Laporan Praktikum

Algoritma dan Struktur Data

Ganjil 2023/2024
Program Studi Teknik Informatika
Institut Teknologi Sumatera

Modul : Hash

Nama : Aulia Putri Sayidina

NIM : 122140060

Kelas (Kelas Asal) : RC

Instruksi sederhana :

- Disarankan kepada **Praktikan Algoritma Struktur Data** untuk mengeditnya menggunakan Google Docs agar tidak berantakan dan rapi,
- Silahkan mengganti **Nama Modul** baik yang ada pada **Cover** dan **Header** sesuai dengan materi praktikum,
- Gunakan text styling seperti **Heading 1**, **Normal Text** yang telah terformat / Text Style lainnya yang digunakan untuk menjaga estetika laporan,
- Gunakan **Syntax Highlighter** untuk merapikan kode yang sudah Praktikan buat ke dalam Laporan Praktikum.

Dirancang oleh Kevin Simorangkir
Terinspirasi dari Format Tata Tulis Laporan Embedded

Materi Praktikum

Konsep pencarian ini mengandalkan fungsi hash untuk mengubah nilai kunci menjadi alamat hash, yang berfungsi sebagai indeks dalam struktur data seperti array. Fungsi hash dapat diimplementasikan menggunakan beberapa metode, antara lain:

a. Modulus

Menggunakan data sebagai kunci atau input fungsi.

Alamat hash dihasilkan dengan mengambil hasil bagi dari data.

b. Pemotongan

Metode ini memilih sebagian tertentu dari data sebagai alamat hash, mengabaikan bagian lainnya.

c. Pelipatan

Kunci dibagi menjadi bagian-bagian, contohnya per dua digit.

Setiap bagian dijumlahkan. Hasilnya kemudian dipotong agar masuk ke dalam rentang alamat pada tabel hash.

Tujuan dari pendekatan ini adalah untuk secara efisien menghubungkan data dengan lokasi atau alamat yang dapat digunakan dalam struktur data tertentu.

Link repl.it Source Code

https://onlinegdb.com/3jKCwT_SQx

Source Code

```
1. //Aulia Putri Sayidina
2. //122140060
3.
4. #include <iostream>
5. #include <string>
6. using namespace std;
7.
8. #define Info(P) (P)->info
9. #define Next(P) (P)->next
10.#define First(H, i) (H)[i].first
11.
```

```
12.typedef string infotype;
13.const int MaxEl = 26;
14.
15.typedef struct Tnode *addressNode;
16.typedef struct Tnode
17.{
18.    infotype info;
19.    addressNode next;
20.} Node;
21.
22.typedef struct THash *addressHash;
23.typedef struct THash
24.{
25.    addressNode first;
26.} Hash;
27.
28.void createEmptyHash(addressHash HashTable)
29.{
30.    for (int i = 0; i < MaxEl; i++)
31.    {
32.        First(HashTable, i) = NULL;
33.    }
34.}
35.
36.addressNode NodeAllocation(infotype x)
37.{
38.    addressNode NewNode;
39.
40.    NewNode = new Node;
41.    Info(NewNode) = x;
42.    Next(NewNode) = NULL;
43.
44.    return NewNode;
45.}
46.
47.bool isEmptyFirst(addressNode First_Node)
48.{
49.    return (First_Node == NULL);
50.}
51.
52.void NodeDeallocation(addressNode hapus)
53.{
54.    delete hapus;
55.}
56.
```

```
57.void insertFirst(addressNode *First_Node, infotype x)
58.{
59.    addressNode NewNode = NodeAllocation(x);
60.    Next(NewNode) = *First_Node;
61.    *First_Node = NewNode;
62.}
63.
64.void insertLast(addressNode *First_Node, infotype x)
65.{
66.    addressNode NewNode = NodeAllocation(x), temp = *First_Node;
67.
68.    if (*First Node == NULL)
69.    {
70.        *First_Node = NewNode;
71.    }
72.    else
73.    {
74.        while (Next(temp) != NULL)
75.        {
76.            temp = Next(temp);
77.        }
78.
79.        Next(temp) = NewNode;
80.    }
81.}
82.
83.void deleteFirst(addressNode *First_Node)
84.{
85.    if (*First_Node != NULL)
86.    {
87.        addressNode temp;
88.
89.        temp = *First_Node;
90.        *First_Node = Next(*First_Node);
91.        Next(temp) = NULL;
92.
93.        NodeDeallocation(temp);
94.    }
95.}
96.
97.void deleteAfter(addressNode *Pred)
98.{
99.    if (*Pred != NULL && Next(*Pred) != NULL)
100.        {
101.            addressNode temp;
```

```
102.
103.            temp = Next(*Pred);
104.            Next(*Pred) = Next(temp);
105.
106.            NodeDeallocation(temp);
107.        }
108.    }
109.
110.    void deleteLast(addressNode *First_Node)
111.    {
112.        if (*First_Node != NULL)
113.        {
114.            addressNode temp, predTemp;
115.
116.            predTemp = NULL;
117.            temp = *First_Node;
118.
119.            while (Next(temp) != NULL)
120.            {
121.                predTemp = temp;
122.                temp = Next(temp);
123.            }
124.
125.            if (predTemp == NULL)
126.            {
127.                *First_Node = NULL;
128.            }
129.            else
130.            {
131.                deleteAfter(&predTemp);
132.            }
133.        }
134.    }
135.
136.    void insertByModFunc(addressHash HashTable, infotype x)
137.    {
138.        int index = x[0] - 'a'; // Menggunakan huruf pertama sebagai indeks
139.        addressNode *First_Node = &First(HashTable, index);
140.
141.        if (isEmptyFirst(*First_Node))
142.        {
143.            insertFirst(First_Node, x);
144.        }
145.        else
146.        {
```

```
147.             insertLast(First_Node, x);
148.         }
149.     }
150.
151.     void printHashTable(addressHash HashTable)
152.     {
153.         for (int i = 0; i < MaxEl; i++)
154.         {
155.             addressNode temp = First(HashTable, i);
156.             cout << "[" << i << "]: ";
157.
158.             while (temp != NULL)
159.             {
160.                 cout << Info(temp) << " -> ";
161.                 temp = Next(temp);
162.             }
163.             cout << "NULL" << endl;
164.         }
165.     }
166.
167.     int main(){
168.         addressHash HashTable;
169.         HashTable = new Hash[MaxEl];
170.         createEmptyHash(HashTable);
171.
172.         insertByModFunc(HashTable, "Budi");
173.         insertByModFunc(HashTable, "Siti");
174.         insertByModFunc(HashTable, "Rahmat");
175.
176.         deleteFirst(&First(HashTable, 1));
177.
178.         deleteLast(&First(HashTable, 0));
179.
180.         insertFirst(&First(HashTable, 2), "Anita");
181.
182.         insertLast(&First(HashTable, 0), "Joko");
183.
184.         printHashTable(HashTable);
185.
186.         delete[] HashTable;
187.
188.         return 0;
189.     }
190.
191.
```

Dokumentasi Hasil Running

```
[0]: Joko -> NULL
[1]: NULL
[2]: Anita -> NULL
[3]: NULL
[4]: NULL
[5]: NULL
[6]: NULL
[7]: NULL
[8]: NULL
[9]: NULL
[10]: NULL
[11]: NULL
[12]: NULL
[13]: NULL
[14]: NULL
[15]: NULL
[16]: NULL
[17]: NULL
[18]: NULL
[19]: NULL
[20]: NULL
[21]: NULL
[22]: NULL
[23]: NULL
[24]: NULL
[25]: NULL
```

Gambar 1. Hasil Running Code Hash

Referensi

Modul perkuliahan Algoritma Struktur Data - Hash dari ITERA