Aplikasi Rekursif dan Pohon dalam Ilmu Komputer

Kevin / 13512097
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13512097@std.stei.itb.ac.id

Abstract – Makalah ini akan membahas tentang aplikasi rekursif dan pohon dalam Ilmu Komputer. Rekursif yang secara umum berarti sesuatu yang mengalami pengulangan dan dalam pengulangannya tercangkup dirinnya sendiri ini ternyata sangat bermanfaat dalam dunia pemrograman. Begitu juga pohon. Ada 3 algoritma rekursif yang menjadi topik pembahasan pada makalah ini yaitu binary search tree, Fibonacci search dan menara Hanoi.

Kata kunci : Rekursif, pohon, Fibonacci, menara Hanoi, Binary Search Tree.

I. PENDAHULUAN

Sadar atau tidak sadar dalam kehidupan sehari-hari kita banyak menemukan kasus yang bersifat rekursif.Rekursif memiliki makna bahwasanya sesuatu tersebut mengalami pengulangan dan dalam pengulangannya tercangkup dirinya sendiri. Contoh kejadian rekursif dalam kehidupan sehari-hari adalah ketika dua cermin diletakkan secara paralel,maka di dalam cermin tersebut akan terbentuk bayangan cermin di dalam cermin sekian banyaknya.

Begitu juga pohon. Struktur keorganisasian himpunan, unit maupun suatu lembaga merupakan contoh dari kasus pohon dalam kehidupan sehari-hari. Suatu struktur dikatakan berbentuk pohon apabila objek yang sebelumnya adalah anggota/anak bisa menjadi kepala/orang tua dan juga memiliki anggota.Meskipun demikian hanya tetap ada satu objek yang menjadi kepala/orangtua dari semua objek lain. Pada kasus satu struktur organisasi, ketua himpunan adalah root dari semua anggotanya,ketua himpunan tersebut memiliki beberapa ketua divisi yang menaungi anggota- anggota biasa. Begitu juga dalam garis keturunan seseorang. Seorang memiliki anak dan anak ini kemudian menjadi orangtua dari anaknya.Demikian seterusnya.

Dalam bidang komputer. Kita juga dapat menjumpai kasus yang bersifat rekursif. Contohnya adalah penggunaan virtual box yang menjadikan program tersebut menjadi sebuah komputer maya di dalam komputer kita. Di dalam komputer maya tersebut kita dapat melakukan banyak hal seperti layaknya di komputer nyata kita. Selain pada hardware, kita juga dapat menjumpai kasus rekursif pada software seperti pada algoritma sorting, searching, delete, insert yang bersifat rekursif. Namun pada makalah ini,kita hanya akan membahas kasus rekursif pada Fibonacci search, binary search tree dan menara Hanoi.

II. TEORI DASAR

II.1 Rekurens

Rekursif adalah sesuatu yang memiliki sifat pengulangan dan dalam pengulangannya terkandung definisi tentang dirinya sendiri. Sedangkan definisi rekursi adalah proses merekursifkan sesuatu.Contohnya dari rekursif adalah mimpi dalam mimpi.Gambar di dalam gambar.

Gambar 1. Gambar yang bersifat rekursif
Sumber : http://www.google.co.id/imgres?sa=X&noj=1&biw=1366&bih=629&tbm=isch&tbnid=ZJ5nsQ7j1LnYdM:&imgrefurl=http://achmatim.net/tag/algoritma/&docid=fG_7NrNIJVsQSM&imgurl=http://achmatim.net/wpcontent/uploads/2012/03/rekursif.jpg&w=282&h=280&ei=QGeuUtH5IcePrQfew4HYDA&zoom=1&ved=1t:3588,r:4,s:0,i:88&iact=rc&page=1&tbnh=141&tbnw=112&start=0&ndsp=18&tx=54&ty=53

Pada dasarnya,fungsi rekursif terdiri atas dua bagian,yaitu:

a. Bagian Basis :
 - Pada bagian ini berisi nilai dari fungsi.Selain, program akan berhenti jika sudah mencapai bagian ini.
b. Bagian Rekurens :
 - Pada bagian ini berisi pengulangan dimana pada bagian ini akan memanggil fungsi dirinya sendiri dengan input parameter yang sesuai.Pada bagian ini juga berisi kaidah-kaidah sesuai dengan yang diinginkan.

Jika di dalam suatu persamaan terdapat sifat rekursif maka persamaan tersebut dinamakan relasi rekurens. Contoh:

$$F(n) = 0 \qquad n=0,$$
$$F(n) = 1 \qquad n=1,$$
$$F(n) = F(n-1) + F(n-2) \qquad n>1, \qquad (1)$$

Sumber : Diktat Kuliah IF2120 Matematika Diskrit edisi keempat

Relasi rekurens di atas merupakan definisi rekursif dari barisan Fibonacci. Penemu yang menemukan barisan Fibonacci adalah Leonardo da Pisa pada sekitar abad 12.

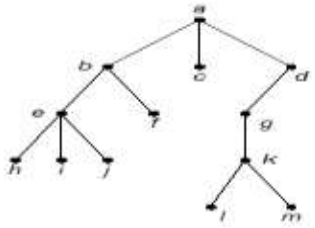
Kasus lain yang bersifat rekursif adalah pada kasus menara Hanoi.Menara Hanoi digagaskan pertama kali oleh matematikawan Prancis yang bernama Edouard Lucas pada tahun 1883. Gagasan ini timbul ketika dia berkunjung ke salah satu kuil yang dijaga oleh Pendeta Brahma.Dalam kuil tersebut terdapat satu ruangan besar dimana di dalamnya terdapat tiga tiang besar yang dikelilingi 64 cakram. Konon ada legenda bahwa ketika 64 cakram tersebut berhasil dipindahkan dari satu tiang ke tiang lainnya dengan aturan cakram yang kecil selalu ada di atas cakram yang lebih besar, maka dunia akan kiamat.

Gambar 2. Menara Hanoi dengan 5 buah cakram.
Sumber : http://www.google.co.id/imgres?noj=1&biw=1366&bih=629&tbm=isch&tbnid=3ssGiIp86l65YM:&imgrefurl=http://mangkatnekat.wordpress.com/2010/07/31/memecahkan-permasalahan-menara-hanoi-dengan-java/&docid=Oz0o1Sk0SLlzZM&imgurl=http://mangkatnekat.files.wordpress.com/2010/07/tower_of_hanoi.jpg&w=640&h=459&ei=uGuUtG4GLcyQGnoYDoAw&zoom=1&ved=1t:3588,r:2,s:0,i:84&iact=rc&page=1&tbnh=190&tbnw=265&start=0&ndsp=15&tx=203&ty=53

Selain satu aturan di atas. Aturan lain dalam memindahkan cakram yaitu dalam satu waktu,cakram yang boleh dipindahkan hanyalah satu.Jika dalam satu tiang terdapat tumpukan cakram.Maka cakram paling ataslah yang diambil.

II.2 Pohon.

Pohon adalah graf terhubung yang tidak mengandung sirkuit dimana satu buah simpul diperlakukan sebagai root(akar) dan sisanya sebagi anaknya.Pohon yang memiliki n buah simpul akan memiliki sisi sebanyak n-1.



Gambar 3. Pohon berakar.
Sumber : Diktat Kuliah IF2120 Matematika Diskrit edisi keempat.

Beberapa istilah dalam Pohon yaitu :

a. Node parent dan child.
Pada gambar di atas,node a berperan sebagai node(simpul) parent(orang tua) dari node child(anak) b,c,d.Sedangkan node b juga berperan sebagai node parent dari node child e dan f.

b. Daun dan simpul dalam.
Sebuah simpul dikatakan simpul dalam jika dia mempunyai simpul anak. Sebaliknya, jika dia tidak mempunyai simpul anak maka dinamakan daun.

c. Aras(level) atau tingkat.
Pada gambar 3 diatas simpul a berada pada tingkat 0. Simpul b pada tingkat 1 dan seterusnya.

d. Kedalaman atau tinggi.
Kedalaman adalah tingkat maksimum pada suatu pohon.

e. Lintasan
Lintasan dari b ke j adalah b,e,j.Panjangnya adalah dua.

f. Saudara kandung
Node yang memiliki parent yang sama dikatakan bersaudara kandung.Salah satu contoh dari gambar 3 di atas adalah simpul b,c,d.

g. Derajat atau Degree.
Jumlah banyaknya anak yang dimiliki oleh suatu simpul.

h. Upapohon.
Anak dari sebuah simpul beserta seluruh anak dari anak dari simpul itu yang membentuk sebuah pohon dinamakan upapohon.

Berdasarkan ada atau tidaknya arah dari setiap simpul parent ke simpul anaknya, maka pohon dibedakan atas :

a. Pohon
Graf yang tidak mempunyai arah.

b. Pohon berakar
Graf yang mempunyai arah dari simpul parent ke simpul anak. Namun pada umumnya orang juga menyebut pohon berakar sebagai "pohon" saja.Demikian juga dalam makalah ini,penulis akan memakai istilah pohon untuk pohon berakar.Karenan pada umumnya pohon yang sering ditemui adalah pohon berakar.

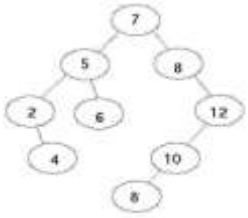
Selain berdasarkan ada atau tidaknya arah setiap simpul, pohon juga dibedakan atas keterurutan anaknya,yaitu :

a. Pohon tidak terurut
Pohon dimana keterutan anaknya tidaklah penting.

b. Pohon terurut.
Pohon dimana keterutan anak adalah penting.

Pohon juga dibedakan berdasarkan derajat maksimumnya. Pohon dengan derajat maksimum dua disebut sebagai pohon biner. Pohon dengan derajat maksimum n disebut sebagai pohon n-ary.

Salah satu pohon yang sering dipakai adalah binary search tree dimana nilai dari node-node yang berada di sebelah kirinya selalu bernilai lebih kecil dari akar dan nilai dari node-node di sebelah kanan selalu lebih besar dari akarnya.



Gambar 4 : Binary Search tree
Sumber : https://www.google.co.id/search?q=binary+search+tree&noj=1&source=lnms&tbm=isch&sa=X&ei=u9uUra2H6_IsASDwIKgDg&ved=0CAkQ_AUoAQ&biw=1366&bih=666#facrc=_&imgdii=_&imgrc=AVlJ1Dog0YiIM%3A%3BWxp1G5FC0gFsQM%3Bhttp%253A%252F%252Fmax.cs.kzoo.edu%252FAP%252FABMaterials%252FTrees%252Fbintree.JPG%3Bhttp%253A%252F%252Fmax.cs.kzoo.edu%252FAP%252FABMaterials%252FTrees%252FBSTLab.html%3B640%3B480

III. IMPLEMENTASI

III.1 Binary Search Tree

Seperti yang telah dijelaskan sebelumnya, binary search tree adalah pohon dimana simpul –simpul kirinya bernilai lebih kecil dari akar dan simpul-simpul di sebelah kanannya bernilai lebih besar dari akar.Sifat ini menjadikannya lebih efisien ketika digunakan untuk searching sebuah bilangan, mendelete sebuah bilangan, dan menginsert sebuah bilangan.

Ketika hendak mencari suatu nilai misalkan X di dalam binary search tree(asumsikan X pasti ada di binary search tree), kita tinggal membandingkan nilai X dengan akarnya,jika sama maka mengembalikan nilai node akar tersebut. Jika ternyata nilai X lebih kecil dari akarnya, maka lakukan rekursi terhadap upapohon sebelah kiri. Namun jika sebaliknya X lebih besar dari akar,maka lakukan rekursi terhadap upapohon sebelah kanan.

```
function SearchBST(X : integer,P : pohon)-> pohon
        If (X=Akar(P)) then                          1
                Return P                             2
        Else if (X<Akar(P)) then                     3
                SearchBSTX(X,Left(P))                4
        Else                                         5
                SearchBSTX(X,Right(P))               6
```

Sumber : DiktatStrukturData_v_041012.pdf

Pada contoh pseudocode diatas,bagian basis adalah baris 1sampai 2 dan bagian rekurens adalah dari baris 3 sampai 5.

Sedangkan untuk kasus Insert kedalam X.

```
Procedure InsSearchTree(input X: integer, input/output P
: pohon)

        if (IsTreeEmpty(P)) then
                MakeTree(X,Nil,Nil,P)
        Else1
                Depend on X,P
                X=Key(P)        :- count<-count+1
                X<Key(P)        :-InsSearchTree(X,
        Left(P)
                X>Key(P)        :-InsSearchTree(X,
        Right(P)
```

Sumber : DiktatStrukturData_v_041012.pdf

Pada procedure InsSearchTree di atas,Key digunakan untuk menunjukan nilai akar,dan count digunakan untuk menyatakan banyaknya simpul yang bernilai X. Idenya sama seperti pada searching yaitu membandingkan X dan akar(P) dan kemudian melakukan fungsi rekursi terhadap upapohon kiri atau kanan atau jika nilai X sudah ada maka tinggal menambah nilai count saja.Basis ditunjukkan pada baris ke-1 sampai baris ke-3,dimana jika ternyata nilai X belum ada maka akan dibuat sebuah pohon baru sebagai upapohon.Dan bagian rekurens ditunjukkan dari baris 4 sampai kode akhir program.

Untuk kasus delete

```
ALGORITMA
   depend on X, Key(P)
     X.Key < Key(P) : DelBTree(Left(P),X)
     X.Key > Key(P) : DelBTree(Right(P),X)
     X.Key = Key(P) : { Delete simpul ini }
                      q ← P
                      depend on q
                        isOneElmt(q)  : -
                        isUnerLeft(q) : p ← Left(q)
                        isUnerRight(q): p ← Right(q)
                        isBiner(q)    : DelNode(Left(q))
                      Dealokasi(q)
```

Sumber : DiktatStrukturData_v_041012.pdf

```
ALGORITMA
   depend on P
     Right(P) ≠ Nil : DelNode(Right(P))
     Right(P) = Nil : {P anak terkanan}
        Key(q) ← Key(P)    {salin info P ke q}
        Count(q) ← Count(P)
        q ← P              {q: yang akan dihapus}
        P ← Left(P)   {pastikan anak kiri "naik", jika ada}
```

Sumber : DiktatStrukturData_v_041012.pdf

Dari pseudocode di atas,kita melihat bahwasanya procedure utam DelBTree memanggil procedure DelNode untuk menghapus sebuah node P. Procedure DelBTRee berfungsi hanya untuk mencari posisi node P dimana Key(P)=X.Key. Basis pada procedure DelBTree adalah X.Key=Key(P),sedangkan untuk dua kasus lainnya ketika X.Key lebih besar dari atau lebih kecil dari maka dilakukan rekursi terhadap procedure DelBtree. Sedangkan basis pada procedure DelNode adalah ketika Right(P)=Nil,bagian rekurensnya adalah Right(P) /= Nil.

Dari contoh-contoh Binary Search Tree di atas, kita melihat bahwasanya dengan menggunakan sifat rekursi, maka persoalan kapan berhenti dan dan rekursi kembali menjadi sangat mudah.Ketika melakukan searching terhadap X,kita tidak perlu search semua simpul pada pohon P tapi cukup mengarahkannya ke kiri atau ke kanan saja jika X tidak sama dengan akar(P).Sehingga X semakin lama semakin dekat dengan simpul P yang diinginkan.

III.2 Fibonacci Search Tree

Aplikasi dari Fibonacci Search Tree pada kasus searching data terurut sebenarnya hampir sama dengan kasus searching pada Binary Search Tree,hanya saja Fibonacci Search Tree memiliki kelemahan ketika data yang mau diakses memiliki jangkauan yang luas atau dengan kata lain data yang mau diproses sangat banyak. Namun kelebihan dari Fibonacci Search Tree adalah ketika kecepatan elemen data yang diakses bergantung pada posisi elemen itu dimana.Misalnya ketika hendak mengakses data tape magnetik dimana kecepatan mengakses sebuah elemen pada satu titik tape bergantung dari jaraknya dari posisi kepala tape pada waktu itu.

Cara kerja dari Fibonacci Search Tree adalah sebagai berikut. .Misalkan kita memiliki struktur data array a ,banyaknya data adalah n=F(k) dimana F(k) adalah bilangan fibonacci ke k. Jika kita ingin melakukan searching suatu nilai key apakah ada di data yang kita miliki atau tidak beserta lokasi nya di data kita jika ada. Maka pertama-tama kita melakukan pengecekan terhadap a[F(k-1)].Jika sama maka program berhenti dengan sempurna(ditandai dengan return indeks data).Jika ternyata key < data ke F(k-1) dan nilai dari F(k-3) tidak sama dengan 0,maka nilai k diganti dengan k-2 dan kemudian memanggil fungsi Search kembali untuk melakukan pengecekan pada a[F(k-2)].Jika ternyata nilai F(k-3)=0 maka program dihentikan yang berarti key tidak ada di data array tersebut. Jika key > a[F(k-1)] dan nilai F(k-2) tidak sama dengan 1 maka memanggil fungsi Search kembali untuk melakukan pengecekan terhadap a[F(k-1) + F(k-3)]. Jika F(k-2)=1,maka program dihentikan yang berarti bahwa key tidak ada pada data tersebut.Pada kasus n tidak sama dengan F(k) maka dipilih F(k) paling kecil yang lebih besar dari n. Data pada indeks selisih dari F(k)-n kemudian diisi dengan data yang lebih besar dari key.

Berikut adalah algoritma Fibonacci Search Tree di C

```
if(key == a[mid]) return mid;
if(key > a[mid]) {
  if(p == 1) return -1;
  mid = mid + q;
  p = p - q;
  q = q-p;
 } else // key < a[mid] {
  if(q == 0) return -1;
  mid = mid - q;
  temp = p - q;
  p = q;
  q = temp;
}
return fibsearch(a, key, mid, p, q);
```

Sumber : http://stackoverflow.com/questions/7599479/fibonacci-search

Parameter Mid pertama-tama diinisialisasi dengan F(k-1),p diinisialisasi dengan F(k-2),q diinisialisasi dengan F(k-3).

III.3 Menara Hanoi

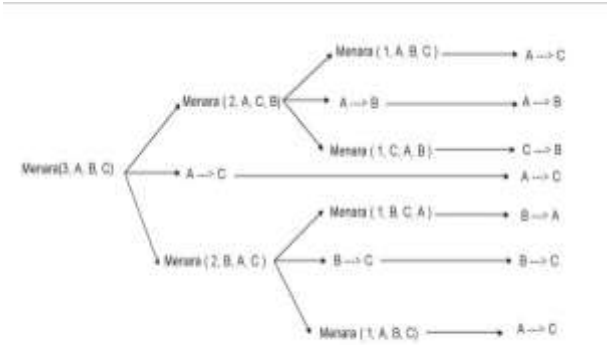
Ide umum untuk menyelesaikan teka teki menara Hanoi

dengan n buah cakram adalah memindahkan n-1 buah cakram dari tiang asal(tiang A) ke tiang persinggahan terlebih dahulu misalnya tiang B. Setelah itu kita pindahkan cakram ke n ke tiang tujuan misalnya tiang C. Kasus memindahkan n-1 buah cakram dari tiang asal(tiang A) ke tiang persinggahan(tiang B) dapat dipandang sebagai kasus memindahkan k buah cakram dari tiang asal A ke tiang tujuan B dengan bantuan tiang persinggahan C(tiang tujuan dan persinggahan ditukar posisinya yaitu antara tiang B dan C) dengan k= n-1.Demikian juga kasus memindahkan n-1 buah cakram dari tiang B ke tiang C sesudah cakram ke-n dipindahkan ke tiang C dapat dipandang sebagai kasus memindahkan k buah cakram dari tiang asal(tiang B) ke tiang tujuan(tiang C) dengan k=n=1.Kali ini kita menukar posisi tiang asal antara tiang A menjadi tiang B.Hal ini terus kita lakukan hingga n=2 buah cakram. Jadi disini berlaku sifat rekursif.

Jadi algoritmanya adalah

If (N=1) then
  Pindahkan cakram dari tiang asal ke tiang tujuan.
Else
  Hanoi(n-1,asal,tujuan,singgah),
  Pindahkan cakram ke n dari tiang asal ke tiang tujuan,
  Hanoi(n-1,singgah,asal,tujuan).

Untuk lebih jelasnya perhatikan diagram berikut.



Sumber : http://rusdyana.wordpress.com/2009/11/12/algoritma-menara-hanoi/

IV. KESIMPULAN

Rekursif dan pohon mempunyai aplikasi yang sangat banyak dan bermanfaat dalam dunia pemrograman.

DAFTAR REFERENSI

[1] Munir,rinaldi.”Diktat Kuliah IF2120 Matematika Diskrit”, edisi keempat,Program Studi Teknik Informatika STEI ITB,2006.

[2] Liem,inggriani,”Diktat Struktur Data”,Program Studi Teknik Informatika ITB.2008.

[3] Ibnu Mas’ud, “Towers Of Hanoi Beserta Aplikasinya”,belum dipublikasi.

[4] http://stackoverflow.com/questions/7599479/fibonacci-search Waktu akses : 15 Desember 2013 , 18.00 WIB.

[5] http://rusdyana.wordpress.com/2009/11/12/algoritma-menara-hanoi/ Waktu akses : 15 Desember 2013 , 21.00 WIB.

[6] http://hokageyudha.blogspot.com/2012/03/towers-of-hanoi.html Waktu akses : 16 Desember 2013 , 16.00 WIB.

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, 16 Desember 2013

Kevin
13512097