Artificial Intelligence
CSC 361

Tutorial
CSP

Constraint Satisfaction Problems

What is a CSP?

- **Finite set of variables $V_1, V_2, \ldots, V_n$**
  - **Nonempty domain of possible values for each variable $D_{V1}, D_{V2}, \ldots D_{Vn}$**

- **Finite set of constraints $C_1, C_2, \ldots, C_m$**
  - **Each constraint C_i limits the values that variables can take,**
    - **e.g., $V_1 \neq V_2$**

A *state* is defined as an *assignment* of values to some or all variables.

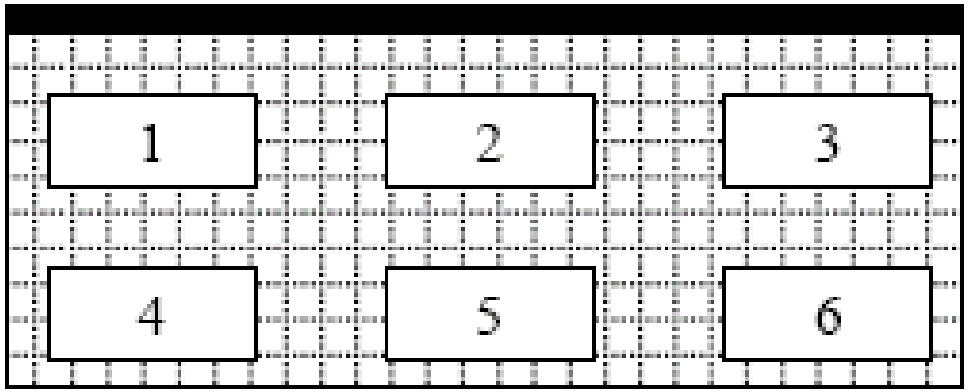
Consistent assignment

- **assignment does not violate the constraints.**

A *solution* to a CSP is a complete assignment that satisfies all constraints.

Examination Problem (Cont.)

During an examination, an invigilator has to arrange six students in a way to make sure that the examination is properly conducted and no one cheats. 3 students have makeup in Mathematics, 2 students have makeup in Computer science and 1 student in physics. As shown in figure below, there exist 6 desks arranged into two rows and three columns.



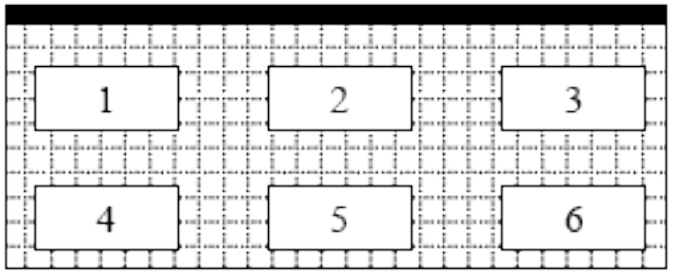
Examination Problem (Cont.)

The problem for the invigilator is to arrange students so that students having the same makeup exam should not be neighbors (in the same column or in the same row for example neighbors of 5 are 2, 4 and 6). Furthermore, the student who has Physics exam is suspected of cheating, and hence should be in the first row (desk 1 or 2 or 3).

a) Propose a **formulation of the problem** in term of CSP by specifying variables, domains and constraints.

b) Propose a solution to help the invigilator.

Examination Problem (Cont.)



- Variables

  D1, D2, D3, D4, D5, D6

- Domain

  D1 Domain = {P, M, M, M, CS, CS} D2 Domain = {P, M, M, M, CS, CS}
  D3 Domain = {P, M, M, M, CS, CS} D4 Domain = {M, M, M, CS, CS}
  D5 Domain = {M, M, M, CS, CS} D6 Domain = {M, M, M, CS, CS}

- Constraints

  D1 <> D2 D1 <> D4
  D2 <> D5 D2 <> D3 D3 <> D6
  D5 <> D6 D5 <> D4

- Solution

  M P M
  Cs M Cs

Crossword Puzzles

Consider the problem of constructing crossword puzzles: fitting words into a rectangular grid. The grid which is given as part of the problem specifies which squares are blank and which are shaded.

- Assume that a list of words (i.e. a dictionary) is provided and that the task is to fill in the blank squares using any subset of the list.

Formulate this problem in two ways:

1. as a general search
2. as a CSP.

Crossword Puzzles (General Search Problem)

- State
  - Any arrangement of n words on the puzzle.
- Initial Sate
  - No words on puzzle.
- Successor Function
  - Fill a word in the puzzle with one of the words in dictionary.
- Goal
  - Fill all the words in the puzzle.
- Path Cost
  - Each fill cost 1.

Crossword Puzzles (CSP)

CSP Problem : As a CSP there are even more choices.

A. You could have…

- **Variable**: Each box in the crossword puzzle.
- **Value** of each variable: A letter.
- **Constraints**: The letters must make words.

Crossword Puzzles (CSP)

B. Alternately, we could have…

- **Variable**: Each string of consecutive horizontal or vertical non-shaded boxes.
- **Domain** of the variables: word list.
- **Constraints**: Two intersecting words must have the same letter in intersecting box.

Course Scheduling

You are in charge of scheduling for computer science classes that meet Mondays, Wednesdays and Fridays. There are 5 classes that meet on these days and 3 professors who will be teaching these classes. You are constrained by the fact that each professor can only teach one class at a time.

The classes are:

- Class 1 - Intro to Programming: meets from 8:00-9:00am
- Class 2 - Intro to Artificial Intelligence: meets from 8:30-9:30am
- Class 3 - Natural Language Processing: meets from 9:00-10:00am
- Class 4 - Computer Vision: meets from 9:00-10:00am
- Class 5 - Machine Learning: meets from 10:30-11:30am

The professors are:

- Professor A, who is qualified to teach Classes 1, 2, and 5.
- Professor B, who is qualified to teach Classes 3, 4, and 5.
- Professor C, who is qualified to teach Classes 1, 3, and 4.

Course Scheduling (Cont.)

1. Formulate this problem as a CSP problem in which there is one variable per class, stating the domains, and constraints. Constraints should be specified formally and precisely, but may be implicit rather than explicit.

Course Scheduling (Cont.)

Variables Domains (or unary constraints)

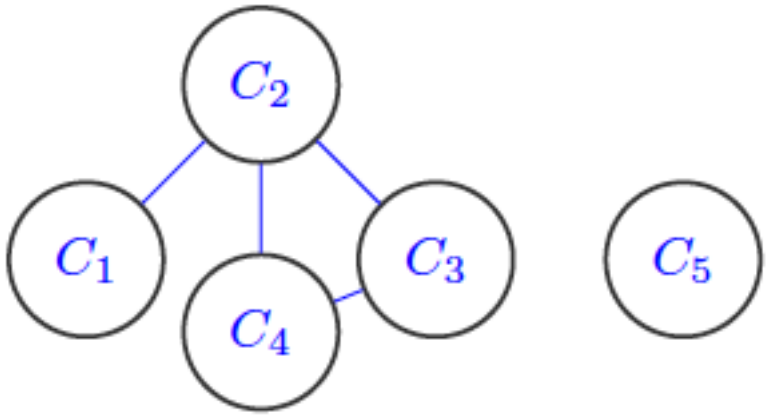
- C1 {A, C}
- C2 {A}
- C3 {B, C}
- C4 {B, C}
- C5 {A, B}

Binary Constraints

- C1 != C2
- C2 != C3
- C2 != C4
- C3 != C4

Course Scheduling (Cont.)

2. Draw the constraint graph associated with your CSP.



Backtracking

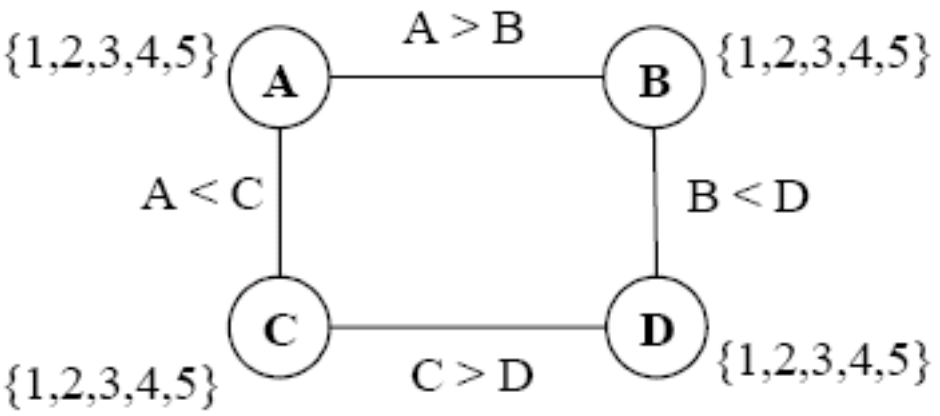
Backtracking search

Similar to Depth-first search

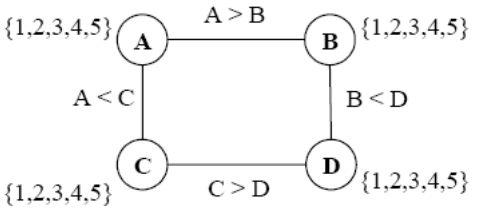
Chooses values for one variable at a time and backtracks when a variable has no legal values left to assign.

Consider the constraint graph below:

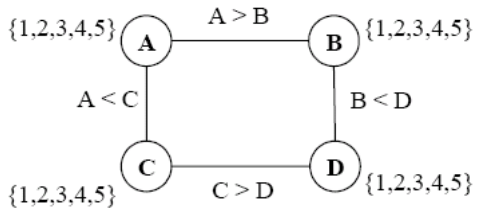
1. Trace Arc Consistency (AC3) algorithm on this graph.
2. Using backtracking with forward checking, find a solution to this problem:
   - Use the MRV heuristic to select variables. Ties are broken by alphabetical order.
   - Values are selected in ascending order.



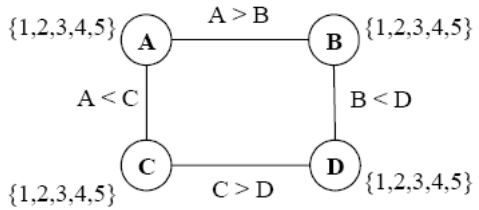
Queue	A {1,2,3,4,5}	B {1,2,3,4,5}	C {1,2,3,4,5}	D {1,2,3,4,5}	Added Q
AB					
BA					
AC					
CA					
CD					
DC					
BD					
DB					



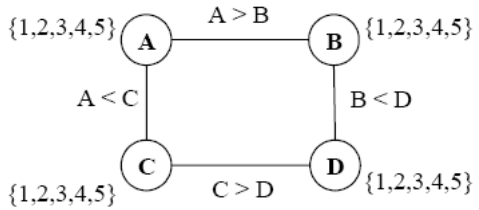
Queue	A {1,2,3,4,5}	B {1,2,3,4,5}	C {1,2,3,4,5}	D {1,2,3,4,5}	Added Q
AB	{2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	CA
BA		{1,2,3,4}			DB
AC					
CA					
CD					
DC					
BD					
DB					



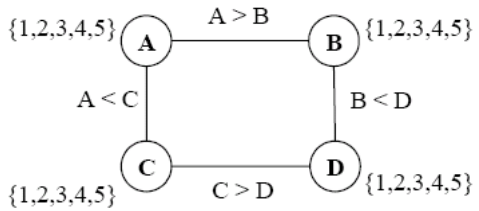
Queue	A {1,2,3,4,5}	B {1,2,3,4,5}	C {1,2,3,4,5}	D {1,2,3,4,5}	Added Q
AB	{2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	CA
BA	{2,3,4,5}	{1,2,3,4}	{1,2,3,4,5}	{1,2,3,4,5}	DB
AC	{2,3,4}				BA
CA					
CD					
DC					
BD					
DB					
BA					



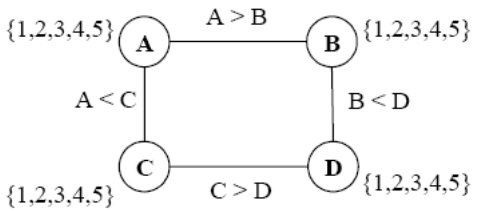
Queue	A {1,2,3,4,5}	B {1,2,3,4,5}	C {1,2,3,4,5}	D {1,2,3,4,5}	Added Q
AB	{2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	CA
BA	{2,3,4,5}	{1,2,3,4}	{1,2,3,4,5}	{1,2,3,4,5}	DB
AC	{2,3,4}	{1,2,3,4}	{1,2,3,4,5}	{1,2,3,4,5}	BA
CA			{3,4,5}		DC
CD					
DC					
BD					
DB					
BA					



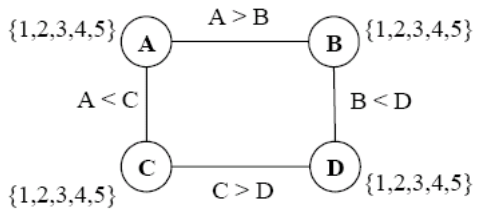
Queue	A {1,2,3,4,5}	B {1,2,3,4,5}	C {1,2,3,4,5}	D {1,2,3,4,5}	Added Q
AB	{2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	CA
BA	{2,3,4,5}	{1,2,3,4}	{1,2,3,4,5}	{1,2,3,4,5}	DB
AC	{2,3,4}	{1,2,3,4}	{1,2,3,4,5}	{1,2,3,4,5}	BA
CA	{2,3,4}	{1,2,3,4}	{3,4,5}	{1,2,3,4,5}	DC
CD					....
DC					
BD					
DB					
BA					



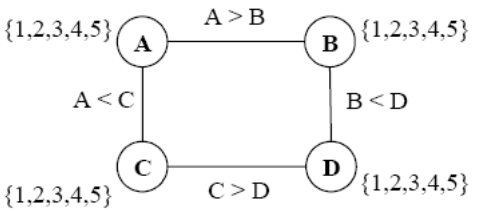
Queue	A {1,2,3,4,5}	B {1,2,3,4,5}	C {1,2,3,4,5}	D {1,2,3,4,5}	Added Q
AB	{2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	CA
BA	{2,3,4,5}	{1,2,3,4}	{1,2,3,4,5}	{1,2,3,4,5}	DB
AC	{2,3,4}	{1,2,3,4}	{1,2,3,4,5}	{1,2,3,4,5}	BA
CA	{2,3,4}	{1,2,3,4}	{3,4,5}	{1,2,3,4,5}	DC
CD	{2,3,4}	{1,2,3,4}	{3,4,5}	{1,2,3,4,5}	....
DC				{1,2,3,4}	BD
BD					
DB					
BA					



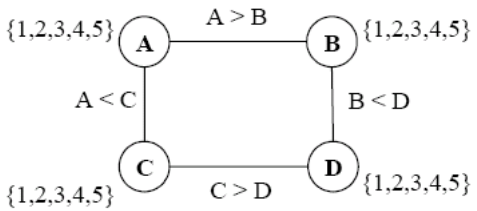
Queue	A {1,2,3,4,5}	B {1,2,3,4,5}	C {1,2,3,4,5}	D {1,2,3,4,5}	Added Q
AB	{2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	CA
BA	{2,3,4,5}	{1,2,3,4}	{1,2,3,4,5}	{1,2,3,4,5}	DB
AC	{2,3,4}	{1,2,3,4}	{1,2,3,4,5}	{1,2,3,4,5}	**BA**
CA	{2,3,4}	{1,2,3,4}	{3,4,5}	{1,2,3,4,5}	DC
CD	{2,3,4}	{1,2,3,4}	{3,4,5}	{1,2,3,4,5}	….
DC	{2,3,4}	{1,2,3,4}	{3,4,5}	{1,2,3,4}	BD
BD		{1,2,3}			**AB**
DB					
BA					
AB					



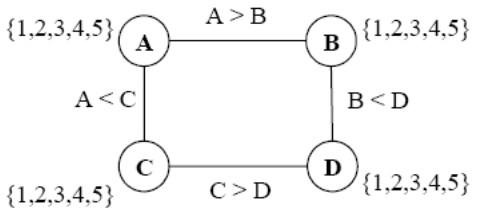
Queue	A {1,2,3,4,5}	B {1,2,3,4,5}	C {1,2,3,4,5}	D {1,2,3,4,5}	Added Q
AB	{2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	CA
BA	{2,3,4,5}	{1,2,3,4}	{1,2,3,4,5}	{1,2,3,4,5}	DB
AC	{2,3,4}	{1,2,3,4}	{1,2,3,4,5}	{1,2,3,4,5}	**BA**
CA	{2,3,4}	{1,2,3,4}	{3,4,5}	{1,2,3,4,5}	DC
CD	{2,3,4}	{1,2,3,4}	{3,4,5}	{1,2,3,4,5}	....
DC	{2,3,4}	{1,2,3,4}	{3,4,5}	{1,2,3,4}	BD
BD	{2,3,4}	{1,2,3}	{3,4,5}	{1,2,3,4}	AB
DB				{2,3,4}	**CD**
BA					
AB					
CD					



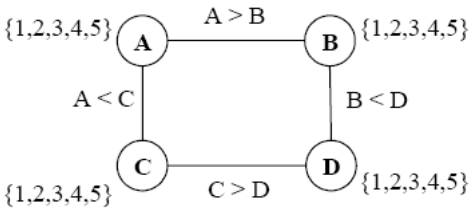
Queue	A {1,2,3,4,5}	B {1,2,3,4,5}	C {1,2,3,4,5}	D {1,2,3,4,5}	Added Q
AB	{2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	CA
BA	{2,3,4,5}	{1,2,3,4}	{1,2,3,4,5}	{1,2,3,4,5}	DB
AC	{2,3,4}	{1,2,3,4}	{1,2,3,4,5}	{1,2,3,4,5}	**BA**
CA	{2,3,4}	{1,2,3,4}	{3,4,5}	{1,2,3,4,5}	DC
CD	{2,3,4}	{1,2,3,4}	{3,4,5}	{1,2,3,4,5}	---
DC	{2,3,4}	{1,2,3,4}	{3,4,5}	{1,2,3,4}	BD
BD	{2,3,4}	{1,2,3}	{3,4,5}	{1,2,3,4}	**AB**
DB	{2,3,4}	{1,2,3}	{3,4,5}	{2,3,4}	**CD**
BA	{2,3,4}	{1,2,3}	{3,4,5}	{2,3,4}	---



Queue	A {1,2,3,4,5}	B {1,2,3,4,5}	C {1,2,3,4,5}	D {1,2,3,4,5}	Added Q
AB	{2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	CA
BA	{2,3,4,5}	{1,2,3,4}	{1,2,3,4,5}	{1,2,3,4,5}	DB
AC	{2,3,4}	{1,2,3,4}	{1,2,3,4,5}	{1,2,3,4,5}	BA
CA	{2,3,4}	{1,2,3,4}	{3,4,5}	{1,2,3,4,5}	DC
CD	{2,3,4}	{1,2,3,4}	{3,4,5}	{1,2,3,4,5}	---
DC	{2,3,4}	{1,2,3,4}	{3,4,5}	{1,2,3,4}	BD
BD	{2,3,4}	{1,2,3}	{3,4,5}	{1,2,3,4}	AB
DB	{2,3,4}	{1,2,3}	{3,4,5}	{2,3,4}	CD
BA	{2,3,4}	{1,2,3}	{3,4,5}	{2,3,4}	---
AB	{2,3,4}	{1,2,3}	{3,4,5}	{2,3,4}	---



Queue	A {1,2,3,4,5}	B {1,2,3,4,5}	C {1,2,3,4,5}	D {1,2,3,4,5}	Added Q
AB	{2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	CA
BA	{2,3,4,5}	{1,2,3,4}	{1,2,3,4,5}	{1,2,3,4,5}	DB
AC	{2,3,4}	{1,2,3,4}	{1,2,3,4,5}	{1,2,3,4,5}	BA
CA	{2,3,4}	{1,2,3,4}	{3,4,5}	{1,2,3,4,5}	DC
CD	{2,3,4}	{1,2,3,4}	{3,4,5}	{1,2,3,4,5}	---
DC	{2,3,4}	{1,2,3,4}	{3,4,5}	{1,2,3,4}	BD
BD	{2,3,4}	{1,2,3}	{3,4,5}	{1,2,3,4}	AB
DB	{2,3,4}	{1,2,3}	{3,4,5}	{2,3,4}	CD
BA	{2,3,4}	{1,2,3}	{3,4,5}	{2,3,4}	---
AB	{2,3,4}	{1,2,3}	{3,4,5}	{2,3,4}	---
CD	{2,3,4}	{1,2,3}	{3,4,5}	{2,3,4}	---



Minimum remaining values (MRV)

Choose variable with the fewest legal moves.

- e.g., will immediately detect failure if X has no legal values.

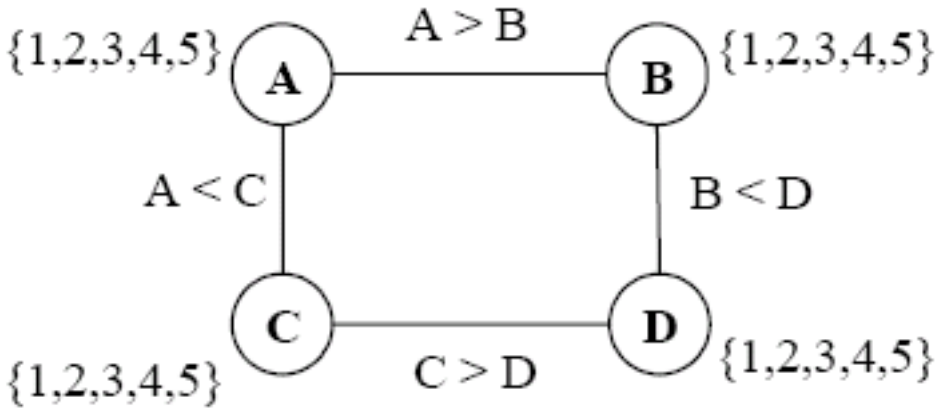
Forward checking

Can we detect inevitable failure early?

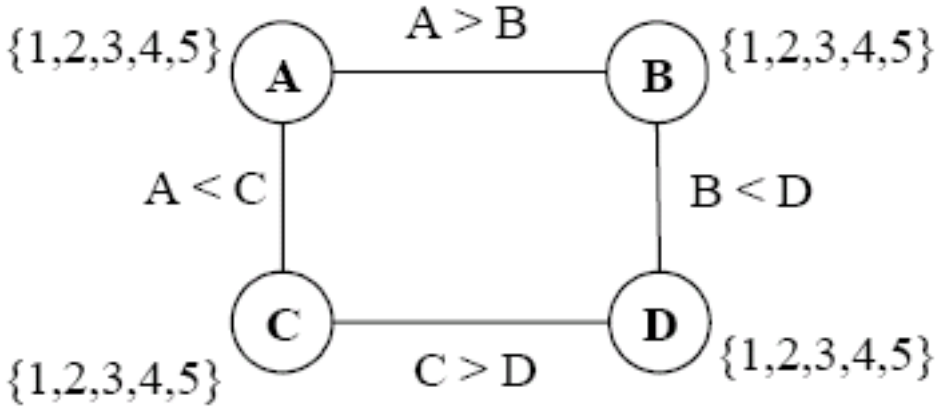
- *And avoid it later?*

Forward checking idea: keep track of remaining legal values for unassigned variables.

Terminate search when any variable has no legal values.



Assignment	A {1,2,3,4,5}	B {1,2,3,4,5}	C {1,2,3,4,5}	D {1,2,3,4,5}	
A=1					



Assignment	A {1,2,3,4,5}	B {1,2,3,4,5}	C {1,2,3,4,5}	D {1,2,3,4,5}	
A=1	1	{}	{2,3,4,5}	{1,2,3,4,5}	Failure
A=2	2	{1}	{3,4,5}	{1,2,3,4,5}	
B=1	2	1	{3,4,5}	{2,3,4,5}	
C=3	2	1	3	{2}	
D=2	2	1	3	2	Goal