THE UNIVERSITY OF
MELBOURNE

COMP90038 Algorithms and Complexity

Assignment Project Exam Help

Lecture 5: Brute Force Methods

https://powcoder.com

(with thanks to Harald Søndergaard)

Add WeChat powcoder

Toby Murray

✉ toby.murray@unimelb.edu.au

👤 DMD 8.17 (Level 8, Doug McDonell Bldg)

🌐 http://people.eng.unimelb.edu.au/tobym

🐦 @tobycmurray

Compulsory Quizzes

THE UNIVERSITY OF MELBOURNE

- **Remember:** you need to **complete** 8 of the quizzes to pass the hurdle
- By “completing” a quiz, we mean **getting all answers right (100%) in one sitting**
- You can have as many attempts as you need, but on at least one of those attempts you need to score 100%
- The first compulsory quiz (for week 2) closes **tomorrow**

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Brute Force Algorithms

- Straightforward problem solving approach, usually based directly on the problem's statement.
- Exhaustive search for solutions is a prime example.
  - Selection sort
  - String matching
  - Closest pair
  - Exhaustive search for combinatorial solutions
  - Graph traversal

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Example: Selection Sort

```
function $\textsc{SelSort}(A[\cdot], n)$
    for $i \leftarrow 0$ to $n - 2$ do
        $min \leftarrow i$
        for $j \leftarrow i + 1$ to $n - 1$ do
            if $A[j] < A[min]$ then
                $min \leftarrow j$
        $t \leftarrow A[i]$          // swap A[i] and A[min]
        $A[i] \leftarrow A[min]$
        $A[min] \leftarrow t$
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Example: Selection Sort

```
function $\text{SelSort}(A[\cdot], n)$
    for $i \leftarrow 0$ to $n - 2$ do
        $min \leftarrow i$
        for $j \leftarrow i + 1$ to $n - 1$ do
            if $A[j] < A[min]$ then
                $min \leftarrow j$
        $t \leftarrow A[i]$          // swap A[i] and A[min]
        $A[i] \leftarrow A[min]$
        $A[min] \leftarrow t$
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Time Complexity:

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Example: Selection Sort

THE UNIVERSITY OF MELBOURNE

```
function SelSort($A[\cdot]$, $n$)
    for $i \leftarrow 0$ to $n - 2$ do
        $min \leftarrow i$
        for $j \leftarrow i + 1$ to $n - 1$ do
            if $A[j] < A[min]$ then
                $min \leftarrow j$
        $t \leftarrow A[i]$          // swap A[i] and A[min]
        $A[i] \leftarrow A[min]$
        $A[min] \leftarrow t$
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Time Complexity: $\Theta(n^2)$

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Example: Selection Sort

THE UNIVERSITY OF MELBOURNE

```
function SelSort($A[\cdot], n$)
    for $i \leftarrow 0$ to $n - 2$ do
        $min \leftarrow i$
        for $j \leftarrow i + 1$ to $n - 1$ do
            if $A[j] < A[min]$ then
                $min \leftarrow j$
        $t \leftarrow A[i]$        // swap A[i] and A[min]
        $A[i] \leftarrow A[min]$
        $A[min] \leftarrow t$
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Time Complexity: $\Theta(n^2)$

We will soon meet better sorting algorithms

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Properties of Sorting Algorithms

- A Sorting algorithm is:
  - **in-place** if it does not require additional memory except, perhaps, for a few units of memory

Assignment Project Exam Help
https://powcoder.com

  - **stable** if it preserves the relative order of elements with identical keys

Add WeChat powcoder

  - **input-insensitive** if its running time is fairly independent of input properties other than size

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Properties of Selection Sort

- While running time is quadratic, selection sort makes only about ***n*** **exchanges**.
- So: **selection sort is a good algorithm for sorting small collections of large records.**
- In-place?
- Stable?
- Input-insensitive?

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Properties of Selection Sort

- While running time is quadratic, selection sort makes only about ***n*** **exchanges**.
- So: **selection sort is a good algorithm for sorting small collections of large records.**

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- In-place? yes
- Stable?
- Input-insensitive?

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Properties of Selection Sort

- While running time is quadratic, selection sort makes only about ***n*** **exchanges**.

- So: **selection sort is a good algorithm for sorting small collections of large records.**

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

- In-place? yes

- Stable? ?

- Input-insensitive?

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Properties of Selection Sort

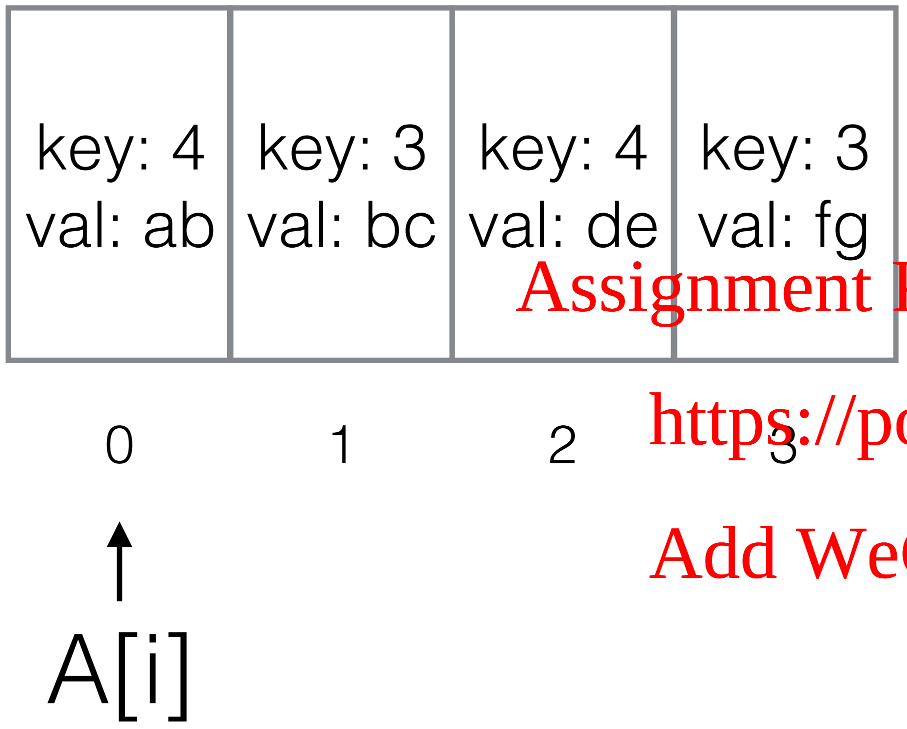
- While running time is quadratic, selection sort makes only about ***n*** **exchanges**.
- So: **selection sort is a good algorithm for sorting small collections of large records.**

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- In-place? yes
- Stable? ?
- Input-insensitive? yes

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Selection Sort Stability



key: 4 val: ab	key: 3 val: bc	key: 4 val: de	key: 3 val: fg
0	1	2	3

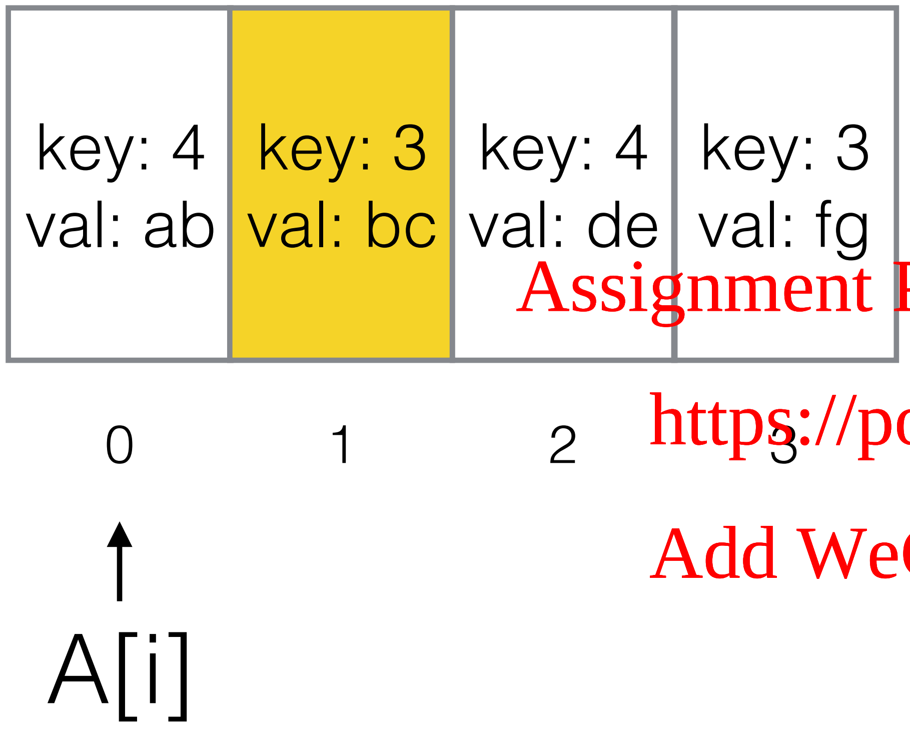
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Selection Sort Stability



key: 4 val: ab	key: 3 val: bc	key: 4 val: de	key: 3 val: fg
0	1	2	3

↑
A[i]

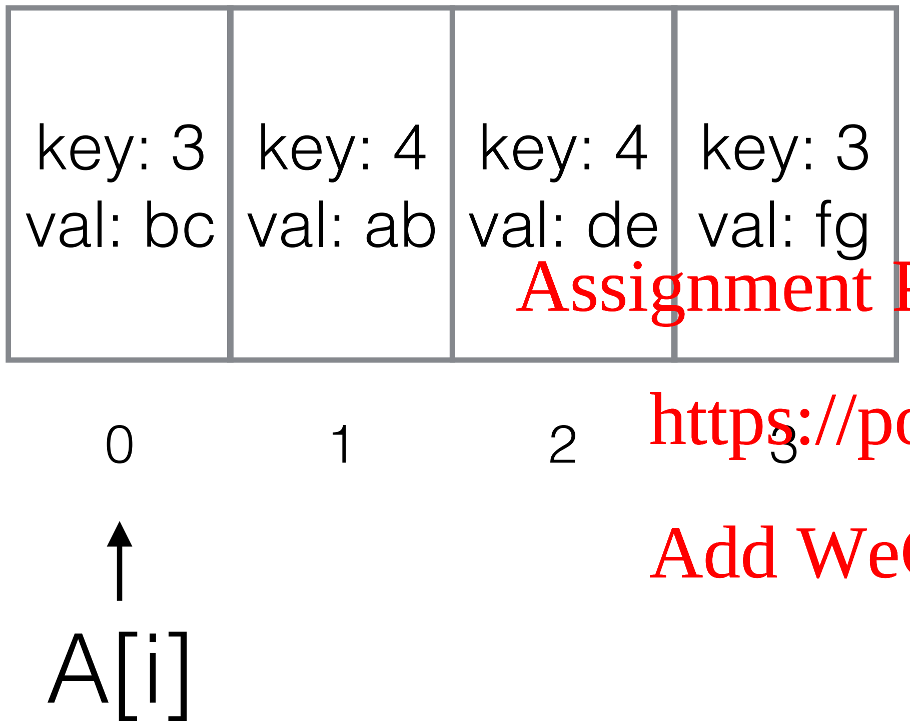
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Selection Sort Stability

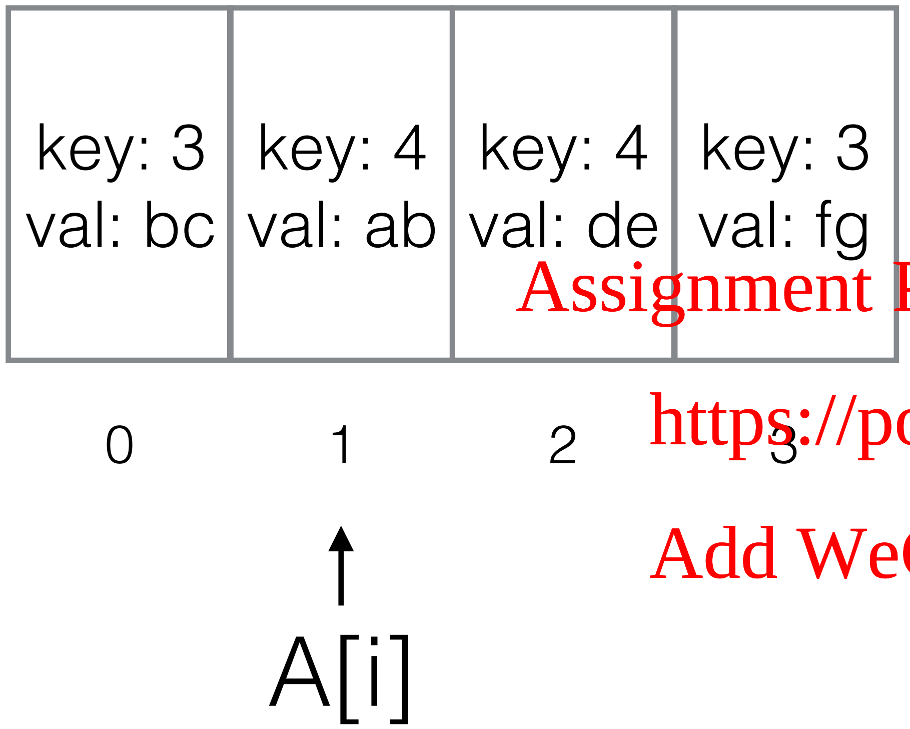


0	1	2	3
key: 3 val: bc	key: 4 val: ab	key: 4 val: de	key: 3 val: fg

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Selection Sort Stability

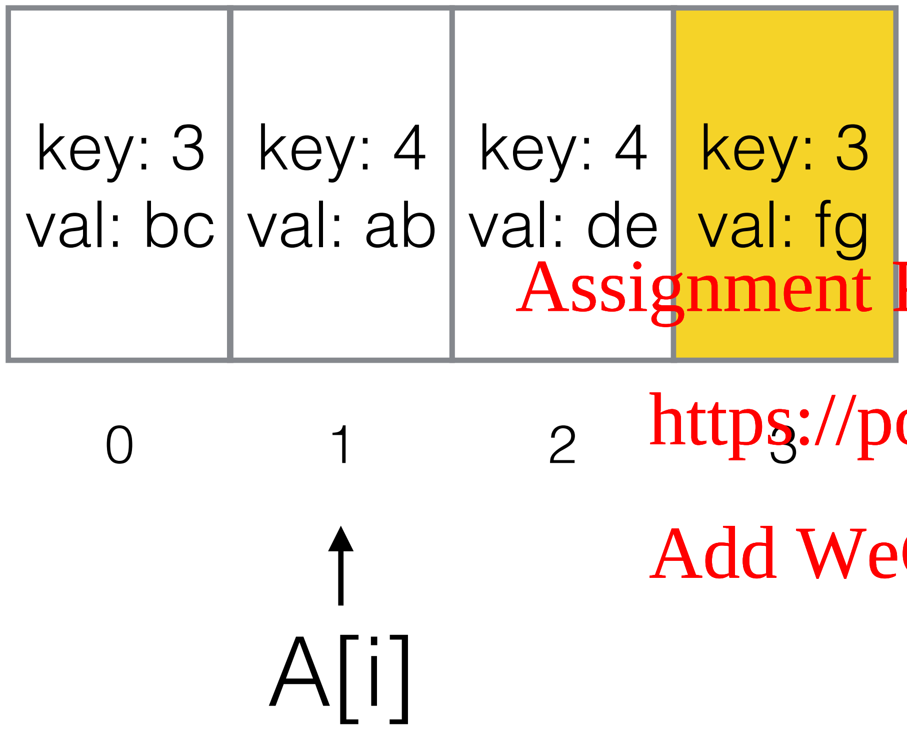


key: 3 val: bc	key: 4 val: ab	key: 4 val: de	key: 3 val: fg
0	1	2	3

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Selection Sort Stability

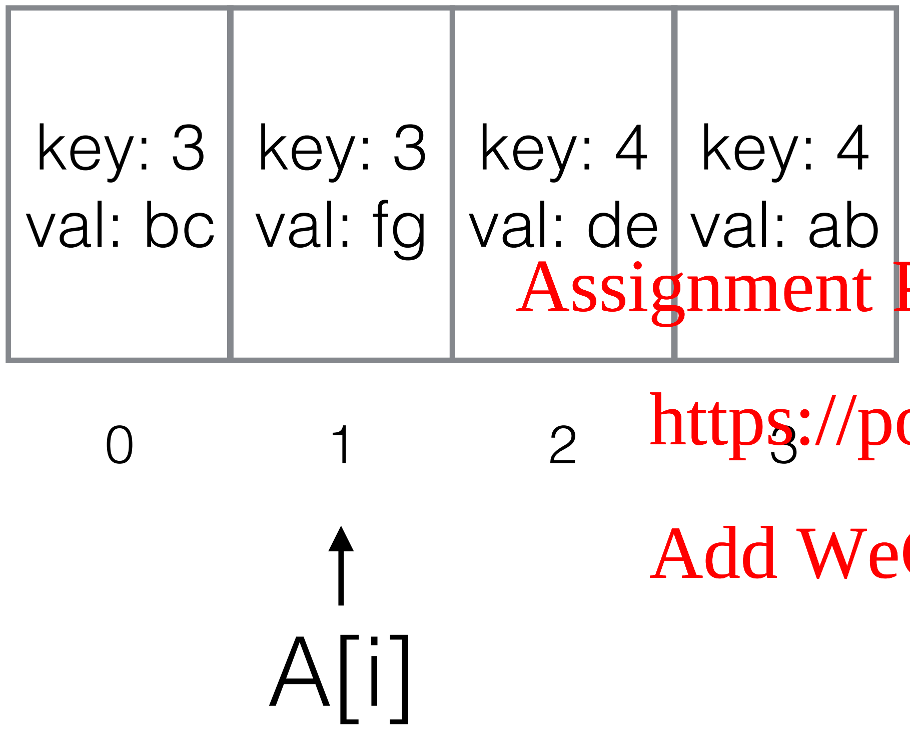


key: 3 val: bc	key: 4 val: ab	key: 4 val: de	key: 3 val: fg
0	1	2	3

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Selection Sort Stability



key: 3 val: bc	key: 3 val: fg	key: 4 val: de	key: 4 val: ab
0	1	2	3

↑

A[i]

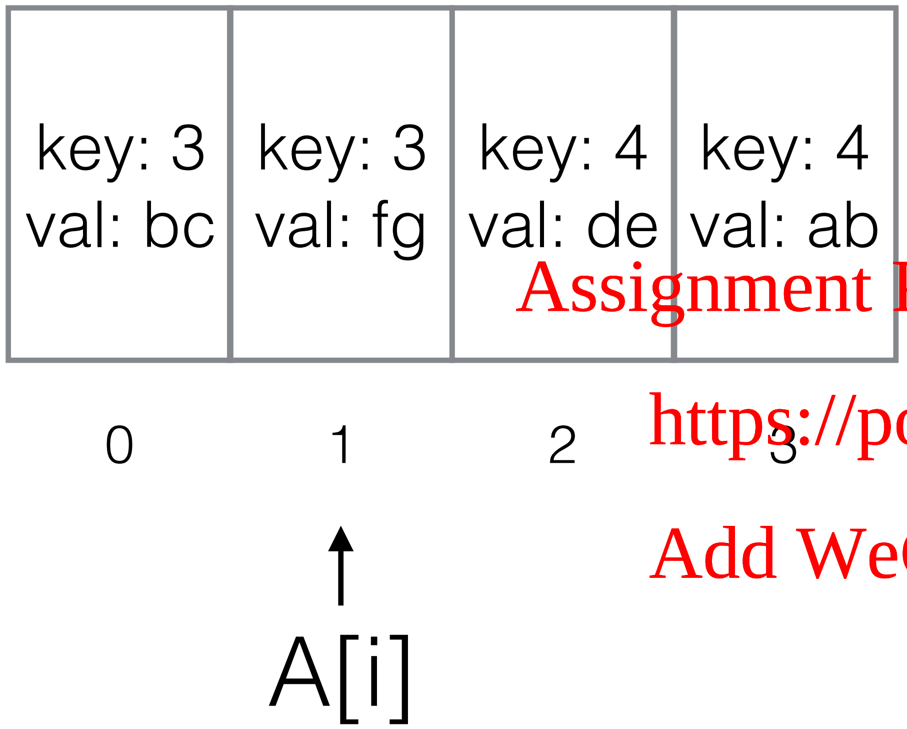
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Selection Sort Stability



key: 3 val: bc	key: 3 val: fg	key: 4 val: de	key: 4 val: ab
0	1	2	3

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

the relative order
of the two "4" records
has changed!

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Properties of Selection Sort

- While running time is quadratic, selection sort makes only about ***n*** **exchanges**.

- So: **selection sort is a good algorithm for sorting small collections of large records.**

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- In-place? yes

- Stable? no

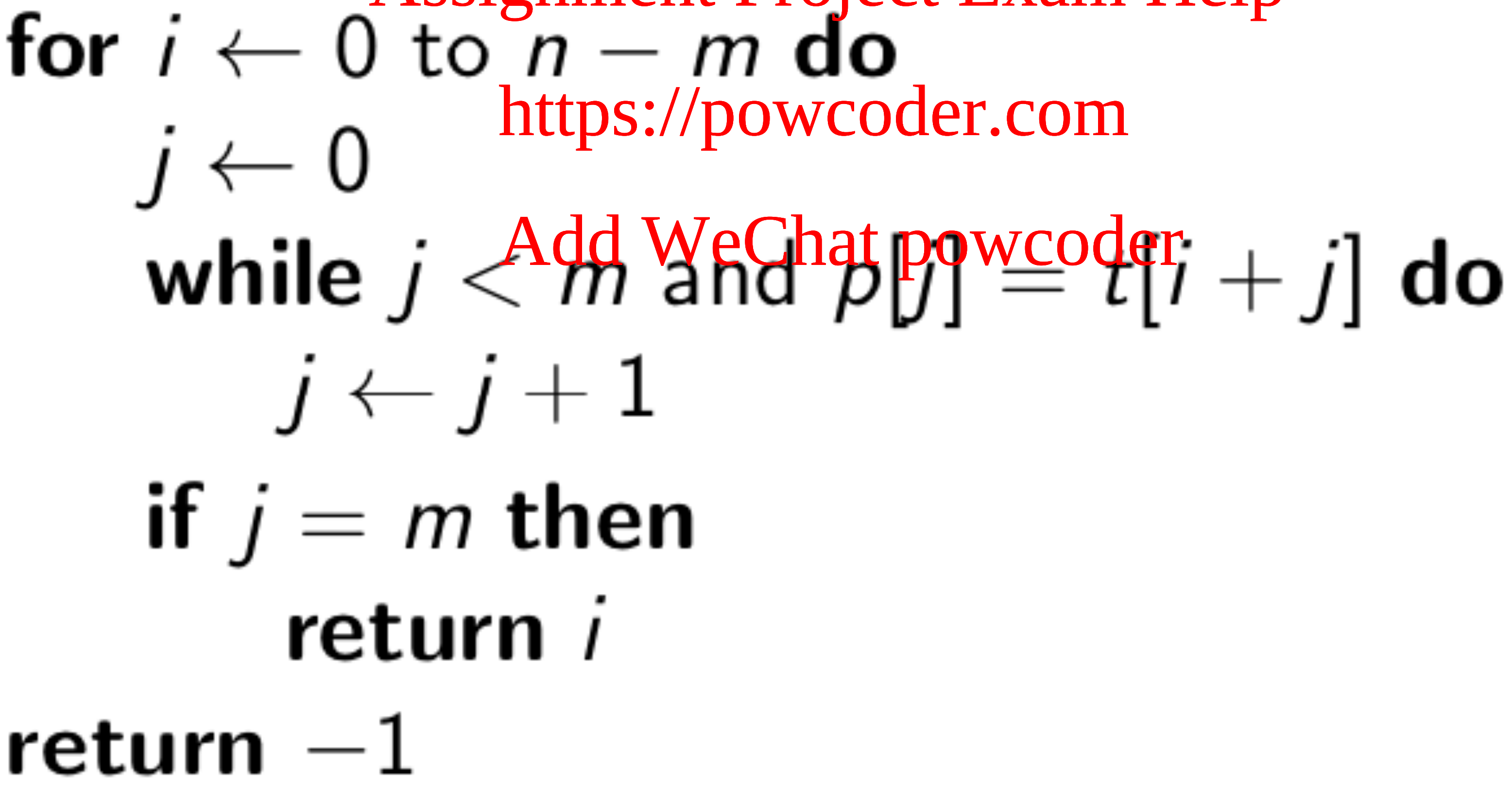
- Input-insensitive? yes

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Brute Force String Matching

- **Pattern** *p*: a string of *m* characters to search **for**
- **Text** *t*: a long string of *n* characters to search **in**
- We use *i* to run through the text and *j* to run through the pattern

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder



```
for $i \leftarrow 0$ to $n - m$ do
    $j \leftarrow 0$
    while $j < m$ and $p[j] = t[i + j]$ do
        $j \leftarrow j + 1$
    if $j = m$ then
        return $i$
return $-1$
```

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Brute Force String Matching

```
for $i \leftarrow 0$ to $n - m$ do
    $j \leftarrow 0$
    while $j < m$ and $p[j] = t[i + j]$ do
        $j \leftarrow j + 1$
    if $j = m$ then
        return $i$
return $-1$
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

t:	N	O	B	O	D	Y		K	N	O	W	S	
	0	1	2	3	4	5	6	7	8	9	10	11	m

p:	N	O	T	
	0	1	2	n

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

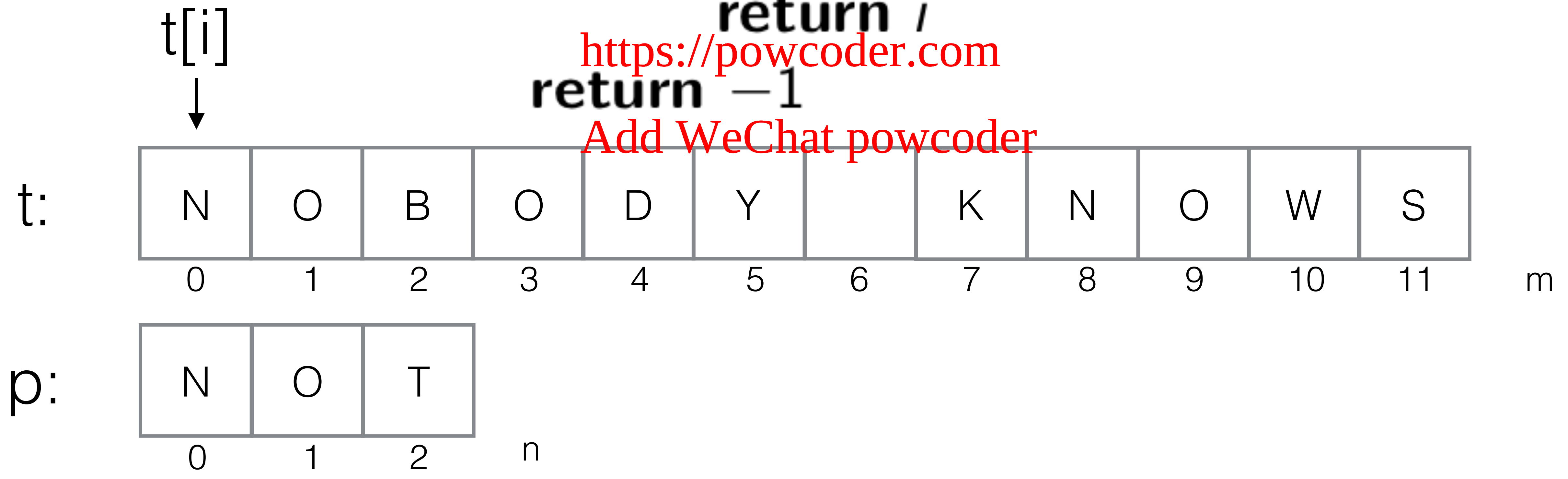
Brute Force String Matching

```
for $i \leftarrow 0$ to $n - m$ do
    $j \leftarrow 0$
    while $j < m$ and $p[j] = t[i + j]$ do
        $j \leftarrow j + 1$
    if $j = m$ then
        return $i$
return $-1$
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder



t[i]

t:	N	O	B	O	D	Y		K	N	O	W	S	
	0	1	2	3	4	5	6	7	8	9	10	11	m

p:	N	O	T	
	0	1	2	n

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

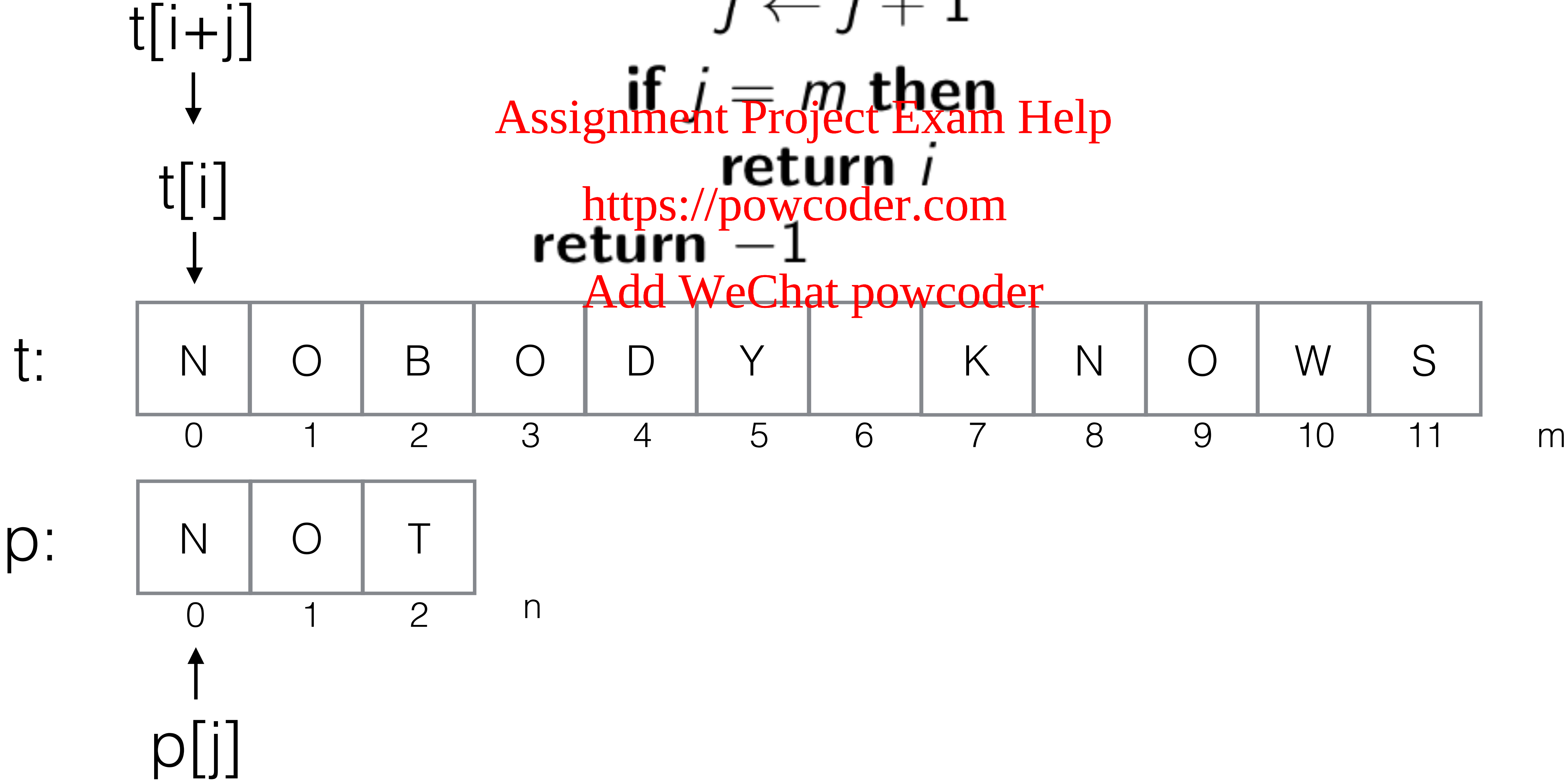
Brute Force String Matching

```
for $i \leftarrow 0$ to $n - m$ do
    $j \leftarrow 0$
    while $j < m$ and $p[j] = t[i + j]$ do
        $j \leftarrow j + 1$
    if $j = m$ then
        return $i$
return $-1$
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder



t:												
	N	O	B	O	D	Y		K	N	O	W	S
	0	1	2	3	4	5	6	7	8	9	10	11

p:			
	N	O	T
	0	1	2

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

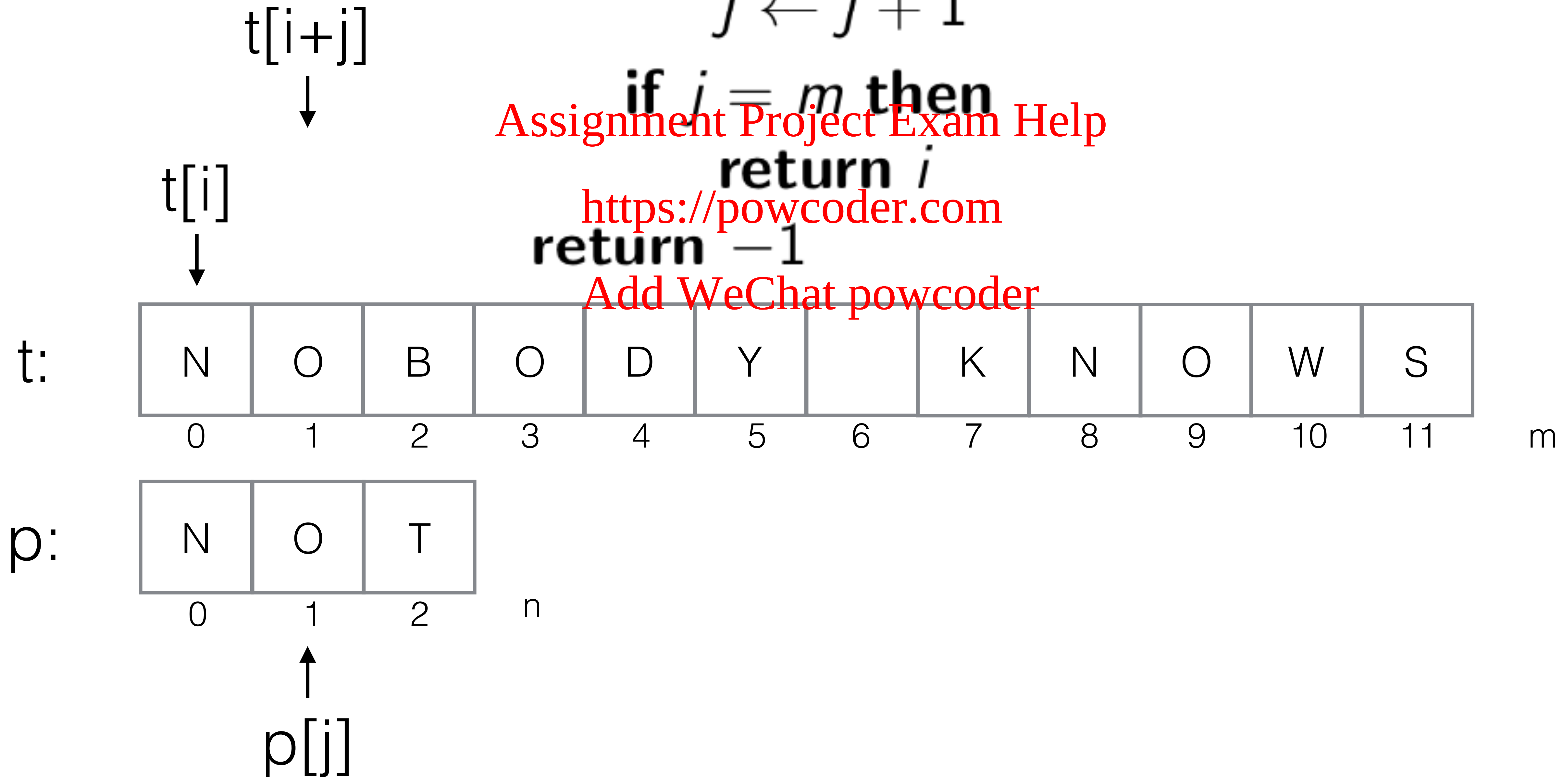
Brute Force String Matching

```
for $i \leftarrow 0$ to $n - m$ do
    $j \leftarrow 0$
    while $j < m$ and $p[j] = t[i + j]$ do
        $j \leftarrow j + 1$
    if $j = m$ then
        return $i$
return $-1$
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

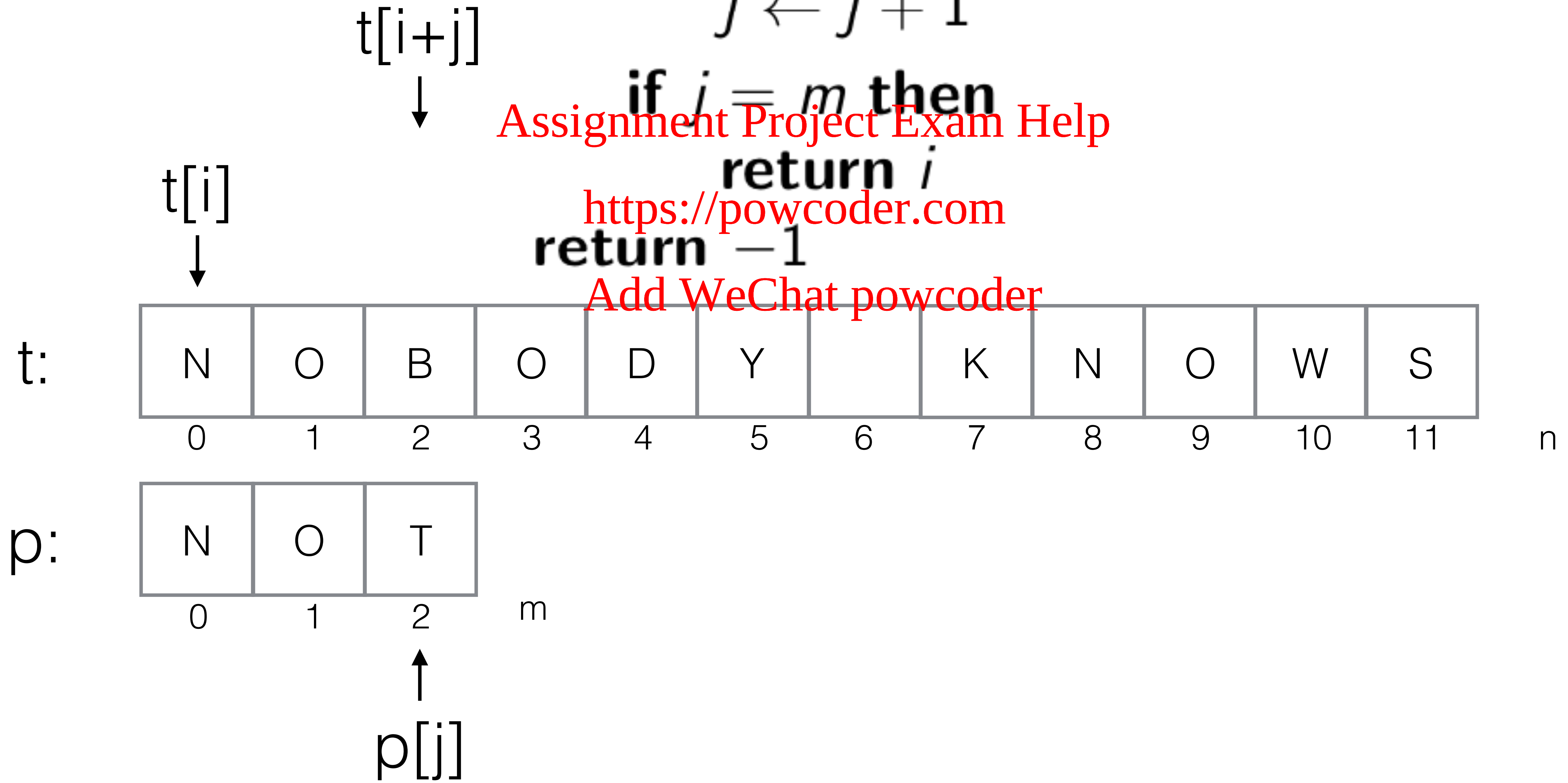
Brute Force String Matching

```
**for** $i \leftarrow 0$ to $n - m$ **do**
    $j \leftarrow 0$
    **while** $j < m$ and $p[j] = t[i + j]$ **do**
        $j \leftarrow j + 1$
    **if** $j = m$ **then**
        **return** $i$
**return** $-1$
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder



t:	N	O	B	O	D	Y		K	N	O	W	S	
	0	1	2	3	4	5	6	7	8	9	10	11	n

p:	N	O	T	
	0	1	2	m

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

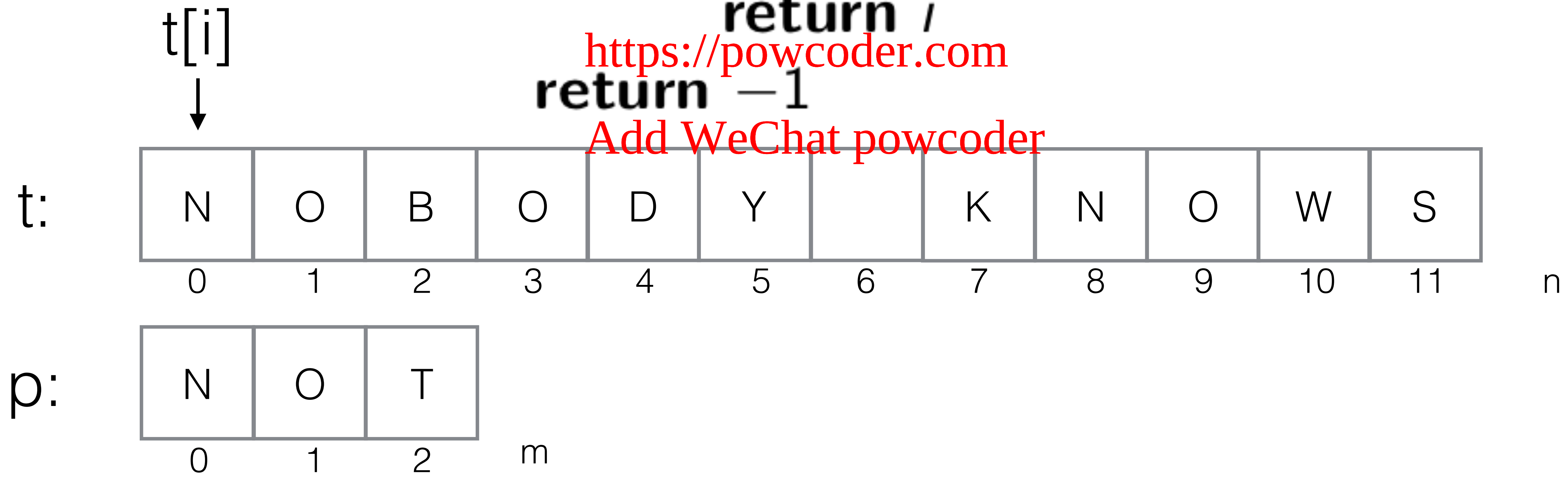
Brute Force String Matching

```
for $i \leftarrow 0$ to $n - m$ do
    $j \leftarrow 0$
    while $j < m$ and $p[j] = t[i + j]$ do
        $j \leftarrow j + 1$
    if $j = m$ then
        return $i$
return $-1$
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder



t:

N	O	B	O	D	Y		K	N	O	W	S
0	1	2	3	4	5	6	7	8	9	10	11

p:

N	O	T
0	1	2

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

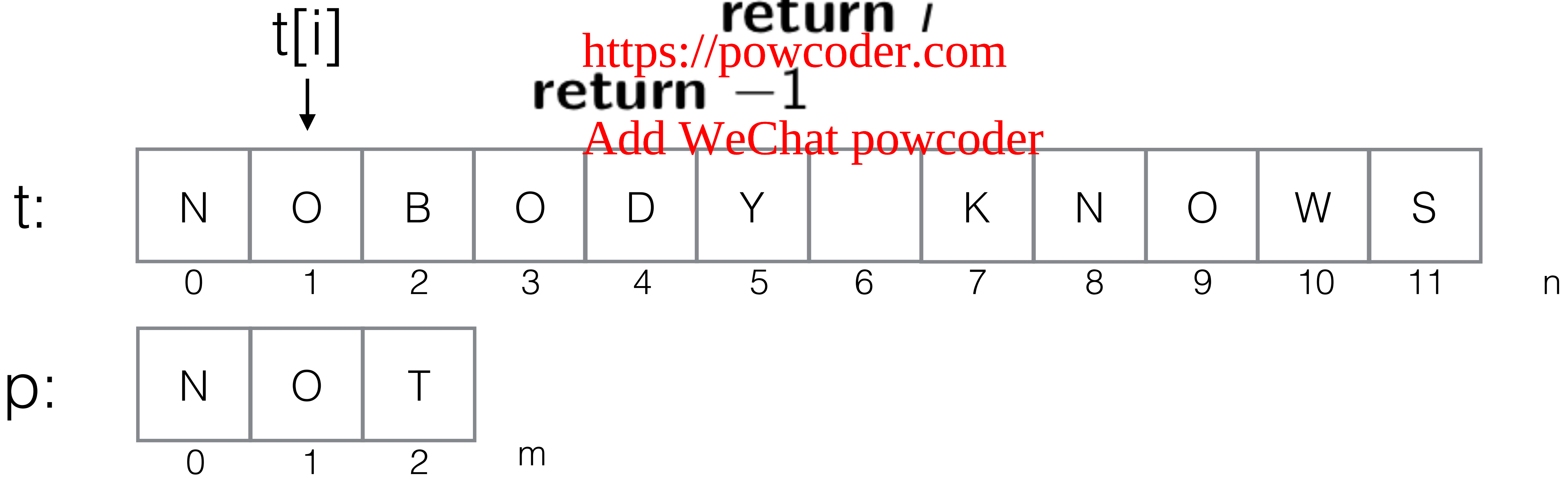
Brute Force String Matching

```
for $i \leftarrow 0$ to $n - m$ do
    $j \leftarrow 0$
    while $j < m$ and $p[j] = t[i + j]$ do
        $j \leftarrow j + 1$
    if $j = m$ then
        return $i$
return $-1$
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder



t:	0	1	2	3	4	5	6	7	8	9	10	11
	N	O	B	O	D	Y		K	N	O	W	S

p:	0	1	2
	N	O	T

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

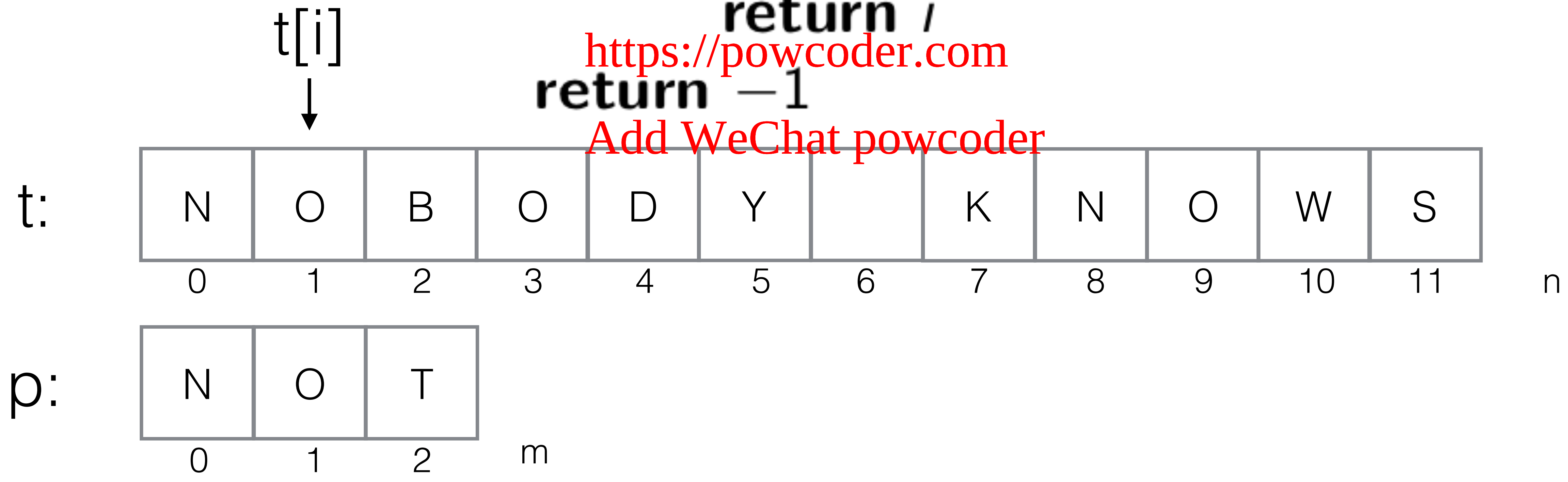
Brute Force String Matching

```
for $i \leftarrow 0$ to $n - m$ do
    $j \leftarrow 0$
    while $j < m$ and $p[j] = t[i + j]$ do
        $j \leftarrow j + 1$
    if $j = m$ then
        return $i$
return $-1$
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder



t:	N	O	B	O	D	Y		K	N	O	W	S	
	0	1	2	3	4	5	6	7	8	9	10	11	n

p:	N	O	T	
	0	1	2	m

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

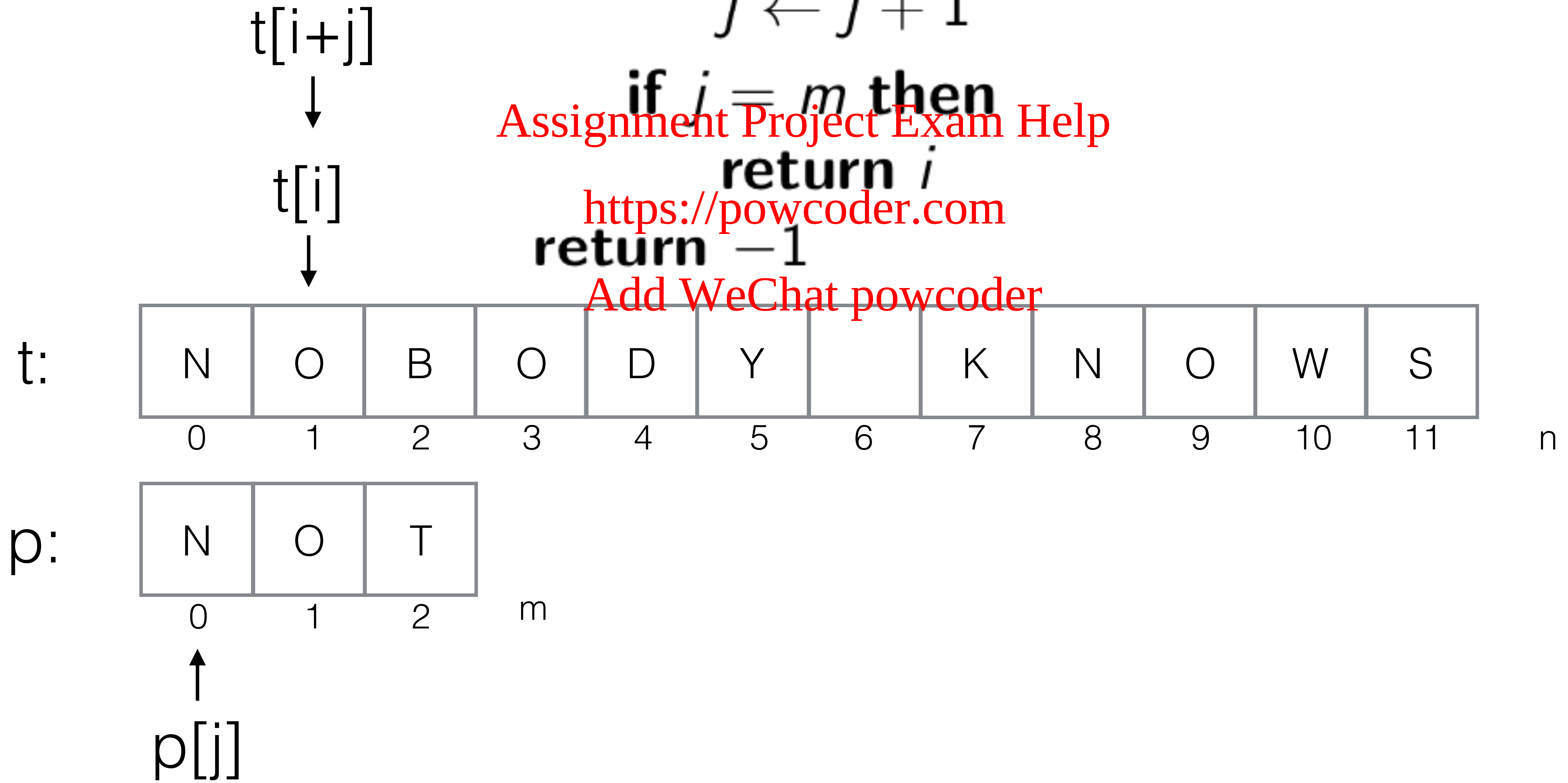
Brute Force String Matching

```
for $i \leftarrow 0$ to $n - m$ do
    $j \leftarrow 0$
    while $j < m$ and $p[j] = t[i + j]$ do
        $j \leftarrow j + 1$
    if $j = m$ then
        return $i$
return $-1$
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder



t:

N	O	B	O	D	Y		K	N	O	W	S
0	1	2	3	4	5	6	7	8	9	10	11

p:

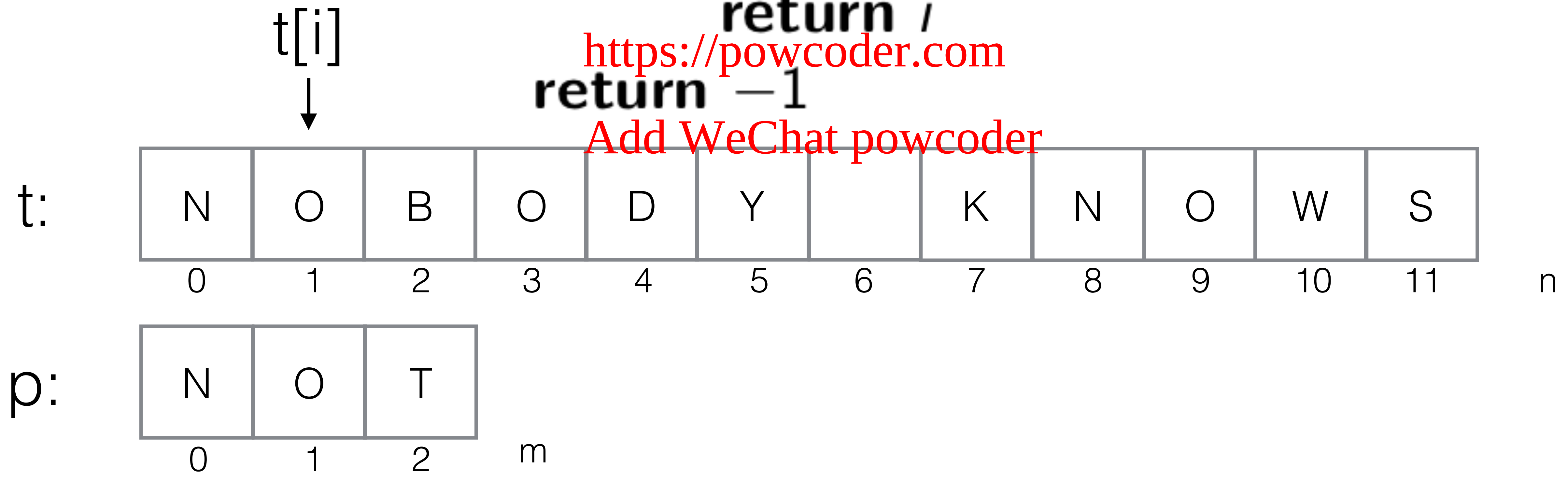
N	O	T
0	1	2

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Brute Force String Matching

```
for $i \leftarrow 0$ to $n - m$ do
    $j \leftarrow 0$
    while $j < m$ and $p[j] = t[i + j]$ do
        $j \leftarrow j + 1$
    if $j = m$ then
        return $i$
return $-1$
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder



t[i]

t:	N	O	B	O	D	Y		K	N	O	W	S	
	0	1	2	3	4	5	6	7	8	9	10	11	n

p:	N	O	T	
	0	1	2	m

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

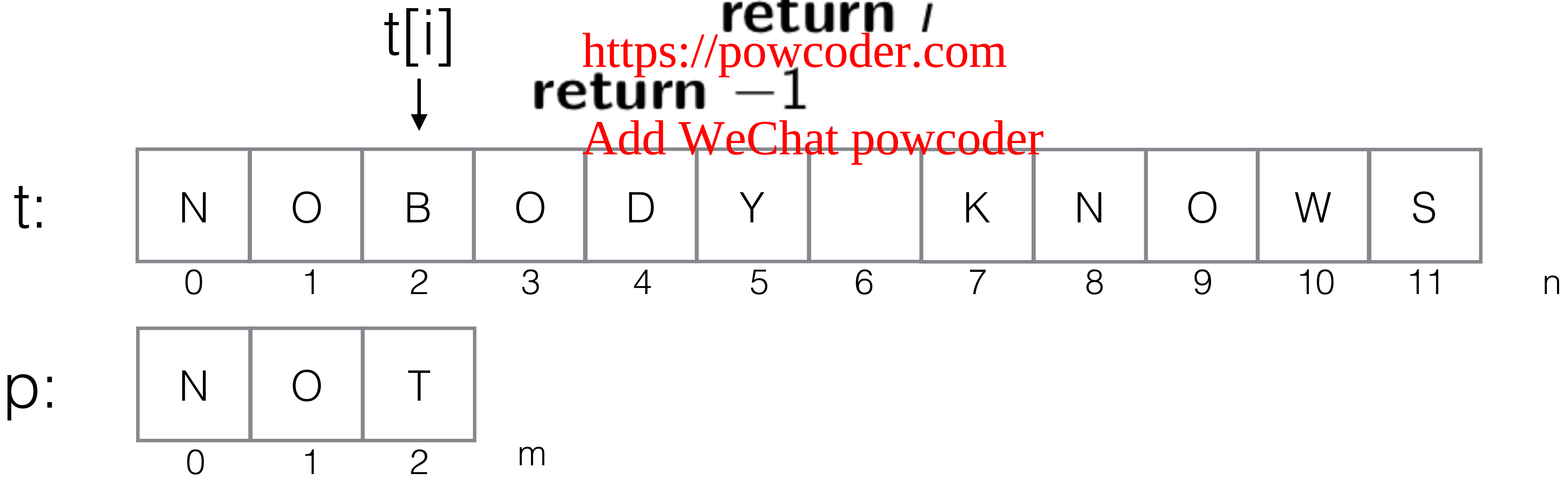
Brute Force String Matching

```
for $i \leftarrow 0$ to $n - m$ do
    $j \leftarrow 0$
    while $j < m$ and $p[j] = t[i + j]$ do
        $j \leftarrow j + 1$
    if $j = m$ then
        return $i$
return $-1$
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder



t:	N	O	B	O	D	Y		K	N	O	W	S	
	0	1	2	3	4	5	6	7	8	9	10	11	n

p:	N	O	T	
	0	1	2	m

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Analysing Brute Force String Matching

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Analysing Brute Force String Matching

Basic operation:

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Analysing Brute Force String Matching

Basic operation: comparison p[j] = t[i+j]

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Analysing Brute Force String Matching

Basic operation: comparison p[j] = t[i+j]

For each n - m + 1 positions in *t*
we make up to m comparisons

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Analysing Brute Force String Matching

THE UNIVERSITY OF MELBOURNE

Basic operation: comparison p[j] = t[i+j]

For each n - m + 1 positions in *t*
we make up to m comparisons

Assuming m much larger than n: $O(mn)$ comparisons.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Analysing Brute Force String Matching

Basic operation: comparison p[j] = t[i+j]

For each n - m + 1 positions in *t*
we make up to m comparisons

Assuming m much larger than n: *O*(*mn*) comparisons.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

But for **random text** over reasonably large alphabet (e.g. English), **average** running time is linear in *n*

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Analysing Brute Force String Matching

THE UNIVERSITY OF MELBOURNE

Basic operation: comparison p[j] = t[i+j]

For each n - m + 1 positions in *t*
we make up to m comparisons

Assignment Project Exam Help

Assuming m much larger than n: *O*(*mn*) comparisons.

https://powcoder.com

Add WeChat powcoder

But for **random text** over reasonably large alphabet (e.g. English), **average** running time is linear in *n*

There are better algorithms, for smaller alphabets such as binary strings or strings of DNA nucleobases. But for many purposes, the brute-force algorithm is acceptable.

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Brute Force Geometric Algorithms: Closest Pair

- **Problem:** Given n points is k-dimensional space, find a pair of points with minimal separating Euclidean distance.

Assignment Project Exam Help

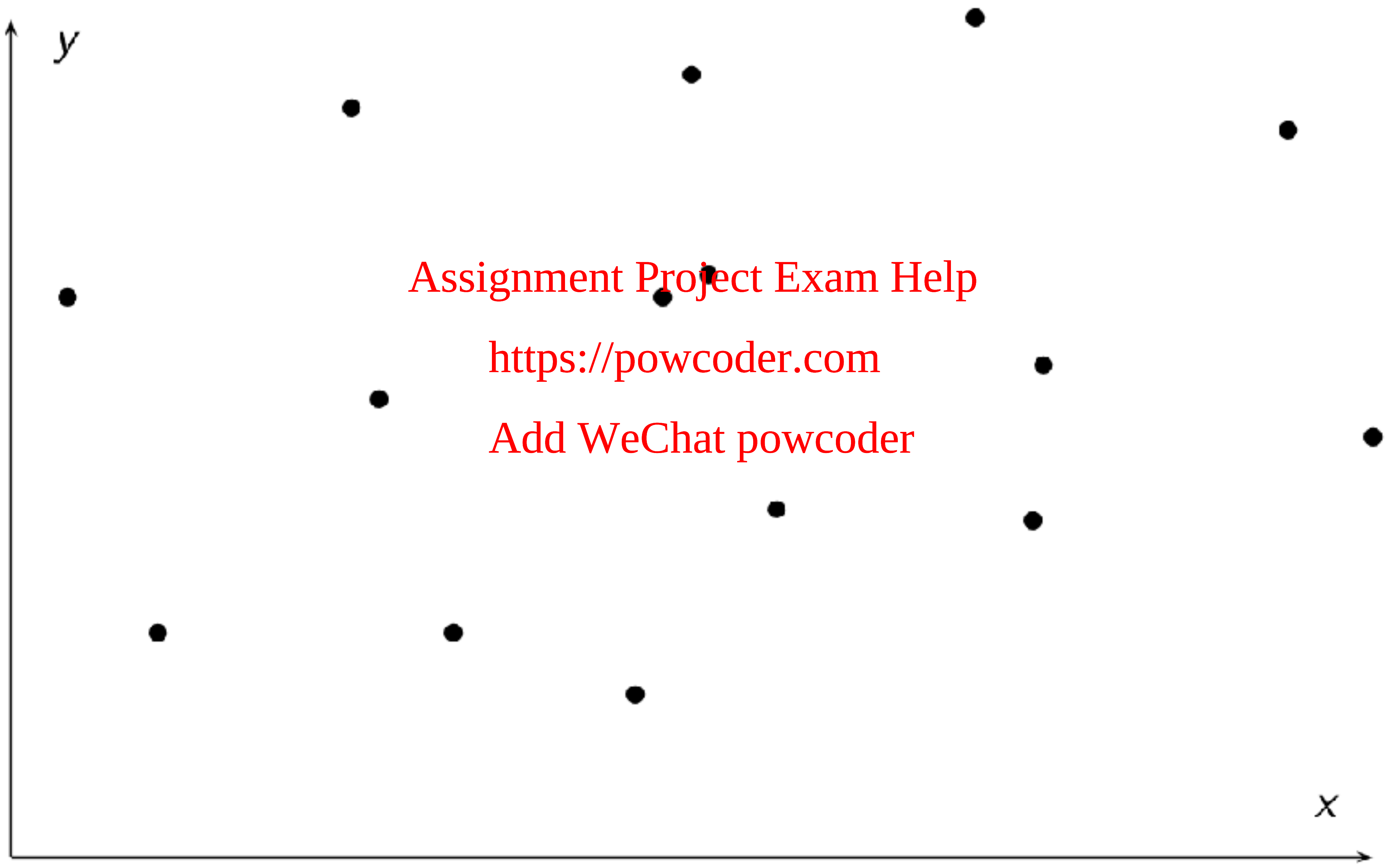
https://powcoder.com

Add WeChat powcoder

- The brute force approach considers each pair in turn (except that once it has found the distance from x to y, it does not need to consider the distance from y to x).
- For simplicity, we look at the 2-dimensional case, the points being $(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})$.

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Closest Pair Problem (2D)



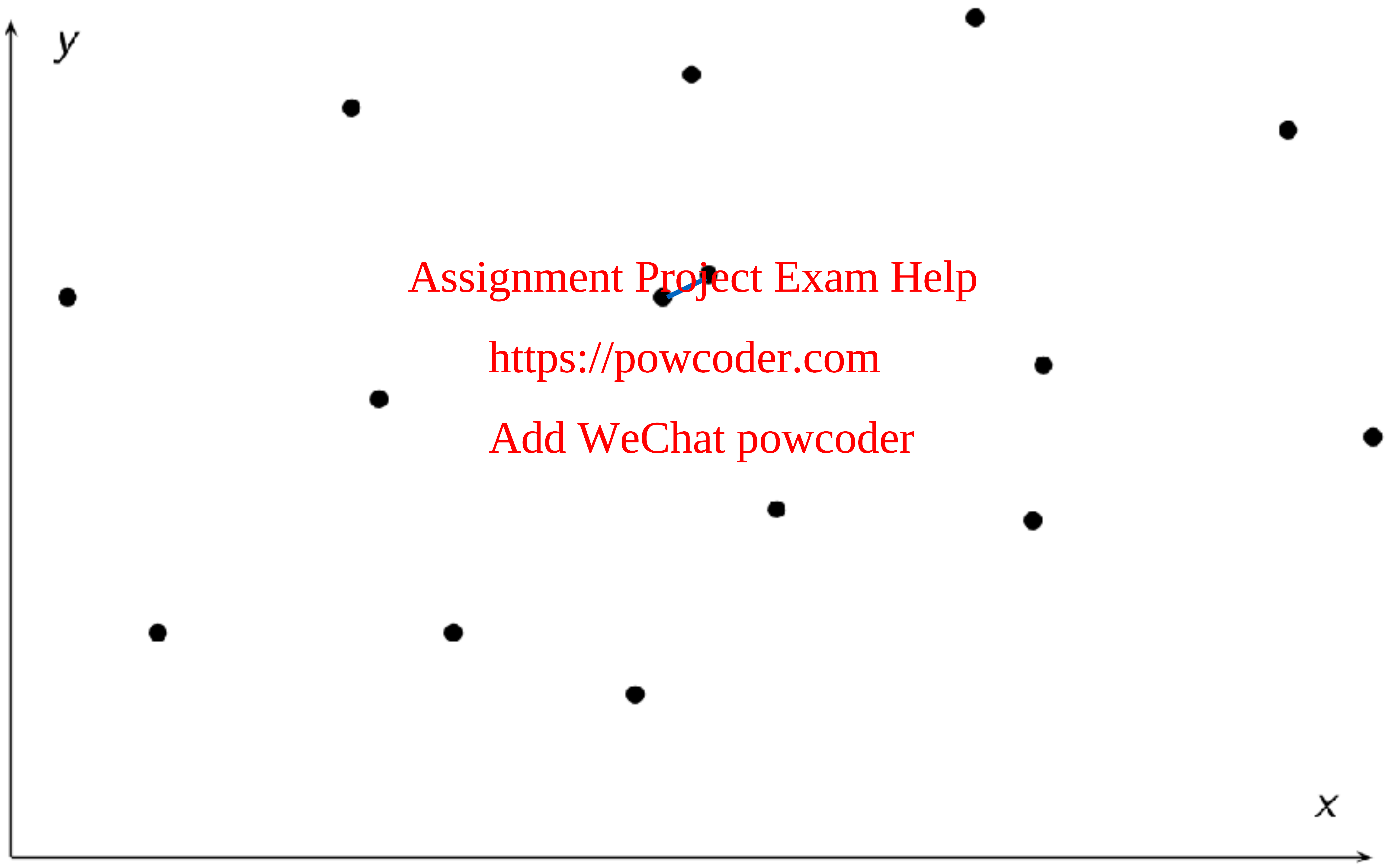
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Closest Pair Problem (2D)



Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Brute Force Closest Pair

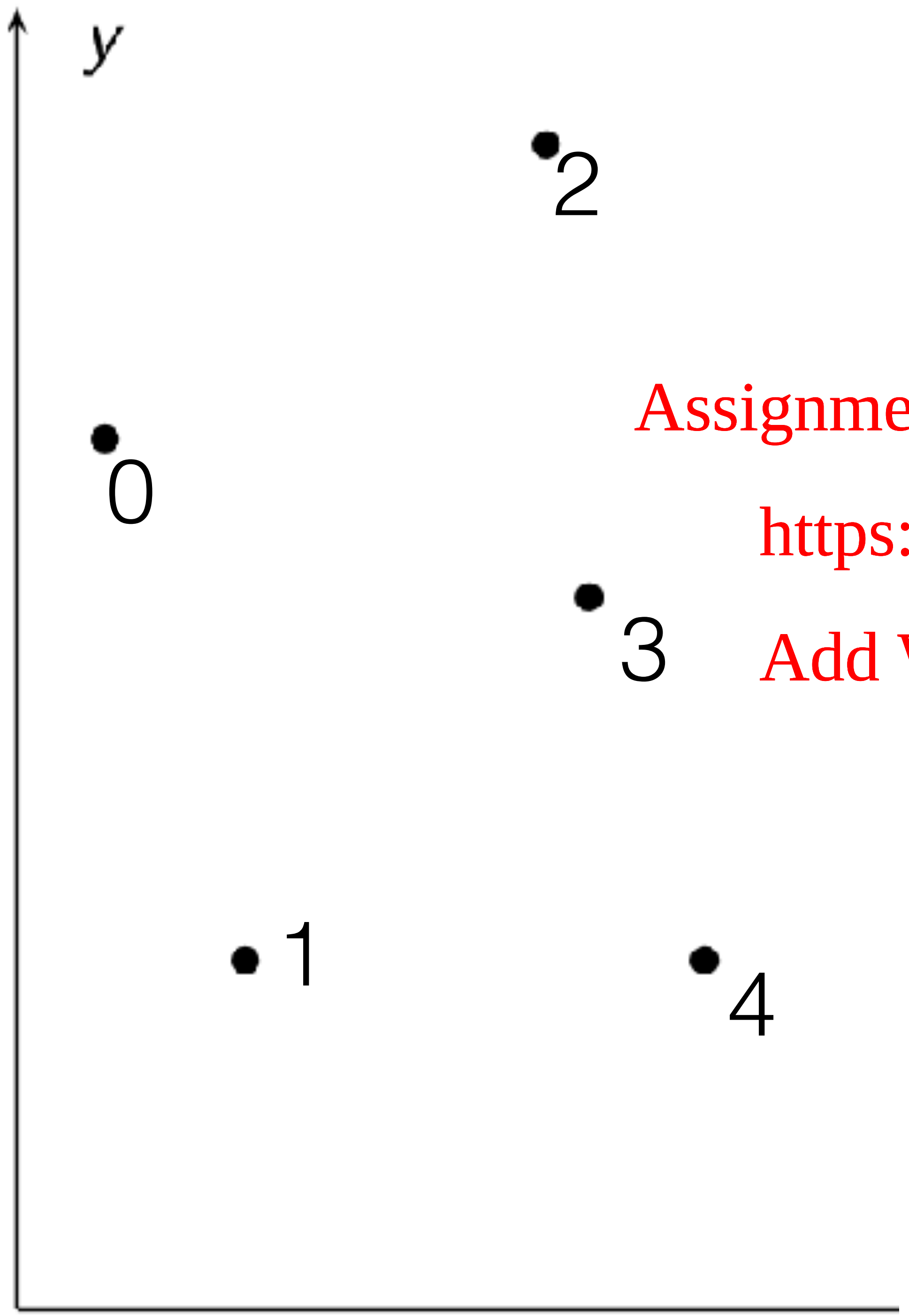
Try all combinations (x_i,y_i) and (x_j,y_j) with $i < j$:

```
$min \leftarrow \infty$
for $i \leftarrow 0$ to $n-2$ do
    for $j \leftarrow i+1$ to $n-1$ do
        $d \leftarrow sqrt((x_i - x_j)^2 + (y_i - y_j)^2)$        ▷ Distance for this pair
        if $d < min$ then
            $min \leftarrow d$                                      ▷ Smallest distance so far
            $p_1 \leftarrow i$                               ▷ Remember this (i,j) combination
            $p_2 \leftarrow j$
return $p_1, p_2$
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Closest Pair Problem (2D)



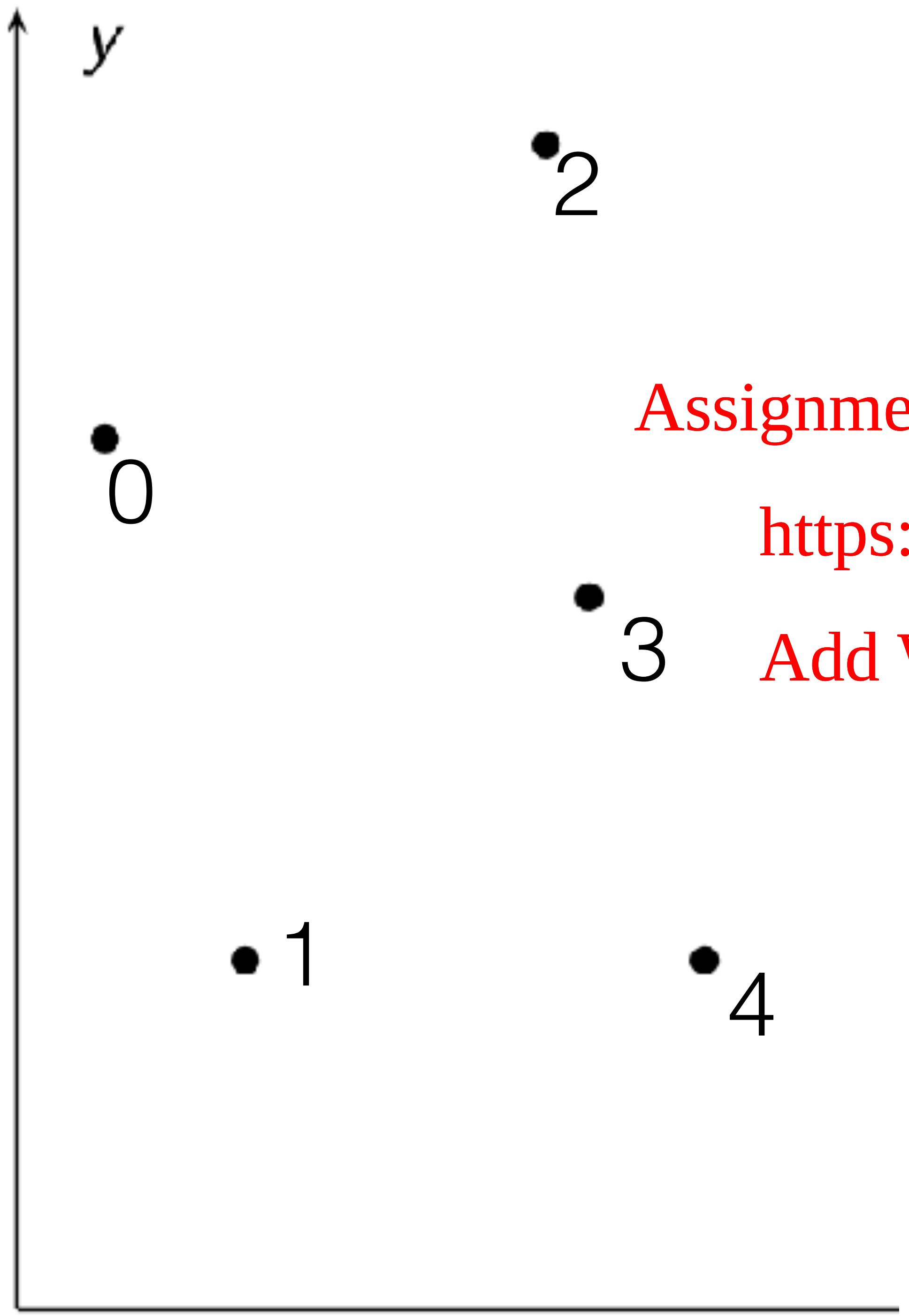
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Closest Pair Problem (2D)



i=0

Assignment Project Exam Help

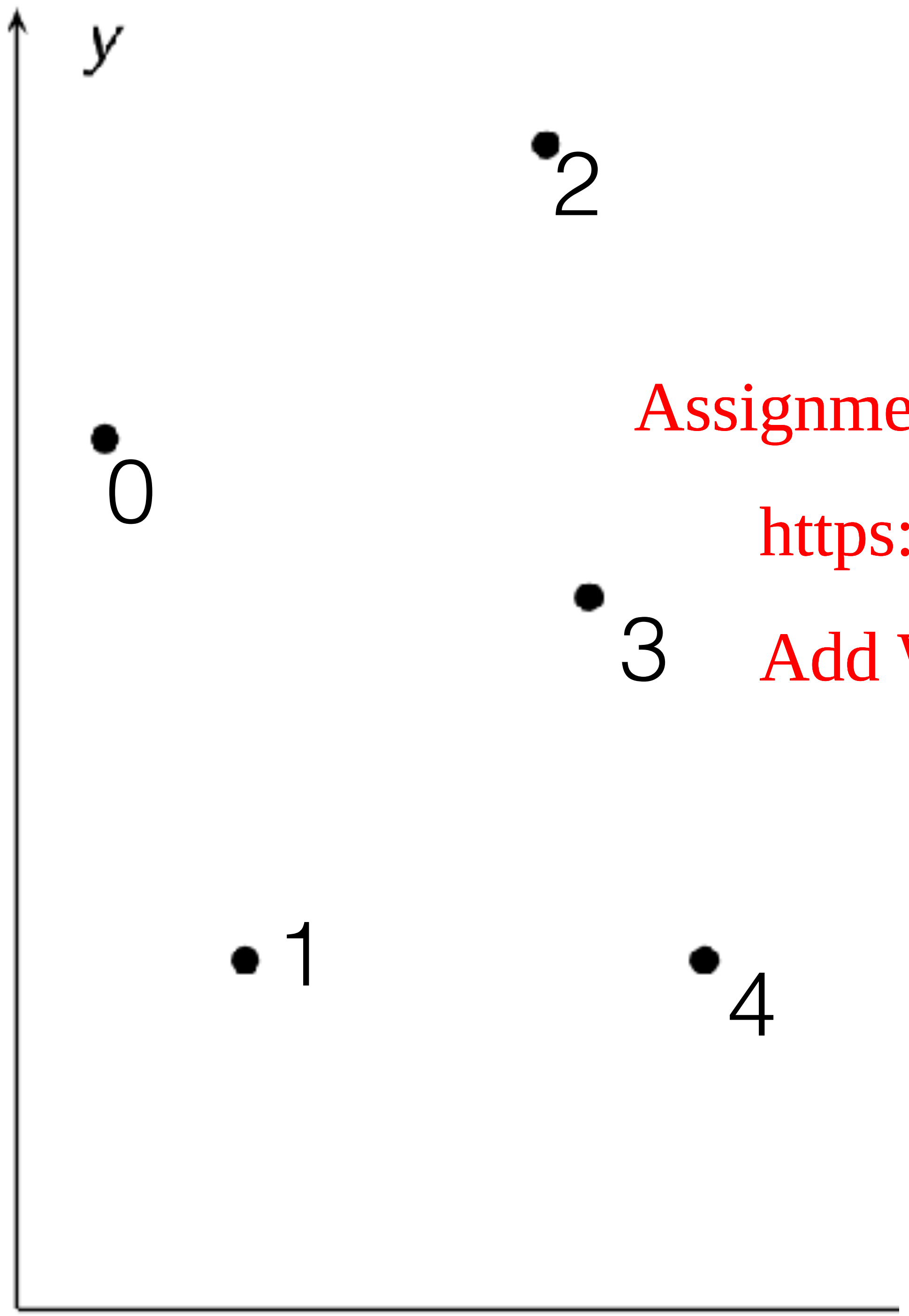
https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Closest Pair Problem (2D)

i=0   min=∞



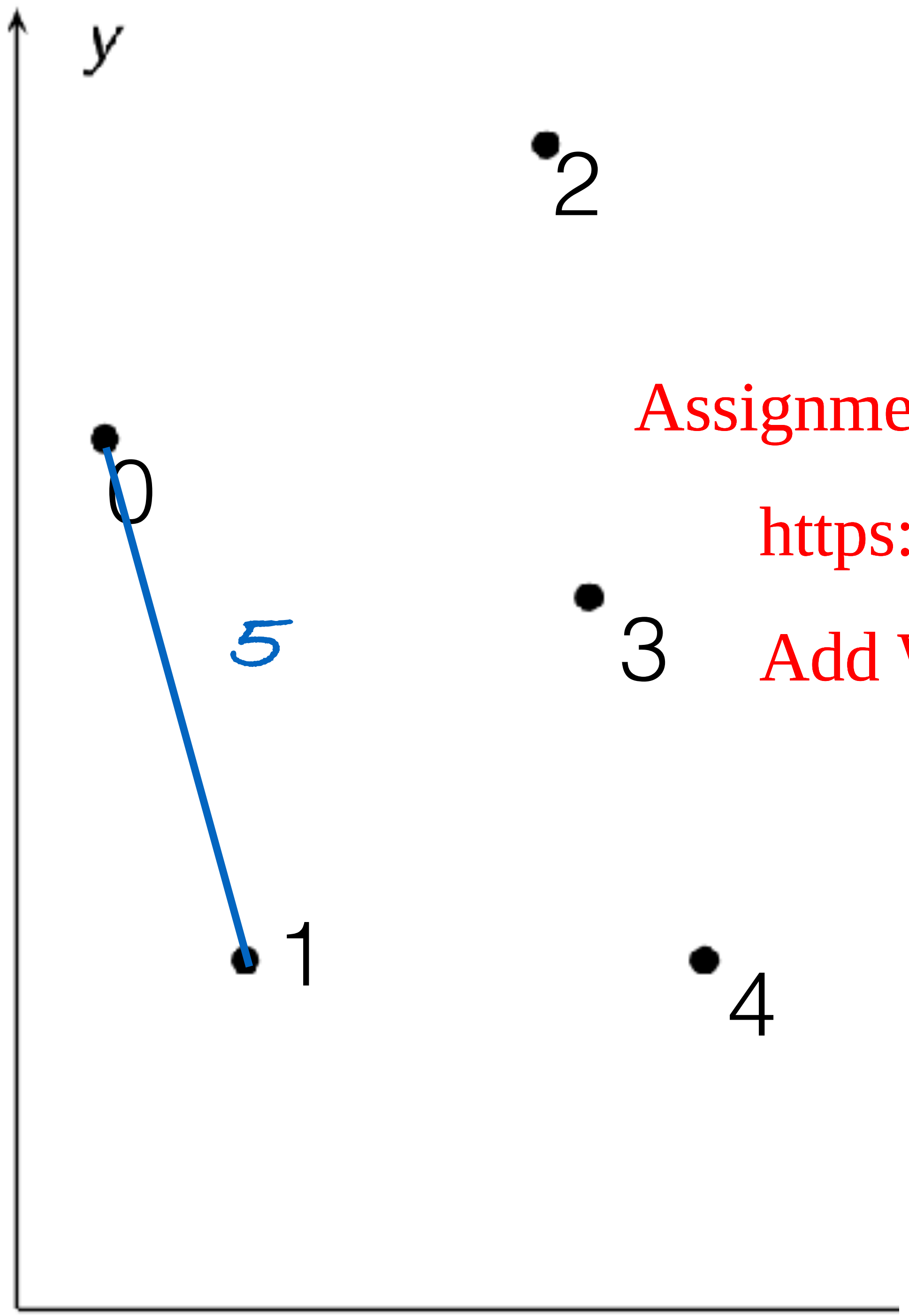
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Closest Pair Problem (2D)



i=0   min=∞

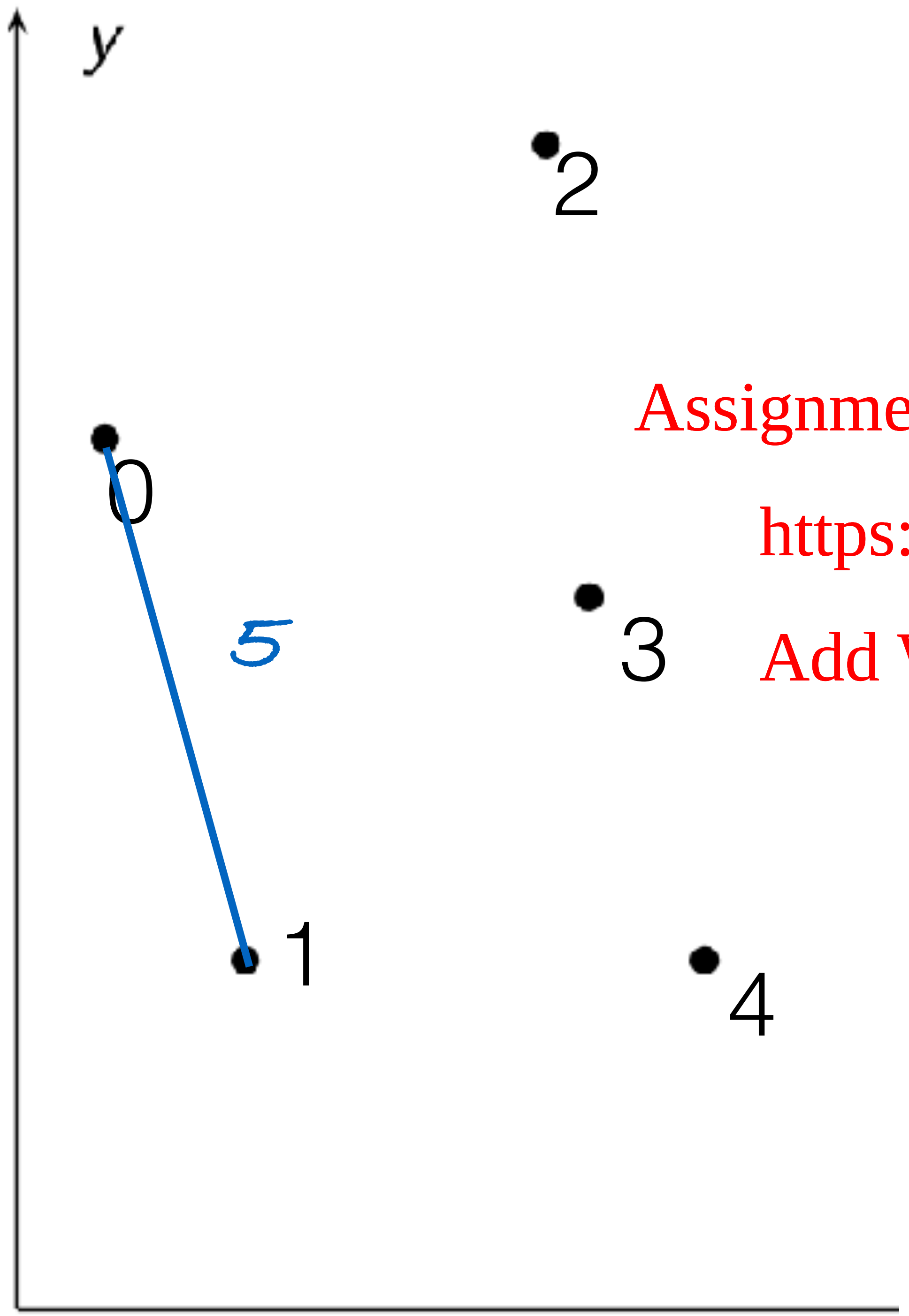
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Closest Pair Problem (2D)



i=0   min=5

Assignment Project Exam Help

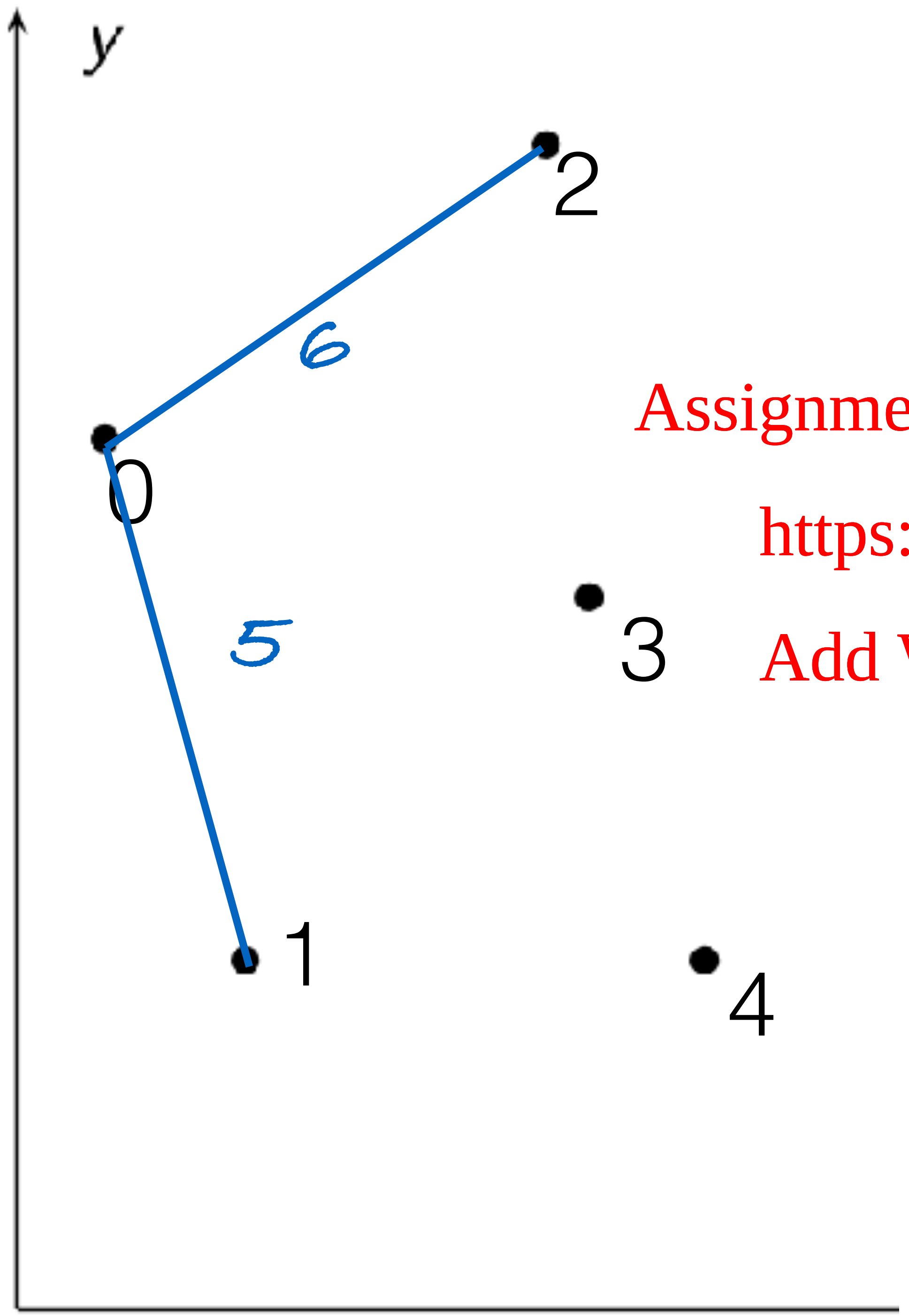
https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Closest Pair Problem (2D)

i=0 min=5



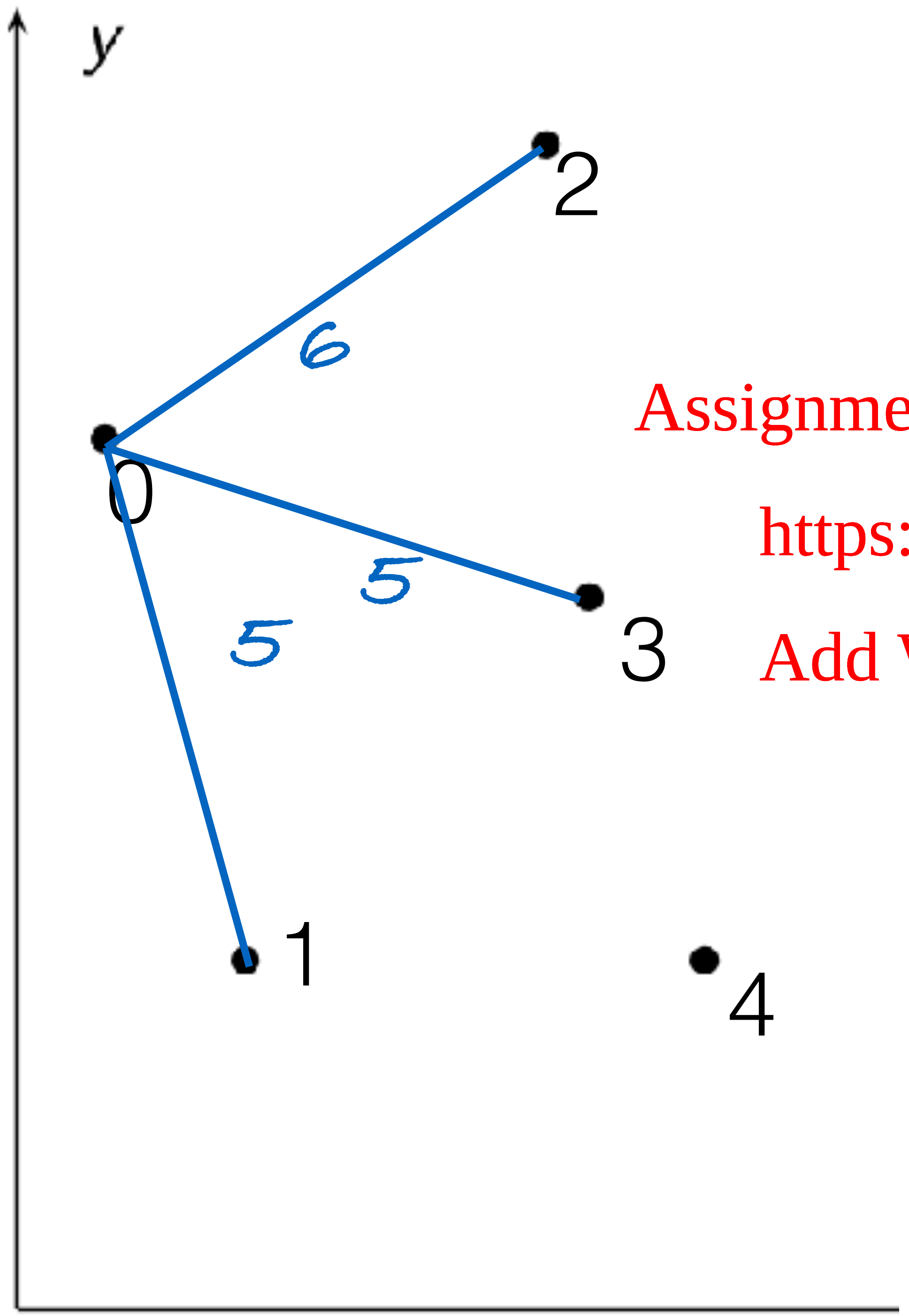
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Closest Pair Problem (2D)



i=0   min=5

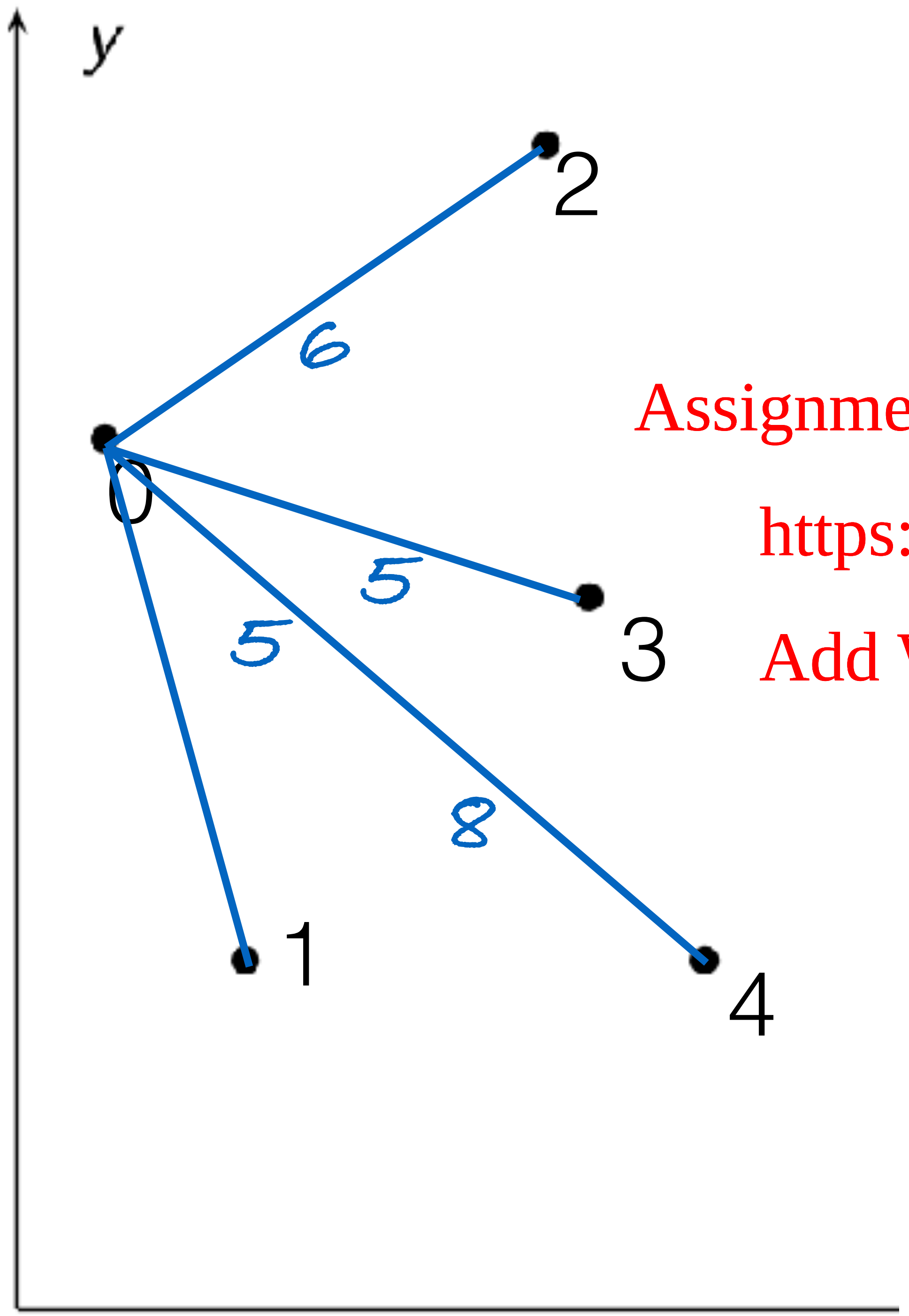
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Closest Pair Problem (2D)



i=0 min=5

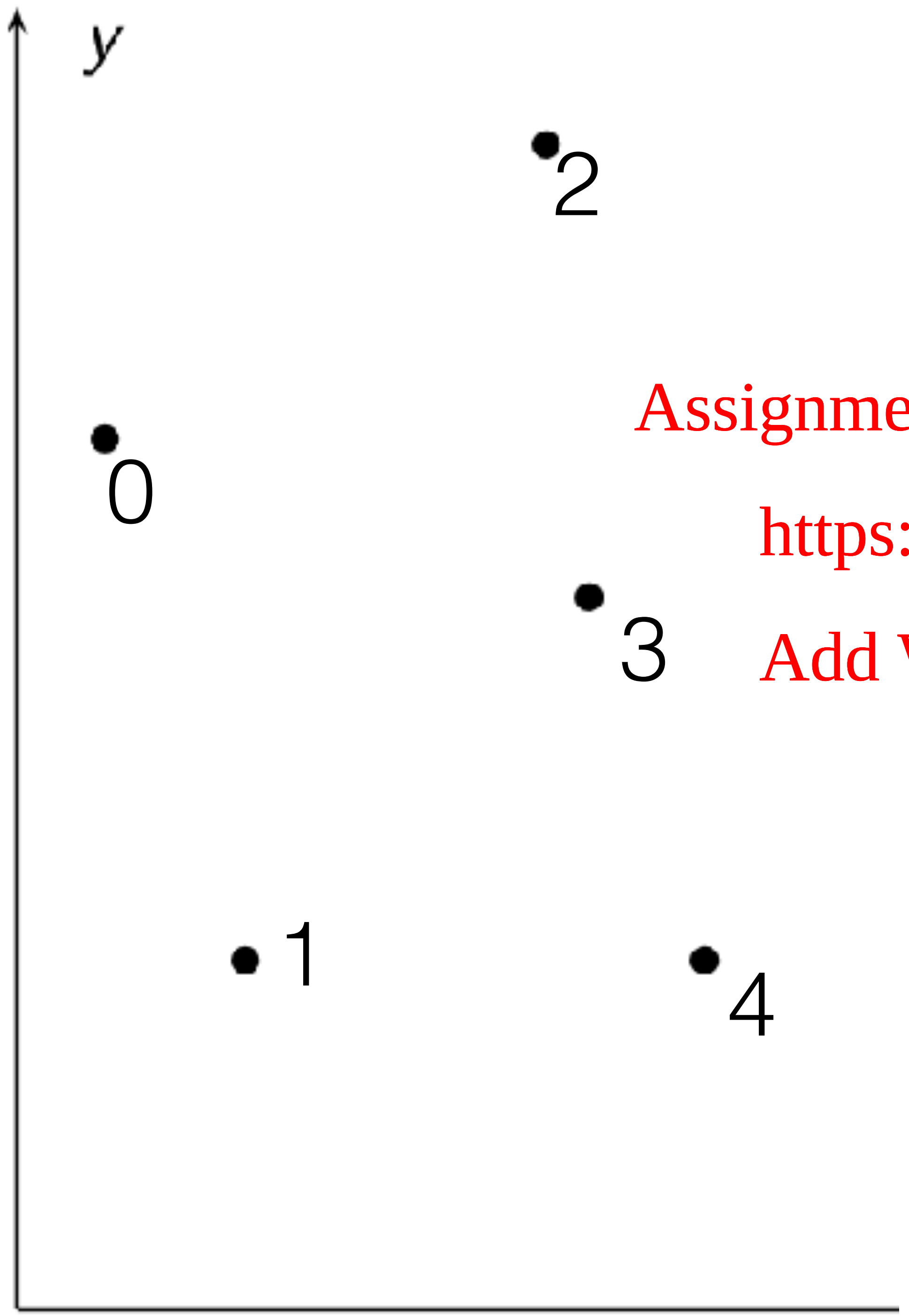
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Closest Pair Problem (2D)



i=1 min=5

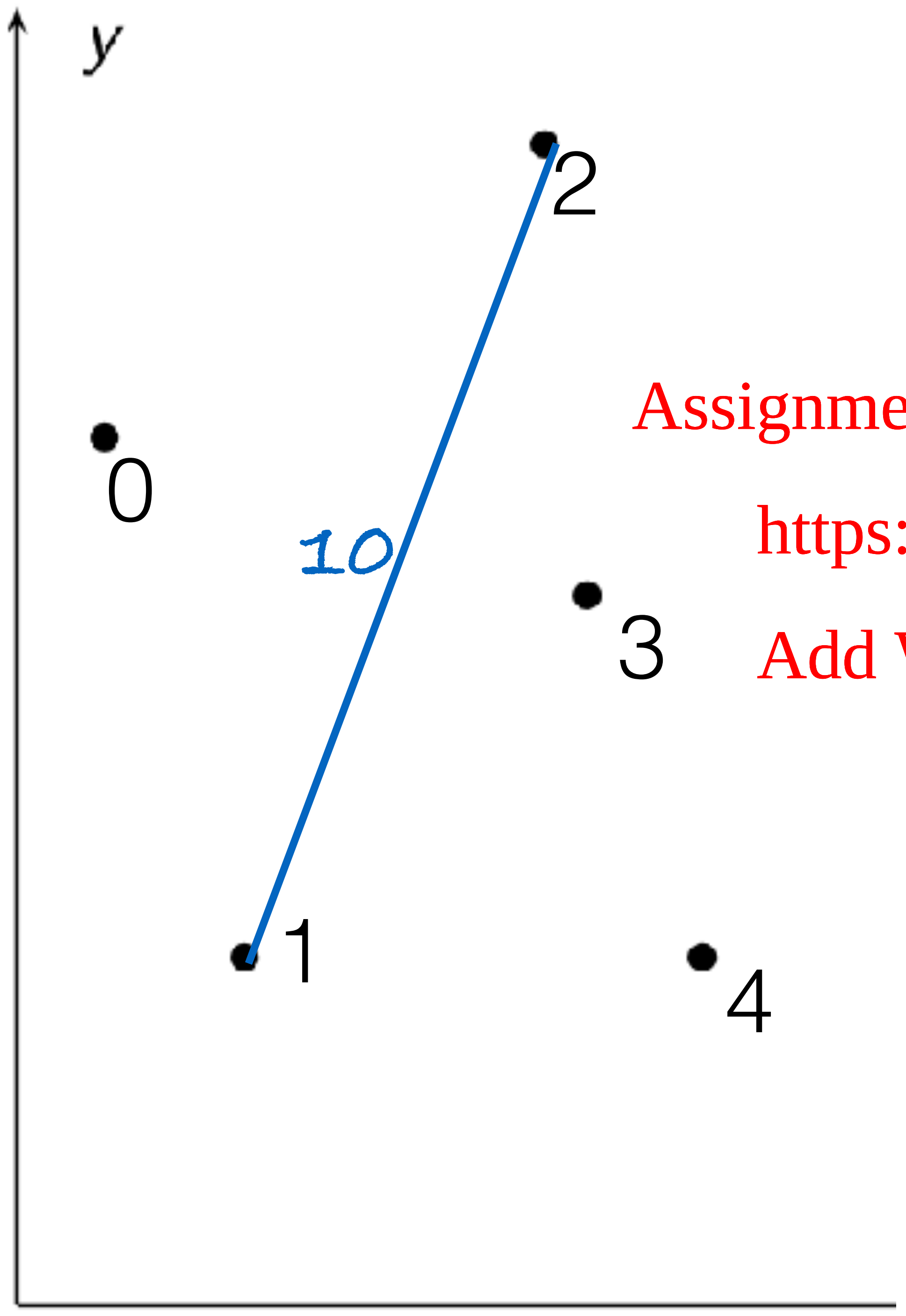
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Closest Pair Problem (2D)



i=1 min=5

Assignment Project Exam Help

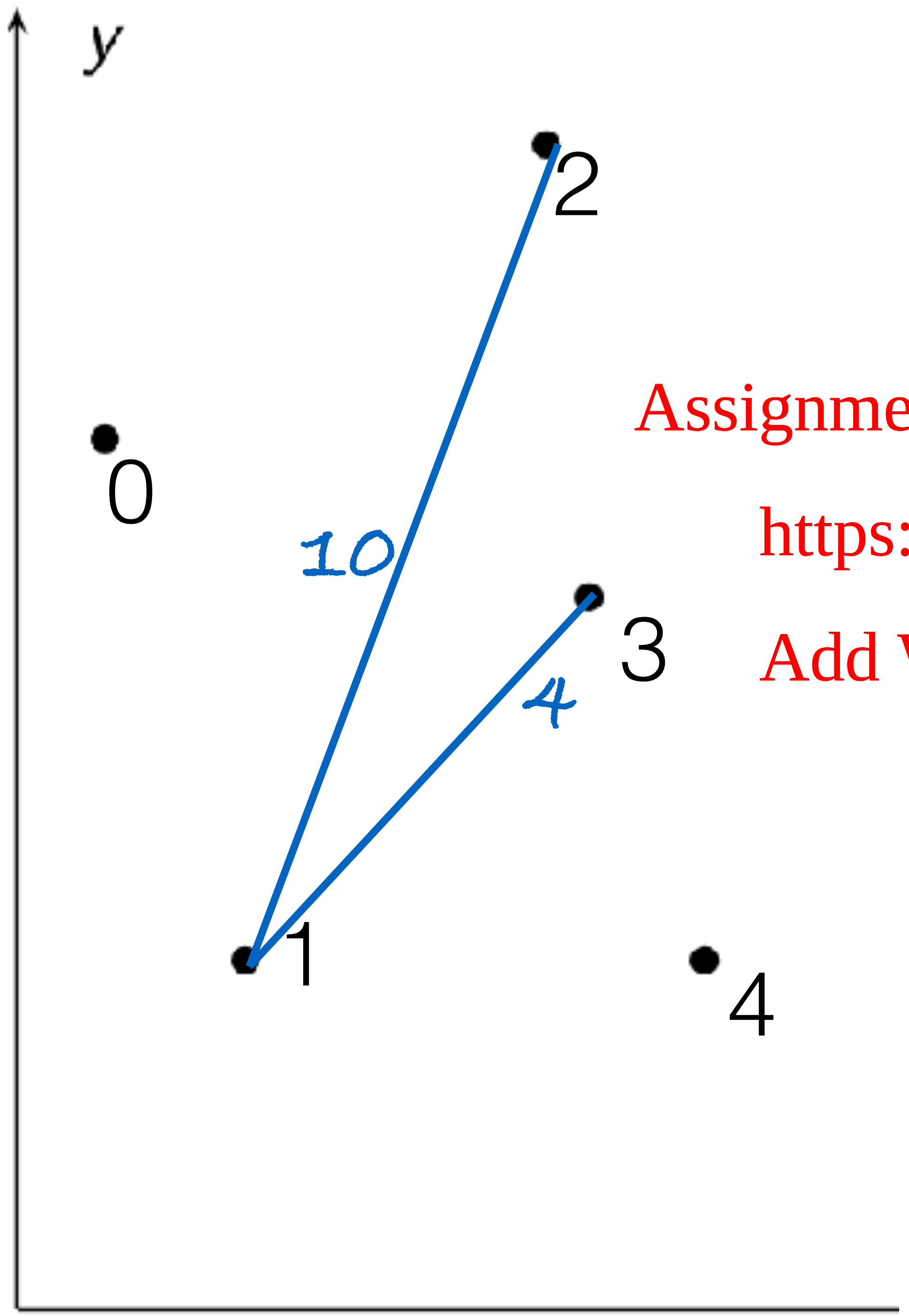
https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Closest Pair Problem (2D)

i=1 min=5



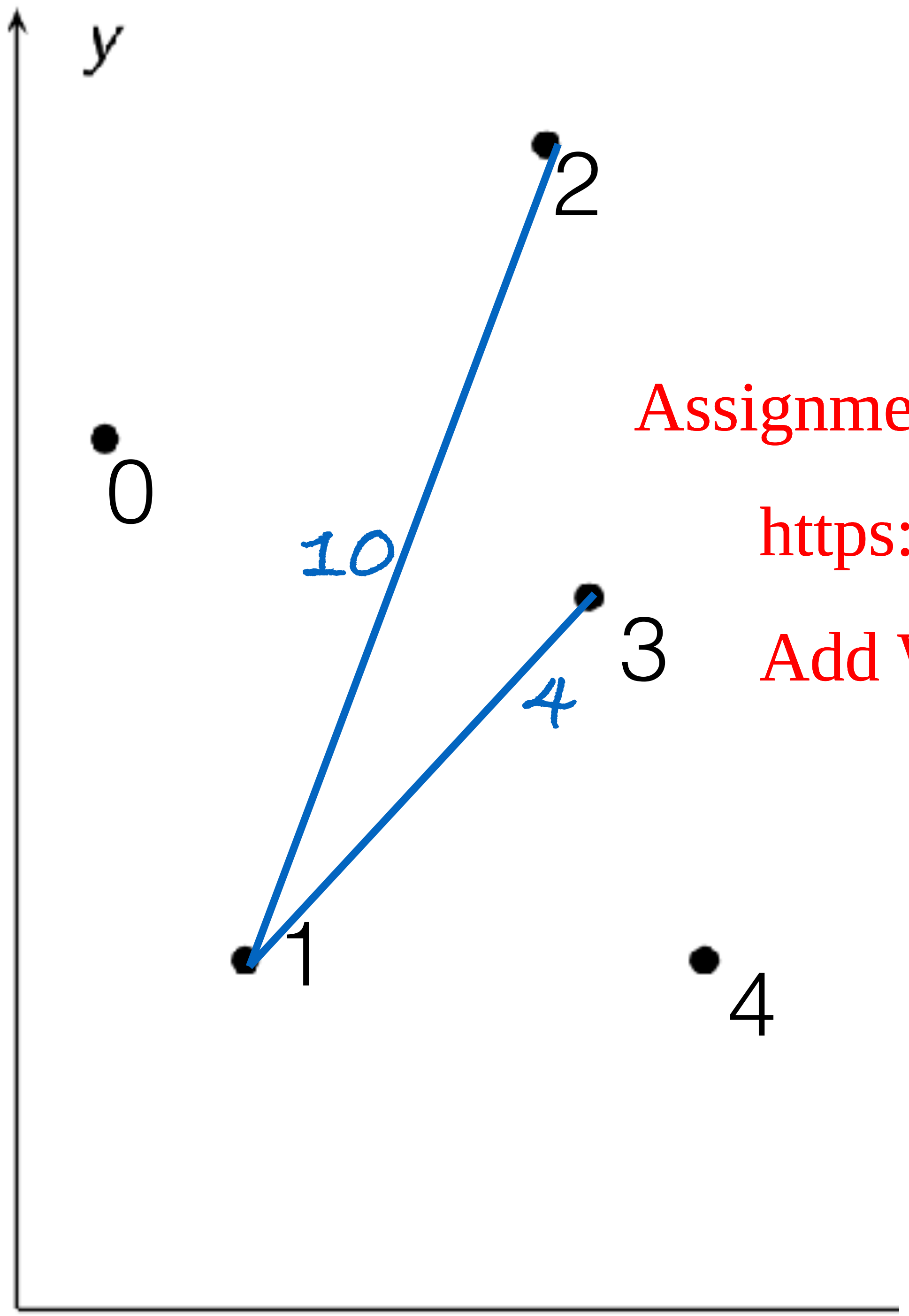
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Closest Pair Problem (2D)



i=1 min=4

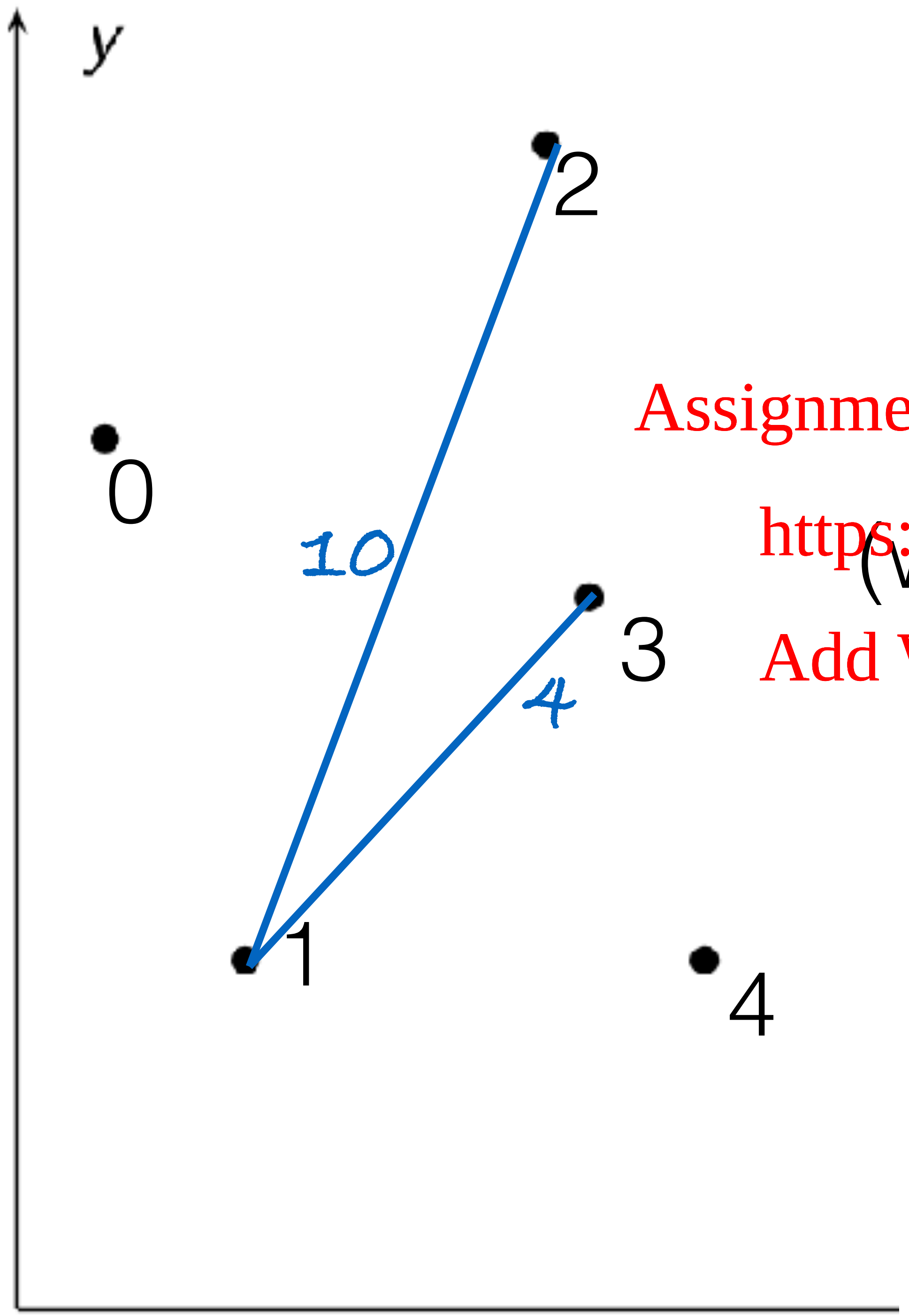
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Closest Pair Problem (2D)



i=1 min=4

and so on until i=3 and j=4 (which will find the minimum here)

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Analysing Brute Force Closest Pair Algorithm

THE UNIVERSITY OF MELBOURNE

- Not hard to see that the algorithm is
- Note, however, that we can speed up the algorithm considerably, by utilising the monotonicity of the square root function
- Does that contradict the claim?
- Later we will see a clever divide and conquer approach leads to a $\Theta(n \log n)$ algorithm

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Analysing Brute Force Closest Pair Algorithm

- Not hard to see that the algorithm is $\Theta(n^2)$

- Note, however, that we can speed up the algorithm considerably, by utilising the monotonicity of the square root function

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

- Does that contradict the claim?

- Later we will see a clever divide and conquer approach leads to a $\Theta(n \log n)$ algorithm

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Analysing Brute Force Closest Pair Algorithm

THE UNIVERSITY OF MELBOURNE

- Not hard to see that the algorithm is $\Theta(n^2)$

- Note, however, that we can speed up the algorithm considerably, by utilising the monotonicity of the square root function $a < b \implies \sqrt{a} < \sqrt{b}$

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- Does that contradict the claim?

- Later we will see a clever divide and conquer approach leads to a $\Theta(n \log n)$ algorithm

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Analysing Brute Force Closest Pair Algorithm

- Not hard to see that the algorithm is $\Theta(n^2)$
- Note, however, that we can speed up the algorithm considerably, by utilising the monotonicity of the square root function

$$a < b \implies \sqrt{a} < \sqrt{b}$$

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- Does that contradict the $\Theta(n^2)$ claim?
- Later we will see a clever divide and conquer approach leads to a $\Theta(n \log n)$ algorithm

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Brute Force Summary

- Simple, easy to program, widely applicable.
- Standard approach for small tasks.
- Reasonable algorithms for some problems.
- **But:** Generally inefficient—does not scale well.
- Use brute force for prototyping, or when it is known that input remains small.

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Exhaustive Search

- Problem type:
  - Combinatorial decision or optimization problems
  - Search for an element with a particular property
  - Domain grows exponentially, for example all permutations

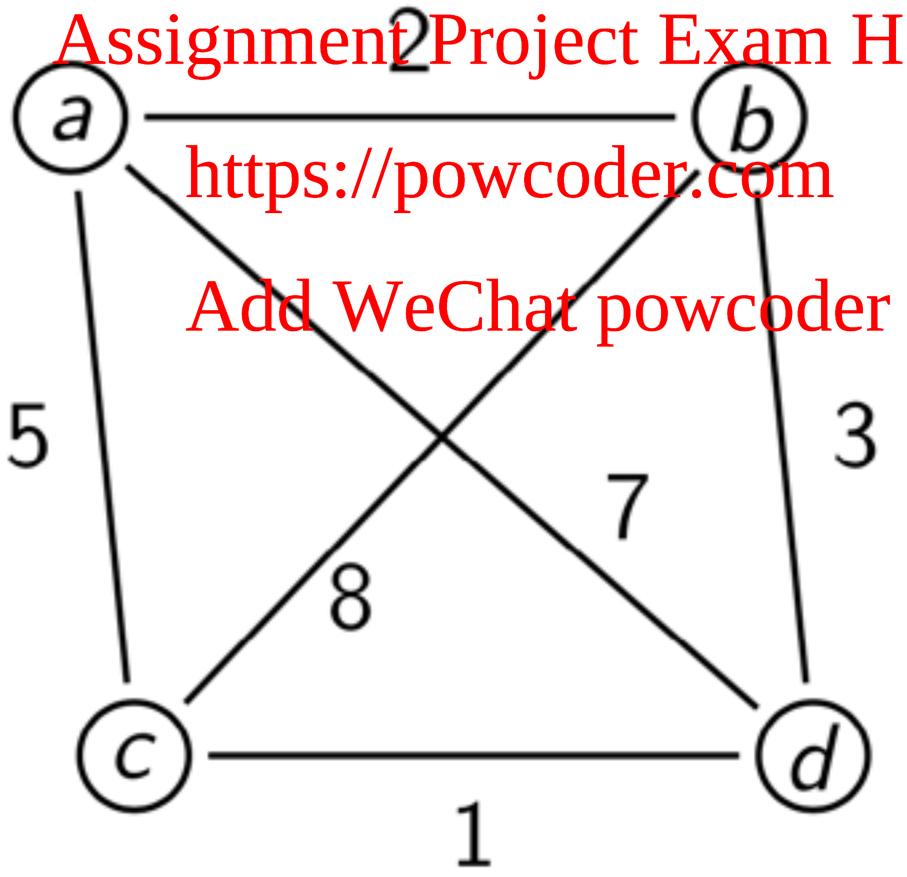
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- The brute-force approach—generate and test:
  - Systematically construct all possible solutions
  - Evaluate each, keeping track of the best so far
  - When all potential solutions have been examined, return the best found

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Example 1: Travelling Salesperson (TSP)

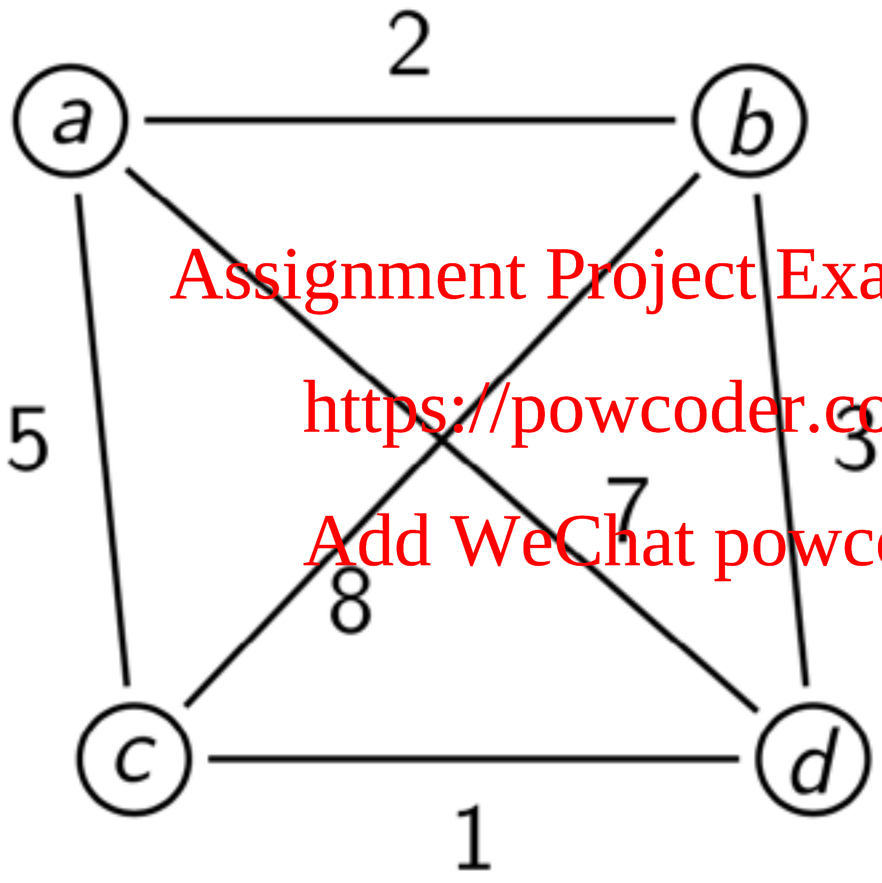
- Find the shortest **tour** (visiting each node exactly once before returning to the start) in a *weighted, undirected* graph.



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Side Note: Graph Concepts



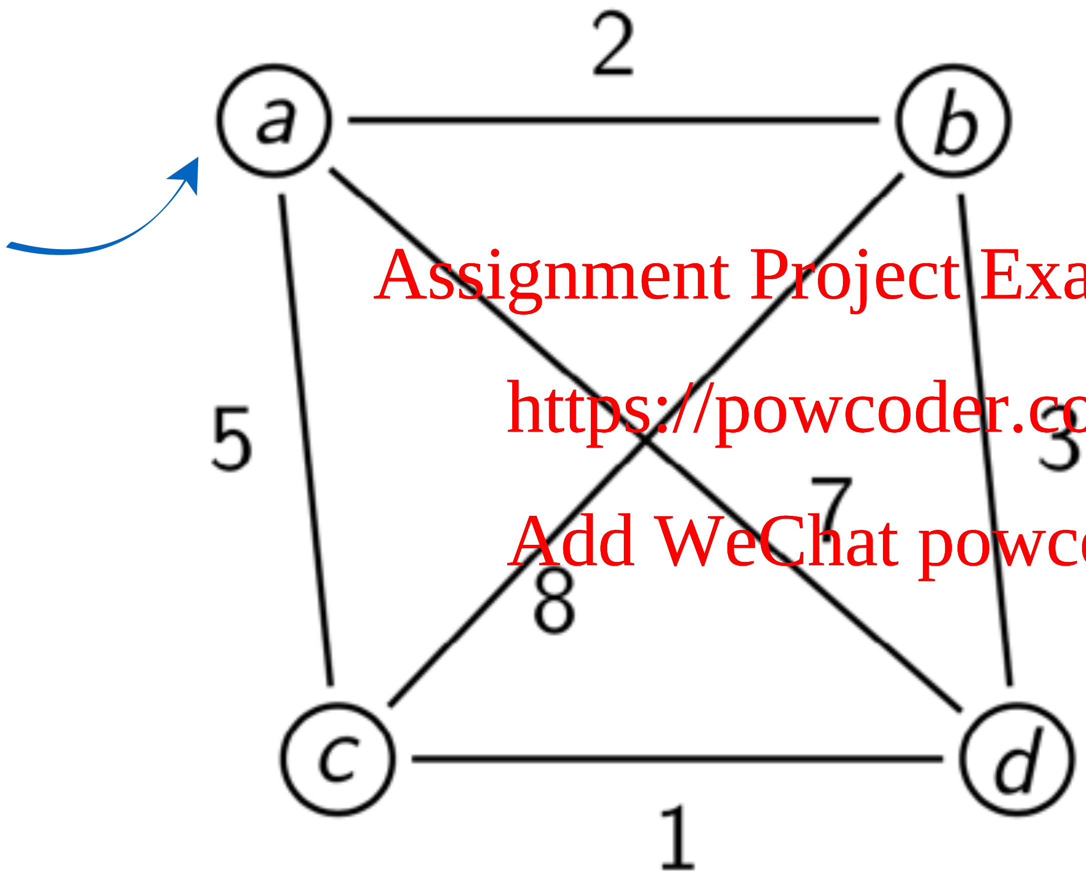
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Side Note: Graph Concepts



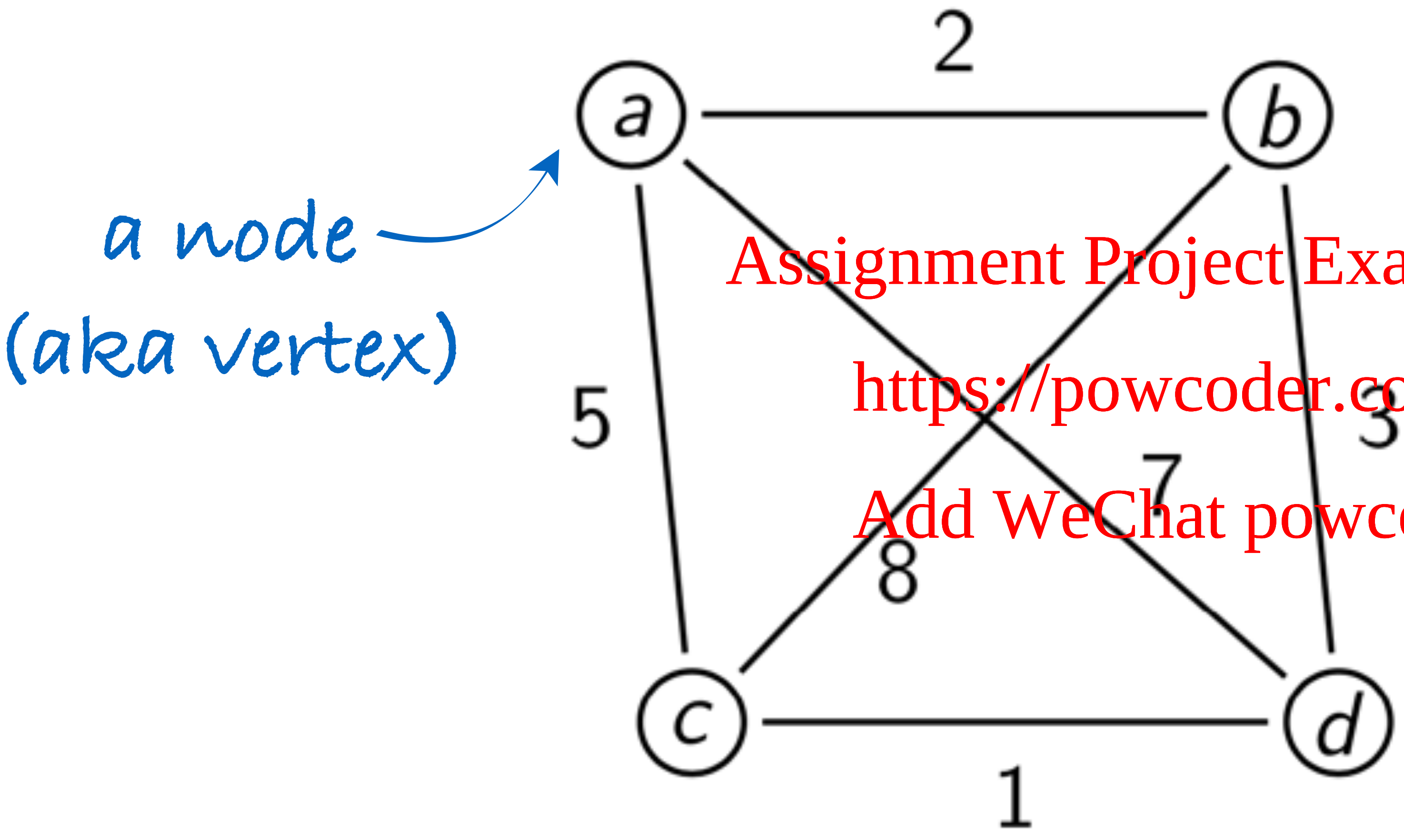
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

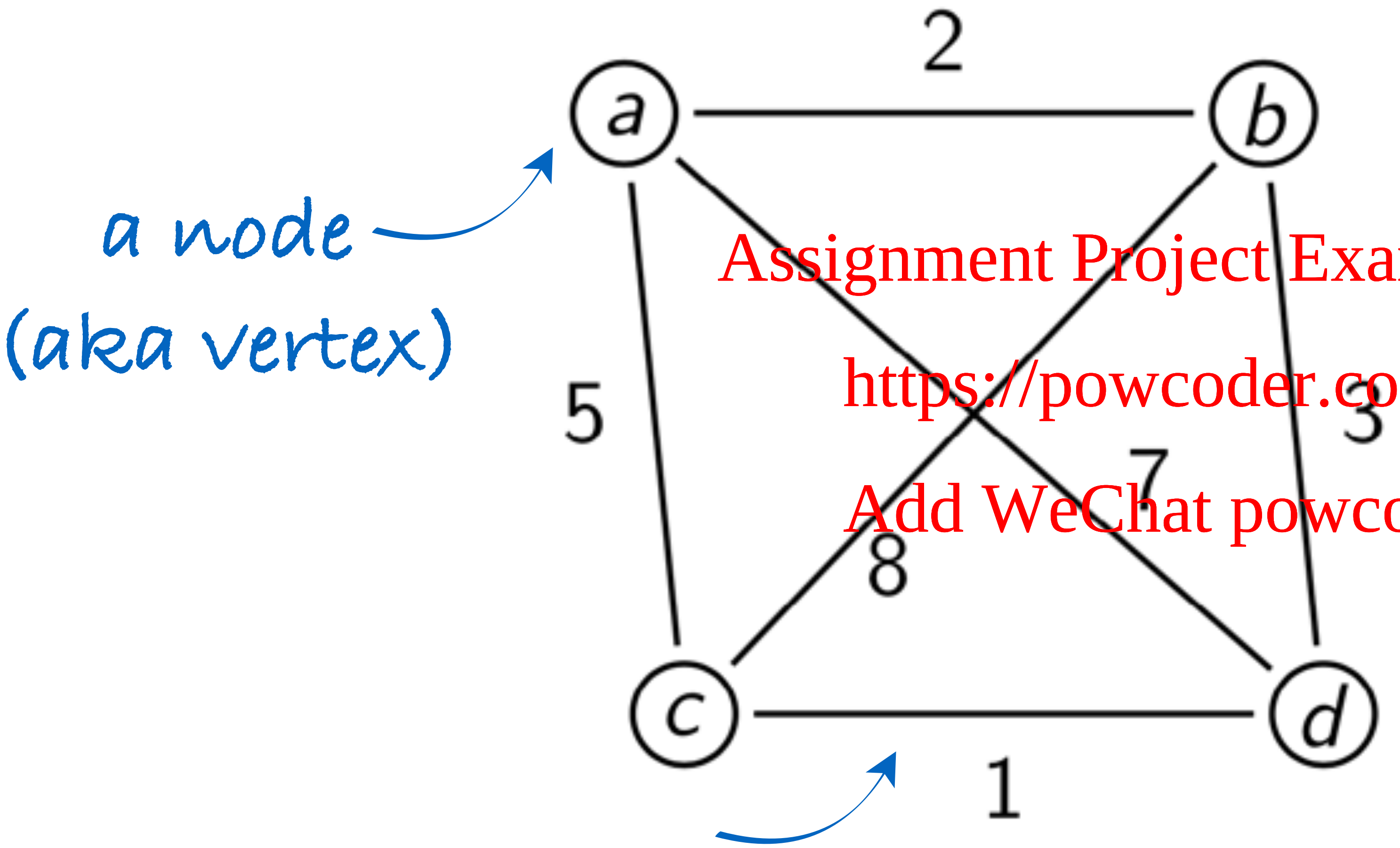
Side Note: Graph Concepts



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Side Note: Graph Concepts



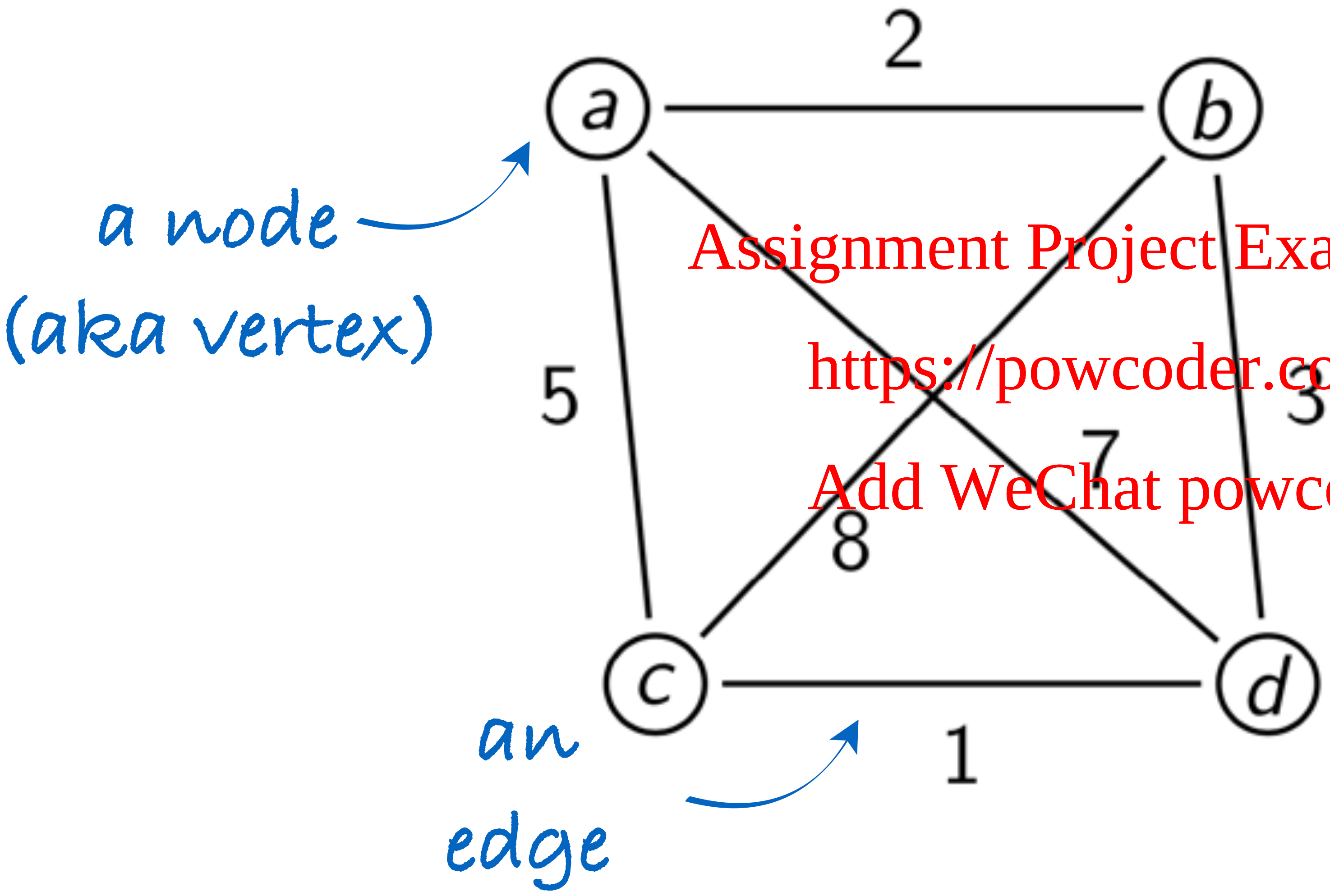
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Side Note: Graph Concepts



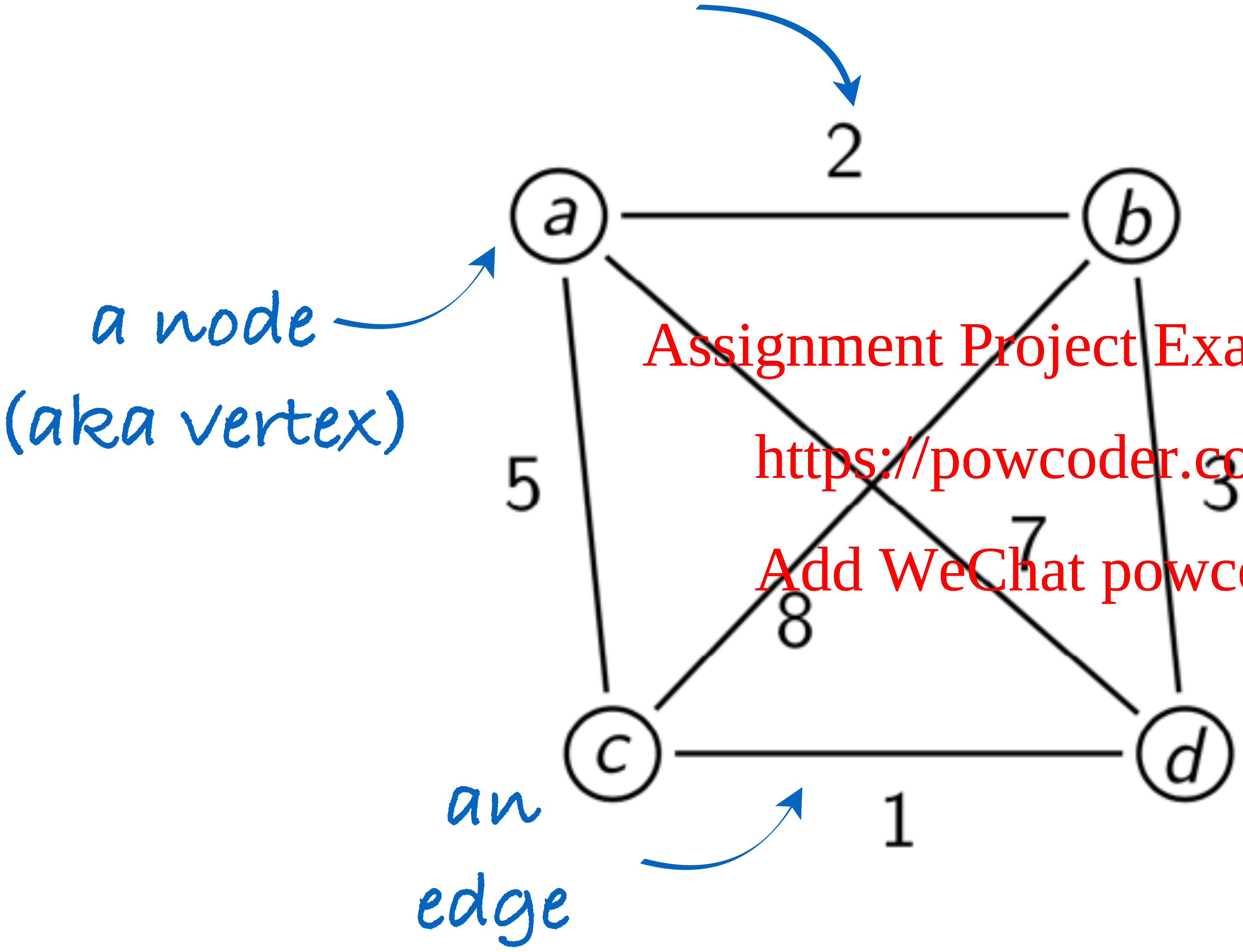
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Side Note: Graph Concepts



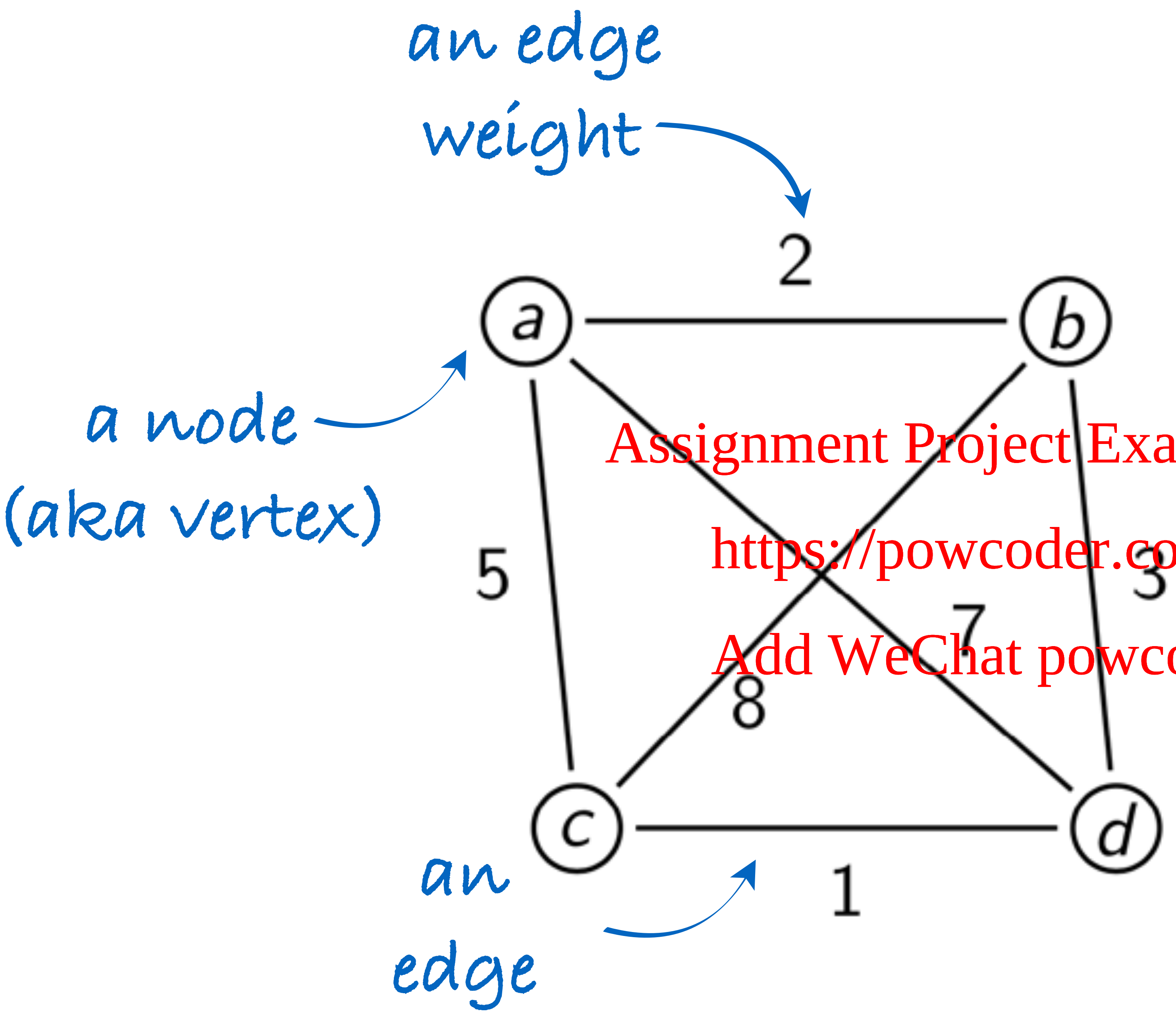
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Side Note: Graph Concepts



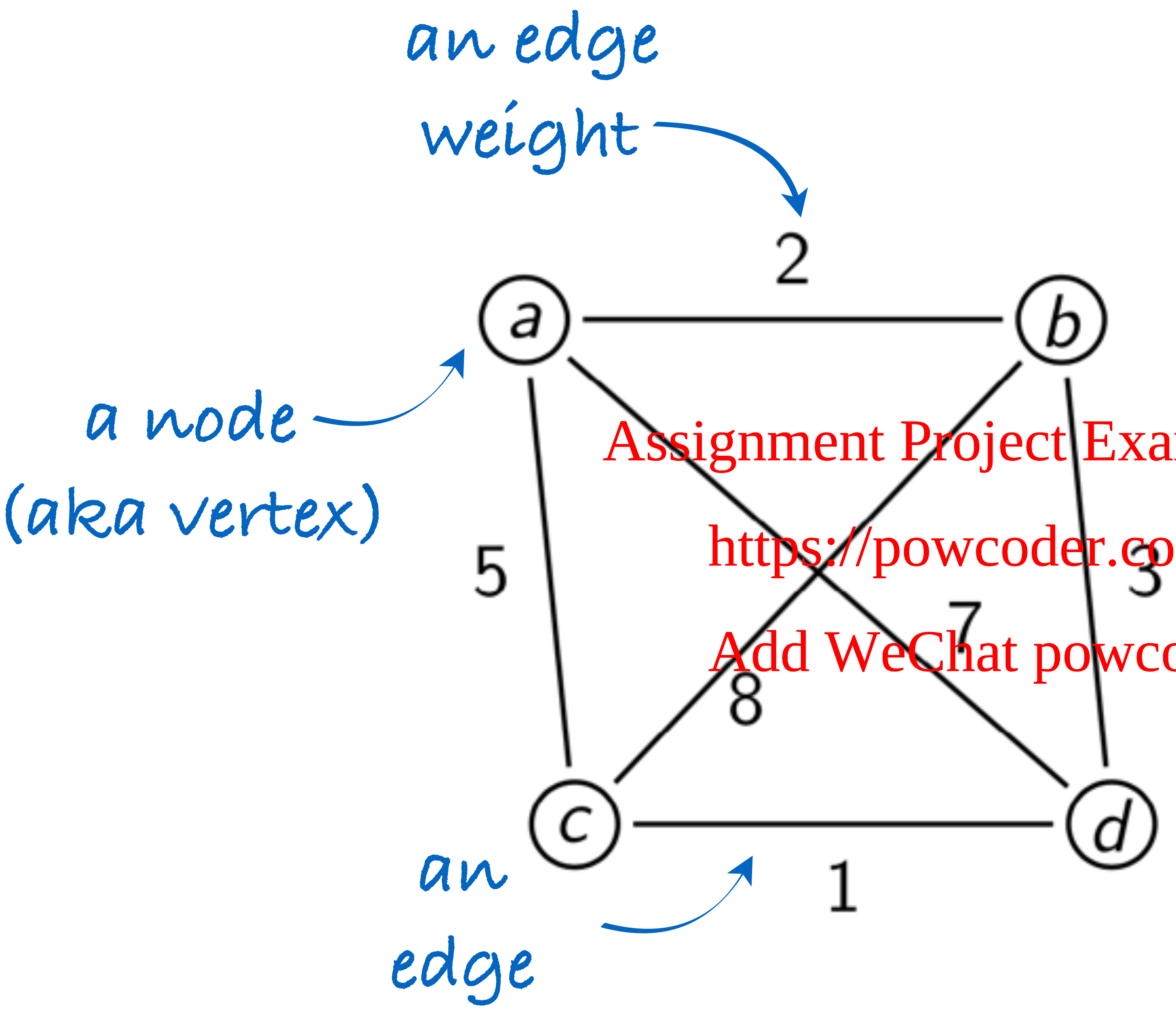
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Side Note: Graph Concepts



an edge weight

a node (aka vertex)

an edge

Assignment Project Exam Help

https://powcoder.com

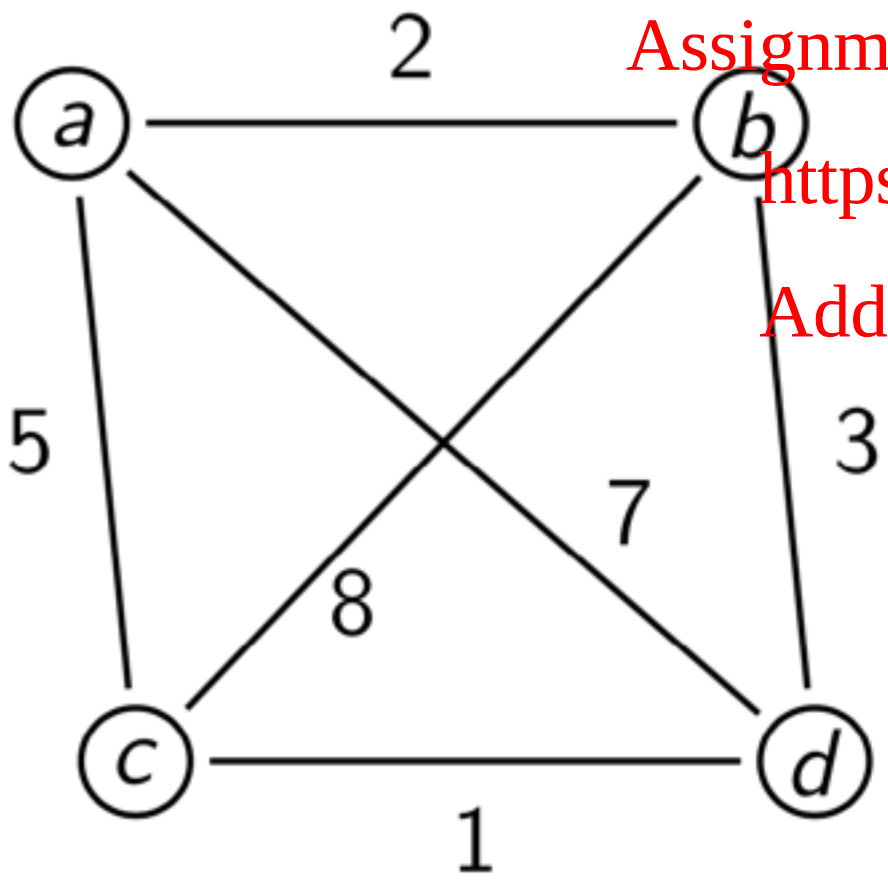
Add WeChat powcoder

This graph is undirected since edges do not have directions associated with them.

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Example 1: Travelling Salesperson (TSP)

- Find the shortest **tour** (visiting each node exactly once before returning to the start) in a *weighted, undirected* graph.



Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Tours

$a - b - c - d - a$	:	18
$a - b - d - c - a$	:	11
$a - c - b - d - a$	:	23
$a - c - d - b - a$	:	11
$a - d - b - c - a$	:	23
$a - d - c - b - a$	:	18

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Example 2: Knapsack

- Given n items with
  - Weights: w_1, w_2, …, w_n
  - Values: v_1, v_2, …, v_n
  - Knapsack of capacity W
- find the most valuable selection of items whose combined weight does not exceed W

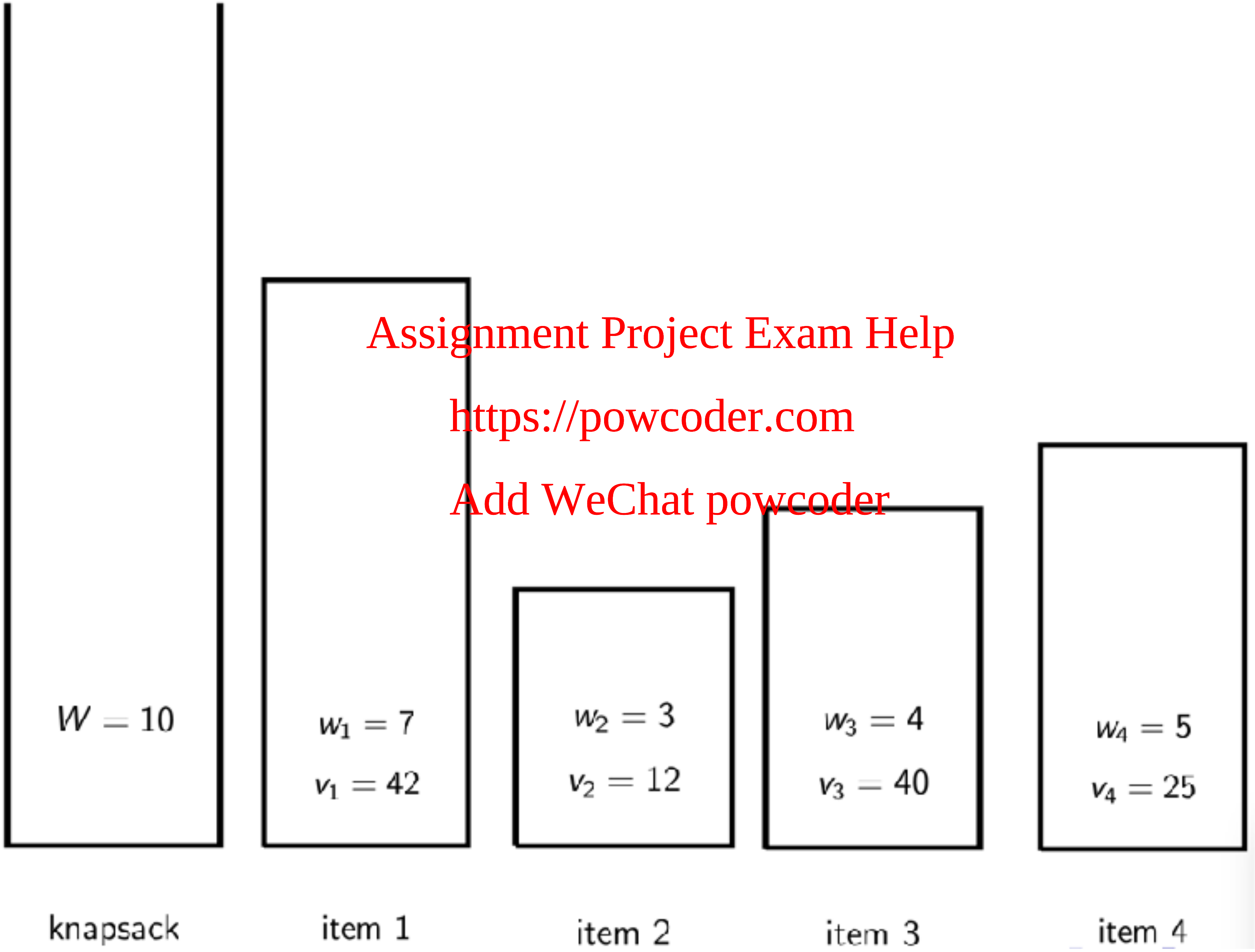
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creativ e

Knapsack Example



Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Knapsack: Example

Set	Weight	Value
$\emptyset$	0	0
$\{1\}$	7	42
$\{2\}$	3	12
$\{3\}$	4	40
$\{4\}$	5	25
$\{1, 2\}$	10	54
$\{1, 3\}$	11	NF
$\{1, 4\}$	12	NF

Set	Weight	Value
$\{2, 3\}$	7	52
$\{2, 4\}$	8	37
$\{3, 4\}$	9	65
$\{1, 2, 3\}$	14	NF
$\{1, 2, 4\}$	15	NF
$\{1, 3, 4\}$	16	NF
$\{2, 3, 4\}$	12	NF
$\{1, 2, 3, 4\}$	19	NF

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

NF means “not feasible”: exhausts the capacity of the knapsack.

Later we’ll find a better algorithm based on **dynamic programming**

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Comments on Exhaustive Search

- Exhaustive search algorithms have acceptable running times **only for very small instances.**

- In many cases there are better alternatives, for example, Eulerian tours, shortest paths, minimum spanning trees, . . .

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- But for some problems, it is **known** that there is essentially no better alternative.

- For a large class of important problems, it **appears** that there is no better alternative, but we have no proof either way.

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Hamiltonian Tours

- The Hamiltonian tour problem is this:

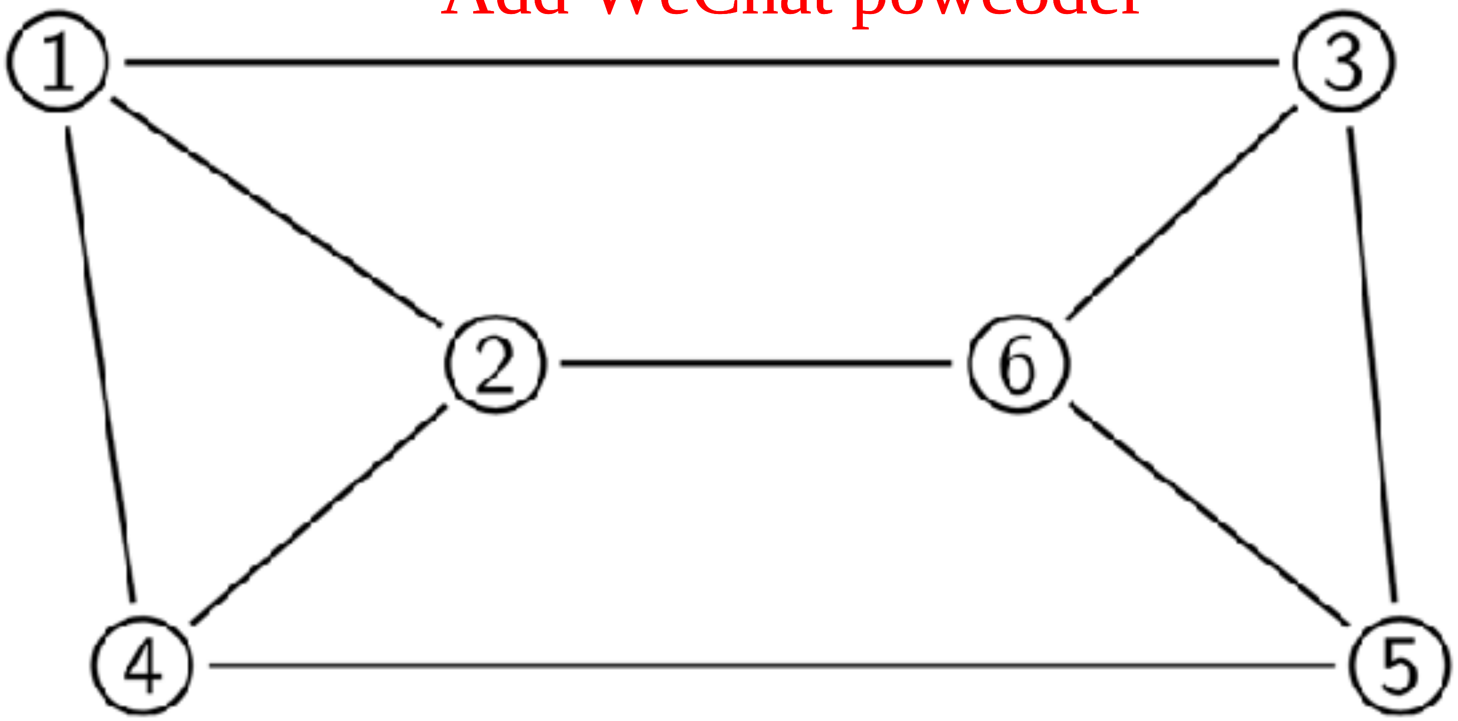
- In a given undirected graph, is there a simple tour (a path that visits each **node** exactly once, except it returns to the starting node)?

Assignment Project Exam Help

- Is there a Hamiltonian tour of this graph?

https://powcoder.com

Add WeChat powcoder



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

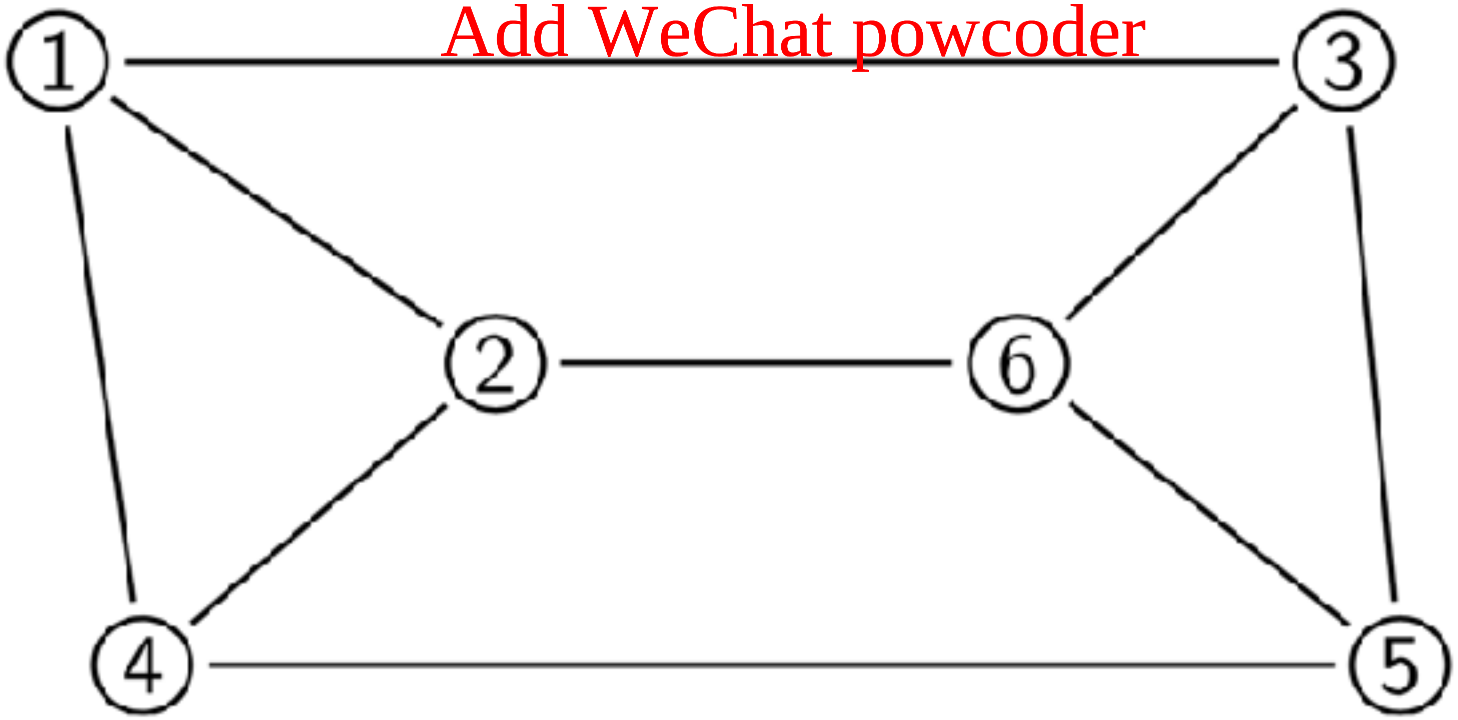
Eulerian Tours

- The Eulerian tour problem is this:
- In a given undirected graph, is there a path which visits each **edge** exactly once?
- Is there a Eulerian tour of this graph?

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Hard and Easy Problems

- Recall that by a **problem** we usually mean a parametric problem: an infinite family of problem "instances".
- Sometimes our intuition about the difficulty of problems is not very reliable. The Hamiltonian Tour problem and the Eulerian Tour problem look very similar, but one is hard and the other is easy. We shall see more examples of this phenomenon later.
- For many important **optimization** problems we do not know of solutions that are essentially better than exhaustive search (a whole raft of **NP-complete** problems, including TSP, knapsack).
- In those cases we may look for **approximation algorithms** that are fast and still find solutions that are reasonably close to the optimal.
- We plan to return to this idea in Week 12.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Next Up

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

- **Recursion** as a problem solving technique

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License