ALGORITMA DAN STRUKTUR DATA

INF1083

LAPORAN PRAKTIKUM 11 : Algoritma Pohon

Oleh :

Akhmad Qasim

2211102441237

Teknik Informatika
Fakultas Sains & Teknologi
Universitas Muhammadiyah Kalimantan Timur

Samarinda, 2023

Laporan Praktikum 11:
Algoritma Pohon

Pokok Bahasan:

- ❖ Algoritma Pohon
- ❖ Binary Heap

Tujuan Pembelajaran:

- ✓ Memahami implementasi algoritma pohon menggunakan struktur list, kelas dan referensi
- ✓ Memahami implementasi operasi binary heap

Percobaan & Latihan:

1. **Representasi List**
   a. **Berikan tampilan output dari kedua program diatas!**
   Tampilan output:

```
['a', ['b', ['d', [], []], ['e', [], []]], ['c', ['f', [], []], []]]
left subtree =  ['b', ['d', [], []], ['e', [], []]]
root =  a
right subtree =  ['c', ['f', [], []], []]
[5, [4, [], []], []]
[3, [9, [4, [], []], []], [7, [], [6, [], []]]]
[3, [9, [11, [4, [], []], []], []], [7, [], [6, [], []]]]
[6, [], []]
```

   b. **Jelaskan kegunaan fungsi-fungsi yang terdapat di program kedua!**
      i. BinaryTree
      Fungsi ini digunakan untuk membuat dan menginisialisasi sebuah binary tree baru dengan root value r dan mengembalikan sebuah array yang berisi root value, left subtree (awalnya kosong), dan right subtree (awalnya kosong).
      ii. insertLeft
      Fungsi ini digunakan untuk memasukkan newBranch sebagai subtree kiri dari root, setelah itu memindahkan subtree kiri yang ada pada root ke subtree kiri dari newBranch. Jika subtree kiri pada root tidak kosong, newBranch akan menjadi subtree kiri baru dan subtree kiri sebelumnya akan menjadi subtree kiri dari newBranch. Jika subtree kiri pada root kosong, newBranch akan menjadi

subtree kiri baru dan subtree kiri sebelumnya akan menjadi subtree kiri kosong. Mengembalikan binary tree root setelah dilakukan penambahan subtree kiri.

iii. insertRight
Fungsi ini digunakan untuk memasukkan newBranch sebagai subtree kanan dari root setelah itu memindahkan subtree kanan yang ada pada root ke subtree kanan dari newBranch. Jika subtree kanan pada root tidak kosong, newBranch akan menjadi subtree kanan baru dan subtree kanan sebelumnya akan menjadi subtree kanan dari newBranch. Jika subtree kanan pada root kosong, newBranch akan menjadi subtree kanan baru dan subtree kanan sebelumnya akan menjadi subtree kanan kosong. Dan akhirnya mengembalikan binary tree root setelah dilakukan penambahan subtree kanan.

iv. getRootVal
Fungsi ini digunakan untuk mengembalikan nilai root dari binary tree root.

v. setRootVal
Fungsi ini digunakan untuk mengubah nilai root dari binary tree root menjadi newVal.

vi. getLeftChild
Fungsi ini digunakan untuk mengembalikan subtree kiri dari binary tree root.

vii. getRightChild
Fungsi ini digunakan untuk mengembalikan subtree kanan dari binary tree root.

2. Node dan Referensi

a. Berikan tampilan output dari program diatas!

Tampilan output:

```
a
None
<__main__.BinaryTree object at 0x0000015AD9697D00>
b
<__main__.BinaryTree object at 0x0000015AD9697C70>
c
hello
```

b. Jelaskan kegunaan kelas BinaryTree pada program diatas!

Kelas BinaryTree pada program di atas digunakan untuk membuat objek binary tree. Setiap objek BinaryTree memiliki atribut key untuk

menyimpan nilai root, leftChild untuk menyimpan subtree kiri, dan rightChild untuk menyimpan subtree kanan.

Dengan menggunakan kelas BinaryTree, kita dapat memanipulasi binary tree dengan lebih mudah dan terstruktur, sehingga memudahkan dalam implementasi dan penggunaan struktur data binary tree dalam program.

3. **Pohon Parse**
   a. **Berikan tampilan output pada program diatas!**
   Tampilan output:

```
10
5
+
3
*
```

   b. **Pada baris keberapa penggunaan algoritma stack pada program diatas?**
   Pada baris ke 6, 8, 13, 17, 22, dan 25.

   c. **Jelaskan kegunaan fungsi buildParseTree pada program diatas!**
   Fungsi buildParseTree digunakan untuk membangun Parse Tree (pohon parsing) dari sebuah ekspresi matematika yang diberikan dalam bentuk notasi postfix (postfix expression). Fungsi ini menerima argumen fpexp yang merupakan ekspresi matematika dalam bentuk string.

4. **Implementasi Operasi Binary Heap**
   a. **Berikan tampilan output pada program diatas!**
   Tampilan output:

```
2
3
5
6
9
```

   b. **Berikan keterangan penjelasan pada baris ke 46 hingga 52 pada program diatas!**
   Tampilan syntax:

```
45      def buildHeap(self, alist):  # Membangun heap
46          i = len(alist) // 2  # Menentukan nilai i
47          self.currentSize = len(alist)  # Menentukan ukuran heap
48          self.heapList = [0] + alist[:]  # Menyalin isi alist ke heapList
49          while (i > 0):  # Melakukan perulangan jika i lebih dari 0
50              self.percDown(i)  # Memanggil fungsi percDown
51              i = i - 1  # Mengurangi nilai i
```

c. Lakukanlah uji coba dengan item yang berbeda pada baris ke-55 pada program diatas dan berikan hasil output dan analisa dari uji coba!

Tampilan syntax:

```
54  bh = BinHeap()
55  bh.buildHeap([2, 4, 6, 3, 5])
```

Tampilan output:

```
2
3
4
5
6
```

Kesimpulan:

Pada praktikum kali ini membahas tentang algoritma pohon dan binary heap. Pohon digunakan untuk merepresentasikan struktur hierarkis, sedangkan binary heap adalah struktur data yang digunakan untuk menyimpan koleksi elemen dengan properti tertentu. Pohon memungkinkan pengaturan data secara hierarkis dan efisien, sementara binary heap menyediakan operasi cepat seperti penyisipan dan penghapusan elemen teratas. Dalam pengembangan perangkat lunak, pemahaman algoritma pohon dan binary heap sangat penting untuk memecahkan masalah yang melibatkan struktur data dan pengelolaan array atau list.