Implementasi Penggunaan Pohon untuk Menuliskan Seluruh Kejadian Permutasi

Stefanus Stanley Yoga Setiawan 13518122
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13518122@std.stei.itb.ac.id

***Abstract*—Salah satu cara untuk mencari banyaknya kejadian yang mungkin adalah dengan permutasi atau kombinasi. Hal ini berkaitan erat dengan dunia pemrograman tentang bagaimana menulis seluruh kejadian permutasi tersebut. Hal ini dapat diselesaikan dengan menggunakan rekursif pada pohon.**

***Keywords*—Pohon, Rekursif, Permutasi.**

I. PENDAHULUAN

Permutasi adalah penyusunan kembali urutan dari kumpulan karakter yang ada menjadi urutan yang berbeda. Semisal dalam *string* "abc", bisa disusun kembali menjadi "acb", "bac", "bca", "cab", "cba". Dalam kehidupan sehari-hari, permutasi dapat digunakan untuk menghitung banyaknya variasi yang dapat dipakai. Contoh penggunaan permutasi pada kehidupan nyata, adalah saat diminta memilih tempat duduk, salah satu kejadian yang mungkin adalah salah satu dari seluruh kejadian permutasi.

Salah satu cabang yang memakai teori kombinatorial ini adalah keamanan *cyber*, yaitu saat melakukan peretasan, bisa digunakan salah satu cara yaitu teknik *brute force* yang menggunakan seluruh kemungkinan yang ada dan dengan memasukkan satu per satu variasi yang didapatkan.

Pada makalah ini, akan dibahas algoritma untuk menentukan seluruh kejadian permutasi dengan Bahasa Java.

II. DASAR TEORI

A. Pohon

1. Definisi Pohon

   Pohon adalah sebuah graf tak berarah yang terhubung dan tidak mengandung sirkuit di dalamnya.

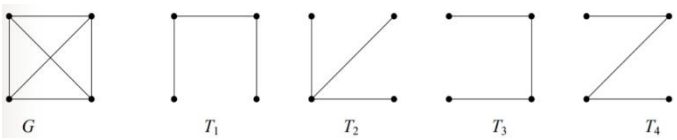
   Misalkan G = (V,E) adalah graf tak berarah sederhana dan jumlah simpulnya n. Maka, semua pernyataan di bawah ini adalah ekivalen:

   1. G adalah pohon
   2. Setiap pasang simpul di dalam G terhubung dengan lintasan tunggal.
   3. G terhubung dan memiliki m = n - 1 buah sisi.
   4. G tidak mengandung sirkuit dan memiliki m = n – 1 buah sisi.
   5. G tidak mengandung sirkuit dan penambahan satu sisi pada graf akan membuat hanya satu sirkuit.
   6. G terhubung dan semua sisinya adalah jembatan.

   Teorema tersebut adalah definisi dari pohon.

2. Pohon Merentang

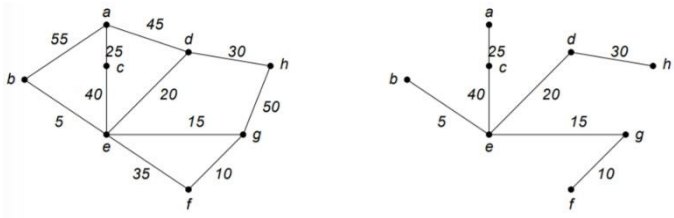
   Pohon merentang dari sebuah graf yang terhubung adalah upagraf merentang yang berupa pohon. Pohon merentang dapat diperoleh dengan memutus sirkuit di dalam graf. Setiap graf terhubung mempunyai paling sedikit satu buah pohon merentang. Graf tak terhubung dengan k komponen mempunya k buah hutan merentang yang disebut hutan merentang.



Gambar 1: Pohon merentang
(Sumber: http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf diakses pada 1 Desember 2019)

3. Pohon Merentang Minimum

   Sebuah graf terhubung-berbobot mumgkin memiliki lebih dari satu buah pohon merentang. Pohon merentang yang berbobot minimum dinamakan pohon merentang minimum.

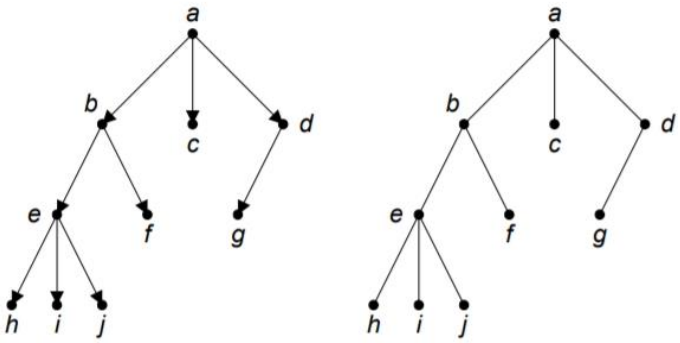


Gambar 2: Pohon merentang minimum
(Sumber: http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf diakses pada 1 Desember 2019)

4. Pohon Berakar

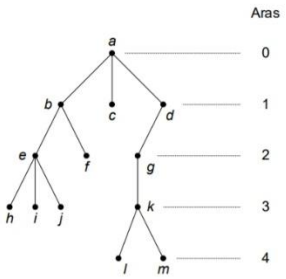
   Pohon berakar adalah pohon yang satu buah simpulnya diperlakukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah dinamakan

pohon berakar.



Gambar 4: Pohon berakar
(Sumber: http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf diakses pada 1 Desember 2019)

5. Terminologi Pohon Berakar
   a. Anak dan Orang Tua (Akar)
      Simpul b, c, dan d adalah anak dari simpul a dan a adalah orang tua dari b, c, maupun d.
   b. Lintasan
      Lintasan dari a ke j adalah a, b, e, j. Panjang lintasan dari a ke j adalah 3.
   c. Saudara kandung
      f adalah saudara kandung e, tetapi g bukan saudara e, karena orang tua mereka berbeda.
   d. Upapohon
      Simpul b dengan simpul di bawah-bawahnya dapat dibilang upapohon dari simpul a.
   e. Derajat
      Derajat sebuah simpul adalah jumlah upapohon (atau jumlah anak) pada simpul tersebut. Derajat a adalah 3, derajat b adalah 2. Derajat d adalah satu dan derajat c adalah 0. Jadi, derajat yang dimaksudkan di sini adalah derajat-keluar. Derajat maksimum dari semua simpul merupakan derajat pohon itu sendiri. Pohon di atas berederajat 3.
   f. Daun
      Simpul yang berderajat nol (atau tidak mempunyai anak) disebut daun. Simpul h, i, j, f, c, l, dan m adalah daun.;
   g. Simpul dalam
      Simpul yang mempunyai anak disebut simpul dalam. Simpul b, d, e, g, dan k adalah simpul dalam.
   h. Tingkat
      Tingkat menyatakan kedalaman suatu simpul ditinjau dari akar. Simpul a berada pada tingkat 0, simpul b, c, d, berada pada tingkat 1, dst.

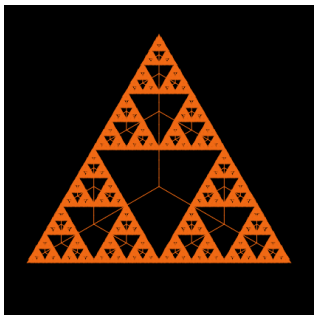


Gambar 5: Level
(Sumber: http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf diakses pada 1 Desember 2019)

   i. Tinggi atau kedalaman
      Tingkat maksimum dari suatu pohon disebut tinggi atau kedalaman pohon tersebut. Pohon di atas mempunya tinggi empat.

B. Rekursif

Rekursif adalah sesuatu yang diulang dengan dirinya sendiri. Pada dunia pemrograman, rekursif biasa dipakai sebagai fungsi rekursif, yaitu fungsi yang memanggil dirinya sendiri.



Gambar 5: rekursif
(Sumber: http://informatika.stei.itb.ac.id/~rinaldi.munir/ diakses pada 1 Desember 2019)

Fungsi rekursif didefinisikan oleh dua bagian:

   a. Basis
      Basis adalah bagian yang berisi nilai fungsi yang terdefinisi secara eksplisit. Bagian ini juga merupakan bagian yang berfungsi memberhentikan rekursif.
   b. Rekurens
      Bagian ini mendefinisikan fungsi dalam terminologi dirinya sendiri. Bagian ini juga berisi kaidah untuk menemukan nilai fungsi pada suatu input dari nilai-nilai lainnya pada input yang lebih kecil.

Berikut adalah contoh program yang menghitung 5! dengan menggunakan teknik rekursif (Bahasa C++).

```
#include <bits/stdc++.h>
using namespace std;

int faktorial(int n){
    if(n == 1){
        return 1;
```

```
    }
    else{
        return n * faktorial(n-1);
    }
}

// Driver
int main(){
    int N = faktorial(5);
    cout << N << endl;
    return 0;
}
```

Gambar 6: 5 Faktorial dengan metode rekursif

C. Kombinatorial

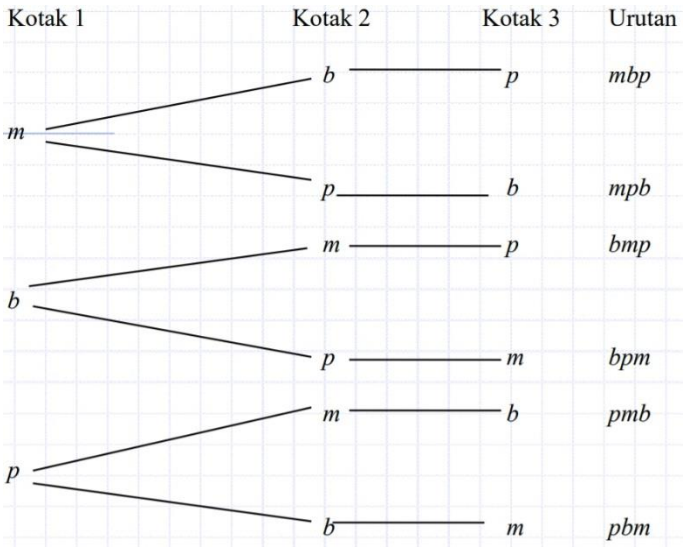
Kombinatorial adalah cabang matematika untuk menghtiung jumlah penyusunan objek-objek tanpa harus mengenumerasi semua kemungkinan susunannya.

Kombinatorial memiliki dua perhitungan yang mendasar yang jika digunakan, dapat memecahkan banyak sekali masalah. 2 perhitungan tersebut adalah kaidah perkalian dan kaidah penjumlahan.

1. Kaidah Perkalian
   Kaidah perkalian adalah perhitungan dengan cara mengalikan kedua kejadian. Kejadian tersebut akan terjadi secara bersamaan.
   Misalkan:
   Percobaan 1: p hasil
   Percobaan 2: q hasil
   Percobaan 1 dan 2: p x q hasil
2. Kaidah Penjumlahan
   Kaidah penjumlahan adalah perhitungan dengan cara menjumlahkan kedua kejadian. Kejadian yang terjadi adalah salah satu di antara dua kejadian tersebut.
   Misalkan:
   Percobaan 1: p hasil
   Percobaan 2: q hasil
   Percobaan 1 atau 2: p + q hasil

D. Permutasi

Permutasi adalah jumlah urutuan berbeda dari pengaturan objek-objek. Permutasi merupakan bentuk khusus aplikasi kaidah perkalian. Permutasi dapat dituliskan dengan "P". P(n,r) adalah jumlah banyaknya cara urutan r dapat dipilih dari n buah elemen.



Gambar 7: Contoh penerapan permutasi pada urutan pengambilan bola
(Sumber: http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2014-2015/Kombinatorial%20(2014).pdf diakses pada 1 Desember 2019)

III. PEMBAHASAN

A. Inisialisasi

```
public class string_permutation{

    public char[] text;
    public int[] occurence;
    public int length;
    public int answer_count = 0;

    public void setter(){
        this.text = new char[256];
        this.occurence = new int[256];
    }
```

Gambar 8: Inisialisasi

Hal pertama yang harus dilakukan dalam menggunakan Bahasa Java adalah menginisialisasi dahulu komponen-komponen yang akan digunakan kelas tersebut. Pada program ini, komponen yang palin gutama adalah *array of char* dan *array of int.*

B. Banyaknya karakter

```
public void getOccuringChar(String str){
    int count[] = new int[256];
    int len = str.length();

    for(int i = 0; i < len; i++){
        count[str.charAt(i)]++;
    }

    char character[] = new char[str.length()];
    int counter = 0;
    for(int i = 0; i < len; i++){
        character[i] = str.charAt(i);
        int mark = 0;
        for(int j = 0; j < len; j++){
            if(character[j] == str.charAt(i)){
                mark++;
            }
        }
        if(mark == 1){
            this.text[counter] = str.charAt(i);
            this.occurence[counter] = count[str.charAt(i)]
            counter++;
        }
    }
    this.length = counter;
}
```

Gambar 9: Banyaknya Karakter

Pada kode ini, keluaran yang didapat adalah daftar karakter yang ada dan banyaknya karakter tersebut muncul dalam sebuah *string*. Pertama kali, kita melakukan pengulangan untuk menghitung karakter yang muncul pada sebuah *string.* Untuk bagian setelah loop pertama, adalah bagian untuk menyimpan hasil simpanan banyaknya karakter yang muncul ke dalam variabel text dan occurence yang dimiliki kelas tersebut.

C. Rekursif

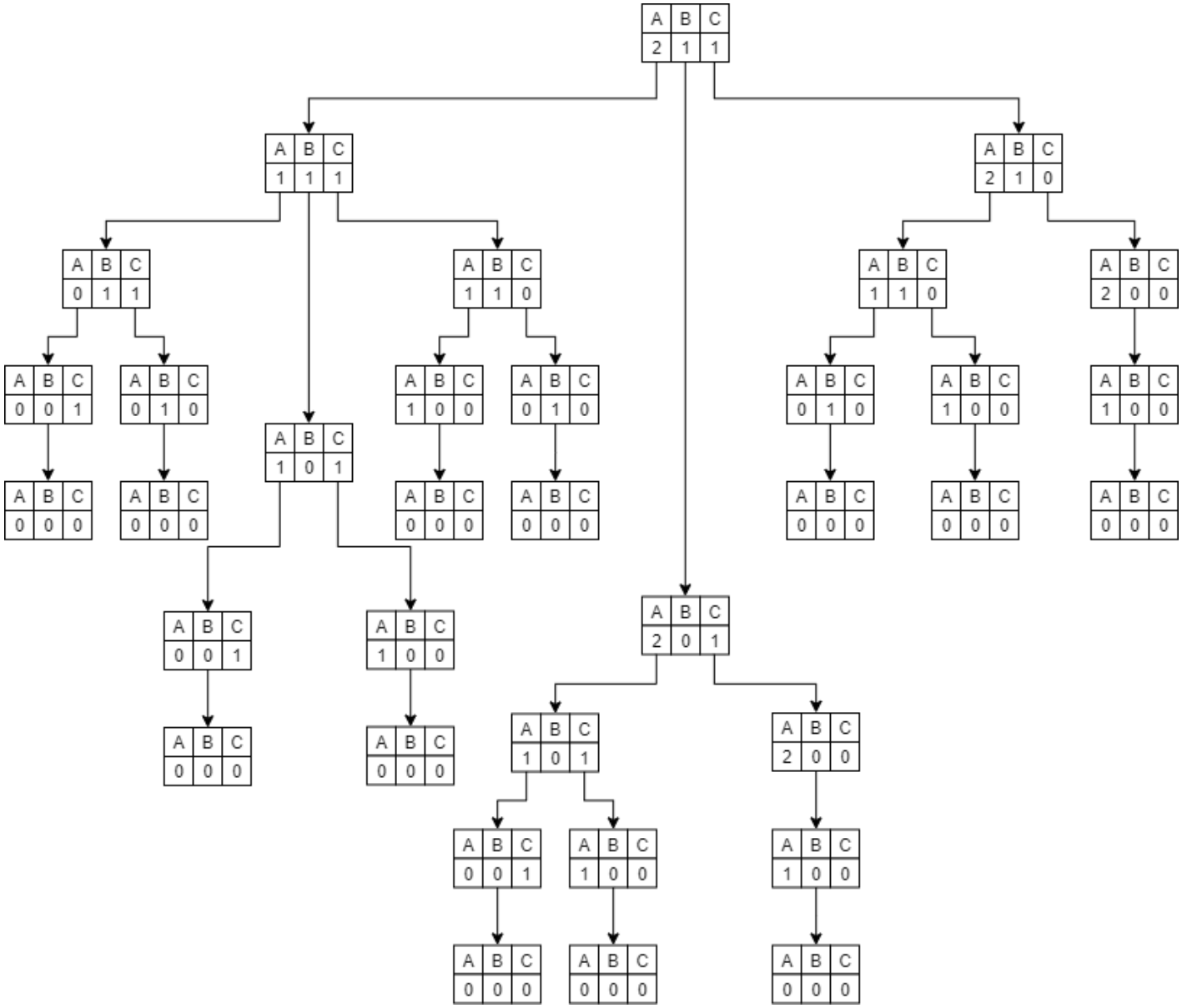
```
public void stringPermutation(char[] str,
int[] count, char[] res, int level){
    if(level == res.length){
        this.answer_count++;
        printArr(res);
        return;
    }
    for(int i = 0; i < str.length; i++){
        if(count[i] == 0){
            continue;
        }
        res[level] = str[i];
        count[i]--;
        stringPermutation(str,count,res,level+1);
        count[i]++;
    }
}

public void printArr(char[] str){
    for(int i = 0; i < str.length; i++){
        System.out.print(str[i]);
    }
    System.out.println();
}
```

Gambar 10: Rekursif

Pada bagian ini, Akan dilakukan rekursif pada fungsi pengulangan ke-2.

D. Pohon

Proses rekursif ini dapat digambarkan dengan menggunakan pohon, saat turun, akan ditulis terlebih dahulu pada bagian kiri pohon, saat naik, dan turun lagi, akan ditulis ke kanannya. Pohon rekursif untuk penulisan semua kemungkinan kejadian yang ada dapat dilihat di ilustrasi bawah. Akan diilustrasikan bagaimana menuliskan seluruh kejadian yang ada pada *string* “AABC”.



Gambar 11: Pohon Rekursif

E. Hasil Tes

```
E:\Learning\MATDIS>java string_permutation.java
Masukkan string: AABC
AABC
AACB
ABAC
ABCA
ACAB
ACBA
BAAC
BACA
BCAA
CAAB
CABA
CBAA
Total permutation = 12
```

Gambar 12: Haisl tes program

Dilihat dari hasil tes program tersebut, dengan masukan "AABC", didapatkan 12 kejadian yang muncul, hal ini terlihat dari ilustrasi pohon di atas.

IV. KESIMPULAN

Aplikasi untuk menuliskan seluruh kejadian pada permutasi dapat digunakan. Dalam hal ini, dapat dilihat kompleksitas program adalah O(n!). Hal ini teknik yang digunakan adalah rekursif. Maka dari itu, program dapat berjalan lama jika digunakan untuk mencari pasangan *string* yang dapat membentuk banyak sekali pasangan.

Penggunaan aplikasi ini sangatlah mudah, pengguna hanya perlu memasukkan inputan berupa string (semisal "AABC" pada contoh di atas), kemudian program akan secara otomatis memunculkan semua kemungkinan yang muncul.

Dengan berkembangnya ilmu pengetahuan sekarang, ada kemungkinan dapat digunakan cara yang lebih cepat untuk membuat aplikasi seperti ini.

Bagaimanapun juga, program yang dibuat ini dapat diaplikasikan pada beberapa hal, seperti menyelesaikan teka-teki 24 dengan metode *brute-force*, yaitu dengan mencari variasi urutan angka yang dapat digunakan. Hal ini juga dapat digunakan untuk meretas sandi seseorang dengan metode *brute-force.*

V. UCAPAN TERIMA KASIH

Penulis berterima kasih kepada Tuhan Yang Maha Esa yang telah berkat-Nya sehingga penulis dapat menyelesaikan makalah ini. Penulis juga berterima kasih kepada Ibu Harlili selaku dosen K2 karena telah membimbing saya dan memberikan ilmu kepada saya karena jika tanpa dia, makalah ini tidak akan terselesaikan sebagaimana mestinya. Penulis juga berterima kasih kepada teman penulis yang atas dukungannya, penulis dapat mengerjakan, dan menyelesaikan makalah ini.

REFERENCES

[1] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf Diakses pada 1 Desember 2019.
[2] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2014-2015/Kombinatorial%20(2014).pdf Diakses pada 1 Desember 2019.
[3] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Rekursi%20dan%20Relasi%20Rekurens%20(2015).pdf Diakses pada 1 Desmber 2019.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 1 Desember 2019

Stefanus Stanley Yoga Setiawan - 13518122