Algoritma Untuk Permainan Tower of Hanoi

William - 13508032
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if18032@students.if.itb.ac.id

Abstrak

Tower of Hanoi adalah sebuah permainan yang sudah berumur ratusan tahun. Walau kelihatannya sederhana, banyak orang yang kesulitan untuk menyelesaikan permainan ini dengan efektif, yaitu dengan jumlah langkah yang minimal. Sejak saat itu, mulailah diciptakan program bantuan untuk menyelesaikan permainan Tower of Hanoi ini. Walau ada banyak cara penyelesaian, program yang baik adalah program yang berjalan dengan algoritma yang efektif. Makalah ini akan membahas algoritma – algoritma yang digunakan, serta mencari algoritma mana yang paling efektif digunakan untuk menyelesaikan permainan ini. Keefektifan disini dihitung dari waktu yang digunakan untuk menyelesaikan permasalahan yang ada.

***Kata Kunci* – Algoritma, Efektif, Tower of hanoi**

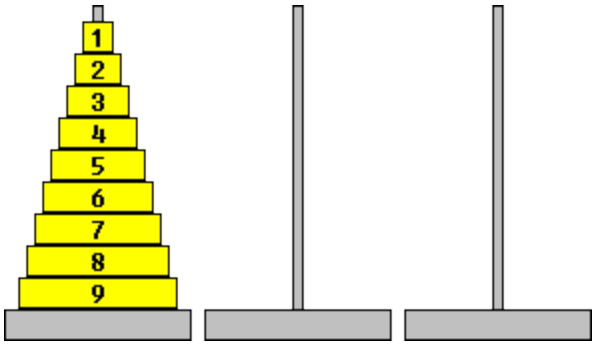
1. LATAR BELAKANG

Tower of Hanoi pertama – tama dikenal dengan nama *Tower of Brahma*, dan pertama kali diciptakan oleh Matematikawan Prancis yang bernama Édouard Lucas pada tahun 1883. *Tower of Hanoi* ini muncul pada sebuah kolom pada majalan *Scientific American* yang setelah diterjemahkan kira – kira berbunyi sebagai berikut:

"Pada sebuah kuil Brahma besar di Benares, pada sebuah alas kuningan dibawah kubah yang menandakan pusat dunia, ada 64 buah piringan yang terbuat dari emas murni yang oleh para pendeta dibawa dan dipindahkan satu demi satu diantara tiang berlian menurut aturan yang diberikan oleh Brahma yaitu bahwa tidak ada piringan yang boleh diletakkan diatas piringan yang lebih kecil. Pada awal dunia, keseluruhan 64 piringan membentuk *Tower of Brahma* pada satu tiang. Dan sekarang, proses perpindahannya masih dan sedang berlangsung. Saat piringan terakhir berada pada tempatnya, dan sekali lagi membentuk *Tower of Brahma*, tapi pada tiang yang berbeda, itulah akhir dari dunia dan semua akan kembali menjadi debu."

Sampai saat ini masih belum jelas apakah Lucas yang menciptakan legenda ini atau ia terinspirasi dari legenda ini. Namun, bila legenda ini benar, dan bila para pendeta dapat memindahkan piringan tersebut dengan kecepatan 1 perpindahan per detik, bila menggunakan jumlah perpindahan yang paling efektif (yaitu $2^{64} – 1$ perpindahan) mereka akan membutuhkan 18,446,744,073,709,551,615 detik untuk menyelesaikannya yang kira – kira setara dengan 585 miliar tahun.

Dari deskripsi yang diberikan oleh Lucas, kita akan mendapatkan sebuah permainan *puzzle* yang kira – kira berbentuk seperti ini:



Gambar 1. *Tower of Hanoi* dengan 9 buah piringan

Disini, tugas kita untuk menyelesaikan permainan ini adalah dengan memindahkan keseluruhan piringan yang ada dari tiang yang paling kiri ke tiang yang paling kanan. Namun, perpindahan yang dilakukan tidak bisa sembarangan; ada peraturan yang harus diikuti, yaitu bahwa hanya boleh memindahkan 1 piringan pada satu saat, dan piringan yang dipindahkan tersebut tidak boleh diletakkan diatas piringan yang ukurannya lebih kecil.

2. SOLUSI

Sebenarnya, bila sudah mengetahui caranya, menyelesaikan permasalahan *Tower of Hanoi* dengan langkah yang optimal tidaklah terlalu sulit.

Untuk mempermudah penjelasan, anggaplah *tower* kita hanya memiliki 4 piringan. Asumsikan kita mengetahui cara terbaik untuk melakukan perpindahan 3 piringan saja, maka kasusnya adalah akan ada 3 buah piringan di tiang B, piringan nomor 4 pada tiang A, dan tiang C yang merupakan target adalah tiang kosong. Dari posisi itu,

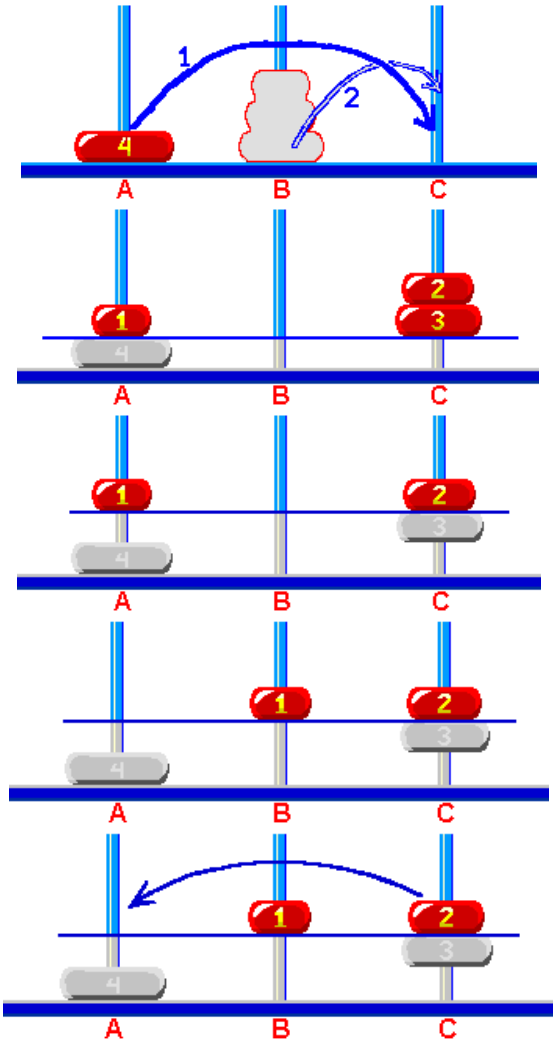
yang harus kita lakukan adalah memindahkan piringan ke 4 dari tiang A ke tiang C, kemudian, dengan suatu cara memindahkan 3 piringan yang ada pada tiang B ke tiang C.

Berarti, tujuan kita sekarang adalah memindahkan piringan ke 3 pada tiang B. Untuk melakukan itu, kita butuh tiang B yang kosong, serta piringan ke 3 pada tiang C, dan piringan 1 dan 2 tersusun rapi pada tiang A. Karena itu, sekarang tujuan kita menjadi bagaimana meletakkan piringan 2 pada tiang A.

Untuk melakukan hal tersebut, kita perlu memindahkan piringan yang lebih kecil dari piringan 2 (piringan 1) dari tiang A, yang berarti piringan 1 tersebut harus dipindahkan ke tiang B. Disini, langkahnya menjadi mudah karena piringan 1 adalah piringan paling kecil, dan tiang B sudah kosong.

Kemudian, yang perlu dilakukan hanyalah penyusunan ulang untuk menyelesaikan permainan.

Representasi grafikalnya dapat dilihat pada gambar dibawah ini[5]:



Setelah memahami cara tersebut, ada beberapa primitif yang bisa kita dapatkan untuk menyelesaikan permainan ini dengan lebih mudah, yaitu bahwa:

1. Setiap pergerakan akan diselingi oleh pergerakan piringan yang paling kecil. (Piring paling kecil akan digerakkan pada langkah ke 1,3,5,…,dst)
2. Piringan paling kecil akan mengunjungi setiap tiang secara berurutan setiap pergerakannya, dan tidak pernah kembali ke tiang sebelumnya pada pergerakan berikutnya. Hal ini juga berlaku untuk semua piringan kecuali piringan yang paling besar.
3. Pada pergerakan yang tidak melibatkan piringan paling kecil (pergerakan ke 2,4,6,…,dst) hanya ada 1 kemungkinan pergerakan
4. Bila piringan dinomori, maka piringan bernomor genap tidak akan pernah diletakkan diatas piringan genap lainnya, begitu juga untuk piringan bernomor ganjil.
5. Untuk langkah pertama, bila jumlah piringannya genap, pindahkan piringan paling kecil ke tiang B. Bila jumlah piringannya ganjil, pindahkan piringan tersebut ke tiang C. Ini merupakan turunan dari primitif nomor 4, karena alas dari setiap piringan dapat dianggap piringan ke N+1. Sehingga, pergerakan piringan pertamanya dapat ditebak.

3. Beberapa Algoritma Yang Dapat Digunakan

Setelah mengetahui bagaimana cara menyelesaikan *Tower of Hanoi*, kita dapat membangun beberapa algoritma untuk digunakan, diantaranya:

a. Algoritma Iteratif

Adalah algoritma paling sederhana untuk menyelesaikan masalah ini, dan diturunkan dari primitive yang didapat, yaitu melakuaka pergerakan secara bergantian antara piringan yang paling kecil dan yang piringan lainnya. Saat memindahkan piringan paling kecil, lakukan pergerakan kea rah yang sama, dan bila sudah mencapai ujung, lanjutkan perpindahan ke ujung lainnya. Dengan cara ini, kira – kira akan didapat pemetaan sebagai berikut:

Untuk ukuran piringan genap:

- Lakukan pergerakan dengan tiang A dan B
- Lakukan pergerakan dengan tiang A dan C
- Lakukan pergerakan dengan tiang B dan C
- Ulangi sampai selesai

Sementara, untuk ukuran piringan ganjil:

- Lakukan pergerakan dengan tiang A dan C
- Lakukan pergerakan dengan tiang A dan B
- Lakukan pergerakan dengan tiang B dan C
- Ulangi sampai selesai

Bila menggunakan cara ini, jumlah perpindahan adalah 2^n-1dengan n adalah jumlah piringan.

b. Algoritma Rekursif

Kunci dari penggunaan algoritma ini adalah pemahaman bahwa permainan ini dapat diselesaikan dengan membagi masalah ke masalah – masalah yang lebih

kecil, dan terus memecah – mecahnya ke masalah yang lebih kecil. Cara pemecahan *Tower of Hanoi* dengan pendekatan ini adalah:

1. Pindahkan N - 1 piringan teratas dari tiang A (sumber) ke tiang B (penengah) sehingga piringan N adalah satu – satunya piringan yang tersisa pada tiang A.
2. Pindahkan piringan N dari tiang A ke tiang C (tujuan).
3. Pindahkan N - 1 piringan dari rongkat B ke C sehingga semuanya terletak diatas piringan N

Kemudian, ulangi perlakuan tersebut untuk N - 1 piringan sebelumnya, sampai menemukan basis pada N = 1 yang berarti hanya lakukan 1 perpindahan dari tiang A ke C.
Pendekatan ini bisa dibentuk formulasi matematikanya dengan teori pemrograman dinamis.

Dengan cara ini, kita dapat membuat pseudocodenya, yaitu seperti dibawah ini:

```
FUNCTION Hanoi (N, A, C, B):
IF N == 0, THEN:
    move N from A to C
ELSE:
    Hanoi(N - 1, A, B, C)  //Langkah 1
    move N from A to C     //Langkah 2
    Hanoi(N - 1, B, C, A)  //Langkah 3
END IF
```

Cara ini menggunakan pendekatan bahwa:

- Bila hanya ada 1 piringan, pindahkanlah piringan tersebut dari A ke C
- Bila N > 1, piringan ke N harus dipindahkan ke tiang C, yang berarti piringan lainnya harus terurut pada tiang B. Berarti,kita harus terlebih dahulu memindahkan N-1 piringan ke tiang B untuk kemudian dipindahkan ke tiang C setelah piringan ke N dipindahkan ke tiang C
- Bila kita mengabaikan piringan ke N, berarti masalahnya adalah bagaimana cara memindahkan N-1 piringan ke tiang B, dan seterusnya
- Dengan cara ini, masalah dengan jumlah N piringan dapat direduksi menjadi N – 1 piringan, kemudian menjadi N – 2 piringan, dan seterusnya sampai N = 1.

Dengan menggunakan induksi matematik, kita dapat menemukan bahwa prosedur ini akan menggunakan jumlah pergerakan yang minimal, dan solusi yang dibentuk hanya 1 dengan jumlah pergerakan sekecil ini. Jumlah pergerakan ini sama dengan metode sebelumnya yaitu $2^n - 1$, didapat dengan menghitung bahwa langkah nomor 1 dan 3 memerlukan T_{n-1} gerakan, dan langkah 2 memerlukan 1 tambahan gerak, sehingga jumlah gerakannya adalah: $2T_{n-1} + 1$

4. ALGORITMA YANG EFEKTIF UNTUK DIGUNAKAN

Algoritma paling efektif saat ini untuk digunakan pada program permainan *Tower of Hanoi* adalah algoritma rekursif yang diciptakan oleh M. Kolar. Setelah melakukan observasi pada primitif nomor 4 pada permainan ini lebih lanjut, ditemukan lagi bahwaada paling banyak 1 piringan bernomor genap diantara semua piringan teratas pada *tower* yang tidak kosong (salah satu dari piringan teratasnya adalah piringan nomor 1).

Dengan basis itu, ada dapat kita dapatkan algoritma iterative sebagai berikut:

Langkah awal:
Piringan pertama dipindahkan ke tiang 3 bila N ganjil, dan ke tiang 2 bila N genap.
Kemudian, langkah berikutnya tergantung pada pasangan piringan yang akan dipindahkan berikutnya, yaitu:

- Bila paritynya genap, tiang tujuan pada pergerakan berikutnya akan tetap sama, dan piringan berikutnya akan dipindahkan kesana dari piringan yang tidak terlibat pada pergerakan ini (piringan ini akan diletakkan diatas piringan yang dipindahkan sebelumnya, sehingga haruslah piringan ganjil)
- Bila paritynya ganjil, perpindahan berikutnya adalah antara tiang berbeda dari tiang pada pergerakan ini, dan tujuan pergerakannya adalah untuk memindahkan piringan yang lebih kecil ke yang lebih besar.

Pada algoritma ini, asumsinya adalah dasar dari semua *tower* diberikan nomor N + 1, dan diperlakukan sebagai piringan yang lebih besar daripada piringan lainnya.

Algoritmanya dalah bahasa c adalah sebagai berikut:
(Kode ini ditulis oleh M. Kolar)

```
/* Kolar's Hanoi Tower algorithm no. 1*/

#include <stdio.h>
#include <stdlib.h>
#define PR   (void)printf(
#define PE   (void)fprintf(stderr,
#define ALLO(x)
{if((x = (int *)malloc((n+3) * sizeof(int)))
== NULL) {\ PE #x " allocation failed!\n");
exit(1);}}

main(int argc, char *argv[])
{
int i, *a, *b, *c, *p, *fr, *to, *sp, n,
n1, n2;
n = atoi(argv[1]);
n1 = n+1;
n2 = n+2;
ALLO(a)
ALLO(b)
ALLO(c)
a[0] = 1; b[0] = c[0] = n1;
a[n1] = b[n1] = c[n1] = n1;
a[n2] = 1; b[n2] = 2; c[n2] = 3;
for(i=1; i<n1; i++)
```

```
  {a[i] = i; b[i] = c[i] = 0;}
fr = a;
if(n&1){to = c; sp = b;}
else   {to = b; sp = c;}
while(c[0]>1)
{
  PR"move disc %d from %d to %d\n",
i=fr[fr[0]++], fr[n2], to[n2]);
  p=sp;
  if((to[--to[0]] = i)&1){sp=to;
    if(fr[fr[0]] > p[p[0]]){to=fr; fr=p;}
  else to=p;}
  else {sp=fr; fr=p;}
 }
}
```

Kode ini dapat dibandingkan dengan kode LLHanoi yang ditulis oleh M. C. Er, yaitu sebuah kode penyelesaian *Tower of Hanoi* secara iteratif tanpa menggunakan *loop* yang sebelumnya dianggap sebagai algoritma tercepat untuk menyelesaikan *Tower of Hanoi*. Kode LLHanoi ini adalah sebagai berikut:

```
/*  Er's LLHanoi Hanoi Tower algorithm */
#include <stdio.h>
#include <stdlib.h>
#define PR      (void)printf(
#define PE      (void)fprintf(stderr,
#define ALLO(x)
{if((x = (int *)malloc((n+2) * sizeof(int)))
== NULL) {\ PE #x " allocation failed!\n");
exit(1); }}

main(int argc, char *argv[])
{
int i, dir, *D, *s, n, n1;
n = atoi(argv[1]);
n1 = n+1;
ALLO(D)
ALLO(s)
for(i=0; i<=n1; i++){D[i] = 1; s[i] = i+1;}
dir = n&1;
for(;;)
  {
    i = s[0];
    if(i>n) break;
    PR"move disc %d from %d ", i, D[i]);
    PR"to %d\n", D[i]=(D[i]+(i&1?dir:1-
dir))%3+1);
    s[0] = 1;
    s[i-1] = s[i];
    s[i] = i+1;
  }
}
```

Bila dibandingkan performansinya, waktu pengeksekusi-annya dapat dilihat pada tabel ini:

n	LLHanoi	Algoritma no. 1
15	0.2	0.1
16	0.3	0.2
17	0.6	0.4
18	1.1	0.8
19	2.3	1.6
20	4.5	3.3
21	9.1	6.7
22	18.4	13.5

Tabel 1. Perbandingan waktu eksekusi (dalam detik) Algoritma LLHanoi dan Algoritma versi 1 Kolar

Bila dilihat algoritma 1 Kolar ini sekitar 27% lebih cepat daripada algoritma LLHanoi. Hal ini mungkin disebabkan karena LLHanoi menggunakan 2 fungsi printf, dan Kolar hanya menggunakan 1. Namun, setelah printfnya dihapus dan performansinya dibandingkan lagi, ditemukan bahwa Algoritma no. 1 Kolar tetap lebih cepat daripada LLHanoi.

Tidak sampai disana, Kolar terus mengembangkan algoritmanya sampai versi ke 4. Pada versi 2, ia melakukan perubahan menggunakan pengetahuan bahwa setelah setiap pergerakan dari piringan genap, pergerakan berikutnya adalah pergerakan piringan ganjil. Dengan cara ini, dapat dieliminasi lebih dari ¼ AND nya dan IF yang mengikutinya.

Setelah itu, Koler membangun algoritma versi 3 dengan mengembangkan algoritma versi 2 nya menggunakan prinsip 'pewarnaan' dari Schoute. Dia juga menggunakan prinsip bahwa *tower* dengan piringan teratas genap, bergerak secara siklis dengan arah B-A-C-B untuk N genap, dan dengan arah C-A-B-C untuk n ganjil. Hal ini didapatkan dari memindahkan piringan genap dari *tower* genap ke arah yang sama, atau memindahkan piringan ganjil ke arah sebaliknya (ke arah *tower* genap). Efektifitas dapat meningkat karena posisi *tower* genapnya dapat diketahui lebih dahulu.

Tidak berhenti disana, Koler kembali mengembangkan algoritma versi 4 dari versi 3 nya, menggunakan sifat bahwa piringan paling kecil akan dipindahkan pada setiap pergerakan ganjil. Algoritma ini, walau menambahkan beberapa baris kode pada algoritma versi 3 nya, dapat mengeliminasi ½ dari IF - nya.

Algoritma versi 4 Koler adalah sebagai berikut:

```
/* Kolar's Hanoi Tower algorithm no. 4 */

#include <stdio.h>
#include <stdlib.h>
#define PR      (void)printf(
#define PE      (void)fprintf(stderr,
#define ALLO(x)
{ if((x = (int *)malloc((n+3 * sizeof(int)))
== NULL) {\ PE #x " allocation failed!\n");
exit(1); }}
```

```
main(int argc, char *argv[])

{
 int i, *a, *b, *c, *p, *o1, *o2, *e, n, n1;

 n = atoi(argv[1]);
 n1 = n+1;
 ALLO(a)
 ALLO(b)
 ALLO(c)

 a[0] = 1; b[0] = c[0] = n1;
 a[n1] = n1; b[n1] = n+2; c[n1] = n+3;
 for(i=1; i<n1; i++)
{a[i] = i; b[i] = c[i] = 0;}

 o1 = a;
 if(n&1) { o2 = b; e = c; }
 else    { o2 = c; e = b; }

e[--(*e)] = 1; (*o1)++;
 /* (call to) tower visualization code */
 p = e; e = o1; o1 = o2; o2 = p;

 while(*c>1)
{
if(o1[*o1] > e[*e])
  o1[--(*o1)] = e[(*e)++];
else
  e[--(*e)] = o1[(*o1)++];
  /* (call to) tower visualization code */
  p = e; e = o1; o1 = o2; o2 = p;
  e[--(*e)] = 1; (*o1)++;
  /* (call to) tower visualization code */
  p = e; e = o1; o1 = o2; o2 = p;
 }
}
```

Pointer e menunjuk pada *tower* genap, o1 pada *tower* ganjil yang akan menjadi genap pada pergerakan ini, dan o2 adalah *tower* ganjil lainnya yang masih tetap ganjil setelah pergerakan. Inisialisasi dari e dan o2 bergantung pada parity dari N.

Kecepatan dari setiap algoritma dapat dilihat pada tabel 2. Perlu diketahui bahwa kecepatan disini adalah kecepatan program berjalan tanpa menampilkan visualisasi *tower*, jadi yang dihitung hanyalah kecepatan proses algoritmanya saja.

N	LLHanoi	Alg 1	Alg 2	Alg 3	Alg 4
23	0.7	0.5	0.4	0.4	0.3
24	1.4	0.9	0.8	0.8	0.6
25	2.8	1.8	1.7	1.5	1.3
26	5.7	3.7	3.3	3.1	2.6
27	11.4	7.3	6.7	6.2	5.1
28	22.8	14.7	13.4	12.5	10.3
29	45.7	29.5	26.9	25.0	20.6

Tabel 2. Perbandingan kecepatan pemrosesan setiap algoritma (dalam detik) tanpa melakukan visualisasi

Sedangkan, bila dilakukan visualisasi sederhana, berikut adalah tabel perbandingan kecepatannya:

N	LLHanoi	Alg 1	Alg 4
13	0.2	0.8	0.2
14	0.4	0.4	0.4
15	0.8	0.8	0.8
16	1.7	1.7	1.7
17	3.6	3.6	3.5
18	7.7	7.6	7.6
19	16.8	16.1	15.9
20	33.9	33.7	33.6

Tabel 3. Perbandingan kecepatan (dalam detik) antara algoritma LLHanoi dengan Algoritma 1 dan 4 Koler

5. KESIMPULAN

Ada banyak algoritma yang dapat digunakan untuk menyelesaikan permainan *Tower of Hanoi* dari algoritma iteratif yang sederhana sampai algoritma rekursif. Namun, algoritma yang paling efektif tetap adalah algoritma yang standar dan sederhana yang tidak menggunakan banyak fungsi IF, AND, dan sebagainya. Permasalahan yang ada hanyalah bagaimana kita mengubah suatu algoritma tersebut menjadi algoritma yang sederhana namun tetap dapat menyelesaikan permasalahan yang ada.

Dalam mencari algoritma yang efektif untuk menyelesaikan suatu masalah, studi mengenai algoritma sangatlah penting untuk memahami dan mencari pendekatan yang tepat untuk mencapai solusi yang paling optimal. Namun tidak hanya itu, hal yang tidak kalah pentingnya yang seringkali juga dilupakan adalah pemahaman mengenai soal / masalah yang diberikan.

Seperti dapat dilihat pada algoritma – algoritma yang dibangun oleh Koler, ia berhasil "mengalahkan" algoritma LLHanoi yang sampai saat itu dianggap sebagai algoritma tercepat untuk menyelesaikan masalah *Tower of Hanoi*. Kuncinya melakukan itu tidaklah hanya dengan mengerti algoritma dan melakukan pemrograman, namun juga kemengertian dia akan masalah yang ada sehingga dia dapat memanfaat sifat – sifat yang tidak akan dapat terlihat jika tidak melakukan observasi dan studi masalah dengan baik. Ia kemudian mengaplikasikan sifat – sifat yang ditemukannya tersebut dan pada akhirnya menemukan algoritma yang lebih cepat daripada LLHanoi.

Pencapaian tersebut tidak akan terjadi apabila seseorang hanya memandang suatu masalah dari 1 sisi saja, dan tidak mau melihat masalah itu dari sisi lain untuk mencari solusi alternatif menggunakan sifat – sifat "tersembunyi" yang mungkin dimiliki oleh masalah tersebut.

6. REFERENSI

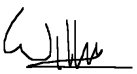
[1] http://ece.uprm.edu/cecord/crc/crc99/papers/heber.pdf
[2] http://en.wikipedia.org/wiki/Tower_of_Hanoi
[3] http://hanoitower.mkolar.org/algo.html
[4] http://stackoverflow.com/questions/1223305/tower-of-hanoi-recursive-algorithm
[5] http://wipos.p.lodz.pl/zylla/games/hanoi-ex.html
[6] http://www.cs.cmu.edu/~cburch/survey/recurse/hanoiimpl.html
[7] http://www.cut-the-knot.org/recurrence/hanoi.shtml
[8] http://www.jaapsch.net/puzzles/hanoi.htm
[9] http://www.towerofhanoi.net/
[10] M.C. Err, *A loop less approach for constructing a fastest algorithm for the Towers of Hanoi problem*, Internet. J. Computer Math., 1986, Vol. 20
[11] Munir, Rinaldi, "Diktat Kuliah IF2251 Strategi Algoritmik", Program Studi Teknik Informatika, ITB, 2007.

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, 8 Desember 2010

ttd



William
13508032