

**UTS LAB DESAIN & ANALISIS ALGORITMA LAB 3**  
**“Implementasi Algoritma”**



**Disusun Oleh :**

1. Putri Andriyani (211401008)
2. Harry Sion Tarigan (211401021)
3. Angela Siadari (211401030)
4. Lorenzo Liunardo (211401061)

**Asisten Lab : Wilson**

**UNIVERSITAS SUMATERA UTARA**  
**MEDAN 2022/2023**

## **PROJECT:**

Lakukan implementasi algoritma untuk menyelesaikan satu masalah minimal dua algoritma.

## **HASIL:**

### **A. Implementasi Algoritma**

Menemukan jumlah vertikal dari sebuah binary tree yang diberikan dengan menggunakan metode hashing dan metode doubly linked list.

### **B. Komparasi Antara Kedua Algoritma**

#### **1. Dengan Metode Hashing**

Pada metode hashing ini kita akan membuat sebuah hash table/map kosong yang mana setiap key-nya mewakili horizontal distance sebuah node dari root node-nya, dan value dari hash table/map akan menyimpan jumlah dari semua node pada horizontal distance yang sama. Di sini, horizontal distance dari root adalah 0 dan untuk subtree di sisi kiri akan memiliki horizontal distance -1 dari horizontal root, sedangkan untuk subtree sisi kanan akan memiliki horizontal distance +1 dari horizontal root. Kemudian, kita menggunakan metode post-order traversal untuk melintasi semua node dan mengupdate jumlah dari masing-masing horizontal distance pada hash table/map. Untuk setiap node, kita mengrekursif untuk menghitung horizontal distance di mana untuk subtree sisi kiri kita mempassing horizontal root -1 dan untuk subtree sisi kanan kita mempassing horizontal root +1.

#### **2. Dengan Metode Doubly Linked List**

Pada metode ini, kita akan menyimpan jumlah vertikal dari binary tree pada sebuah doubly linked list, di mana setiap node pada doubly linked list akan menyimpan jumlah semua node dari binary tree pada garis vertikal yang sama. Algoritma ini dimulai dengan kita membuat sebuah node doubly linked list yang akan menyimpan jumlah node pada vertical line yang melalui root node. Lalu, node->prev dan node->next akan menyimpan jumlah node yang ada pada vertical line masing-masing melewati subtree dari root node sisi kanan dan kiri. Idennya adalah kita akan membuat linked list secara rekursif dan mengupdate node dengan jumlah vertikal saat kita melintasi tree tersebut.

### C. Kesimpulan

Algorithm	Time Complexity	Space Complexity	Notes
Hashing	$O(n \log(n))$	$O(n)$	<ul style="list-style-type: none"><li>• <math>n</math> = ukuran dari binary tree-nya.</li></ul>
Doubly Linked List	$O(n)$	$O(h+v)$	<ul style="list-style-type: none"><li>• <math>h</math> = tinggi dari binary tree-nya.</li><li>• <math>v</math> = banyaknya vertical lines.</li></ul>

Kesimpulannya adalah dengan menggunakan metode doubly linked list kita dapat membuat programnya lebih efisien secara waktu dan ruangnya. Di mana pada doubly linked list mempunyai waktu akses dengan notasi  $O(n)$  sedangkan ketika menggunakan hashing membutuhkan waktu  $O(n \log(n))$ . Kemudian dengan menggunakan metode doubly linked list juga lebih efisien ruang karena hanya membutuhkan  $n$  nodes doubly linked list di mana  $n$  adalah banyak vertical lines pada binary tree yang kita miliki sedangkan pada hashing membutuhkan ruang yang lebih besar daripada jumlah vertical lines yang kita punya.

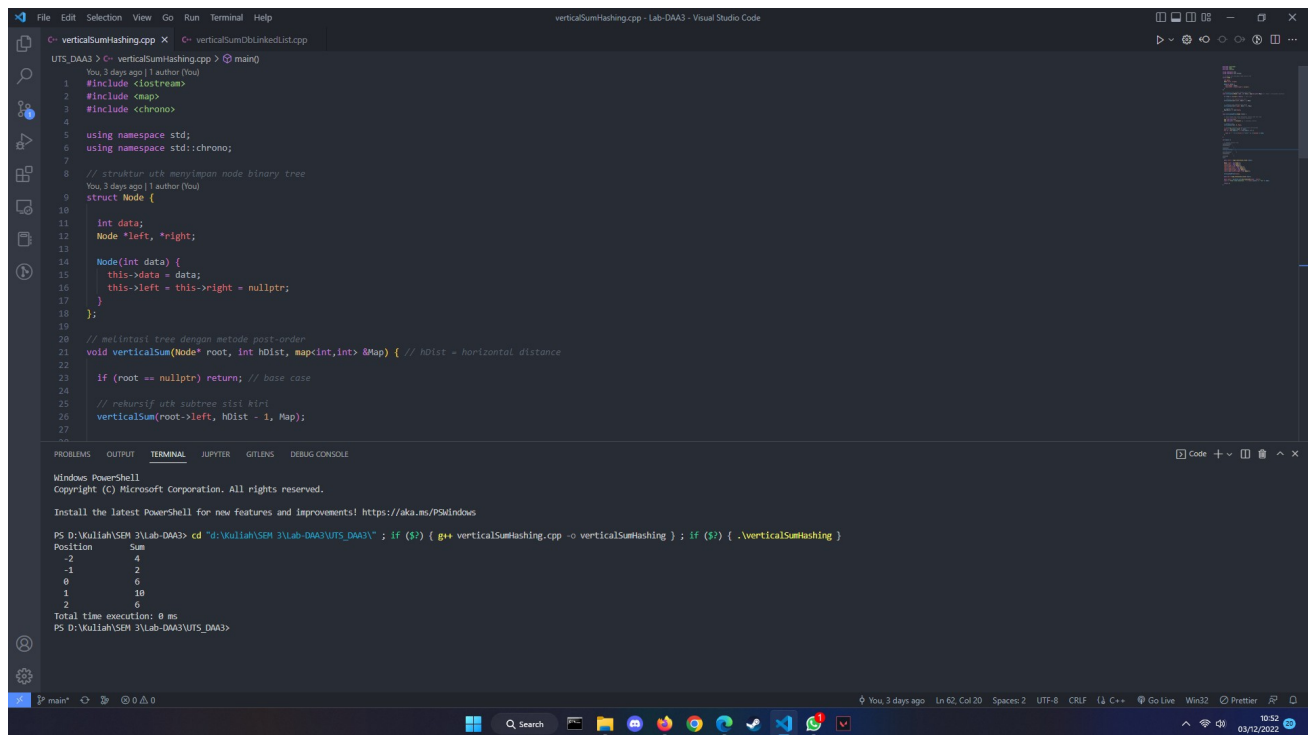
### D. Link Source Code Github

<https://github.com/hawryyy30/lab3daa-uts>

### E. Bukti Source Code berhasil di-run

## Metode

## Hashing



```
File Edit Selection View Go Run Terminal Help
verticalSumHashing.cpp X verticalSumDbLinkedList.cpp

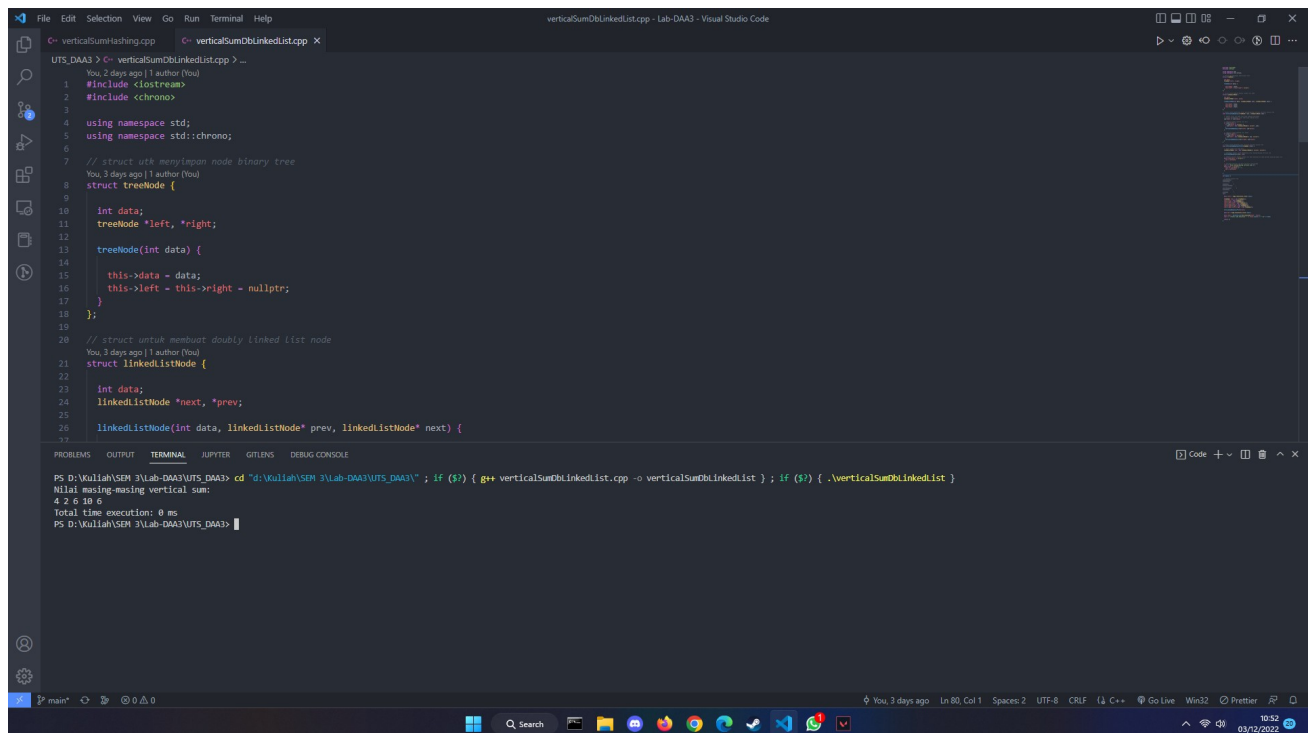
UTS_DAA3 > C- verticalSumHashing.cpp > main()
You, 3 days ago | author (You)
1 #include <iostream>
2 #include <map>
3 #include <chrono>
4
5 using namespace std;
6 using namespace std::chrono;
7
8 // struktur utk menyimpan node binary tree
9 You, 3 days ago | author (You)
10 struct Node {
11
12     int data;
13     Node *left, *right;
14
15     Node(int data) {
16         this->data = data;
17         this->left = this->right = nullptr;
18     }
19 };
20
21 // melintasi tree dengan metode post-order
22 void verticalSum(Node* root, int hDist, map<int,int> &Map) { // hDist = horizontal distance
23
24     if (root == nullptr) return; // base case
25
26     // rekursif utk subtree sisi kiri
27     verticalSum(root->left, hDist - 1, Map);
28
29     // melintasi tree dengan metode post-order
30     Map[hDist] += root->data;
31
32     // rekursif utk subtree sisi kanan
33     verticalSum(root->right, hDist + 1, Map);
34 }
35
36 int main() {
37     // ...
38 }
39
40 PROBLEMS OUTPUT TERMINAL JUPYTER GITLENS DEBUG CONSOLE
10:52 03/12/2022

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\Kuliah\SEM 3\Lab-DAA3\UTS_DAA3> cd "d:\Kuliah\SEM 3\Lab-DAA3\UTS_DAA3"; if ($?) { g++ verticalSumHashing.cpp -o verticalSumHashing }; if ($?) { .verticalSumHashing }
Position      Sum
-----
-2            4
-1            2
0             6
1            10
2             6
Total time execution: 0 ms
PS D:\Kuliah\SEM 3\Lab-DAA3\UTS_DAA3>
```

## Metode Double Linked List



```
File Edit Selection View Go Run Terminal Help
verticalSumDbLinkedList.cpp X

UTS_DAA3 > C- verticalSumDbLinkedList.cpp > ...
You, 2 days ago | author (You)
1 #include <iostream>
2 #include <chrono>
3
4 using namespace std;
5 using namespace std::chrono;
6
7 // struktur utk menyimpan node binary tree
8 You, 3 days ago | author (You)
9 struct treeNode {
10
11     int data;
12     treeNode *left, *right;
13
14     treeNode(int data) {
15         this->data = data;
16         this->left = this->right = nullptr;
17     }
18 };
19
20 // struktur untuk membuat doubly linked list node
21 You, 3 days ago | author (You)
22 struct linkedListNode {
23
24     int data;
25     linkedListNode *next, *prev;
26
27     linkedListNode(int data, linkedListNode* prev, linkedListNode* next) {
28
29         this->data = data;
30         this->prev = prev;
31         this->next = next;
32     }
33 }
34
35 int main() {
36     // ...
37 }
38
39 PROBLEMS OUTPUT TERMINAL JUPYTER GITLENS DEBUG CONSOLE
10:52 03/12/2022

PS D:\Kuliah\SEM 3\Lab-DAA3\UTS_DAA3> cd "d:\Kuliah\SEM 3\Lab-DAA3\UTS_DAA3"; if ($?) { g++ verticalSumDbLinkedList.cpp -o verticalSumDbLinkedList }; if ($?) { .verticalSumDbLinkedList }
Nilai masing-masing vertical sum:
4 2 6 10 6
Total time execution: 0 ms
PS D:\Kuliah\SEM 3\Lab-DAA3\UTS_DAA3>
```