|  |  |
| --- | --- |
| **Height in C++** | |
| #include <iostream>  #include <vector>  #include <stack>  using namespace std;  // Node class definition  class Node {  public:  int data;  vector<Node\*> children;  Node(int val) {  data = val;  }  };  // Function to construct the tree from the given array  Node\* construct(vector<int>& arr) {  Node\* root = nullptr;  stack<Node\*> st;  for (int i = 0; i < arr.size(); ++i) {  if (arr[i] == -1) {  st.pop();  } else {  Node\* t = new Node(arr[i]);  if (!st.empty()) {  st.top()->children.push\_back(t);  } else {  root = t;  }  st.push(t);  }  }  return root;  }  // Function to calculate the height of the tree  int height(Node\* node) {  if (node->children.empty()) {  return 0;  }  int maxChildHeight = 0;  for (Node\* child : node->children) {  int childHeight = height(child);  if (childHeight > maxChildHeight) {  maxChildHeight = childHeight;  }  }  return maxChildHeight + 1;  }  // Main function  int main() {  vector<int> arr = {10, 20, -1, 30, 50, -1, 60, -1, -1, 40, -1, -1};  Node\* root = construct(arr);  int h = height(root);  cout << h << endl;  return 0;  } | **Input Array:**  {10, 20, -1, 30, 50, -1, 60, -1, -1, 40, -1, -1}  **🛠️ Tree Construction (construct function):**  We use a **stack** to maintain the current path in the tree. When we encounter -1, we pop a node from the stack (finished with that node's children). Here's a **step-by-step construction** of the tree:   | **Step** | **arr[i]** | **Stack Top** | **Action** | **Tree Change** | | --- | --- | --- | --- | --- | | 0 | 10 | — | Create node(10), push | root = 10 | | 1 | 20 | 10 | Add 20 as child to 10, push | 10 → 20 | | 2 | -1 | 20 | Pop 20 |  | | 3 | 30 | 10 | Add 30 as child to 10, push | 10 → 30 | | 4 | 50 | 30 | Add 50 as child to 30, push | 30 → 50 | | 5 | -1 | 50 | Pop 50 |  | | 6 | 60 | 30 | Add 60 as child to 30, push | 30 → 60 | | 7 | -1 | 60 | Pop 60 |  | | 8 | -1 | 30 | Pop 30 |  | | 9 | 40 | 10 | Add 40 as child to 10, push | 10 → 40 | | 10 | -1 | 40 | Pop 40 |  | | 11 | -1 | 10 | Pop 10 (tree complete) |  |   ✅ Final Tree:  10  ├── 20  ├── 30  │ ├── 50  │ └── 60  └── 40  **🌲 Height Calculation:**  The **height** of a tree is the number of edges in the longest path from the root to a leaf node.  We traverse each subtree and compute the max height:   * Leaf nodes like 20, 50, 60, and 40 → height = 0 * Node 30 has children 50 and 60 → height = 1 * Root 10 has children:   + 20 → 0   + 30 → 1   + 40 → 0 → max child height = 1 → root height = 1 + 1 = 2   ✅ **Final Height: 2**  **✅ Output:**  2 |
| 2 | |