

图的存储

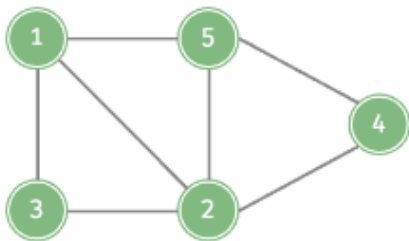
判定简单图: <https://hydro.ac/d/shalldream/p/35> 模板

度的数量: <https://hydro.ac/d/shalldream/p/34> 模板

图的基本概念

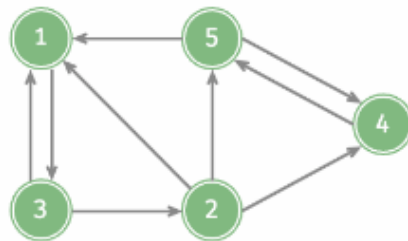
一个图 G 由 **顶点 (Vertex)** 的集合和 **边 (Edge)** 的集合组成, 记为 $G = (V, E)$, 其中:

- V (**顶点集合**): 图中所有顶点的集合, 通常记为 $V = v_1, v_2, \dots, v_n$ 。
- E (**边集合**): 图中所有边的集合, 表示顶点之间的连接关系, 通常记为 $E = e_1, e_2, \dots, e_m$ 。
- 无向图: 边没有方向, 表示顶点之间的双向关系。如果顶点 A 和顶点 B 之间有一条边, 则可以从 A 到 B , 也可以从 B 到 A 。
- 有向图: 边有方向, 表示顶点之间的单向关系。顶点 A 到顶点 B 的边表示只能从 A 到 B , 而不能从 B 到 A 。一般无向图可以看做特殊的有向图, 即 A, B 之间有一条边, 表示 A 到 B 有一条有向边, B 到 A 有一条有向边。



无向图

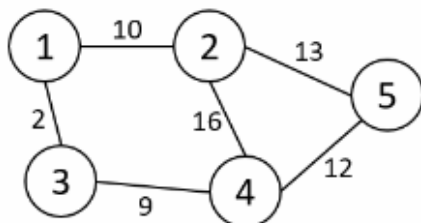
(边无方向)



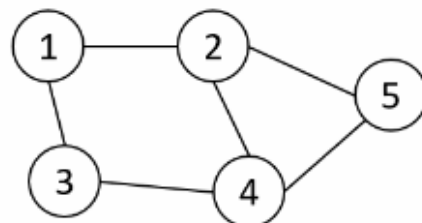
有向图

(边有方向)

- 加权图: 边具有权重 (或权值), 通常表示距离、时间、费用等。
- 无权图: 边没有权重, 所有边的权重默认为相同 (通常可以认为是 1)。

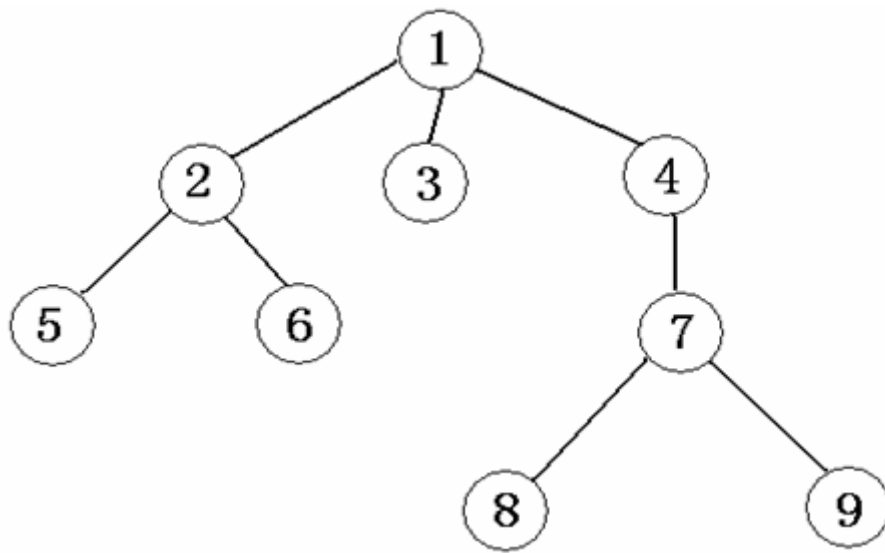


加权图

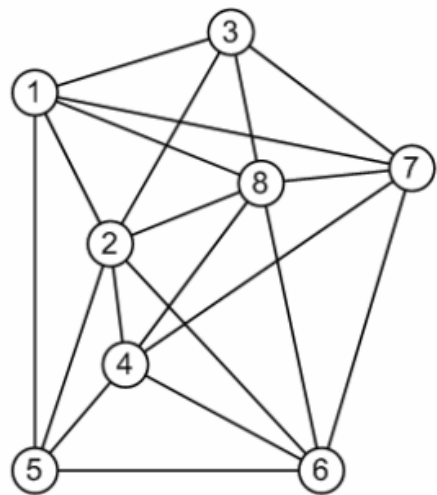
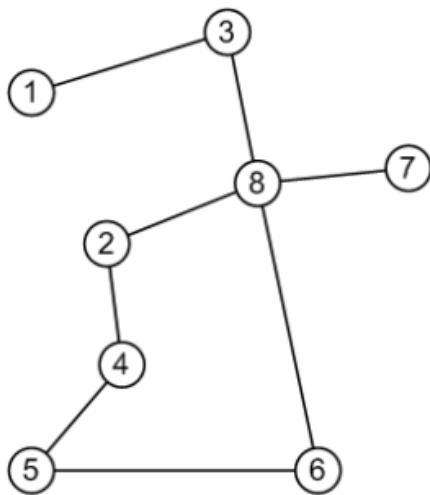


非加权图

- 树: 是一种特殊的无向图, 是一个无环连通图。树具有 n 个顶点 $n - 1$ 条边。



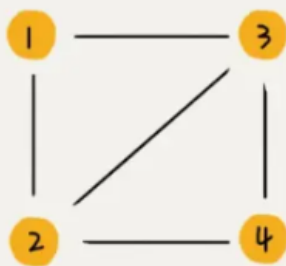
- **稀疏图**：边数接近点数。
- **稠密图**：边数远大于点数。



邻接矩阵存图

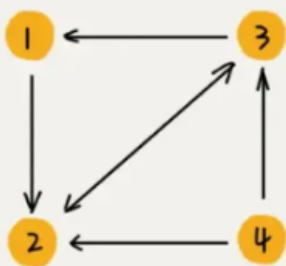
邻接矩阵，可以理解为是二维数组，一般用于存稠密图，初始时数组元素均为一个 (INF/0)，表示均互不可达。我们用 $g[a][b]=c$ ，代表点 a 到 b 有一条权值为 c 的有向边，若是无权图则用 $g[a][b]=1$ 表示。

无向图



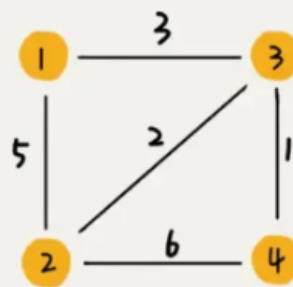
$$\text{adj} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

有向图



$$\text{adj} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

带权无向图



$$\text{adj} = \begin{bmatrix} 0 & 5 & 3 & 0 \\ 5 & 0 & 2 & 6 \\ 3 & 2 & 0 & 1 \\ 0 & 6 & 1 & 0 \end{bmatrix}$$

```

1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  const int N = 100; // 最大顶点数
6  const int INF = 1e9; // 设为无穷大, 表示不可达
7  int g[N][N]; // 邻接矩阵
8
9  void initGraph(int n) {
10     for (int i = 0; i < n; i++)
11         for (int j = 0; j < n; j++)
12             g[i][j] = (i == j ? 0 : INF); // 自环设为 0, 其他初始化为 INF
13 }
14
15 void addEdge(int a, int b, int c) {
16     g[a][b] = c; // 有向带权图
17     // g[b][a] = c; // 如果是无向图, 则取消注释
18 }
19
20 void printGraph(int n) {
21     for (int i = 0; i < n; i++) {
22         for (int j = 0; j < n; j++) {
23             if (g[i][j] == INF) cout << "INF ";
24             else cout << g[i][j] << " ";
25         }
26         cout << endl;
27     }
28 }
29
30 int main() {
31     int n = 4;
32     initGraph(n);
33
34     addEdge(0, 1, 4);
35     addEdge(0, 2, 2);
36     addEdge(1, 2, 5);
37     addEdge(1, 3, 7);

```

```

38     printGraph(n);
39     return 0;
40 }
41
42

```

```

1  import java.util.Arrays;
2
3  public class Main {
4      static final int N = 100; // 最大顶点数
5      static final int INF = (int) 1e9; // 设为无穷大, 表示不可达
6      static int[][] g = new int[N][N]; // 邻接矩阵
7
8      public static void main(String[] args) {
9          int n = 4;
10         initGraph(n);
11
12         addEdge(0, 1, 4);
13         addEdge(0, 2, 2);
14         addEdge(1, 2, 5);
15         addEdge(1, 3, 7);
16
17         printGraph(n);
18     }
19
20     static void initGraph(int n) {
21         for (int i = 0; i < n; i++) {
22             Arrays.fill(g[i], INF);
23             g[i][i] = 0; // 自环设为 0
24         }
25     }
26
27     static void addEdge(int a, int b, int c) {
28         g[a][b] = c; // 有向带权图
29         // g[b][a] = c; // 如果是无向图, 则取消注释
30     }
31
32     static void printGraph(int n) {
33         for (int i = 0; i < n; i++) {
34             for (int j = 0; j < n; j++) {
35                 System.out.print(g[i][j] == INF ? "INF " : g[i][j] + " ");
36             }
37             System.out.println();
38         }
39     }
40 }
41

```

```

1  N = 100 # 最大顶点数
2  INF = float('inf') # 设为无穷大, 表示不可达
3  g = [[INF] * N for _ in range(N)] # 邻接矩阵初始化
4
5  def init_graph(n):
6      for i in range(n):
7          g[i][i] = 0 # 自环设为 0
8

```

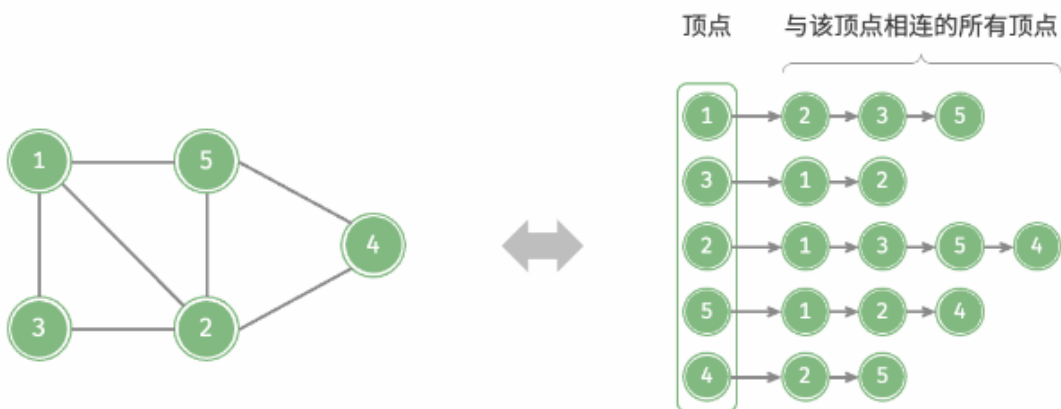
```

9  def add_edge(a, b, c):
10     g[a][b] = c # 有向带权图
11     # g[b][a] = c # 如果是无向图，则取消注释
12
13  def print_graph(n):
14     for i in range(n):
15         print(" ".join("INF" if g[i][j] == INF else str(g[i][j]) for j in
16             range(n)))
17
18  n = 4
19  init_graph(n)
20
21  add_edge(0, 1, 4)
22  add_edge(0, 2, 2)
23  add_edge(1, 2, 5)
24  add_edge(1, 3, 7)
25
26  print_graph(n)

```

邻接表存图

邻接表存图：定义一个数组，数组中每个元素都是一个扩容数组。



```

1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  const int N = 100; // 最大顶点数
7  vector<pair<int, int>> g[N]; // 邻接表，每个元素是 (目标顶点, 权重)
8
9  void addEdge(int u, int v, int w) {
10     g[u].emplace_back(v, w);
11     g[v].emplace_back(u, w); // 无向图
12 }
13
14 void printGraph(int n) {
15     for (int i = 0; i < n; i++) {
16         cout << "Vertex " << i << ": ";
17         for (auto [v, w] : g[i]) {
18             cout << "(" << v << ", " << w << ") ";
19         }
20     }
21 }

```

```

20         cout << endl;
21     }
22 }
23
24 int main() {
25     addEdge(0, 1, 4);
26     addEdge(0, 2, 2);
27     addEdge(1, 2, 5);
28     addEdge(1, 3, 7);
29
30     printGraph(4);
31     return 0;
32 }
33

```

```

1  import java.util.*;
2
3  public class Main {
4      static final int N = 100; // 最大顶点数
5      static List<int[]>[] g = new ArrayList[N]; // 每个 List 里的元素是 (目标顶
        点, 权重)
6
7      public static void main(String[] args) {
8          for (int i = 0; i < N; i++) g[i] = new ArrayList<>();
9          addEdge(0, 1, 4);
10         addEdge(0, 2, 2);
11         addEdge(1, 2, 5);
12         addEdge(1, 3, 7);
13
14         printGraph(4);
15     }
16
17     static void addEdge(int u, int v, int w) {
18         g[u].add(new int[]{v, w});
19         g[v].add(new int[]{u, w}); // 无向图
20     }
21
22     static void printGraph(int n) {
23         for (int i = 0; i < n; i++) {
24             System.out.print("vertex " + i + ": ");
25             for (int[] edge : g[i]) {
26                 System.out.print("(" + edge[0] + ", " + edge[1] + ") ");
27             }
28             System.out.println();
29         }
30     }
31 }
32

```

```

1  N = 100 # 最大顶点数
2  g = [[] for _ in range(N)] # 邻接表, 存 (目标顶点, 权重)
3
4  def add_edge(u, v, w):
5      g[u].append((v, w))
6      g[v].append((u, w)) # 无向图
7

```

```

8  def print_graph(n):
9      for i in range(n):
10         print(f"Vertex {i}: {[v, w] for v, w in g[i]}")
11
12  add_edge(0, 1, 4)
13  add_edge(0, 2, 2)
14  add_edge(1, 2, 5)
15  add_edge(1, 3, 7)
16
17  print_graph(4)
18

```

判定简单图

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int N=5e2+10;
4  int g[N][N];
5  int n,m;
6  int main(){
7      scanf("%d%d",&n,&m);
8      for(int i=1;i<=m;i++){
9          int a,b;
10         scanf("%d%d",&a,&b);
11         if(a==b){
12             printf("No\n");
13             return 0;
14         }
15         if(g[a][b]==1) {
16             printf("No\n");
17             return 0;
18         }
19         g[a][b]=g[b][a]=1;
20     }
21     printf("Yes");
22 }

```

```

1  import java.util.Scanner;
2
3  public class Main {
4      static final int N = 502;
5      static int[][] g = new int[N][N];
6
7      public static void main(String[] args) {
8          Scanner scanner = new Scanner(System.in);
9          int n = scanner.nextInt();
10         int m = scanner.nextInt();
11
12         for (int i = 0; i < m; i++) {
13             int a = scanner.nextInt();
14             int b = scanner.nextInt();
15
16             if (a == b || g[a][b] == 1) {
17                 System.out.println("No");
18                 return;
19             }
20         }
21     }
22 }

```

```

20
21         g[a][b] = g[b][a] = 1;
22     }
23
24     System.out.println("Yes");
25     scanner.close();
26 }
27 }
28

```

```

1  import sys
2
3  N = 502
4  g = [[0] * N for _ in range(N)]
5
6  n, m = map(int, sys.stdin.readline().split())
7
8  for _ in range(m):
9      a, b = map(int, sys.stdin.readline().split())
10
11     if a == b or g[a][b] == 1:
12         print("No")
13         sys.exit()
14
15     g[a][b] = g[b][a] = 1
16
17 print("Yes")
18

```

度的数量

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  int n,m;
4  vector<int> g[1000005];
5  int main(){
6      cin>>n>>m;
7      for(int i=1;i<=m;i++){
8          int x,y;
9          cin>>x>>y;
10         g[x].push_back(y);
11         g[y].push_back(x);
12     }
13     for(int i=1;i<=n;i++){
14         cout<<g[i].size()<<" ";
15     }
16     return 0;
17 }

```

```

1  import java.util.ArrayList;
2  import java.util.Scanner;
3
4  public class Main {
5      static int n, m;
6      static ArrayList<Integer>[] g;

```



```

7
8     public static void main(String[] args) {
9         Scanner scanner = new Scanner(System.in);
10        n = scanner.nextInt();
11        m = scanner.nextInt();
12
13        g = new ArrayList[n + 1]; // 由于节点编号从 1 开始, 因此 n+1
14
15        for (int i = 1; i <= n; i++) {
16            g[i] = new ArrayList<>();
17        }
18
19        for (int i = 0; i < m; i++) {
20            int x = scanner.nextInt();
21            int y = scanner.nextInt();
22            g[x].add(y);
23            g[y].add(x);
24        }
25
26        for (int i = 1; i <= n; i++) {
27            System.out.print(g[i].size() + " ");
28        }
29
30        scanner.close();
31    }
32 }
33

```

```

1  import sys
2
3  n, m = map(int, sys.stdin.readline().split())
4  g = [[] for _ in range(n + 1)]
5
6  for _ in range(m):
7      x, y = map(int, sys.stdin.readline().split())
8      g[x].append(y)
9      g[y].append(x)
10
11  print(" ".join(str(len(g[i])) for i in range(1, n + 1)))
12

```