Introduction

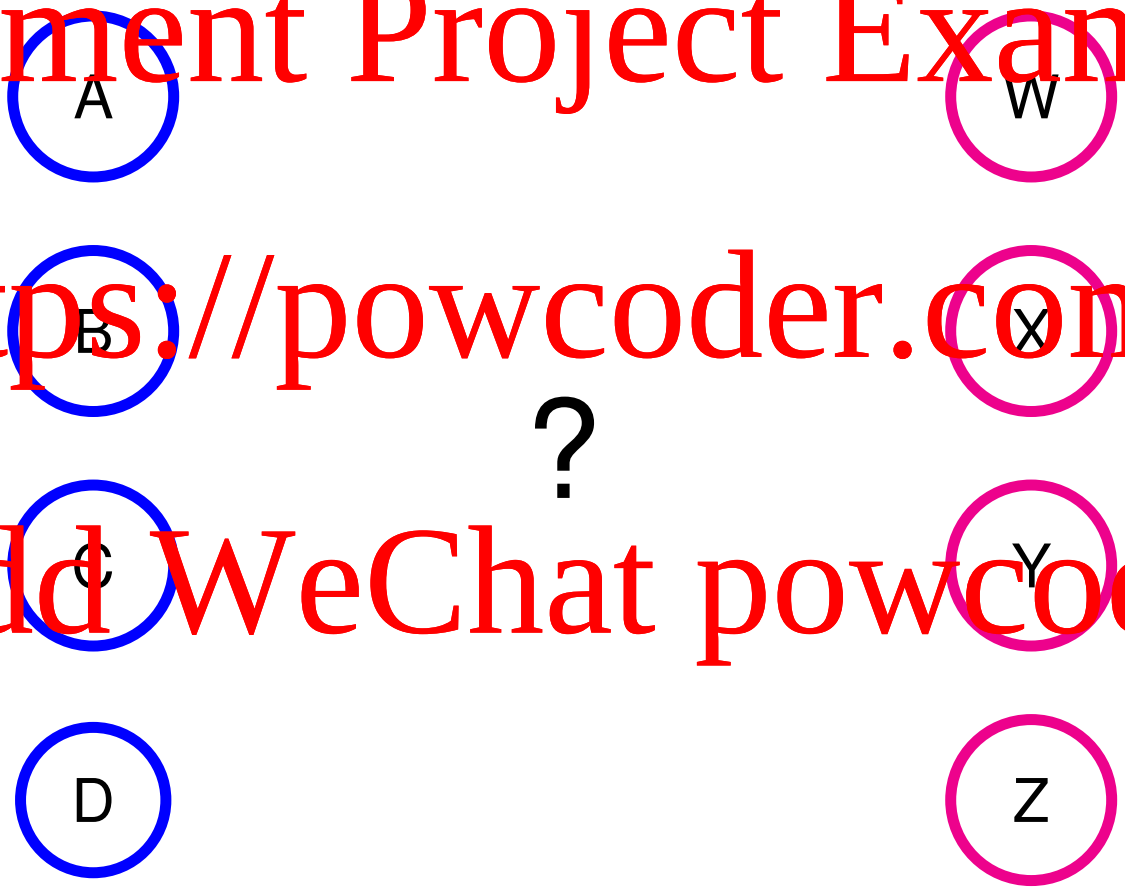
Illustrative Problem

Use the Stable Matching problem to illustrate key concepts:

- What is a problem?
- What is an algorithm?
- What is a measure of progress?
- How do we analyse an algorithm's running time?
- How do we prove that an algorithm is correct?

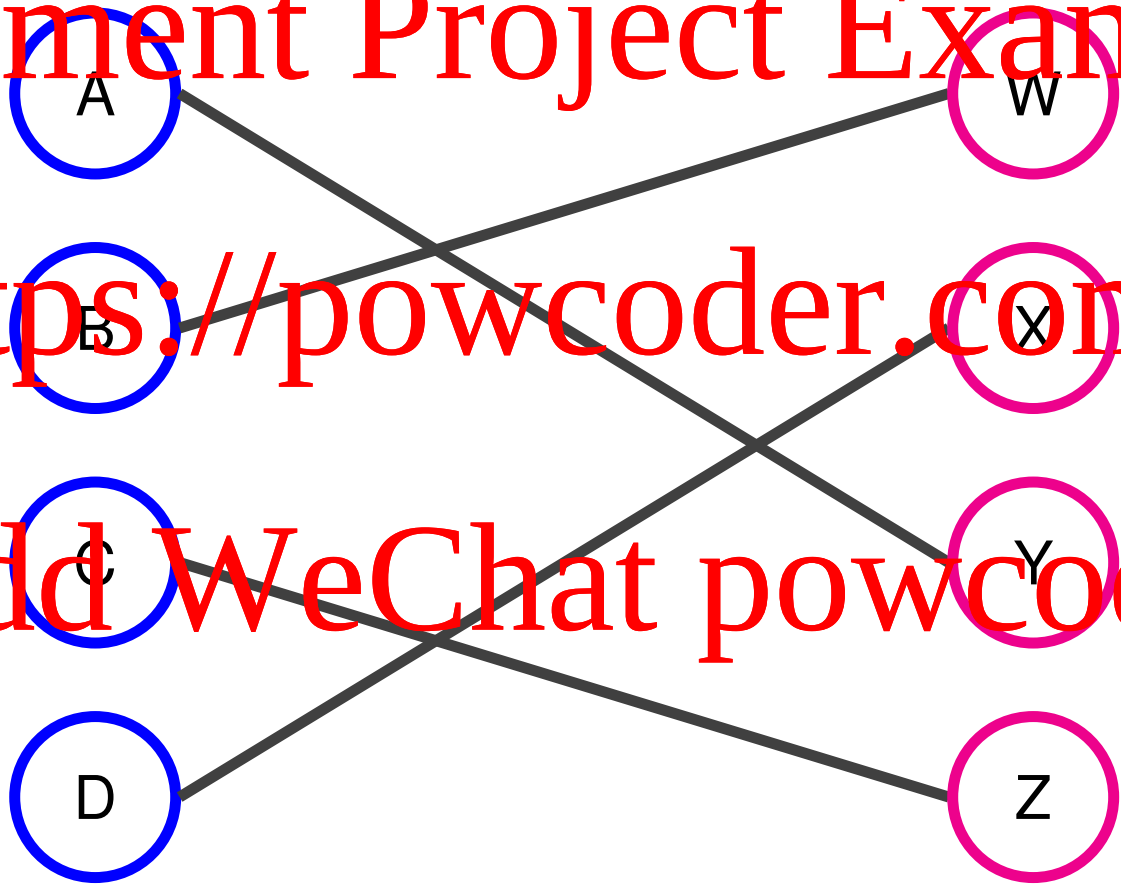
Problem to Consider

Matching people up.



Perfect Match

Everyone paired with exactly one person



Preferences

blue people's preferences

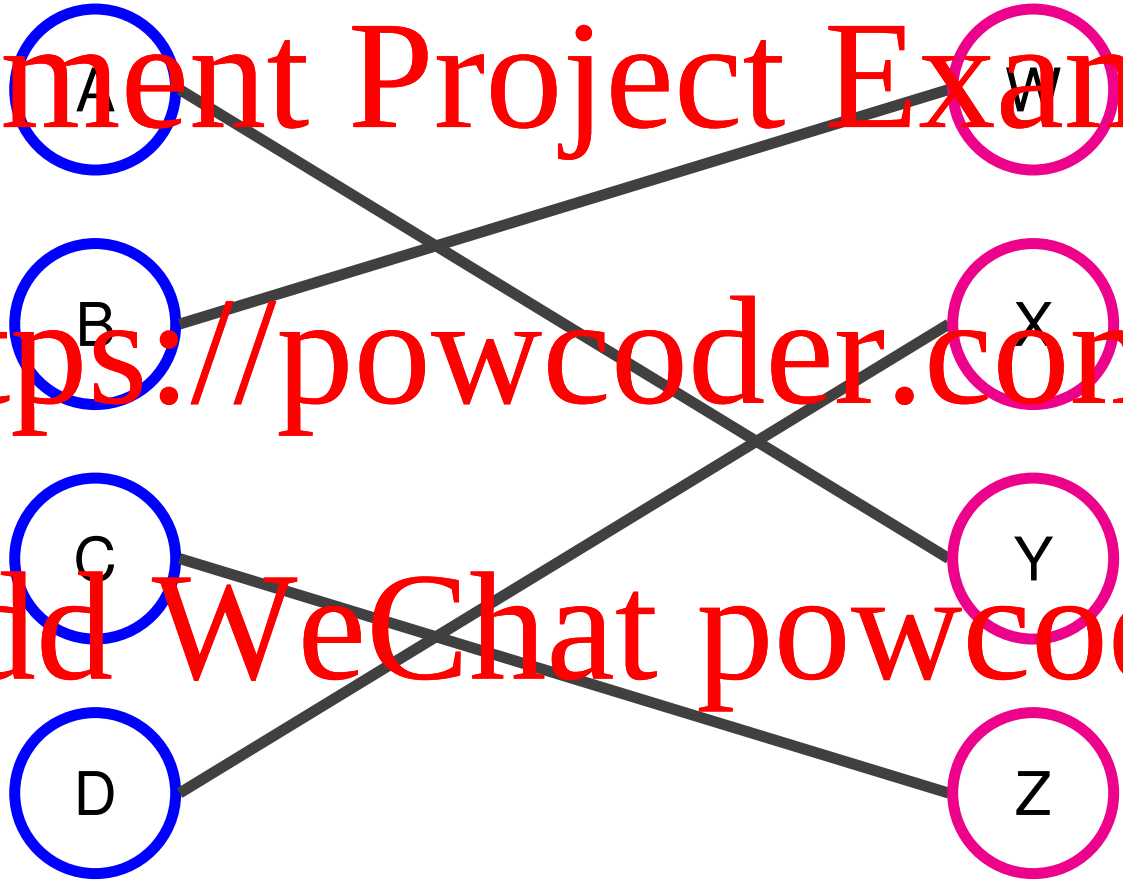
	1st	2nd	3rd	4th
A	X	Z	W	Y
B	X	Y	W	Z
C	W	X	Y	Z
D	Z	X	Y	W

red people's preferences

	1st	2nd	3rd	4th
W	A	B	D	C
X	C	D	B	A
Y	A	D	B	C
Z	D	A	B	C

Stability

Might some couple elope?



Stability

Need to consider preferences of blue people and red people

A	X	Z	W	Y
B	X	Y	W	Z
C	W	X	Y	Z
D	Z	X	Y	W

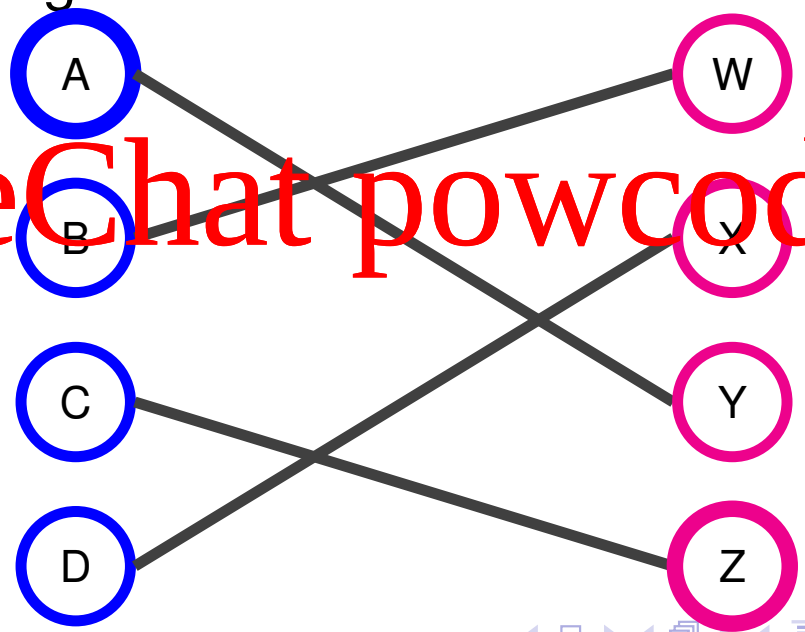
W	A	B	D	C
X	C	D	B	A
Y	A	D	B	C
Z	D	A	B	C

An Instability

- Consider A and Z
- A has been matched with Y
- Z has been matched with C
- A prefers Z to Y
- Z prefers A to C
- Both A and Z would prefer to be matched
- Instability in the matching

A	X	Z	W	Y
B	X	Y	W	Z
C	W	X	Y	Z
D	Z	X	Y	W

W	A	B	D	C
X	C	D	B	A
Y	A	D	B	C
Z	D	A	B	C

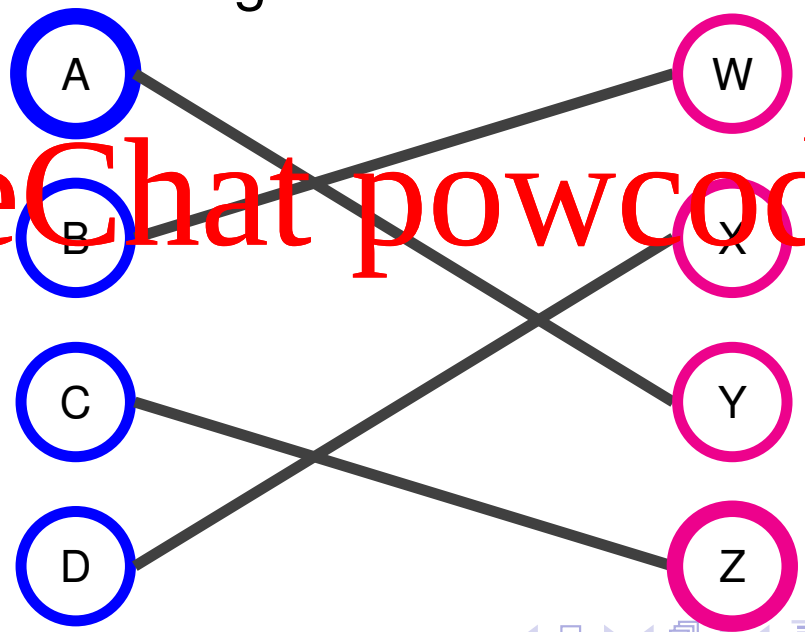


It Takes Two to Tango

- Consider C and Y
- C has been matched with Z
- Y has been matched with A
- C prefers Y to Z
- *but* Y prefers A to C
- Only C would prefer the match with Y
- Not an instability in the matching

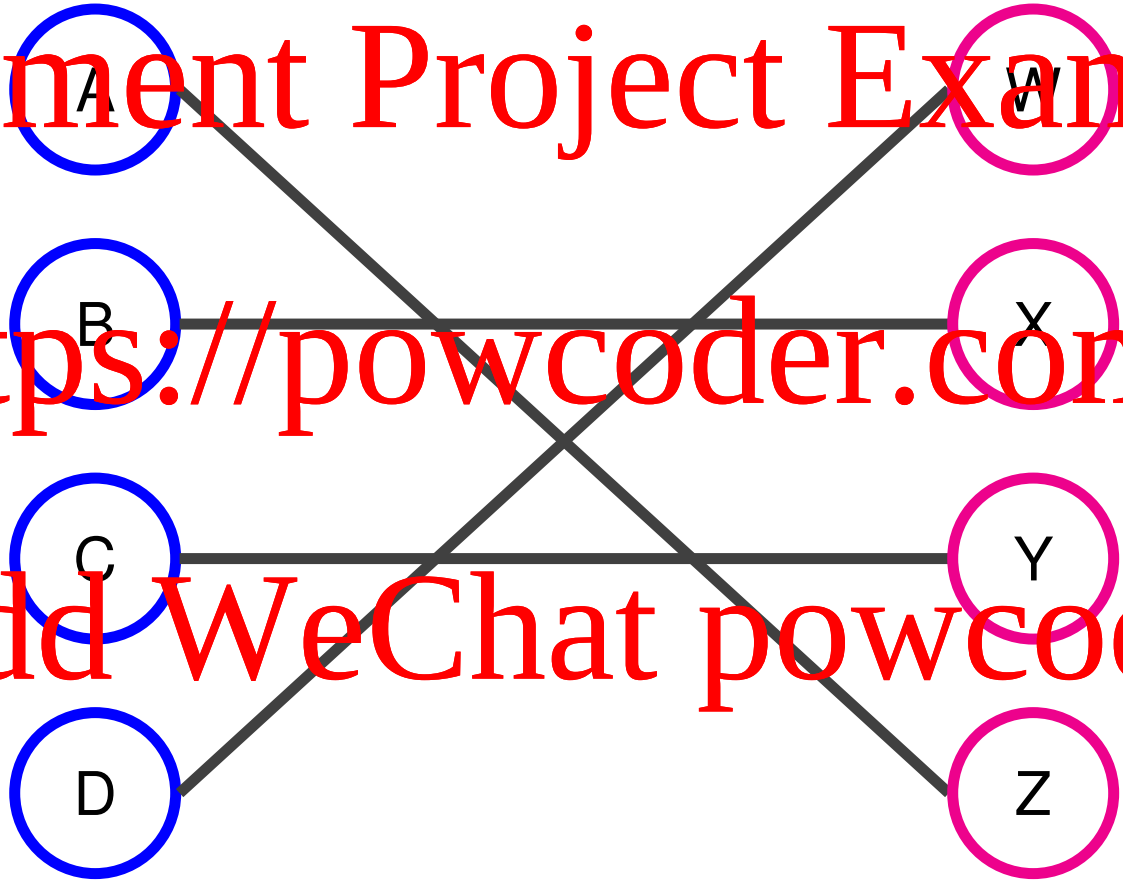
A	X	Z	W	Y
B	X	Y	W	Z
C	W	X	Y	Z
D	Z	X	Y	W

W	A	B	D	C
X	C	D	B	A
Y	A	D	B	C
Z	D	A	B	C



Another Matching

Is it stable?



Preferences

A	Y	Z	W	Y
B	X	Y	W	Z
C	W	X	Y	Z
D	Z	X	Y	W

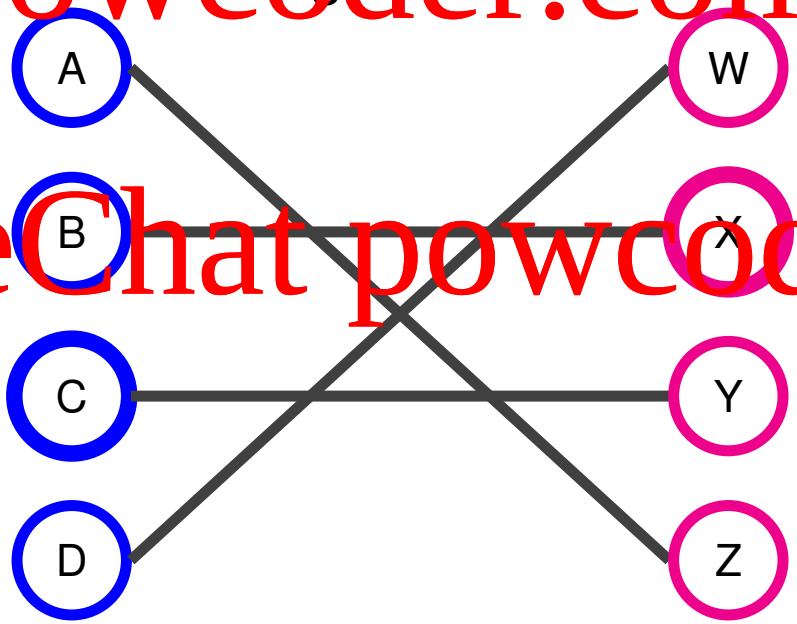
W	A	B	D	C
X	C	D	B	A
Y	A	D	B	C
Z	D	A	B	C

Any Instabilities?

- A pair where both prefer each other to the assigned match
- What about C and X?
- C would prefer W or X to Y
- W does *not* like C!
- X however would prefer C to her match B
- This is an instability in the matching

A	X	Z	W	Y
B	X	Y	W	Z
C	W	X	Y	Z
D	Z	X	Y	W

W	A	B	D	C
X	C	D	B	A
Y	A	D	B	C
Z	D	A	B	C



Definitions

- A matching is **perfect** if everyone is paired with exactly one person
- A perfect matching is **stable** if there are no instabilities

Questions

- Can a stable matching always be found?
- How can we construct a stable match when one exists?
- **Stable Match Problem**
- Applies to other situations:
  — matching hospitals and patients
  — matching employers and applicants

What a Problem is and isn't

What a **problem** is:

- Specification of a relationship between **inputs** and **outputs**
- Usually a **mapping** — one valid output for a given input

What a **problem** isn't:

- Not a matter of **doing** something in a particular way!
- Not procedural

The Stable Match Problem is a fairly generic problem

What an Algorithm is and isn't

What an **algorithm** is:

- Specification as to how to tackle a problem
- Can be abstract
- We will use informal pseudo-code language

What an **algorithm** isn't:

- Not the same as a program
- Programs implement algorithms
- Same algorithm implemented in different languages

Solving a problem means getting valid output for each input given to the algorithm

Question for you

- Write down a specification of the sorting problem.

Question for you

- Write down a specification of the sorting problem.

  Input: *A sequence of values A and an ordering relation r*
  Output: *A permutation of A that is in sorted order according to the relation r*

Question for you

- Write down a specification of the problem that the binary search algorithm solves

Question for you

- Write down a specification of the problem that the binary search algorithm solves

  Input: *A sequence of values A that are ordered and a search item a*

  Output: *true if a is in A, and false otherwise.*

Question for you

- Write down a specification of the Stable Match problem.

Question for you

- Write down a specification of the Stable Match problem.

Input: *Two sets of entities, A and B, together with the preferences for each element of A in the form of a ranking of the elements of B, and the preferences of each element of B in the form of a ranking of the elements of A.*

Output: *A set of pairs S where the first element of the pair is from A and the second element of the pair is from B, such that S is a perfect match and S is stable.*

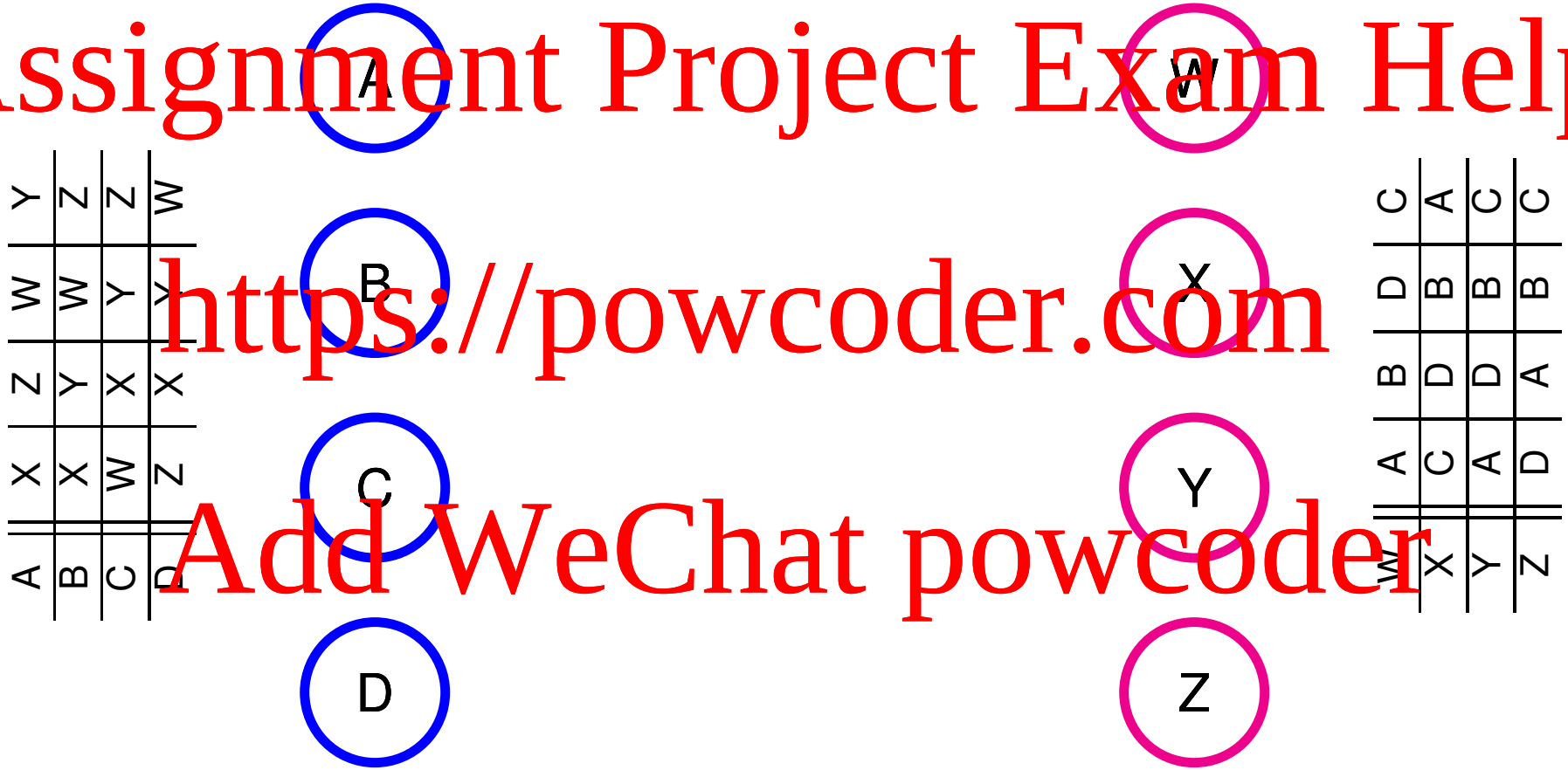
Algorithm

initialize each person to be free
while some blue person is free & has red people to propose to
    choose one such blue person, α
    let β be next best option α's preference list
    if β is free
        assign α and β to be engaged
    else if β prefers α to β's fiancé α'
        assign α and β to be engaged, and α' to be free
    else
        β rejects α

Gale and Shapley, 1962

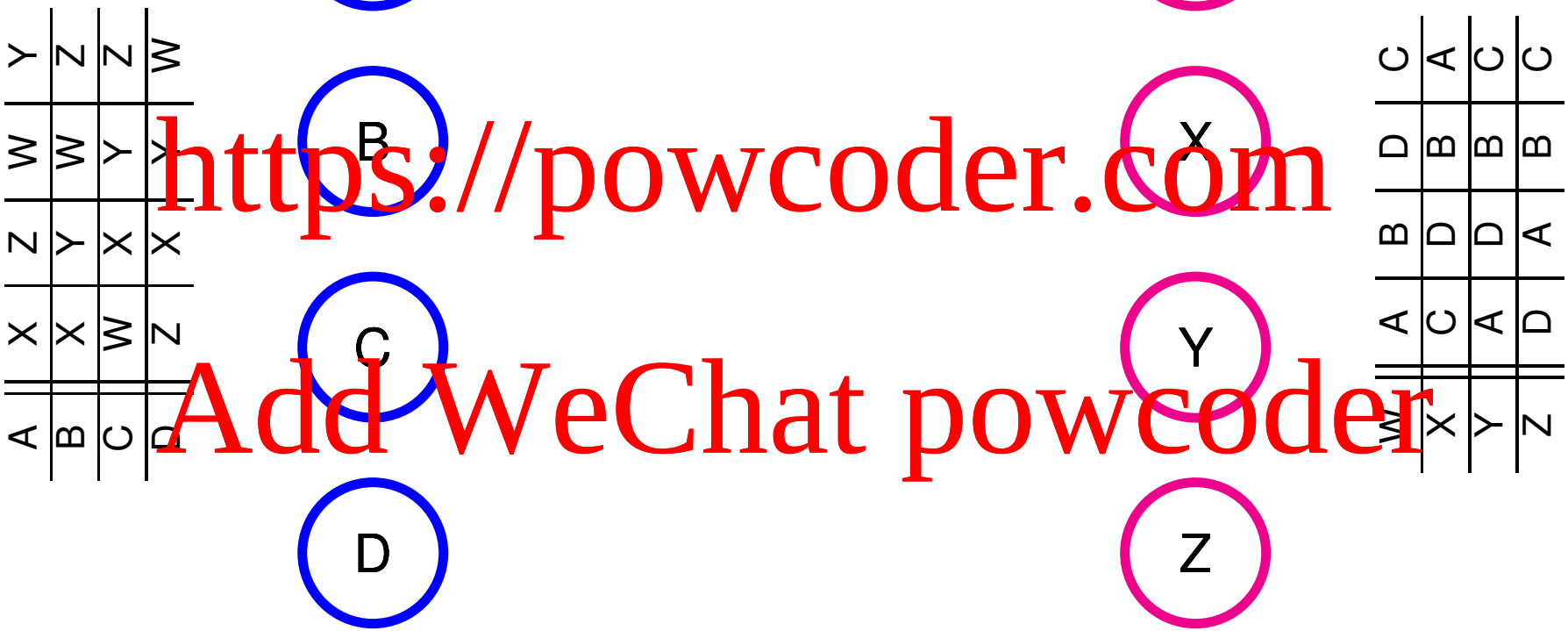
Illustration

Initially, everyone is free



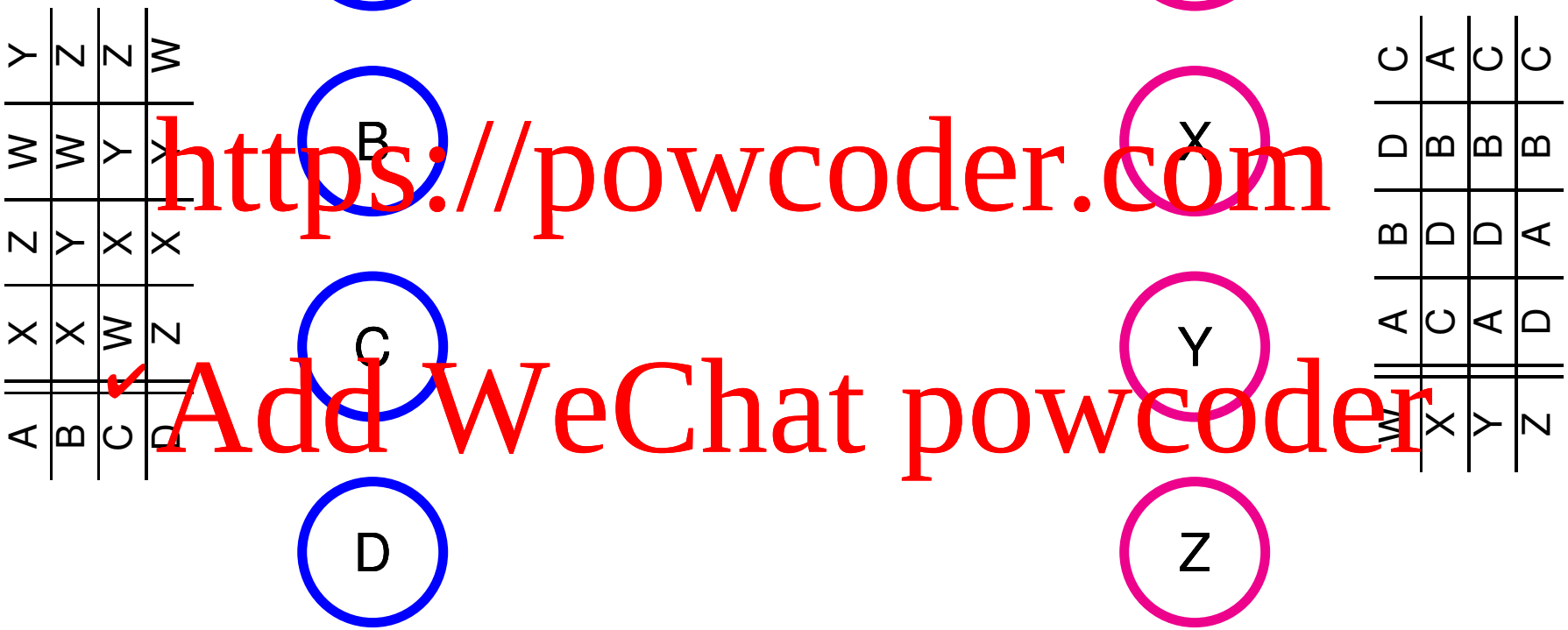
Illustration

Choose $\alpha = \mathsf{C}$



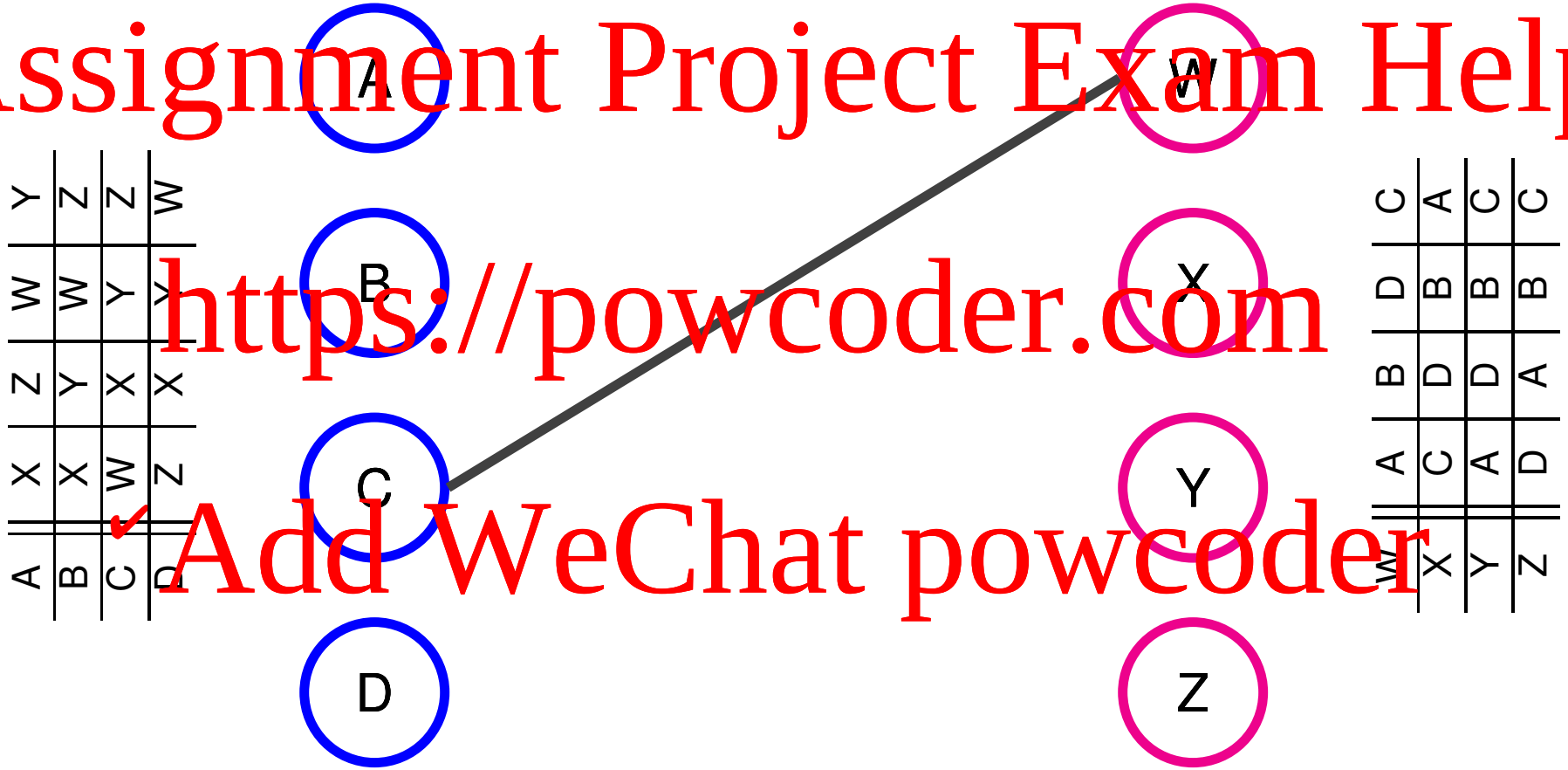
Illustration

$\beta = \mathsf{W}$



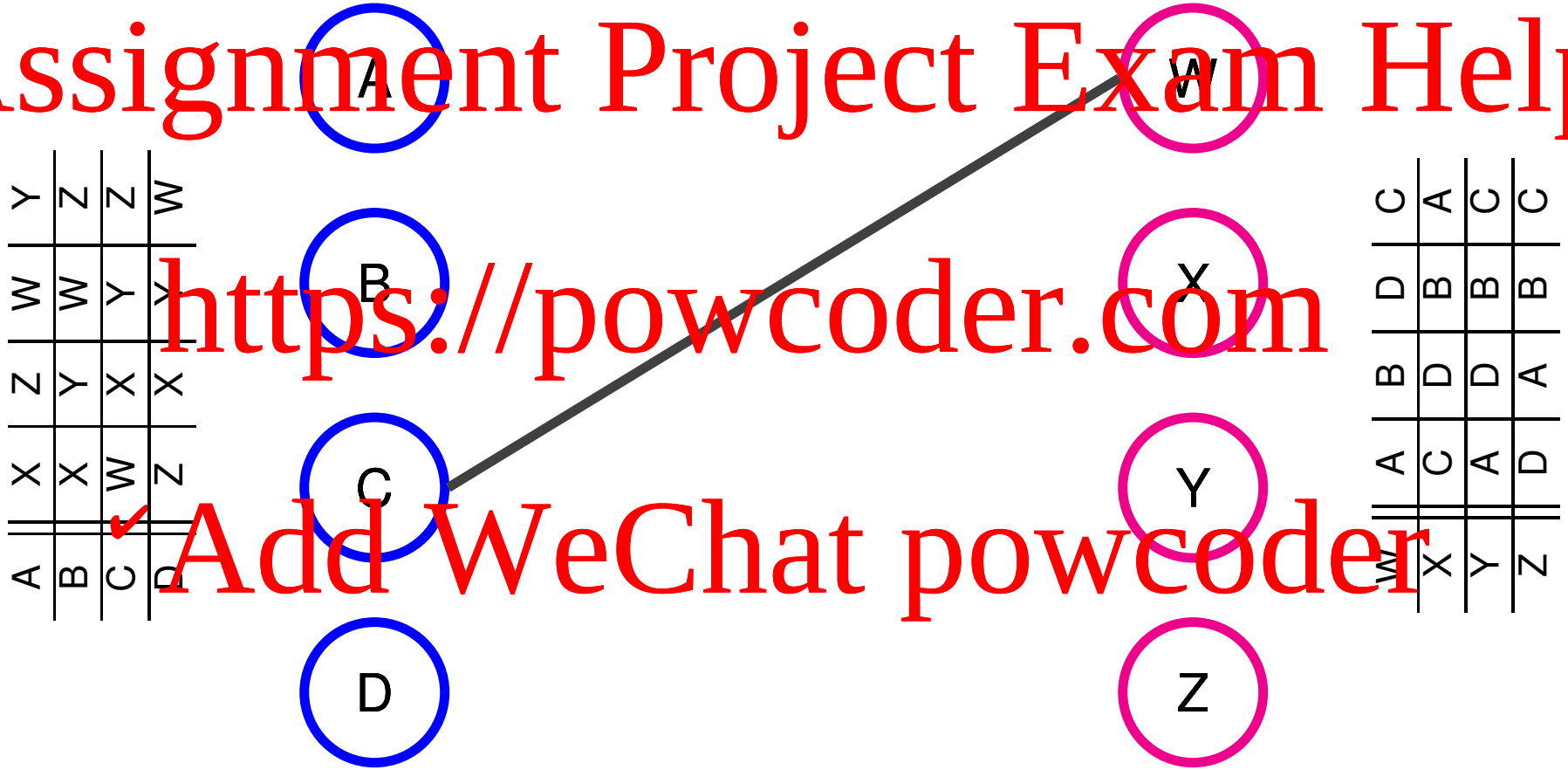
Illustration

W is free so C and W become engaged



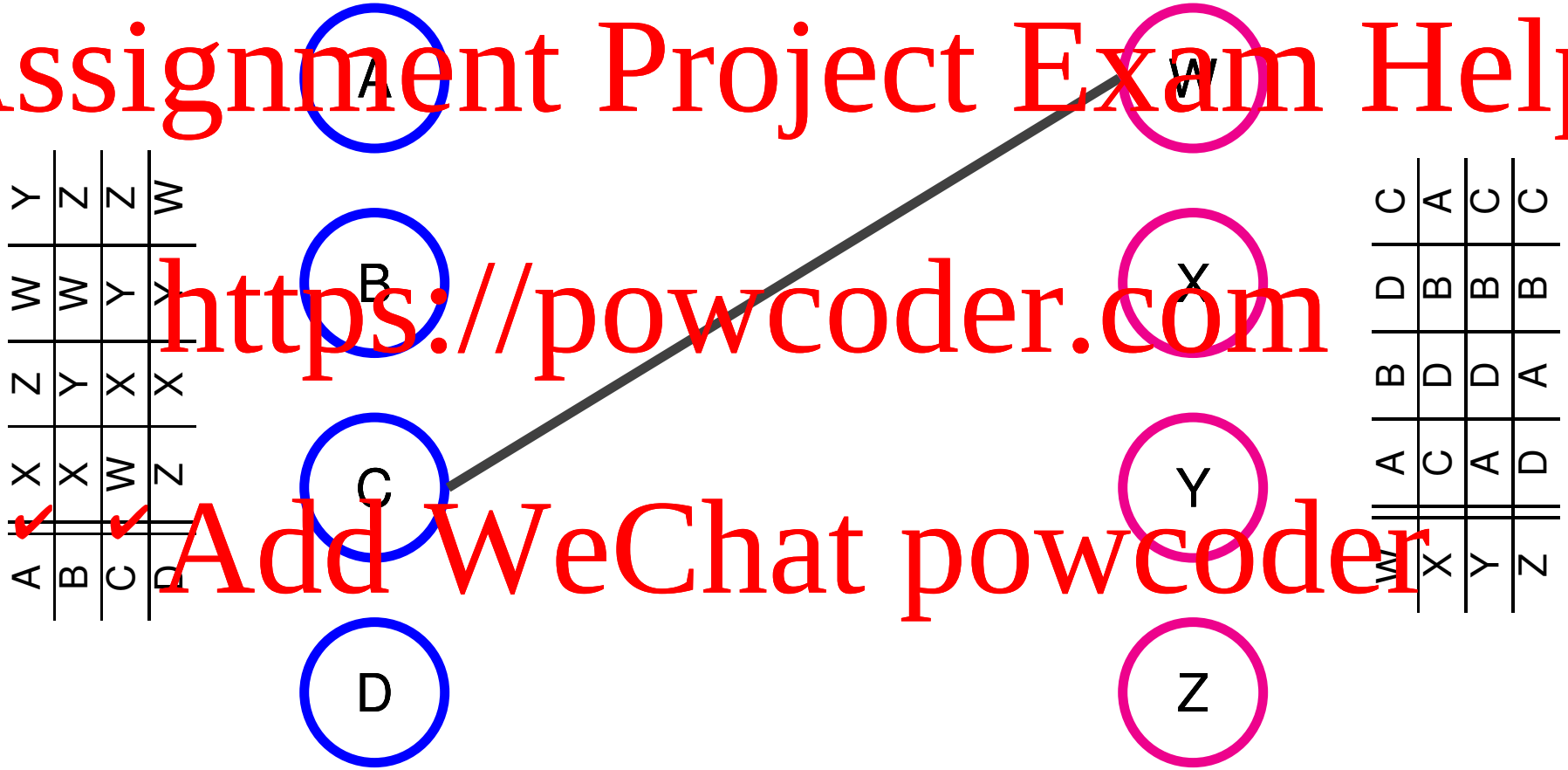
Illustration

Choose $\alpha = \mathsf{A}$



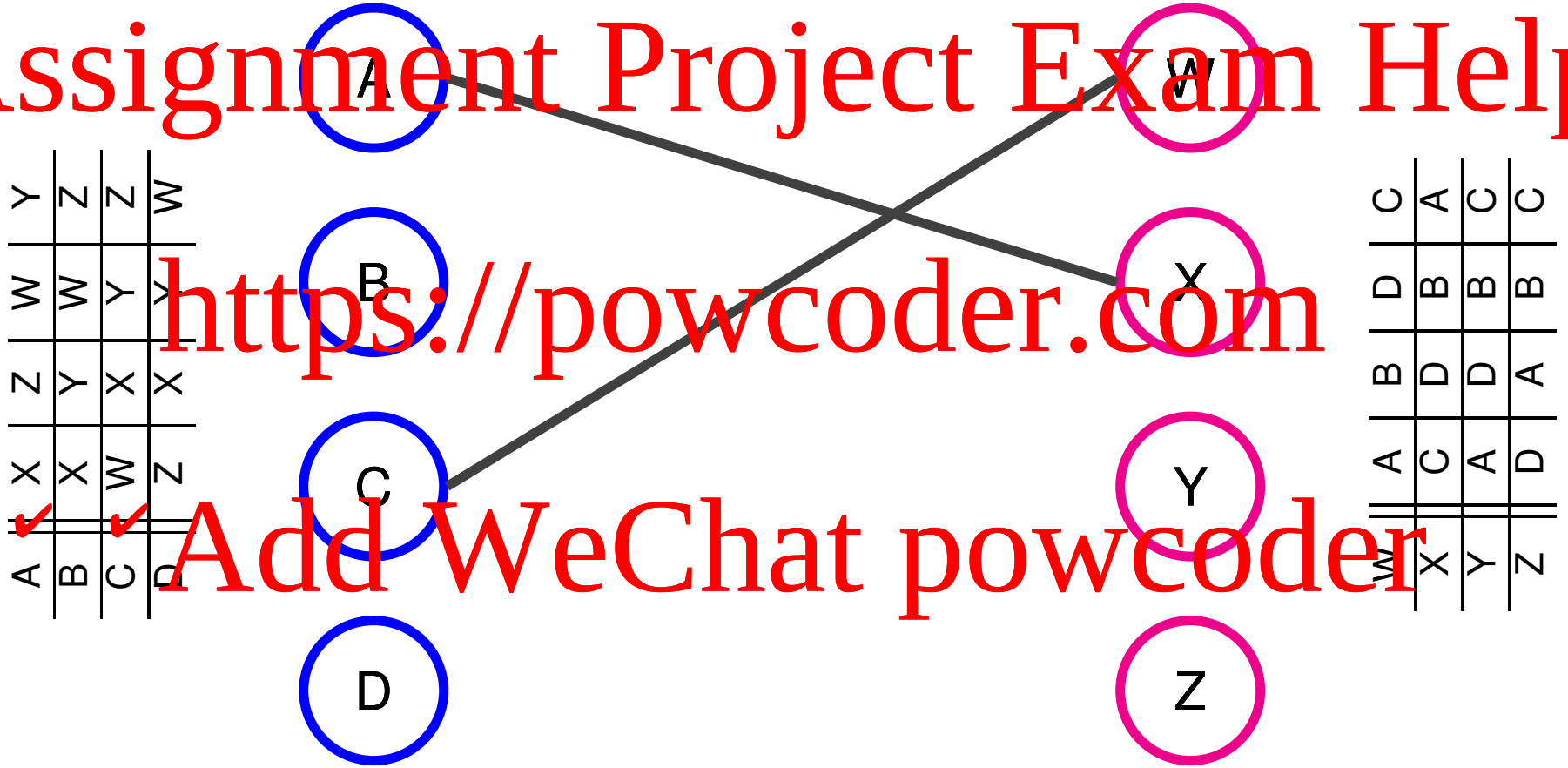
Illustration

$\beta = \mathsf{X}$



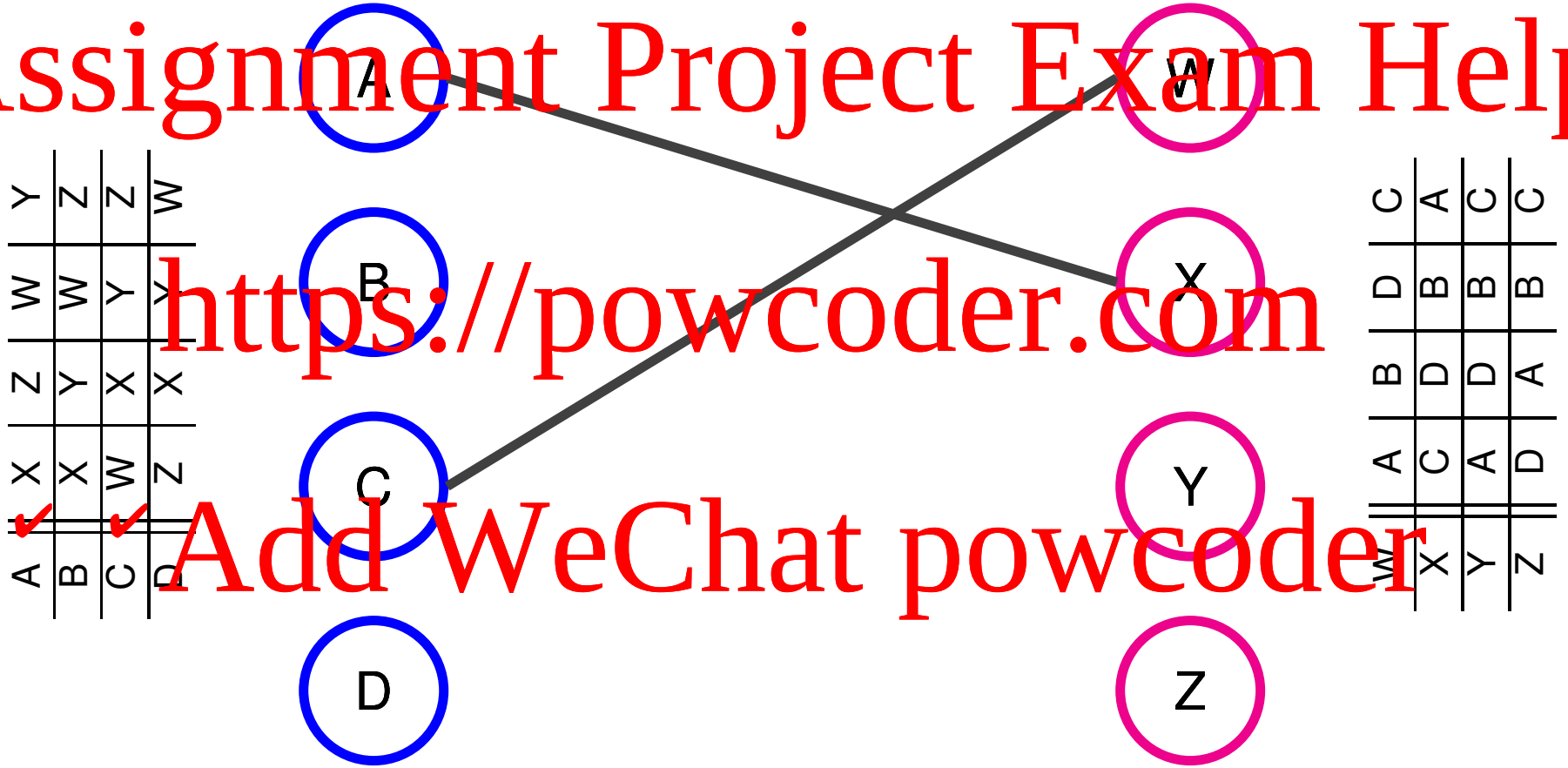
Illustration

X is free, so A and X become engaged to each other



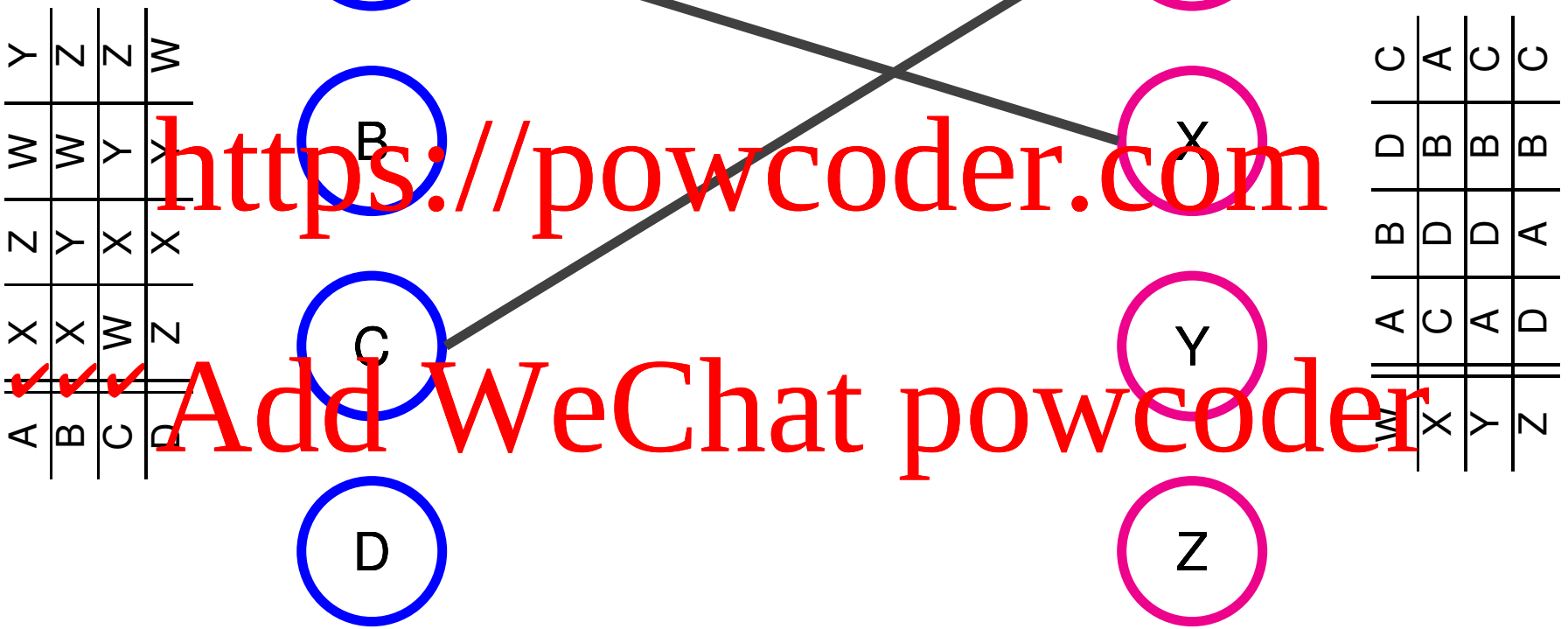
Illustration

Choose $\alpha = \mathsf{B}$



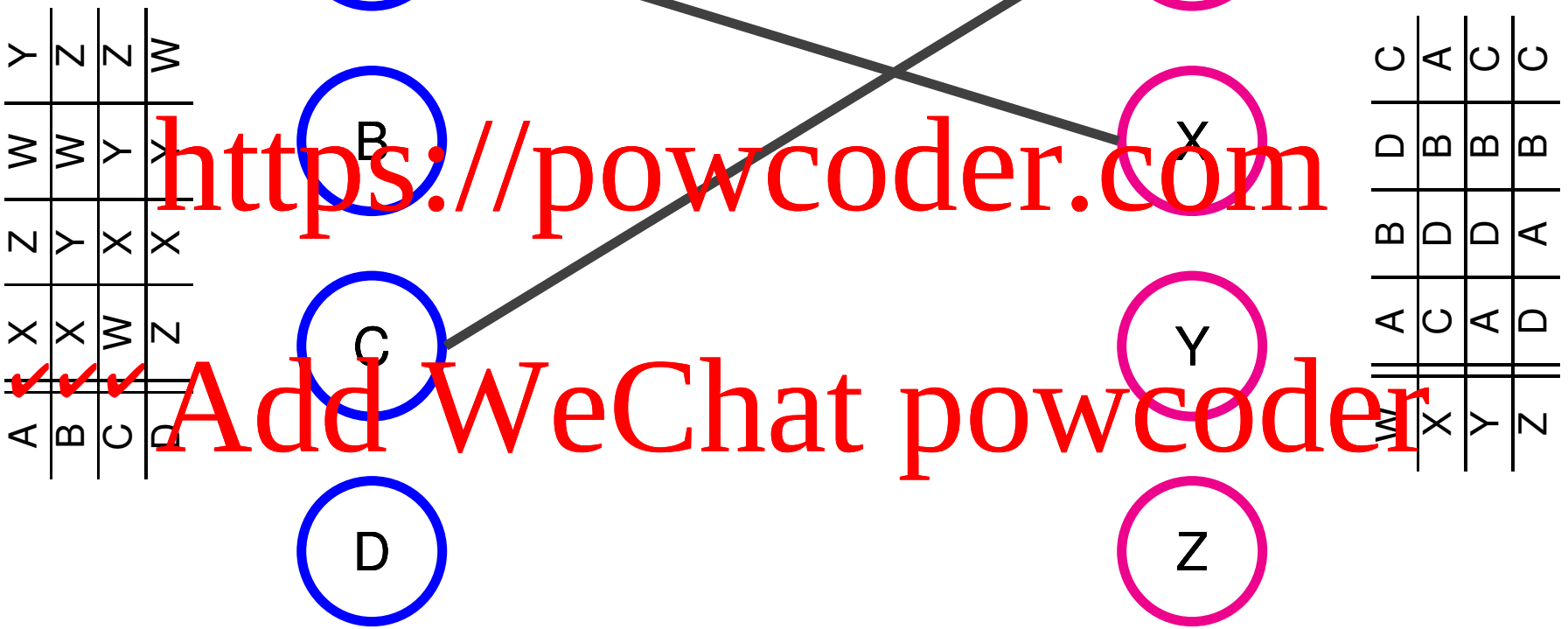
Illustration

$\beta = \mathsf{X}$



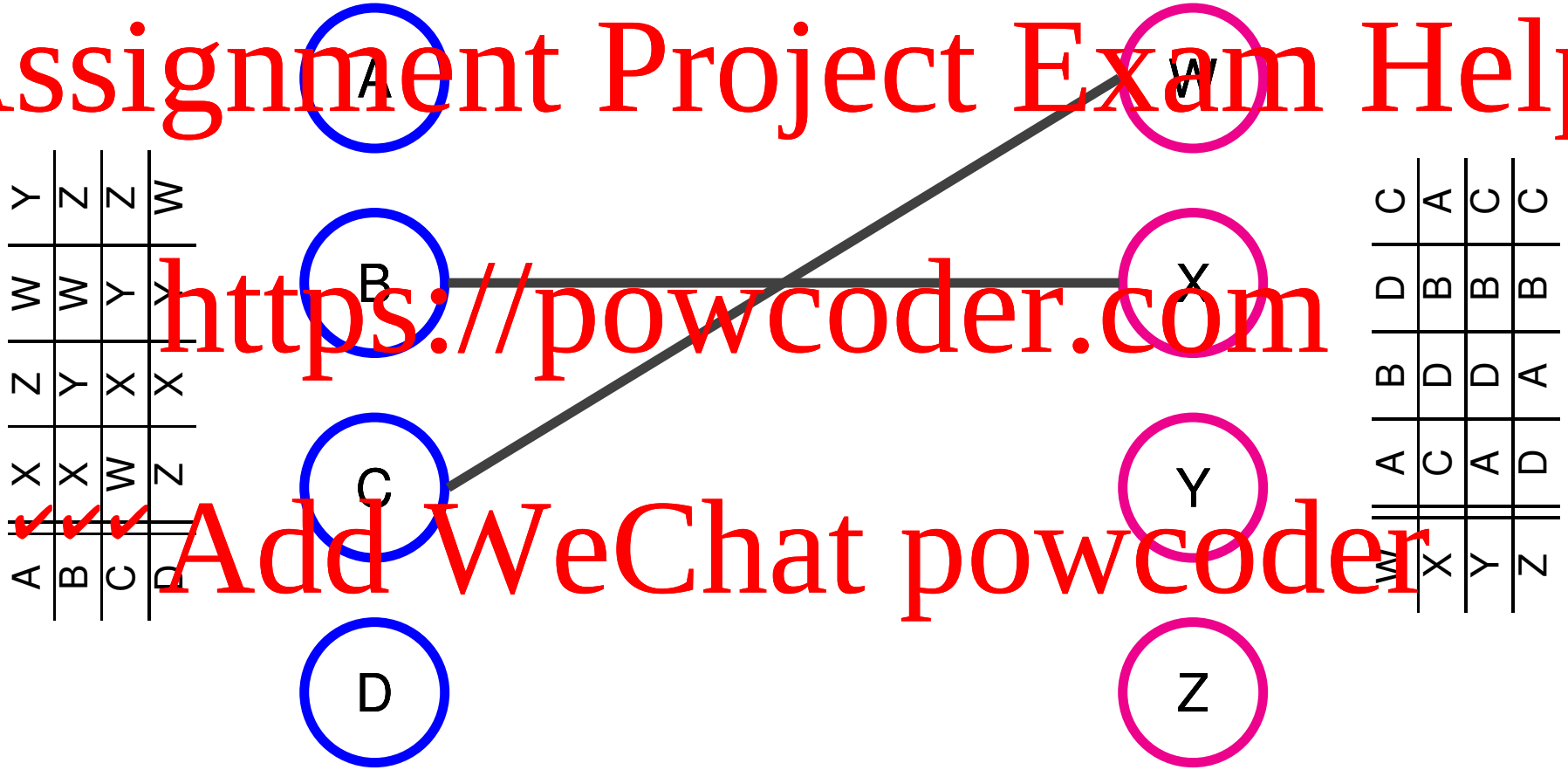
Illustration

X is already engaged to A, but prefers B



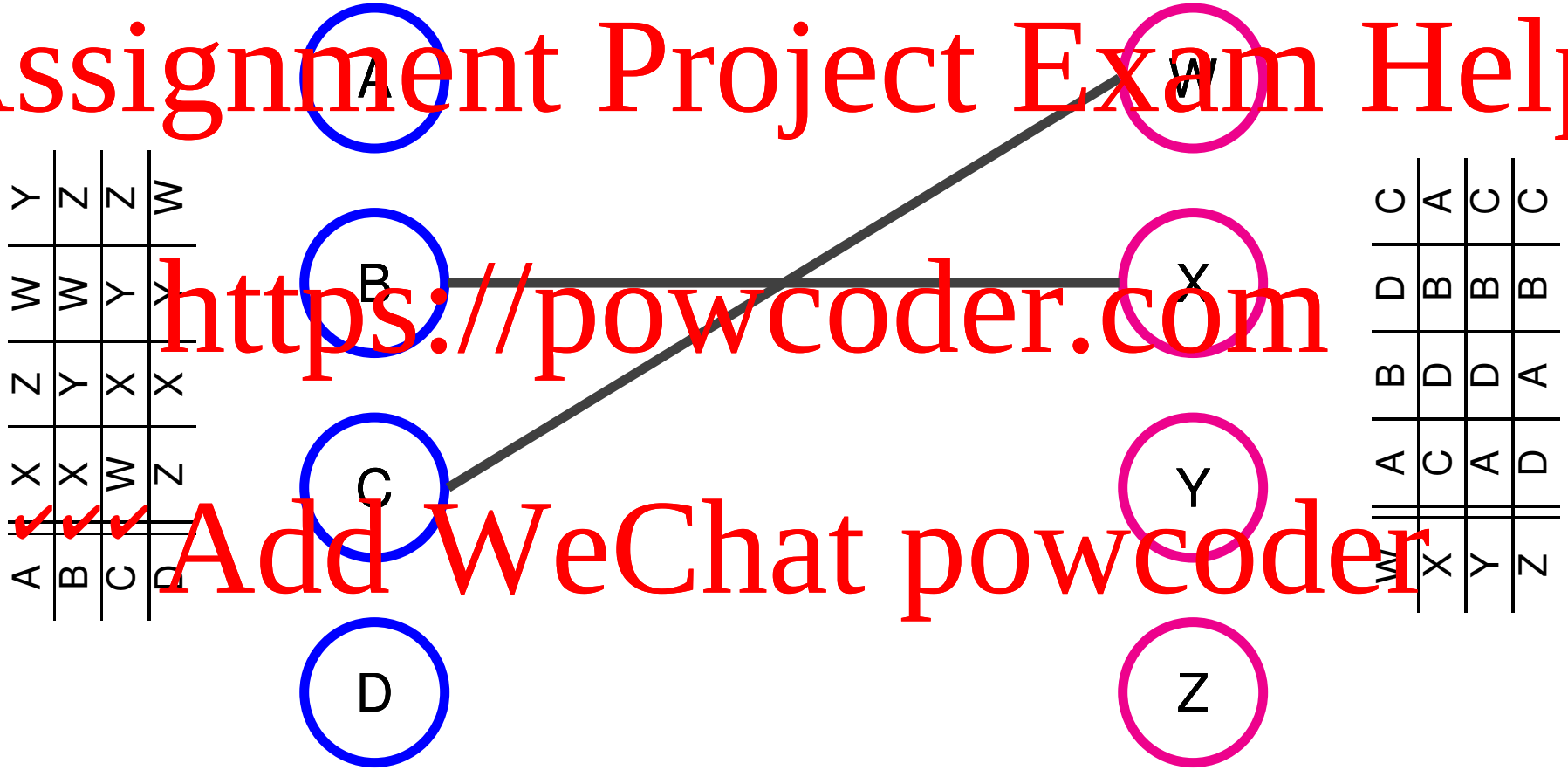
Illustration

A and X break engagement and B and X become engaged



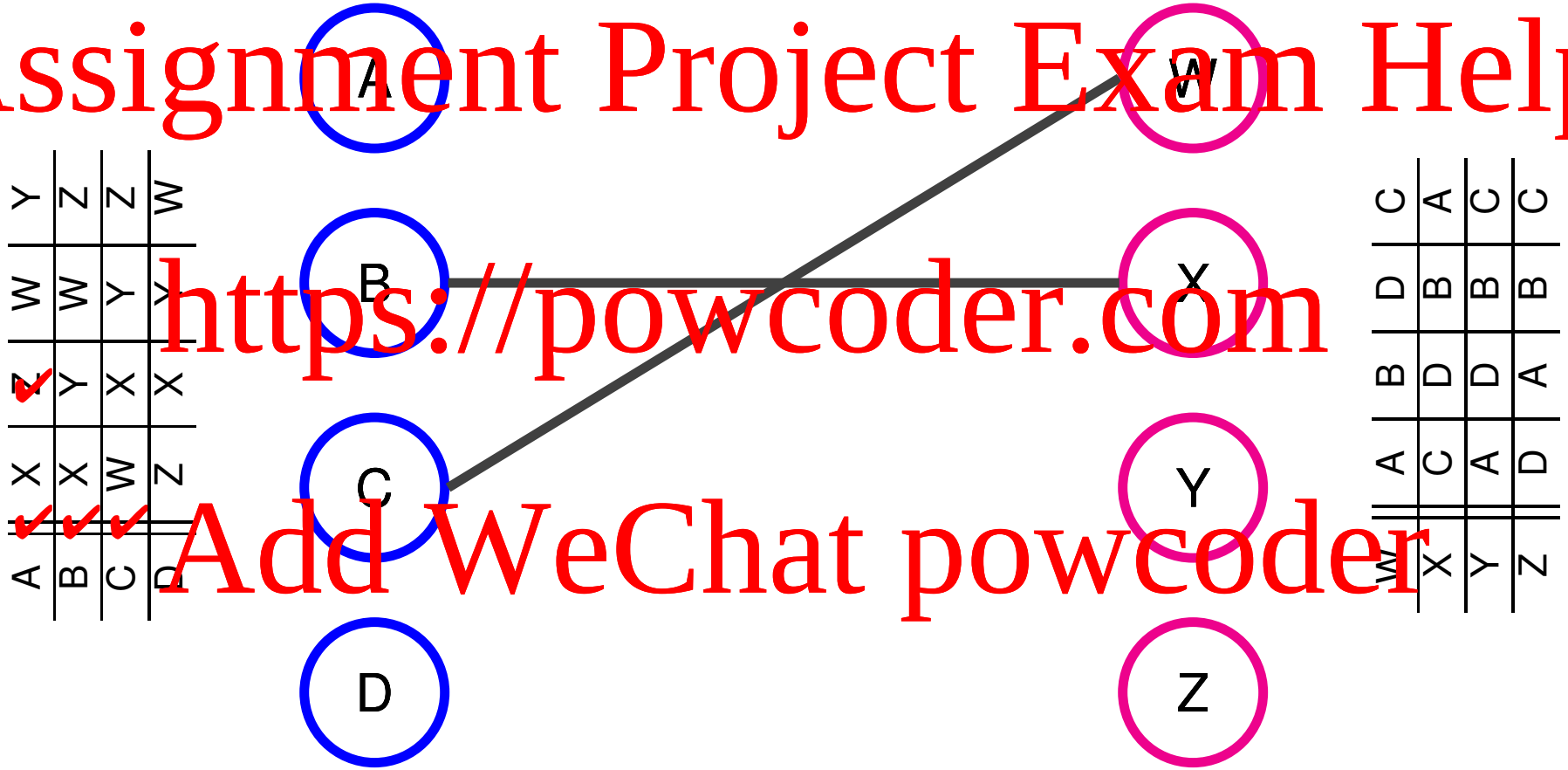
Illustration

Choose $\alpha = \text{A}$



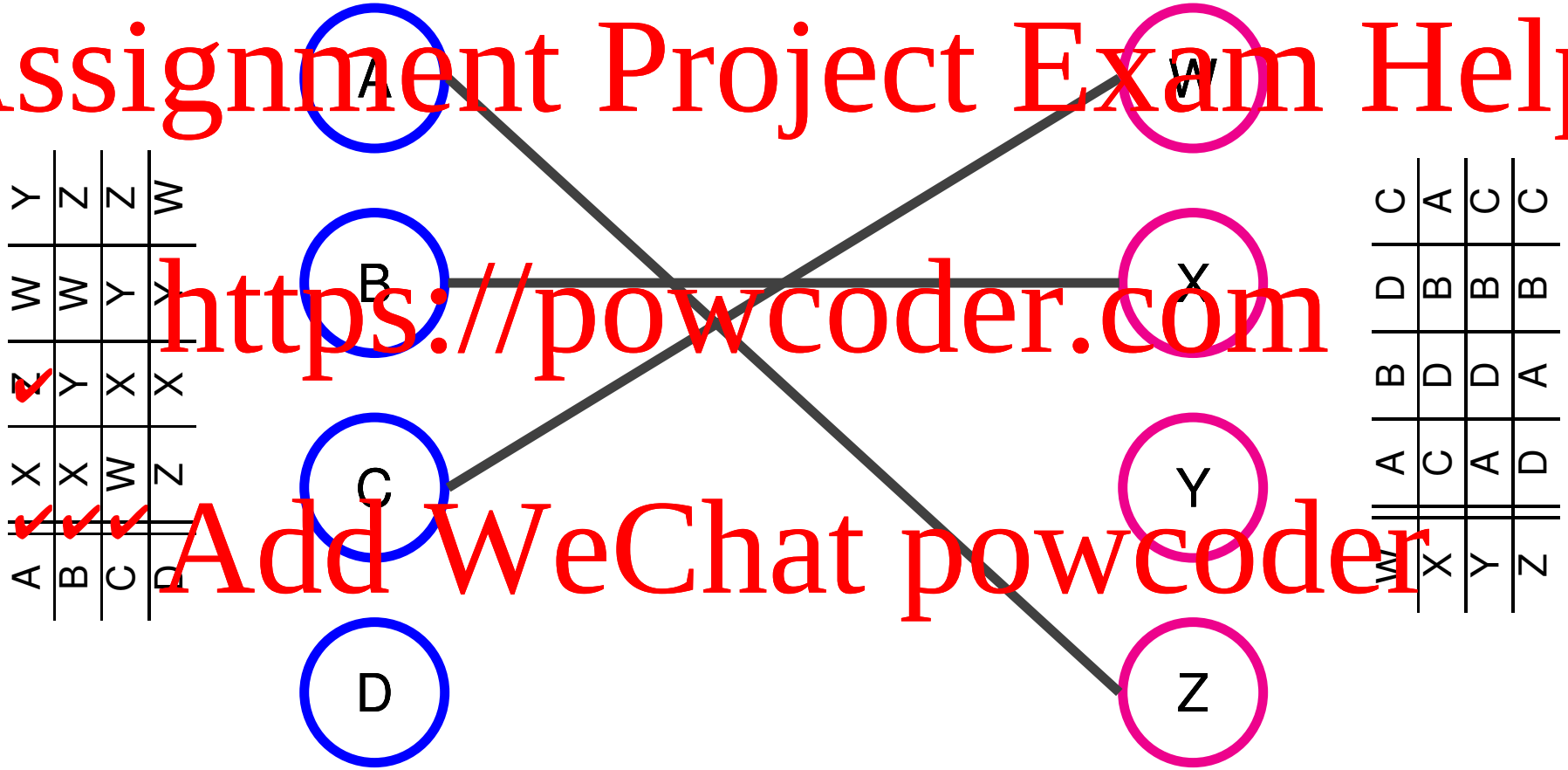
Illustration

$\beta = \mathsf{Z}$



Illustration

Z is free, so A and Z become engaged to each other



Illustration

$\alpha = \mathsf{D}$

	A	B	C	D
Y	Z	Z	W	
W	W	Y	Y	
Z ✓	Y	X	X	
X ✓	X ✓	W ✓	Z	

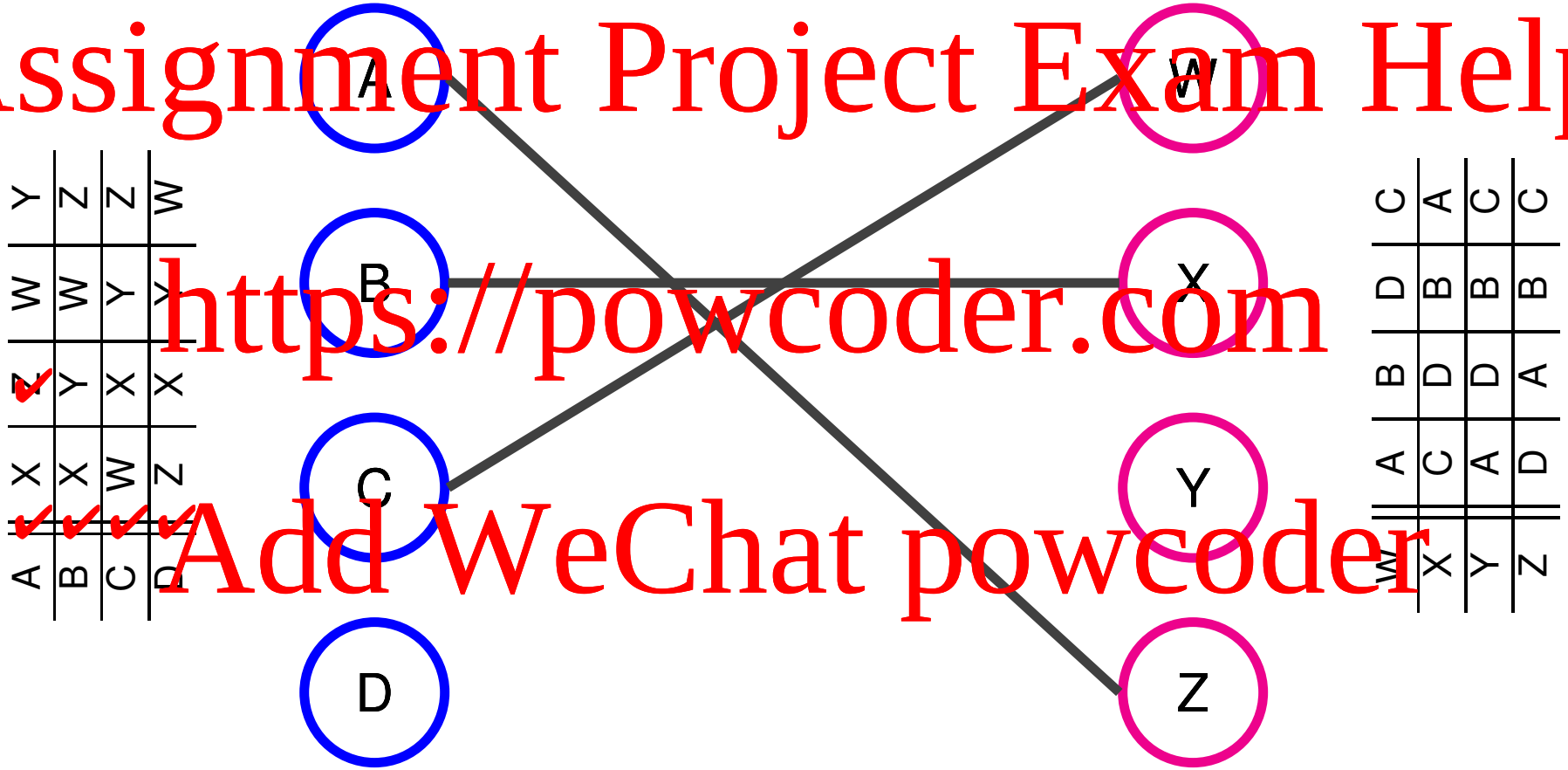
	W	X	Y	Z
C	A	C	C	
D	B	B	B	
B	D	A	D	
A	C	D	A	

Illustration

$\beta = \{Z\}$

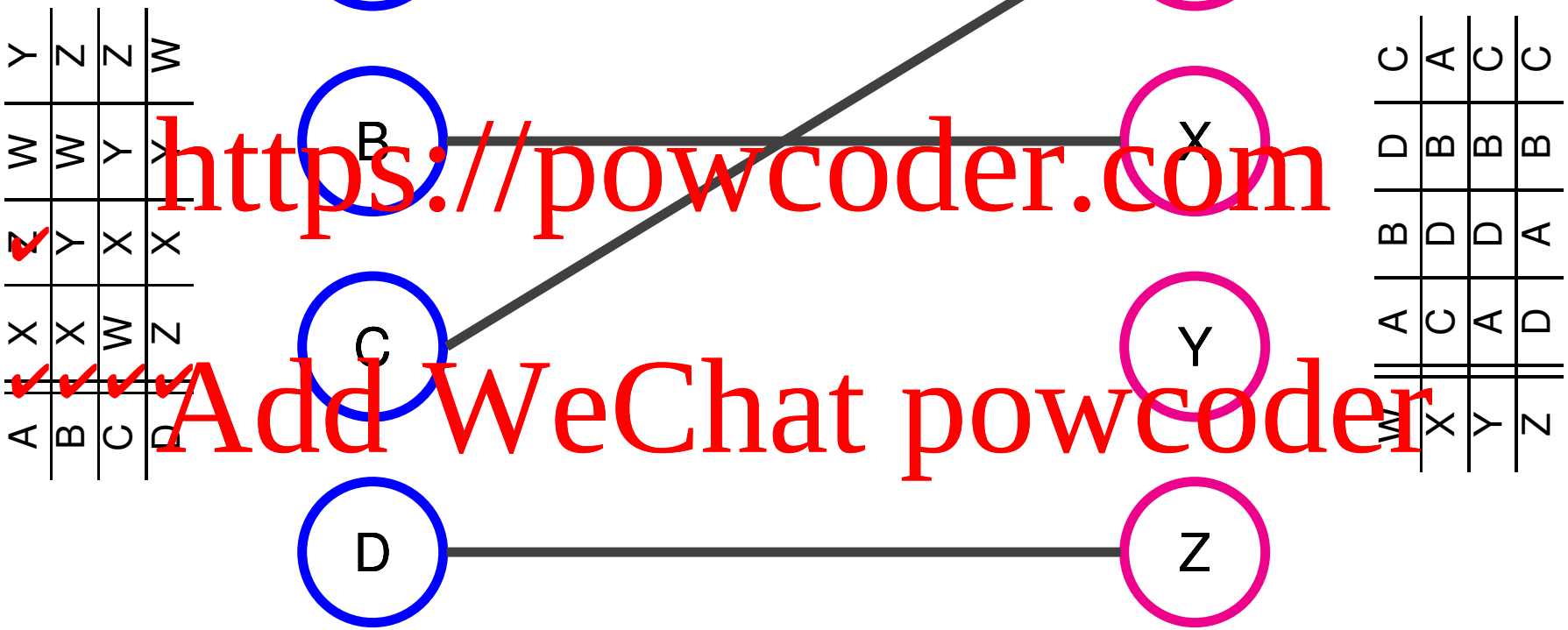
Illustration

Z is already engaged to A, but prefers D



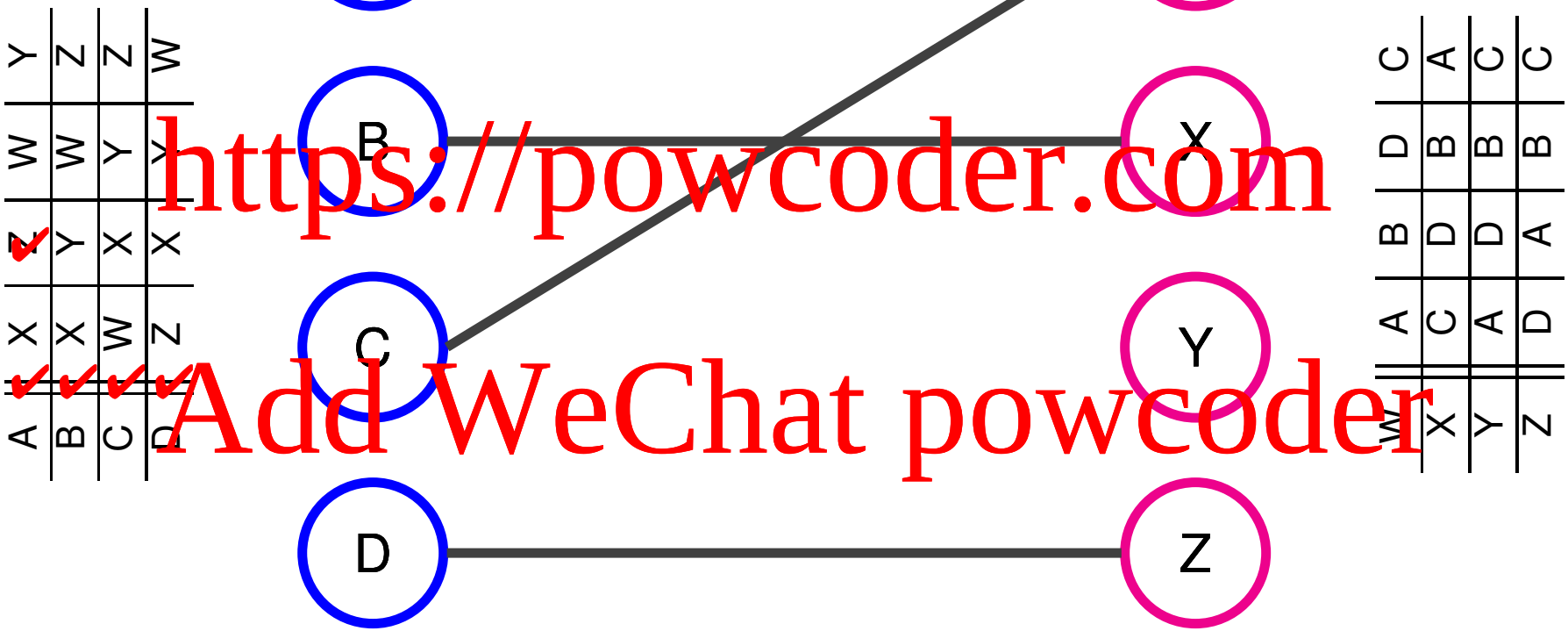
Illustration

A and Z break engagement and D and Z become engaged



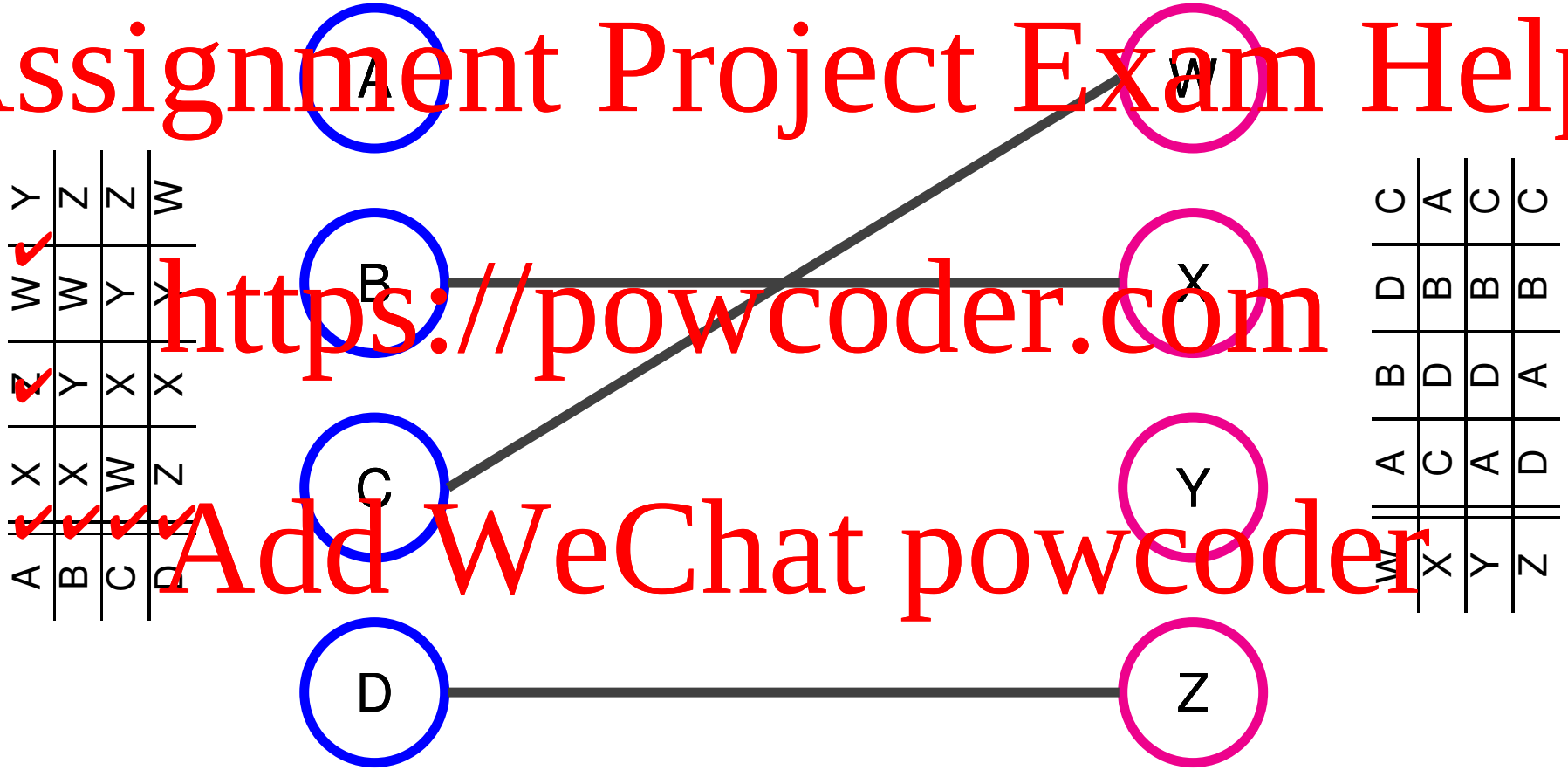
Illustration

$\alpha = \{A\}$



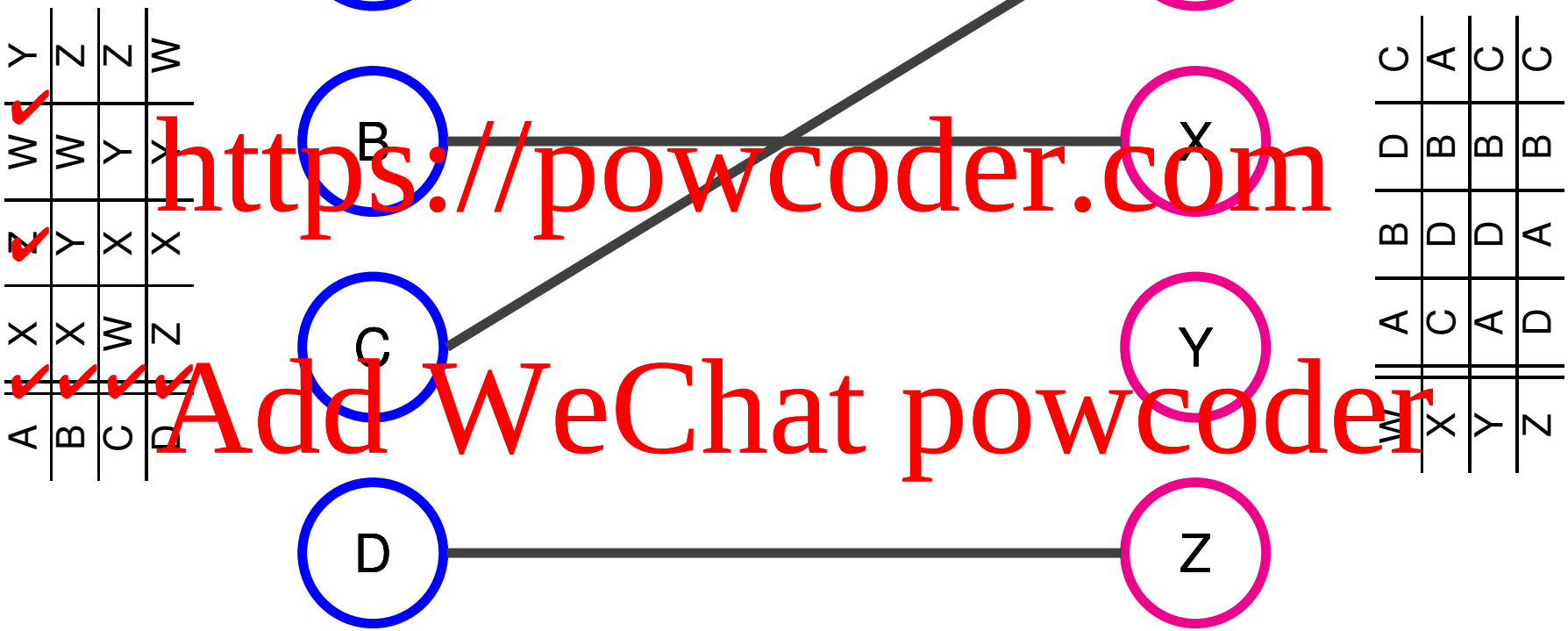
Illustration

$\beta = \{W\}$



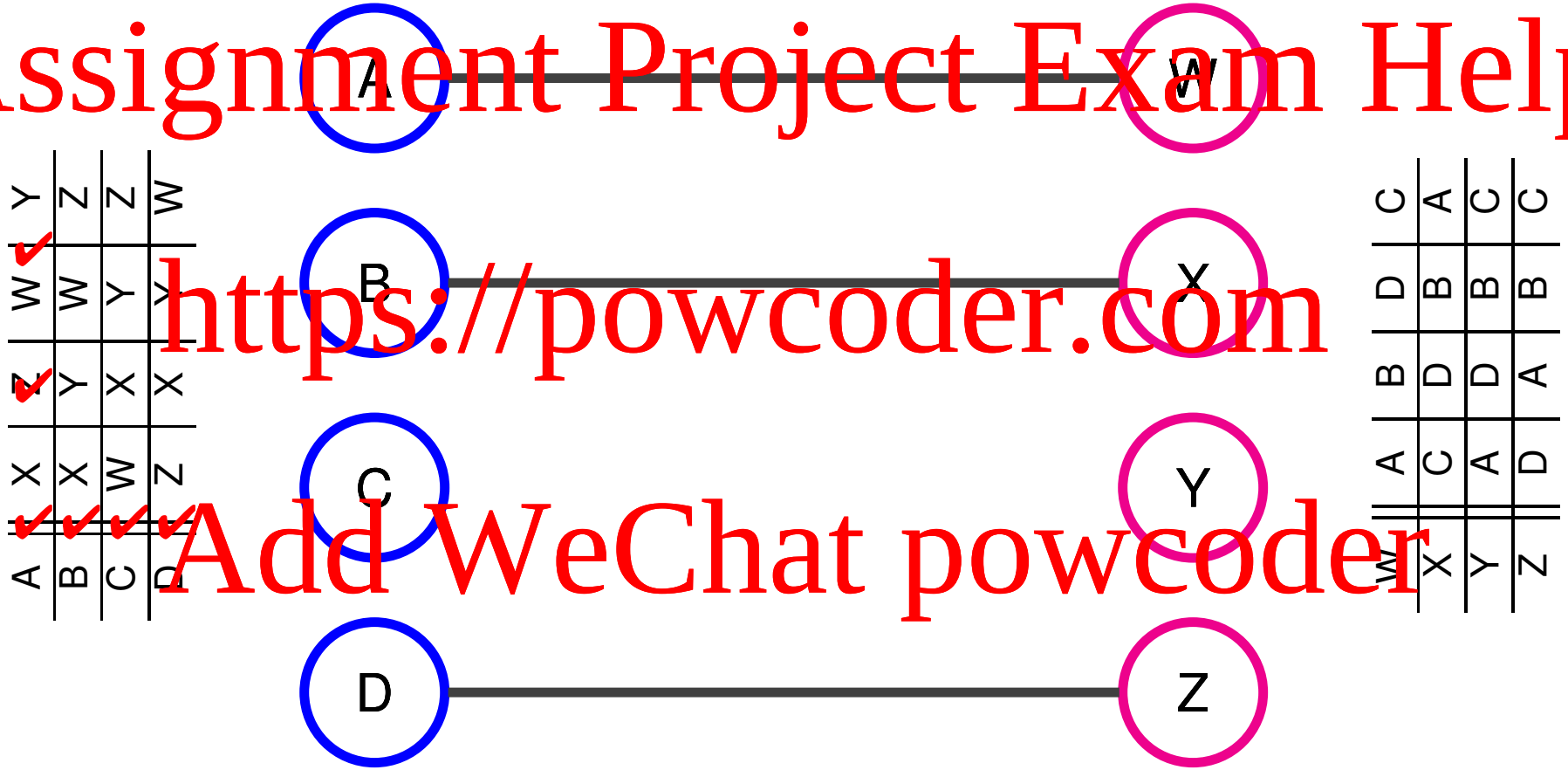
Illustration

W is already engaged to C, but prefers A



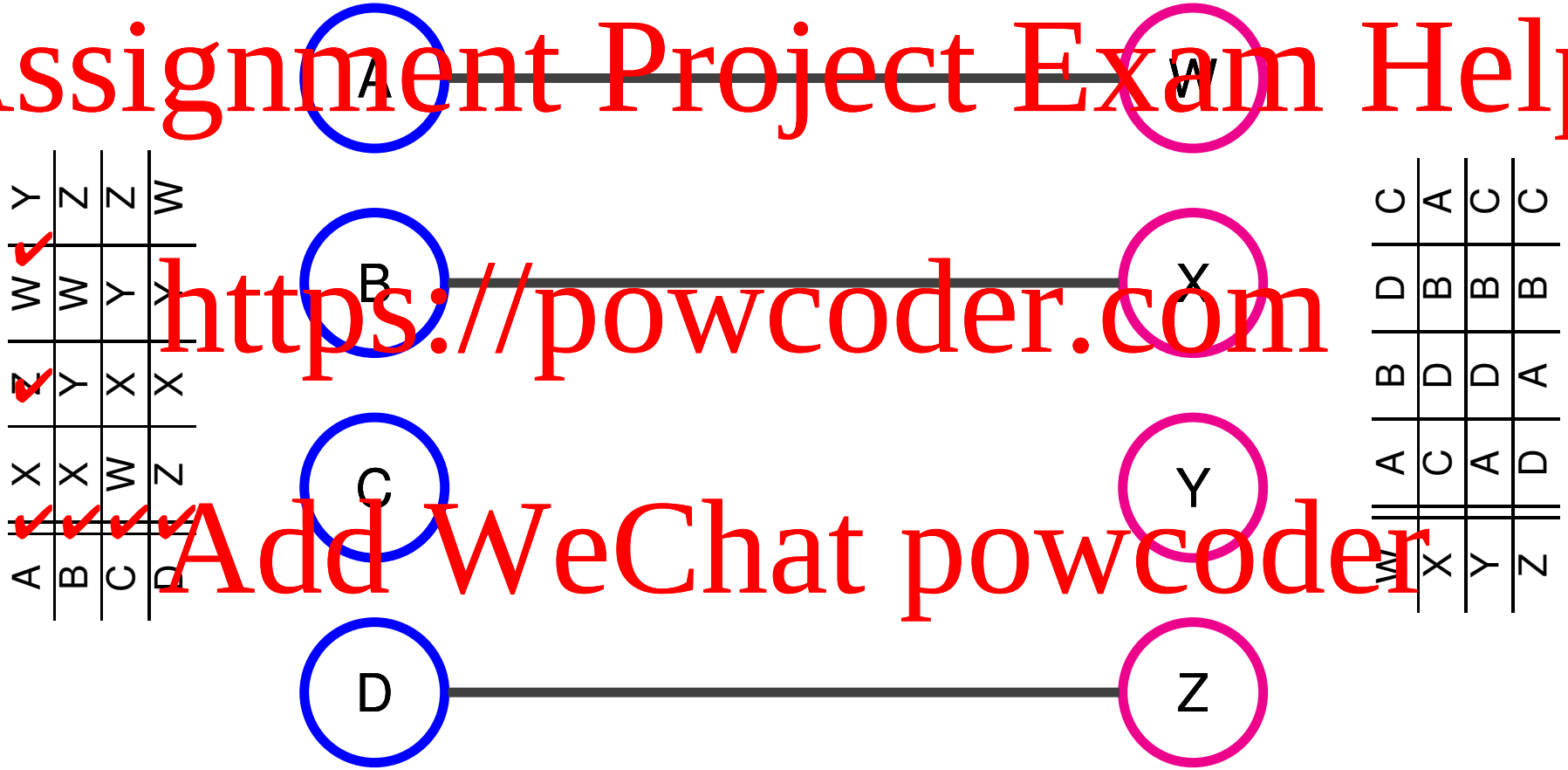
Illustration

C and W break engagement and A and W become engaged



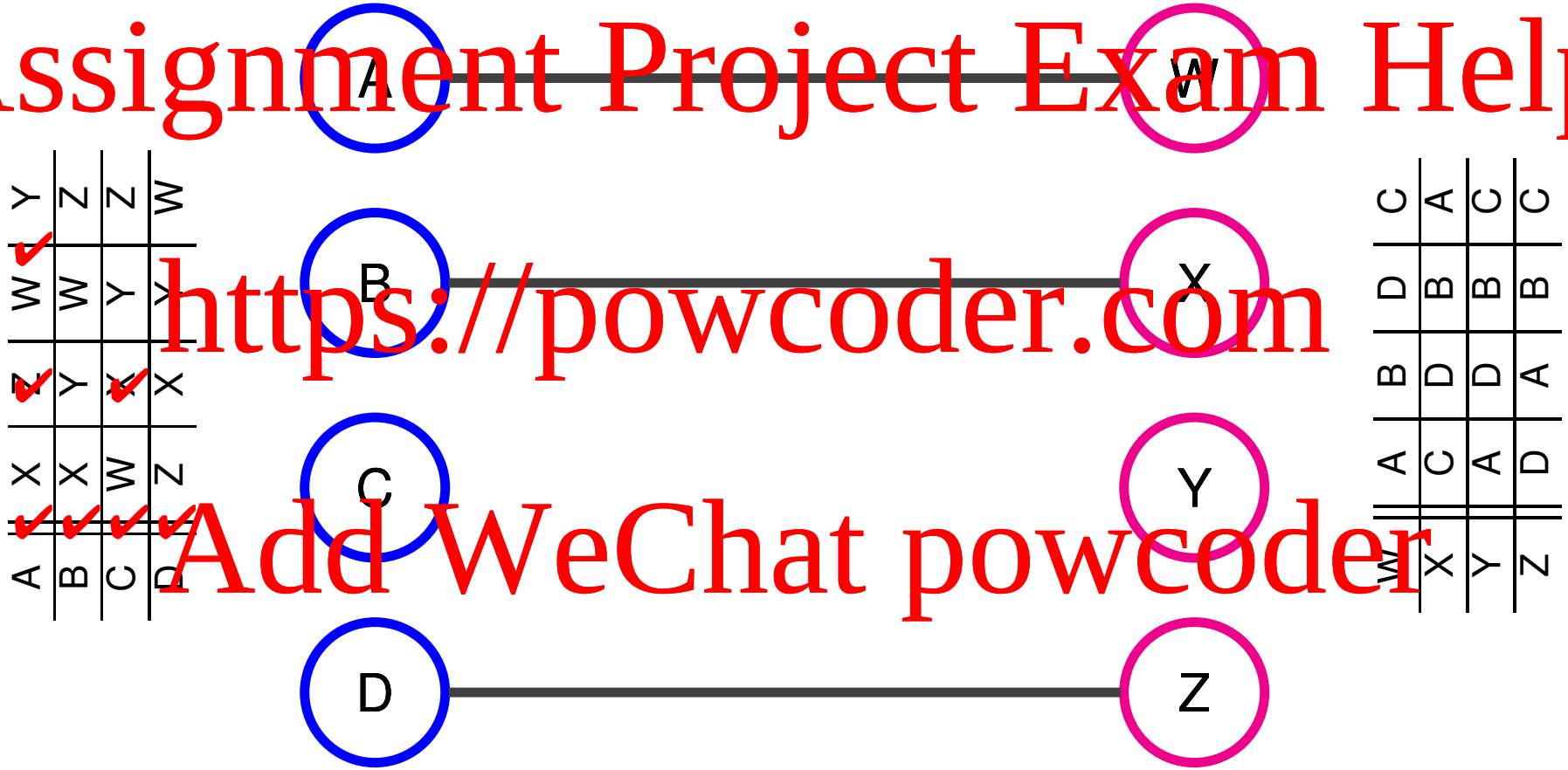
Illustration

$\alpha = \{A, B, C, D\}$



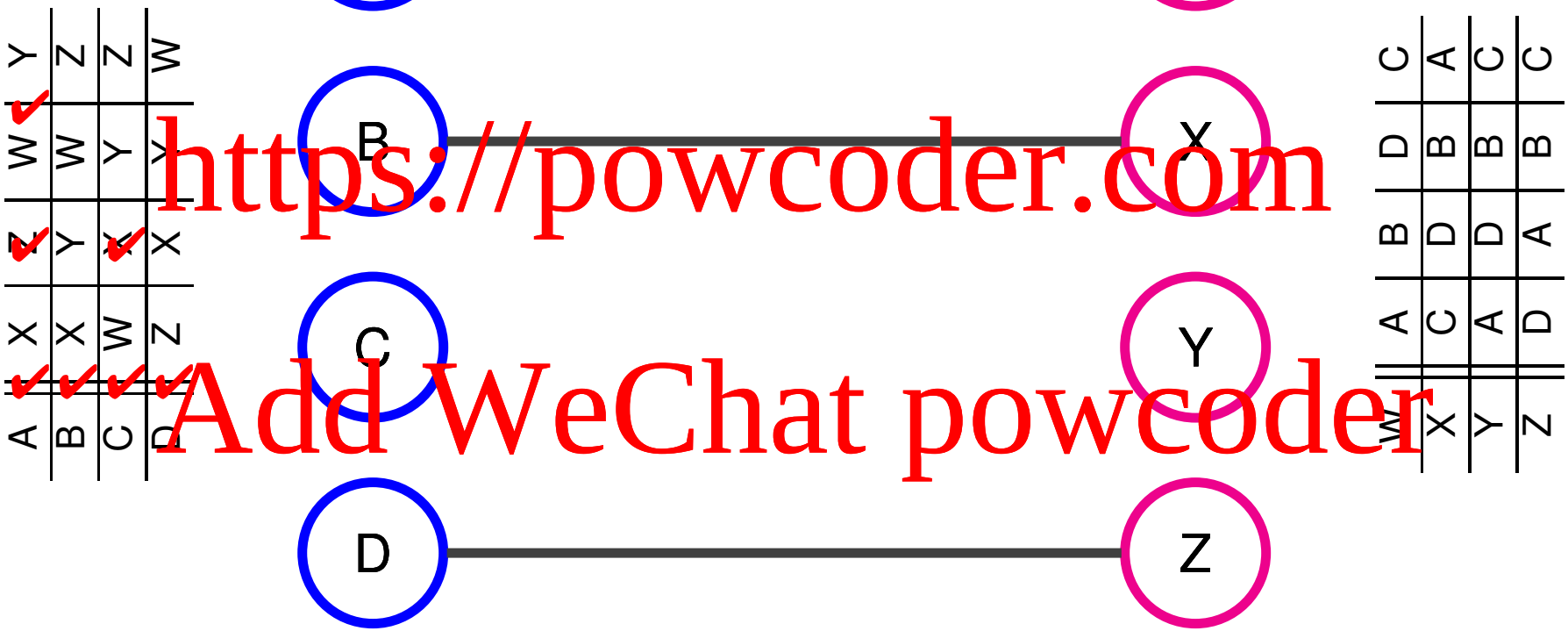
Illustration

$\beta = \mathsf{X}$



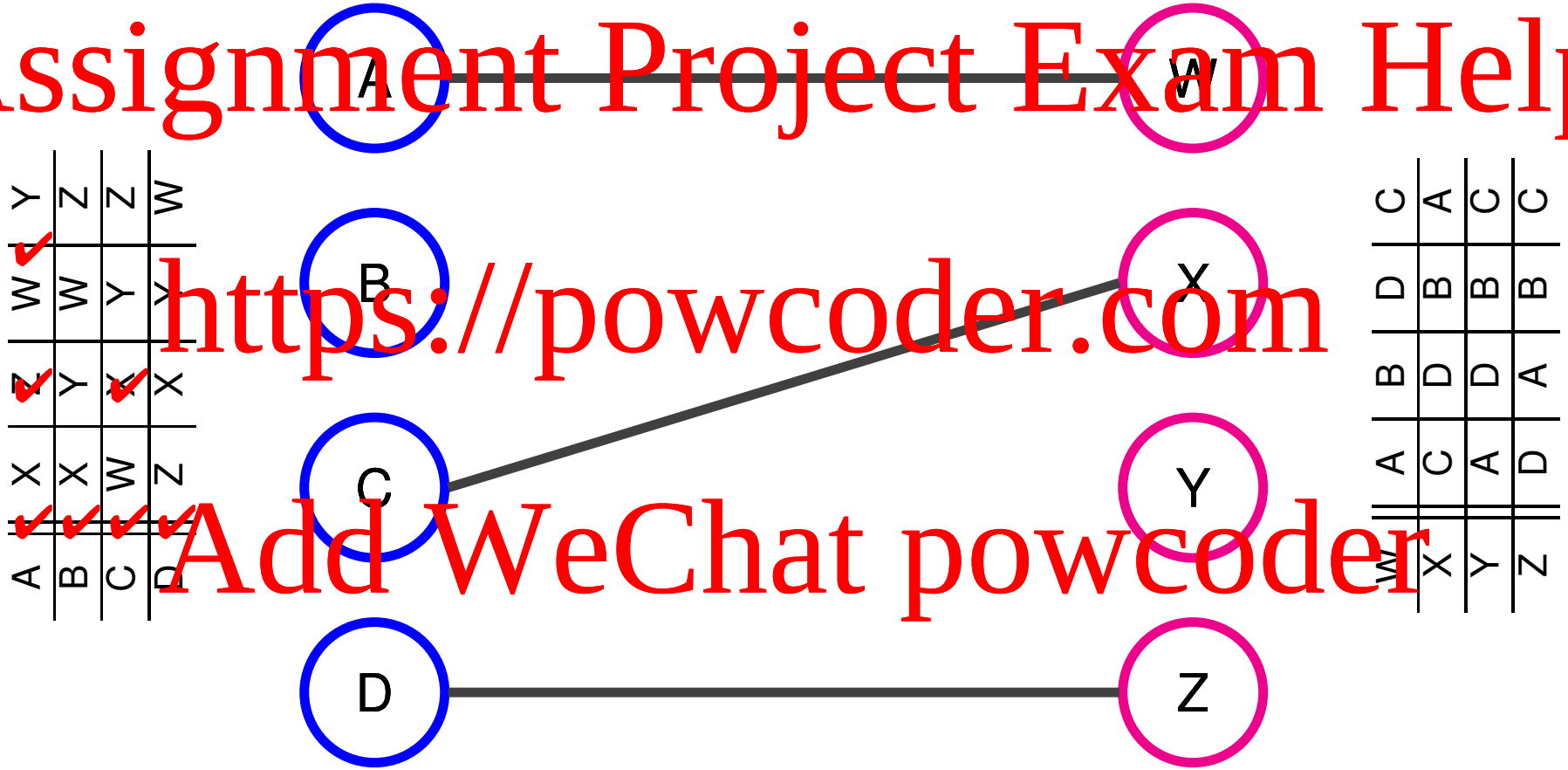
Illustration

X is already engaged to B, but prefers C



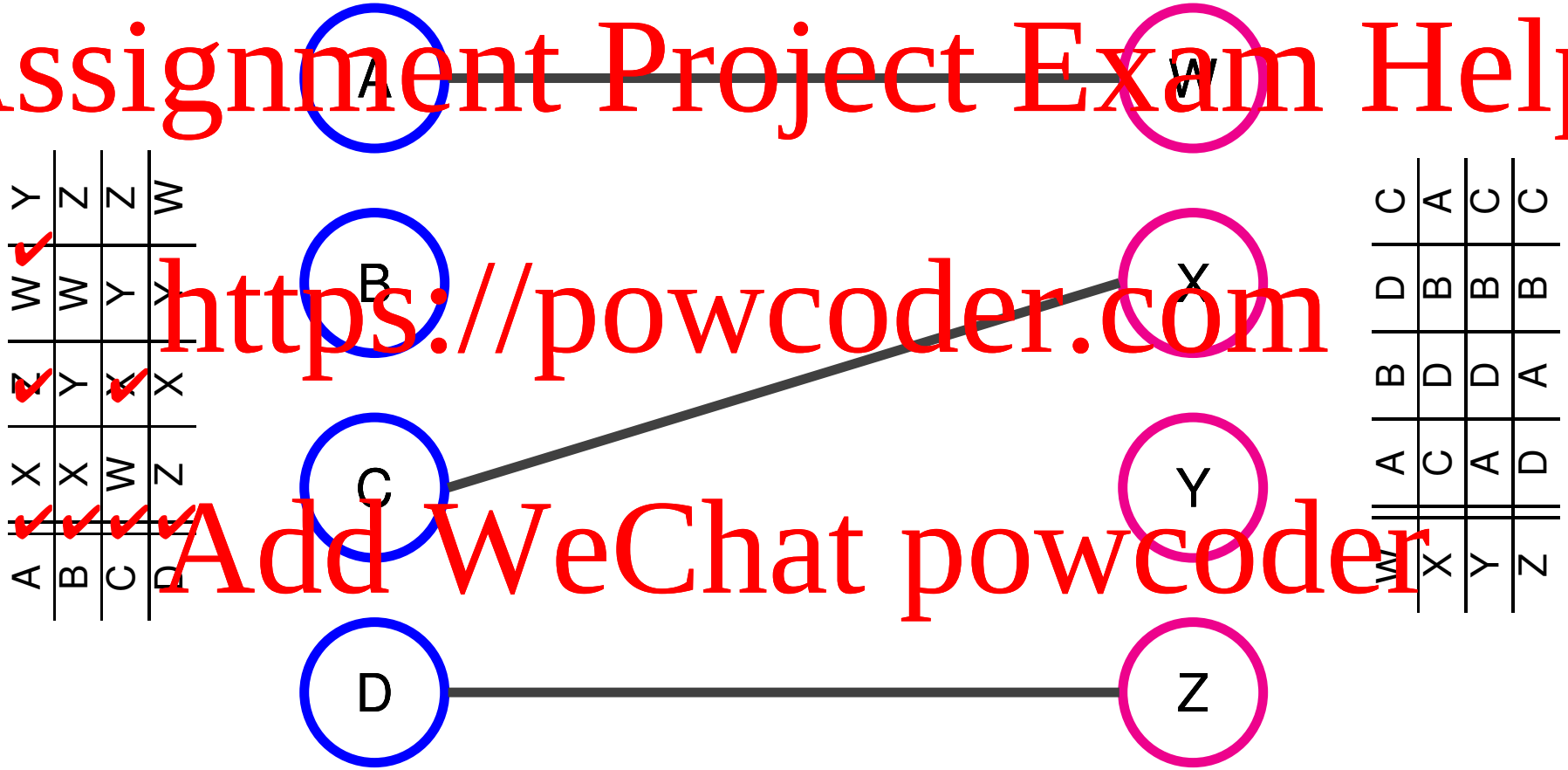
Illustration

B and X break engagement and C and X become engaged



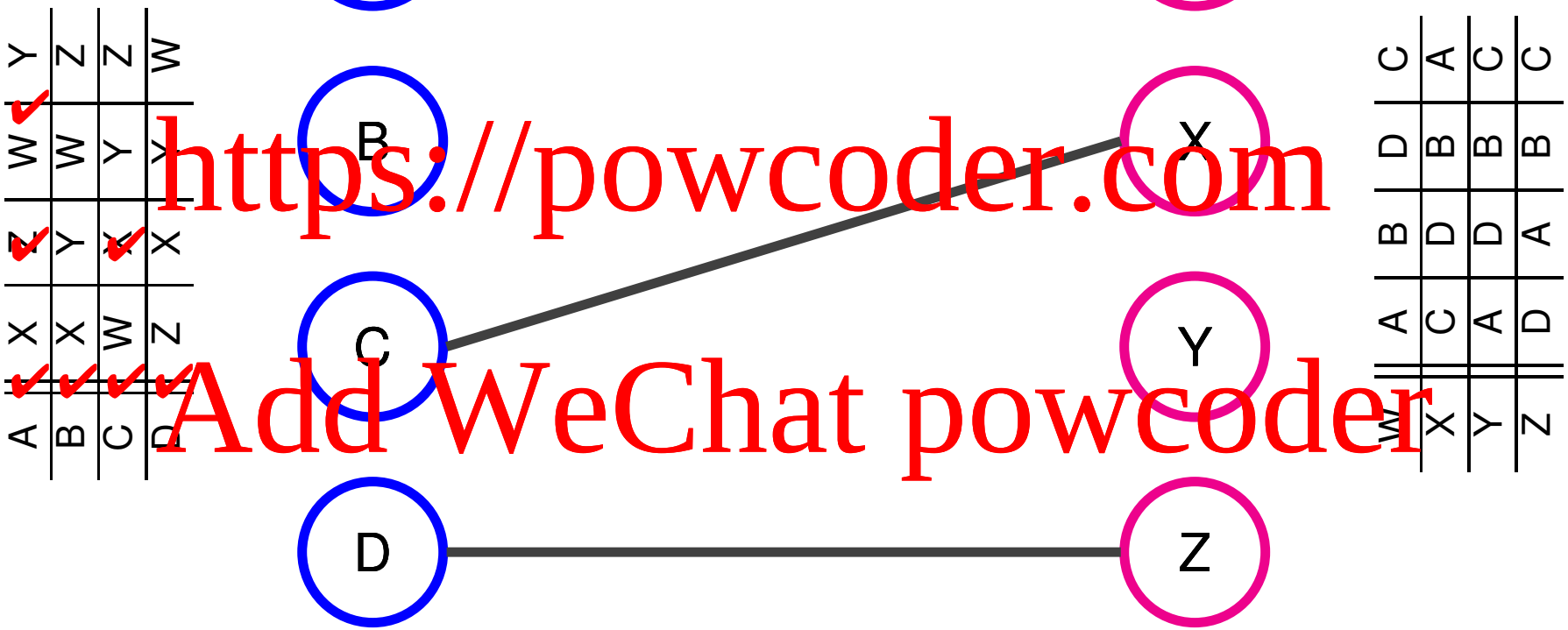
Illustration

$\alpha = \mathsf{B}$



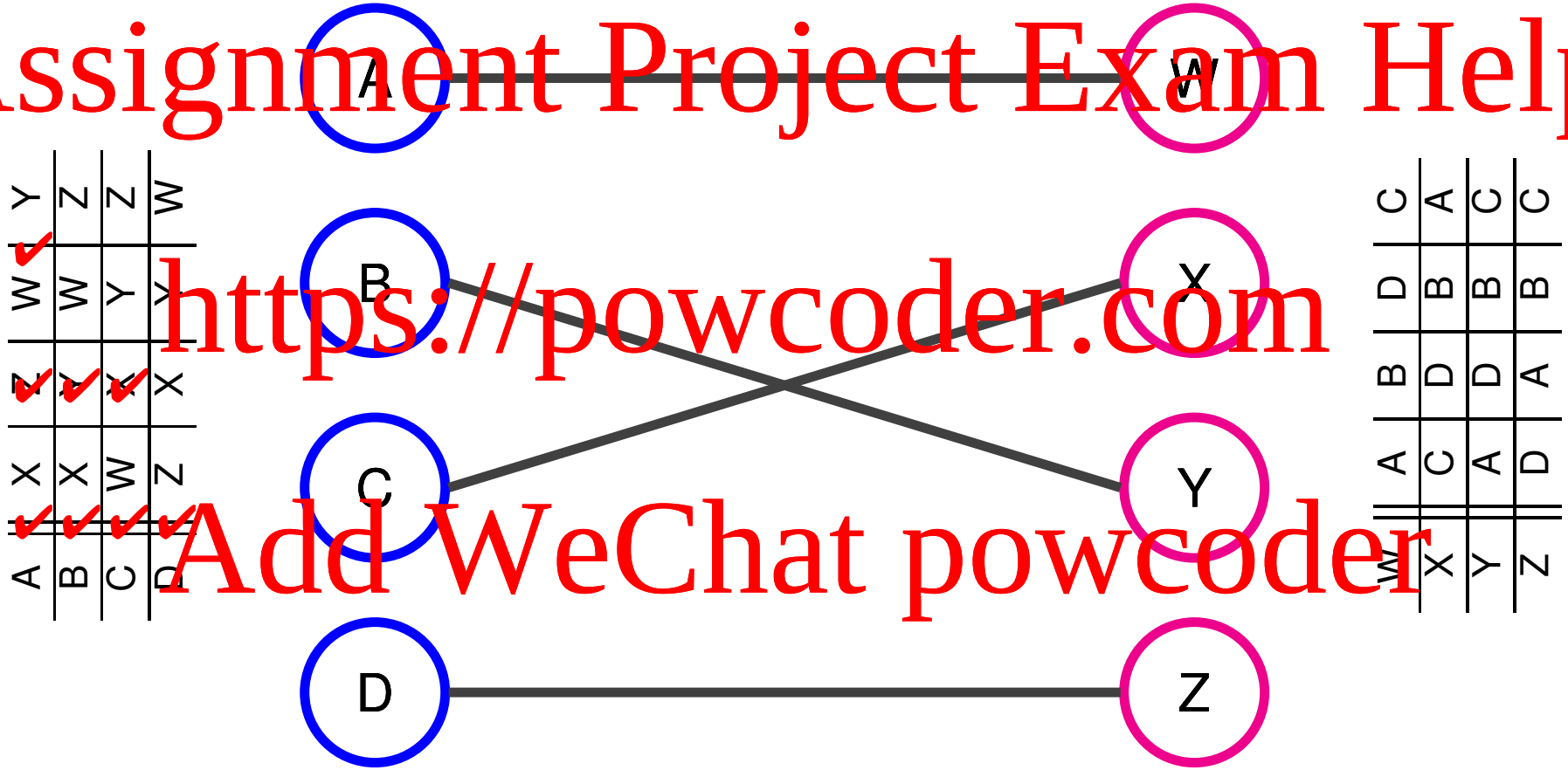
Illustration

$\beta = \mathsf{Y}$



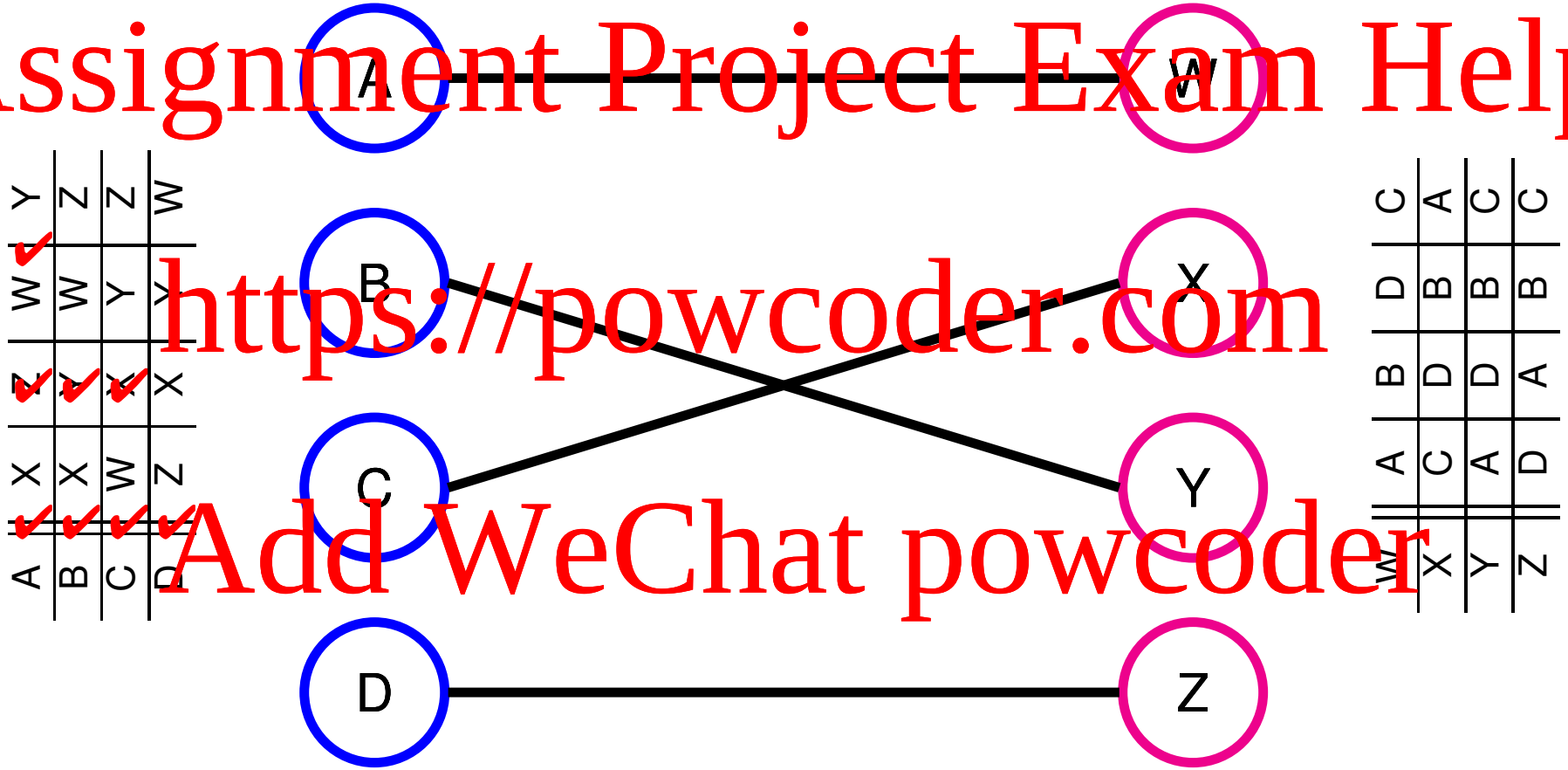
Illustration

Y is free, so B and Y become engaged to each other



Illustration

No one is now free, so a stable match has been established



Example for you

Apply the algorithm to this problem instance.

A	Y	W	Z	X
B	Z	W	Y	X
C	Z	Y	X	W
D	W	Y	X	Z

W	C	D	B	A
X	A	B	D	C
Y	C	B	D	A
Z	C	B	A	D

Always choose blue person in order of rows in table

Example for you

Apply the algorithm to this problem instance.

A	Y	W	Z	X
B	Z	W	Y	X
C	Z	Y	X	W
D	W	Y	X	Z

W	C	D	B	A
X	A	B	D	C
Y	C	B	D	A
Z	C	B	A	D

Always choose blue person in order of rows in table

C paired with Z
D paired with W
B paired with Y
A paired with X

Algorithm

initialize each person to be free
while some blue person is free & has red people to propose to
    choose one such blue person, α
    let β be next best option α's preference list
    if β is free
        assign α and β to be engaged
    else if β prefers α to β's fiancé α'
        assign α and β to be engaged, and α' to be free
    else
        β rejects α

Gale and Shapley, 1962

Questions for you

- Give a problem instance involving two blue people and two red where there are two stable matches. Which one will the algorithm find?

Questions for you

- Give a problem instance involving two blue people and two red where there are two stable matches. Which one will the algorithm find?

blueA	redA	redB
blueB	redB	redA

redA	blueB	blueA
redB	blueA	blueB

Issues

- Is the algorithm guaranteed to terminate?
- How efficient is the algorithm?
- Can we be certain that the algorithm always finds a stable match?
- How fair is the algorithm?
- Does it matter how we resolve the non-determinism?

Termination

- How many times could the **while** loop be executed?
- Need some measure of progress
  — a value that monotonically increases as number of executions of loop increases
- Consider: *the number of blue people that are still free*
- Reject: this doesn't necessarily decrease each time loop is executed
- Consider: *the number of engaged couples*
- Reject: this doesn't necessarily increase each time loop is executed

Measure of Progress

- *The number of proposals made so far*
- Always increases by 1 when loop executed
- Upper limit on number of proposals is n^2
  — where n is number of blue people (and number of red people)
- Gives upper limit on executions of **while** loop
- The algorithm will therefore always terminate

Efficiency

- Running time will depend on how long it takes for each execution of loop
- Possible to do it in constant time
- We will look at this shortly
- Needs more detailed consideration of data structures

Proving Correctness

- A stable match is always found
- Let's show that it would be a contradiction to assert that . . .

there is an unstable pair in a match produced by the algorithm

- A proof by contradiction

Continuing Proof of Correctness

Suppose we have the following:

- an unstable pair α and β
- α paired by the algorithm with $\beta' \neq \beta$
- β paired by the algorithm with $\alpha' \neq \alpha$

Instability means:

- α prefers β to β'
- β refers α to α'

Completing Proof of Correctness

Case 1: α didn't propose to β at any point

- α proposes in decreasing order of preference
- α must have proposed to β'
- So β' must be preferred to β by α
- Contradicts previous assumption

Case 2: β proposed to α but was rejected (immediately or at later point)

- β would only reject α in favour of a blue person she preferred
- So α' must be preferred to α by β
- Contradicts previous assumption