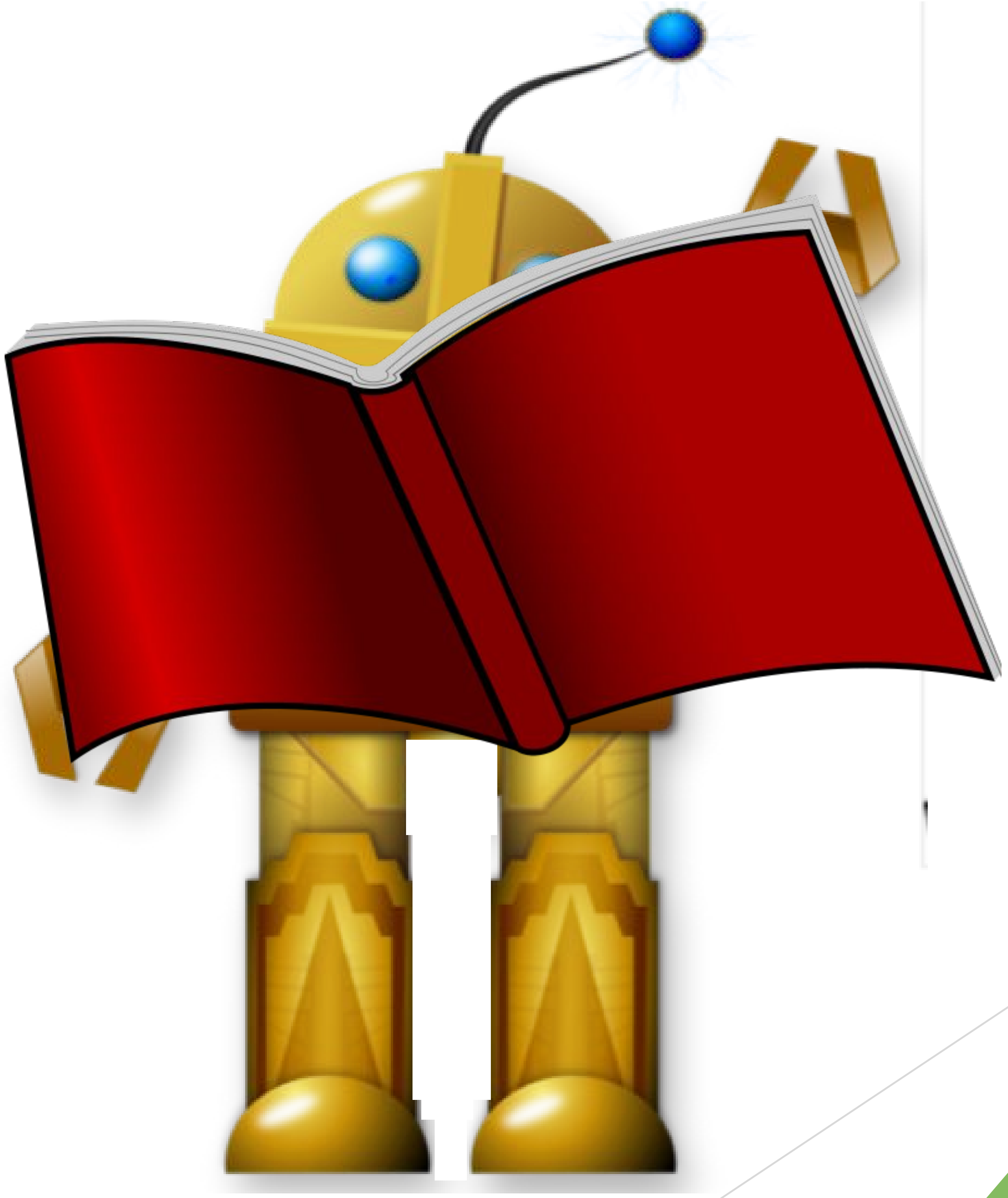
Review



Midterm

- Intelligent Agents
- Search (Informed and Uninformed)
- Hill Climbing
- Constraint Satisfaction Problems
- Adversarial Search
- Logical Agents - Propositional Logic

Review

- Informed search
- Propositional Logic
- CSPs

Problem

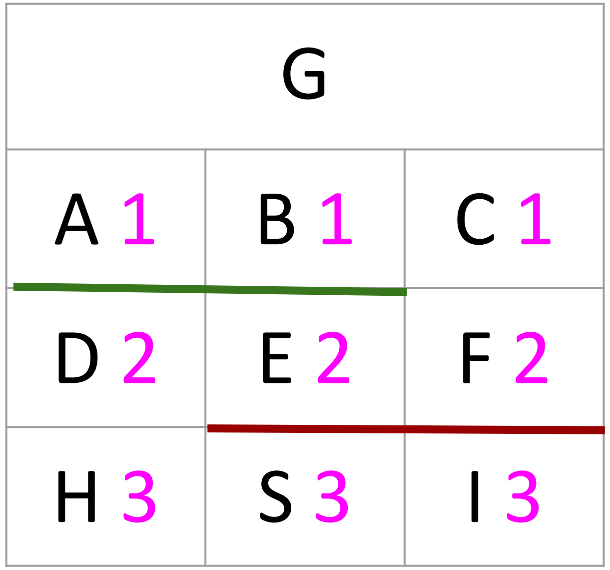
Costs:

- no walls: 1
- green wall: 2
- red wall: 4

Heuristic

Costs:

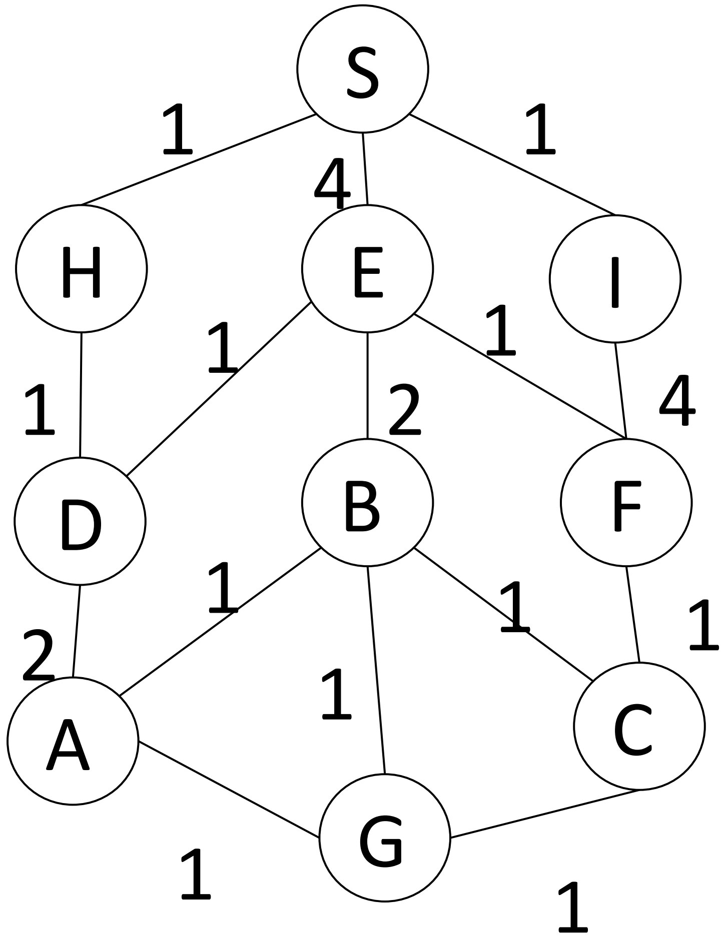
- no walls: 1
- green wall: 2
- red wall: 4



Heuristic

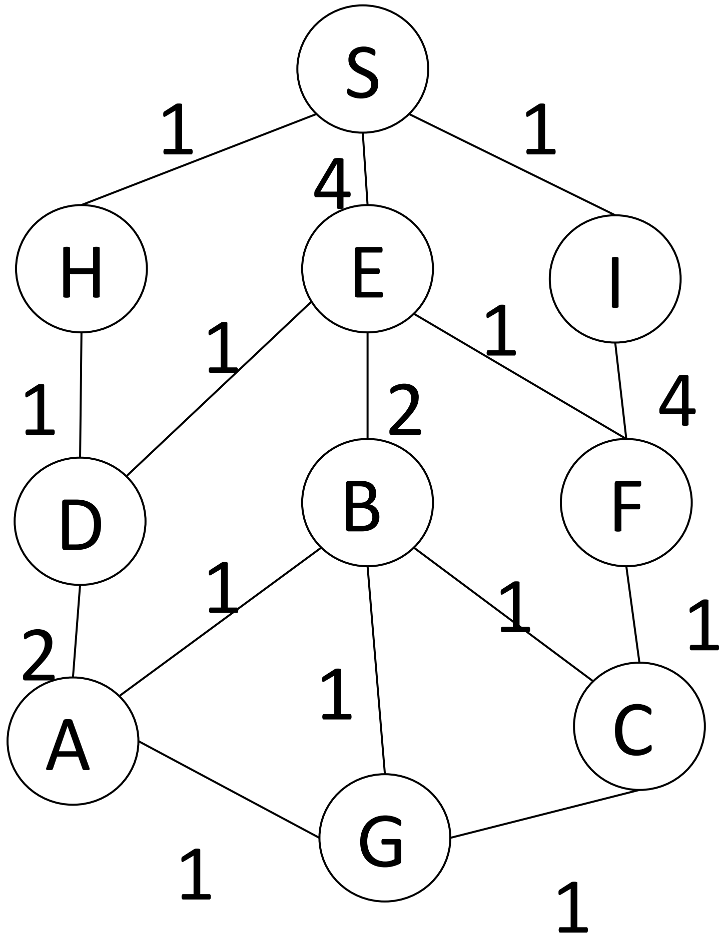
- admissible?

Search Graph



G		
A 1	B 1	C 1
D 2	E 2	F 2
H 3	S 3	I 3

Greedy Solution

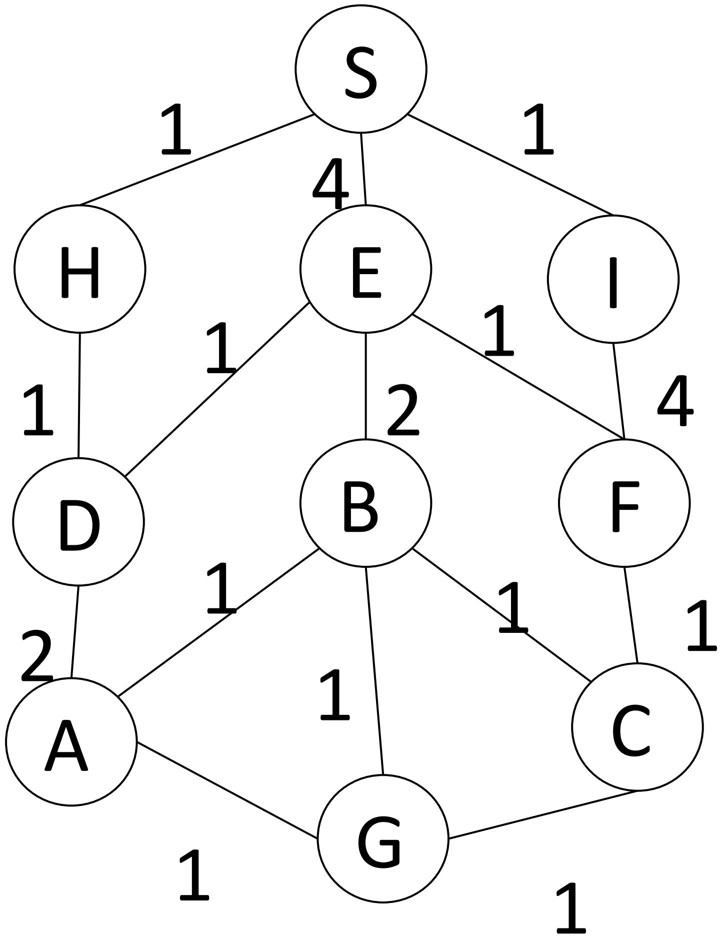


G		
A 1	B 1	C 1
D 2	E 2	F 2
H 3	S 3	I 3

Greedy: SEBG

Cost: 4 + 2 + 1 = 7

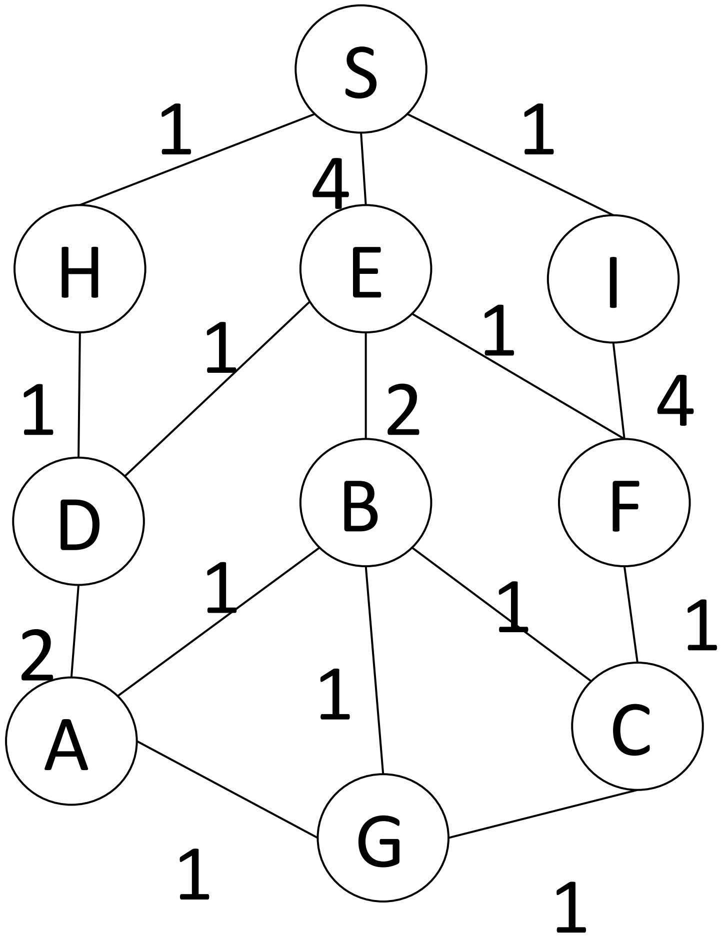
A* Solution



G		
A 1	B 1	C 1
D 2	E 2	F 2
H 3	S 3	I 3

Fringe	g	h	f
S	0	3	3

A* Solution



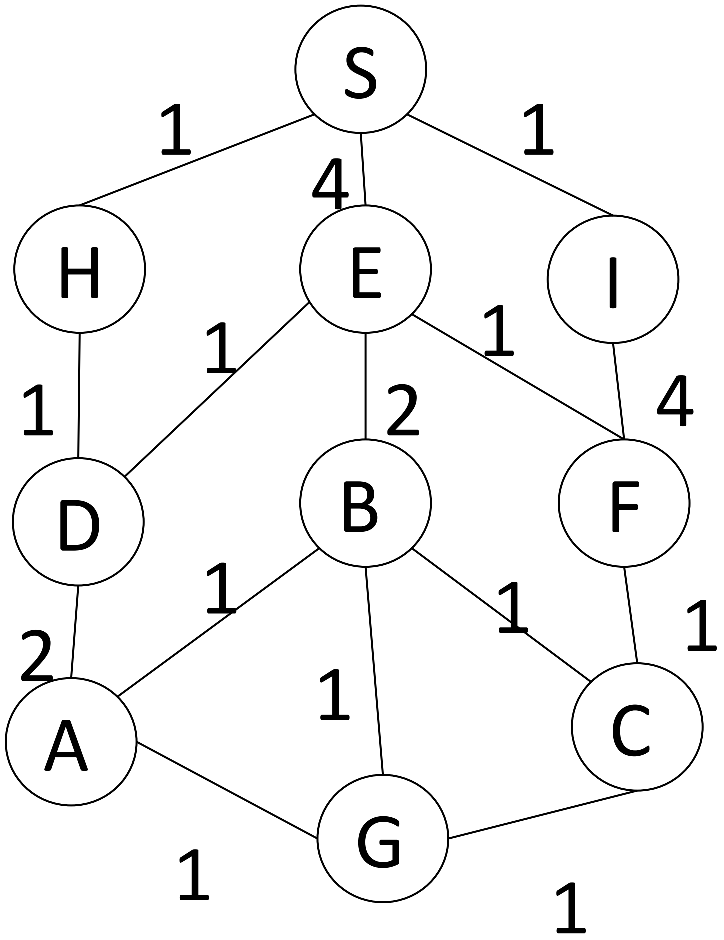
G		
A 1	B 1	C 1
D 2	E 2	F 2
H 3	S 3	I 3

Fringe	g	h	f
~~S~~	0	3	3
SE	4	2	6
SH	1	3	4
SI	1	3	4

Nodes expanded:

S

A* Solution



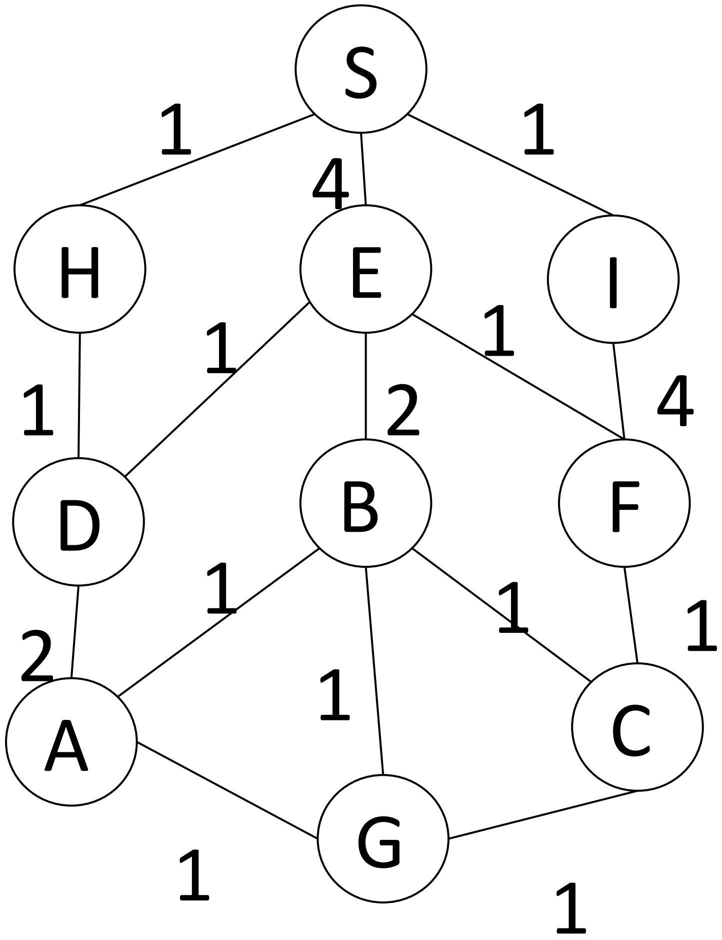
G		
A 1	B 1	C 1
D 2	E 2	F 2
H 3	S 3	I 3

Fringe	g	h	f
~~S~~	0	3	3
SE	4	2	6
~~SH~~	1	3	4
SI	1	3	4
SHD	2	2	4
SHS	2	3	5

Nodes expanded:

S, H

A* Solution



G		
A 1	B 1	C 1
D 2	E 2	F 2
H 3	S 3	I 3

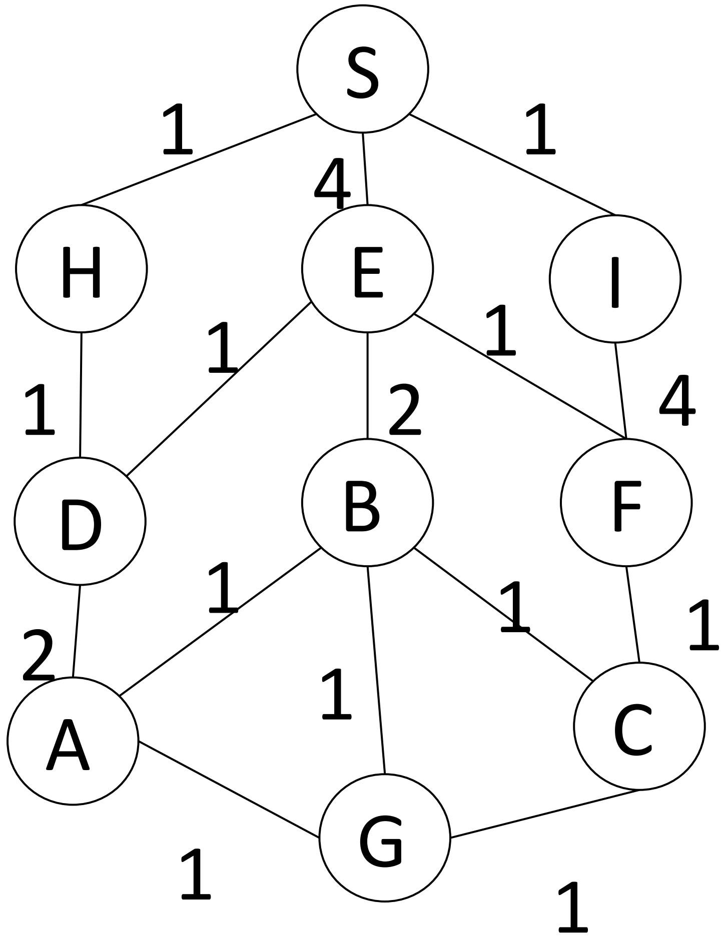
We'll ignore SHDH

Fringe	g	h	f
~~S~~	0	3	3
SE	4	2	6
~~SH~~	1	3	4
SI	1	3	4
~~SHD~~	2	2	4
SHS	2	3	5
SHDA	4	1	5
SHDE	3	2	5

Nodes expanded:

S, H, D

A* Solution



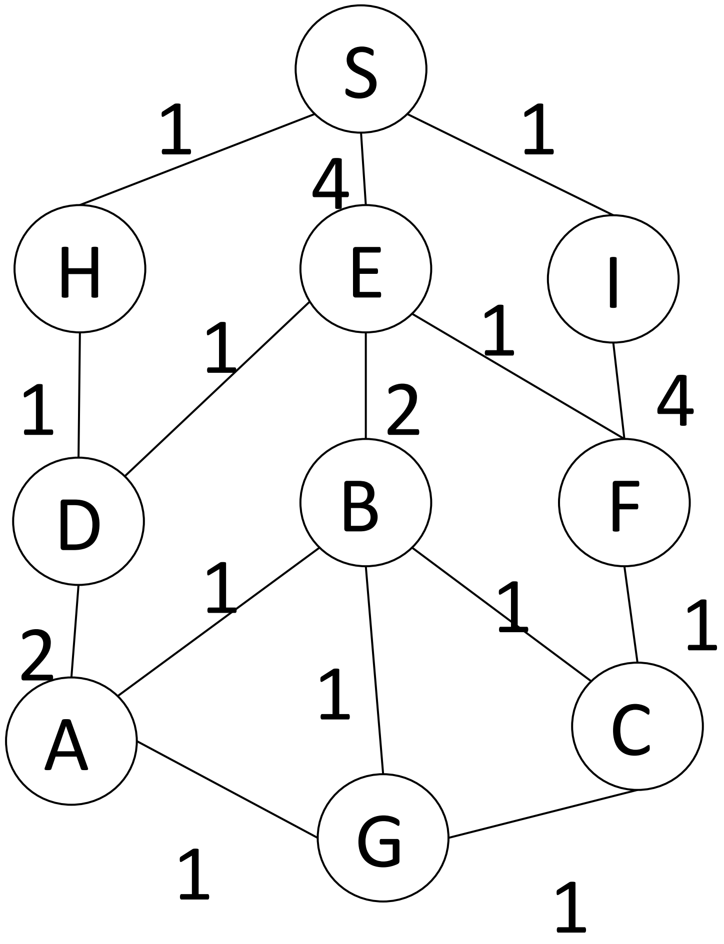
G		
A 1	B 1	C 1
D 2	E 2	F 2
H 3	S 3	I 3

Fringe	g	h	f
~~S~~	0	3	3
SE	4	2	6
~~SH~~	1	3	4
~~SI~~	1	3	4
~~SHD~~	2	2	4
SHS	2	3	5
SHDA	4	1	5
SHDE	3	2	5
SIF	5	2	7

Nodes expanded:

S, H, D, I

A* Solution



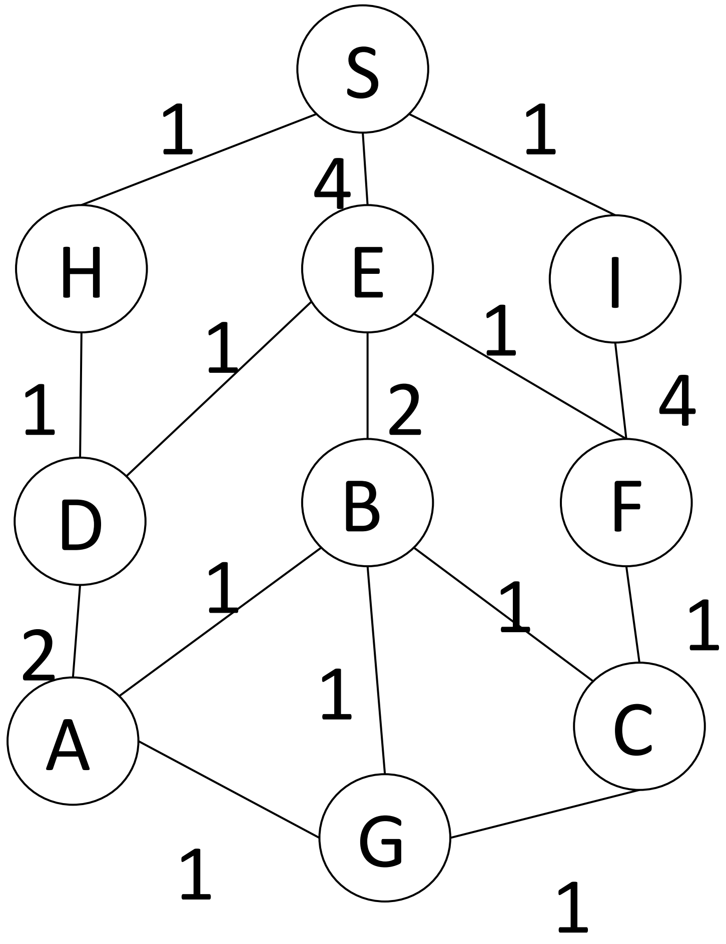
G		
A 1	B 1	C 1
D 2	E 2	F 2
H 3	S 3	I 3

Fringe	g	h	f
~~S~~	0	3	3
SE	4	2	6
~~SH~~	1	3	4
~~SI~~	1	3	4
~~SHD~~	2	2	4
SHS	2	3	5
~~SHDA~~	4	1	5
SHDE	3	2	5
SIF	5	2	7
SHDAG	5	0	5
SHDAB	5	1	6

Nodes expanded:

S, H, D, I, A

A* Solution



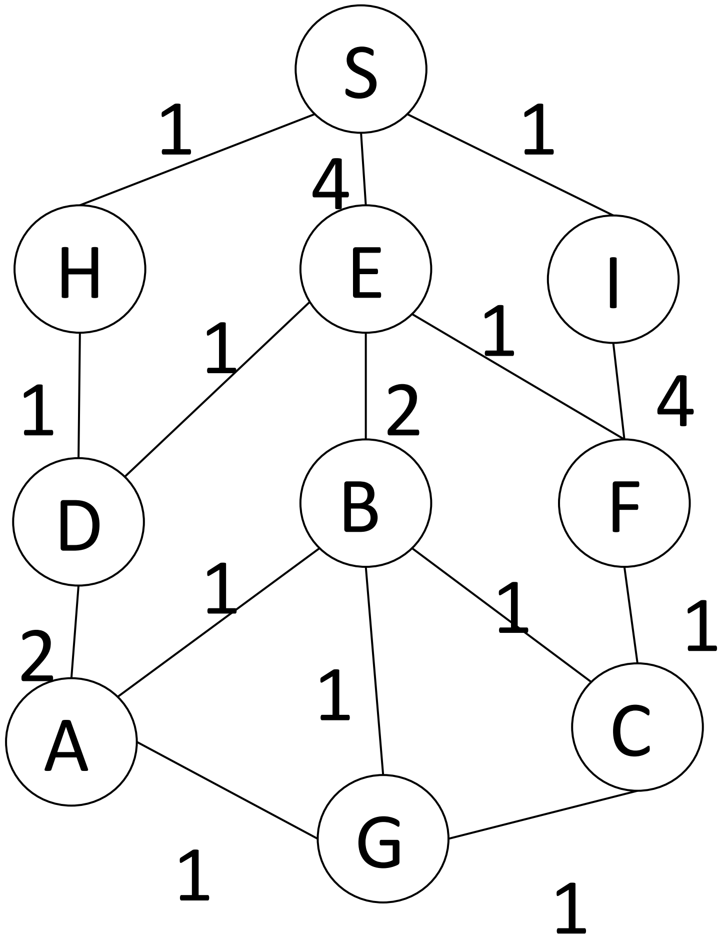
G		
A 1	B 1	C 1
D 2	E 2	F 2
H 3	S 3	I 3

Nodes expanded:

S, H, D, I, A, E

Fringe	g	h	f
~~S~~	0	3	3
SE	4	2	6
~~SH~~	1	3	4
~~SI~~	1	3	4
~~SHD~~	2	2	4
~~SHS~~	2	3	5
~~SHDA~~	4	1	5
~~SHDE~~	3	2	5
SIF	5	2	7
SHDAG	5	0	5
SHDAB	5	1	6
SHDEF	4	2	6
SHDEB	7	1	8

A* Solution



G		
A 1	B 1	C 1
D 2	E 2	F 2
H 3	S 3	I 3

Fringe	g	h	f
~~S~~	0	3	3
SE	4	2	6
~~SH~~	1	3	4
~~SI~~	1	3	4
~~SHD~~	2	2	4
~~SHS~~	2	3	5
~~SHDA~~	4	1	5
~~SHDE~~	3	2	5
SIF	5	2	7
$\boxed{\text{SHDAG}}$	5	0	5
SHDAB	5	1	6
SHDEF	4	2	6
SHDEB	7	1	8

Nodes expanded:

S, H, D, I, A, E

Path: SHDAG

Cost: 5

Propositional Logic

$P \Rightarrow Q \vee Q \Rightarrow P$

A. valid

B. satisfiable

C. unsatisfiable

$(\alpha \wedge \beta)$	$\equiv$	$(\beta \wedge \alpha)$	commutativity of $\wedge$
$(\alpha \vee \beta)$	$\equiv$	$(\beta \vee \alpha)$	commutativity of $\vee$
$((\alpha \wedge \beta) \wedge \gamma)$	$\equiv$	$(\alpha \wedge (\beta \wedge \gamma))$	associativity of $\wedge$
$((\alpha \vee \beta) \vee \gamma)$	$\equiv$	$(\alpha \vee (\beta \vee \gamma))$	associativity of $\vee$
$\neg(\neg\alpha)$	$\equiv$	α	double-negation elimination
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\beta \Rightarrow \neg\alpha)$	contraposition
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\alpha \vee \beta)$	implication elimination
$(\alpha \Leftrightarrow \beta)$	$\equiv$	$((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$	biconditional elimination
$\neg(\alpha \wedge \beta)$	$\equiv$	$(\neg\alpha \vee \neg\beta)$	De Morgan
$\neg(\alpha \vee \beta)$	$\equiv$	$(\neg\alpha \wedge \neg\beta)$	De Morgan
$(\alpha \wedge (\beta \vee \gamma))$	$\equiv$	$((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$	distributivity of $\wedge$ over $\vee$
$(\alpha \vee (\beta \wedge \gamma))$	$\equiv$	$((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$	distributivity of $\vee$ over $\wedge$

Propositional Logic

$P \Rightarrow Q \lor Q \Rightarrow P$

$(\neg P \lor Q) \lor (\neg Q \lor P)$

$\neg P \lor P \lor \neg Q \lor Q$

valid

$(\alpha \land \beta) \equiv (\beta \land \alpha)$	commutativity of $\land$
$(\alpha \lor \beta) \equiv (\beta \lor \alpha)$	commutativity of $\lor$
$((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma))$	associativity of $\land$
$((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma))$	associativity of $\lor$
$\neg(\neg\alpha) \equiv \alpha$	double-negation elimination
$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$	contraposition
$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \lor \beta)$	implication elimination
$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha))$	biconditional elimination
$\neg(\alpha \land \beta) \equiv (\neg\alpha \lor \neg\beta)$	De Morgan
$\neg(\alpha \lor \beta) \equiv (\neg\alpha \land \neg\beta)$	De Morgan
$(\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma))$	distributivity of $\land$ over $\lor$
$(\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma))$	distributivity of $\lor$ over $\land$

Propositional Logic

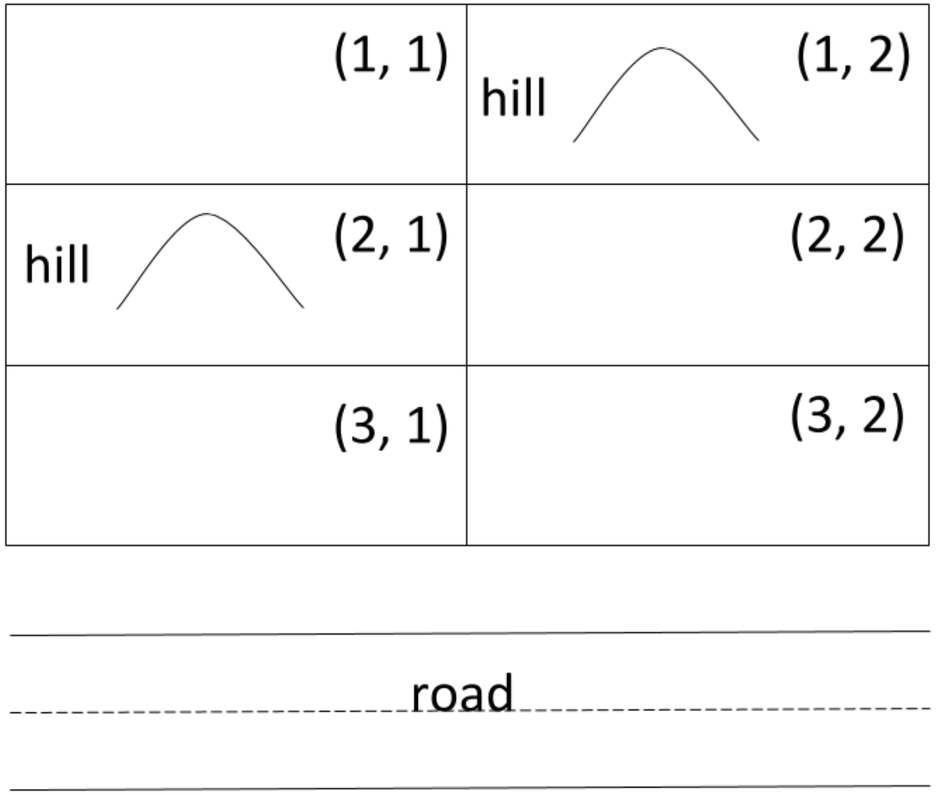
$A \vee \neg A$: valid

$A \wedge \neg A$: unsatisfiable

CSP

Our task is to design the layout of a small college.

The campus will have four structures: an administration structure (A), a bus stop (B), a classroom (C), and a dormitory (D). Each structure (including the bus stop) must be placed somewhere on the grid.



CSP

The layout must satisfy the following constraints:

1. The bus stop (B) must be adjacent to the road.
2. The administration structure (A) and the classroom (C) must both be adjacent to the bus stop (B).
3. The classroom (C) must be adjacent to the dormitory (D).
4. The administration structure (A) must not be adjacent to the dormitory (D).
5. The administration structure (A) must not be on a hill.
6. The dormitory (D) must be on a hill.
7. All structures must be in different grid squares.

CSP

Unary Constraints:

1. The bus stop (B) must be adjacent to the road.
2. The administration structure (A) and the classroom (C) must both be adjacent to the bus stop (B).
3. The classroom (C) must be adjacent to the dormitory (D).
4. The administration structure (A) must not be adjacent to the dormitory (D).
5. The administration structure (A) must not be on a hill.
6. The dormitory (D) must be on a hill.
7. All structures must be in different grid squares.

CSP

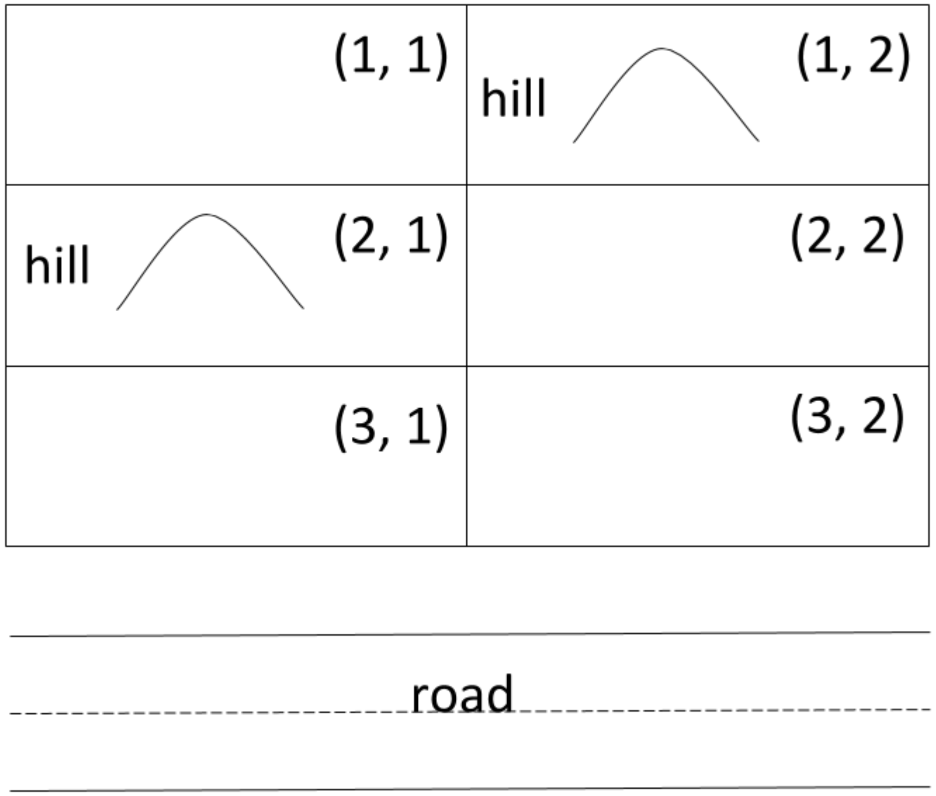
Initial domains:

A: {(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)}

B: {(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)}

C: {(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)}

D: {(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)}



CSP

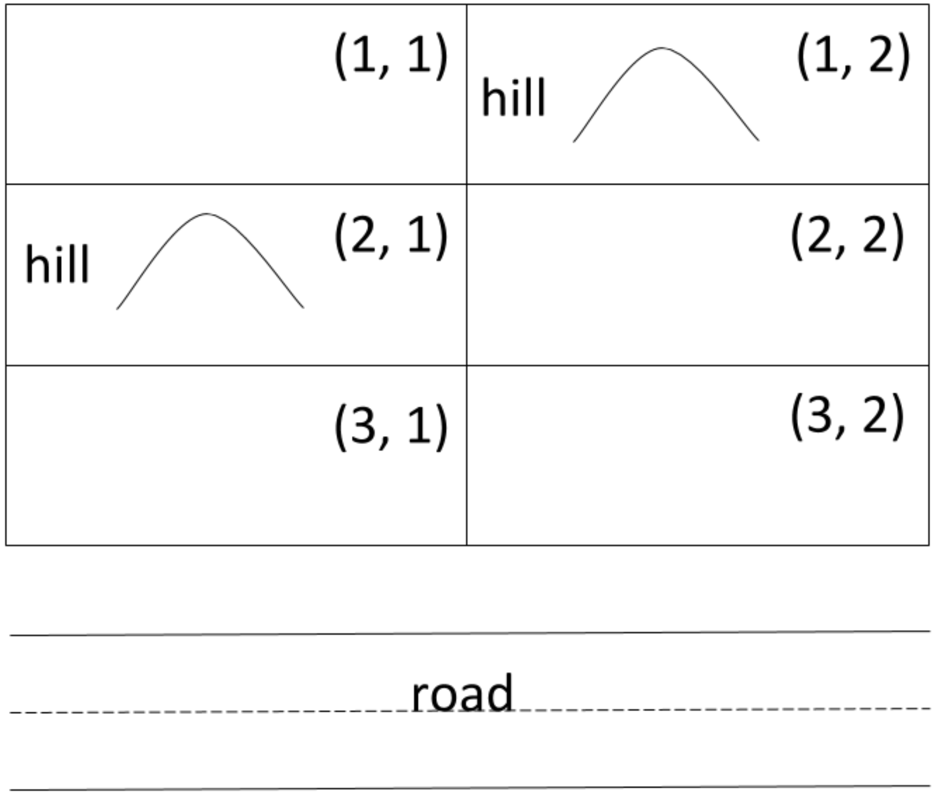
The bus stop (B) must be adjacent to the road.

A: {(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)}

B: {(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)}

C: {(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)}

D: {(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)}



CSP

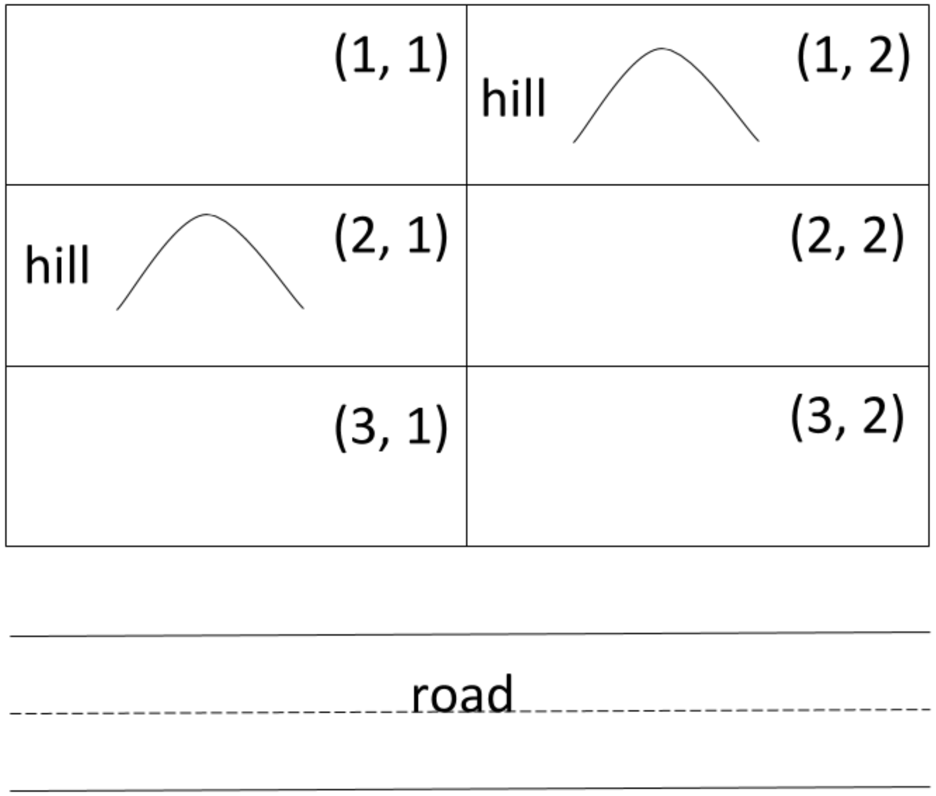
The bus stop (B) must be adjacent to the road.

A: {(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)}

B: {~~(1, 1), (1, 2), (2, 1), (2, 2),~~ (3, 1), (3, 2)}

C: {(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)}

D: {(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)}



CSP

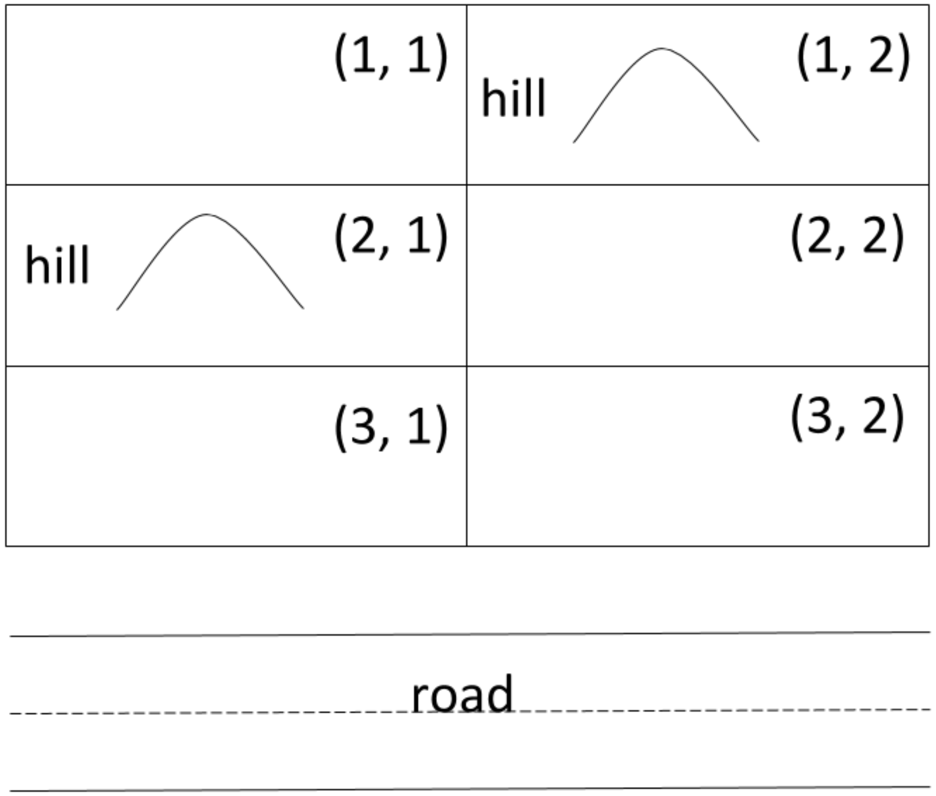
The administration structure (A) must not be on a hill.

A: {(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)}

B: {~~(1, 1), (1, 2), (2, 1), (2, 2),~~ (3, 1), (3, 2)}

C: {(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)}

D: {(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)}



CSP

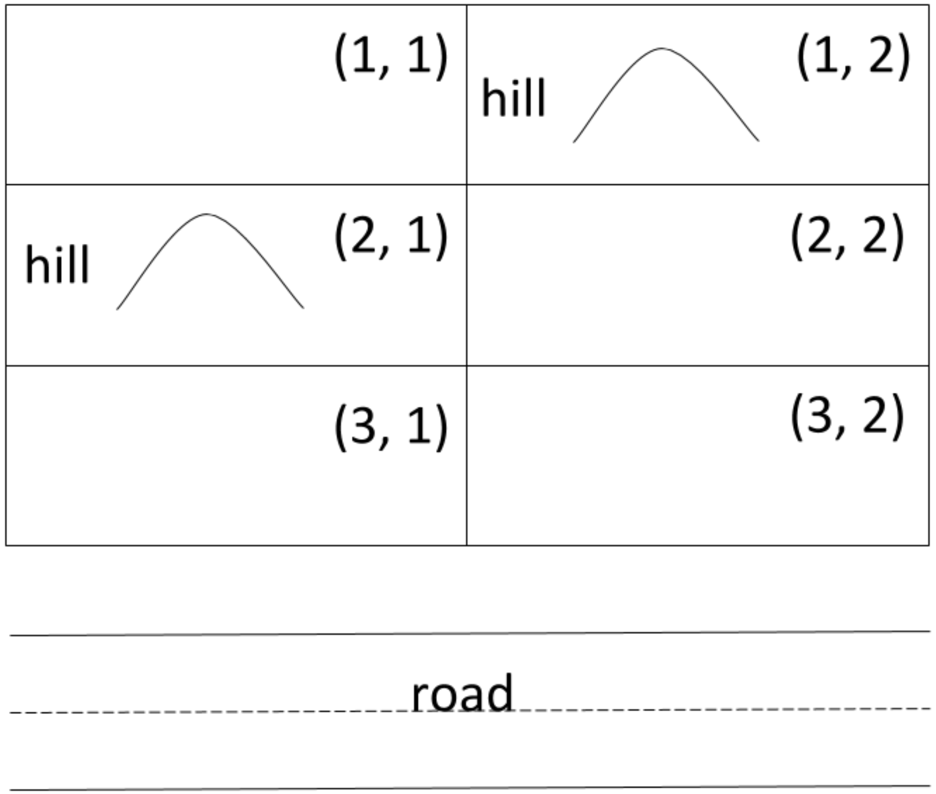
The administration structure (A) must not be on a hill.

A: {(1, 1), ~~(1, 2)~~, ~~(2, 1)~~, (2, 2), (3, 1), (3, 2)}

B: {~~(1, 1), (1, 2), (2, 1), (2, 2),~~ (3, 1), (3, 2)}

C: {(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)}

D: {(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)}



CSP

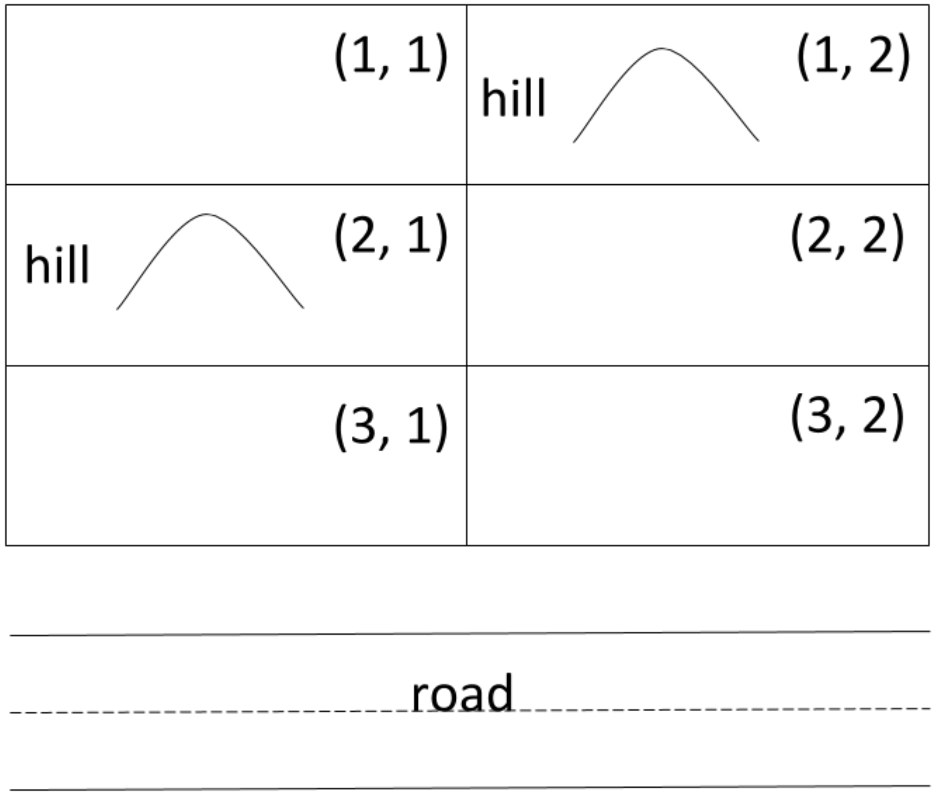
The dormitory (D) must be on a hill.

A: {(1, 1), ~~(1, 2)~~, ~~(2, 1)~~, (2, 2), (3, 1), (3, 2)}

B: {~~(1, 1), (1, 2), (2, 1), (2, 2),~~ (3, 1), (3, 2)}

C: {(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)}

D: {(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)}



CSP

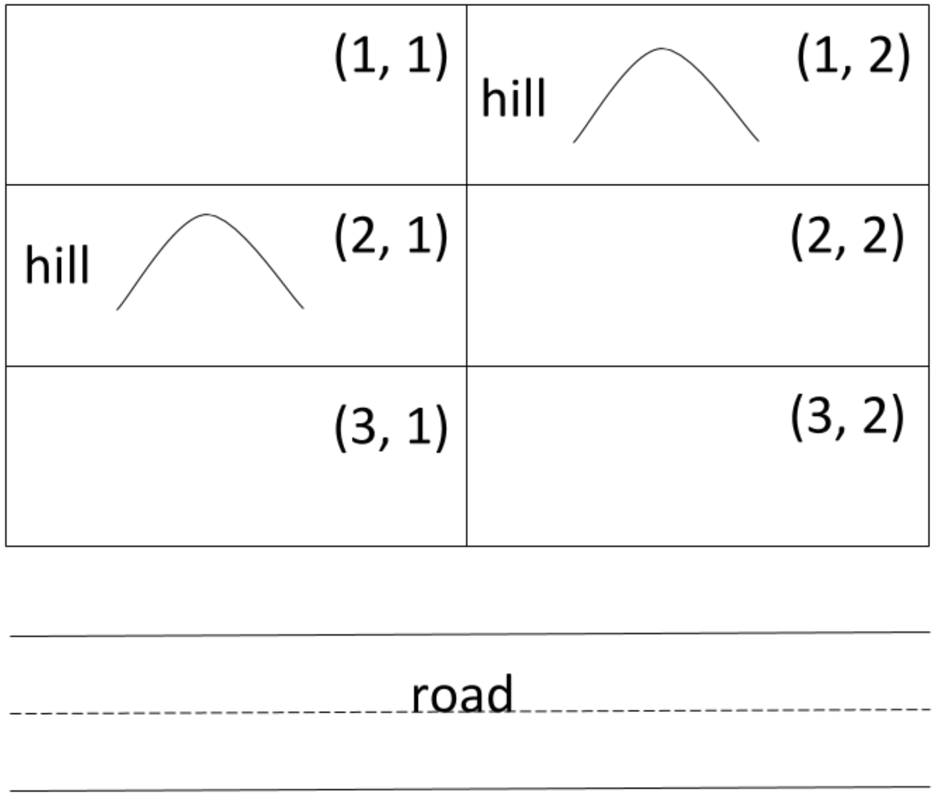
The dormitory (D) must be on a hill.

A: {(1, 1), ~~(1, 2)~~, ~~(2, 1)~~, (2, 2), (3, 1), (3, 2)}

B: {~~(1, 1), (1, 2), (2, 1), (2, 2),~~ (3, 1), (3, 2)}

C: {(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)}

D: {~~(1, 1)~~, (1, 2), (2, 1), ~~(2, 2), (3, 1), (3, 2)}~~



CSP

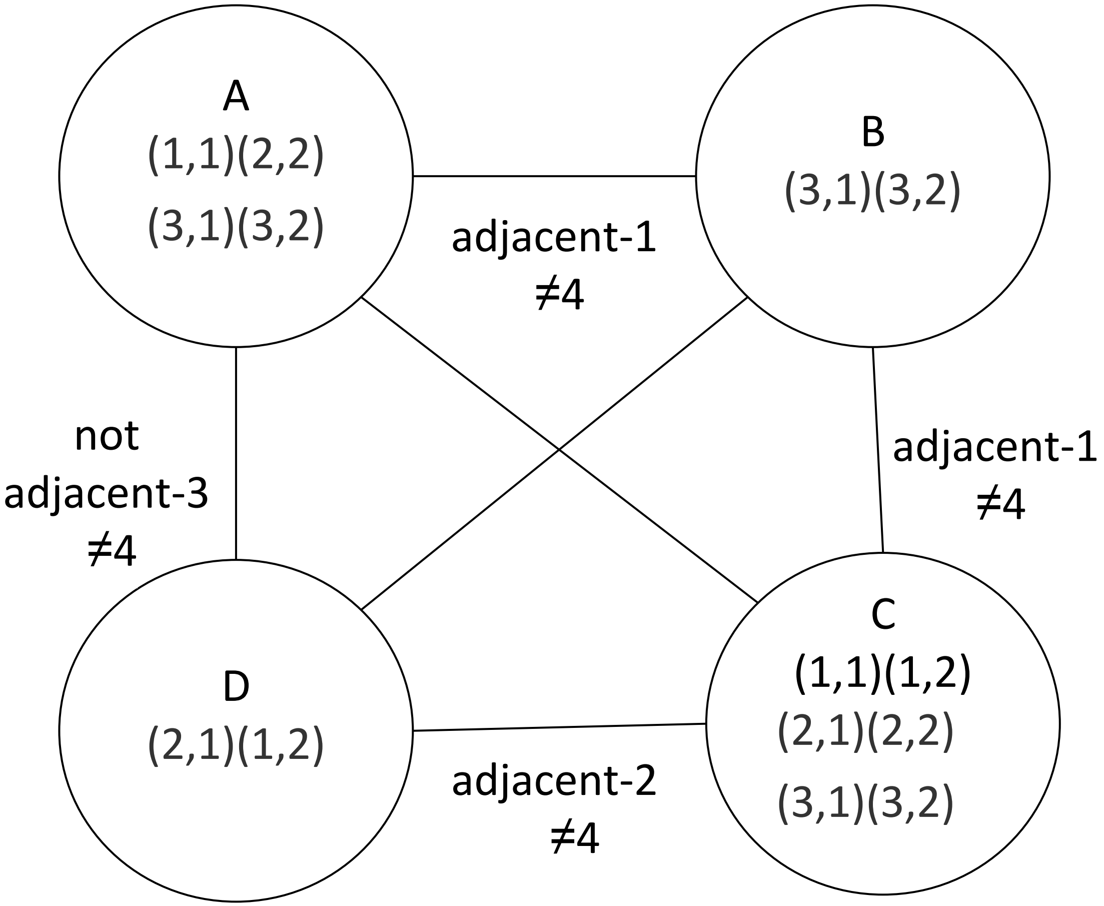
Binary Constraints:

1. The bus stop (B) must be adjacent to the road.
2. The administration structure (A) and the classroom (C) must both be adjacent to the bus stop (B).
3. The classroom (C) must be adjacent to the dormitory (D).
4. The administration structure (A) must not be adjacent to the dormitory (D).
5. The administration structure (A) must not be on a hill.
6. The dormitory (D) must be on a hill.
7. All structures must be in different grid squares.

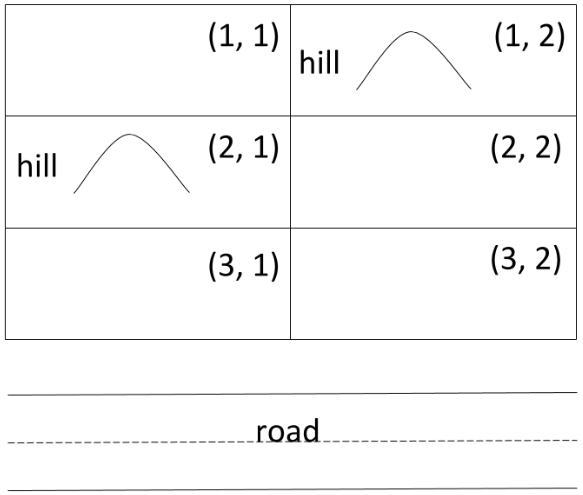
CSP

Binary Constraints:

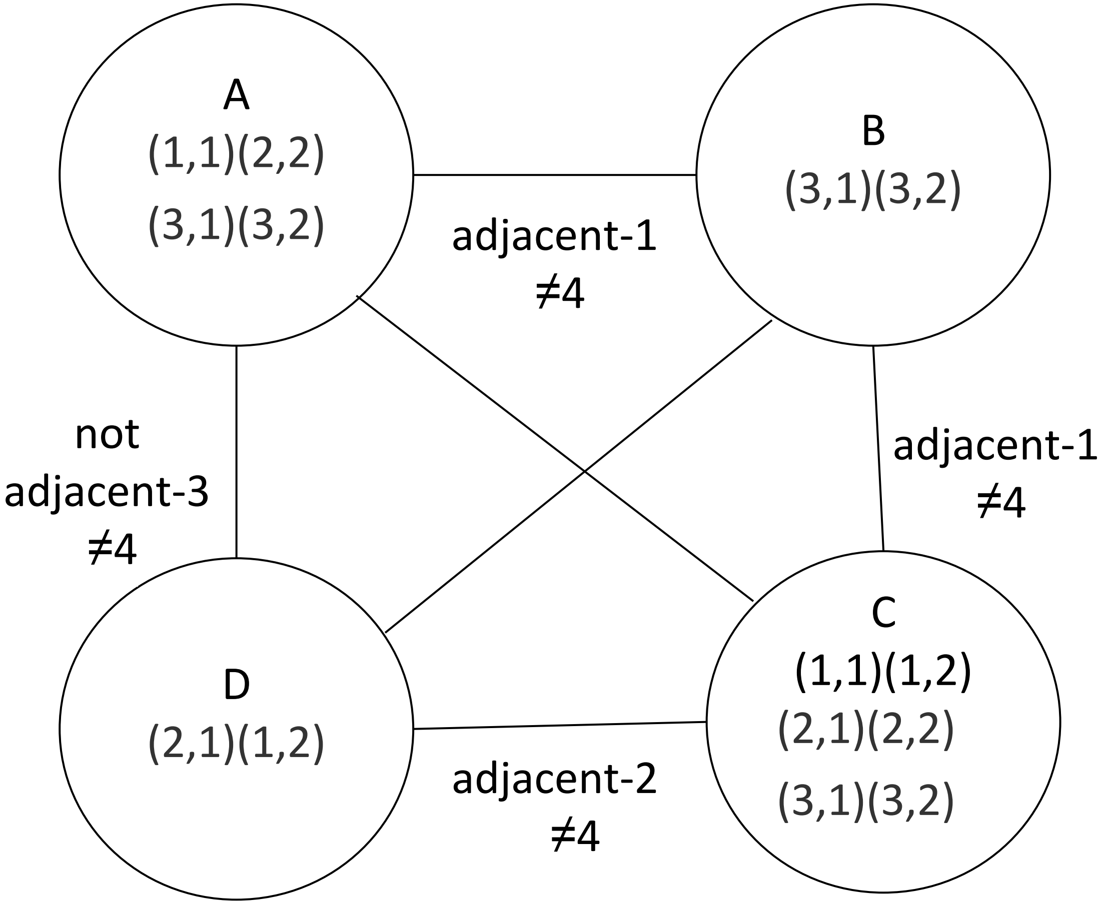
1. The administration structure (A) and the classroom (C) must both be adjacent to the bus stop (B).
2. The classroom (C) must be adjacent to the dormitory (D).
3. The administration structure (A) must not be adjacent to the dormitory (D).
4. All structures must be in different grid squares.



CSP



$A \longrightarrow B$

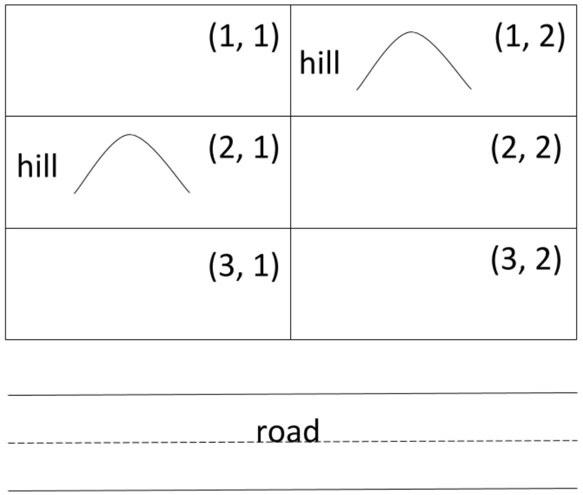


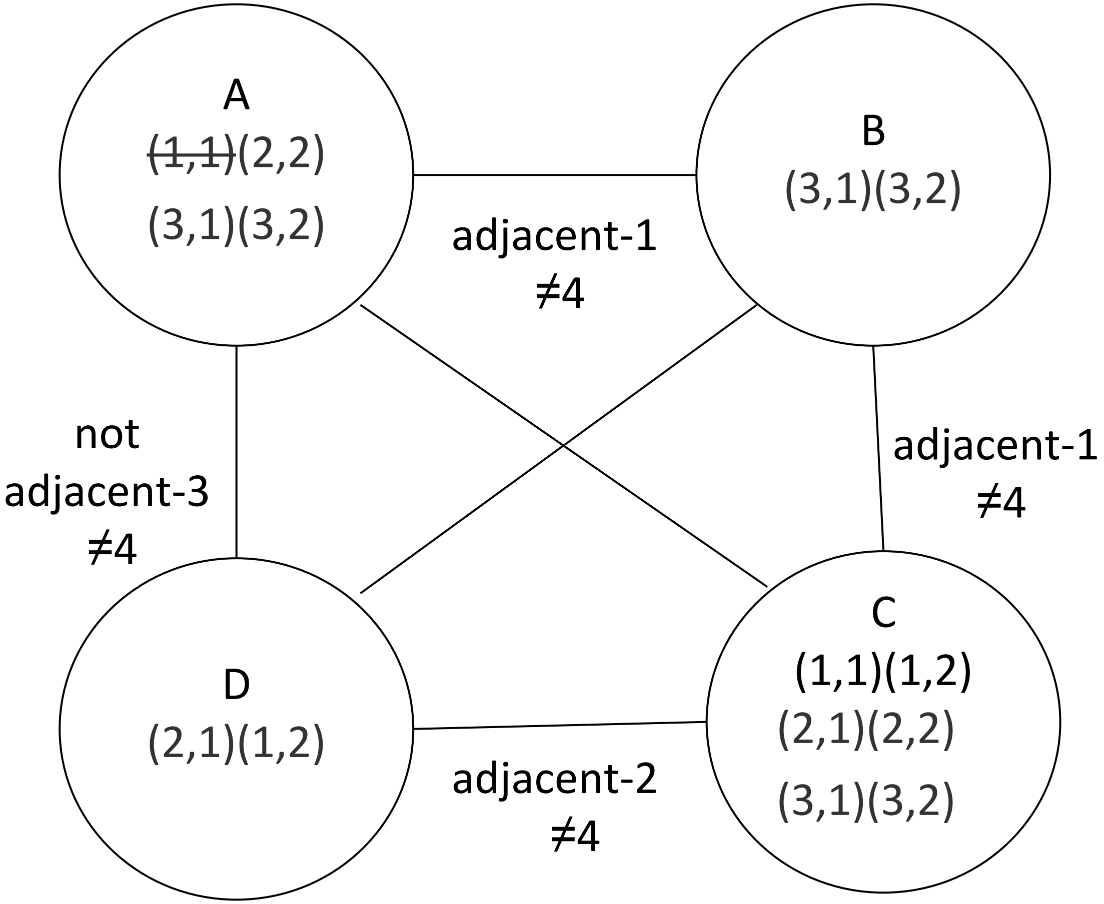
An arc $X \longrightarrow Y$ is consistent iff for *every* x in the tail there is *some allowed* y in the head which could be assigned without violating a constraint.

For A in (1, 1): no value for B that satisfies constraint: Remove (1, 1) from A's domain

CSP

A ⟶ B

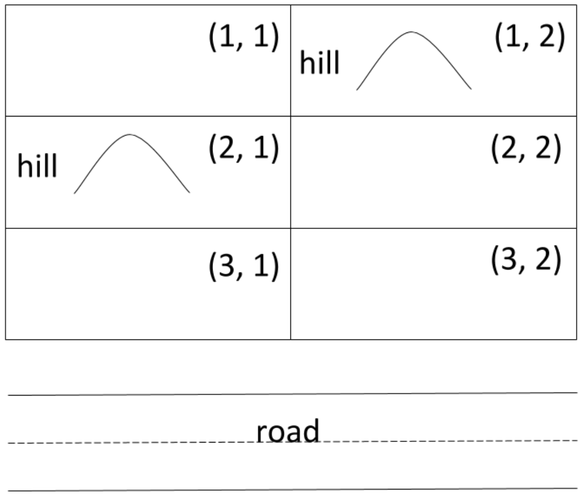


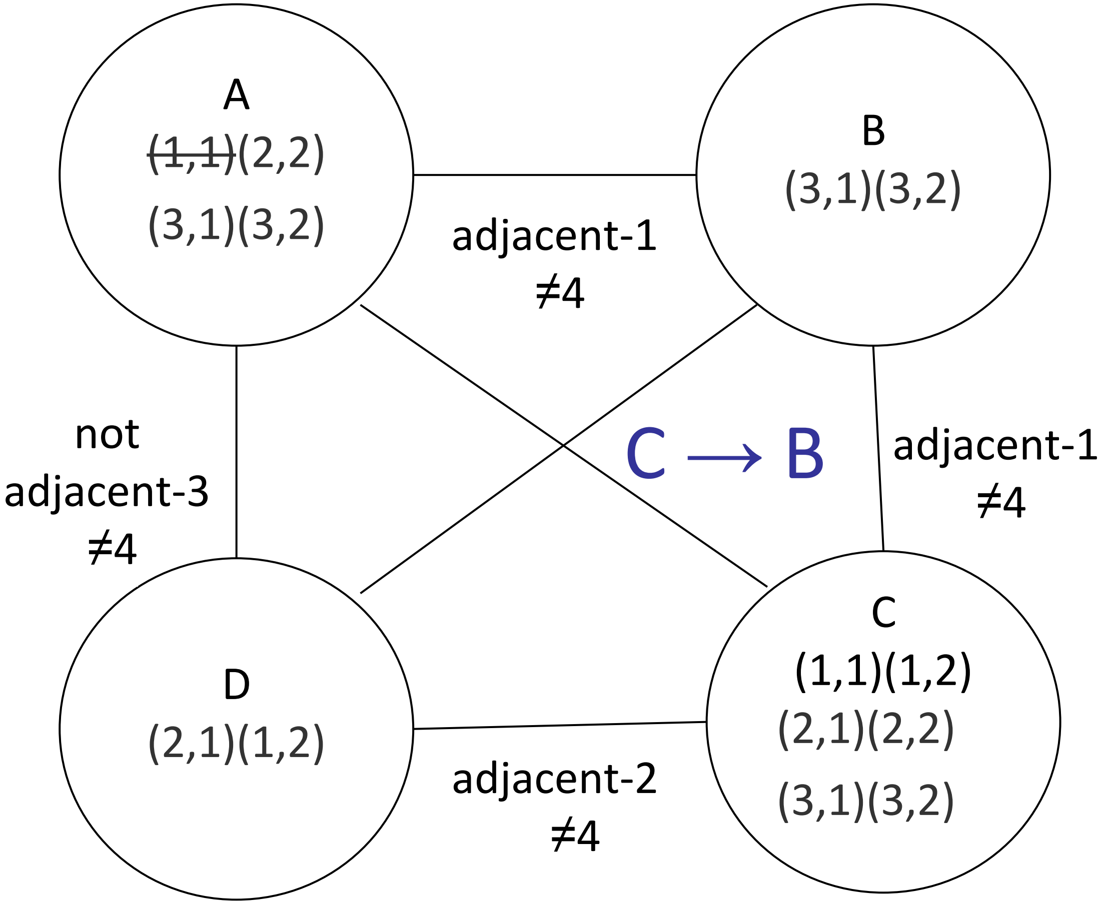


An arc X ⟶ Y is consistent iff for *every* x in the tail there is *some allowed* y in the head which could be assigned without violating a constraint.

For A in (1, 1): no value for B that satisfies constraint: Remove (1, 1) from A's domain

CSP

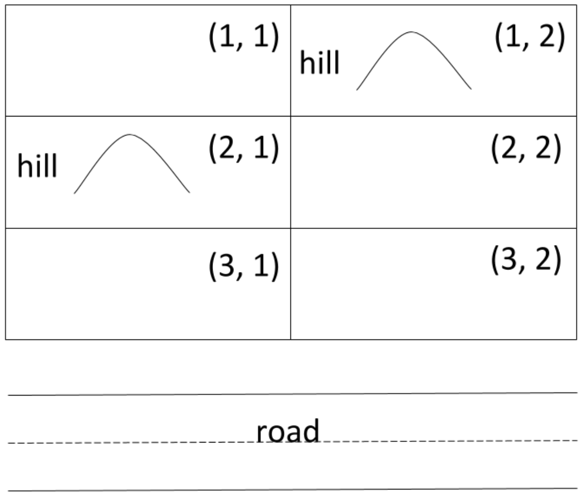


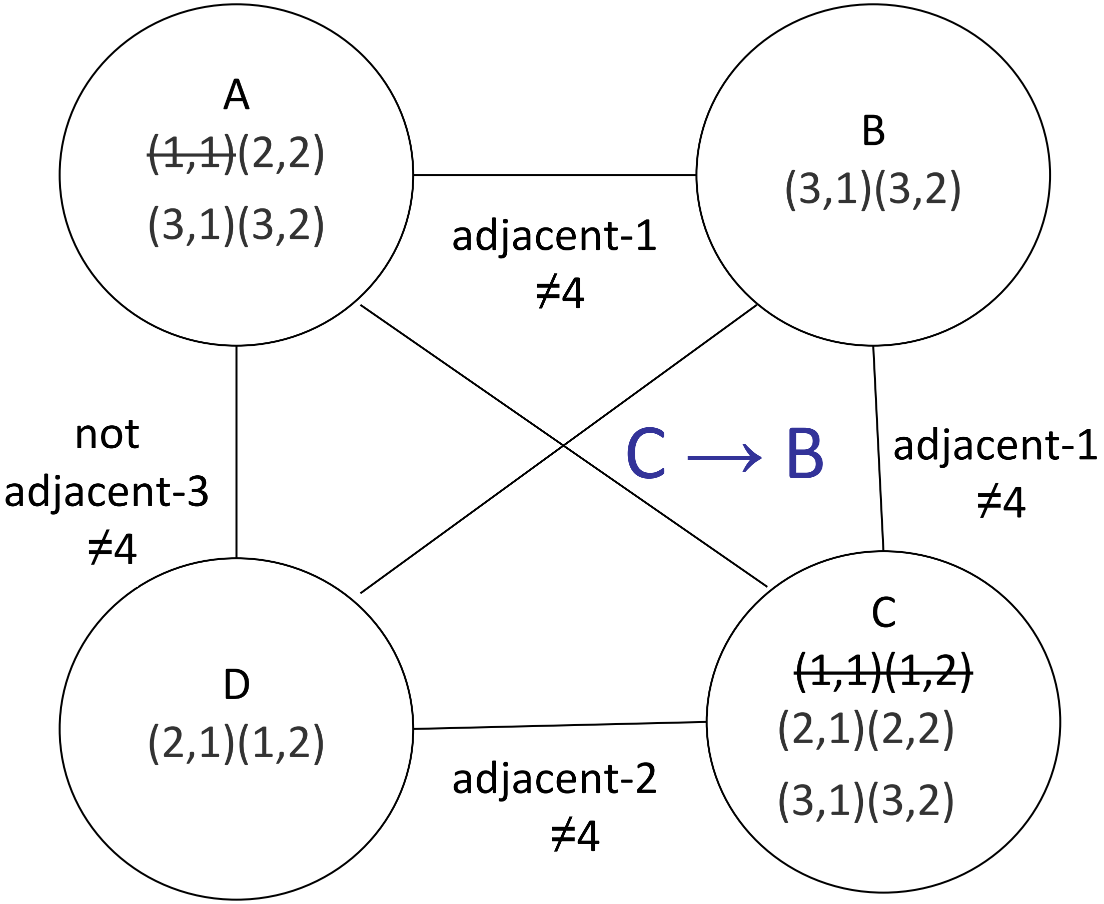


An arc X ⟶ Y is consistent iff for *every* x in the tail there is *some allowed* y in the head which could be assigned without violating a constraint.

Remove (1, 1) and (1, 2) from C's domain

CSP





An arc X ⟶ Y is consistent iff for *every* x in the tail there is *some allowed* y in the head which could be assigned without violating a constraint.

Remove (1, 1) and (1, 2) from C's domain

CSP - AC-3

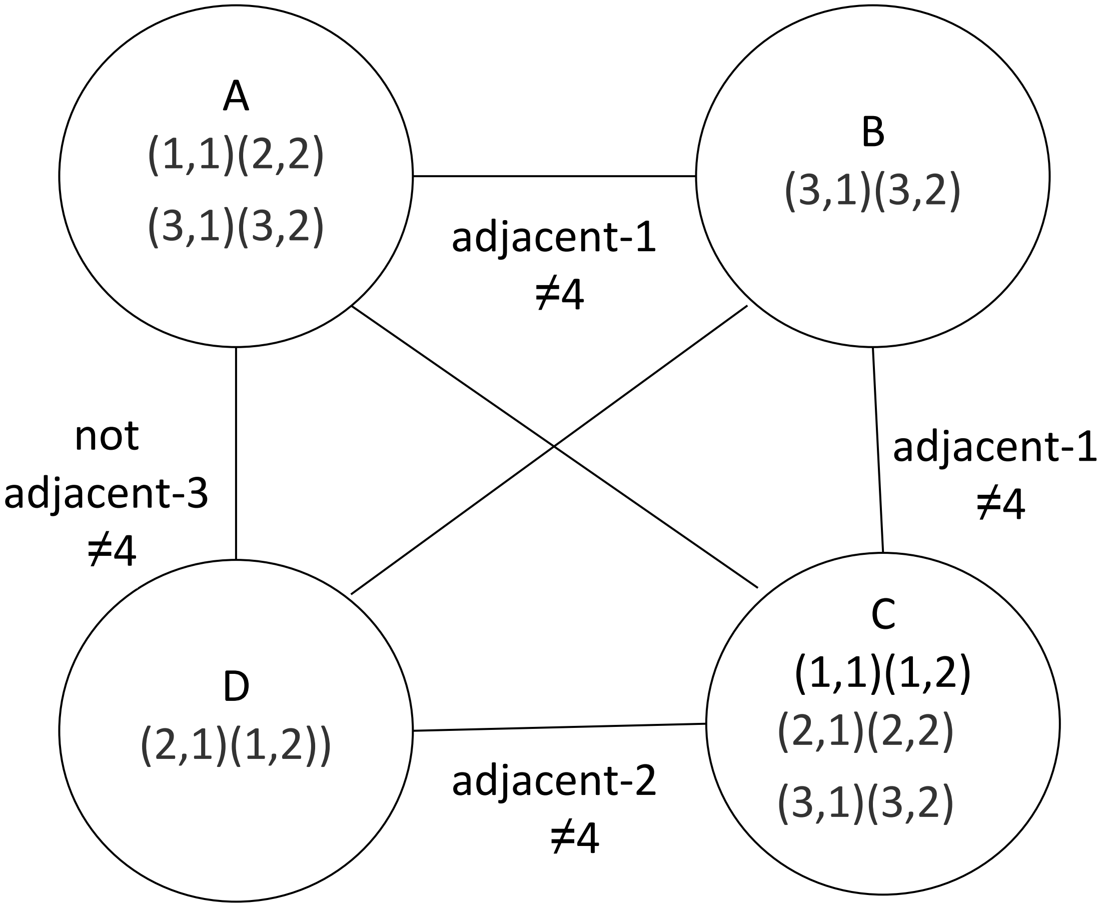
```
function AC-3( csp) returns the CSP, possibly with reduced domains
   inputs: csp, a binary CSP with variables {X1, X2, ..., Xn}
   local variables: queue, a queue of arcs, initially all the arcs in csp

   while queue is not empty do
      (Xi, Xj) ← REMOVE-FIRST(queue)
      if REMOVE-INCONSISTENT-VALUES(Xi, Xj) then
         for each Xk in NEIGHBORS[Xi] do
            add (Xk, Xi) to queue
```

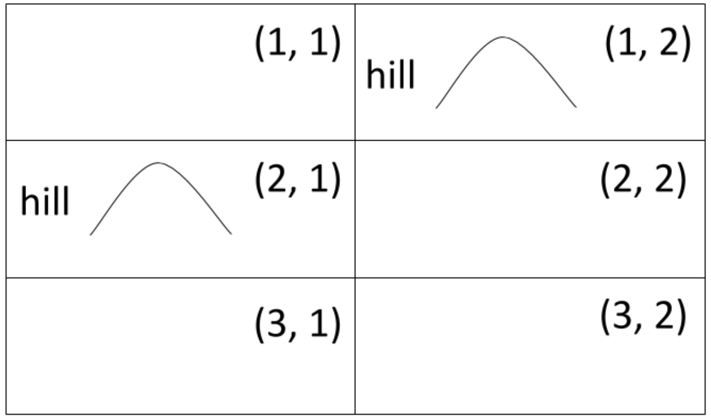
```
function REMOVE-INCONSISTENT-VALUES( Xi, Xj) returns true iff succeeds
   removed ← false
   for each x in DOMAIN[Xi] do
      if no value y in DOMAIN[Xj] allows (x,y) to satisfy the constraint Xi ↔ Xj
         then delete x from DOMAIN[Xi];  removed ← true
   return removed
```

CSP

Queue:

A⟶B, B⟶A, B⟶C, C⟶B, D⟶C, C⟶D, A⟶D,
D⟶A, A⟶C, C⟶A, B⟶D, D⟶B





CSP

Queue:

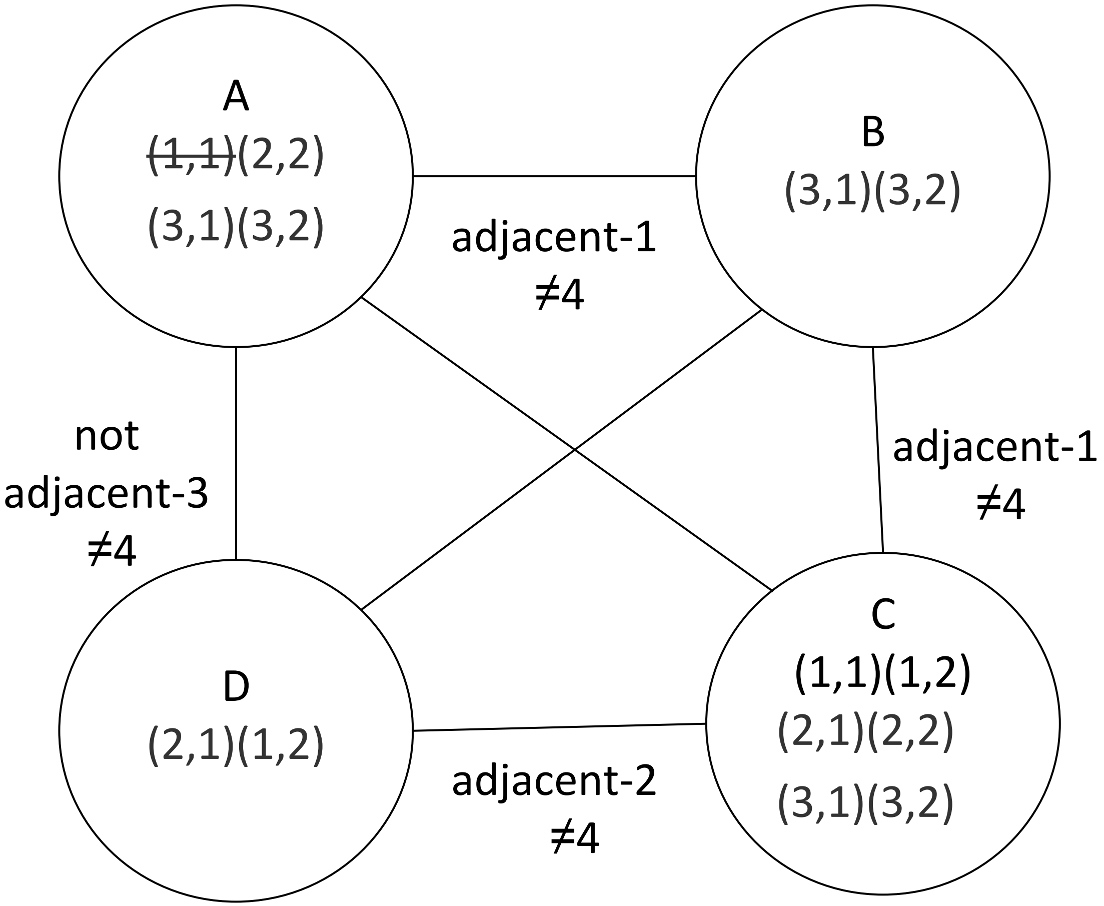
A⟶B, B⟶A, B⟶C, C⟶B, D⟶C, C⟶D, A⟶D, D⟶A, A⟶C, C⟶A, B⟶D, D⟶B

Enforce A⟶B

Remove (1, 1) from domain.

Add arcs B⟶A, D⟶A, C⟶A (already in queue)

(1, 1)	hill (1, 2)
hill (2, 1)	(2, 2)
(3, 1)	(3, 2)

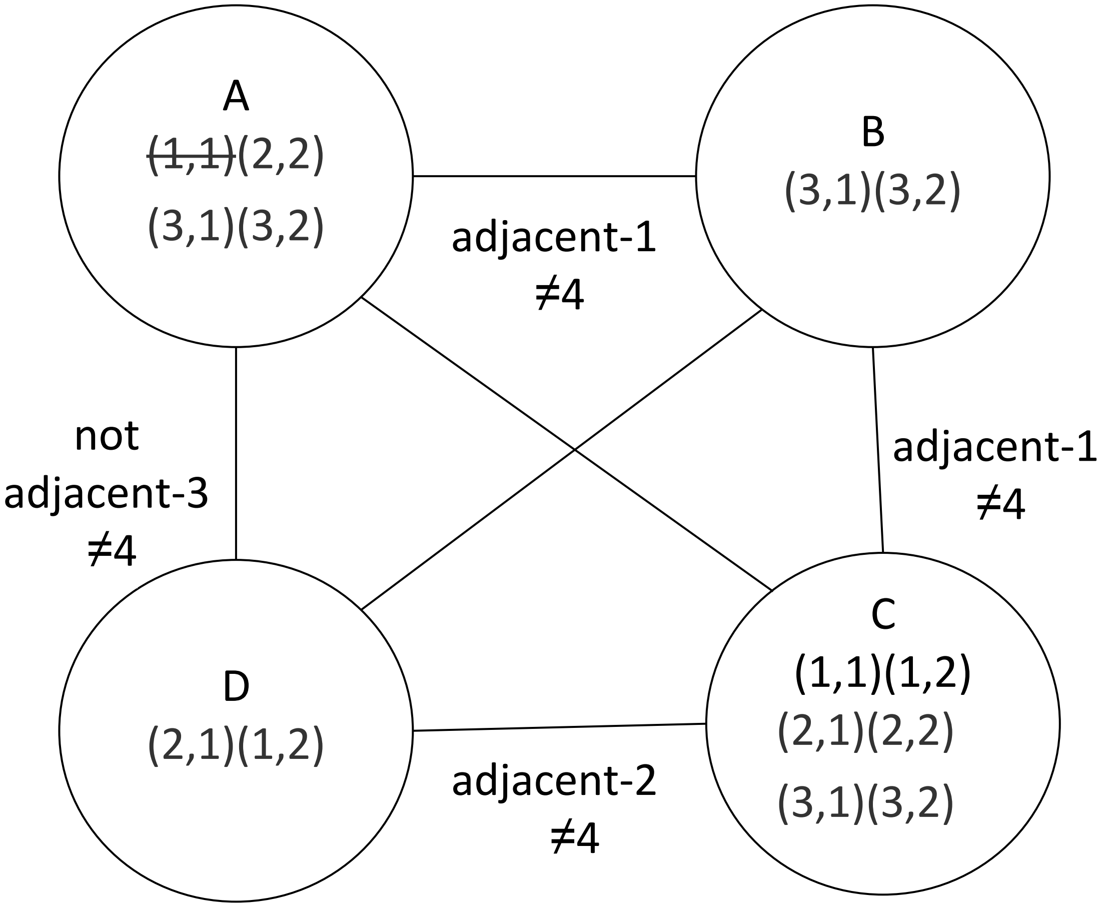


CSP

Queue:

B⟶A, B⟶C, C⟶B, D⟶C, C⟶D, A⟶D,D⟶A, A⟶C, C⟶A, B⟶D, D⟶B

Enforce B⟶A ✓



(1, 1)	hill (1, 2)
hill (2, 1)	(2, 2)
(3, 1)	(3, 2)

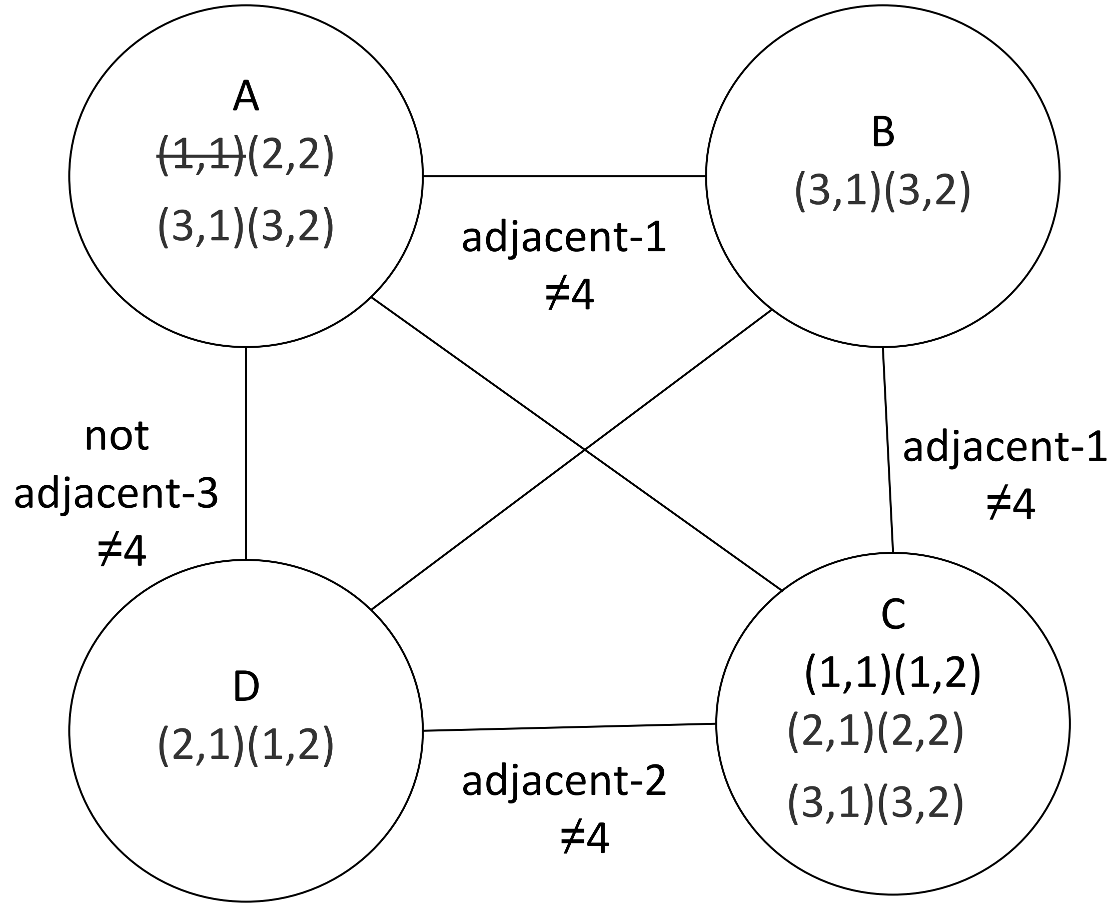
CSP

Queue:

B⟶C, C⟶B, D⟶C, C⟶D, A⟶D,D⟶A, A⟶C, C⟶A, B⟶D, D⟶B

Enforce B⟶C ✓

(1, 1)	hill (1, 2)
hill (2, 1)	(2, 2)
(3, 1)	(3, 2)



CSP

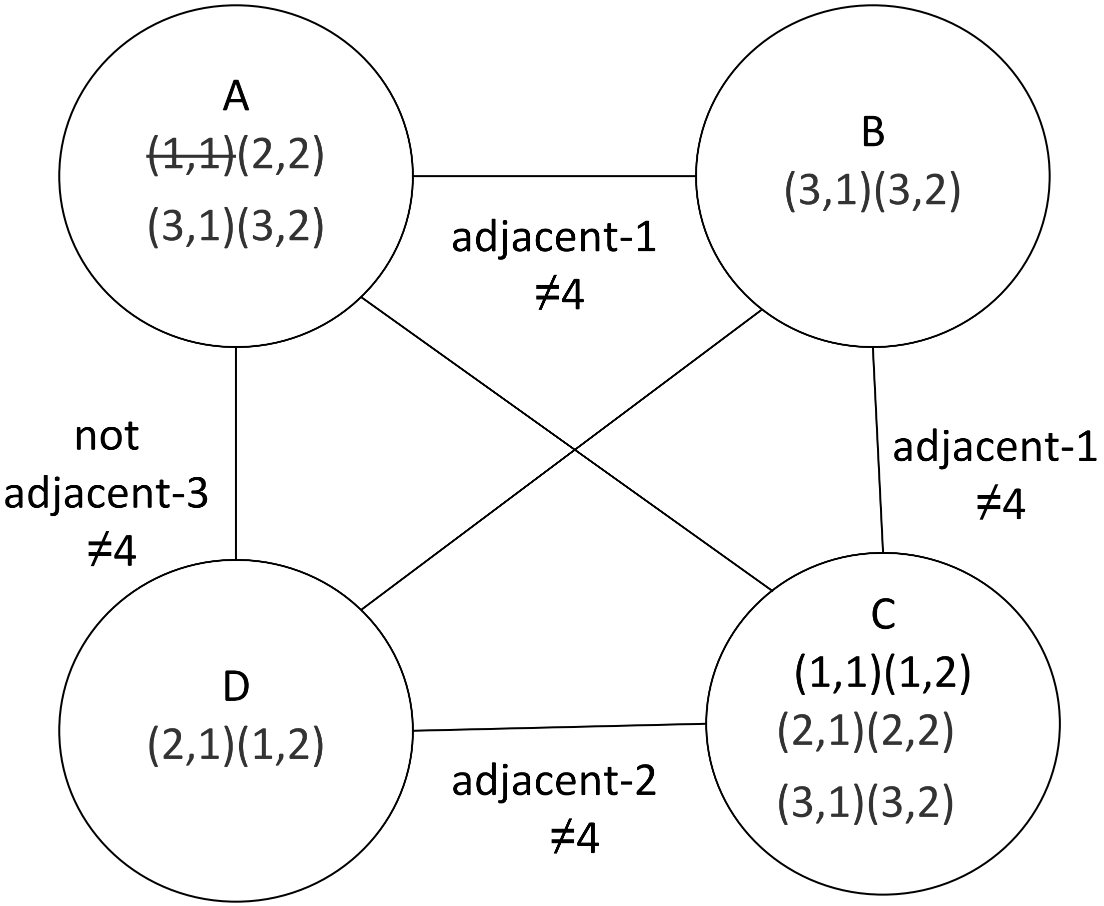
Queue:

C⟶B, D⟶C, C⟶D, A⟶D,D⟶A, A⟶C, C⟶A, B⟶D, D⟶B

Enforce C⟶B

Remove (1, 1) and (1, 2) from C's domain

(1, 1)	hill (1, 2)
hill (2, 1)	(2, 2)
(3, 1)	(3, 2)



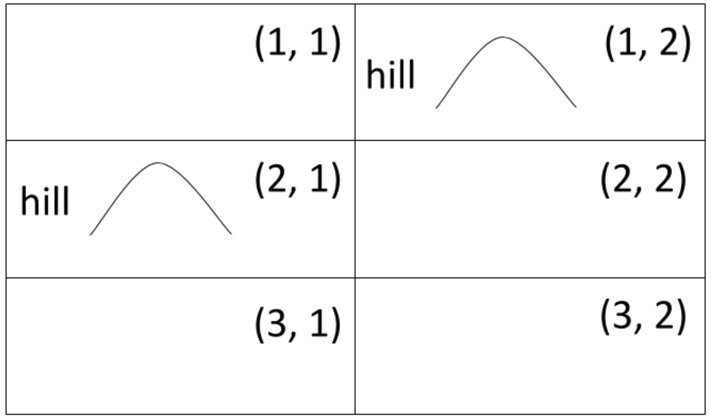
CSP

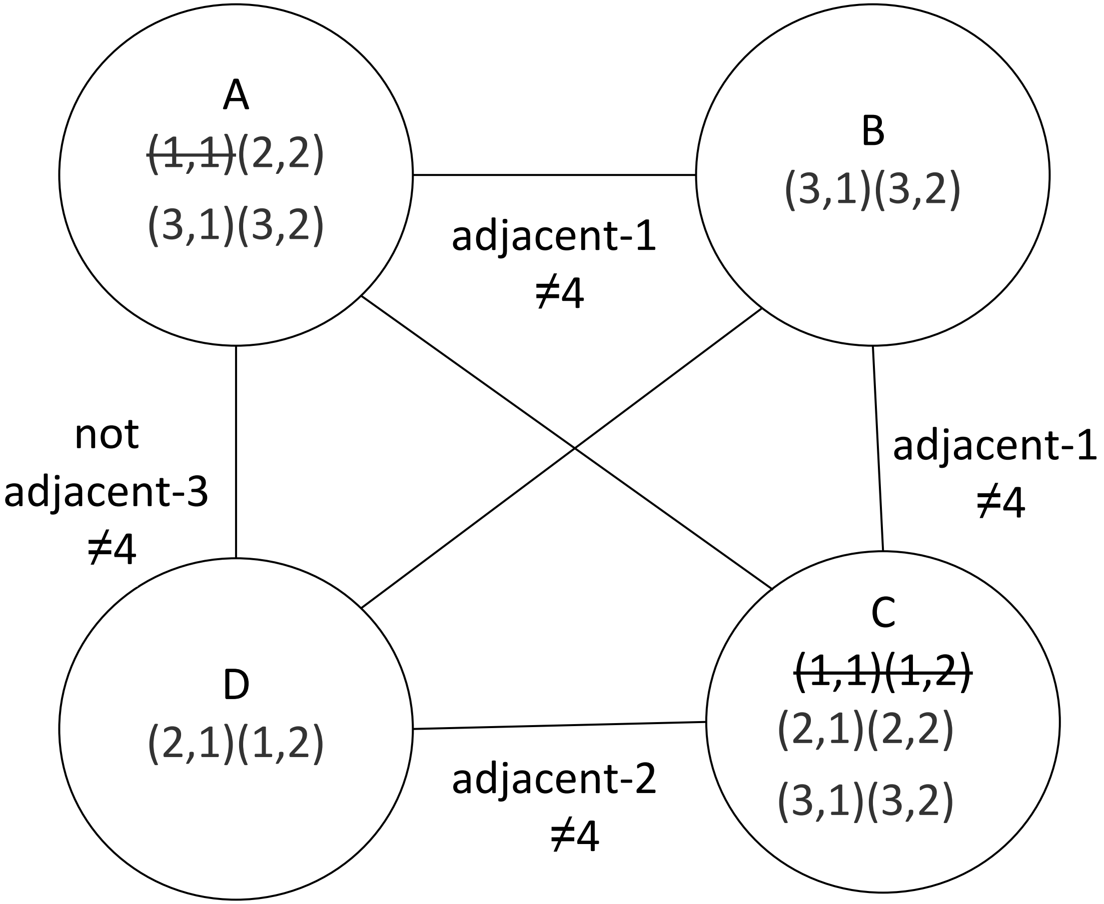
Queue:

C⟶B, D⟶C, C⟶D, A⟶D,D⟶A, A⟶C, C⟶A, B⟶D, D⟶B

Enforce C⟶B

Remove (1, 1) and (1, 2) from C's domain





CSP

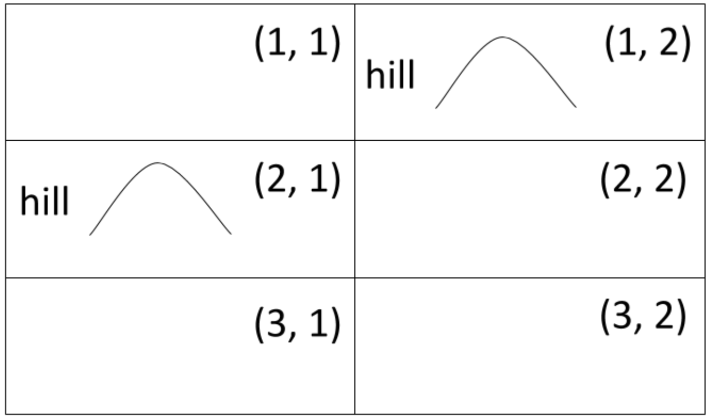
Queue:

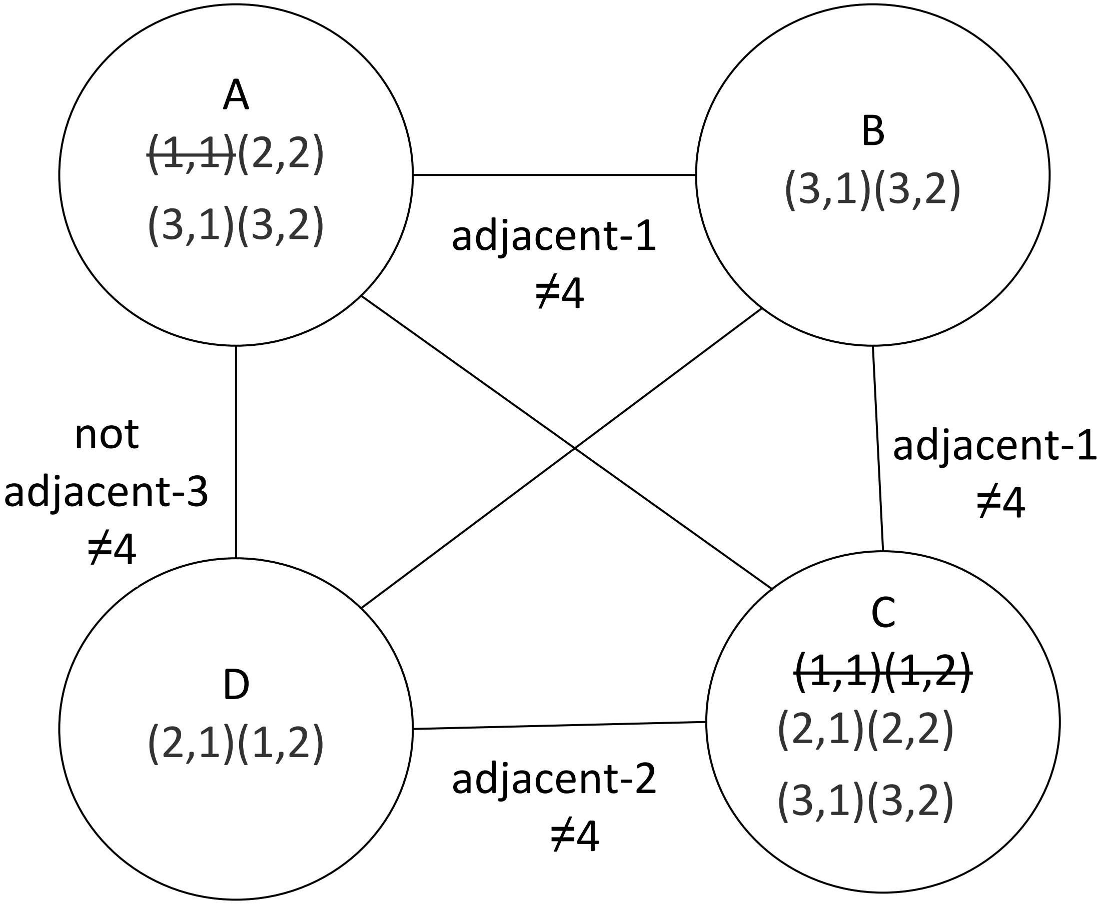
C⟶B, D⟶C, C⟶D, A⟶D,D⟶A, B⟶C, A⟶C, C⟶A, B⟶D, D⟶B

Enforce C⟶B

Remove (1, 1) and (1, 2) from C's domain

Add arcs B⟶C, A⟶C and D⟶C





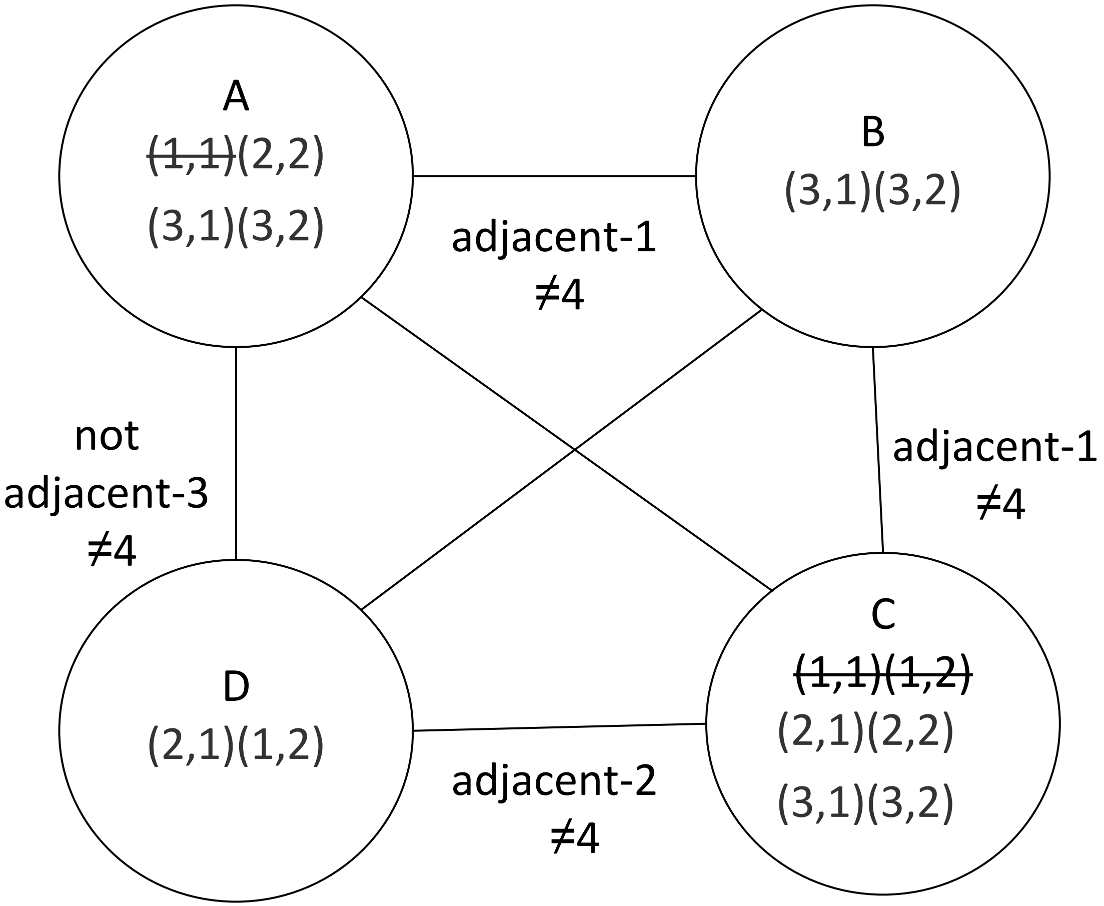
CSP

Queue:

D→C, C→D, A→D,D→A, B→C, A→C, C→A, B→D, D→B

Enforce D→C

(1, 1)	hill (1, 2)
hill (2, 1)	(2, 2)
(3, 1)	(3, 2)



CSP

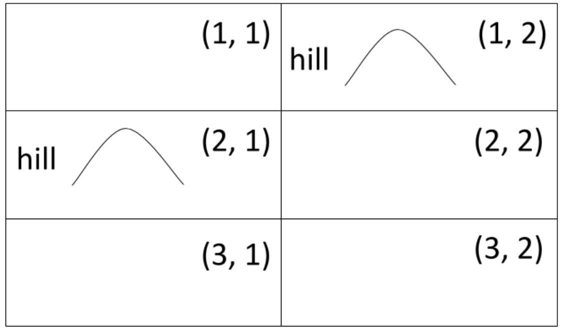
Queue:

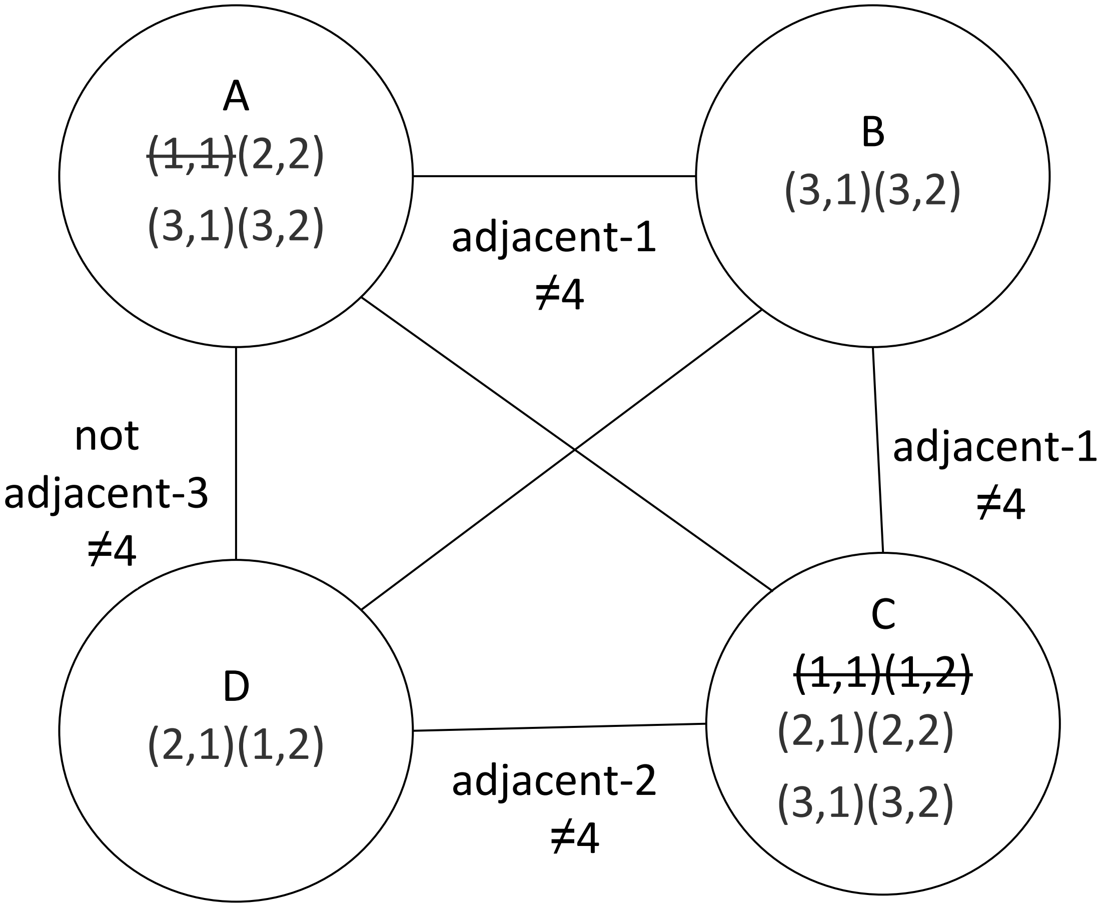
C⟶D, A⟶D,D⟶A, B⟶C, A⟶C, C⟶A, B⟶D, D⟶B, D⟶C

Enforce C⟶D

Remove (2, 1) and (3,2) from C's domain

Add B⟶C, A⟶C and D⟶C to queue





CSP

Queue:

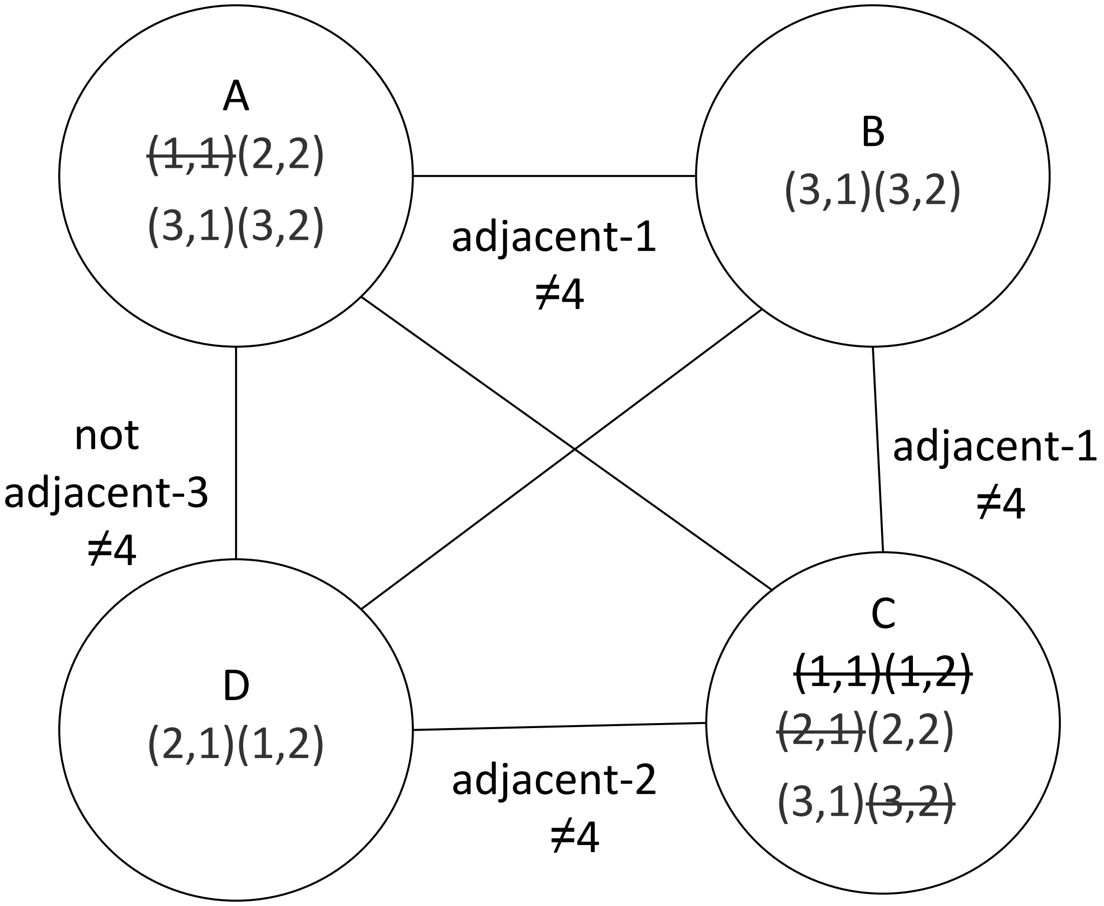
C⟶D, A⟶D,D⟶A, B⟶C, A⟶C, C⟶A, B⟶D, D⟶B, D⟶C

Enforce C⟶D

Remove (2, 1) and (3,2) from C's domain

Add B⟶C, A⟶C and D⟶C to queue

(1, 1)	hill (1, 2)
hill (2, 1)	(2, 2)
(3, 1)	(3, 2)



CSP

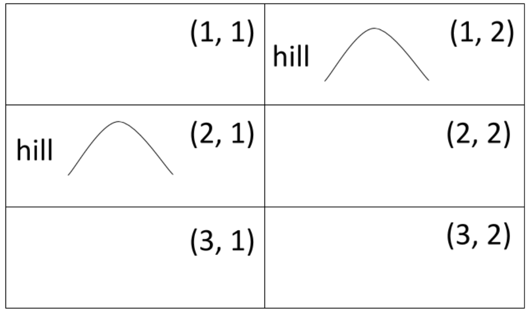
Queue:

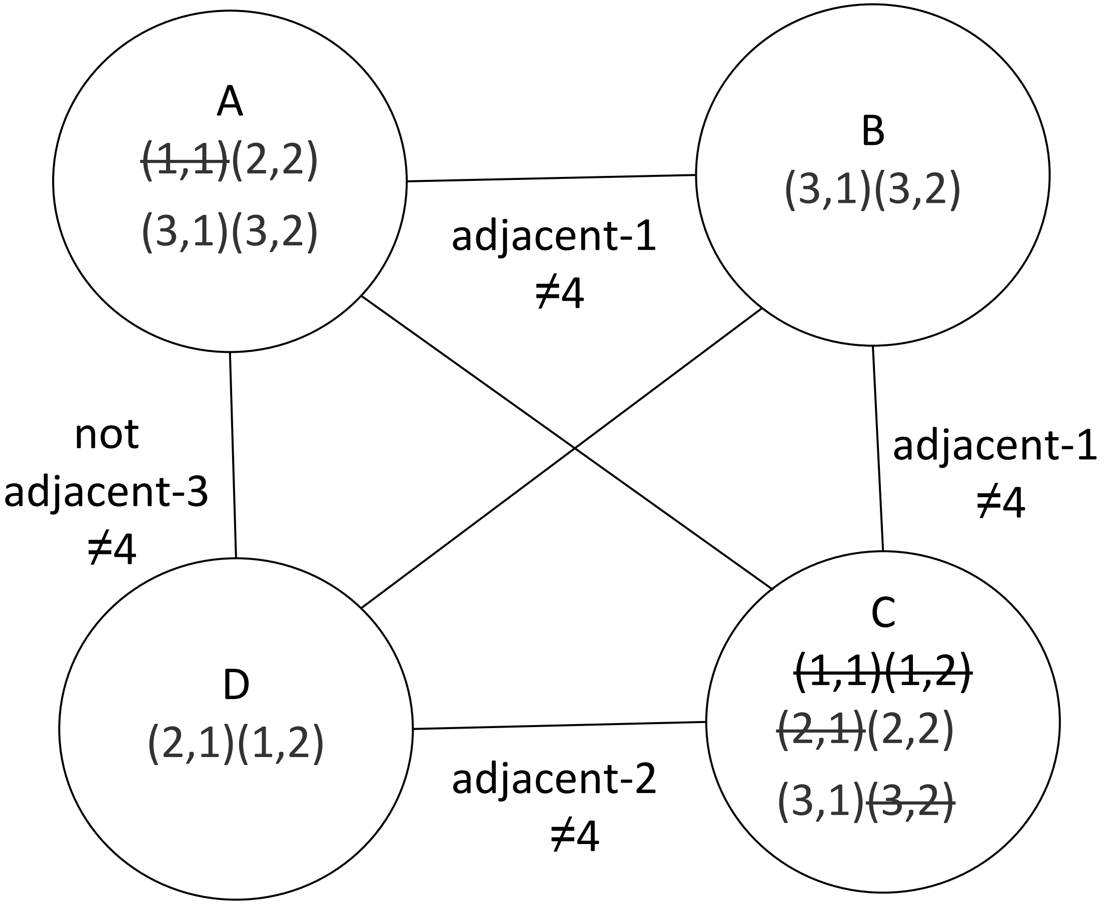
A⟶D,D⟶A, B⟶C, A⟶C, C⟶A, B⟶D, D⟶B, D⟶C

Enforce A⟶D

Remove (2, 2) from A's domain

Add D⟶A, C⟶A and B⟶A to queue





CSP

Queue:

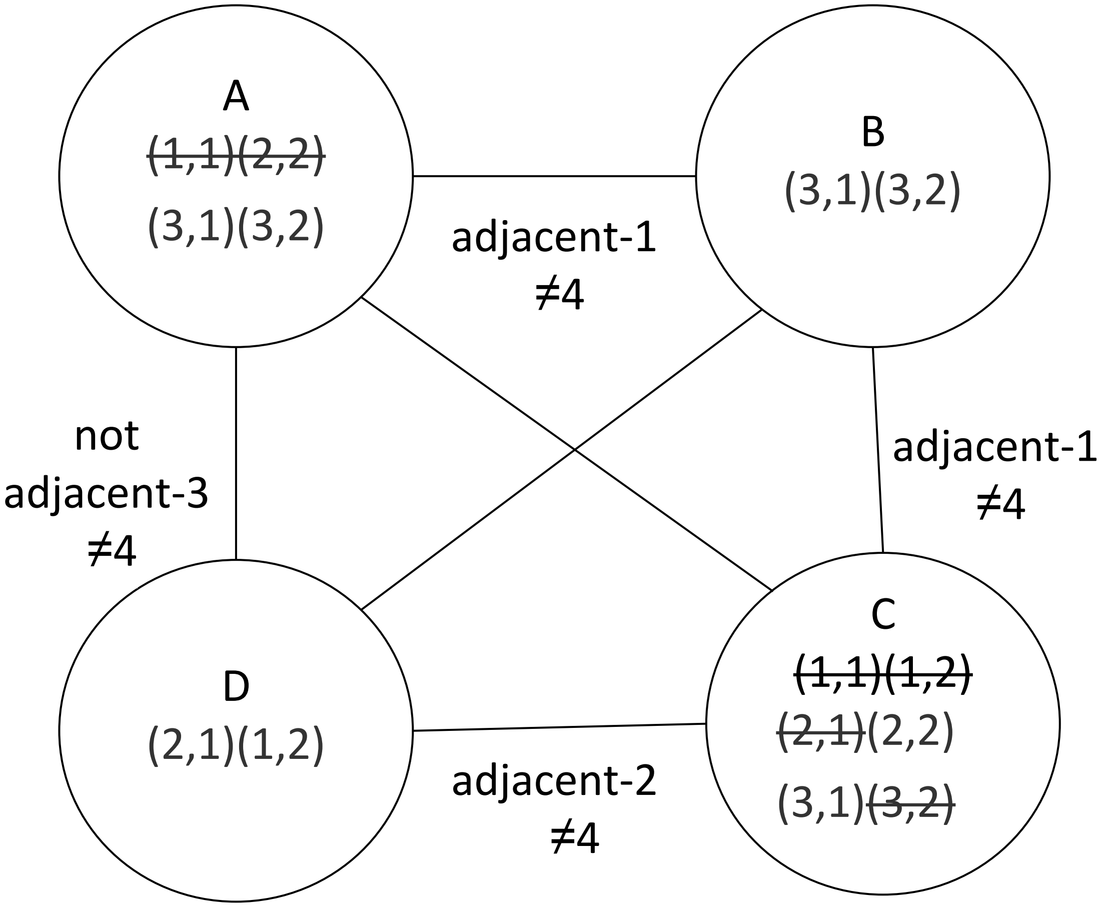
A⟶D,D⟶A, B⟶C, A⟶C, C⟶A, B⟶D, D⟶B, D⟶C

Enforce A⟶D

Remove (2, 2) from A's domain

Add D⟶A, C⟶A and B⟶A to queue

(1, 1)	hill (1, 2)
hill (2, 1)	(2, 2)
(3, 1)	(3, 2)



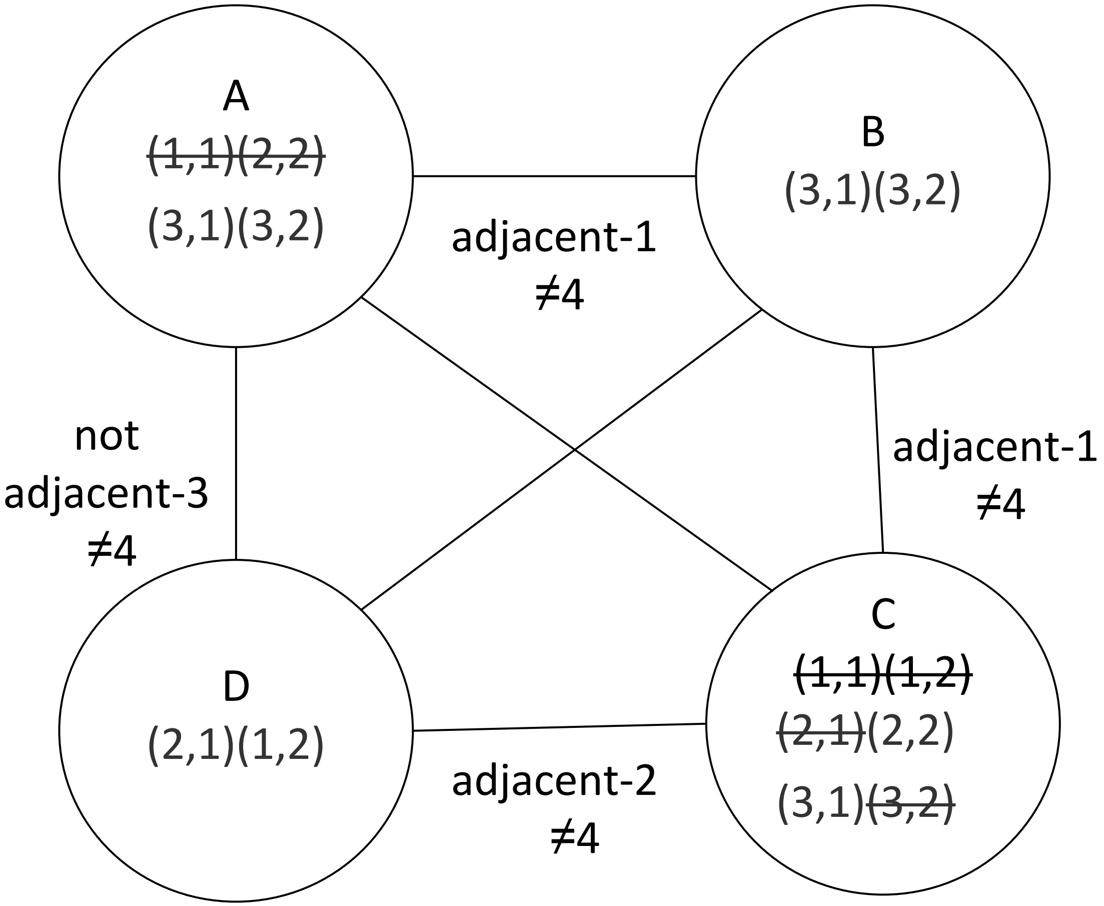
CSP

Queue:

D⟶A, B⟶C, A⟶C, C⟶A, B⟶D, D⟶B, D⟶C, B⟶A

Enforce D⟶A

(1, 1)	hill (1, 2)
hill (2, 1)	(2, 2)
(3, 1)	(3, 2)



CSP

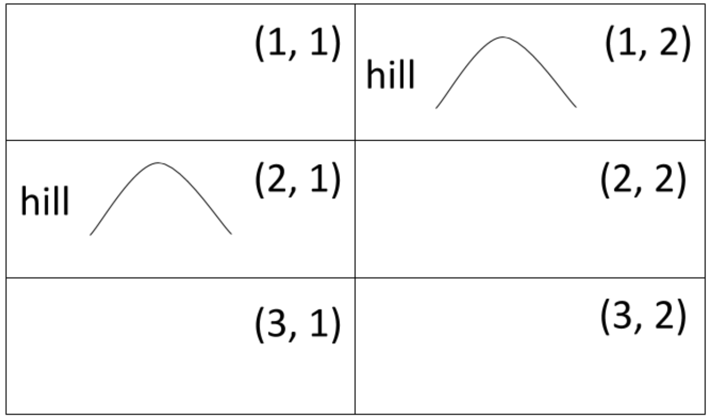
Queue:

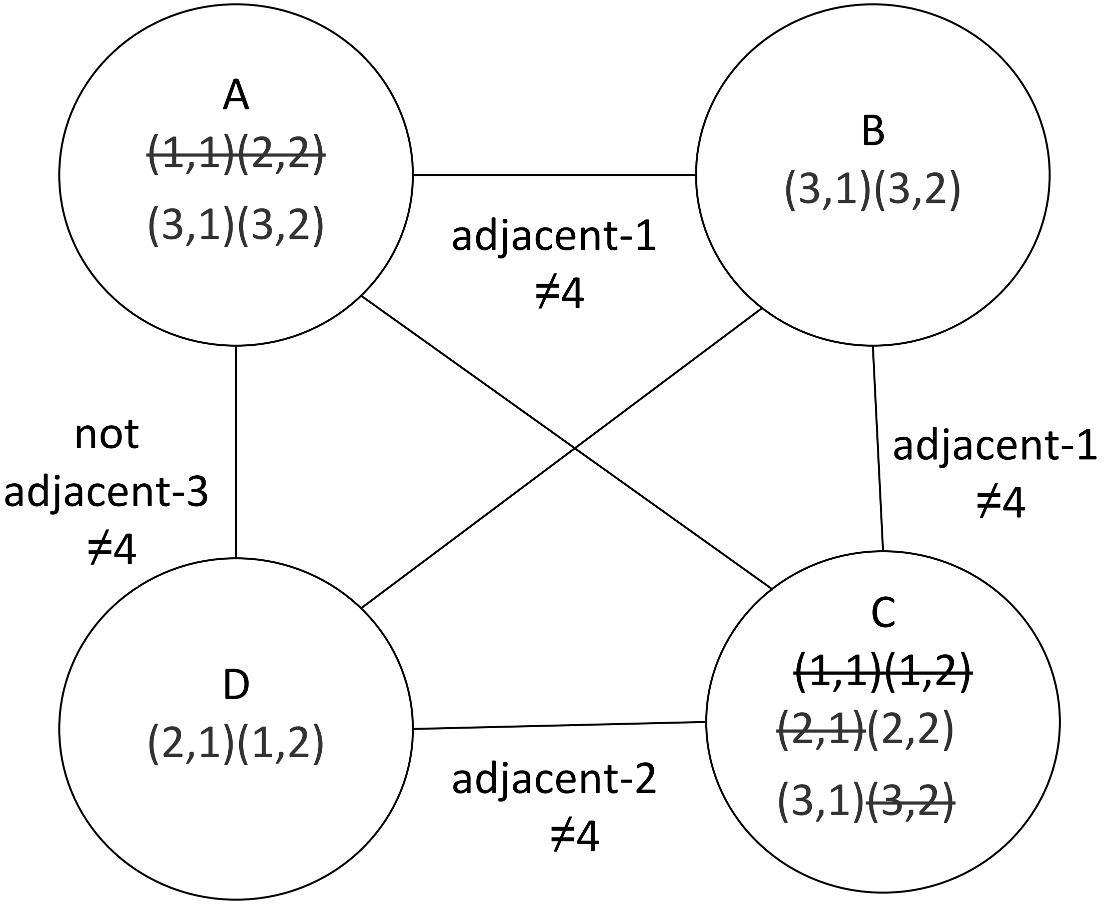
B⟶C, A⟶C, C⟶A, B⟶D, D⟶B, D⟶C, B⟶A

Enforce B⟶C

Remove (3, 1) from B's domain

Add A⟶B, D⟶B and C⟶B





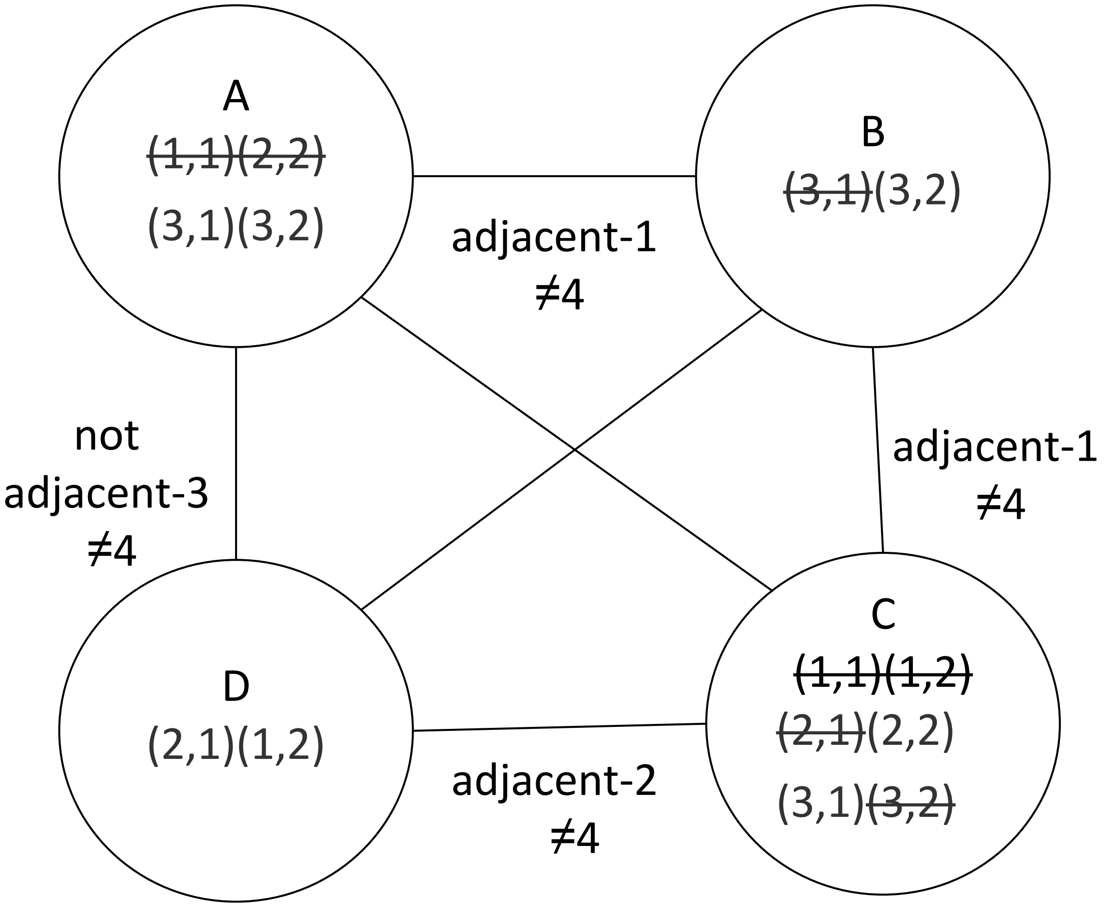
CSP

Queue:

A⟶C, C⟶A, B⟶D, D⟶B, D⟶C, B⟶A, A⟶B, C⟶B

Enforce A⟶C, C⟶A, B⟶D, D⟶B

(1, 1)	hill (1, 2)
hill (2, 1)	(2, 2)
(3, 1)	(3, 2)



CSP

Queue:

D⟶C, B⟶A, A⟶B, C⟶B

We keep going until the queue is empty