Graph Representation (C++)

#csspree Online

Representation Methods

Representation Methods

1. Adjacency Matrix

Representation Methods

1. Adjacency Matrix
2. Adjacency List

Representation Methods

1. Adjacency Matrix
2. Adjacency List
3. Edge List

1. Adjacency Matrix

1. Adjacency Matrix

for n nodes, use n x n matrix to store edge information

1. Adjacency Matrix

for n nodes, use n x n matrix to store edge information

```
int graph[5][5];
```

1. Adjacency Matrix

for n nodes, use n x n matrix to store edge information

```
int graph[5][5];
```

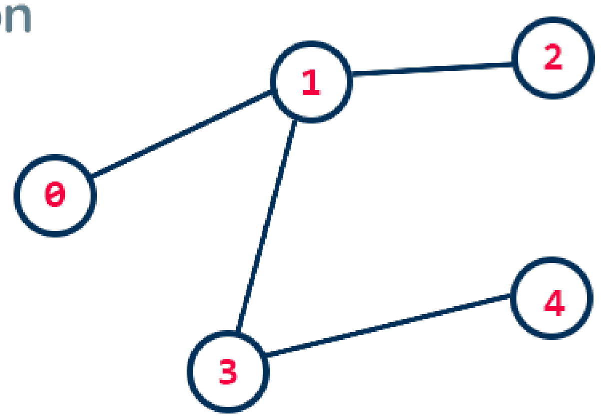
	0	1	2	3	4
0					
1					
2					
3					
4					

1. Adjacency Matrix

for n nodes, use n x n matrix to store edge information

```
int graph[5][5];
```

	0	1	2	3	4
0					
1					
2					
3					
4					

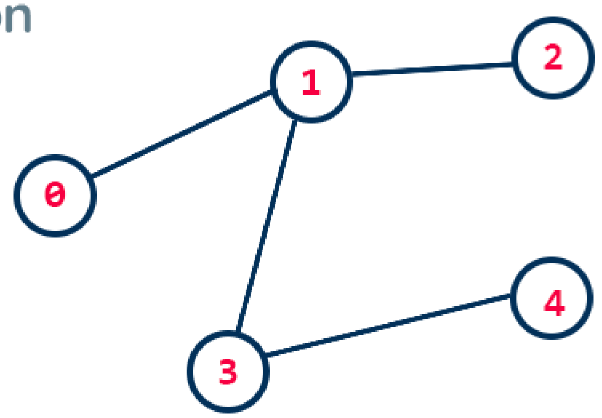


1. Adjacency Matrix

for n nodes, use n x n matrix to store edge information

```c
int graph[5][5];
```

	0	1	2	3	4
0					
1					
2					
3					
4					



Input:

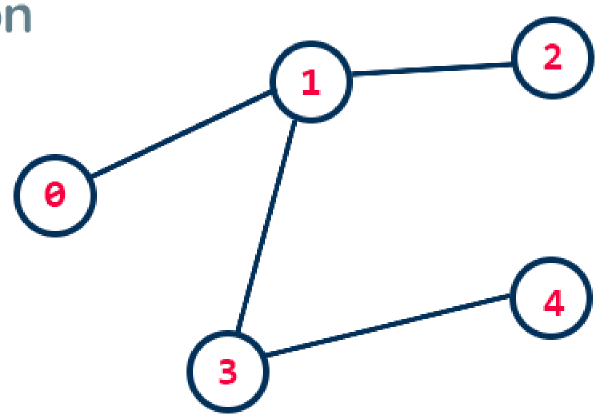
```
5 4    // number of nodes = 5, number of edges = 4
```

1. Adjacency Matrix

for n nodes, use n x n matrix to store edge information

```
int graph[5][5];
```

	0	1	2	3	4
0					
1					
2					
3					
4					



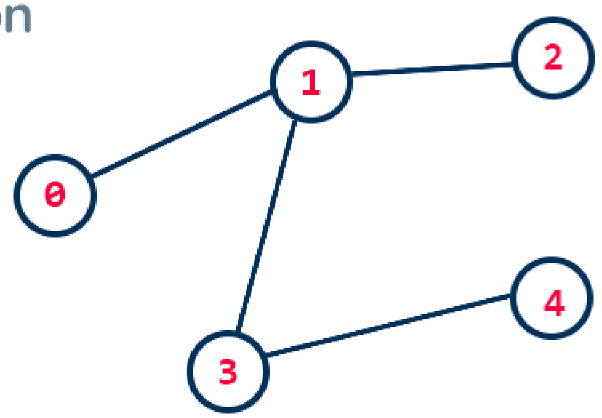
```
Input:

5 4     // number of nodes = 5, number of edges = 4
0 1
1 2     // u v (there is an edge between u, v)
1 3
3 4
```

1. Adjacency Matrix

for n nodes, use n x n matrix to store edge information

```
int graph[5][5];
```

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & & 1 & & & \\ 1 & 1 & & & & \\ 2 & & & & & \\ 3 & & & & & \\ 4 & & & & & \end{array}$$



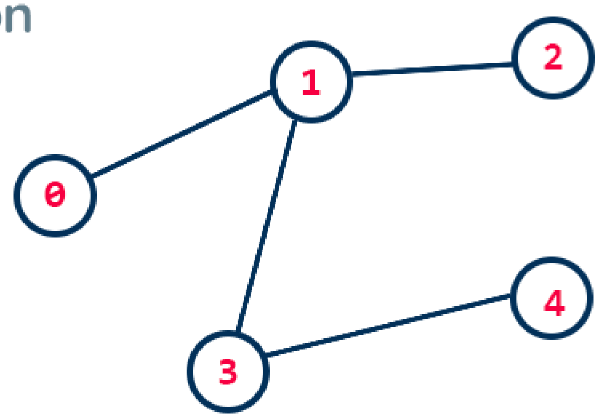
```
Input:

5 4     // number of nodes = 5, number of edges = 4
0 1
1 2     // graph[0][1] = 1; graph[1][0] = 1;
1 3
3 4
```

1. Adjacency Matrix

for n nodes, use n x n matrix to store edge information

```
int graph[5][5];
```

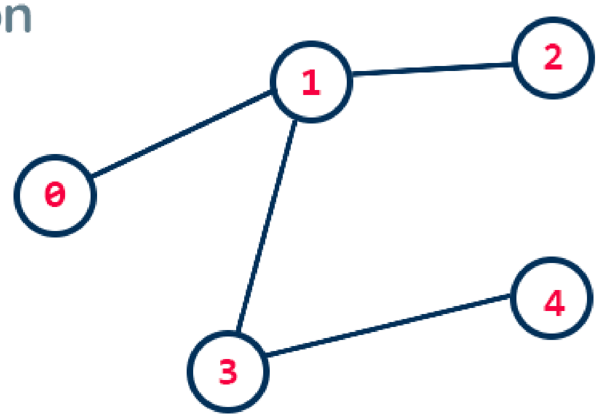
$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & & 1 & & & \\ 1 & 1 & & 1 & & \\ 2 & & 1 & & & \\ 3 & & & & & \\ 4 & & & & & \end{array}$$



```
Input:

5 4     // number of nodes = 5, number of edges = 4
0 1
1 2     // graph[1][2] = 1; graph[2][1] = 1;
1 3
3 4
```

1. Adjacency Matrix

for n nodes, use n x n matrix to store edge information



```
int graph[5][5];
```

	0	1	2	3	4
0		1			
1	1		1	1	
2		1			
3		1			
4					

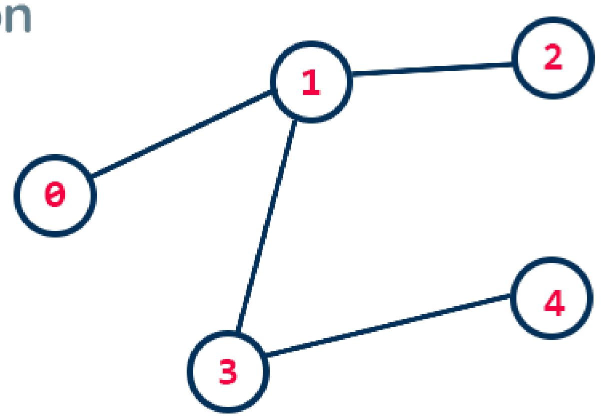
```
Input:

5 4    // number of nodes = 5, number of edges = 4
0 1
1 2    // graph[1][3] = 1; graph[3][1] = 1;
1 3
3 4
```

1. Adjacency Matrix

for n nodes, use n x n matrix to store edge information

```
int graph[5][5];
```

	0	1	2	3	4
0		1			
1	1		1	1	
2		1			
3		1			1
4				1	



Input:

```
5 4     // number of nodes = 5, number of edges = 4
0 1
1 2     // graph[3][4] = 1; graph[4][3] = 1;
1 3
3 4
```

2. Adjacency List

2. Adjacency List

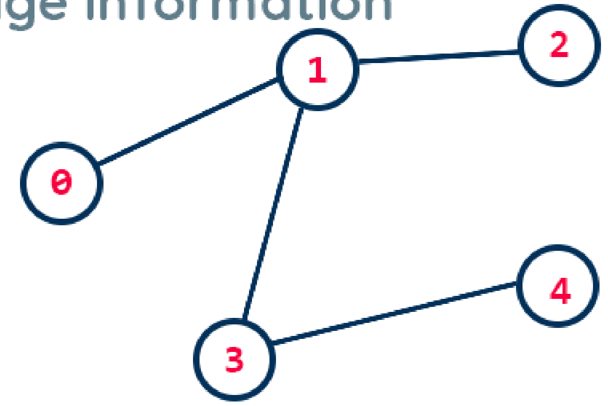
for n nodes, use array of vectors of size n to store edge information

```
vector <int> graph[5];
```

2. Adjacency List

for n nodes, use array of vectors of size n to store edge information

```
vector <int> graph[5];
```

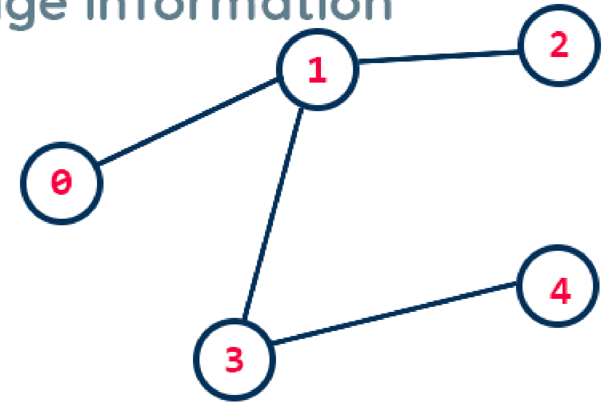


Input:

```
5 4    // number of nodes = 5, number of edges = 4
0 1
1 2
1 3
3 4
```

2. Adjacency List

for n nodes, use array of vectors of size n to store edge information



```
vector <int> graph[5];
```

0

1

2

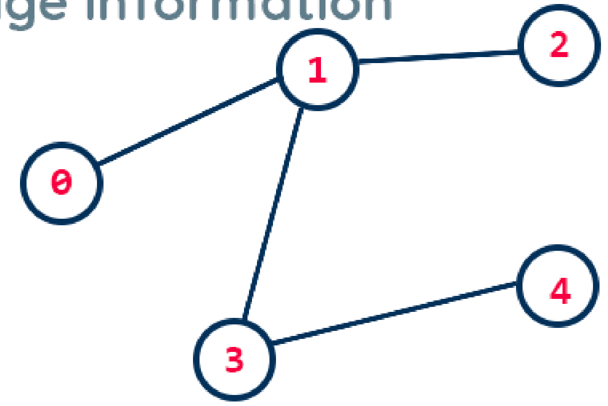
3

4

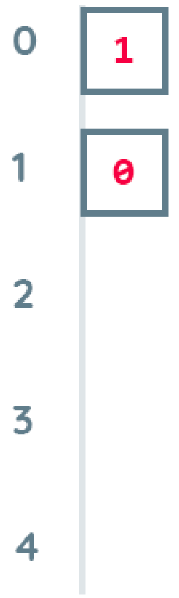
```
Input:
5 4    // number of nodes = 5, number of edges = 4
0 1
1 2
1 3
3 4
```

2. Adjacency List

for n nodes, use array of vectors of size n to store edge information



```
vector <int> graph[5];
```

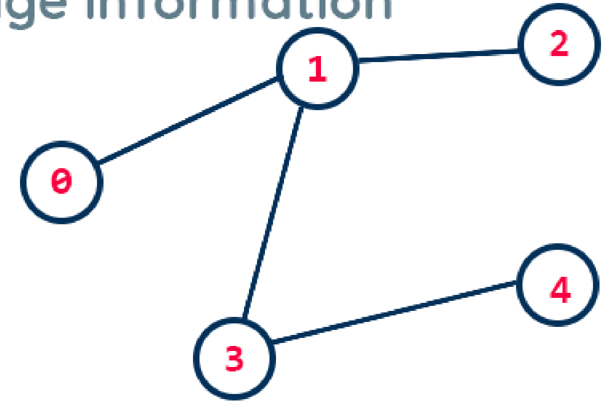


Input:

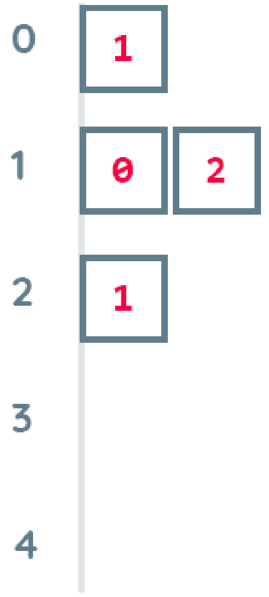
```
5 4    // number of nodes = 5, number of edges = 4
0 1
1 2    // graph[0].push_back(1); graph[1].push_back(0);
1 3
3 4
```

2. Adjacency List

for n nodes, use array of vectors of size n to store edge information



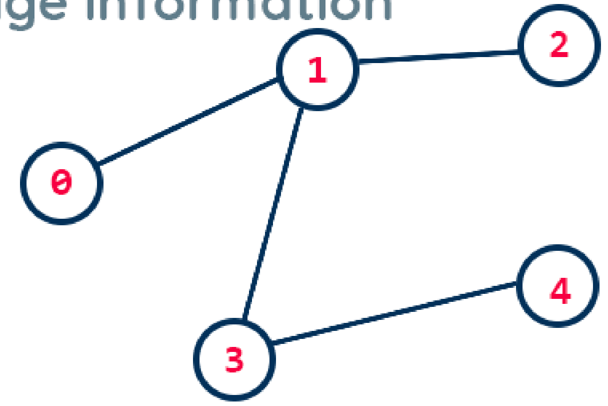
```
vector <int> graph[5];
```



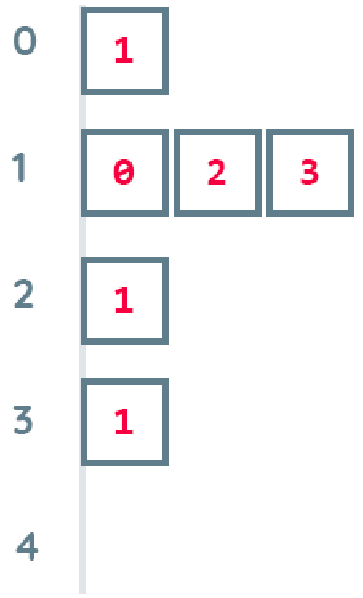
```
Input:
5 4     // number of nodes = 5, number of edges = 4
0 1
1 2     // graph[1].push_back(2); graph[2].push_back(1);
1 3
3 4
```

2. Adjacency List

for n nodes, use array of vectors of size n to store edge information



```
vector <int> graph[5];
```

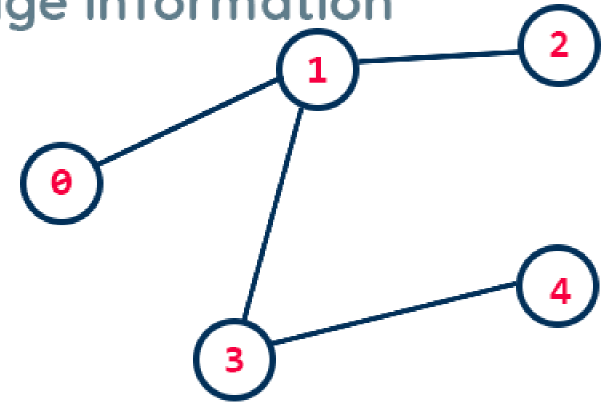


Input:

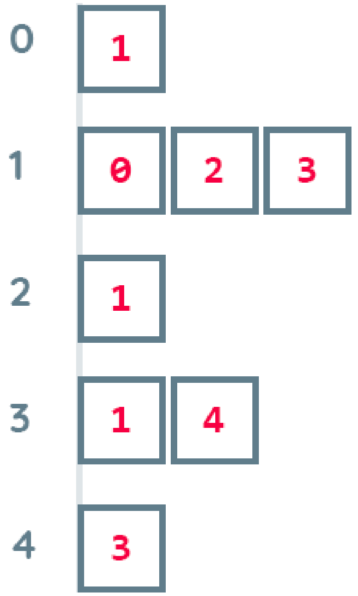
```
5 4     // number of nodes = 5, number of edges = 4
0 1
1 2     // graph[1].push_back(3); graph[3].push_back(1);
1 3
3 4
```

2. Adjacency List

for n nodes, use array of vectors of size n to store edge information



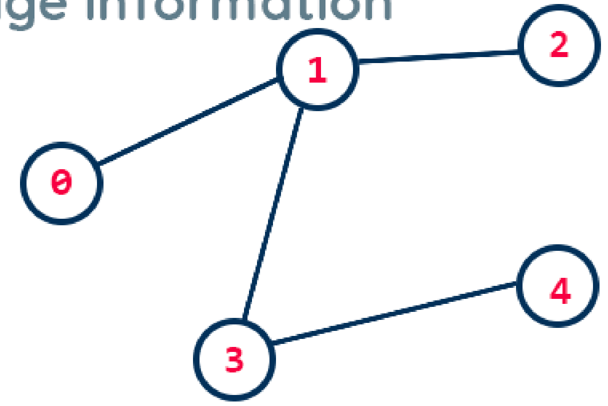
```
vector <int> graph[5];
```



```
Input:

5 4     // number of nodes = 5, number of edges = 4
0 1
1 2     // graph[3].push_back(4); graph[4].push_back(3);
1 3
3 4
```

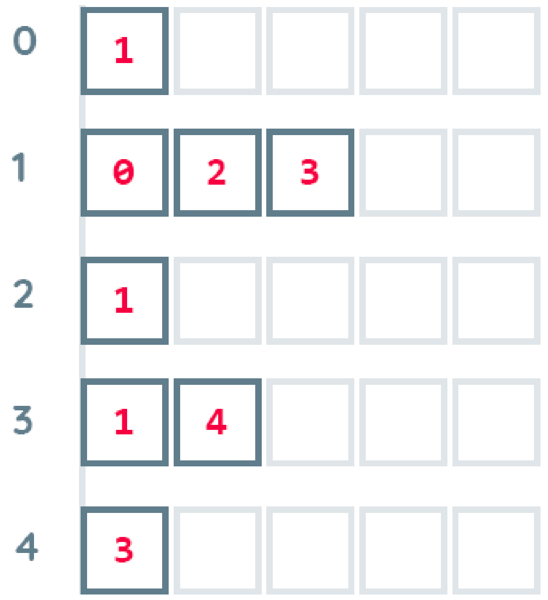
2. Adjacency List

for n nodes, use array of vectors of size n to store edge information



```
vector <int> graph[5];
```

```
// savings
```



```
Input:

5 4     // number of nodes = 5, number of edges = 4
0 1
1 2
1 3
3 4
```