Why is the time complexity of both DFS and BFS O(V + E)

The basic algorithm for BFS:

```
set start vertex to visited
load it into queue
    while queue not empty
        for each edge incident to vertex
            if its not visited
                load into queue
                mark vertex
```

So I would think the time complexity would be:

```
v1 + (incident edges) + v2 + (incident edges) + .... + vn + (incident edges)
```

where `v` is vertex `1` to `n`

Firstly, is what I've said correct? Secondly, how is this `O(N + E)` , and intuition as to why would be really nice. Thanks

algorithm graph time-complexity

edited Apr 30 at 18:45
Budhijeevi
13 5

asked Jul 13 '12 at 10:24
ordinary
1,179 3 12 38

5 Answers

Your sum

```
v1 + (incident edges) + v2 + (incident edges) + .... + vn + (incident edges)
```

can be rewritten as

```
(v1 + v2 + ... + vn) + [(incident_edges v1) + (incident_edges v2) + ... + (incident_edges
vn)]
```

and the first group is `O(N)` while the other is O(E).

answered Jul 13 '12 at 10:29
Mihai Maruseac
12.2k 5 28 92

superb explanation, probably best one i have seen – JavaDeveloper Jun 3 '14 at 19:56

DFS(analysis):

- Setting/getting a vertex/edge label takes `O(1)` time
- Each vertex is labeled twice
  - once as UNEXPLORED
  - once as VISITED
- Each edge is labeled twice
  - once as UNEXPLORED
  - once as DISCOVERY or BACK
- Method incidentEdges is called once for each vertex
- DFS runs in `O(n + m)` time provided the graph is represented by the adjacency list structure
- Recall that `Σv deg(v) = 2m`

BFS(analysis):

- Setting/getting a vertex/edge label takes O(1) time
- Each vertex is labeled twice
  - once as UNEXPLORED
  - once as VISITED
- Each edge is labeled twice
  - once as UNEXPLORED
  - once as DISCOVERY or CROSS
- Each vertex is inserted once into a sequence `Li`
- Method incidentEdges is called once for each vertex
- BFS runs in `O(n + m)` time provided the graph is represented by the adjacency list structure
- Recall that `Σv deg(v) = 2m`

edited Jul 13 '12 at 10:33

Mihai Maruseac
12.2k 5 28 92

answered Jul 13 '12 at 10:28

TheNewOne
690 4 14

tnx for the edit i'm new here so i still try to manage with the edit screen :) – TheNewOne Jul 13 '12 at 10:36

thanks for being specific by mentioning that the graphs are to be represented by the adjacency list structure, it was bugging me why DFS is O(n+m), I would think it was O(n + 2m) because each edge is traversed twice by backtracking. – mib1413456 Dec 2 '14 at 12:18

Very simplified without much formality: every edge is considered exactly twice, and every node is processed exactly once, so the complexity has to be a constant multiple of the number of edges as well as the number of vertices.

edited Jul 18 at 9:42

answered Dec 17 '14 at 6:04

JavaFreak
69 6

I think every edge has been considered twice and every node has been visited once, so the total time complexity should be O(2E+V).

answered Jul 21 at 8:23

Kehe CAI
41 3

Even I feel the same. Can anyone give further explanation on this ? – Chaitanya Aug 4 at 6:47

Time complexity is `O(E+V)` instead of `O(2E+V)` because if the time complexity is n^2+2n+7 then it is written as O(n^2).

Hence, O(2E+V) is written as O(E+V)

because difference between n^2 and n matters but not between n and 2n.

edited Sep 12 at 11:30

Am_I_Helpful
8,481 4 14 40

answered Sep 10 at 15:39

user2466859
6 5

@Am_I_Helpful somebody is asking above for 2E in big-oh notation....that why 2 is not considered in time complexity. – user2466859 Sep 10 at 17:38

@Am_I_Helpful just see the post above my answer....there the user named Kehe CAI has written "I think every edge has been considered twice and every node has been visited once, so the total time complexity should be O(2E+V)." So i answered acordingly....Got it !!! – user2466859 Sep 12 at 9:59

I removed my downvote *only* because you edited your answer, – Am_I_Helpful Sep 12 at 11:31