering neighbor node and the goal node as arguments. Then it calculates the neighbor nodes row, column index values, and also calculates the goal nodes row, column index values. Then it takes the sum of the absolute value difference of row indexes and column indexes.

3.5 Task 04 – A* Search

The system begins with getting the start node and goal node row and column index values, Then at the beginning it assigns the start node as the processing node, Now the algorithm starts processing neighbor nodes around the start node. Firstly the system similar to DFS searches the up left node first if it exist for the and gets its heuristic value and assigns it to the lowest_heu_val variable if it is the lowest_heu_value for the following iteration.

```
def a_star_search(maze_nodes, start_node, goal_node, barrier_nodes):
    goal_found = False
    visited_nodes = []
    s_row, s_column = getting_node_index(maze_nodes, start_node)      #
    getting_start_node_row, column_value
    g_row, g_column = getting_node_index(maze_nodes, goal_node)      # getting
    goal_node_row, column_value

    # path_value = 1
    # heuristic_value = 1000      # for this maze 1000 works as a infinity
    amount cuz the heuristic + path values total is for any case is less than 1000
    # total_travel_value = heuristic_value + path_value

    p_row, p_column = s_row, s_column      # at first the 'processing_node'
```

```
is start node
    process_node = start_node
```

then It checks if left_neighbor node exists or not if exist calculates the heuristic value and if it is less than the previous calculated heuristic value then it assigns it to lowest_heu_value . likewise this process is done until all the neighbor nodes heuristic values are calculated. At the end after considering all the neighbor nodes the the algorithm moves to the lowest heuristic value node, likewise this process repeats until the goal is found.

```
while not goal_found:
    # Check left, right, up, and down and diagonal neighbors
    total_time += 1
    lowest_heu_value = None
    lowest_heu_value_node = None

    # checking if an Up-left node is available for the processing node
    if p_row > 0 and p_column > 0 and process_node not in barrier_nodes:
        # if exist getting the new upLeft neighbour node's value using the
        # relevant row, column values.
        up_left_neighbour_node = getting_node_value(p_row - 1, p_column - 1)
        up_left_neighbour_node_heu_val = get_heuristic_value(maze_nodes,
        up_left_neighbour_node, goal_node) # getting the heuristic value

        if lowest_heu_value is None or lowest_heu_value >
        up_left_neighbour_node_heu_val:
            lowest_heu_value = up_left_neighbour_node_heu_val
            lowest_heu_value_node = up_left_neighbour_node

    # Left
    if p_column > 0 and process_node not in barrier_nodes:
        left_neighbour_node = getting_node_value(p_row, p_column - 1)
        left_neighbour_node_heu_val = get_heuristic_value(maze_nodes,
        left_neighbour_node, goal_node)
        if lowest_heu_value is None or lowest_heu_value >
        left_neighbour_node_heu_val:
            lowest_heu_value = left_neighbour_node_heu_val
            lowest_heu_value_node = left_neighbour_node

    # Down-left
    if p_row < maze_nodes.shape[0] - 1 and p_column > 0 and process_node not
    in barrier_nodes: # maze.shape[0] return 6 for this 6*6 maze
        down_left_neighbour_node = getting_node_value(p_row + 1, p_column -
        1)
        down_left_neighbour_node_heu_val = get_heuristic_value(maze_nodes,
        down_left_neighbour_node, goal_node)
        if lowest_heu_value is None or lowest_heu_value >
        down_left_neighbour_node_heu_val:
            lowest_heu_value = down_left_neighbour_node_heu_val
            lowest_heu_value_node = down_left_neighbour_node

    # Up
    if p_row > 0 and process_node not in barrier_nodes:
        up_neighbour_node = getting_node_value(p_row - 1, p_column)
```

```

        up_neighbour_node_heu_val = get_heuristic_value(maze_nodes,
up_neighbour_node, goal_node)
        if lowest_heu_value is None or lowest_heu_value >
up_neighbour_node_heu_val:
            lowest_heu_value = up_neighbour_node_heu_val
            lowest_heu_value_node = up_neighbour_node

    # down
    if p_row < maze_nodes.shape[0] - 1 and process_node not in barrier_nodes:
        down_neighbour_node = getting_node_value(p_row + 1, p_column)
        down_neighbour_node_heu_val = get_heuristic_value(maze_nodes,
down_neighbour_node, goal_node)
        if lowest_heu_value is None or lowest_heu_value >
down_neighbour_node_heu_val:
            lowest_heu_value = down_neighbour_node_heu_val
            lowest_heu_value_node = down_neighbour_node

    # up_right
    if p_row > 0 and p_column < maze_nodes.shape[1] - 1 and process_node not
in barrier_nodes:
        up_right_neighbour_node = getting_node_value(p_row - 1, p_column + 1)
        up_right_neighbour_node_heu_val = get_heuristic_value(maze_nodes,
up_right_neighbour_node, goal_node)
        if lowest_heu_value is None or lowest_heu_value >
up_right_neighbour_node_heu_val:
            lowest_heu_value = up_right_neighbour_node_heu_val
            lowest_heu_value_node = up_right_neighbour_node

    # right
    if p_column < maze_nodes.shape[1] - 1 and process_node not in
barrier_nodes:
        right_neighbour_node = getting_node_value(p_row, p_column + 1)
        right_neighbour_node_heu_val = get_heuristic_value(maze_nodes,
right_neighbour_node, goal_node)
        if lowest_heu_value is None or lowest_heu_value >
right_neighbour_node_heu_val:
            lowest_heu_value = right_neighbour_node_heu_val
            lowest_heu_value_node = right_neighbour_node

    # down_right
    if p_row < maze_nodes.shape[0] - 1 and p_column < maze_nodes.shape[1] - 1
and process_node not in barrier_nodes:
        down_right_neighbour_node = getting_node_value(p_row + 1, p_column +
1)
        down_right_neighbour_node_heu_val = get_heuristic_value(maze_nodes,
down_right_neighbour_node, goal_node)
        if lowest_heu_value is None or lowest_heu_value >
down_right_neighbour_node_heu_val:
            lowest_heu_value = down_right_neighbour_node_heu_val
            lowest_heu_value_node = down_right_neighbour_node

    # changing the new processing node which has the lowest heuristic value
to use it for the next iteration
    p_row, p_column = getting_node_index(maze_nodes, lowest_heu_value_node)

```

```
process_node = getting_node_value(p_row, p_column)

visited_nodes.append(lowest_heu_value_node)

if p_row == g_row and p_column == g_column:
    goal_found = True

print("Visited nodes : ")
for node in visited_nodes:
    print(node, end=' ')
```

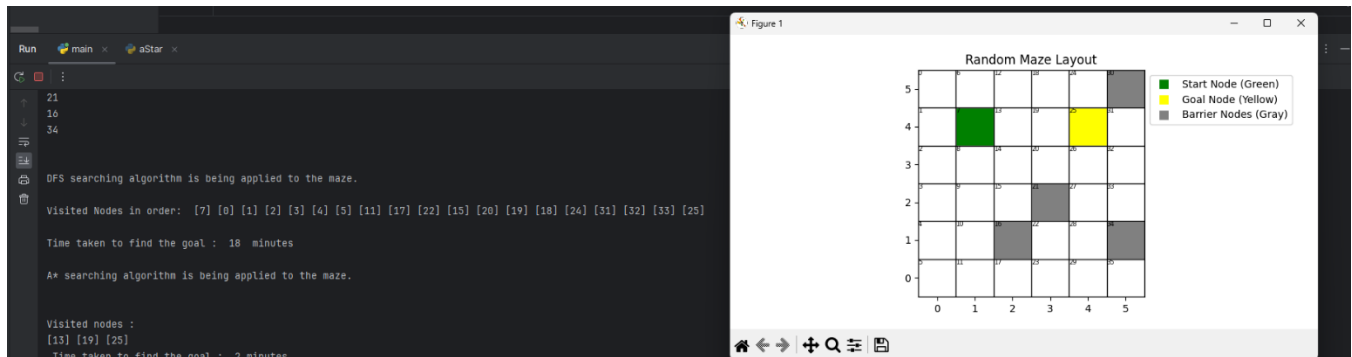
So for one of these iterations as for the specification it takes 1 minute. Then the Total time spent will be equal to length of visited list- 1.

After each loop the the lowest_heu_value and lowest_heu_value_node is assigned to null.

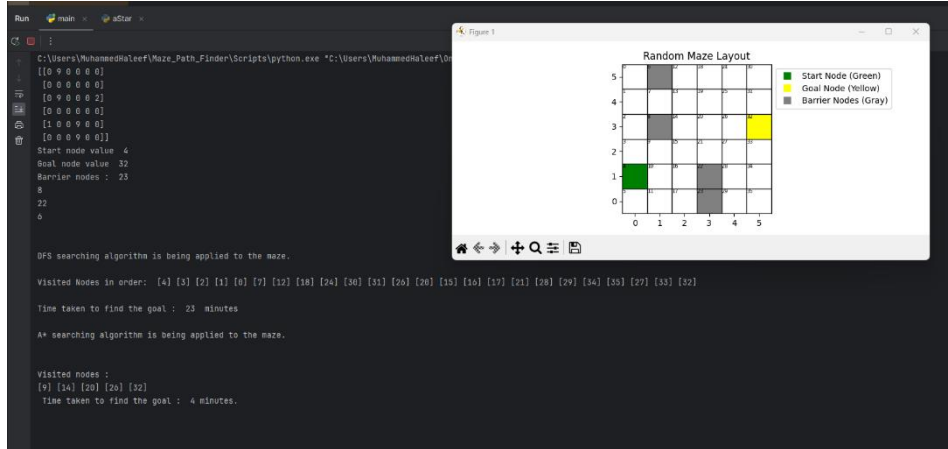
```
lowest_heu_value = None
lowest_heu_value_node = None
```

3.6 Task 05

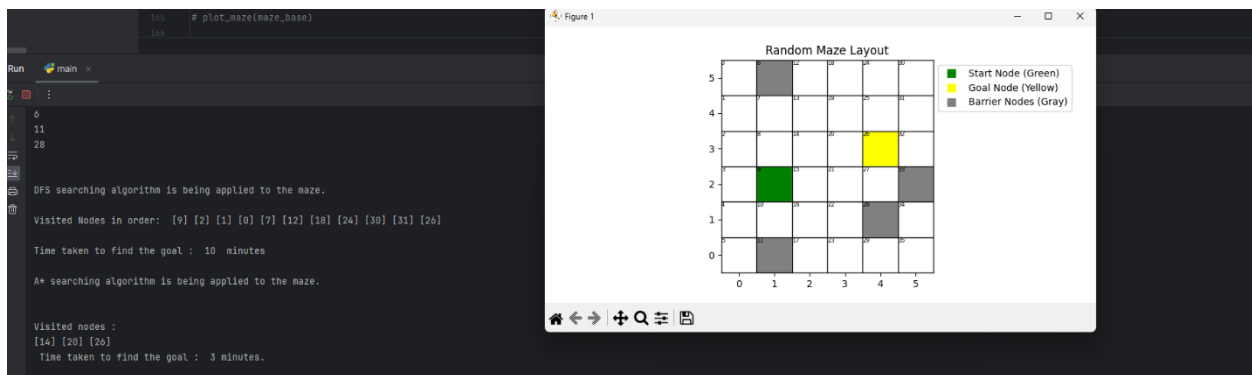
Output 1:



Output 2:



Output 3:



Completeness : both search algorithms find a path for the random generated maze. Therefore both can be considered as complete algorithms.

Optimality : As A* considers the heuristic values respective to the goal node it finds the path to goal in the most optimal way. In most of the cases DFS algorithm will not find the optimal path.

Output value	01		02		03	
Algorithm	A*	DFS	A*	DFS	A*	DFS
Time Complexity	2	18	23	4	3	10

4 Question 04 (Fuzzy-Logic)

Overview:

Implementation:

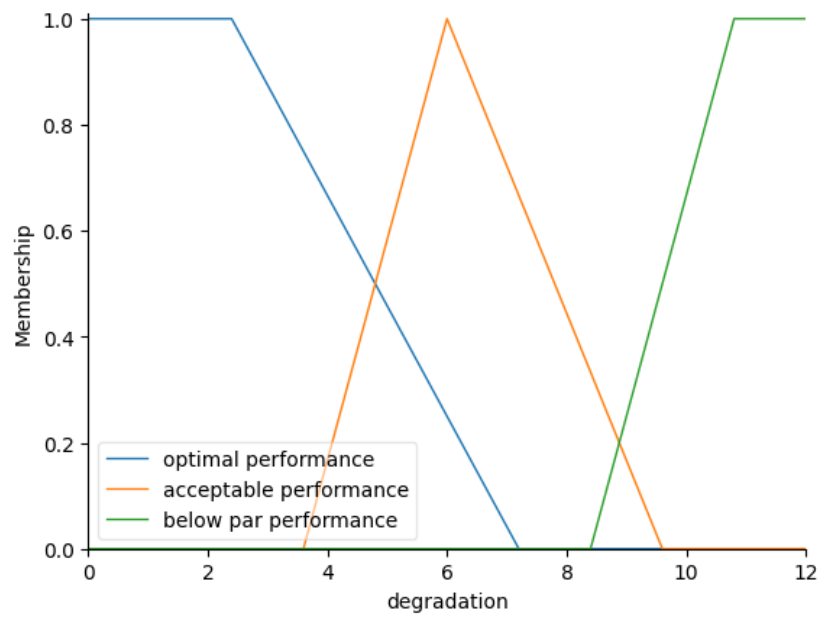
After getting a small understanding about an error detection and mitigation system. Defined the user input crisp variables value scale.

```
degradation = ctrl.Antecedent(np.arange(0, 12.1, 0.1), 'degradation') #10
years : 120 months
redundancy = ctrl.Antecedent(np.arange(0, 11, 1), 'redundancy') #10 same
files
error_history = ctrl.Antecedent(np.arange(0, 101, 1), 'error_history') # no
of errors occurred
```

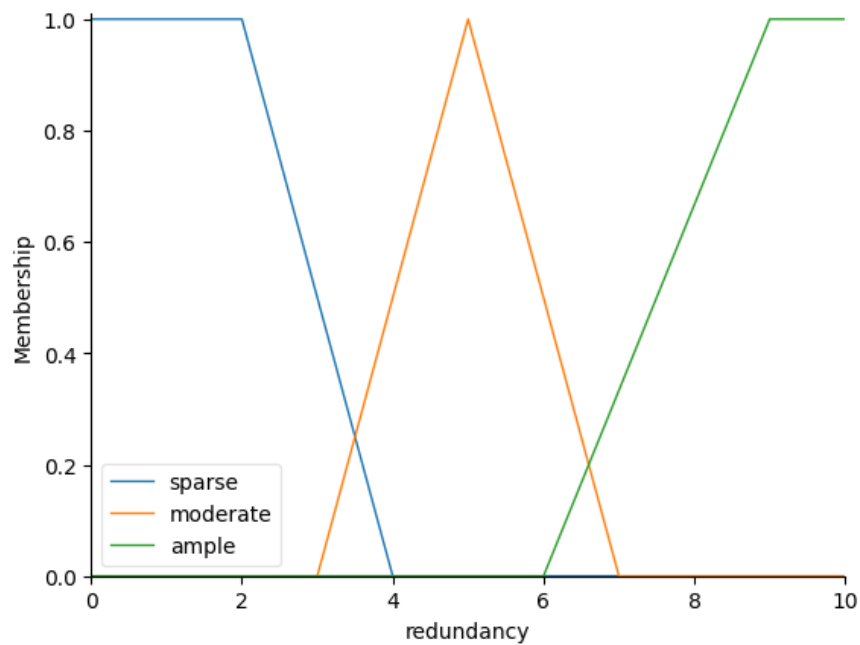
The degradation level value scale was defined considering months, from 0 to 12.1 each 0.1 denotes a month, the total range is 10 years. Then for Redundancy the range was defined assuming the number of replicated/backup files available. Then Error history was taken as the number of errors occurred in the past, which is defined as 0 to 101.

Afterwards the degree of membership function for each of these input variables were defined. Initially used a very basic membership function for all these input variables, then to get a some-what optimized membership function and fuzzy value set small fine tuning and improved the membership functions to give a better fuzzy value set for a crisp input value.

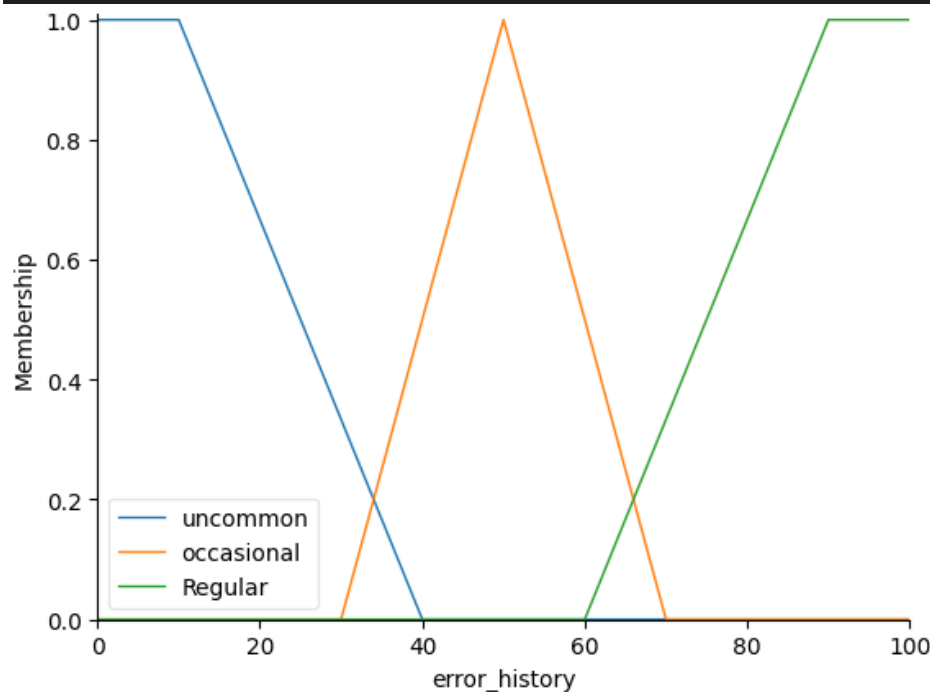
```
# define membership function for 'degradation'
degradation['optimal performance'] = mf.trapmf(degradation.universe, [0, 0,
2.4, 7.2])
degradation['acceptable performance'] = mf.trimf(degradation.universe, [3.6,
6, 9.6])
degradation['below par performance'] = mf.trapmf(degradation.universe, [8.4,
10.8, 12, 12])
# displaying the degradation membership function plot
# degradation.view()
```



```
#define membership function for 'error history'
redundancy['sparse'] = mf.trapmf(redundancy.universe, [0, 0, 2, 4])
redundancy['moderate'] = mf.trimf(redundancy.universe, [3, 5, 7])
redundancy['ample'] = mf.trapmf(redundancy.universe, [6, 9, 10, 10])
# displaying the redundancy membership function plot
# redundancy.view()
```



```
#defining membership function for 'error history'
error_history['uncommon'] = mf.trapmf(error_history.universe, [0, 0, 10, 40])
error_history['occasional'] = mf.trimf(error_history.universe, [30, 50, 70])
error_history['Regular'] = mf.trapmf(error_history.universe, [60, 90, 100, 100])
# displaying the error_history membership function plot
# error_history.view()
```



When defining the fuzzy value set used a set of more meaningful words for the given input crisp values. for example, rather than dividing error_history into high, medium and low, defined the sets as uncommon, occasional and regular. Followed the same when defining the other fuzzy sets as well.

mostly used trapezoids to define each fuzzy set because using other shapes requires substantial domain knowledge and extensive research. Therefore, for the given context, I utilized trapezoids and triangles to represent the fuzzy values.

Afterwards defined the output variables fuzzy value set and the membership function.

```
# define the output variable
severe_error_likelyhood = ctrl.Consequent(np.arange(0,101,1),
'severe_error_likelyhood')
severe_error_likelyhood['severe'] =
fuzz.trapmf(severe_error_likelyhood.universe, [60, 85, 100, 100])
severe_error_likelyhood['moderate'] =
fuzz.trimf(severe_error_likelyhood.universe, [20, 50, 70])
```

```
severe_error_likelihood['low'] =
fuzz.trapmf(severe_error_likelihood.universe, [0, 0, 25, 40])
```

Then defined 3 fuzzy rule to consider when calculating the output variable fuzzy set values.

```
rule1 = ctrl.Rule((error_history['Regular'] | degradation['below par
performance'] | redundancy['sparse'] ) & ~redundancy['ample'],
severe_error_likelihood['severe'])    #low redundancy doesnt hugely that
it is an severe error
rule2 = ctrl.Rule(error_history['occasional'] | degradation['acceptable
performance'] | ~redundancy['moderate'],
severe_error_likelihood['moderate'])
rule3 = ctrl.Rule(error_history['uncommon'] | degradation['optimal
performance'] | redundancy['ample'], severe_error_likelihood['low'])
```

Rule 1 is defined to substitute the min/ max values of input variables fuzzy set which affects the severe_error_likelihoods ['severe'] fuzzy value. If the error history is regular(high), and data redundancy is sparse(low) and degradation is below-par(low) the output variable is set to the maximum value of these input variables. Then again having an ample no of files increases the data redundancy and reduces the severe error likelihood. So the complement value (1- value) was also considered in rule 1.

Likewise included 2 more rules which define what are the input variables fuzzy sets which affect the error likelihood's fuzzy values moderate and low.

The following code gets user input crisp values and prints the severe_error_likelihood value and display the relevant graph.

1. Output

This is an output of severe error likelihood value = 35% calculated on the following input variable values;

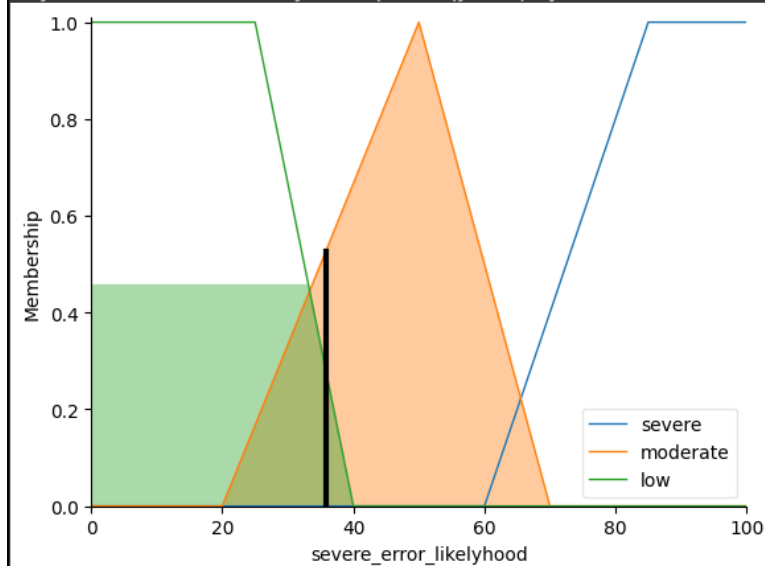
Data redundancy – 4, degradation level – 50 and error history – 50.

Since most of the crisp values are in a medium level and there is a medium redundancy. The output is severe error likelihood is reduced.

```
Enter the following required values to find the severe error-likelihood percentage and mitigation suggestions.
Make sure to enter values belonging to the required range.
Enter no of duplicate files(data redundancy) [0 - 10]: 4
Enter how old is the system(degradation level) ([0 - 120]: 50
No of errors occurred in the past(error history) [0-100]: 50
```

Severe Error Likelihood: 35%

Do you want to view the fuzzy value plots? (yes/no): yes



```
Enter the following required values to find the severe error-likelihood percentage and mitigation suggestions.
Make sure to enter values belonging to the required range.
Enter no of duplicate files(data redundancy) [0 - 10]: 2
Enter how old is the system(degradation level) ([0 - 120]: 50
No of errors occurred in the past(error history) [0-100]: 50
```

Severe Error Likelihood: 56%

Do you want to view the fuzzy value plots? (yes/no): yes

