

```

void Graph::BFS(int s) {
    bool* visited = new bool[V];
    for (int i = 0; i < V; ++i)
        visited[i] = false;

    queue<int> Q;
    vector<int>::iterator it;

    Q.push(s);
    visited[s] = true;

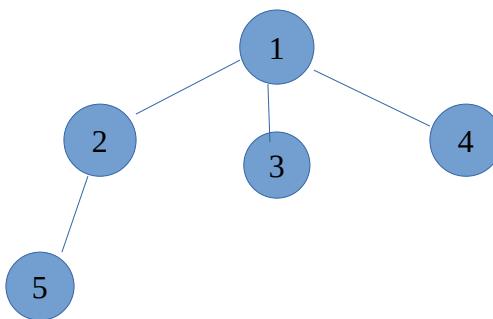
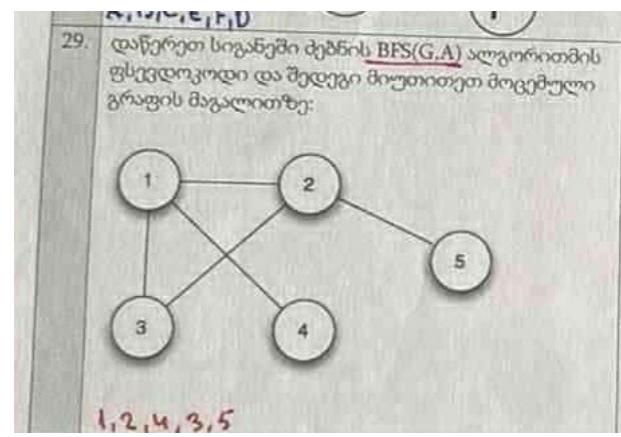
    while (!Q.empty()) {
        s = Q.front();
        Q.pop();

        for (it = adj[s].begin(); it != adj[s].end(); ++it) {
            if (!visited[*it]) {
                Q.push(*it);
                visited[*it] = true;
                cout << s << "->" << *it << endl;
            }
        }
    }
}

```

visit	1	2	3	4	5
	1	1	1	1	1

Queue	1	2	3	4	5
	/	/	/	/	/



30.

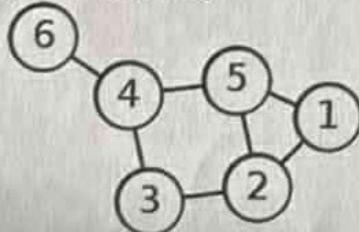
visit

1	2	3	4	5	6
1	1	1	1	1	1

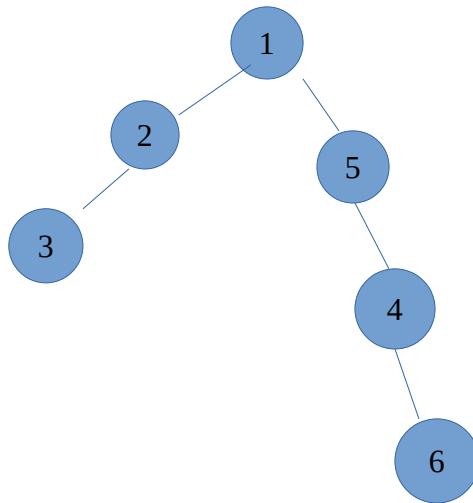
Queue

1	2	5	3	4	6
---	---	---	---	---	---

30. დაწერეთ სიგანეში მებნის BFS(G,A) ალგორითმის ფსევდოკოდი და შედეგი მიუთხოვეთ მოცემული გრაფის მაგალითზე:



1, 2, 5, 3, 4, 6



31.

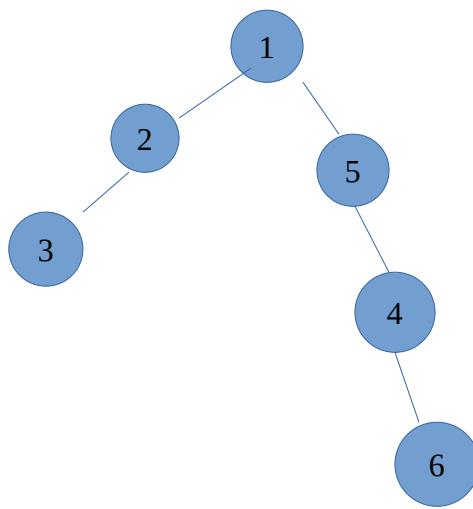
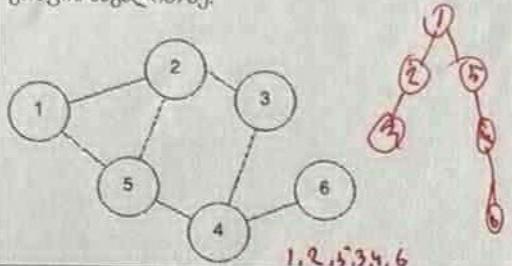
visit

1	2	3	4	5	6
1	1	1	1	1	1

Queue

1	2	5	3	4	6
---	---	---	---	---	---

31. დაწერეთ სიგანები ქებნის BFS(G,A) ალგორითმის ფსვდოკოდი და შედეგი მიუთითოთ მოცემული გრაფის მაგალითზე:



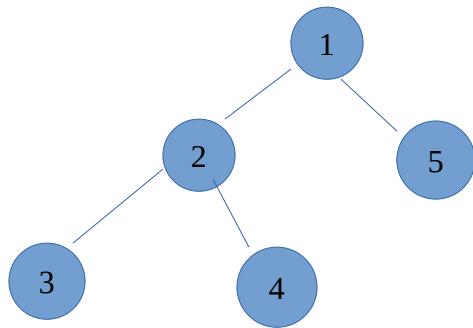
32.

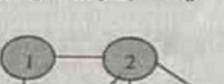
visit

visit	1	2	3	4	5
	1	1	1	1	1

Queue

1	2	5	3	4
---	---	---	---	---



- | | | |
|-----|---|--|
| 32. | <p>დაწერეთ სიგანგმი ქვენის <u>BFS(G,A)</u> ალგორითმის
ფუნქციური და შედეგი მუცითეთ მოცემული
გრაფის მაგალითზე:</p> | |
| |  <p style="text-align: center;">3 2 4 1 5
1, 2, 5, 3, 4</p> <p style="text-align: right;"> 1
 2
 3 </p> <p style="text-align: center;">1, 2, 5, 3, 4</p> | |
| 33. | <p>აღწერეთ ბინარული ძებნის ხემი მონაცემის ჩამის
ალგორითმი და დაწერეთ <u>INSERT(node*root, int</u></p> | |