JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

USULAN TUGAS AKHIR

1. IDENTITAS PENGUSUL

NAMA	**:**	**NUR AHMAD WAHID**
NRP	**:**	**5110100702**
DOSEN WALI	**:**	**Dr. Nanik Suciati, S.Kom., M.Kom.**
DOSEN PEMBIMBING	**:**	**1. Arya Yudhi Wijaya, S.Kom., M.Kom.**
		2. Rully Soelaiman, S.Kom., M.Kom.

2. JUDUL TUGAS AKHIR

"Desain dan Analisis Struktur Data Non Linier Rooted Tree Dinamis"

3. LATAR BELAKANG

Tree merupakan struktur data graf tak berarah yang tidak memiliki satu pun *cycle*. Beda halnya dengan *tree* pada umumnya, *rooted tree* yang merupakan graf berarah memiliki tepat 1 *vertex* sebagai *root* dan semua *edge* yang terhubung pada *tree* diarahkan berlawanan dari *root*. *Rooted tree* dalam prakteknya banyak digunakan di dunia nyata. Contoh sederhana seperti organisasi file-file dalam memori komputer ke dalam sistem direktori dimana *root* direpresentasikan sebagai *root* direktori dengan semua *internal vertex* yang terhubung di dalamnya direpresentasikan sebagai subdirektori. Adapun yang bertindak sebagai *leaves* adalah file-file dan direktori kosong [1].

Banyak permasalahan-permasalahan yang dapat ditemukan dalam pengimplementasian struktur data *rooted tree.* Salah satunya adalah penarikan data yang dimiliki oleh setiap *sub-tree* dengan kondisi struktur *tree* yang dapat berubah-ubah. Dibutuhkan desain struktur data serta algoritma yang efisien baik *space* yang dibutuhkan maupun kecepatan akses pencariannya.

Permasalahan tersebut dapat diabstraksi ke dalam bentuk yang lebih sederhana, yaitu pada setiap *vertex* yang terhubung pada *tree* memiliki bobot masing-masing. bagaimana menarik informasi jumlah bobot dari *vertex-vertex* yang terhubung pada suatu *sub-tree* secara tepat dan efisien dengan *root* yang dapat berubah-ubah.

4. RUMUSAN MASALAH

Beberapa permasalahan yang akan diselesaikan pada tugas akhir ini adalah:

a. Bagaimana merancang dan menganalisis struktur data dan algoritma yang efisien untuk menyelesaikan masalah perhitungan jumlah *weight* dari *vertex-vertex* yang terhubung pada satu *sub-tree* dari *rooted tree* dengan kondisi *root* yang dapat berubah secara dinamis?
b. Bagaimana mengimplementasikan struktur data dan algoritma yang telah terancang untuk menyelesaikan masalah secara efisien?
c. Bagaimana mengukur kompleksitas waktu dan ruang (*space*) algoritma yang diterapkan pada struktur data yang telah dirancang?

5. BATASAN MASALAH

Beberapa batasan masalah yang menjadi batas pada tugas akhir ini adalah:

a. Jenis struktur data yang digunakan adalah *n-ary rooted tree.*
b. Jumlah *vertex* yang terhubung pada *tree* tidak lebih dari 10^5.
c. Bobot yang dimiliki setiap *vertex*-nya dapat mampu ditampung dalam tipe data *integer* 32 bit bertanda.
d. Bahasa pemrograman yang digunakan adalah C++.

6. TUJUAN PEMBUATAN TUGAS AKHIR

Tujuan dari pembuatan tugas akhir ini adalah:

a. Menganalisis dan merancang algoritma yang efisien untuk menyelesaikan masalah penarikan informasi dari struktur data *rooted tree* dinamis.
b. Mengimplementasikan struktur data dan algoritma yang telah terancang untuk menyelesaikan masalah penarikan informasi dari struktur data *rooted tree* dinamis secara efisien.
c. Menganalisis kompleksitas waktu dan ruang (*space*) algoritma yang diterapkan pada struktur data yang telah dirancang.

7. MANFAAT TUGAS AKHIR

Manfaat dari tugas akhir ini adalah untuk menghasilkan kajian yang komprehensif serta implementasi algoritma yang digunakan untuk menyelesaikan masalah penarikan data dalam hal ini jumlah dari setiap *weight* yang dimiliki *node* pada *sub-tree* x yang terhubung pada struktur data *rooted tree* dinamis dengan konsumsi waktu dan *memory* yang efisien.

8. TINJAUAN PUSTAKA

a. *Tree*

Dalam teori graf, *tree* merupakan *connected graph* yang tidak memiliki satu pun *cycle. Tree* sudah digunakan sejak tahun 1857 ketika seorang ilmuwan matematika Arthur Cayley [2] menggunakannya untuk menghitung senyawa kimia dengan jenis tertentu. Dewasa ini, *tree* sudah banyak digunakan di banyak cabang ilmu. Terkhusus untuk cabang ilmu komputer, *tree* banyak digunakan dalam pengimplementasian berbagai macam algoritma. Contohnya, *tree* digunakan untuk meletakkan *item* ke dalam suatu *list.* Pada cabang ilmu teknik kompresi, algoritma *Huffman Coding* menggunakan *tree* sebagai dasarnya untuk membangun *coders* yang efisien pada transmisi data dan penyimpanan.

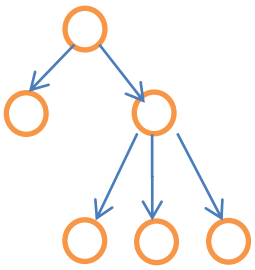
b. *Rooted Tree*

Struktur data *tree* pada dasarnya adalah graf tak berarah serta tak memiliki *vertex* yang spesifik sebagai *root*-nya. Berbeda dengan *rooted tree* yang merupakan graf berarah yang memiliki satu *vertex* sebagai *root* dan setiap *edge* nya secara implisit berarah menjauh dari *root*. Beberapa defenisi yang perlu diketahui:

1. *Depth* atau *level* suatu *vertex v* adalah jaraknya terhadap *vertex root*.
2. *Height* dari *rooted tree* adalah *path* yang terpanjang dari *root*.
3. *Vertex v* disebut sebagai *parent* dari *vertex w* dan *vertex w* sebagai *child* dari *vertex* x jika *vertex v* adalah *vertex* yang pertama kali ditemui ketika menelusuri *path* dari *root* dibanding *vertex w*.
4. *Vertex-vertex* yang memiliki *parent* yang sama disebut *siblings.*
5. *Leaves* adalah *vertex-vertex* yang tidak memiliki *children.*
6. *Internal vertex* adalah *vertex-vertex* yang terhubung pada *rooted tree* yang memiliki sedikitnya 1 *child.*

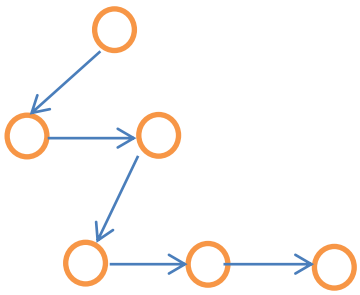
a. Left Child Right Sibling Binary Tree

Pada umumnya untuk merepresentasikan sebuah *tree* digunakan pendekatan seperti pada Gambar 1.



Gambar 1**.** *Rooted tree* tanpa representasi *left child right sibling binary tree*.

Representasi ini mempunyai kekurangan dimana setiap *child* dari *vertex* harus dialokasikan terpisah ke dalam *array* tambahan sehingga setiap penambahan *child* pada suatu *vertex* dibutuhkan pengalokasian memori baru. Ada representasi yang lebih efisien yang biasa disebut *left child right-sibling binary tree* [3]. Ilustrasinya seperti pada Gambar 2.



Gambar 2**.** *Rooted tree* dengan representasi *left child right sibling binary tree*.

Setiap *vertex* pasti maksimal terhubung ke 2 *vertex* lainnya. 1 *vertex* pasti memiliki maksimal 1 *vertex* yang bertindak sebagai *child* nya dan 1 *vertex* yang bertindak sebagai *right-sibling* nya. Dengan kata lain tidak diperlukan alokasi memori tambahan untuk *children* yang dimiliki oleh setiap *vertex* bahkan ketika ada penambahan *child* sekalipun. Beberapa keuntungan yang dapat diperoleh dari representasi *tree* tersebut adalah:

1. Setiap *vertex* pasti memiliki ukuran yang *fixed.* Sehingga tidak diperlukan memori (*array*) tambahan.
2. Banyak algoritma yang dapat diimplementasikan menggunakannya mengingat struktur *tree* tersebut tidak lain berupa bentuk *binary tree.*

9. RINGKASAN TUGAS AKHIR

Desain struktur data yang baik sangat penting demi memenuhi tujuan kemudahan serta kecepatan dalam pengaksesannya. Performansi penarikan data dari suatu struktur data tergantung seberapa baiknya struktur data serta algoritma yang digunakan untuk menariknya. Demikian pula dengan struktur data *tree*. Representasi data terhadap memori nyata sangat berpengaruh. Dengan dibentuknya suatu *n-ary tree* dengan representasi *left child right-sibling binary tree* dapat memudahkan dalam penelusuran struktur data. Ada 2 macam aksi yang dapat dilakukan terhadap struktur data *tree* yang dibangun, yaitu memindahkan *root* dari satu *vertex* ke *vertex* lain dan *query* jumlah *weight* dari *vertex-vertex* yang terhubung pada satu *sub-tree*. Adapun properti yang dimiliki struktur data ini sebagai berikut:

buildTree():	membangun *tree* dengan batasan yang telah ditentukan.
link(x, y):	memberi *edge* antara *vertex x* dengan *vertex y* dengan *x* sebagai *parent*-nya.
createNode(index, weight):	membuat *node* dengan input *index* dan *weight.*
changeRoot(x):	menjadikan *node* dengan indeks *x* sebagai *root* pada *tree.*

getSum(x): *return* data berupa jumlah *weight* dari *node-node* yang terhubung pada *sub-tree* dengan *x* sebagai *parent*-nya.

Kajian komprehesif akan menghasilkan rancangan algoritma yang efisien dalam menyelesaikan masalah penarikan data pada struktur data yang dinamis, dalam hal ini adalah *tree*. Salah satu situs *online judge* telah menyediakan fasilitas yang dapat digunakan untuk memeriksa kebenaran dan efisiensi algoritma yang telah dirancang [4]. Dalam laman situs tersebut dijelaskan permasalahan yang karakteristiknya sama dengan masalah penarikan data pada struktur data dinamis. Diberikan masukan berupa sejumlah *vertex* dengan bobot yang dimilikinya masing-masing sebanyak *N* dengan inisialisasi *root* berada pada *vertex* indeks 1. Terdapat *M* operasi dimana tiap satu operasi adalah salah satu dari operasi berikut:

a. Memindahkan *root* pada *tree* ke *vertex* yang ditentukan pada input.
b. Menarik data berupa jumlah dari bobot-bobot yang dimiliki *vertex-vertex* yang berada pada *sub-tree* yang ditentukan pada input.

Beberapa batasan dari permasalahan tersebut adalah sebagai berikut:

a. Nilai *N* tidak akan pernah melebihi 10^5.
b. Nilai *M* tidak akan pernah melebihi 10^5.
c. Waktu maksimal dari program untuk menyelesaikan permasalahan ini adalah 1 detik.

Dengan adanya fasilitas tersebut diharapkan proses pengujian kebenaran dan efisiensi rancangan algoritma yang dihasilkan berjalan dengan lancar serta output yang tepat.

10. METODOLOGI

a. Penyusunan proposal tugas akhir

Penyusunan proposal tugas akhir merupakan tahap awal dalam proses pengerjaan tugas akhir. Dalam proposal ini diajukan bagaimana menganalisis dan merancang algoritma yang efisien untuk menyelesaikan masalah penugasan dinamis.

b. Studi literatur

Tahap berikutnya adalah mencari informasi dan studi literatur apa saja yang bisa dijadikan referensi untuk melakukan pengerjaan tugas akhir. Literatur yang digunakan adalah beberapa *paper* dan buku yang berhubungan dengan topik yang dibahas.

c. Implementasi perangkat lunak

Implementasi merupakan tahap untuk menerapkan algoritma yang dipilih ke dalam bahasa pemrograman. Algoritma yang telah dipilih diimplementasikan dengan bahasa pemrograman C++.

d. Pengujian dan evaluasi

Proses pengujian akan dilakukan dengan mengunggah *source code* hasil implementasi pada situs *online judge* SPOJ. Diharapkan hasil yang didapat adalah implementasi yang baik dan benar dengan konsumsi waktu dan *memory* yang lebih efisien dari implementasi beberapa *user* yang telah menyelesaikan masalah ini sebelumnya. Sampai pada saat proposal ini diajukan, masih terdapat enam belas *user* yang telah menyelesaikan masalah ini.

e. Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat. Sistematika penulisan buku tugas akhir secara garis besar antara lain:

1. Pendahuluan
   a. Latar Belakang
   b. Rumusan Masalah
   c. Batasan Tugas Akhir
   d. Tujuan
   e. Metodologi
   f. Sistematika Penulisan
2. Tinjauan Pustaka
3. Desain dan Implementasi
4. Pengujian dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka

11. JADWAL KEGIATAN

Jadwal pengerjaan tugas akhir dapat dilihat pada Tabel 1.

Tabel 1. Jadwal Kegiatan

Tahapan	2014															
	Februari				Maret				April				Mei			
Penyusunan Proposal																
Studi Literatur																
Implementasi																
Pengujian dan Evaluasi																
Penyusunan Buku																

12. DAFTAR PUSTAKA

[1] Kenneth Rossen, *Descrete Mathematics and Its Applications, Sixth Edition*. New York: McGraw-Hill, 2007.

[2] "Tree (Graph Theory)," Januari 2014. [Online]. Available: http://en.wikipedia.org/wiki/Tree_(graph_theory). [Accessed 23 Februari 2014].

[3] "Left Child Right-Sibling Binary Tree," Januari 2014. [Online]. Available: http://en.wikipedia.org/wiki/left_child_right_sibling_binary_tree. [Accessed 23 Februari 2014].

[4] "Dynamically Rooted Tree," Mei 2012. [Online]. Available: http://www.spoj.com/problems/DRTREE. [Accessed 3 Februari 2014].

[5] Daniel D. Sleator and Robert Endre Tarjan, "A Data Structure for Dynamic Trees," *Journal of Computer and System Sciences*, vol. 26, no. 3, pp. 362-391, Juni 1983.