University of Science and Technology
Faculty of Computer Science and Information Technology

Artificial Intelligence (AI)

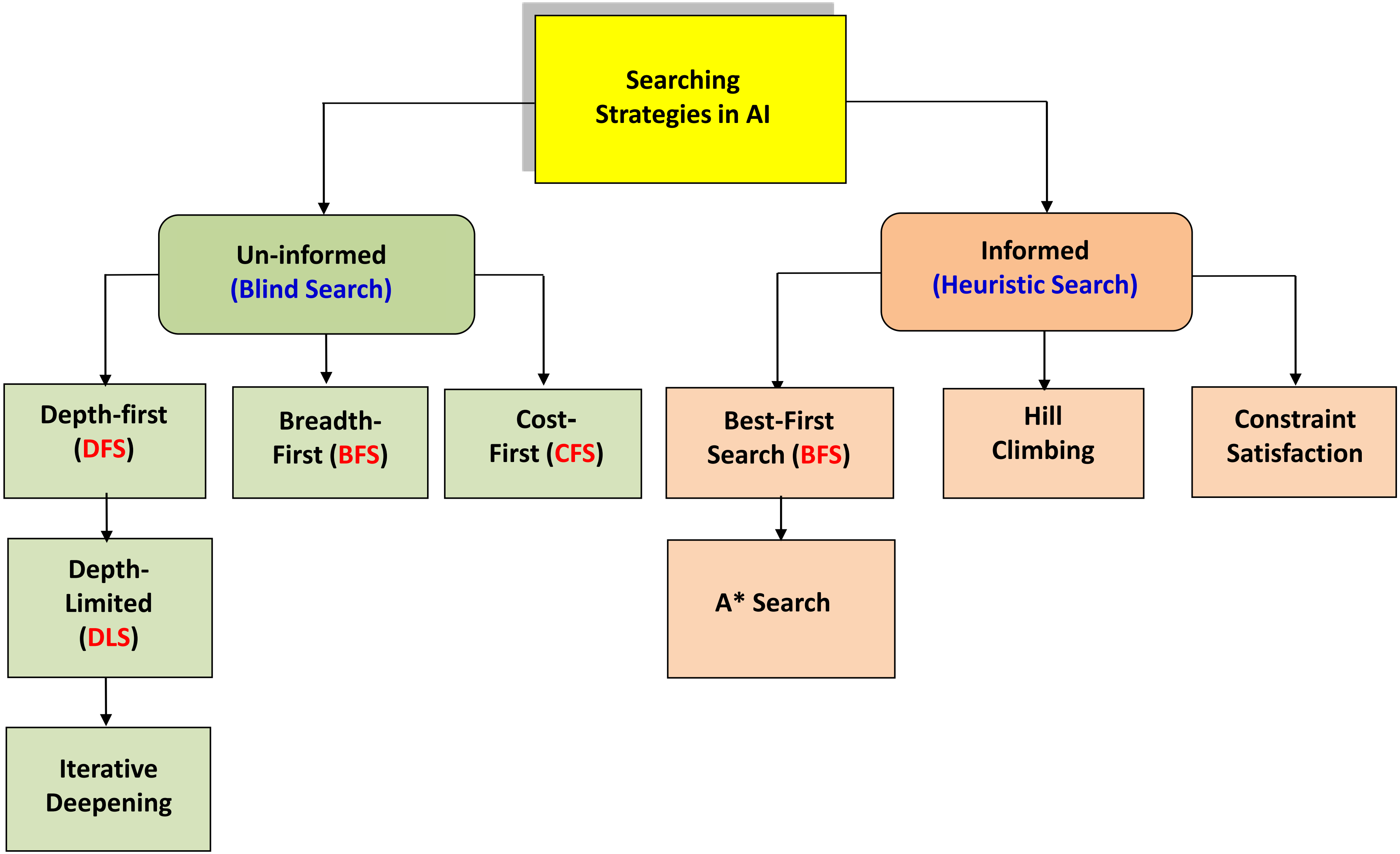
4th Year B.Sc : Information Technology

Academic Year : 2017-2018

Instructor : Diaa Eldin Mustafa Ahmed

Problem Solving (Searching Techniques)-2/2

Informed (Heuristic Search)



Informed (Heuristic Search)

- For complex problems, the traditional algorithms, presented above, are unable to find the solution within some practical time and space limits.
- Consequently, many special techniques are developed, using heuristic functions.
  - Blind search is not always possible, because they require too much time or Space (memory).
  - Heuristics are **rules of thumb**; they do not guarantee for a solution to a problem.
  - Heuristic Search is a weak techniques but can be effective if applied correctly; they require domain specific information.

Heuristic Search compared with other search

The Heuristic search is compared with Brute force or Blind search techniques

Compare Algorithms

Brute force / Blind search	Heuristic search
◇ Only have knowledge about already explored nodes	◇ Estimates "distance" to goal state
◇ No knowledge about how far a node is from goal state	◇ Guides search process toward goal state
	◇ Prefer states (nodes) that lead close to and not away from goal state

Informed Search

- A search strategy which searches the most promising branches of the state-space first can:
  - Find a solution more quickly,
  - Find solutions even when there is limited time available,
  - Often find a better solution, since more profitable parts of the state-space can be examined, while ignoring the unprofitable parts.
- A search strategy which is better than another at identifying the most promising branches of a search-space is said to be more informed.

Best-first search strategy

- Combing depth-first search and breadth-first search.
- Selecting the node with the best estimated cost among all nodes.
- This method has a global view.

e.g. 8-puzzle problem

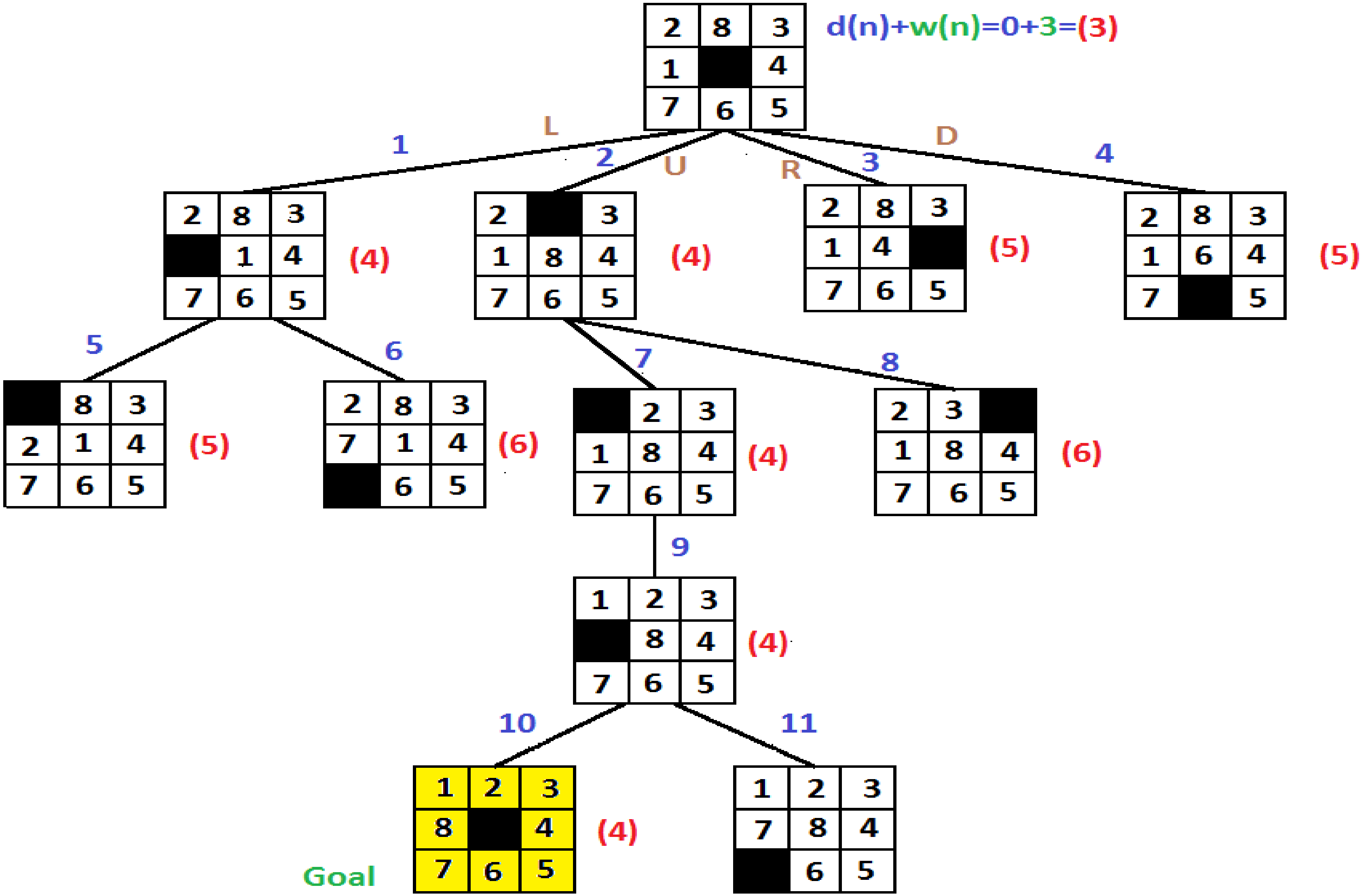
evaluation function $f(n) = d(n) + w(n)$

where :

$d(n)$: is the depth of node n.

$w(n)$: is # of misplaced tiles in node n.

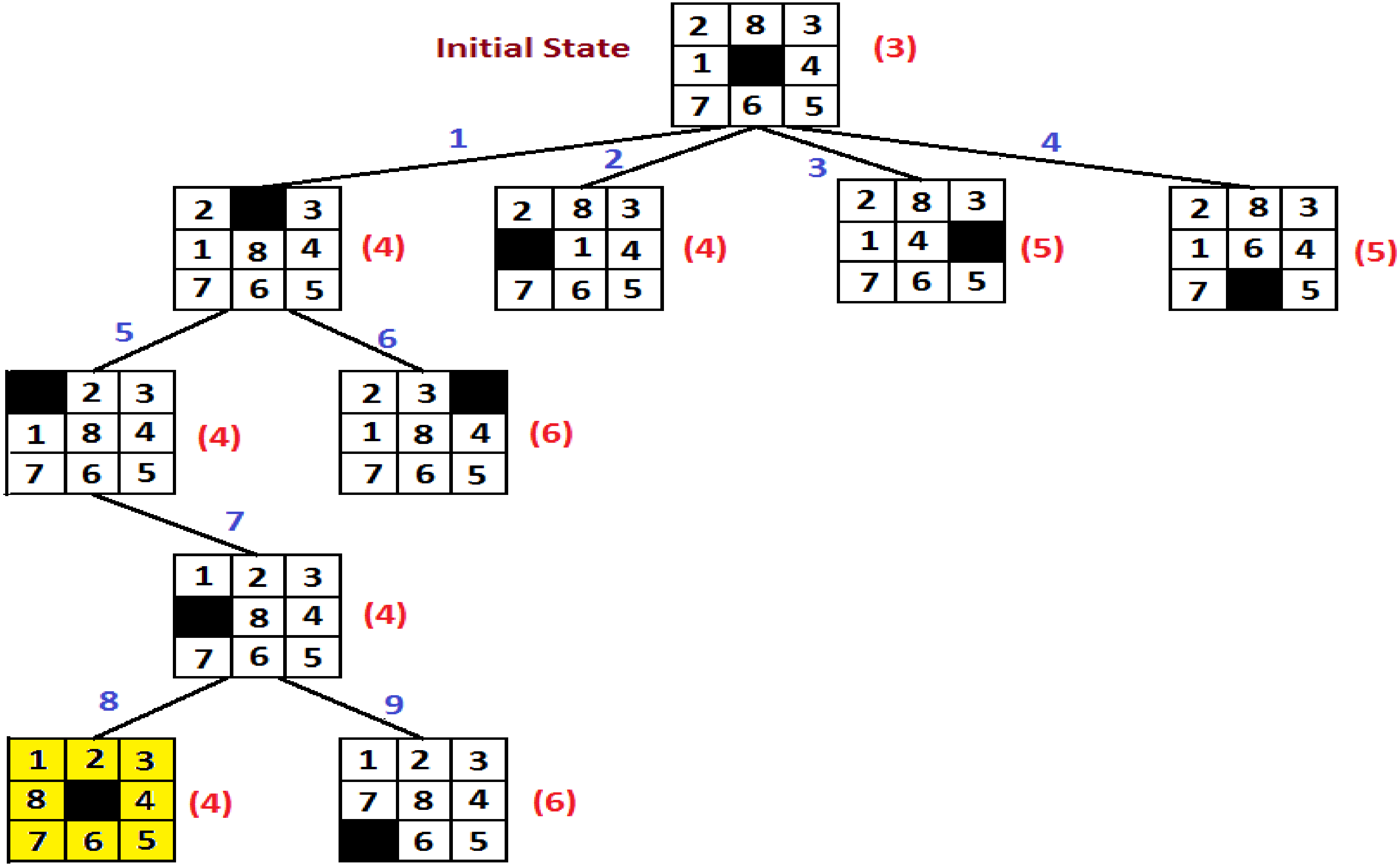
Best-first search strategy



Hill climbing

- Hill climbing is a mathematical optimization technique which belongs to the family of local search.
- It is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by incrementally changing a single element of the solution.
- If the change produces a better solution, an incremental change is made to the new solution, repeating until no further improvements can be found.
- A variant of depth-first search.
- The method selects the locally optimal node to expand.

Hill climbing



Goal Node

An 8-puzzle problem solved by a hill climbing method.

Constraint Satisfaction Problems (CSPs)

- Constraints arise in most areas of human endeavor.
- Constraints are a natural medium for people to express problems in many fields.
- Many real problems in AI can be modeled as Constraint Satisfaction Problems (CSPs) and are solved through search.
- Examples of constraints :
  - The sum of three angles of a triangle is 180 degrees,
  - The sum of the currents flowing into a node must equal zero.
- Constraint is a logical relation among variables.
  - The constraints relate objects without precisely specifying their positions ; moving any one, the relation is still maintained.
  - **Example** : "circle is inside the square".

Constraint Satisfaction Problems (CSPs)

- Constraint satisfaction
  - The Constraint satisfaction is a process of finding a solution to a set of constraints.
  - The constraints articulate allowed values for variables.
  - Finding solution is evaluation of these variables that satisfies all constraints.
- The CSPs are all around us while managing work, home life, budgeting expenses and so on;
  - where we do not get success in finding solution, there we run into problems.
  - we need to find solution to such problems satisfying all constraints.
  - the Constraint Satisfaction problems (CSPs) are solved through search.

Constraint Satisfaction Problems (CSPs)

- Many problems in AI can be considered as problems of constraint satisfaction, in which the goal state satisfies a given set of constraint.
- Constraint satisfaction problems can be solved by using any of the search strategies.
- The general form of the constraint satisfaction procedure is as follows:

Constraint Satisfaction Problems (CSPs)

- Until a complete solution is found or until all paths have led to lead ends, do
  1. select an unexpanded node of the search graph.
  2. Apply the constraint inference rules to the selected node to generate all possible new constraints.
  3. If the set of constraints contains a contradiction, then report that this path is a dead end.
  4. If the set of constraints describes a complete solution then report success.
  5. If neither a constraint nor a complete solution has been found then apply the rules to generate new partial solutions.
  6. Insert these partial solutions into the search graph.

(CSPs)- Cryptarithmetic Example

Example: consider the following cryptarithmetic problem.

```
      S E N D
  +   M O R E
  --------------
    M O N E Y
  --------------
```

- The aim is to assign each letter a unique integer in the range 0..9 so that the sum is correct.
- Define the problem as a constraint satisfaction problem (CSP) in terms of :
  - Variables **V**,
  - Domains **D** and
  - Constraints **C**.
- Show how an analysis of the problem can be used to reduce the domains of the variables and create additional constraints.

(CSPs)- Cryptarithmetic Example

CONSTRAINTS:-

1. No two digit can be assigned to same letter.
2. Only single digit number can be assign to a letter.
3. No two letters can be assigned same digit.
4. Assumption can be made at various levels such that they do not contradict each other.
5. The problem can be decomposed into secured constraints. A constraint satisfaction approach may be used.
6. Any of search techniques may be used.
7. Backtracking may be performed as applicable us applied search techniques.
8. Rule of arithmetic may be followed.

(CSPs)- Cryptarithmetic Example

Initial Solution

Initially the problem can be stated as follows :

$V = \{S, E, N, D, M, O, R, Y\}$

$D_S = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$D_E = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$D_N = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$D_D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$D_M = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$D_O = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$D_R = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$D_Y = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

(CSPs)- Cryptarithmetic Example

Next we can reason that the largest values for **S** and **M** would be 8 and 9. If there was a carry digit (we will represent these as C1, C2, C3 and C4) this means that **S** + **M** + C3 < 19. This infers that **M** = **1**. This gives us this revised set of domains.

D_S	=	{2, 3, 4, 5, 6, 7, 8, 9}
D_E	=	{0, 2, 3, 4, 5, 6, 7, 8, 9}
D_N	=	{0, 2, 3, 4, 5, 6, 7, 8, 9}
D_D	=	{0, 2, 3, 4, 5, 6, 7, 8, 9}
D_M	=	{1}
D_O	=	{0, 2, 3, 4, 5, 6, 7, 8, 9}
D_R	=	{0, 2, 3, 4, 5, 6, 7, 8, 9}
D_Y	=	{0, 2, 3, 4, 5, 6, 7, 8, 9}

```
C4 C3 C2 C1
    S  E  N  D
+   M  O  R  E
--------------------
 M  O  N  E  Y
--------------------
```

(CSPs)- Cryptarithmetic Example

And our problem now looks like this

$$\begin{array}{lr} & S\ E\ N\ D \\ + & 1\ O\ R\ E \\ \hline = & 1\ O\ N\ E\ Y \end{array}$$

We can now turn our attention to the thousands column. C3 could be zero or one (representing a carry or not). That is,

$S + M + C3 = 10 + O$

We'll consider both cases of carry below

(CSPs)- Cryptarithmetic Example

	$C_3 = 1$	$C_3 = 0$
	S + 1 + 1 = 10 + O	S + 1 + 0 = 10 + O
Simplify	S + 2 = 10 + O	
Subtract 2 (or1) from both sides	S = 8 + O	S = 9 + O
Conclusions	If O is 0, S = 8 If O is 1, S = 9	If O is 0, S = 9

Therefore, S must equal 8 or 9.
Let's try to prove that S = 8.

```
     8END
+    10RE
= 10NEY
```

(CSPs)- Cryptarithmetic Example

From the above, we showed that O=0 if S=8. If this is the correct answer we can see that a carry is required in the hundreds column (C2) as E + 0 = N is not valid as E and N would take the same value, which is not allowed.

Or, to show it is invalid another way

$$E + 0 + C3(0) = 10 + N$$

Simplify $E = N + 10$

Which is invalid, as N would have to be zero, which is already the case with O

So, assuming there is a carry, we have

$$E + 0 + 1 = 10 + N$$

Simplifying gives $E + 1 = 10 + N$

Subtract 1 from both sides $E = 9 + N$

(CSPs)- Cryptarithmetic Example

The only possible value for N is zero (else E would be outside its domain) but O is already zero, so this answer is invalid.

Therefore, S cannot equal 8, so it S = 9, giving

		9	E	N	D
+		**1**	O	R	E
=	**1**	O	N	E	Y

Now, consider the thousands column. We have either

$$9 + 1\ C_3(0) = O$$

or

$$9 + 1 + C_3(1) = O$$

(CSPs)- Cryptarithmetic Example

If the carry, C3, is zero then M = 1 and O = 0.
If the carry, C3, is one then M = 1 and O = 1.
As O $\neq$ 1 (as M=1), then O = 0, giving

$$
\begin{array}{rl}
 & 9END \\
+ & \underline{10RE} \\
= & \underline{10NEY}
\end{array}
$$

Turning our attention to the tens column, we have N + R = E. We can see that we must also have a carry as the hundreds column (E + 0 = N) needs to have a carry from the previous column else E = N, which is invalid.
We can also state that N = E + 1 by definition of the hundreds column and the carry, C2.
From the above we can derive N + R + C1 = 10 + E. In the table below we will consider at this formula

(CSPs)- Cryptarithmetic Example

	$C_1 = 1$	$C_1 = 0$
	$N + R + 1 = 10 + E$	$N + R + 0 = 10 + E$
Subtract 1	$N + R = 9 + E$	
Substitute $N = E + 1$	$E + 1 + R = 9 + E$	$E + 1 + R = 10 + E$
Subtract E	$1 + R = 9$	$1 + R = 10$
Subtract 1	$R = 8$	$R = 9$
Possible	Yes	No, as $S = 9$

Therefore, we can set R to 8, giving

```
   9END
+  108E
= 10NEY
```

(CSPs)- Cryptarithmetic Example

We could continue with this algebraic analysis and solve the puzzle completely but if we assume that we could not go any further then we have the following to give our CSP search.

V = {S, E, N, D, M, O, R, Y}

D_S = { 9}

D_E = {2, 3, 4, 5, 6}

D_N = {2, 3, 4, 5, 6, 7}

D_D = {2, 3, 4, 5, 6, 7}

D_M = {1}

D_O = {0}

D_R = {8}

D_Y = {2, 3, 4, 5, 6, 7}

$C_1 = N = E + 1$

(note, how this constraint also further limits the domain of E).

(CSPs)- Cryptarithmetic Example

In Summary

V = {**S, E, N, D, M, O, R, Y**}

D_S = { 9}

D_E = {2, 3, 4, 5, 6}

D_N = {2, 3, 4, 5, 6, 7}

D_D = {2, 3, 4, 5, 6, 7}

D_M = {1}

D_O = {0}

D_R = {8}

D_Y = {2, 3, 4, 5, 6, 7}

C_1 = If the same letter occurs more than once, it must be assigned the same value

C_2 = No two different letters may be assigned the same digit

C_3 = N = E + 1

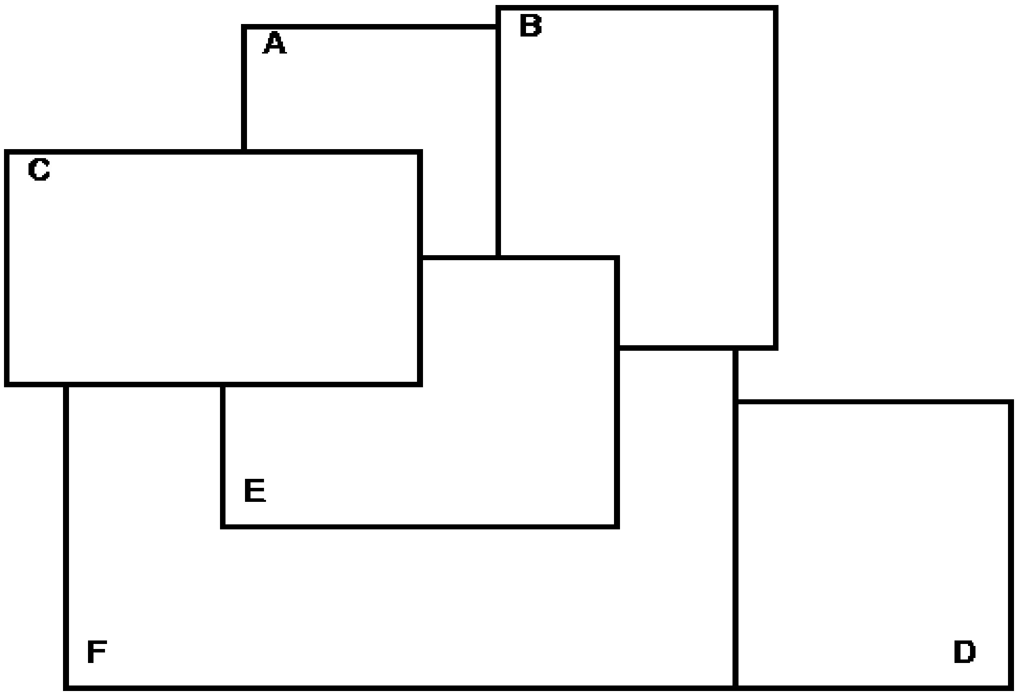
(CSPs)- Cryptarithmetic Example

- .

(CSPs)- Cryptarithmetic Example

- .

- Using the most-constrained-variable CSP heuristic colour the following map using the colours Blue, Red and Green. Show you reasoning at each step of the algorithm.



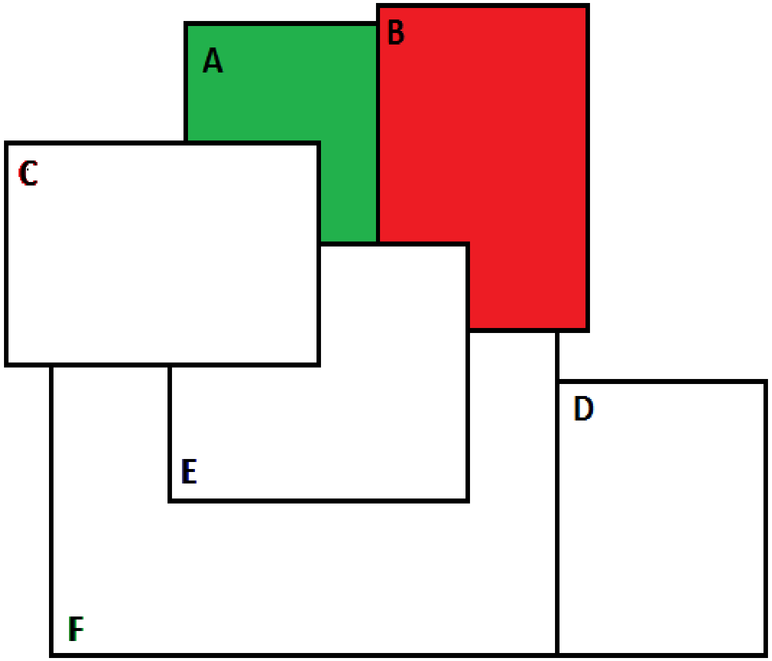
Now, let us consider this in the context of a CSP problem.

We start with the following variables and their domains

A	=	{**Red**, **Blue**, **Green**}
B	=	{**Red**, **Blue**, **Green**}
C	=	{**Red**, **Blue**, **Green**}
D	=	{**Red**, **Blue**, **Green**}
E	=	{**Red**, **Blue**, **Green**}
F	=	{**Red**, **Blue**, **Green**}

After making the first two assignments, and using forward tracking to reduce the domains of the other variables, we have the following assignments (variables marked * have been instantiated)

A*	=	{**Green**}
B*	=	{**Red**}
C	=	{**Red**, **Blue**}
D	=	{**Red**, **Blue**, **Green**}
E	=	{**Blue**}
F	=	{**Blue**, **Green**}



To decide which variable to instantiate we choose the variable with the smallest domain. In this case **E**. After instantiating **E** the assignments look like this (again forward tracking has been used to reduce the domain of the remaining variables)

A* = {**Green**}

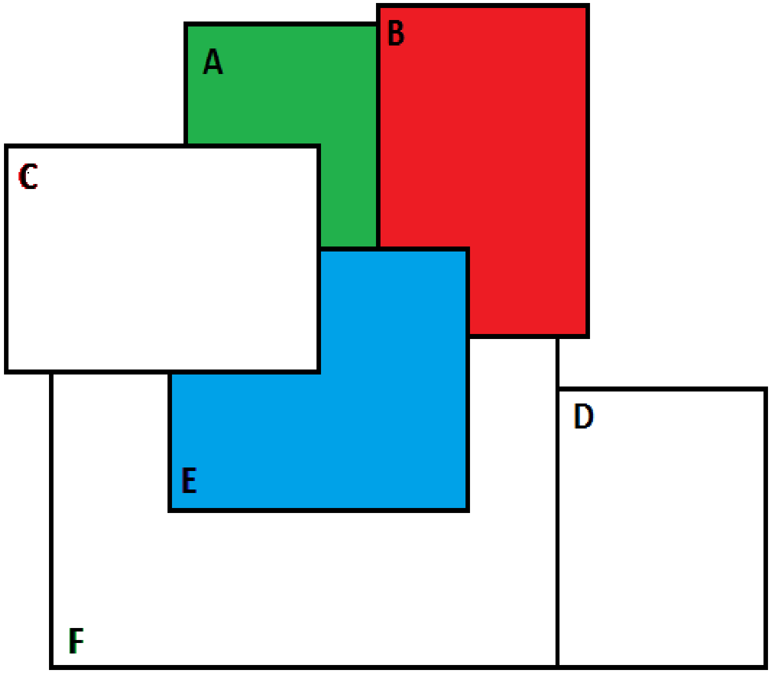
B* = {**Red**}

C = {**Red**}

D = {**Red**, **Blue**, **Green**}

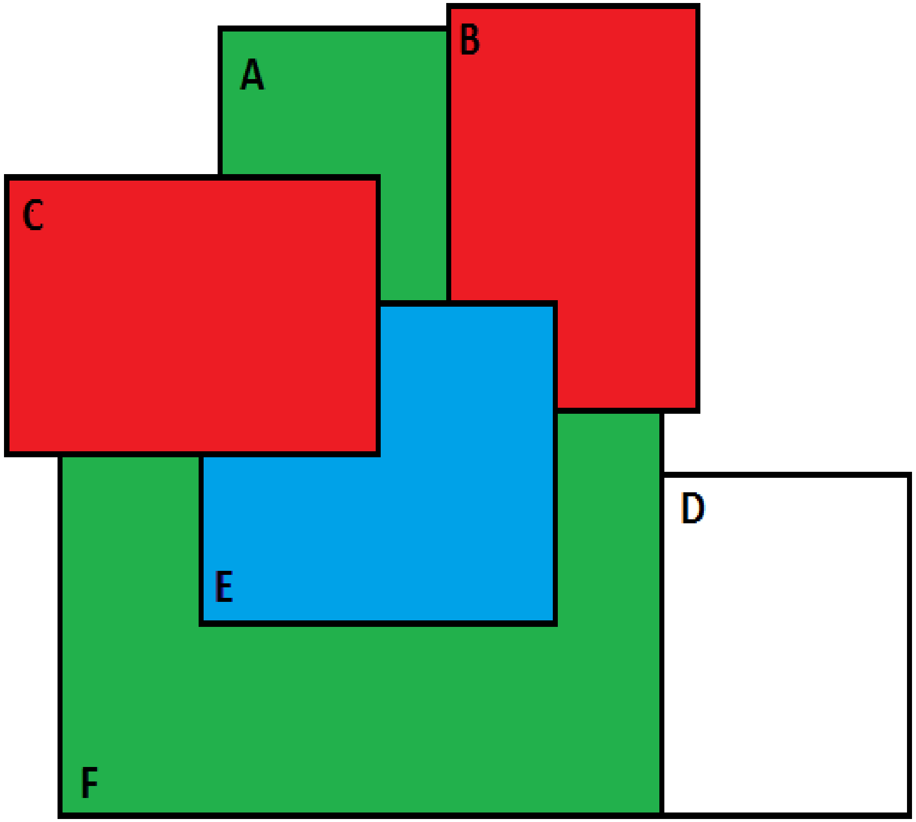
E* = {**Blue**}

F = {**Green**}



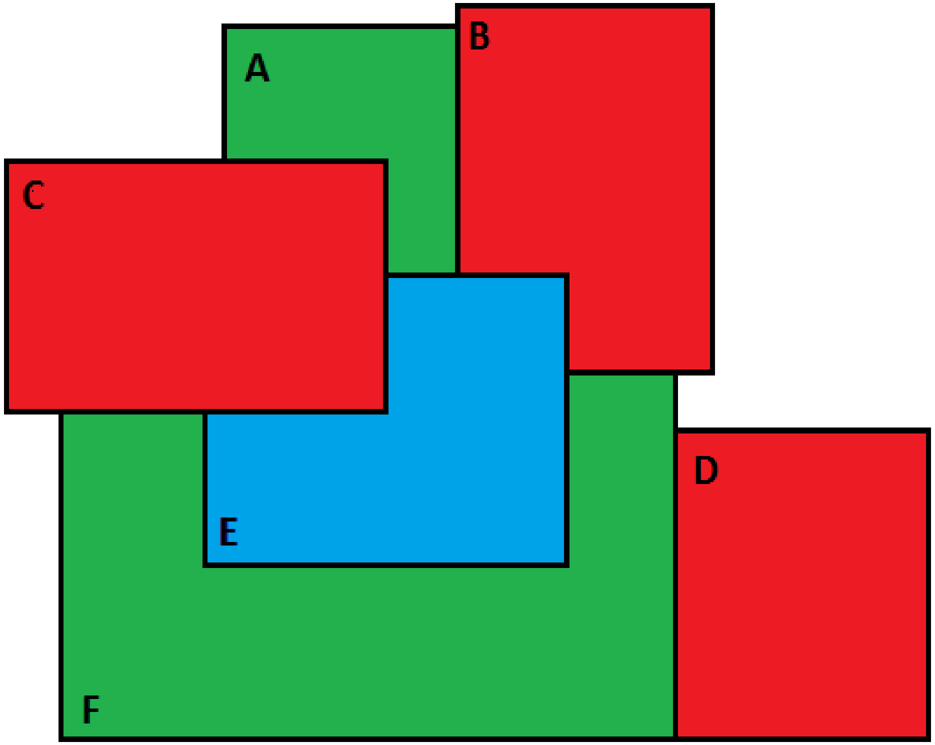
We can now choose C (or F) and then F (or C). This results in the following

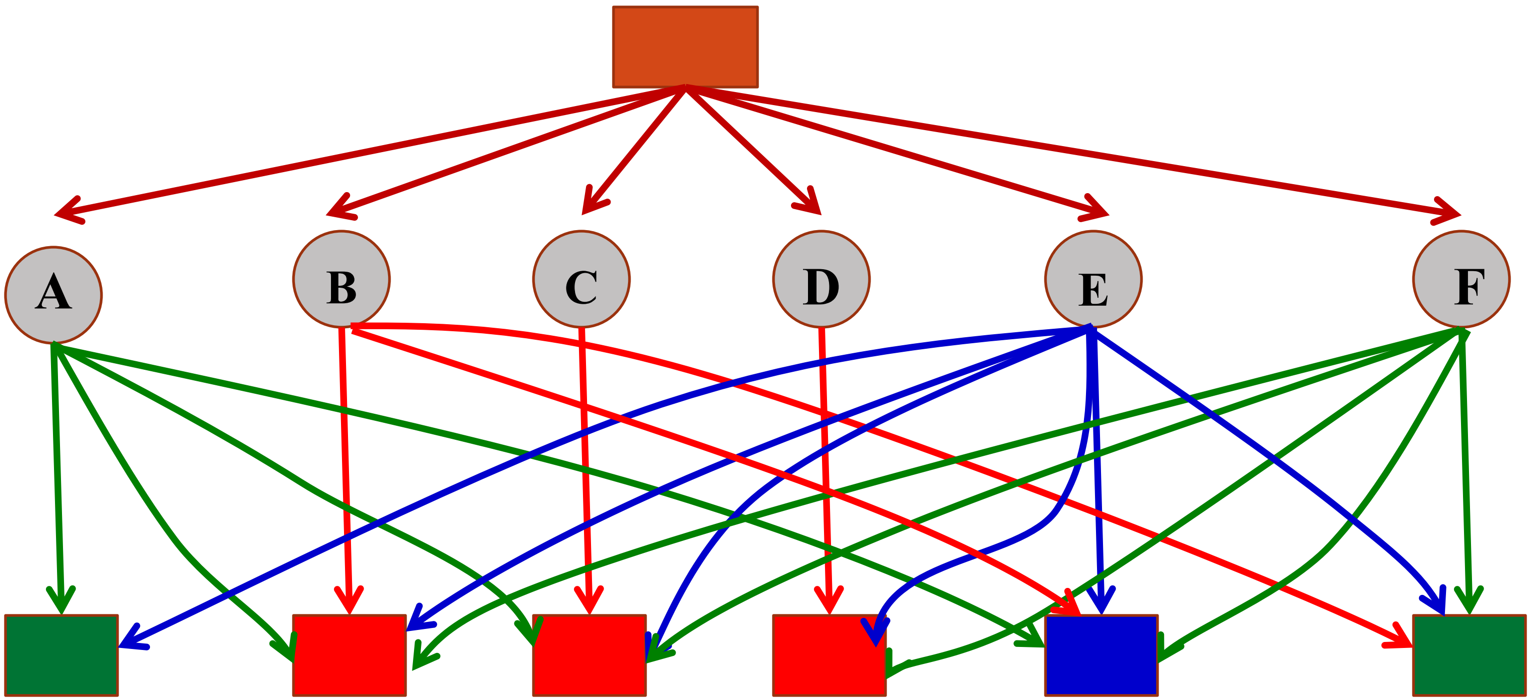
A*	**=**	**{Green}**
B*	**=**	**{Red}**
C*	**=**	**{Red}**
D	**=**	**{Red, Blue}**
E*	**=**	**{Blue}**
F*	**=**	**{Green}**



Lastly, we can assign D Red or Blue. We choose (arbitrarily) Red, leading to the final solution

A*	**=**	**{Green}**
B*	**=**	**{Red}**
C*	**=**	**{Red}**
D*	**=**	**{Red}**
E*	**=**	**{Blue}**
F*	**=**	**{Green}**




A
B
C
D
E
F

Step -1

$\boxed{\begin{array}{c} M=1 \\ O=0 \end{array}}$

S=8 or 9

Let E=2

E(2) +O(0)+C2(1 or 0) = N

If C2=1 | If C2=0

E(2)+O(0)+C2(1)=N(3)

E(2)+O(0)+C2(0)=N(2)/x

Contradiction (Rule 3)

N(3)+R+C1(1 or 0)=E(2)

If C1=1 | If C1=0

R=8

S=9 , C3=0

D+E(2)=Y

D>7 (to generate a carry)

X Contradiction(Rule 3)

R=9

S=8,C3=1

D+E(2)=Y

D=4 | D=5 | D>7

X

Solution not Satisfied

(To Satisfy Y should generate carry)

Contradiction for value of 0 Comes

X

After Step 1 we derive are more conclusion that Y contradiction should generate a Carry. That is D+2>9

Step – 2

$\boxed{\begin{array}{c} M=1 \\ O=0 \end{array}}$

Or S=8 , S=9 C3 = 1 , C3 = 0

Let E=3

E(3)+O(0)+C2(1 or 0)=M

C2=1 | C2=0

E(3)+O(0)+C2(1)=N(4)

E(3)+O(0)+C2(0)=N(3)

X

Contradiction

N(4)+R+C1(1 or 0)=E(3)

C1=1 | C1=0

X

R=8

S=9

D+E(3)=y

Contraction (Y should generate carry in that case C1 cannot be equal do 0)

Constraint Satisfaction Problems (CSPs)

- **Examples of CSPs:**
  Some popular puzzles like, the Latin Square, the Eight Queens, and Sudoku are stated below.
- **Latin Square Problem :** How can one fill an **n** × **n** table with **n** different symbols such that each symbol occurs exactly once in each row and each column ?

Solutions : The Latin squares for **n** = 1, 2, 3 and 4 are :

$$\begin{pmatrix} 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{pmatrix} \quad \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \end{pmatrix}$$

Constraint Satisfaction Problems (CSPs)

❑Eight Queens Puzzle Problem : How can one put **8** queens on a **(8 x 8)** chess board such that no queen can attack any other queen ?

Solutions: The puzzle has **92** distinct solutions. If rotations and reflections of the board are counted as one, the puzzle has **12** unique solutions.

Best-first search

- To implement an informed search strategy, we need to slightly modify the skeleton for agenda-based search that we've already seen.
- Again, the crucial part of the skeleton is where we update the agenda.
- Rather than simply adding the new agenda items to the beginning (depth-first) or end (breadth-first) of the existing agenda, we add them to the existing agenda in order according to some measure of how promising we think a state is, with the most promising ones first. This gives us best-first search .
- Best-first search isn't so much a search strategy, as a mechanism for implementing many different types of informed search

.

- .

Thank You

End