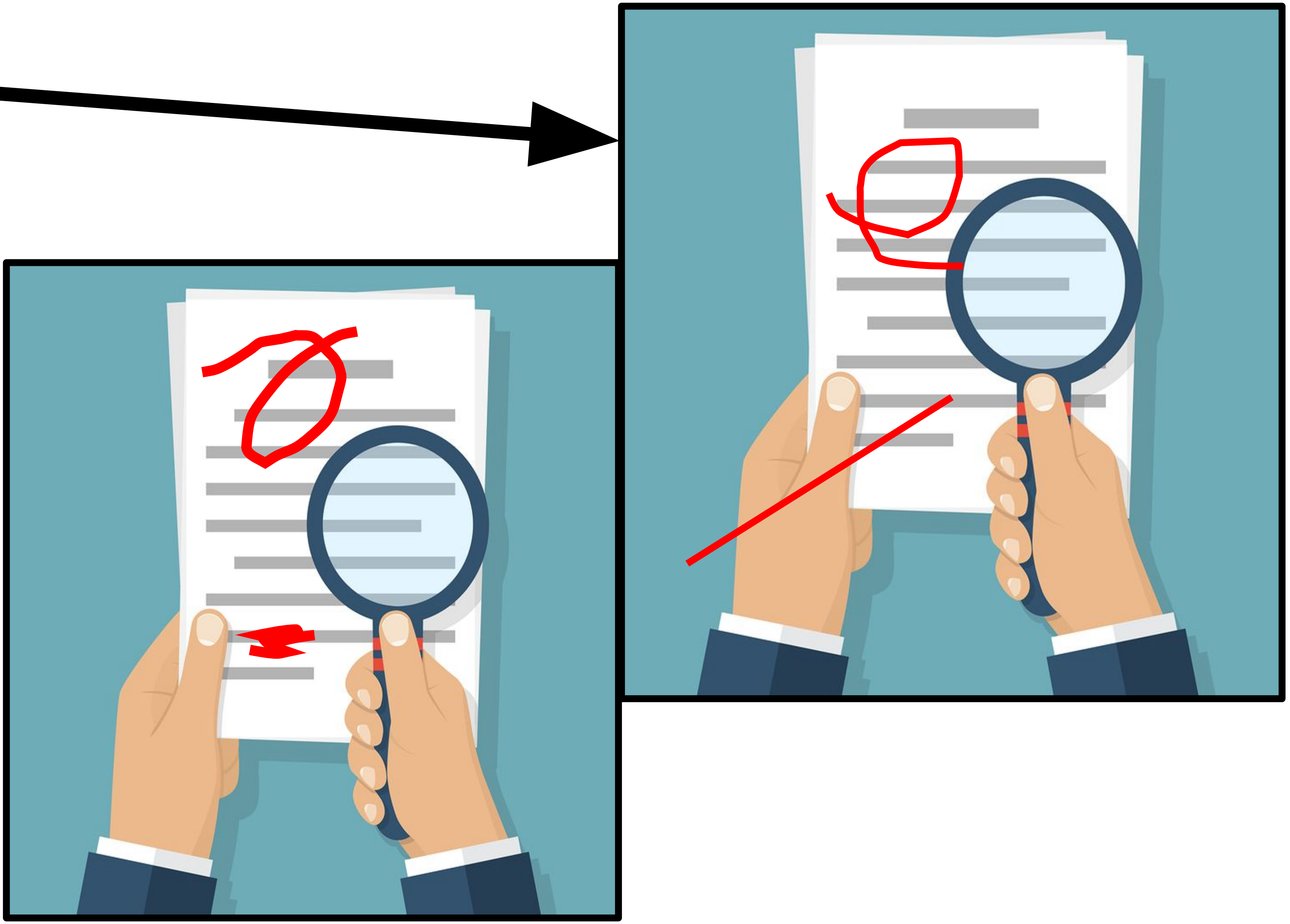
CS 301
Requirements
Characteristics

Eswaran Narasimhan

12 – Sep - 2024

Software Requirements Validation

- □ **Commenting**
- □ Inspections
- □ Walk-throughs
- □ Perspective-based reading
- □ Validation through prototypes
- □ Using checklists for validation



Software Requirements Validation

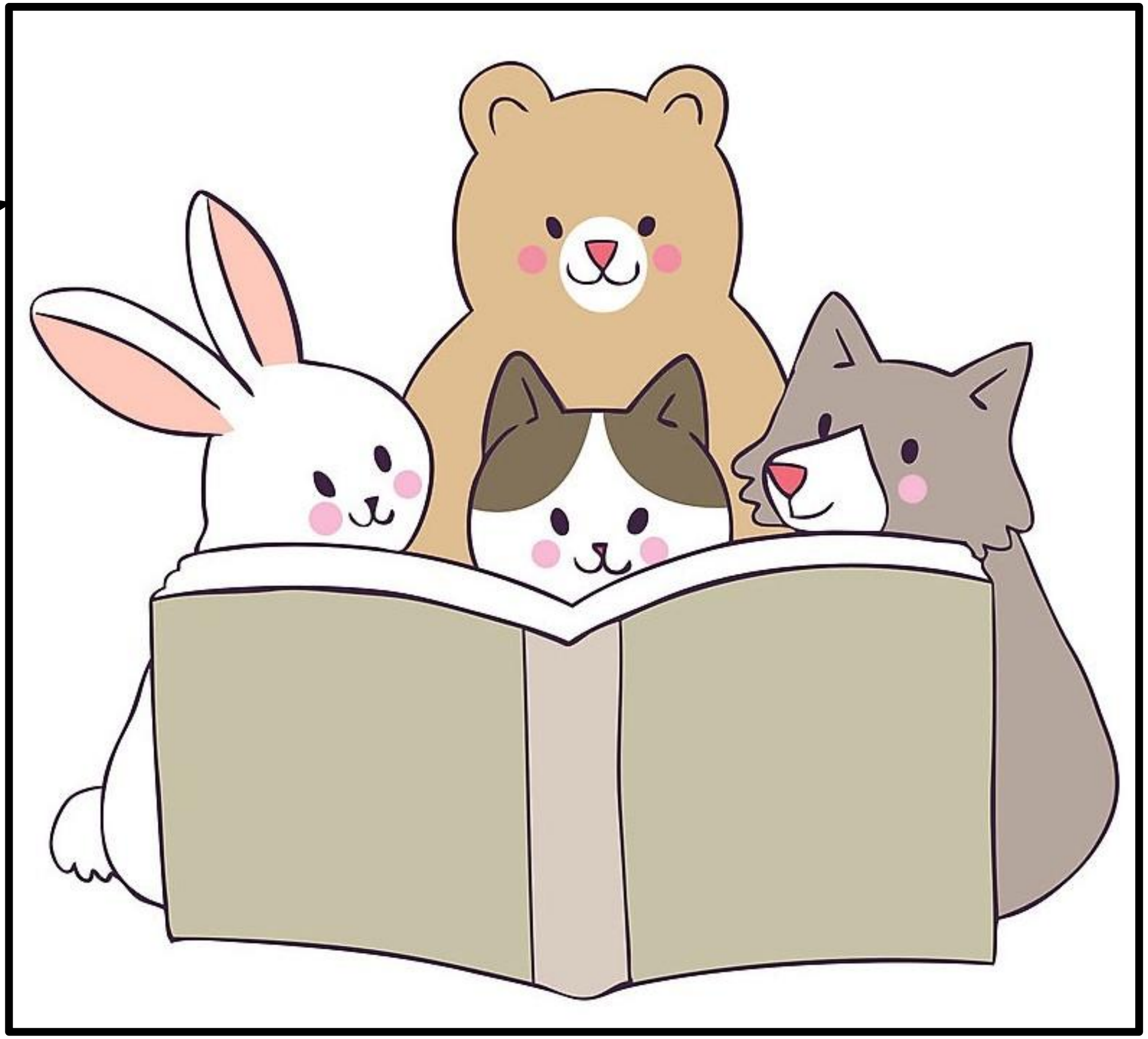
- ❑ Commenting
- ❑ Inspections
- ❑ Walk-throughs
- ❑ Perspective-based reading
- ❑ Validation through prototypes
- ❑ Using checklists for validation

Software Requirements Validation

- ❑ **Commenting**
- ❑ **Inspections**
- ❑ **Walk-throughs**
- ❑ Perspective-based reading
- ❑ Validation through prototypes
- ❑ Using checklists for validation

Software Requirements Validation

- □ Commenting
- □ Inspections
- □ Walk-throughs
- □ Perspective-based reading
- □ Validation through prototypes
- □ Using checklists for validation



Software Requirements Validation

- □ Commenting
- □ Inspections
- □ Walk-throughs
- □ Perspective-based reading
- □ Validation through prototypes
- □ Using checklists for validation

Software Requirements Validation

- ❑ Commenting
- ❑ Inspections
- ❑ Walk-throughs
- ❑ Perspective-based reading
- ❑ Validation through prototypes
- ❑ Using checklists for validation

Software Requirements Prioritization

- ❑ Why Prioritize?
- ❑ Techniques to Prioritize

Software Requirements Prioritization

- ❑ Why Prioritize?
- ❑ Techniques to Prioritize

- ❑ Multiple requirements
- ❑ Budgetary constraints
- ❑ Tight deadlines
- ❑ Multiple conflicting stakeholders
- ❑ Order of Development
- ❑ Dependency Matrix

Software Requirements Prioritization

- ❑ Why Prioritize?
- ❑ Techniques to Prioritize

- ❑ Multiple requirements
- ❑ Budgetary constraints
- ❑ Tight deadlines
- ❑ Multiple conflicting stakeholders
- ❑ Order of Development
- ❑ Dependency Matrix

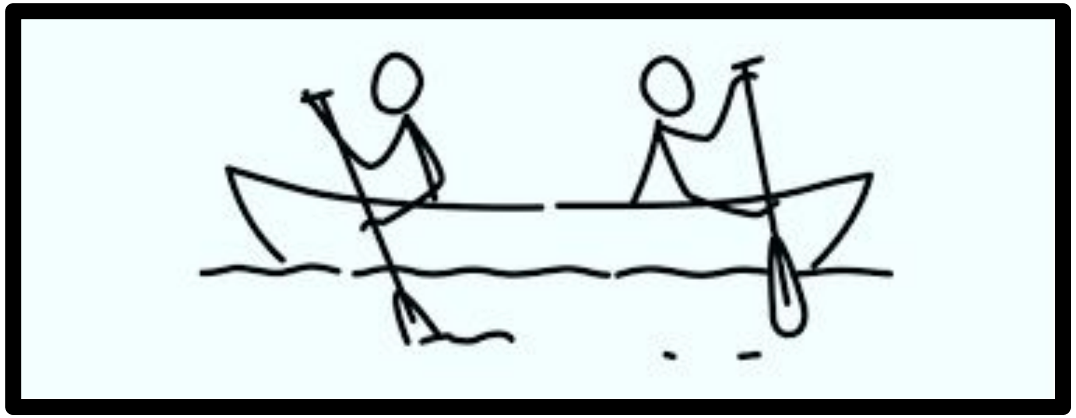
Software Requirements Prioritization

- ❑ Why Prioritize?
- ❑ Techniques to Prioritize

- ❑ Multiple requirements
- ❑ Budgetary constraints
- ❑ Tight deadlines
- ❑ Multiple conflicting stakeholders
- ❑ Order of Development
- ❑ Dependency Matrix

Software Requirements Prioritization

- ❑ Why Prioritize?
- ❑ Techniques to Prioritize

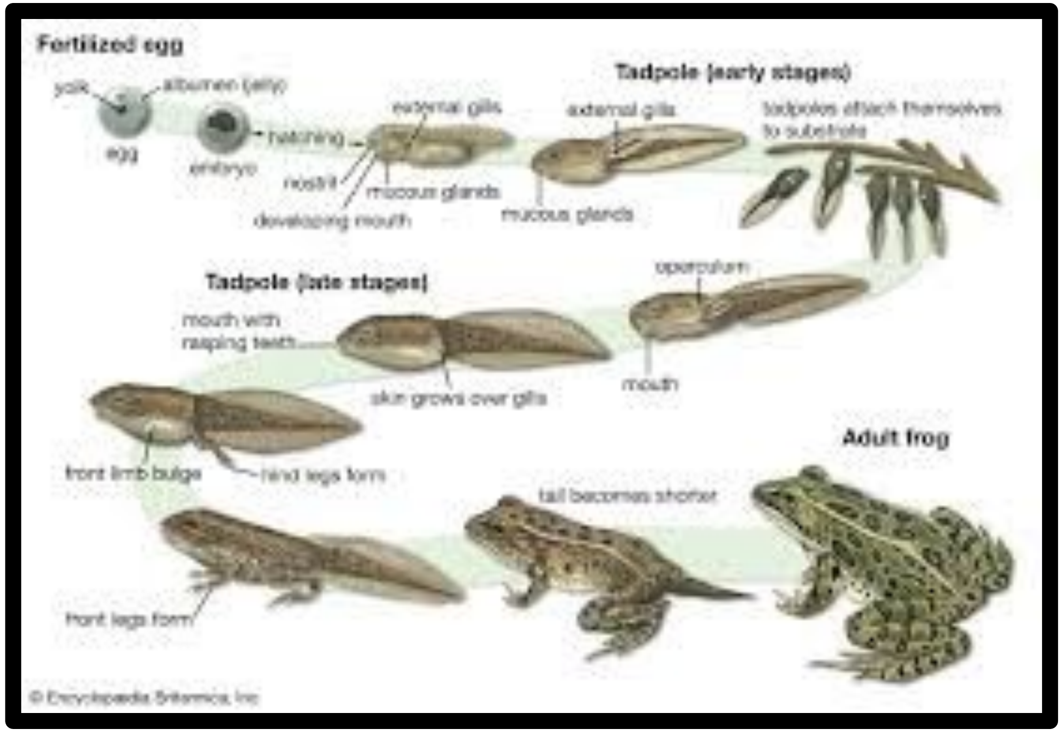


- ❑ Multiple requirements
- ❑ Budgetary constraints
- ❑ Tight deadlines
- ❑ Multiple conflicting stakeholders
- ❑ Order of Development
- ❑ Dependency Matrix

Software Requirements Prioritization

- □ Why Prioritize?
- □ Techniques to Prioritize

- □ Multiple requirements
- □ Budgetary constraints
- □ Tight deadlines
- □ Multiple conflicting stakeholders
- □ Order of Development
- □ Dependency Matrix



© Encyclopaedia Britannica, Inc

Software Requirements Prioritization

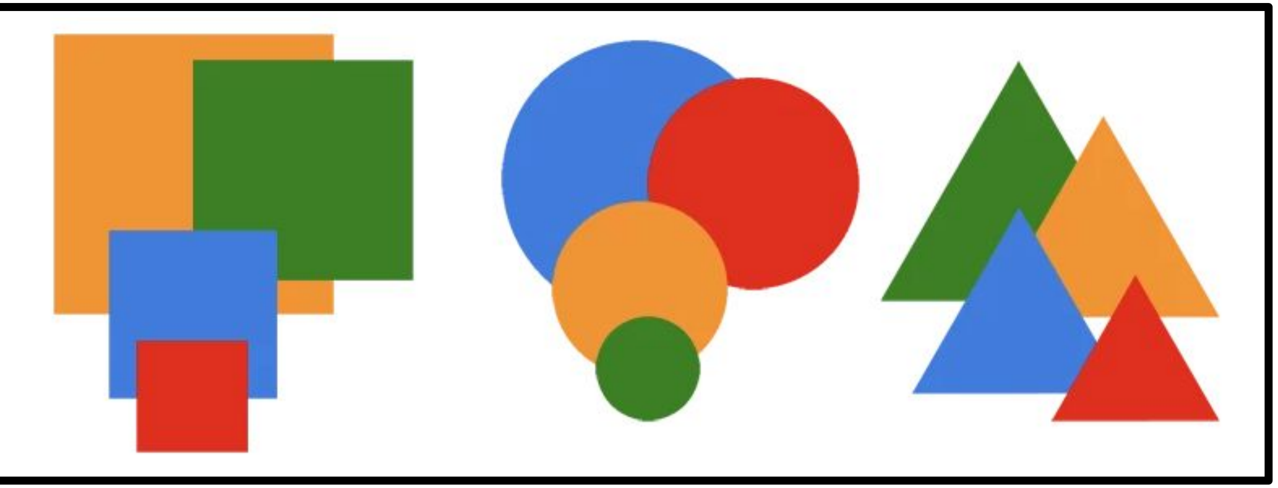
- ❑ Why Prioritize?
- ❑ Techniques to Prioritize

- ❑ Simple Ranking
- ❑ Grouping
- ❑ MoSCoW Technique
- ❑ Bubble Sort Technique
- ❑ Constrained Utility Method
- ❑ Critical Questioning

Software Requirements Prioritization

- ❑ Why Prioritize?
- ❑ Techniques to Prioritize

- ❑ Simple Ranking
- ❑ Grouping
- ❑ MoSCoW Technique
- ❑ Bubble Sort Technique
- ❑ Constrained Utility Method
- ❑ Critical Questioning



Software Requirements Prioritization

- ❑ Why Prioritize?
- ❑ Techniques to Prioritize

MUST HAVE	SHOULD HAVE	COULD HAVE	WON'T HAVE
❑ Non-Negotiables ❑ MVP ❑ Can't deliver on target without this ❑ Not legal without it ❑ Unsafe without it ❑ Not viable without it	❑ Important not vital ❑ May be painful without, but viable for the project ❑ May need a workaround	❑ Desirable but not as much as previous ❑ If time and money permits, then yes	❑ Won't have it this time ❑ Nice to have but no impact if absent ❑ Out of budget

- ❑ Simple Ranking
- ❑ Grouping
- ❑ MoSCoW Technique
- ❑ Bubble Sort Technique
- ❑ Constrained Utility Method
- ❑ Critical Questioning

Software Requirements Prioritization

MUST HAVE

- □ Non-Negotiables
- □ MVP
- □ Can’t deliver on target without this
- □ Not legal without it
- □ Unsafe without it
- □ Not viable without it

https://kecg.co/task-prioritisation-hack-using-moscow-method/

Software Requirements Prioritization

MUST HAVE

- ❑ Non-Negotiables
- ❑ MVP
- ❑ Can’t deliver on target without this
- ❑ Not legal without it
- ❑ Unsafe without it
- ❑ Not viable without it

SHOULD HAVE

- ❑ Important not vital
- ❑ May be painful without, but viable for the project
- ❑ May need a workaround

https://kecg.co/task-prioritisation-hack-using-moscow-method/

Software Requirements Prioritization

MUST HAVE

- □ Non-Negotiables
- □ MVP
- □ Can’t deliver on target without this
- □ Not legal without it
- □ Unsafe without it
- □ Not viable without it

SHOULD HAVE

- □ Important not vital
- □ May be painful without, but viable for the project
- □ May need a workaround

COULD HAVE

- □ Desirable but not as much as previous
- □ If time and money permits, then yes

https://kecg.co/task-prioritisation-hack-using-moscow-method/

Software Requirements Prioritization

MUST HAVE

- □ Non-Negotiables
- □ MVP
- □ Can’t deliver on target without this
- □ Not legal without it
- □ Unsafe without it
- □ Not viable without it

SHOULD HAVE

- □ Important not vital
- □ May be painful without, but viable for the project
- □ May need a workaround

COULD HAVE

- □ Desirable but not as much as previous
- □ If time and money permits, then yes

WON’T HAVE

- □ Won't have it this time
- □ Nice to have but no impact if absent
- □ Out of budget

https://kecg.co/task-prioritisation-hack-using-moscow-method/

Software Requirements Prioritization

- □ Why Prioritize?
- □ Techniques to Prioritize

- □ Simple Ranking
- □ Grouping
- □ MoSCoW Technique
- □ Bubble Sort Technique
- □ Constrained Utility Method
- □ Critical Questioning

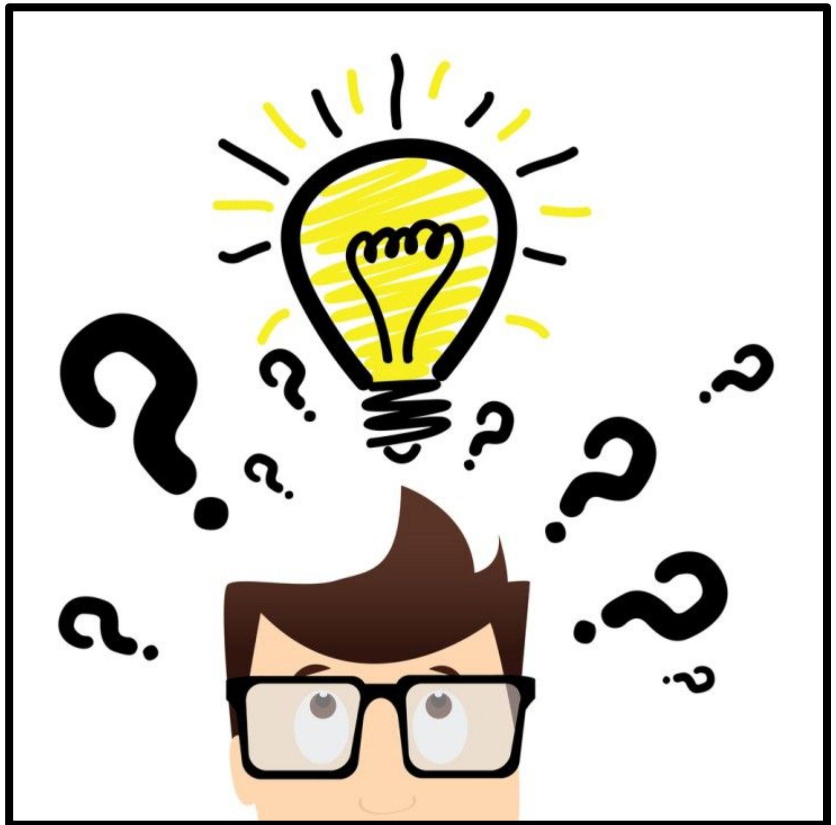
Software Requirements Prioritization

- ❑ Why Prioritize?
- ❑ Techniques to Prioritize

- ❑ Simple Ranking
- ❑ Grouping
- ❑ MoSCoW Technique
- ❑ Bubble Sort Technique
- ❑ Constrained Utility Method
- ❑ Critical Questioning

Software Requirements Prioritization

- ❑ Why Prioritize?
- ❑ Techniques to Prioritize



- ❑ Simple Ranking
- ❑ Grouping
- ❑ MoSCoW Technique
- ❑ Bubble Sort Technique
- ❑ Constrained Utility Method
- ❑ Critical Questioning

Algorithms–requirements/validation

- Problem definition
- Development of a model
- Specification of the algorithm
- Checking the correctness of the algorithm
- Analysis of algorithm

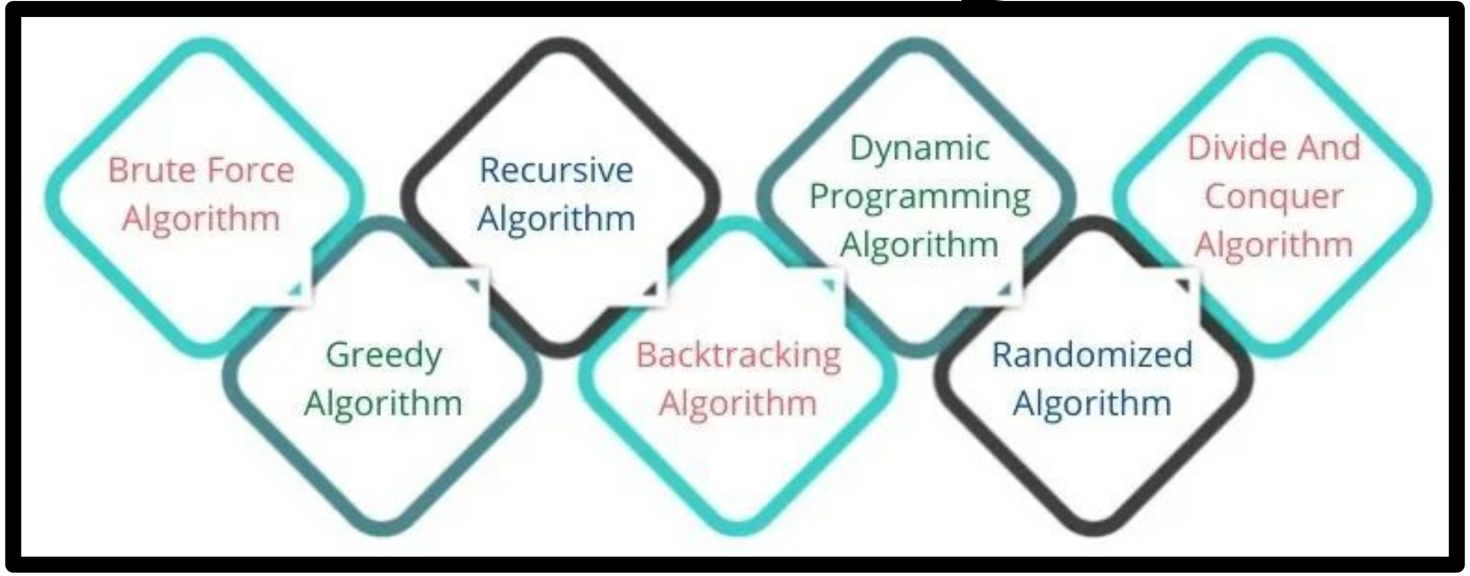
- Binary Search
- Selection, Bubble, Insertion, Quicksort & Merge
- Huffman Coding
- Breadth First Search
- Depth First Search
- Gradient Descent
- Kruskal & Dijkstra's Algorithm
- Diffie-Hellman Key Exchange

Algorithms–requirements/validation

- ❑ Problem definition
- ❑ Development of a model
- ❑ Specification of the algorithm
- ❑ Checking the correctness of the algorithm
- ❑ Analysis of algorithm

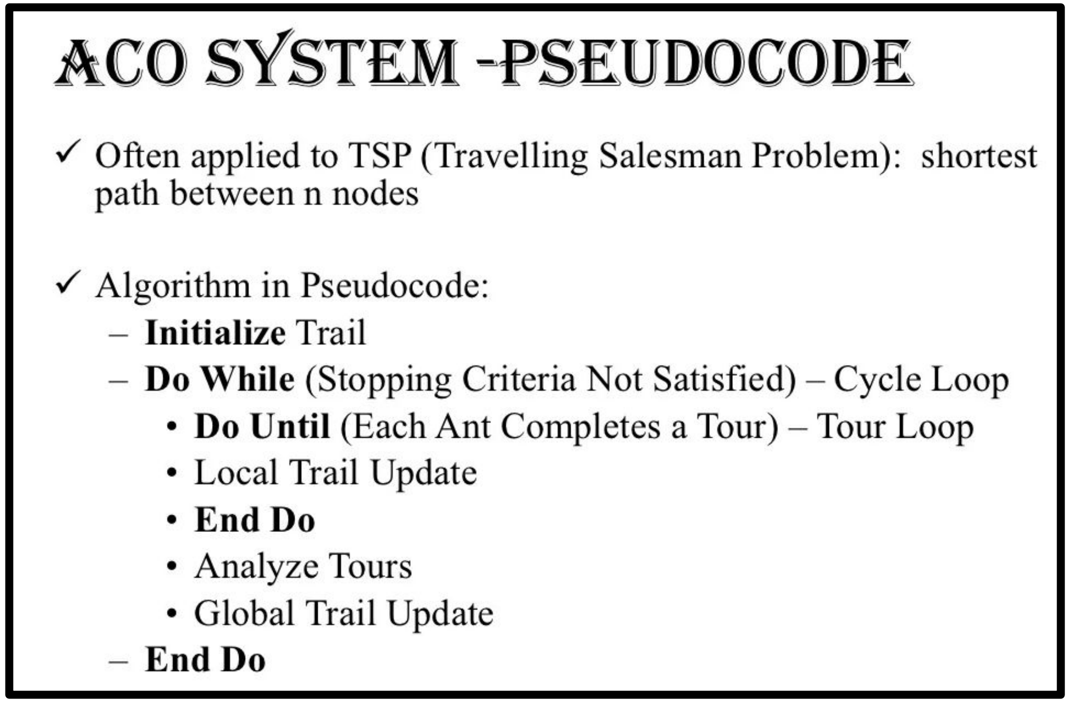
Algorithms–requirements/validation

- □ Problem definition
- □ Development of a model
- □ Specification of the algorithm
- □ Checking the correctness of the algorithm
- □ Analysis of algorithm



Algorithms–requirements/validation

- □ **Problem definition**
- □ **Development of a model**
- □ **Specification of the algorithm**
- □ **Checking the correctness of the algorithm**
- □ **Analysis of algorithm**

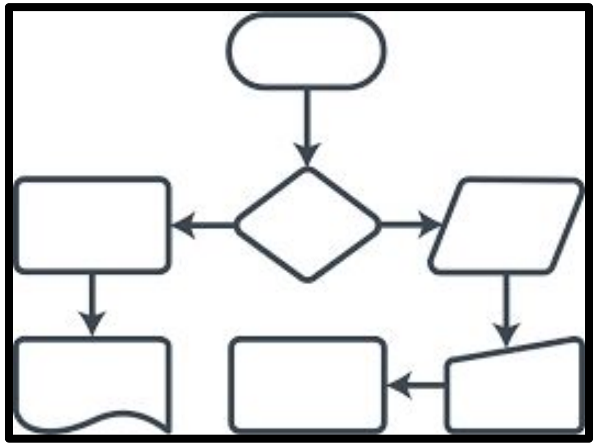


ACO SYSTEM -PSEUDOCODE

✓ Often applied to TSP (Travelling Salesman Problem): shortest path between n nodes

✓ Algorithm in Pseudocode:

```
– Initialize Trail
– Do While (Stopping Criteria Not Satisfied) – Cycle Loop
    • Do Until (Each Ant Completes a Tour) – Tour Loop
    • Local Trail Update
    • End Do
    • Analyze Tours
    • Global Trail Update
– End Do
```



Algorithms–requirements/validation

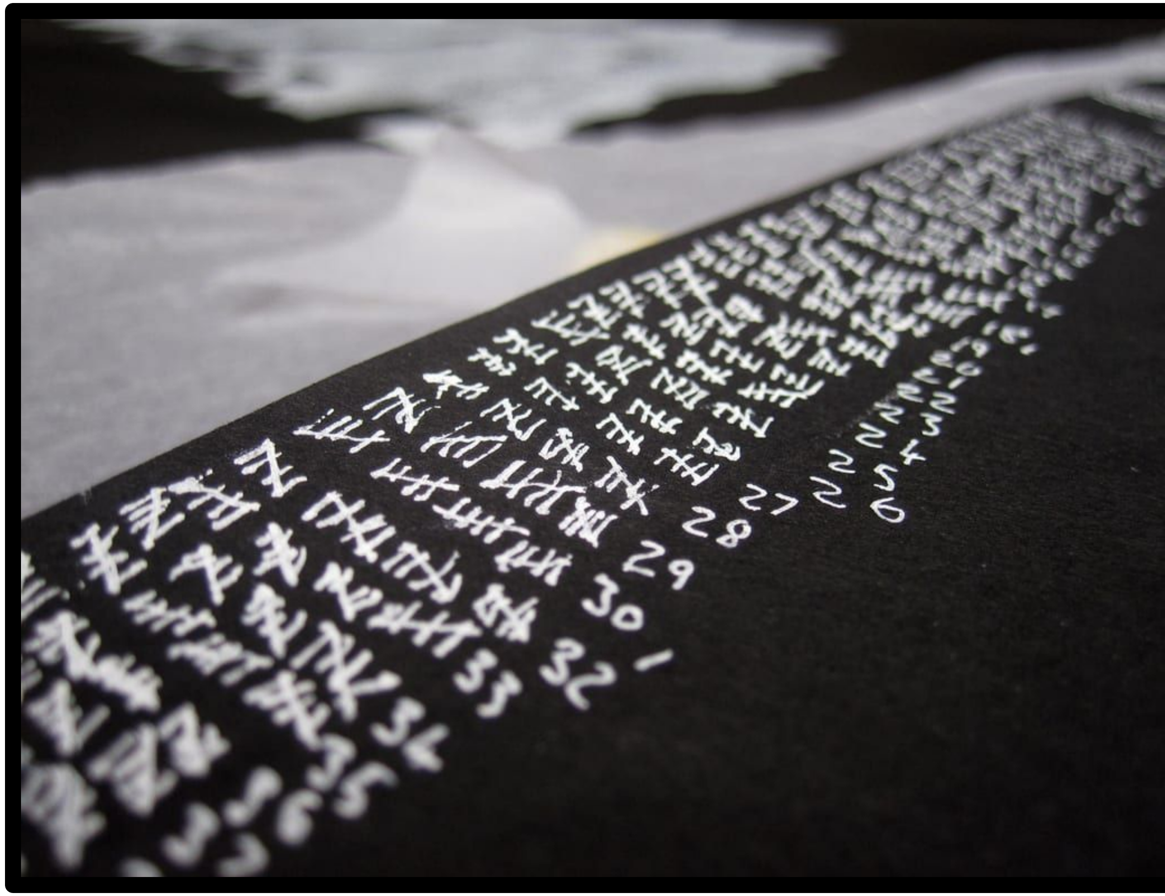
❑ Problem definition

❑ Development of a model

❑ Specification of the algorithm

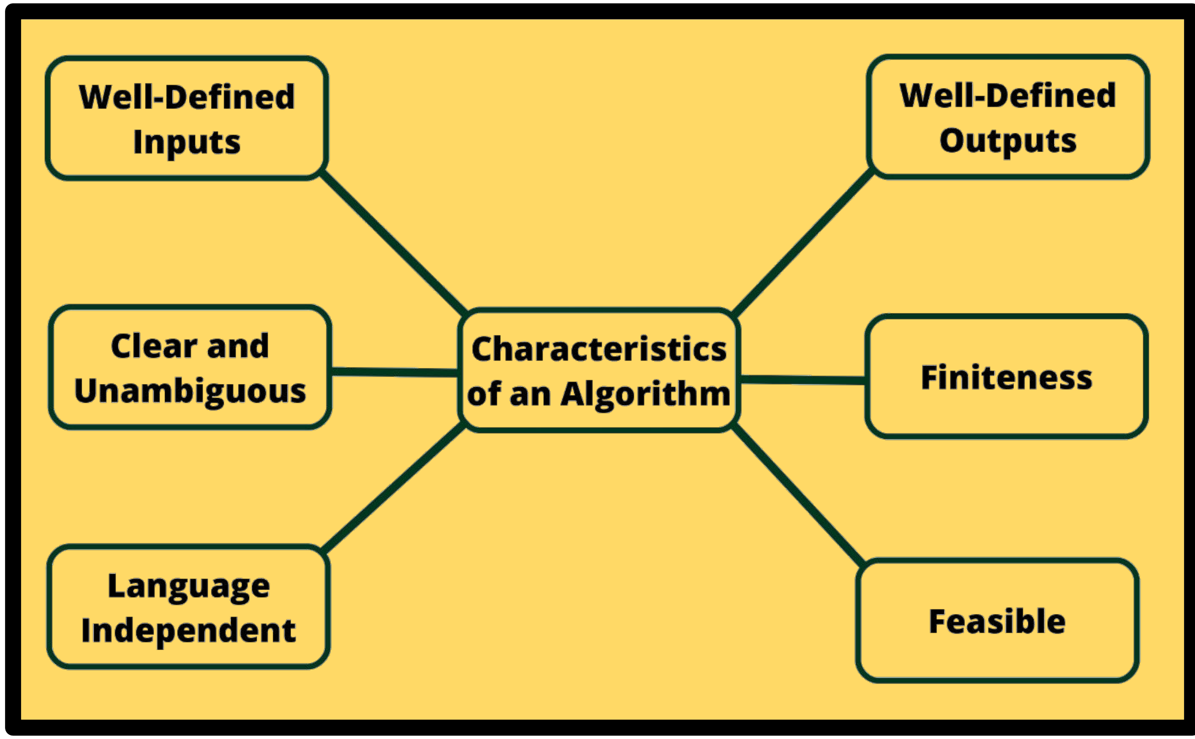
❑ Checking the correctness of the algorithm

❑ Analysis of algorithm



Algorithms–requirements/validation

- ❑ Problem definition
- ❑ Development of a model
- ❑ Specification of the algorithm
- ❑ Checking the correctness of the algorithm
- ❑ Analysis of algorithm


Well-Defined Inputs
Well-Defined Outputs
Clear and Unambiguous
Characteristics of an Algorithm
Finiteness
Language Independent
Feasible