

Articulation Point in C++

```
#include <bits/stdc++.h>
using namespace std;

//User function Template for C++

class Solution {
private:
    int timer = 1;
    void dfs(int node, int parent, vector<int> &vis, int tin[], int low[], vector<int> &mark, vector<int> adj[]) {
        vis[node] = 1;
        tin[node] = low[node] = timer;
        timer++;
        int child = 0;
        for (auto it : adj[node]) {
            if (it == parent) continue;
            if (!vis[it]) {
                dfs(it, node, vis, tin, low, mark, adj);
                low[node] = min(low[node], low[it]);
                if (low[it] >= tin[node] && parent != -1) {
                    mark[node] = 1;
                }
                child++;
            }
            else {
                low[node] = min(low[node], tin[it]);
            }
        }
        if (child > 1 && parent == -1) {
            mark[node] = 1;
        }
    }
public:
    vector<int> articulationPoints(int n, vector<int> adj[])
    {
        vector<int> vis(n, 0);
        int tin[n];
        int low[n];
        vector<int> mark(n, 0);
        for (int i = 0; i < n; i++) {
            if (!vis[i]) {
                dfs(i, -1, vis, tin, low, mark, adj);
            }
        }
        vector<int> ans;
        for (int i = 0; i < n; i++) {
            if (mark[i] == 1) {
                ans.push_back(i);
            }
        }
        if (ans.size() == 0) return { -1};
        return ans;
    }
};

int main() {

    int n = 5;
    vector<vector<int>> edges = {
        {0, 1}, {1, 4},
        {2, 4}, {2, 3}, {3, 4}
```

Graph Overview

Given edges:

```
0 - 1
  |
  4
 / \
2 - 3
```

Adjacency List:

Node	Neighbors
0	1
1	0, 4
2	4, 3
3	2, 4
4	1, 2, 3

Q Variables Recap

- `tin[node]`: Time of first visit
- `low[node]`: Lowest reachable discovery time
- A node is an **articulation point** if:
 - Not root and `low[child] >= tin[node]`
 - Root and has ≥ 2 children

DFS Trace Table

Step	Node	Parent	tin	low	Action & Reasoning
1	0	-1	1	1	Start DFS from 0
2	1	0	2	2	Visit from 0
3	4	1	3	3	Visit from 1
4	2	4	4	4	Visit from 4
5	3	2	5	5	Visit from 2
6	4	3	-	3	Back edge to 4
7	2	4	-	3	<code>low[2] = min(4, 3)</code>
8	4	1	-	3	<code>low[4] = min(3, 3)</code>
9	1	0	-	2	<code>low[1] = min(2, 3)</code>
10	0	-1	-	1	Done

Articulation Point Analysis

We now check for articulation conditions.

- **Node 1:**

<pre> }; vector<int> adj[n]; for (auto it : edges) { int u = it[0], v = it[1]; adj[u].push_back(v); adj[v].push_back(u); } Solution obj; vector<int> nodes = obj.articulationPoints(n, adj); for (auto node : nodes) { cout << node << " "; } cout << endl; return 0; } </pre>	<ul style="list-style-type: none"> ○ $\text{low}[4] = 3 \geq \text{tin}[1] = 2 \rightarrow \checkmark$ articulation point • Node 4: <ul style="list-style-type: none"> ○ $\text{low}[2] = 3 \geq \text{tin}[4] = 3$ ○ $\text{low}[3] = 5 \geq \text{tin}[4] = 3 \rightarrow \checkmark$ articulation point • Node 0: <ul style="list-style-type: none"> ○ Root with only 1 child $\rightarrow \text{X}$ not articulation point <p>✓ Final Result</p> <p>Articulation Points: 1 4</p>
<p>Output:- 1 4</p>	