Complexities of Graph Operations

Let's discuss the performance of the two graph representation approaches.

We'll cover the following

Time Complexities

Below, you can find the time complexities for the 4 basic graph methods.

Note that in this table, **V** means the total number of vertices, and **E** means the total number of edges in the Graph.

Operation	Adjacency List	Adjacency Matrix
Add Vertex	O(1)	$O(V^2)$
Remove Vertex	O(V+E)	$O(V^2)$
Add Edge	O(1)	O(1)
Remove Edge	O(E)	O(1)

Adjacency List

- The addition of edge in adjacency lists take constant time, as we only need to insert at the **tail** in the doubly-linked list with a tail pointer of the corresponding vertex.
- The addition of a vertex is not constant in the worst case. The adjacency list is an array of linked lists. So, we would need to allocate a bigger array, copy the contents from the previous array, which takes O(n). But this operation can be considered O(1) on average because we could do it smartly as follows. We start with a small array initially. Then, when there is a need to insert a vertex, instead of allocating one size bigger array, we allocate an array that is twice as big. That way, in the long run, the cost of re-allocation is averaged out over many other operations. So, O(1) is the average or amortized cost of add vertex in the adjacency list.
- Removing an edge takes $O(E)$ time because–in the worst case–all the edges could be at a single vertex, and hence, we would have to traverse all **E** edges to reach the last one.
- Removing a vertex takes O(V + E) time because we have to delete all its edges, and then reindex the rest of the list one step back in order to fill the deleted spot.

Adjacency Matrix #

- Edge operations are performed in constant time, as we only need to manipulate the value in the particular cell.
- Vertex operations are performed in $O(V^2)$ since we need to add rows and columns. We will also need to fill all the new cells.

Comparison

Both representations are suitable for different situations. If your model frequently manipulates vertices, the adjacency list is a better choice.

If you are dealing primarily with edges, the adjacency matrix is the more efficient approach.

Keep these complexities in mind because they will give you a better idea about the time complexities of the several algorithms we'll see in this section.

Let’s move towards a different type of graph called a Bipartite Graph in the upcoming lesson.

Back

Graph Implementation

Next

What is a Bipartite Graph?

Mark as Completed

Report an Issue

Ask a Question (https://discuss.educative.io/tag/complexities-of-graph-operations__graphs__data-structures-for-coding-interviews-in-java)