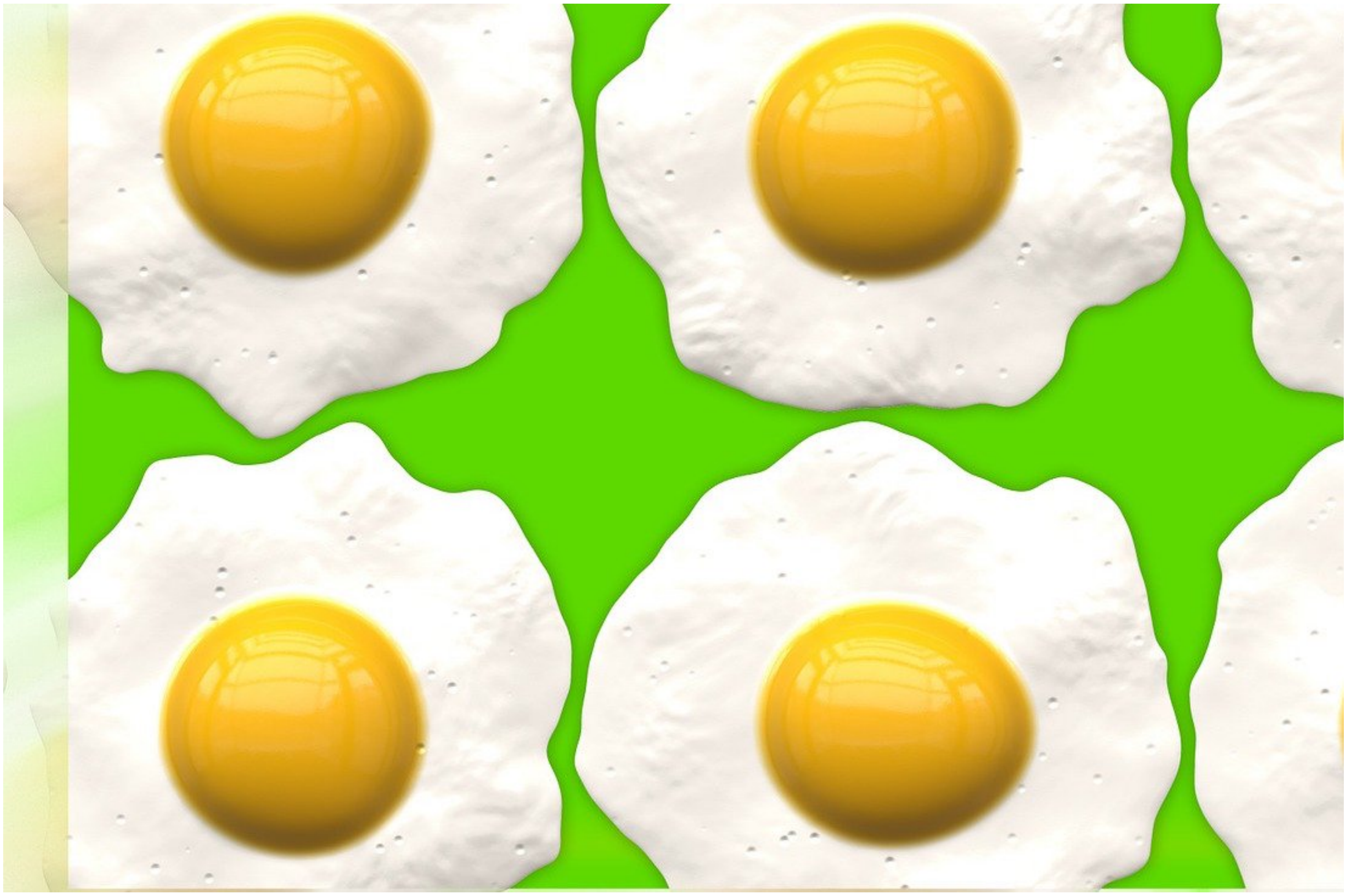
LearnTeach

- My Profile

- [View]
- [Edit]
- Logout



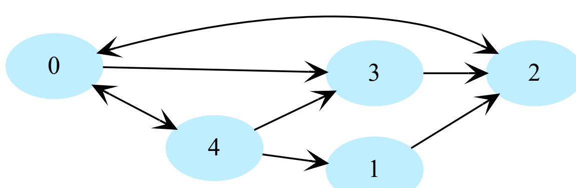
Clone a Directed Graph

Given root node of a directed graph, clone this graph by creating its deep copy such that cloned graph has same vertices and edges as original graph.

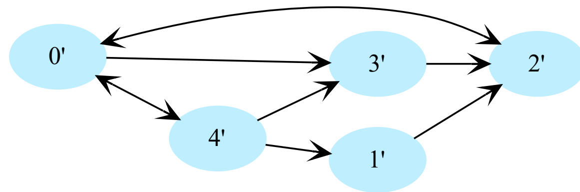
Description

Given root node of a directed graph, clone this graph by creating its deep copy such that cloned graph has same vertices and edges as the original graph.

Let's look at below graphs as an example. If input graph is G = (V, E) where V is set of vertices and E is set of edges, then output graph (cloned graph) G' = (V', E') such that V = V' and E = E'.



Original Graph



Cloned Graph

Hints

- Use hash table
- Depth first traversal

Solution

Runtime Complexity

Linear, O(n).

Memory Complexity

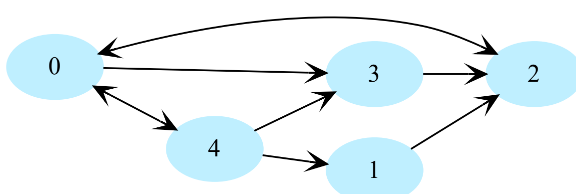
Logarithmic, O(n). 'n' is number of vertices in graph.

We can have most n entries in hash table, so worst case space complexity is O(n).

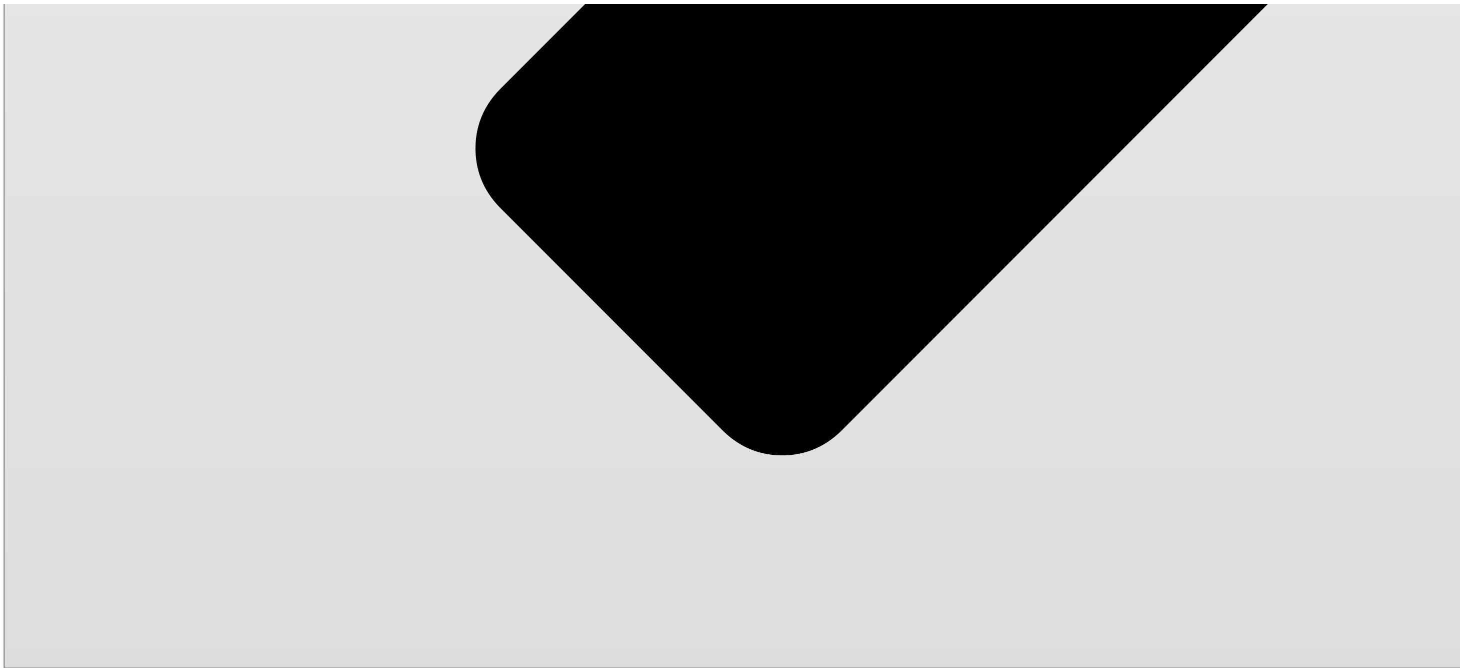
We use depth first traversal and create a copy of each node while traversing the graph. To avoid getting stuck in cycles, we'll use a hashtable to store each completed node and will not revisit nodes that exist in the hashtable. Hashtable key will be a node in the original graph, and its value will be the corresponding node in cloned graph.

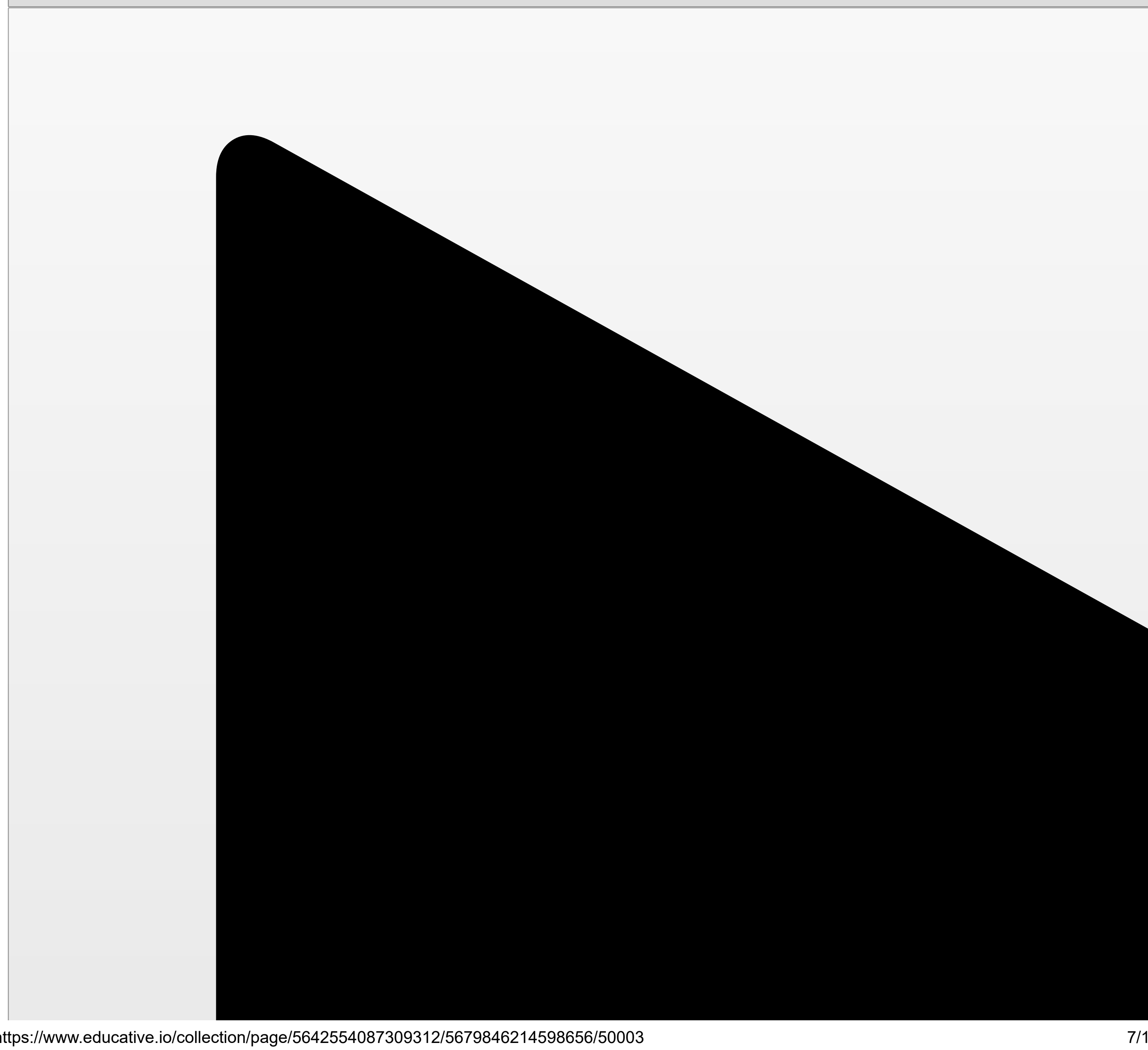
For above graph lets assume root is node '0'. We'll start with the root node i.e. '0'.

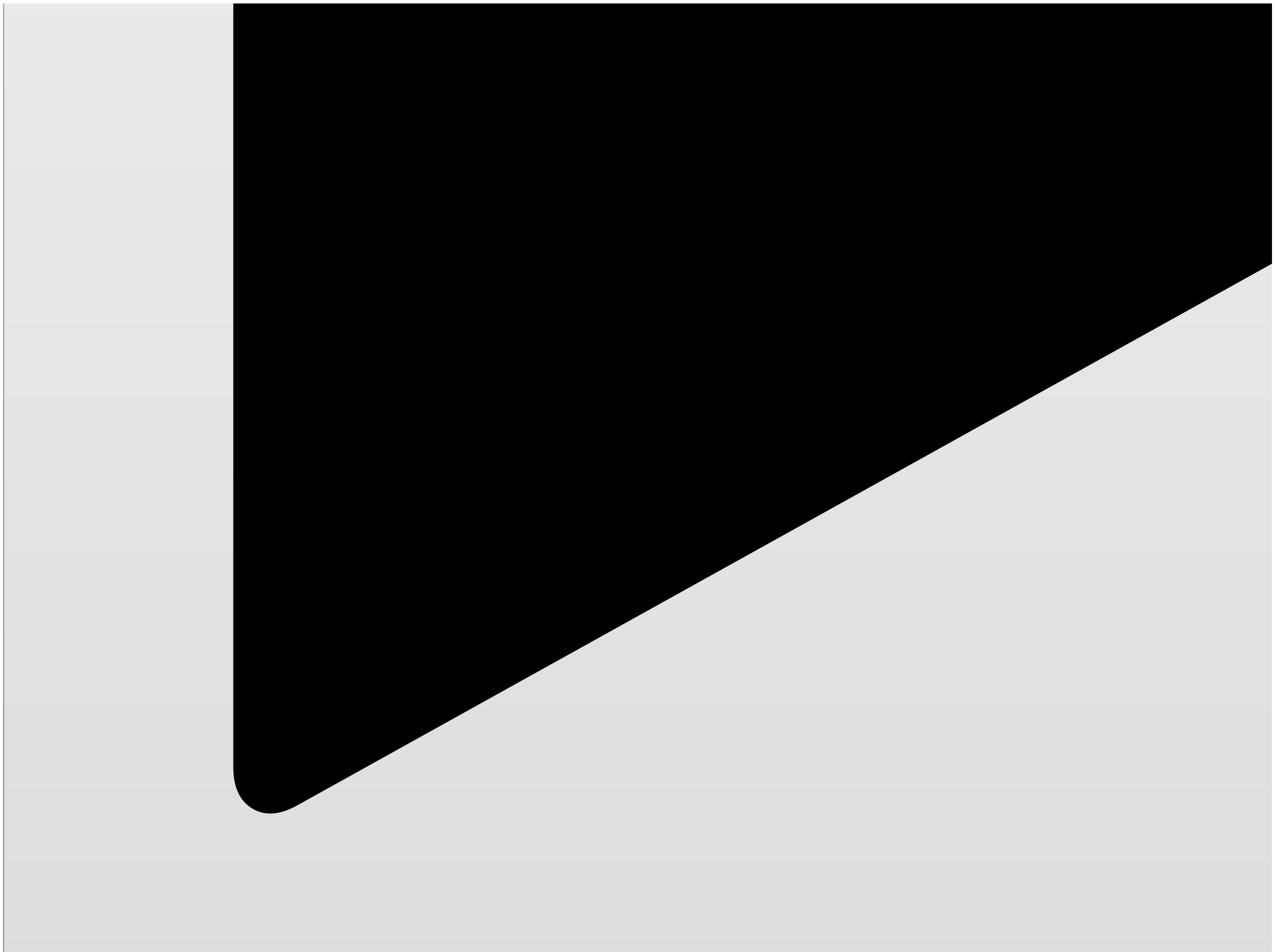
Suppose the original graph looks like this.We will now clone the exact same graph by performing a depth first traversal.

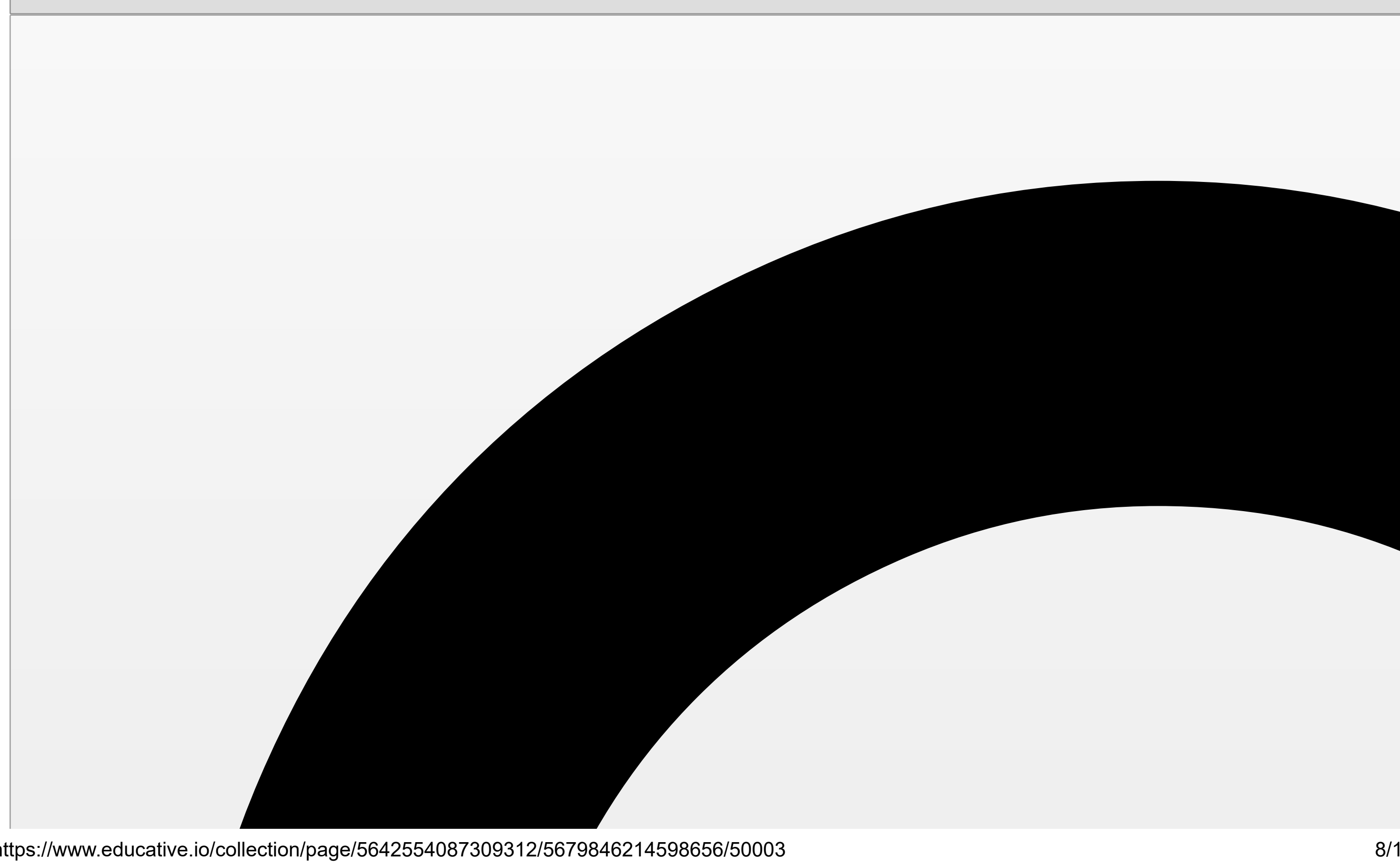


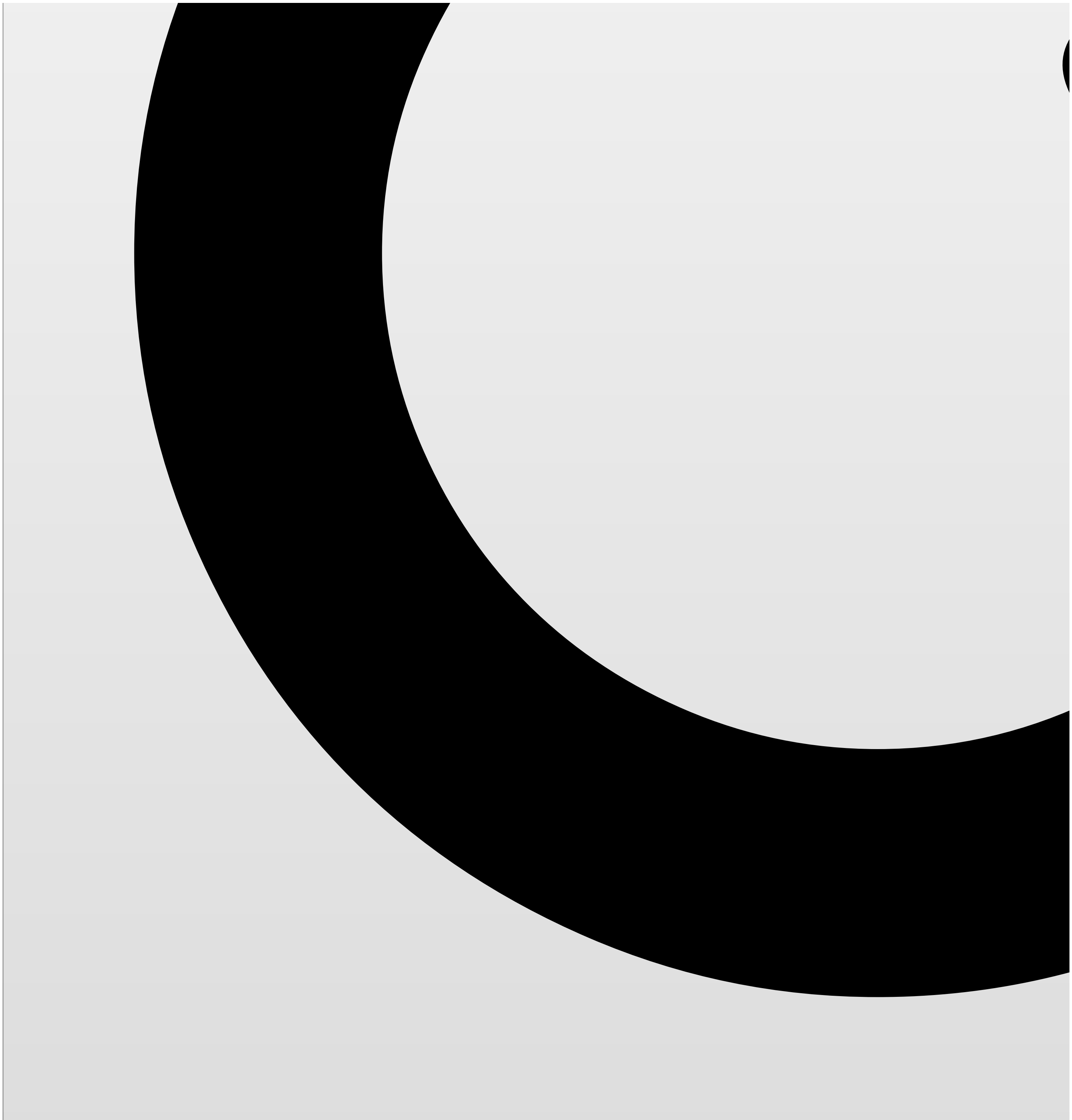
1 of 10

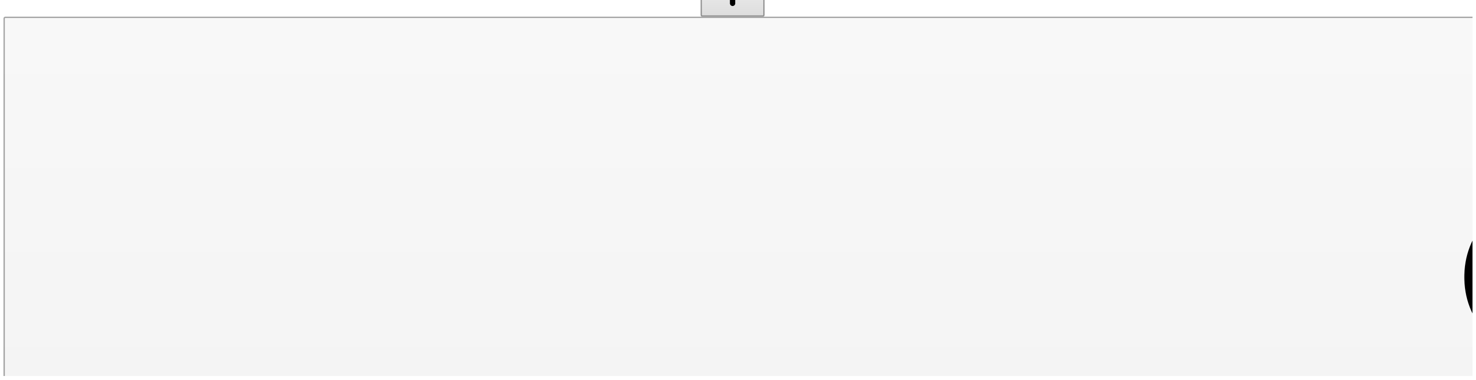


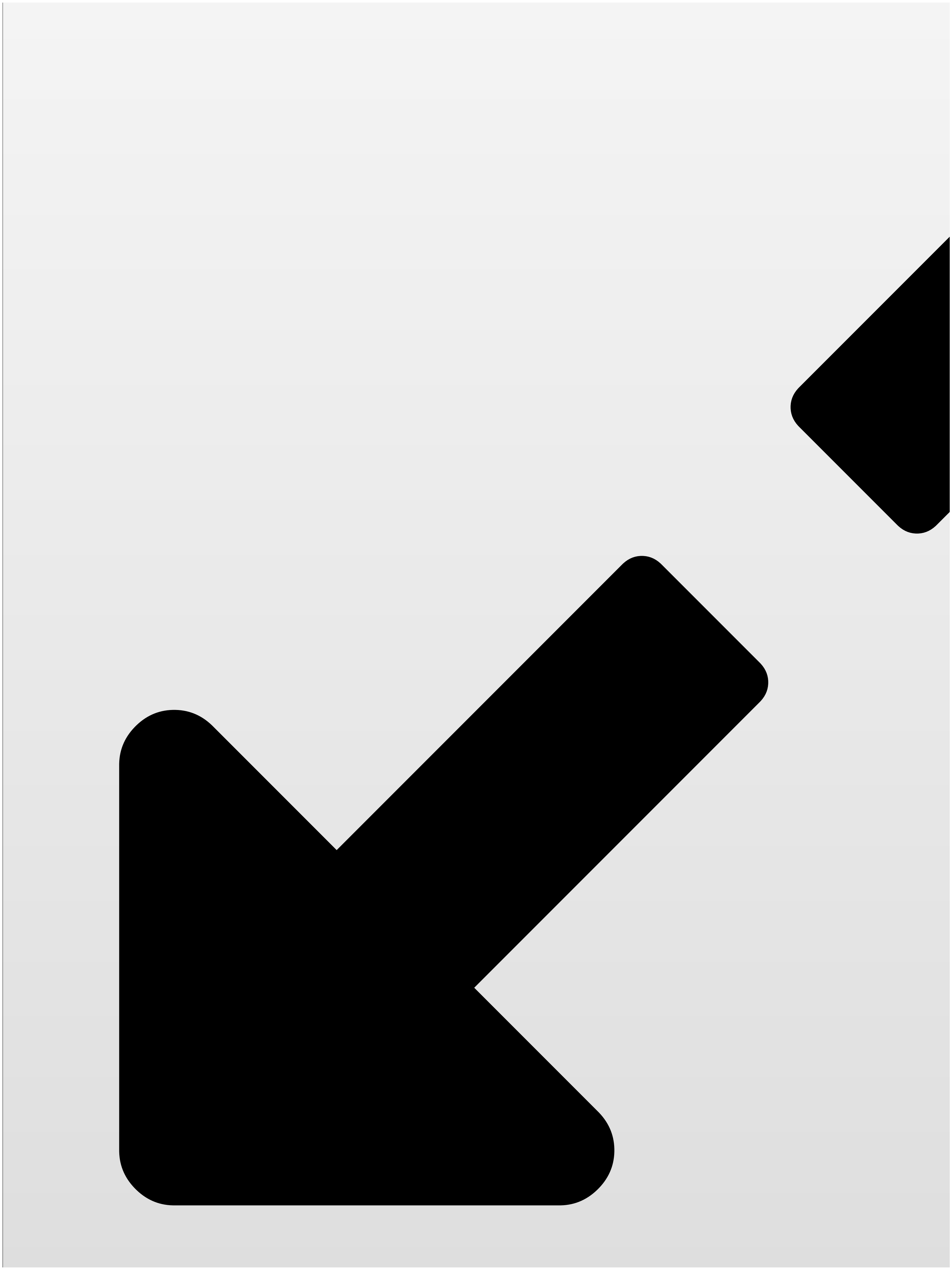












- C++
- Java
- Python
- JS
- Ruby

```
struct Node {
  int data;
  list<Node*> neighbors;
  Node(int d) : data(d) {}
};

Node* clone_rec(Node* root,
        unordered_map<Node*,
        Node*>& nodes_completed) {

  if (root == nullptr) {
    return nullptr;
  }

  Node* pNew = new Node(root->data);
  nodes_completed[root] = pNew;
```

```
  for (Node* p : root->neighbors) {

    auto x = nodes_completed.find(p);

    if (x == nodes_completed.end()){
      pNew->neighbors.push_back(clone_rec(p, nodes_completed));
    } else {
      pNew->neighbors.push_back(x->second /*value*/);
    }
  }

  return pNew;
}

Node* clone(Node* root) {
  unordered_map<Node*, Node*> nodes_completed;
  return clone_rec(root, nodes_completed);
}
```

Cover Image courtesy of [Queensland Museum]

[Mark as completed]
[← *Previous*Expression Evaluation*Next* →Minimum Spanning Tree]

Send feedback or ask a question

- Home
- Featured
- Team
- Blog
- FAQ
- Terms of Service
- Contact Us