Meet In The Middle - II

Special class

Vivek Chauhan • Jan 17, 2021

Vivek Chauhan

< Educator Image >

- < Educator Credentails >
-

unacademy | CODECHEF

2021 : The Year To QUIT PROCRASTINATING And LEARN CODING

Join Our Exclusive BATCHES

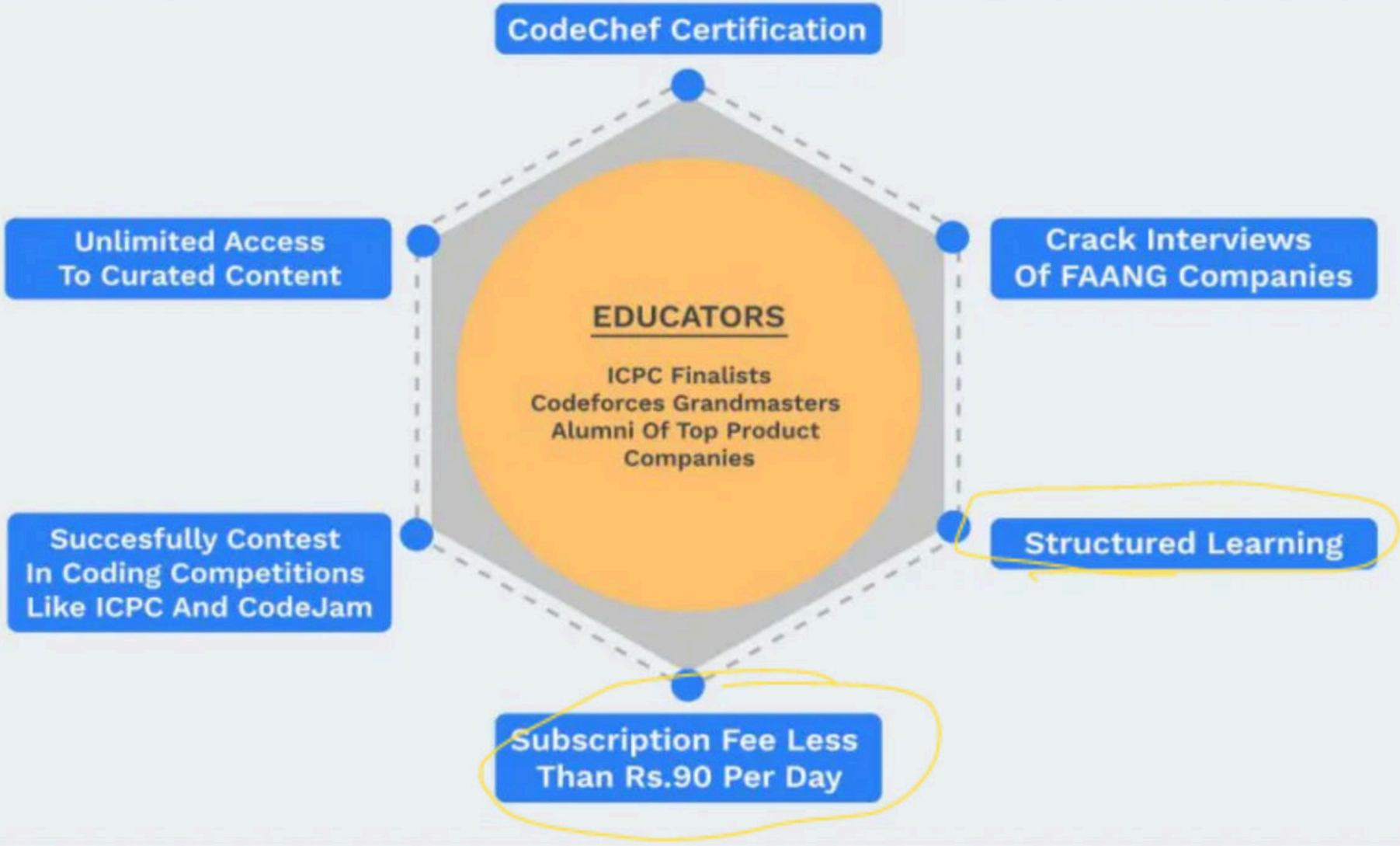
PINNACLE: COMPREHENSIVE AND CONCISE TRACK TO BECOME AN EXPERT **GOING LIVE ON 18TH JAN 2021**

- Conquest 2021: Year Long Journey for Intermediate Coders to Become Experts (C++) - Live on 8th Jan 2021
- C++: Conquest 2021: From Programming Fundamentals to Career Readiness - Live on 8th Jan 2021
- Python: Conquest 2021: From Programming Fundamentals to Career Readiness - Live on 8th Jan 2021
- Java: Conquest 2021: From Programming Fundamentals to Career Readiness - Live on 8th Jan 2021

Resolve to become an expert level programmer in 2021 and subscribe at an expense even lesser than INR 90/ day

Exclusive Batch Starting On 18th Jan 2021

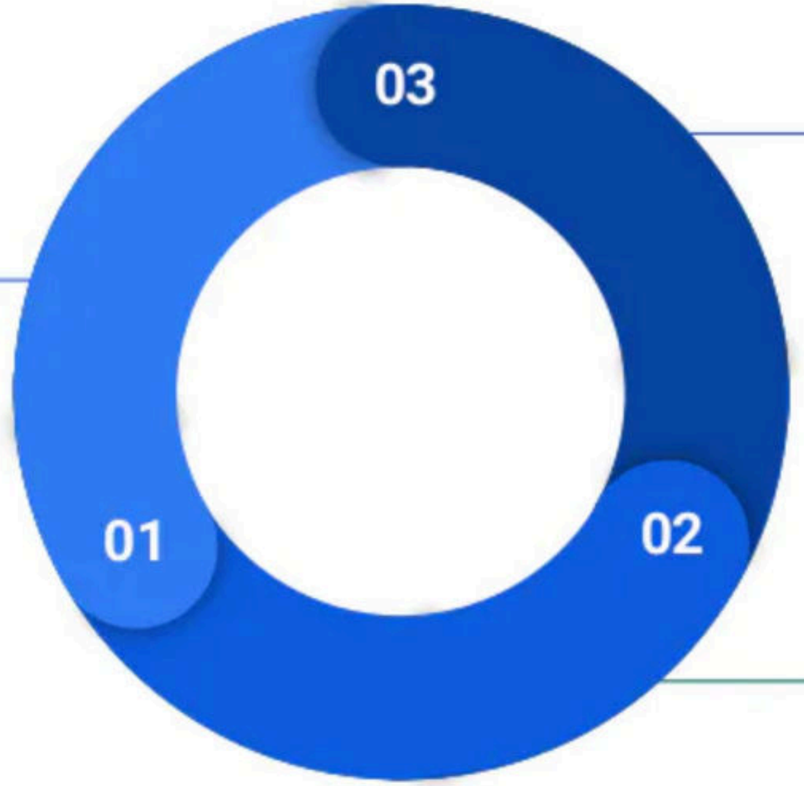
PINNACLE - Batch For Intermediates And Beginners



What you will get

Live Interactive Classes

Attend live interactive classes with our top educators. Interact during class with educators to get all your doubts resolved



Doubt Support

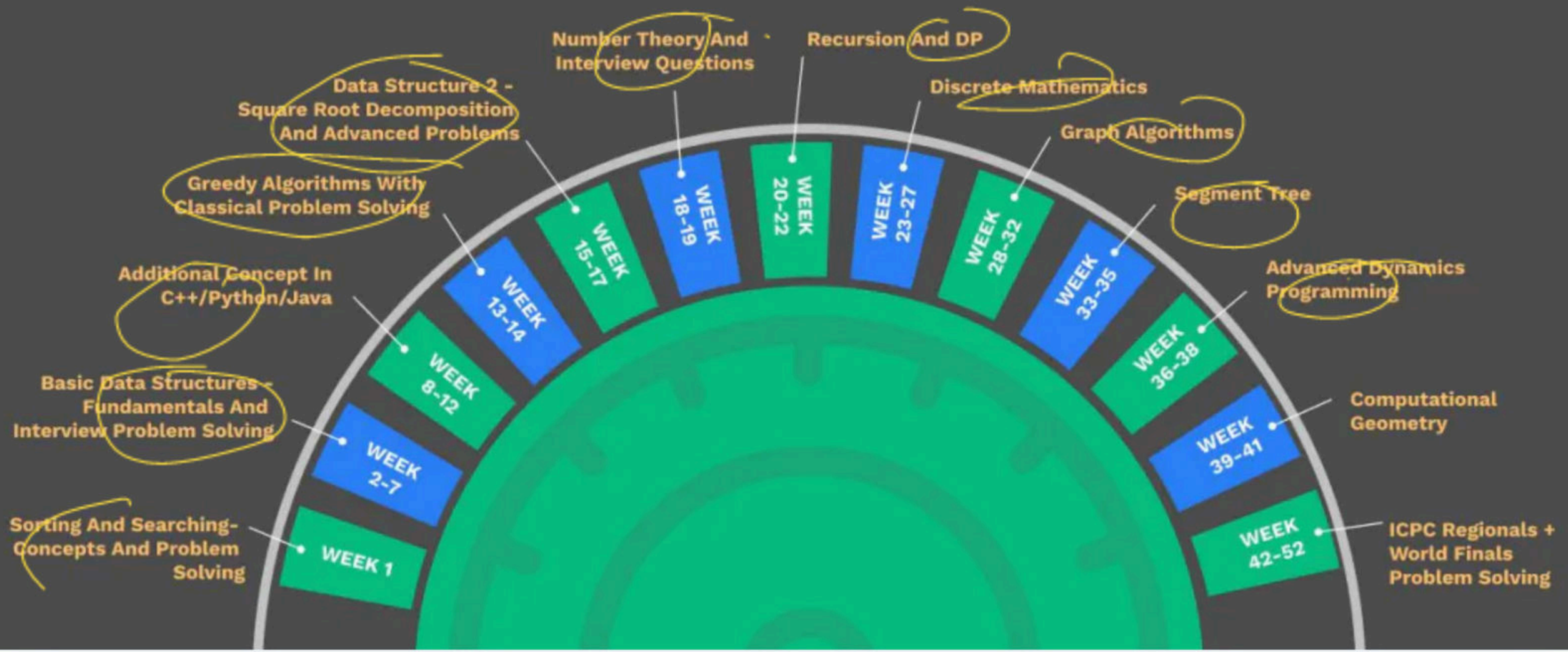
If you get stuck in any problem post class- Get your doubts resolved by our expert panel of teaching assistants and community members instantly

Practice Relevant Problems @ CodeChef

Each class comes with a set of curated practice problems to help you apply the concepts in real time.

Topic-wise Batch Structure

PINNACLE



One Subscription and Unlimited Access to All Batches/ Courses

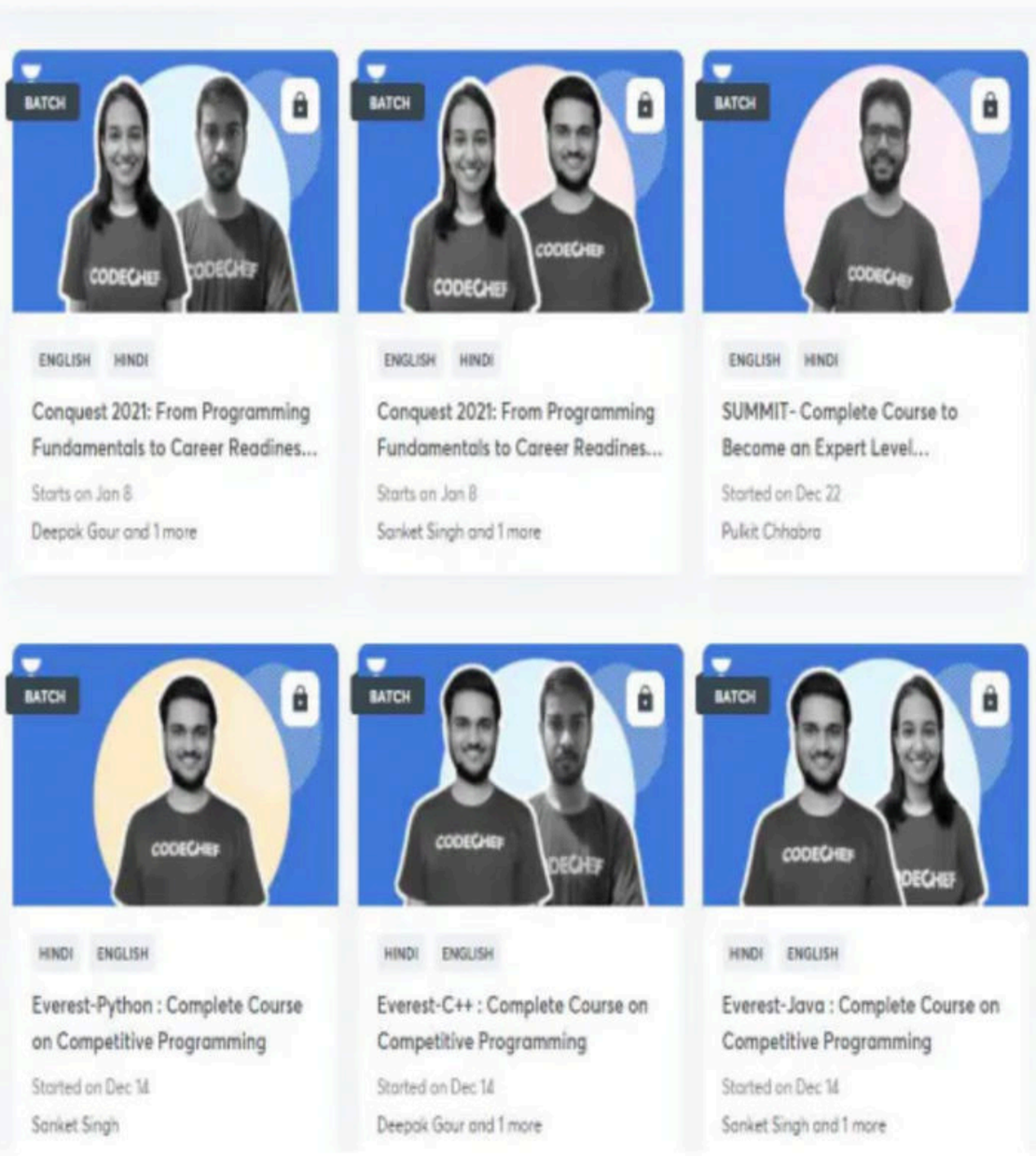
Batch Getting Live on 18th January 2021:

PINNACLE: Comprehensive and Concise Track to Become an Expert (C++)

Recently Live Batches

- Conquest 2021: From Programming Fundamentals to Career Readiness **(C++)**
- Conquest 2021: From Programming Fundamentals to Career Readiness **(Java)**
- Conquest 2021: From Programming Fundamentals to Career Readiness **(Python)**
- Conquest 2021: Year Long Journey for Intermediate Coders to Become Experts (C++)

And many more for all levels of programmers
Visit the Batches section in Unacademy



Educators

Tanuj Khattar

ACM ICPC World Finalist - 2017, 2018. Indian IOI Team Trainer 2016-2018. Worked @ Google, Facebook, HFT. Quantum Computing Enthusiast.

Sanket Singh

Software Development Engineer @ LinkedIn | Former SDE @ Interviewbit | Google Summer of Code 2019 @ Harvard University | Former Intern @ISRO

Pulkit Chhabra

Codeforces: 2246 | Codechef: 2416 | Former SDE Intern @CodeNation | Former Intern @HackerRank

Riya Bansal

Software Engineer at Flipkart | Former SDE and Instructor @ InterviewBit | Google Women TechMakers Scholar 2018

Triveni Mahatha

Qualified ICPC 2016 World Final. Won multiple Codechef Long Challenges (India). ICPC Onsite Regionals' Problem setter and Judge. IIT Kanpur.

Deepak Gour

ICPC World Finalist 2020 | Former Instructor @InterviewBit | Software Engineer at AppDynamics

Educators

Himanshu Singh

World Finalist ICPC 2020, Winner Techgig Code Gladiators 2020, Winner TCC '19, 2020 CSE Graduate from IIT BHU, Works at Nutanix

Nishchay Manwani

Hey I am Nishchay Manwani from CSE, IIT Guwahati and I'm a Seven star on Codechef and International Grandmaster on Codeforces.

Murugappan S

Software engineer at Google. Have won many programming contests. Max Rating of 2192 in codeforces and 2201 in codechef.

Vivek Chauhan

Codechef: 7 stars (2612) India Rank 6, Codeforces: MASTER (2279), Won Codechef Long Challenges(India), TCO20 Southern Asia Runner up

and many more joining soon...

Teaching Assistants support on chat and Doubts Forum

CODECHEF Discuss

You may face issue with markdown in posts. In such cases, report it here along with the post link.

unacademy Live Classes / CodeChef Practice & Doubts / CodeChef Doubt Forum

Clear your Doubts with our Expert Panel of Teaching Assistants & Community Members

Leave no room for doubts. Create a topic.

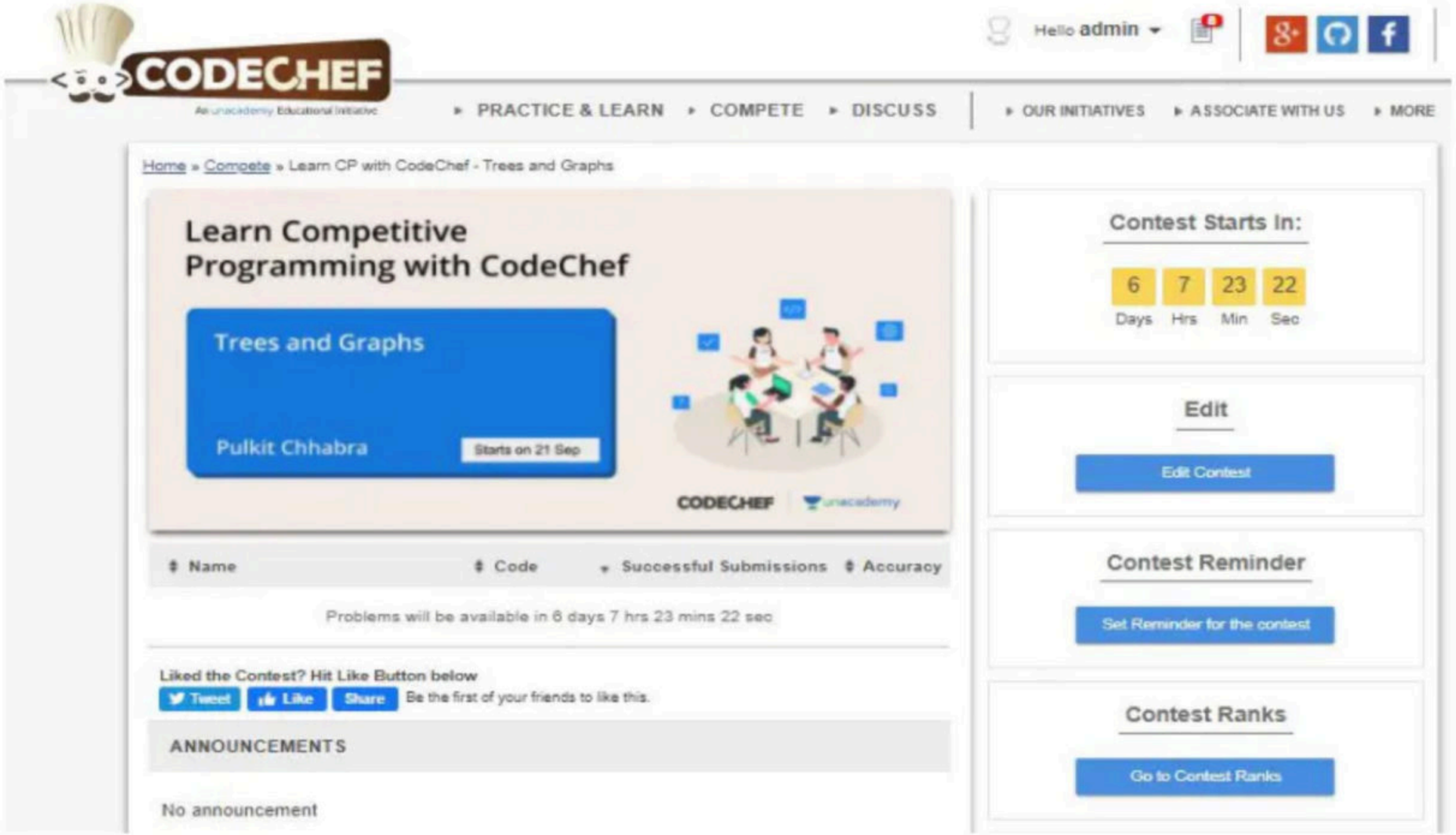
CODECHEF unacademy

Learn CP on Unacademy Plus | all tags | Latest | Top | Bookmarks | Edit | + New Topic

Topic	Replies	Views	Activity
About the Learn CP on Unacademy Plus category	1	6	2d

There are no more Learn CP on Unacademy Plus topics. Why not create a topic?

Course-wise Practice Problems



Flexible Subscription Plans

Competitive Programming subscription

Choose a plan and proceed

No cost EMI available on 6 months & above subscription plans

1 month		₹5,400 per month	₹5,400 Total (Incl. of all taxes)
3 months	11% OFF	₹4,800 per month	₹14,400 Total (Incl. of all taxes)
6 months	25% OFF	₹4,050 per month	₹24,300 Total (Incl. of all taxes)
12 months	54% OFF	₹2,475 per month	₹29,700 Total (Incl. of all taxes)

vivek_1998299

Awesome! You got 10% off

Proceed to pay

vivek_1998299

Proceed to pay

Meet in the Middle

Vivek Chauhan
(vivek_1998299)

Prerequisites

1. Basics of graph theory
2. Basics of bit manipulation

1) Easy - med

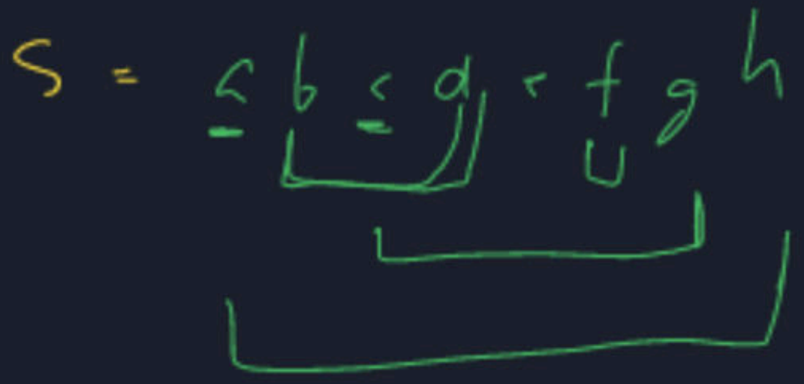
2) Hard

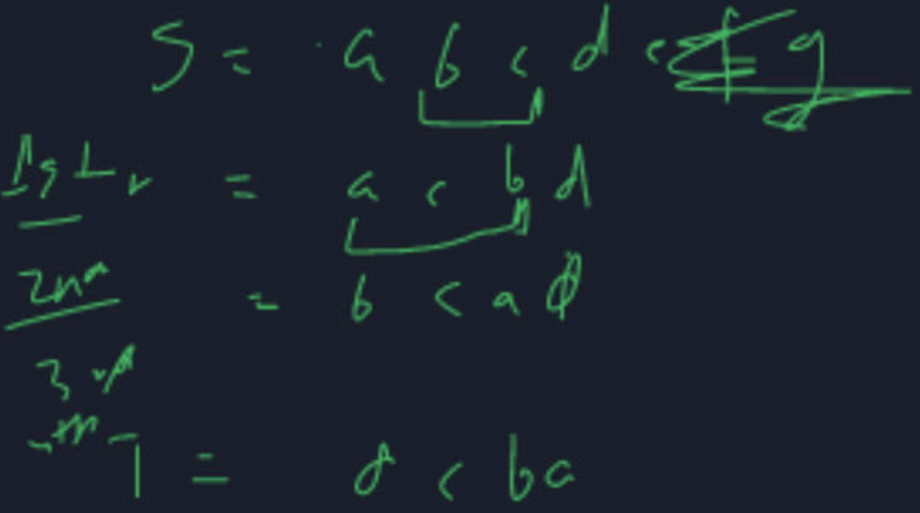
You are given two strings **S** and **T** of length **N**.
Figure out if we can get string **T** starting from string **S** and applying 4 substring reversal operations.

Constraints:

1 <= **N** <= ~~300~~

S and **T** consists of lowercase English alphabets only





N = 4
S = aaba
T = aaab

Yes :

S = a a b a

1st :

a a b a

2nd :

b a a a

3rd :

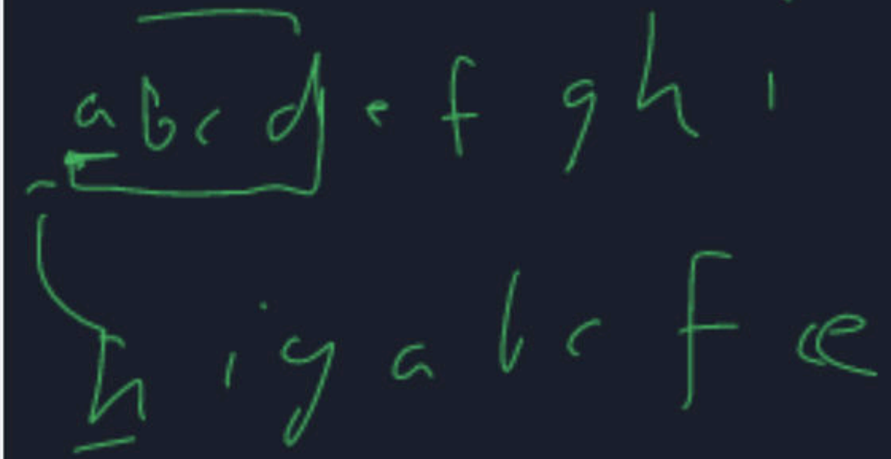
a a a b

4th

a a a b

T = a a a b

N = 4
S = abcd
T = cdba



N = 4

A = abcd

B = dcab

What will be the answer?

N = 4
S = abcd
T = cdba

```
for(L1=0; L1<N; L1++) { O(N)
  for(R1=L1; R1<N; R1++) { O(N)
    for(L2=0; L2<N; L2++) { O(N)
      for(R2=L2; R2<N; R2++) {
        for(L3=0; L3<N; L3++) {
          for(R3=L3; R3<N; R3++) {
            for(L4=0; L4<N; L4++) {
              for(R4=L4; R4<N; R4++) {
```

L1 R1

L2 R2

L3 R3 $O(n^8)$

L4 R4

a l c r

2

L4 R1

N = 4
S = abcd
T = cdba

```
string *temp = S;                    O(N)
reverse(temp, L1, R1);               O(N)
reverse(temp, L2, R2);               O(N)
reverse(temp, L3, R3);               O(N)
reverse(temp, L4, R4);
if(temp == T)                        O(N)
    return Yes;
```

$O(N!) \times O(N)$

$O(N^7)$

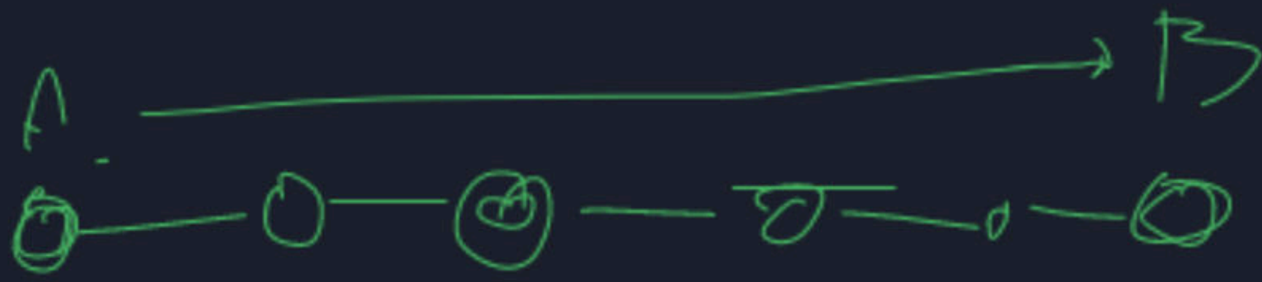
$(20)^9 = 10^{\text{T}_5}$

333

}

return No;

N = 4
S = abcd
T = cdba



S1 S2 . I2 T1

P

[0,2] [1,3]

abcd ⟹ cbad ⟹ cdab

S

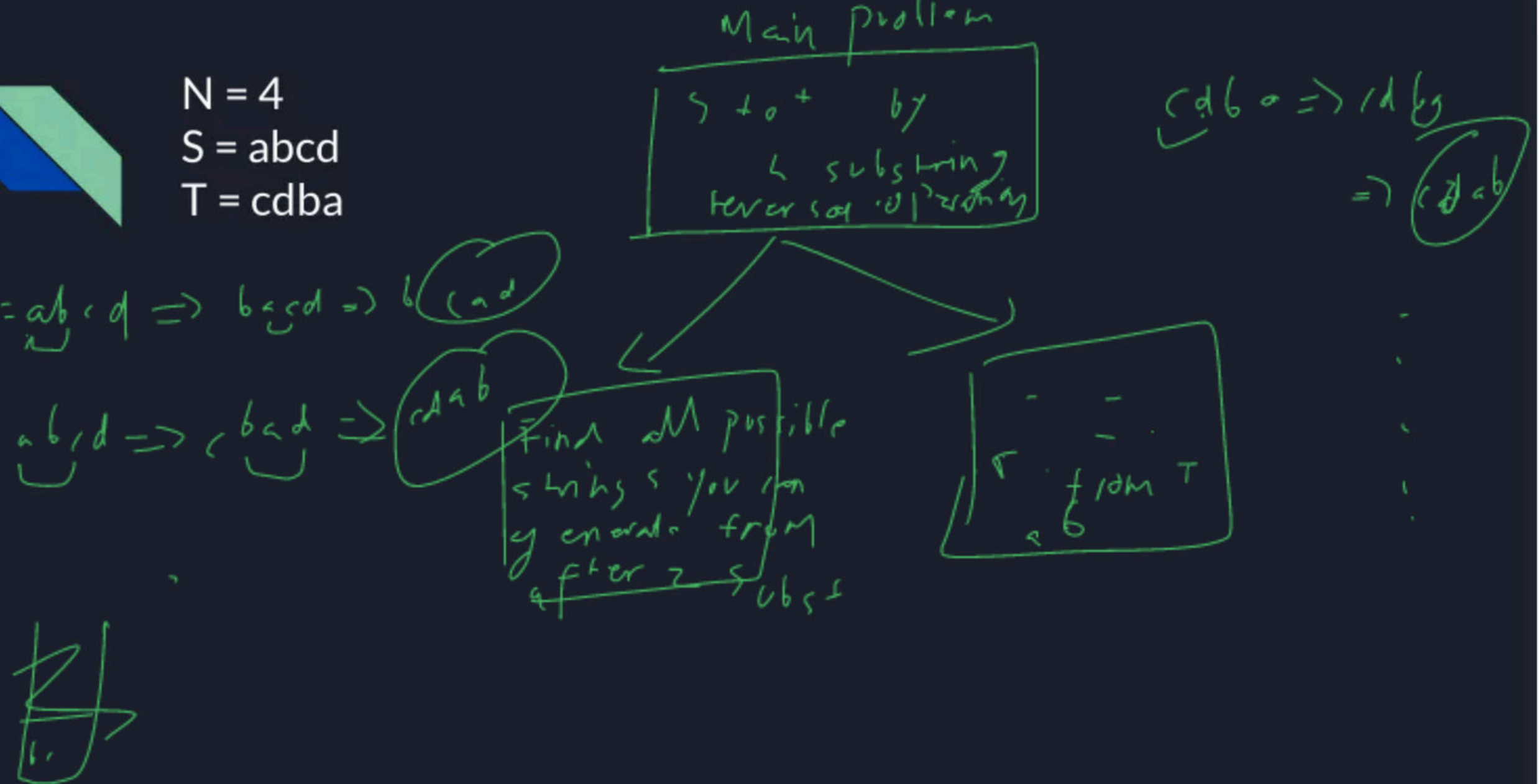
P

cdab ⟸ [0,0] cdab

cdab ⟸ [2,3] cdab

T

N = 4
S = abcd
T = cdba



N = 4
S = abcd
T = cdba

T =>

SoA

N = 4
S = abcd
T = cdba

N = 4
S = abcd
T = cdba

N = 4
S = abcd
T = cdba

N = 4
S = abcd
T = cdba

N = 4
S = abcd
T = cdba

N = 4
S = abcd
T = cdba

N = 4
S = abcd
T = cdba

N = 4
S = abcd
T = cdba

```cpp
bool solve(string S, string T, int N) {
  set<string> mySet;
  for (int L1 = 0; L1 < N; L1++) {
    for (int R1 = L1; R1 < N; R1++) {
      for (int L2 = 0; L2 < N; L2++) {
        for (int R2 = L2; R2 < N; R2++) {
          string tempS = S;
          reverse(tempS.begin() + L1, tempS.begin() + R1 + 1);
          reverse(tempS.begin() + L2, tempS.begin() + R2 + 1);
          mySet.insert(tempS);
        }
      }
    }
  }

  for (int L3 = 0; L3 < N; L3++) {
    for (int R3 = L3; R3 < N; R3++) {
      for (int L4 = 0; L4 < N; L4++) {
        for (int R4 = L4; R4 < N; R4++) {
          string tempT = T;
          reverse(tempT.begin() + L3, tempT.begin() + R3 + 1);
          reverse(tempT.begin() + L4, tempT.begin() + R4 + 1);
          if (mySet.find(tempT) != mySet.end()) {
            return true;
          }
        }
      }
    }
  }
  return false;
}
```

left

right

```cpp
bool solve(string S, string T, int N) {
  set<string> mySet;
  for (int L1 = 0; L1 < N; L1++) {
    for (int R1 = L1; R1 < N; R1++) {
      for (int L2 = 0; L2 < N; L2++) {
        for (int R2 = L2; R2 < N; R2++) {
          string tempS = S;
          reverse(tempS.begin() + L1, tempS.begin() + R1 + 1);
          reverse(tempS.begin() + L2, tempS.begin() + R2 + 1);
          mySet.insert(tempS);
        }
      }
    }
  }

  for (int L3 = 0; L3 < N; L3++) {
    for (int R3 = L3; R3 < N; R3++) {
      for (int L4 = 0; L4 < N; L4++) {
        for (int R4 = L4; R4 < N; R4++) {
          string tempT = T;
          reverse(tempT.begin() + L3, tempT.begin() + R3 + 1);
          reverse(tempT.begin() + L4, tempT.begin() + R4 + 1);
          if (mySet.find(tempT) != mySet.end()) {
            return true;
          }
        }
      }
    }
  }
  return false;
}
```

S

L1 R1

L2 R2

```cpp
bool solve(string S, string T, int N) {
  set<string> mySet;
  for (int L1 = 0; L1 < N; L1++) {
    for (int R1 = L1; R1 < N; R1++) {
      for (int L2 = 0; L2 < N; L2++) {
        for (int R2 = L2; R2 < N; R2++) {
          string tempS = S;
          reverse(tempS.begin() + L1, tempS.begin() + R1 + 1);
          reverse(tempS.begin() + L2, tempS.begin() + R2 + 1);
          mySet.insert(tempS);
        }
      }
    }
  }

  for (int L3 = 0; L3 < N; L3++) {
    for (int R3 = L3; R3 < N; R3++) {
      for (int L4 = 0; L4 < N; L4++) {
        for (int R4 = L4; R4 < N; R4++) {
          string tempT = T;
          reverse(tempT.begin() + L3, tempT.begin() + R3 + 1);
          reverse(tempT.begin() + L4, tempT.begin() + R4 + 1);
          if (mySet.find(tempT) != mySet.end()) {
            return true;
          }
        }
      }
    }
  }
  return false;
}
```

Assume the time-complexity of set.insert and set.find are O(N*logN), what will the be the time complexity of solution?

A. O(N^4)
B. O(N^4*logN)
C. O(N^5)
D. O(N^5*logN)

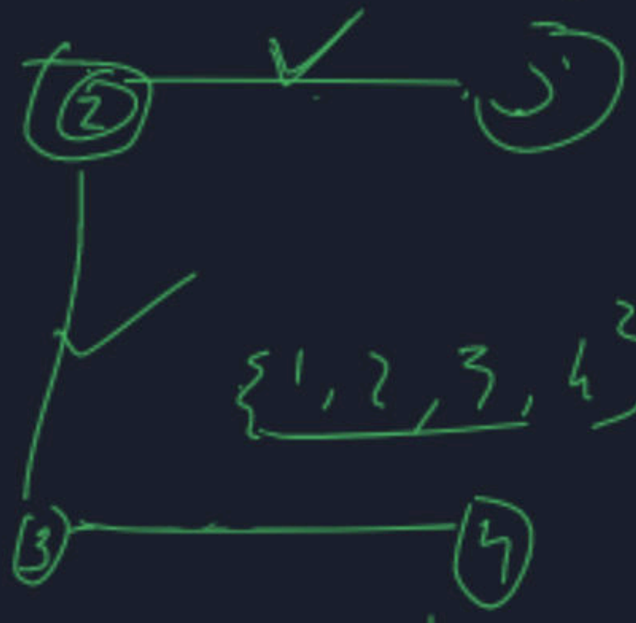
```
bool solve(string S, string T, int N) {
  set<string> mySet;
  for (int L1 = 0; L1 < N; L1++) {
    for (int R1 = L1; R1 < N; R1++) {
      for (int L2 = 0; L2 < N; L2++) {
        for (int R2 = L2; R2 < N; R2++) {
          string tempS = S;
          reverse(tempS.begin() + L1, tempS.begin() + R1 + 1);
          reverse(tempS.begin() + L2, tempS.begin() + R2 + 1);
          mySet.insert(tempS);
        }
      }
    }
  }

  for (int L3 = 0; L3 < N; L3++) {
    for (int R3 = L3; R3 < N; R3++) {
      for (int L4 = 0; L4 < N; L4++) {
        for (int R4 = L4; R4 < N; R4++) {
          string tempT = T;
          reverse(tempT.begin() + L3, tempT.begin() + R3 + 1);
          reverse(tempT.begin() + L4, tempT.begin() + R4 + 1);
          if (mySet.find(tempT) != mySet.end()) {
            return true;
          }
        }
      }
    }
  }
  return false;
}
```

{1, 2, 3} Minimum vertex cover => {2, 3} => 2

Vertex cover

Vertex cover of size



A set of vertices is called as a vertex cover if and only if for each edge at least one endpoint lies in the set

vertex cover => 2

You are given an undirected graph with **N** vertices and **M** edges. Find the size of the minimum vertex cover of the graph.

Constraints:

3 <= **N** <= 30

1 <= **M** <= 50

Vertex cover : Set of vertices is called as vertex cover if and only if for each edge Atleast one endpoint lies in the set

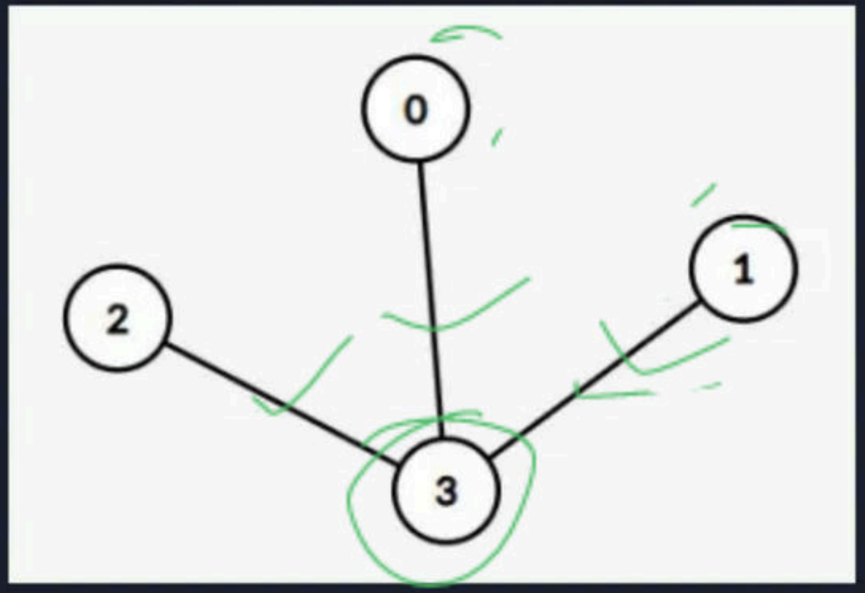
Min vertex cover : vertex cover of minimum size

N = 4
M = 3

{0, 1, 2} => vertex cover

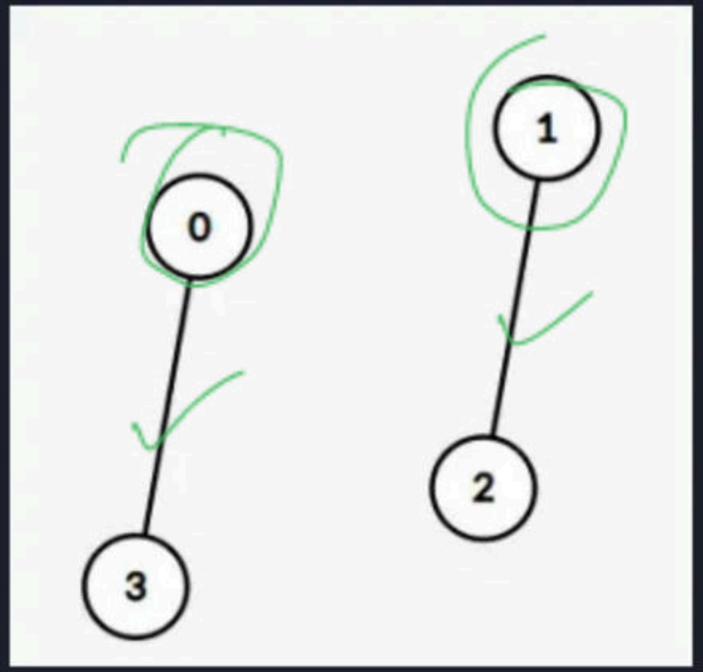
=> 3

~~{0, 1, 2}~~ {3} => 1



N = 4
M = 2

{0, 1} => 2



N = 6
M = 6

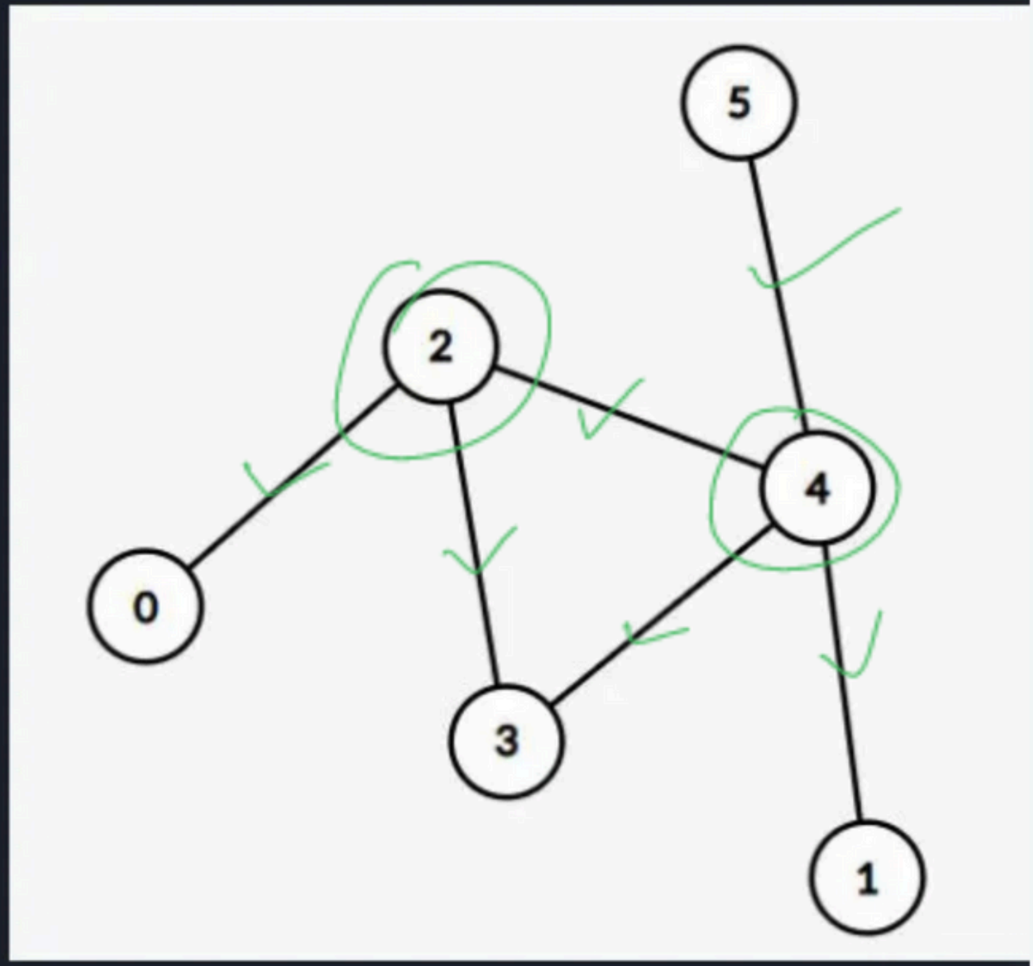
What will be the answer?

A. 1

B. 2

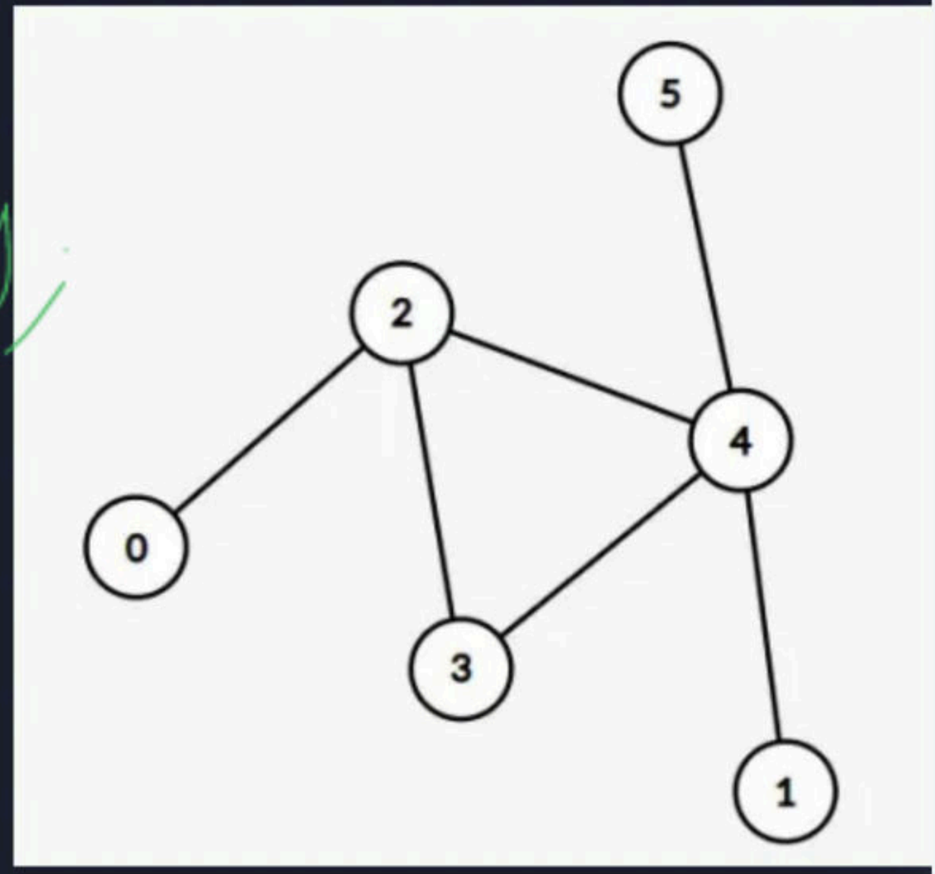
C. 3

D. 4



N = 6
M = 6

```
res = inf
for(each vertex cover Vc of graph) {
    res = min(res, Vc.size());
}
```



N = 6
M = 6

{0, 1, 4, 5}

{0, 2}

S = {0, 1, 2, ... N-1}

res = inf

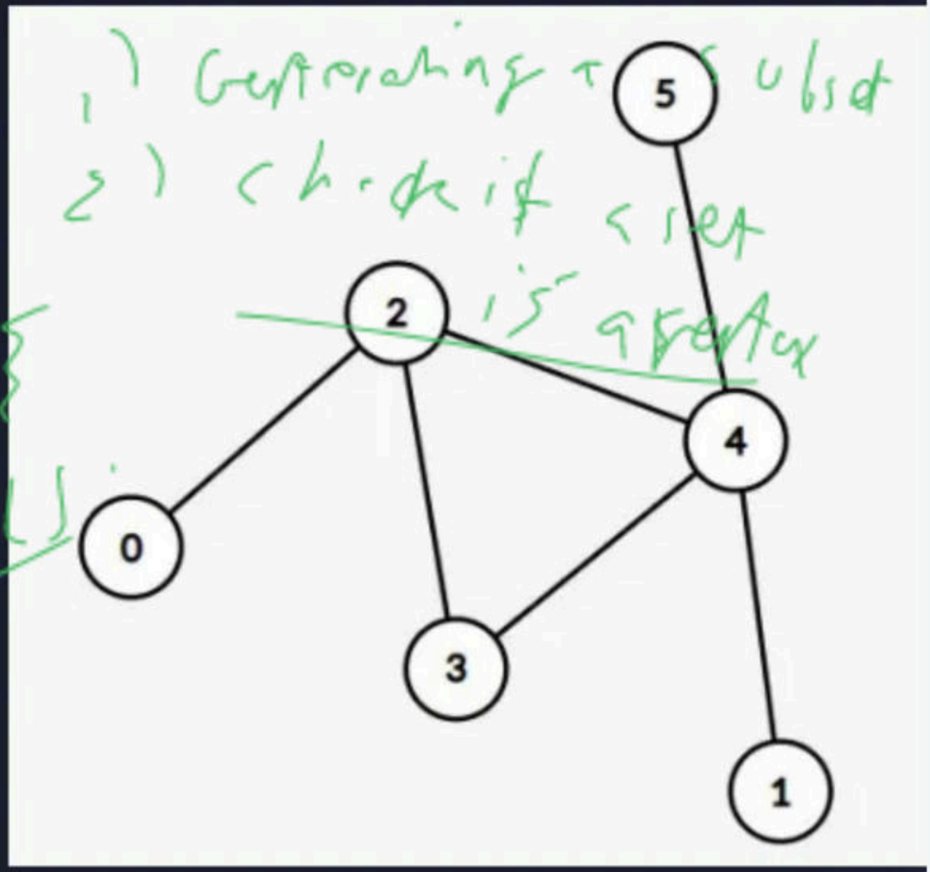
for (each subset S1 of S) {

if (S1 is a vertex cover) {

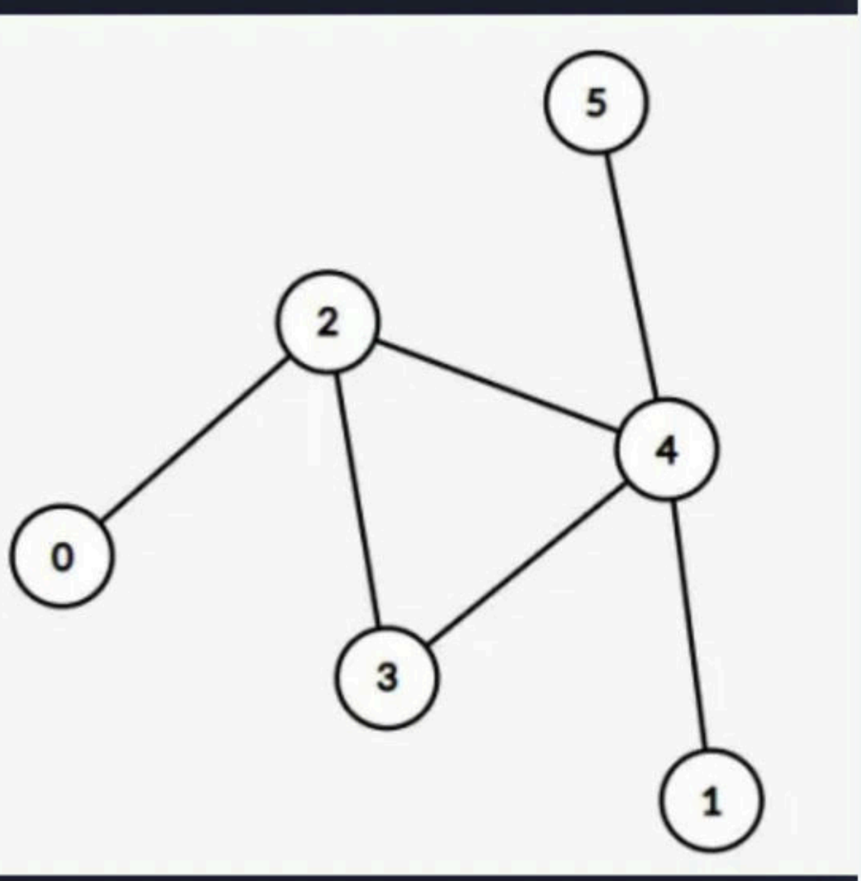
res = min(res, S1.size())

}

}



N = 6
M = 6



N = 6
M = 6

$O(2^N \times M)$

too high

10^4

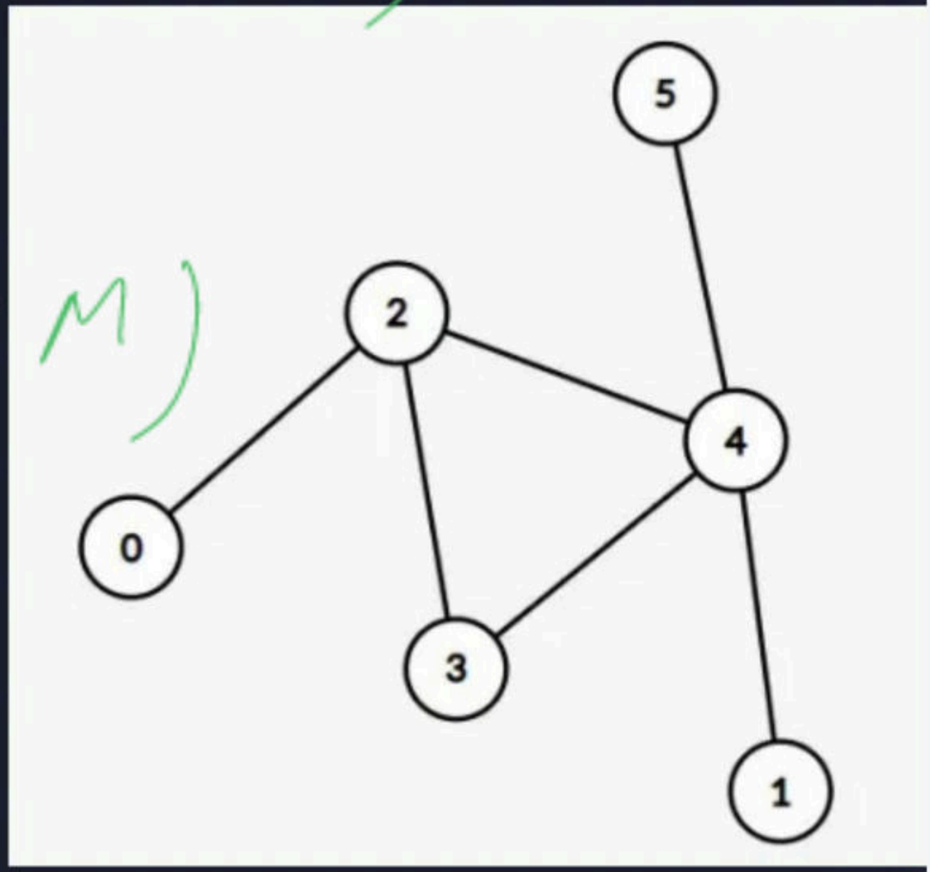
for (mask = 0; mask < 2^N ; mask++) { $O(2^N)$

S_1 = {set bits in mask}

if (S_1 is a vertex cover) → $O(M)$

res = min(res, S_1.size())

}



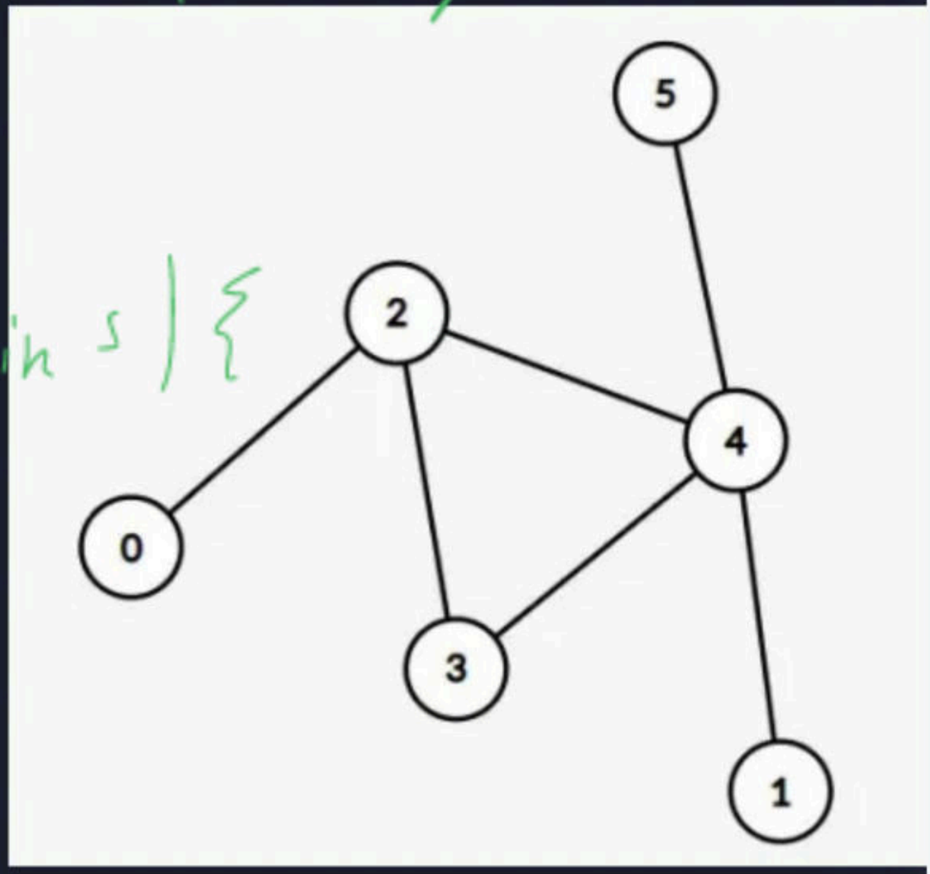
N = 6
M = 6

S = {0, 1, 4, 6}

O(M)

O(M x logN)

```
for (i=1; i<=M; i++) {
    a, b
    if (a is not in S && b is not in S) {
        return false;
    }
}
return true;
```



N = 6
M = 6

$S_{Left} = \{1, 2\} \cup \{0\}$

$S_{Left} = \{1, 2, 0\}$

$< \frac{N}{2}$

$\{0, 1, 2\}$

$> = \frac{N}{2}$

$\{3, 4, 5\}$

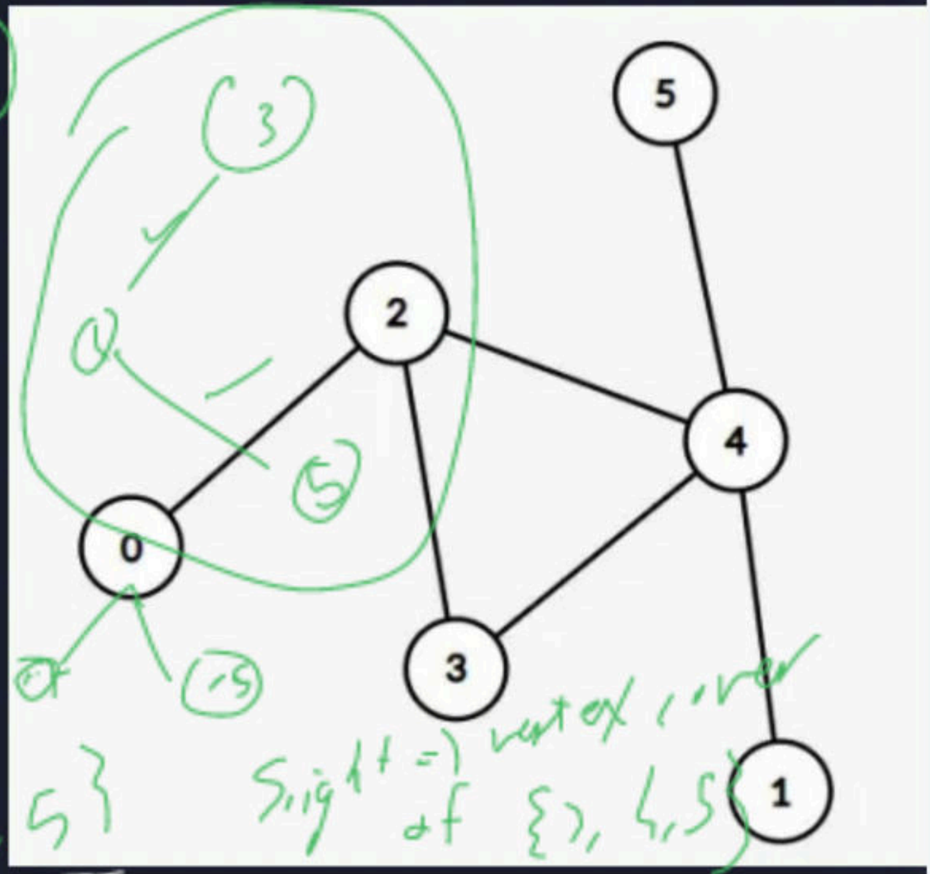
A

B

S_{Left}

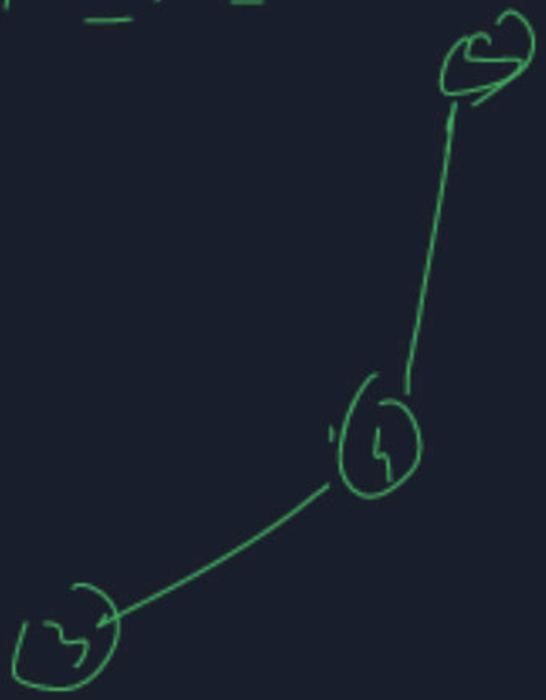
$S_{Right} = \{3,$

1) Edges inside A
2) Edges inside B
3) Edges between A and B

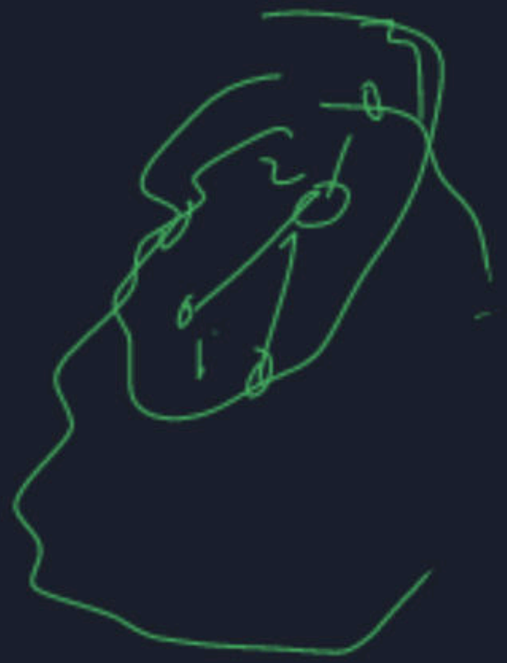


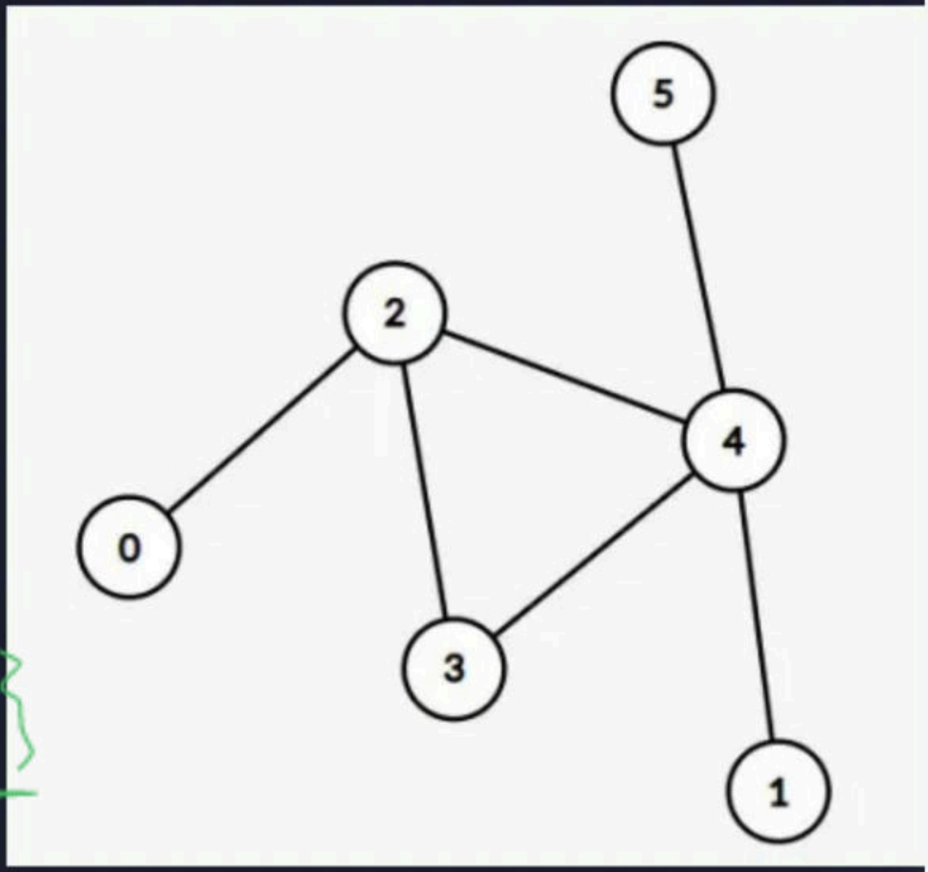
N = 6
M = 6

{3, 1, 5}



N = 6
M = 6





$O(3^{n/3} \times m)$

N = 6

M = 6

{0, 1, {4, 5}} ⇒

$V_{13} = \{4\}$

rem_{13} {3, 5}

$O(2^{n/3} \times m)$

1) Go through all vertex covers V_B of B

a) $S_{right} = \{V_3\}$ ✗

b) $rem_{13} = B - V_{13}$ ✓

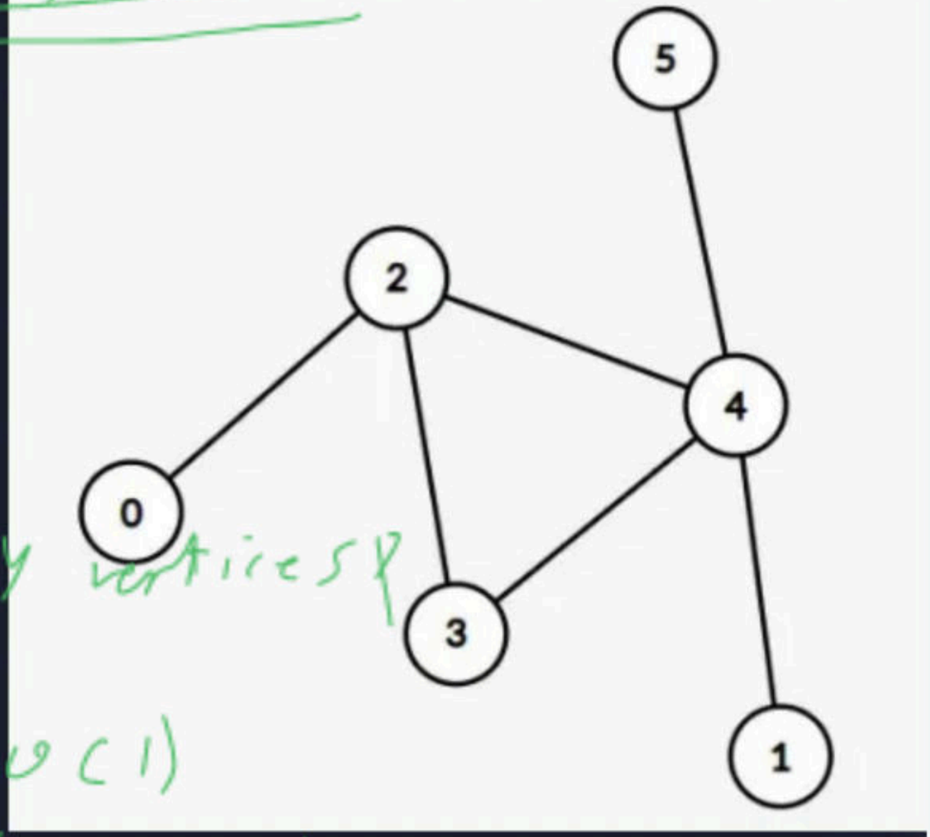
c) $adj_A = adj(rem_B)$ ✗

d) $S_{left} = \{adj_A\}$ # compulsory vertices ✗

e) $rem_A = A - S_{left}$

f) V_A = min vertex cover of rem_A $O(1)$

g) $S_{left} = \{adj_A\} \cup \{$min vert cover$\}$



N = 6
M = 6

Main problem

Min vertex cover of graph

A

Find min vertex cover for all subgraphs of A

B

Find all vertex covers of B

$O(3^{n/2} \times m)$

$O(3^{n/2} \times m + 2^{n/2} \times m)$

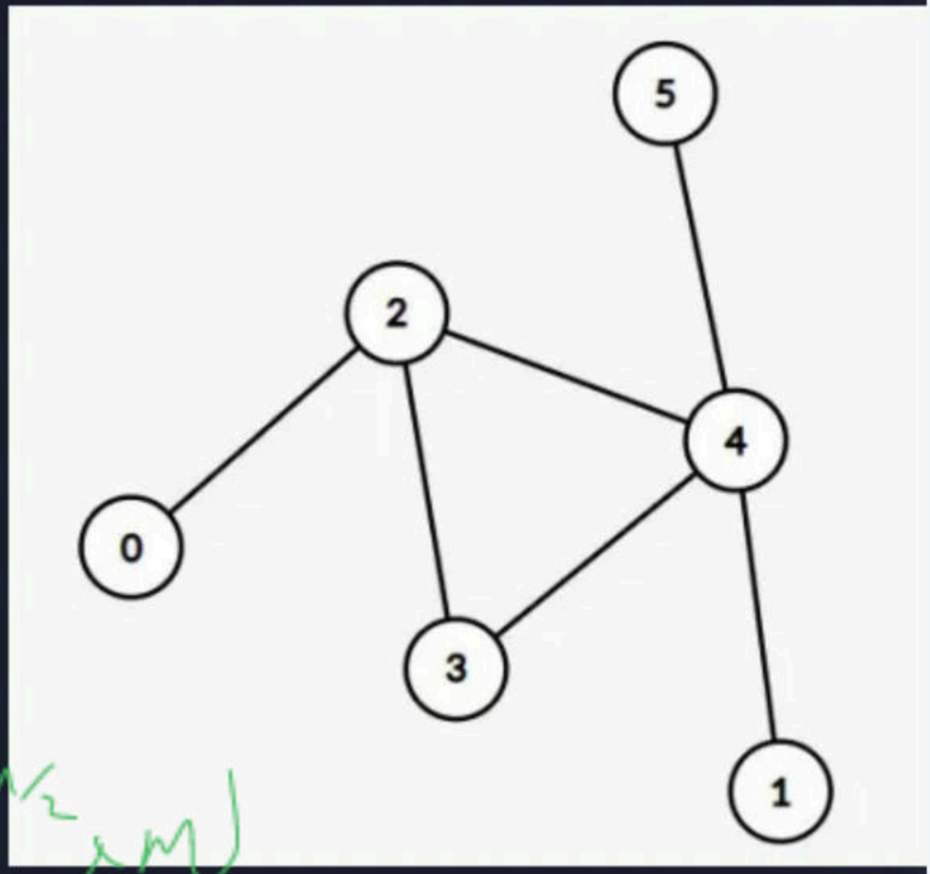
$O(2^{n/2} \times m)$

$O(3^{n/2} \times m)$

{0, 1}

{0, 2}

{1, 2}



N = 6
M = 6

B

$2^n \times M$

$2^{n/2} \times M$

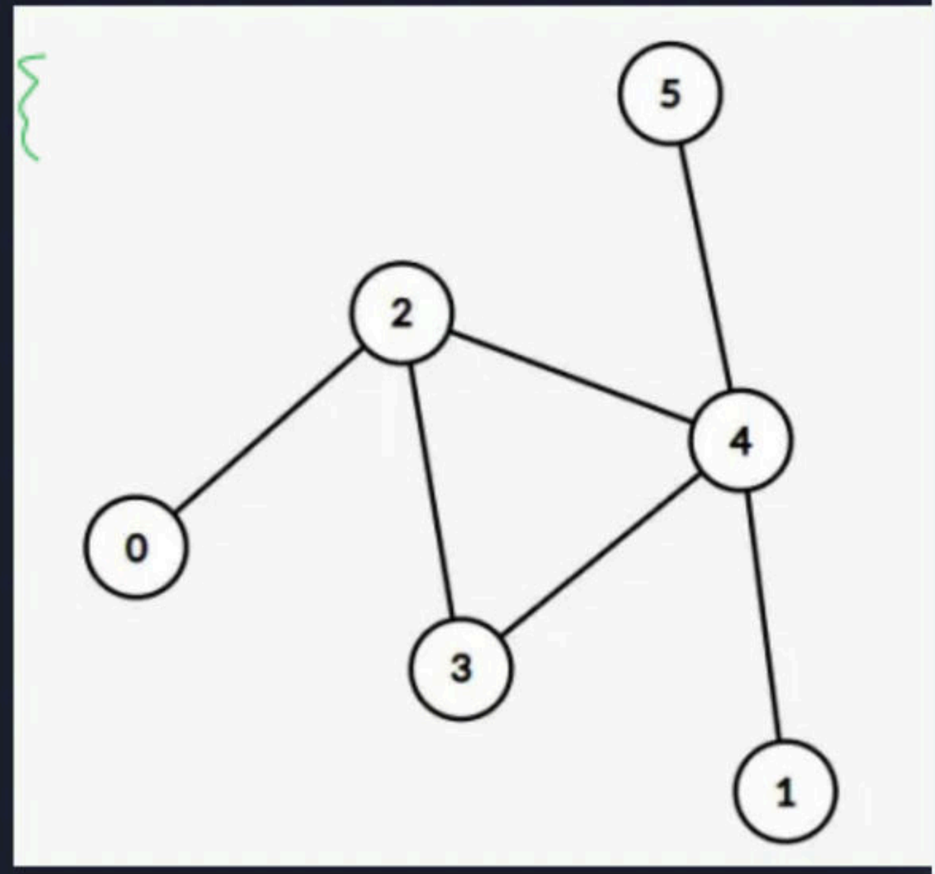
for each vertex cover V_{13} of B)

{

res.append(V_{13})

}

for(ma



$2^0 \times M + 2^1 \times M + 2^1 \times M + 2^2 \times M + 2^1 \times M$

N = 6

M = 6

$2^2 \times M + 2^2 \times M + 2^3 \times M$

$2^n \times M$

0 0 0 => 0

0 0 1 => 1

0 1 0 => 1

0 1 1 => 2

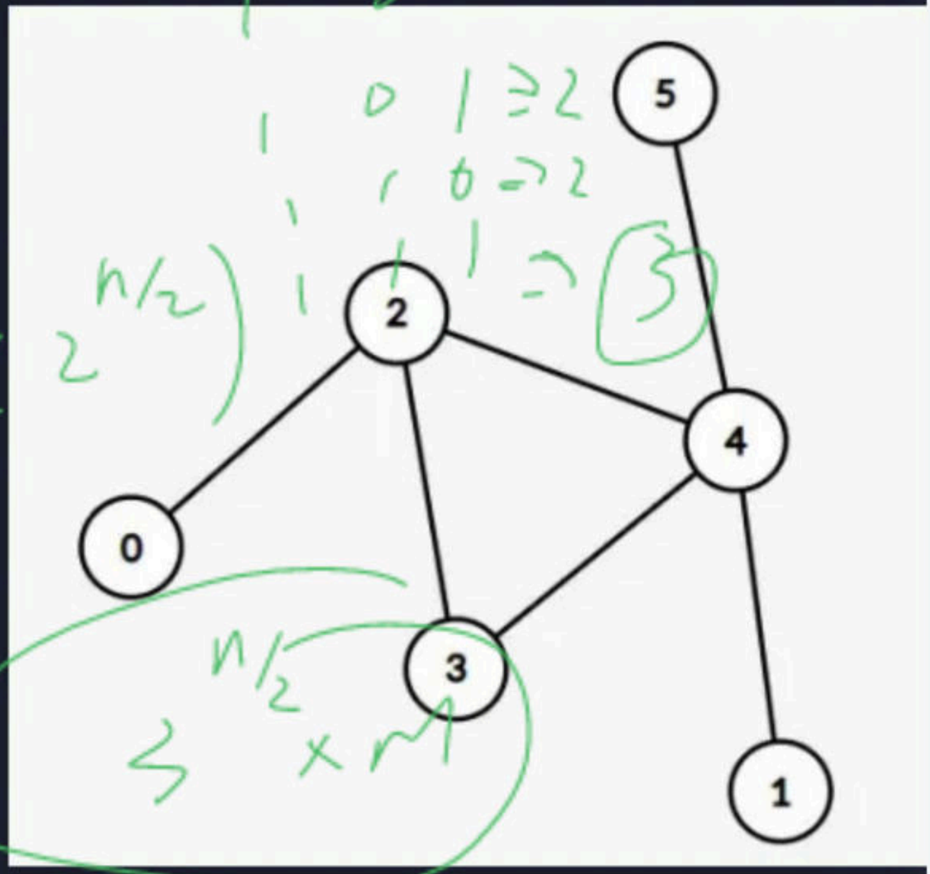
1 0 0 => 1

G with n vertex : cover = $2^n \times M$

for (each subgraph A' of A) { O(

dp[A'] = min vertex of A')

}



N = 6
M = 6

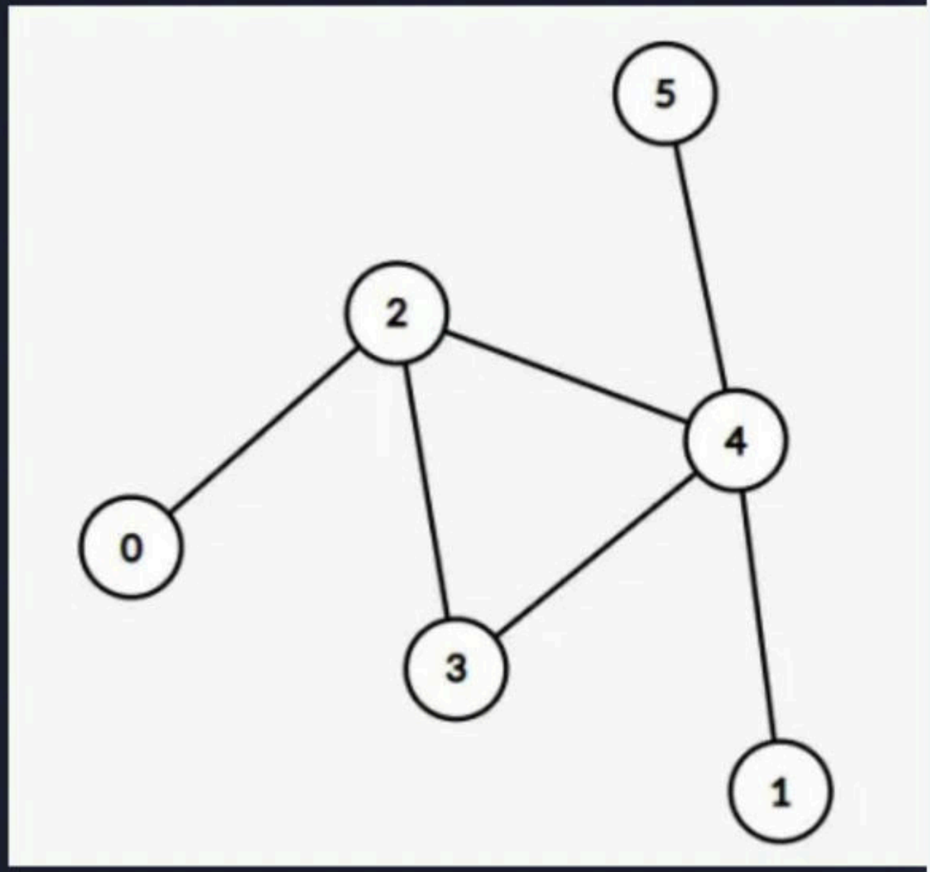
$n = 30$

$3^{15} \times 50$

$$\left(\sum_{i=0}^{2^n - 1} 2^{\#\text{set bits in } i} \right) \times M$$

$$\frac{}{N_r}$$

$3^{} \times M$



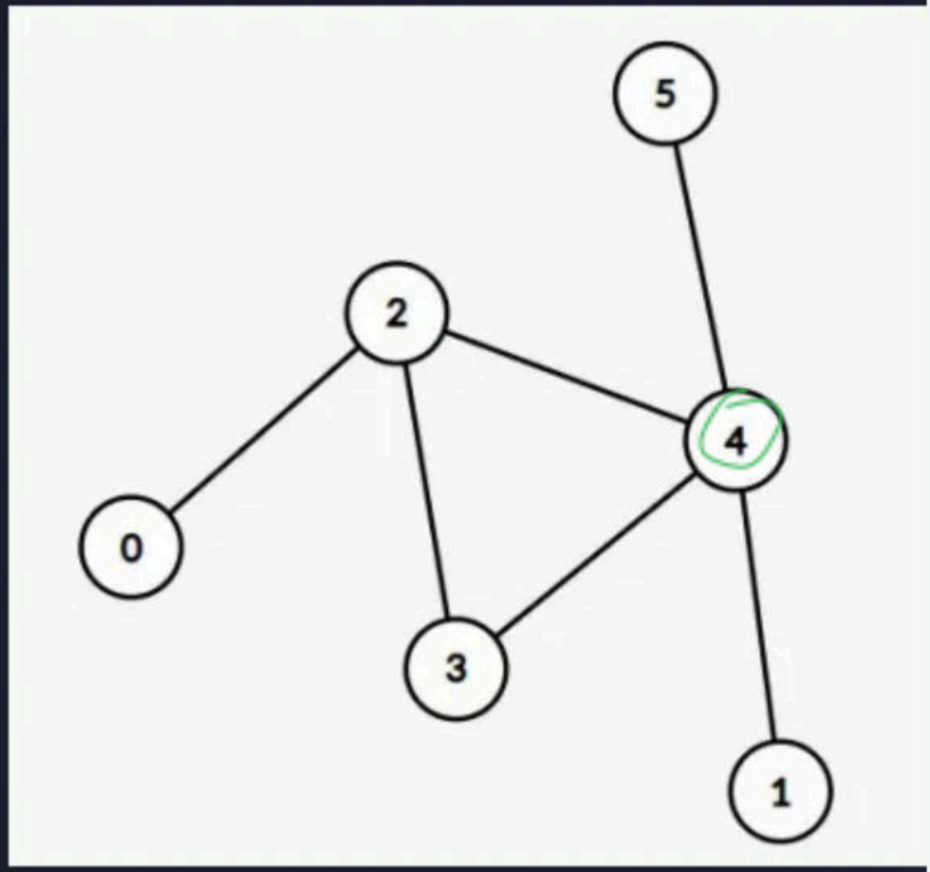
N = 6
M = 6

{0, 1, 2, 3, 4, 5}

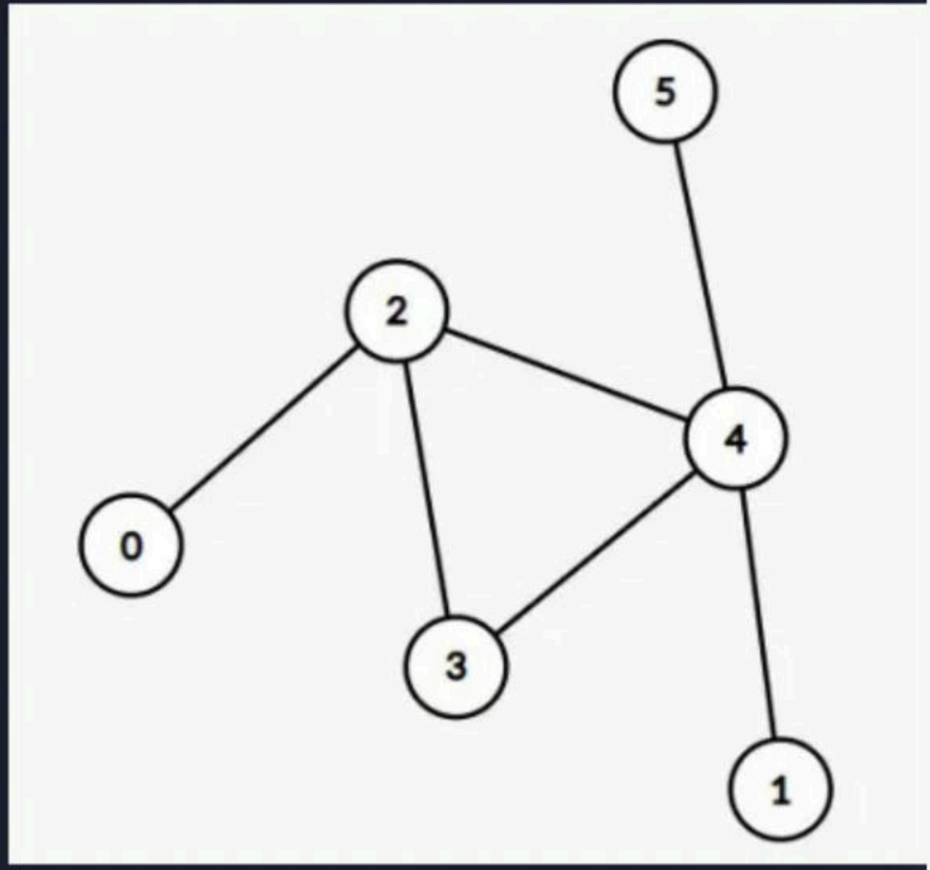
5 4 3 2 1 0

1 0 0 1 1 1

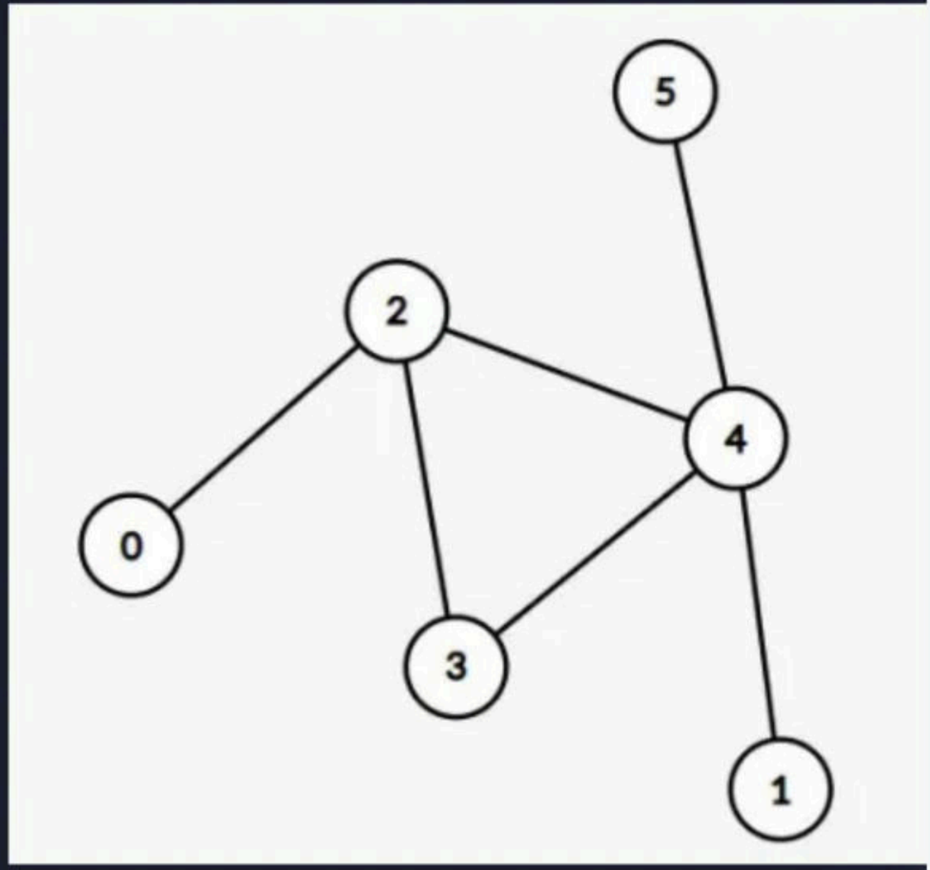
O(1)



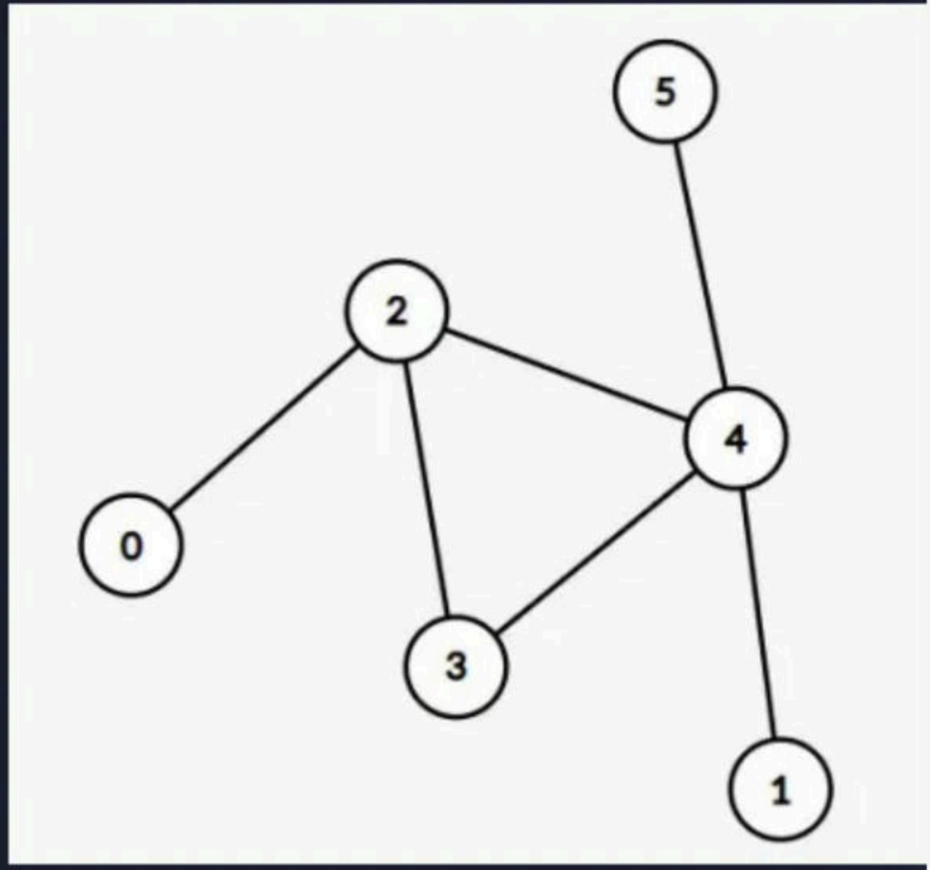
N = 6
M = 6



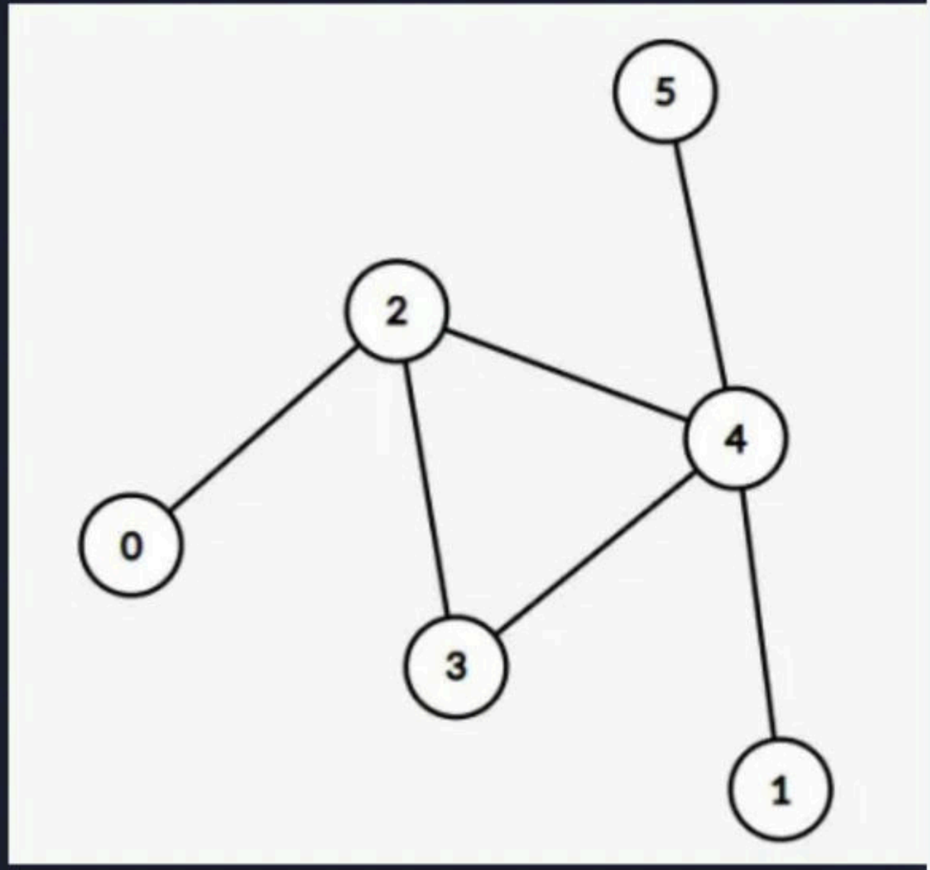
N = 6
M = 6



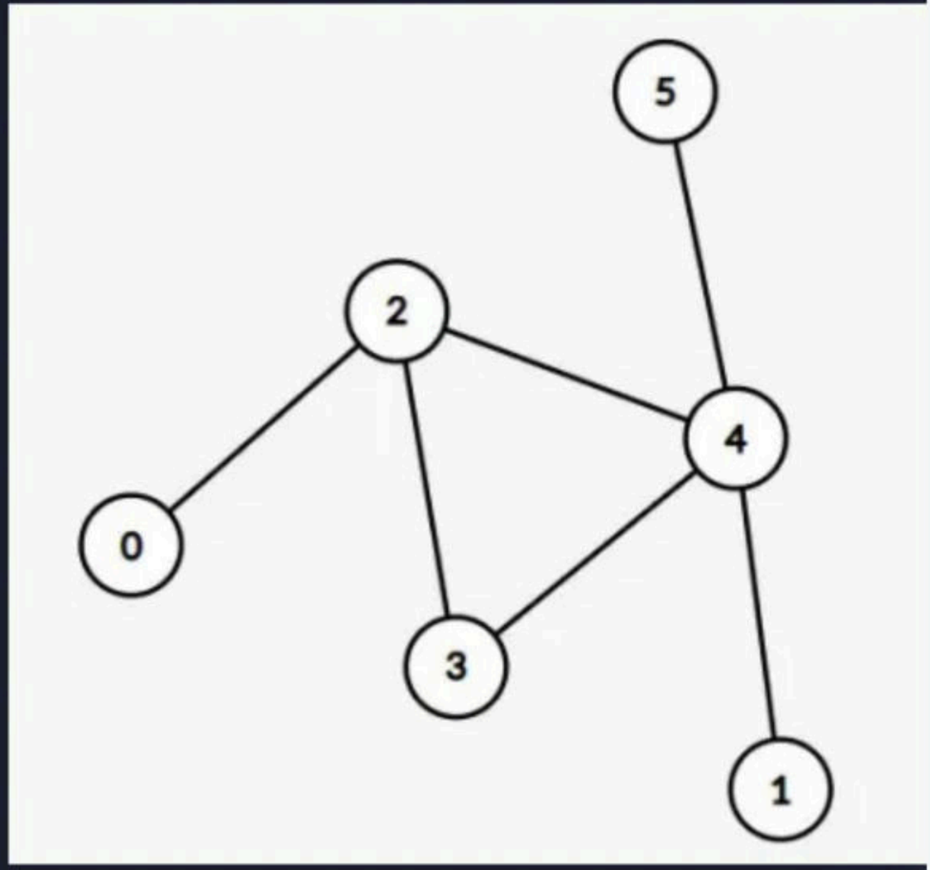
N = 6
M = 6



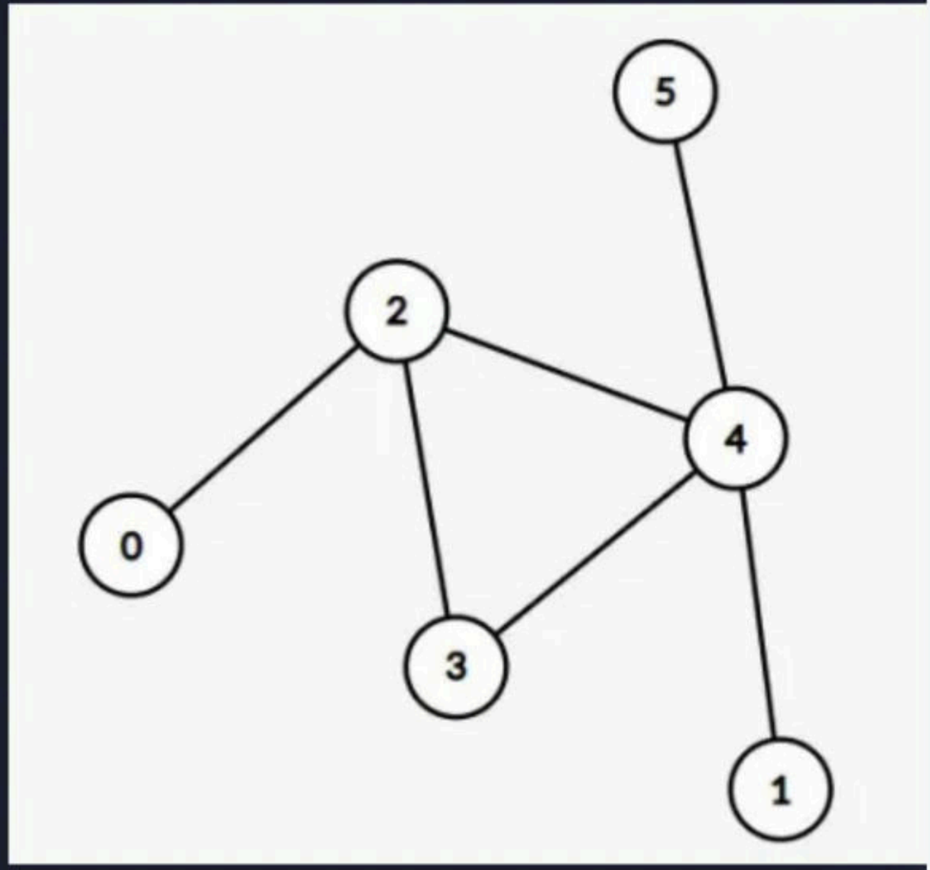
N = 6
M = 6



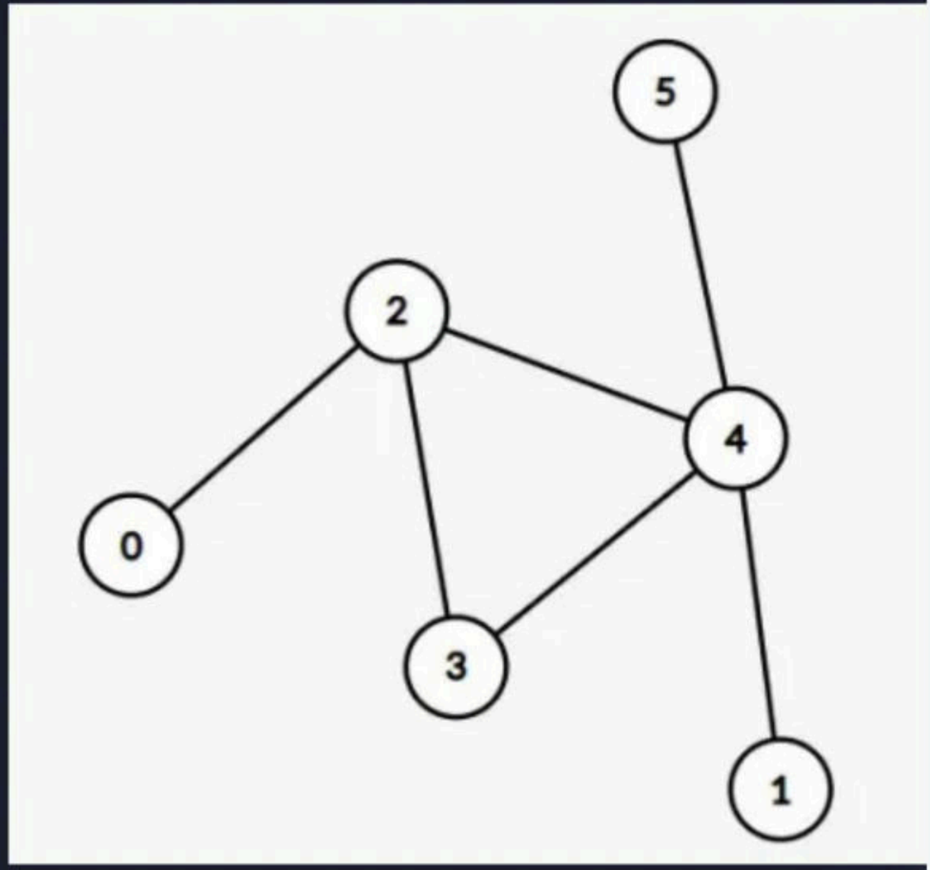
N = 6
M = 6



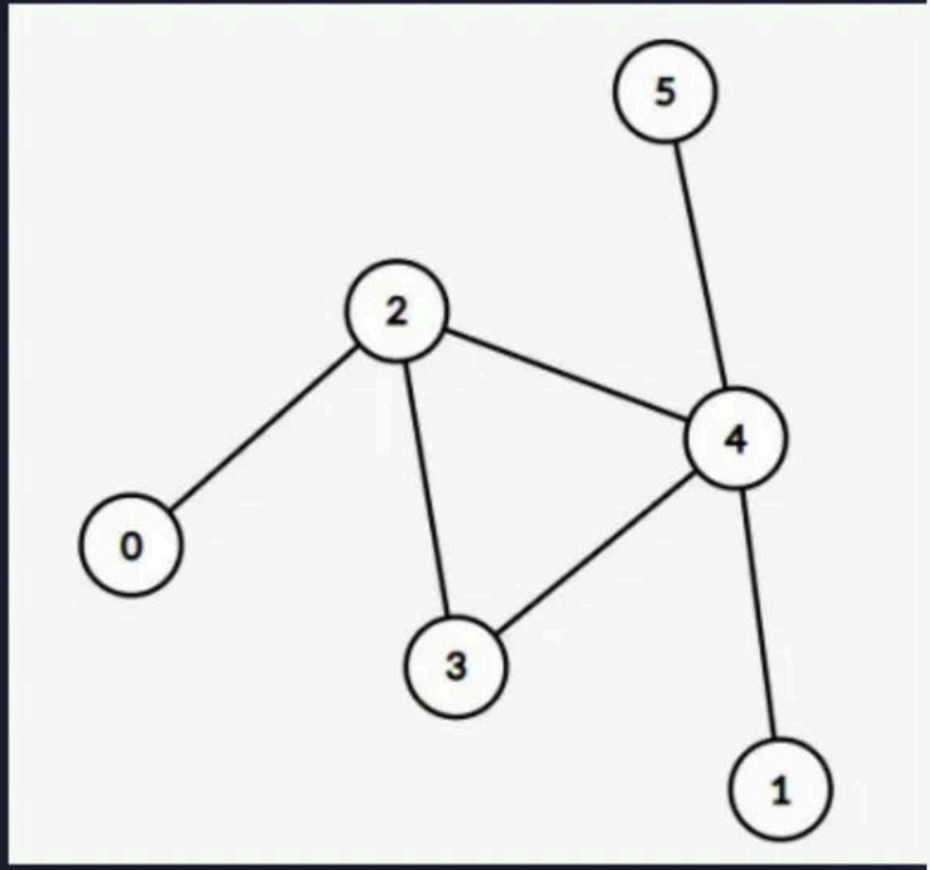
N = 6
M = 6



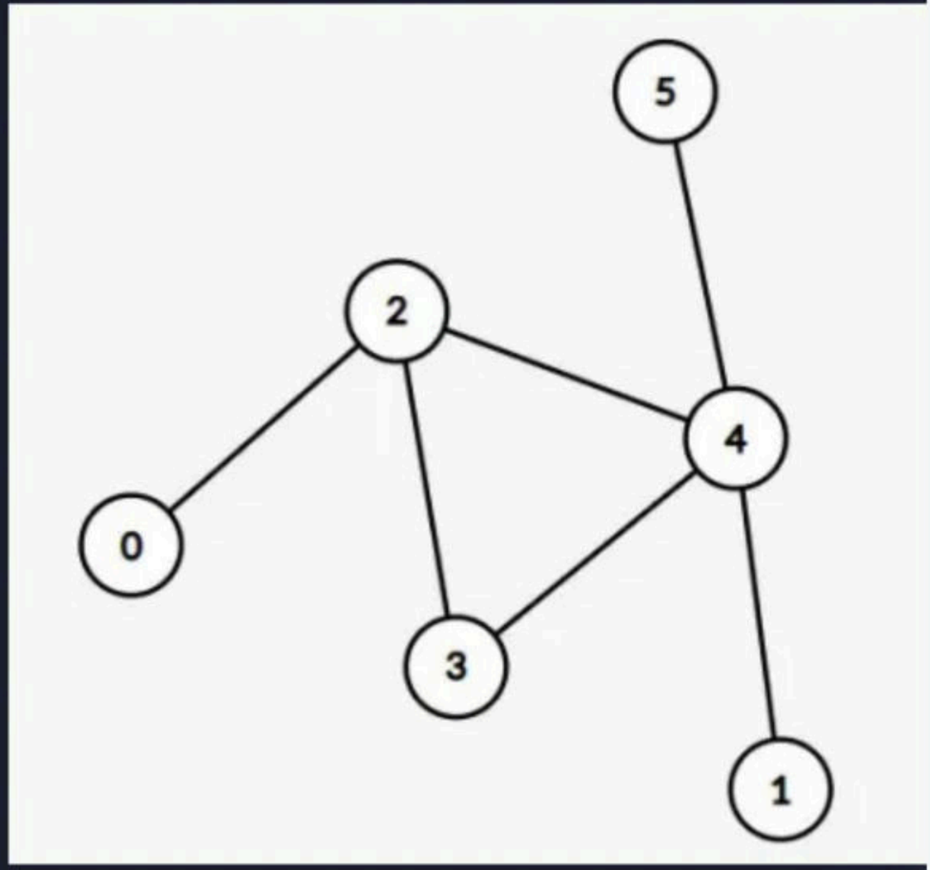
N = 6
M = 6



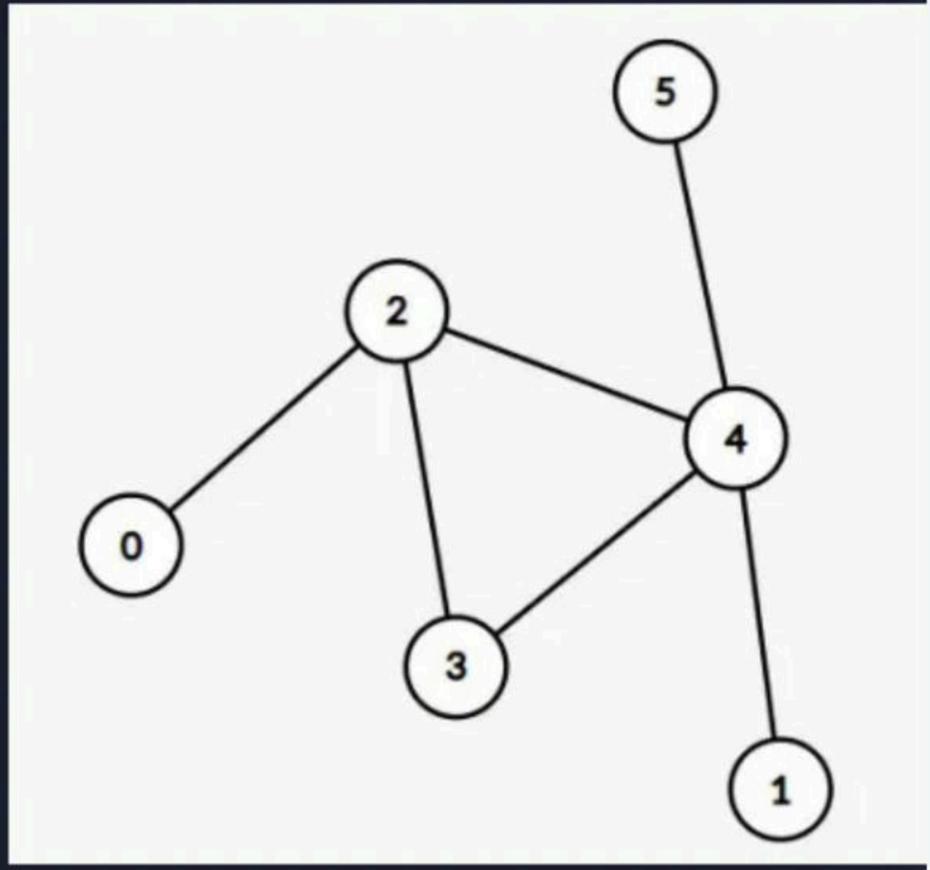
N = 6
M = 6



N = 6
M = 6

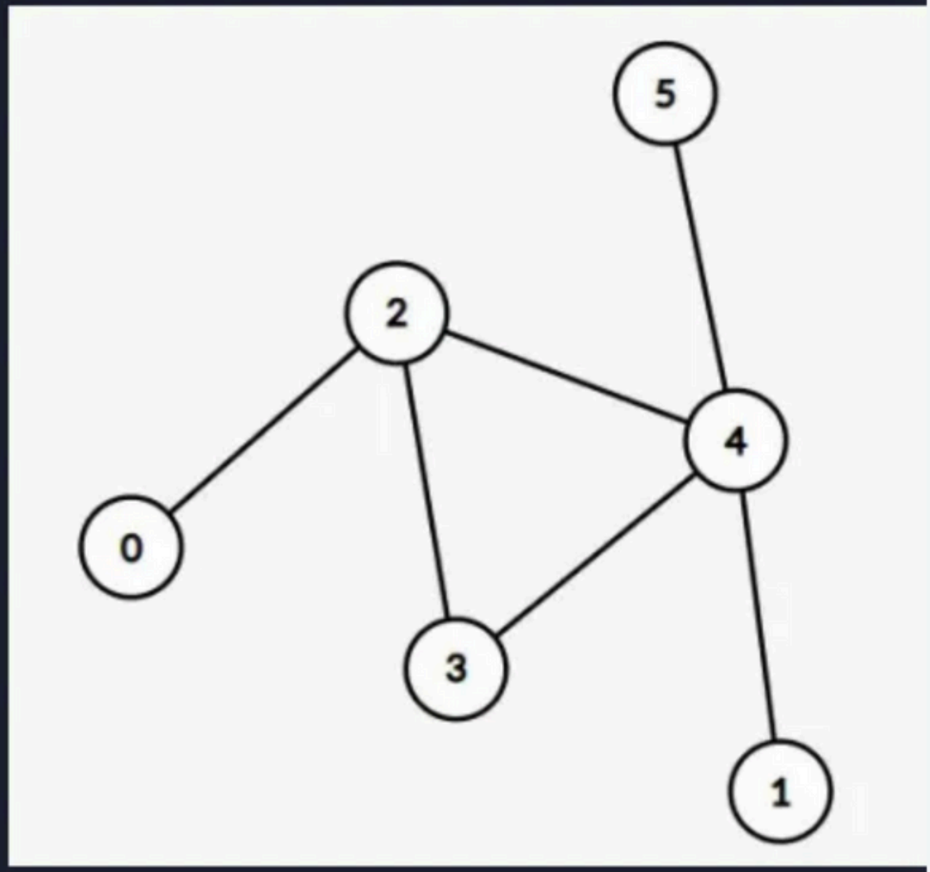


N = 6
M = 6



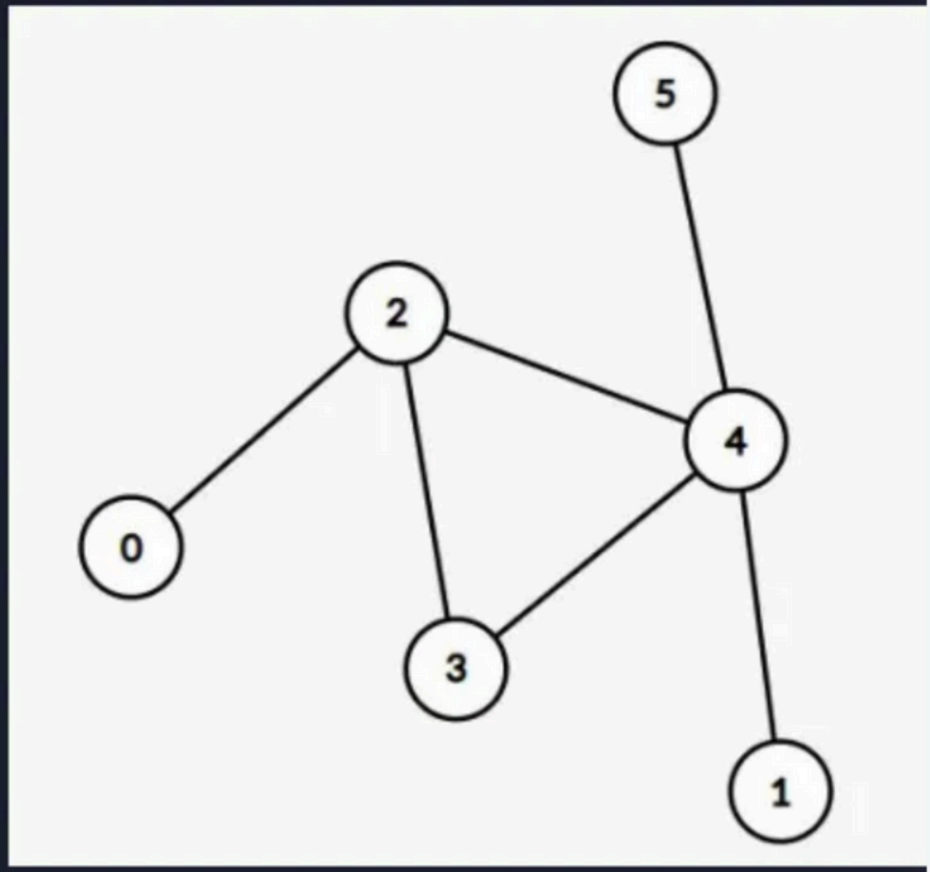
N = 6
M = 6

```
int res = A[0];
for (int i = mask; i > 0; i = (i - 1) & mask) {
  res += A[i];
}
```



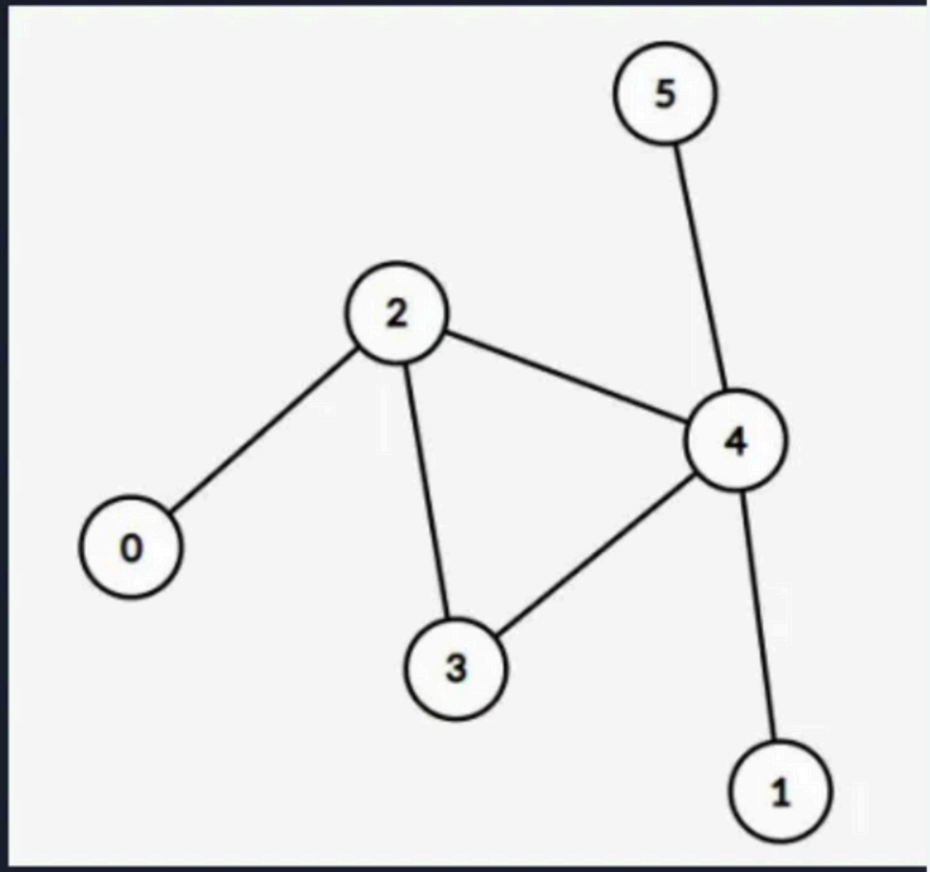
N = 6
M = 6

```
int res = A[0];
for (int i = mask; i > 0; i = (i - 1) & mask) {
  res += A[i];
}
```

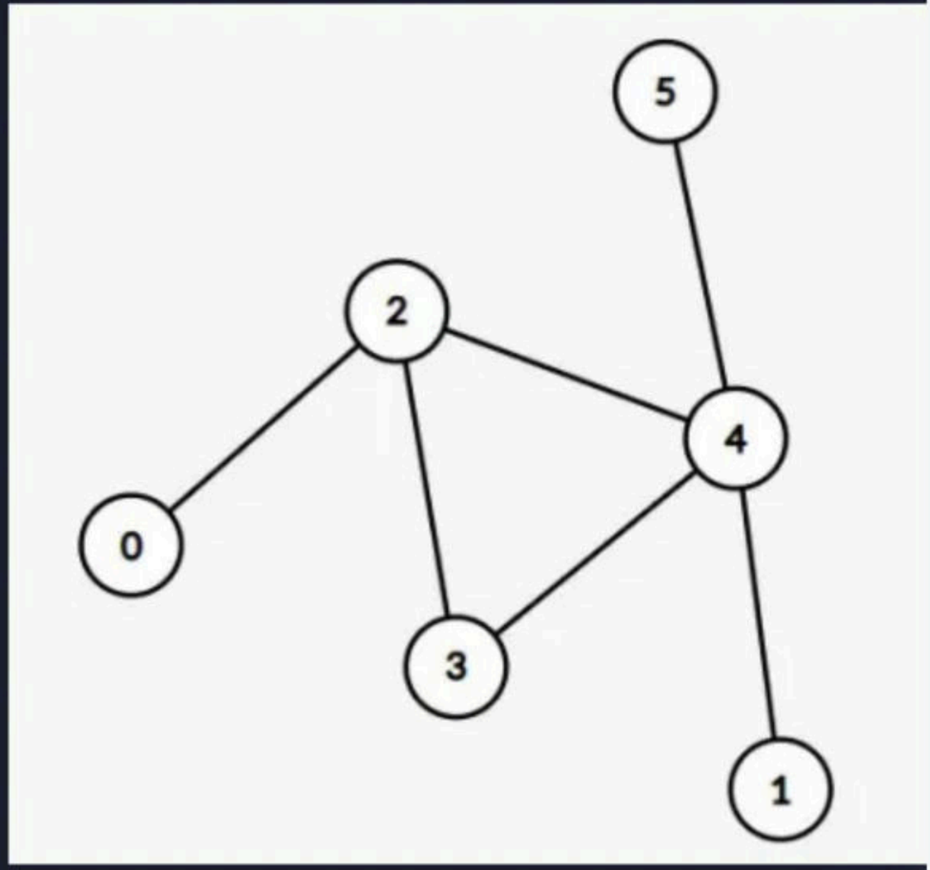


N = 6
M = 6

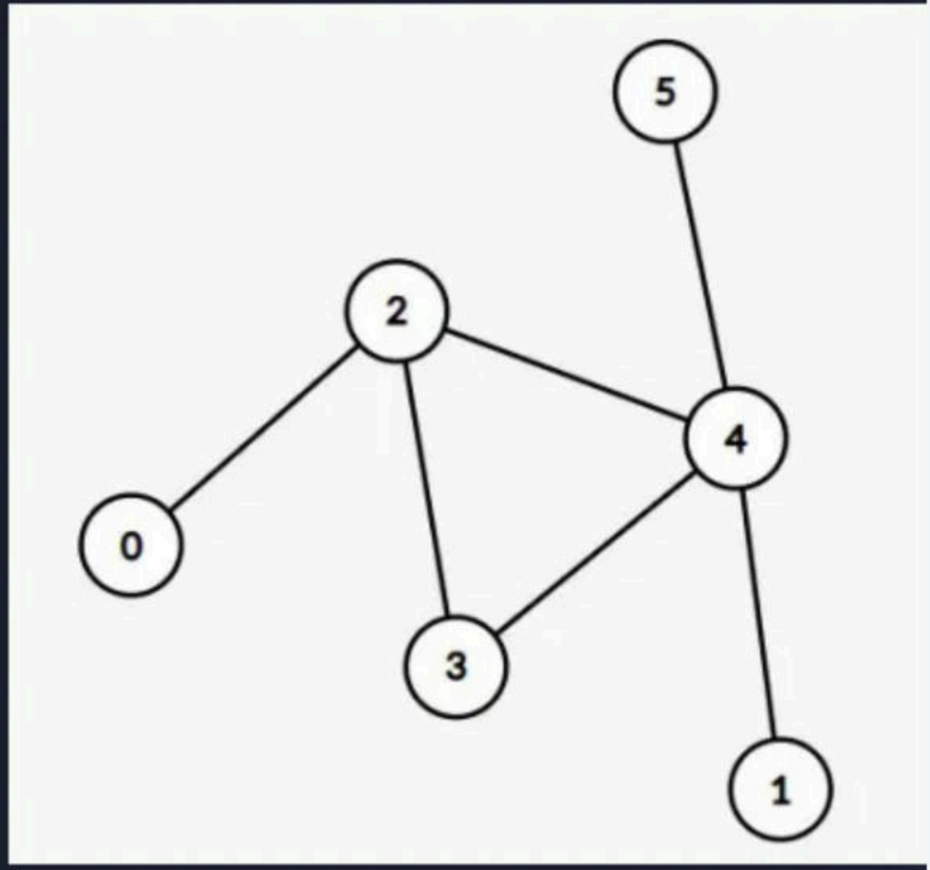
```
int res = A[0];
for (int i = mask; i > 0; i = (i - 1) & mask) {
  res += A[i];
}
```



N = 6
M = 6

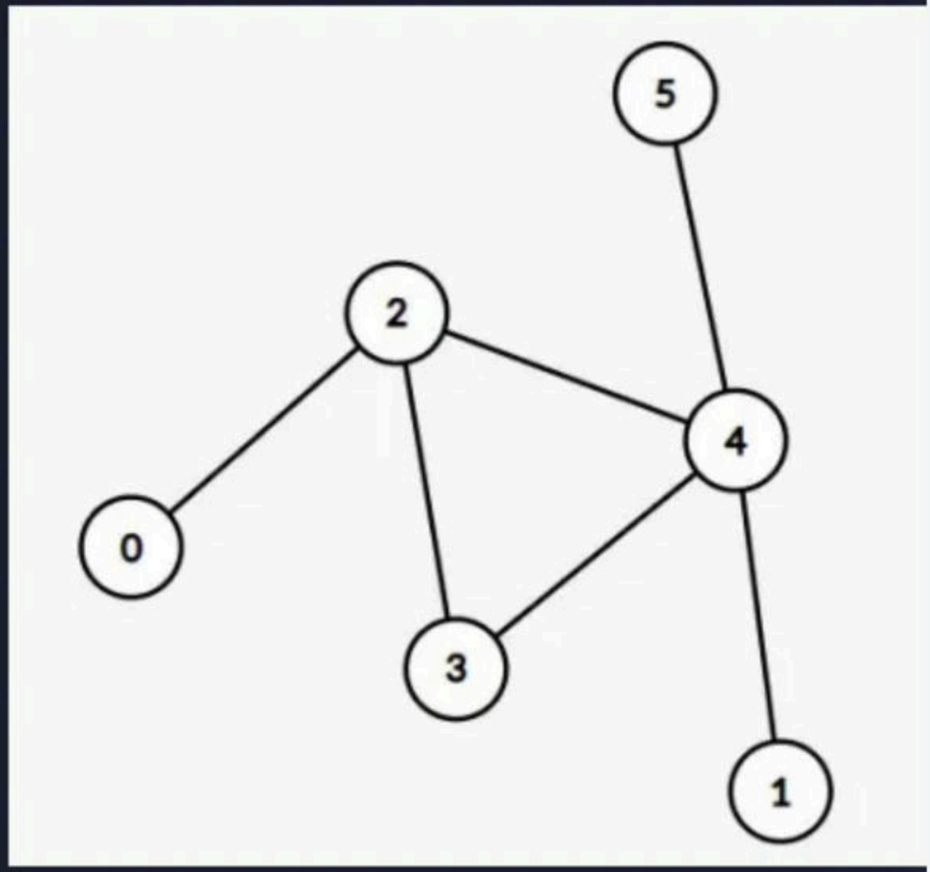


N = 6
M = 6

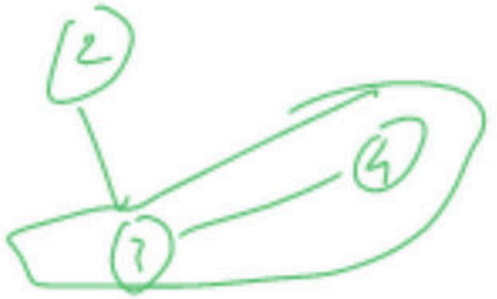


N = 6
M = 6

{2, 3}



```cpp
// this function checks whether <vertextCover> is the vertex cover of graph
// <graph>
bool check(int graph, int vertexCover, int edges[][2], int M) {
    for (int i = 1; i <= M; i++) {
        int a = edges[i][0];
        int b = edges[i][1];
        if (((graph >> a) & 1) == 0 or ((graph >> b) & 1) == 0) {
            // this edge doesn't lie in graph
            continue;
        }
        if (((vertexCover >> a) & 1) == 0 && ((vertexCover >> b) & 1) == 0) {
            // no edge vertex lies in vertex cover
            return false;
        }
    }
    return true;
}
```



```
// this function checks whether <vertextCover> is the vertex cover of graph
// <graph>
bool check(int graph, int vertexCover, int edges[][2], int M) {
  for (int i = 1; i <= M; i++) {
    int a = edges[i][0];
    int b = edges[i][1];
    if (((graph >> a) & 1) == 0 or ((graph >> b) & 1) == 0) {
      // this edge doesn't lie in graph
      continue;
    }
    if (((vertexCover >> a) & 1) == 0 && ((vertexCover >> b) & 1) == 0) {
      // no edge vertex lies in vertex cover
      return false;
    }
  }
  return true;
}
```

```
int solve(int edges[][2], int N, int M) {
  const int inf = 1e9;
  int leftSize = N / 2;
  int rightSize = N - leftSize;

  vector<int> adj[N + 5];
  for (int i = 1; i <= M; i++) {
    adj[edges[i][0]].push_back(edges[i][1]);
    adj[edges[i][1]].push_back(edges[i][0]);
  }

  int dp[(1 << leftSize) + 5];

  for (int graph = 0; graph < (1 << leftSize); graph++) {
    int minVertexConver = inf;

    if (check(graph, 0, edges, M))
      minVertexConver = 0;
    for (int vertexCover = graph; vertexCover > 0;
         vertexCover = (vertexCover - 1) & graph) {
      if (check(graph, vertexCover, edges, M)) {
        minVertexConver = min(minVertexConver, countSetBits(vertexCover));
      }
    }
    dp[graph] = minVertexConver;
  }

  int res = inf;
  for (int vertexCover = 0; vertexCover < (1 << rightSize); vertexCover++) {
    int graph = (1 << rightSize) - 1;
    if (!check(graph << leftSize, vertexCover, edges, M))
      continue;

    int rightVertexCover = countSetBits(vertexCover);
    int remGraph = graph ^ vertexCover;

    int leftVerticesToTake = 0;

    for (int remVertex = 0; remVertex < rightSize; remVertex++) {
      if ((remGraph >> remVertex) & 1) {
        int actualVertex = remVertex + leftSize;
        for (int &leftVertex : adj[actualVertex]) {
          if (leftVertex < leftSize)
            leftVerticesToTake |= (1 << leftVertex);
        }
      }
    }

    int leftVertexCover = countSetBits(leftVerticesToTake) +
                          dp[((1 << leftSize) - 1) ^ leftVerticesToTake];
    res = min(res, leftVertexCover + rightVertexCover);
  }
  return res;
}
```

```
int solve(int edges[][2], int N, int M) {
  const int inf = 1e9;
  int leftSize = N / 2;
  int rightSize = N - leftSize;

  vector<int> adj[N + 5];
  for (int i = 1; i <= M; i++) {
    adj[edges[i][0]].push_back(edges[i][1]);
    adj[edges[i][1]].push_back(edges[i][0]);
  }

  int dp[(1 << leftSize) + 5];

  for (int graph = 0; graph < (1 << leftSize); graph++) {
    int minVertexConver = inf;

    if (check(graph, 0, edges, M))
      minVertexConver = 0;
    for (int vertexCover = graph; vertexCover > 0;
         vertexCover = (vertexCover - 1) & graph) {
      if (check(graph, vertexCover, edges, M)) {
        minVertexConver = min(minVertexConver, countSetBits(vertexCover));
      }
    }
    dp[graph] = minVertexConver;
  }
  int res = inf;
  for (int vertexCover = 0; vertexCover < (1 << rightSize); vertexCover++) {
    int graph = (1 << rightSize) - 1;
    if (!check(graph << leftSize, vertexCover, edges, M))
      continue;

    int rightVertexCover = countSetBits(vertexCover);
    int remGraph = graph ^ vertexCover;

    int leftVerticesToTake = 0;

    for (int remVertex = 0; remVertex < rightSize; remVertex++) {
      if ((remGraph >> remVertex) & 1) {
        int actualVertex = remVertex + leftSize;
        for (int &leftVertex : adj[actualVertex]) {
          if (leftVertex < leftSize)
            leftVerticesToTake |= (1 << leftVertex);
        }
      }
    }

    int leftVertexCover = countSetBits(leftVerticesToTake) +
                          dp[((1 << leftSize) - 1) ^ leftVerticesToTake];
    res = min(res, leftVertexCover + rightVertexCover);
  }
  return res;
}
```

Summary