Perbandingan Penerapan Algoritma Greedy dengan Program Dinamis untuk Penyelesaian Menara Hanoi

Karina Novita Suryani 13508048
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung40132, Indonesia
if18048@students.if.itb.ac.id

***Abstrak*—Makalah ini menceritakan tentang permasalahan menara hanoi. Menara hanoi adalah sebuah permainan yang terdiri dari 3 menara dan n disk di mana disk – disk tersebut tersusun dari mulai yang terbesar (paling bawah) hingga terkecil. Permasalahan yang terjadi adalah bagaimana memindahkan n disk tersebut dari menara satu ke menara tiga dengan keterurutan seperti sebelumnya, di mana hanya ada satu disk yang bisa dipinadahkan tiap waktu, hanya bisa memindahka disk paling atas, dan tidak boleh menaruh disk yang lebih besar di atas disk yang lebih kecil**

Terdapat banyak pendekatan untuk penyelesaia masalah ini, antara lain bisa menggunakan algoritma greedy atau menggunakan program dinamis. Pada makalah ini akan dibandingkan waktu pengerjaan untuk kedua pendekatan solusi dari masalah.

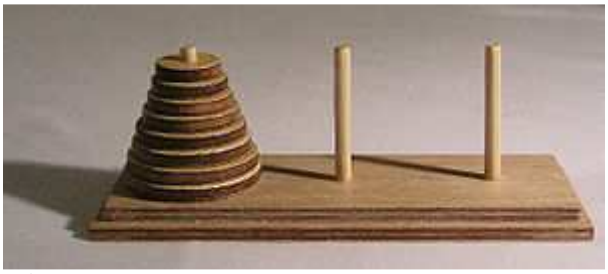
***Kata Kunci*—pencocokan string, program dinamis, *edit distance*, tak eksak, koreksi, ejaan.**

I. PENDAHULUAN

1.1 Menara Hanoi

Pada suatu hari, di vietnam, seorang pendeta dari sebuah kuil harus memindahkan 64 piringan emas suci dari satu berlian ke berlian lain yang disusun dari piringan paling besar (terbawah) dari piringan paling kecil. Piringan ini sangat rapuh sehingga hanya bisa dibawa satu per satu. Selain itu, piringan lebih besar tidak boleh ditaruh diatas piringan yang lebih kecil. Terdapat berlian lain di antara berlian tujuan dan berlian awal.

Saat Pendeta telah selesai memindahkan 64 piringan tersebut, maka legenda mengatakan bahwa dunia akan berakhir. Untuk itu, kita harus mengetahui berapa lama puzzle ini akan selesai dan berapa banyak langkah yang dibutuhkan untuk mencapai solusi tersebut?



Gambar 1 Model dari Menara Hanoi yang memiliki 8 piringan

Puzzle ini dikenalkan olen N Claus (de Siam), seorang professor dari Universitas Li-Sou-Louis, dan anagram Edouard Lucas (D'Ameins), seorang professor dari Lyce Saint-Louis.Tidak jelas benar apakah Lucas menemukan legenda ini atau terinspirasi olehnya. Bila legenda ini benar, dan pendeta itu bisa memindahkan satu cakram tiap detik, menggunakan pemindahan paling sedikit, maka akan memakan waktu $2^{64}-1$ detik atau kurang lebih 584,582 milyar tahun.

Permainan Menara Hanoi sering digunakan dalam penelitian psikologis dalam hal pemecahan masalah. Selain itu, juga sering digunakan dalam pengajaran algorima rekursif bagi pelajar pemrograman. Permainan ini juga digunakan sebagai ujian ingatan oleh ahli psikolog syaraf dalam berupaya mengevaluasi amnesia. Pendertia Amnesia menunjukkan kemajuan saat berlatih dengan puzzle menara hanoi.

Selain itu, menara hanoi juga digunakan untuk mengetest kemampuan planning seseorang secara prikologi. Planning adalah kemampuan untuk memikirkan langkah-langkah yang harus digunakan untuk mencapai tujuan tertentu. Para ilmuan telah menggunakan metode ini untuk menyelidiki proses kognitif dan saraf yang terlibat dalam perencanaan selama hampir 20 tahun. Studi kami menunjukkan bahwa pasien dengan kerusakan lobus frontal memiliki kemampuan perencanaan yang buruk, membutuhkan lebih banyak bergerak. Perilaku 'pasien lebih impulsif, cenderung untuk memulai tugas tanpa memikirkan solusi yang tepat. Ketika diminta untukmenghitung jumlah langkah yang diperlukan untuk berhasil menyelesaikan teka-teki perencanaan (menghilangkan sifat impulsif mereka untuk memulai tugas) waktu perencanaan pada pasien ini jauh lebih besar daripada yang diamati pada sukarelawan sehat (Owen et al, 1995.). Hal ini sangat menunjukkan bahwa daerah-daerah frontal dari otak mendasari kemampuan untuk merencanakan sesuatu.

.

1.2 Algoritma Greedy

Algoritma Greedy adalah algoritma yang populer untuk memcahkan persoalan mengenai optimasi. Pada persoalan optimasi biasanya kita diberi sekumpulan kendala dan mencari solusi yang paling optiman. Alggoritma ini sederhana dan *straightforward*. Prinsip

yang digunakan greedy adalah *"take what you can, get now!"*.

Algoritma Greedy membentuk solusi langkah per langkah. Terdapat banyak pilihan yang perlu dieksplorasi pada setiap langkah solusi. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Keputusan yang telah diambil tidak bisa dirubah di keputusan selanjutnya.

Pendekatan yang digunakan dalam algoritma greedy adalah membuat pilihan yang "tampaknya" memberikan pilihan terbaik, dengan membuat optimum lokal yang harapannya juga mengarah ke solusi optimum global.

Persoalan optimasi dalam konteks algoritma greedy disusun oleh elemen sebagai berikut :

1. Himpunan kandidat,C
   Himpunan ini berisi elemen-elemen pembentuk solusi.
2. Himpunan solusi, S
   Himpunan ini berisi kandidat-kandidat yang terpilih menjadi solusi dari permasalahan. Himpunan solusi adalah bagian dari himpunan kandidat.
3. Fungsi seleksi
   Fungsi ini adalah fungsi yang digunakan pada setiap langkah memilih kandidat yang paling memenuhi atau memungkinkan untuk mencapai solusi optimal. Biasanya kandidat di-*assign* sebuah nilai numerik, lalu fungsi seleksi menyeleksi kandidat-kandidat yang punya nilai terbesar atau terkecil.
4. Fungsi kelayakan
   Fungsi ini digunakan untuk memeriksa apakah kandidat tidak melanggar batasan yang sudah ditentukan
5. Fungsi Obyektif
   Fungsi yang memaksimumkan atau yang meminimumkan nilai solusi.

Algoritma Greedy memiliki skema penyelesaian umum yang sama. Skema algoritma greedy dapat kita sebagai berikut.

```
function greedy(input C: himpunan_kandidat)→ himpunan_kandidat
{ Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy
  Masukan: himpunan kandidat C
  Keluaran: himpunan solusi yang bertipe himpunan_kandidat
}
Deklarasi
    x : kandidat
    S : himpunan_kandidat

Algoritma:
    S ← {}   { inisialisasi S dengan kosong }
    while (not SOLUSI(S)) and (C ≠ {} ) do
      x ← SELEKSI(C)        { pilih sebuah kandidat dari C}
      C ← C - {x}           { elemen himpunan kandidat berkurang satu }
      if LAYAK(S ∪ {x}) then
         S ← S ∪ {x}
      endif
    endwhile
    {SOLUSI(S) or C = {} }

    if SOLUSI(S) then
      return S
    else
      write('tidak ada solusi')
    endif
```

Gambar 2 Skema Umum Penyelesaian dengan Algoritma Greedy

Algoritma Greedy tidak selalu memberikan hasil optimum yang berlaku untuk optimum global. Hal ini dikarenakan algoritma Greedy tidak mencari keseluruh alternatif solusi dan pemilihan fungsi seleksi yang berbeda. Akan tetapi algoritma ini jauh lebih cepat dibandingkan exhaustive search karena telah mempertimbangkan fungsi seleksi.

1.3 Program Dinamis

Program Dinamis (*dynamic programming*) adalah metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan langkah (step) atau tahapan (stage) sehingga solusi dari peresoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan. Pada penyelesaian persoalan dengan metode ini:

1. Terdapat sejumlah berhingga pilihan yang mungkin,
2. Solusi pada setiap tahap dibangun dari hasil solusi tahap sebelumnya,
3. Kita menggunakan persyaratan optimasi dan kendala untuk membatasi sejumlah pilihan yang harus dipertimbangkan pada suatu tahap.

Cara penyelesaian dengan metode program dinamis mengingatkan kita pada metode greedy, karena metode greedy juga membentuk solusi secara bertahap. Pada metode greedy, kita membuat keputusan pada setiap tahap dengan cara mengambil pilihan yang paling memenuhi ukuran optimasi yang digunakan.

Pada Algoritma Greedy, pengambilan keputusan pada setiap tahap hanya berdasarkan informasi lokal saja dan tidak pernah salah. Dengan ini pengambilan keputusan pada setiap langkah greedy tidak pernah mempertimbangkan lebih jauh apakah pilihan tersebut pada langkah-langkah selanjutnya merupakan pilihan yang tepat.

Pada metode penyelesaian *exhaustive search*, kita mengiterasi seluruh kemungkinan solusi untuk mencari solusi sebenarnya. Akan tetapi cara ini sangatlah tidak mangkus. Oleh karena itu, kita bisa mengatasi banyaknya iterasi yang dilakukan untuk mencari solusi dengan cara menggunakan Program Dinamis.

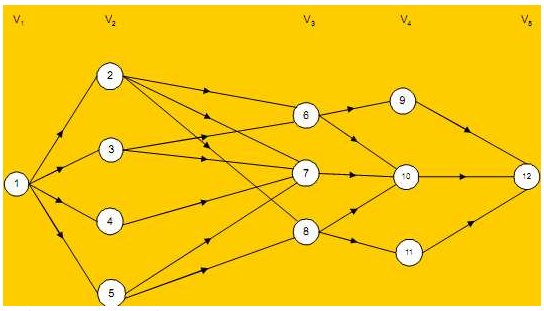
Program Dinamis memiliki prinsip optimalitas. Prinsip ini berbungi *jika solusi total optimal, maka bagian solusi sampai tahap ke-k juga optimal*. Prinsip optimalitas juga berarti bahwa jika kita bekerja dari tahap k ke tahap k+1, kita dapat menggunakan hasil optimal dari tahap k tanpa harus kambali ke awal. Langkah ini membuat kita mengurangi jumlah iterasi dan mengambil langkah yang lebih mengarah ke solusi. Jika pada setiap tahap kita menghitug ongkos (cost), maka dapat dirumuskan

Ongkos pada tahap k + 1 = (ongkos yang dihasilkan pada tahap k) + (ongkos dari tahap k ke tahap k + 1).

Berbeda dengan Greedy, pada metode Greedy hanya ada satu rangkaian keputusn yang pernah dihasilkan. Sedangan pada program dinamis, dihasilkan serangkaian keputusan optimal yang saling berkaitan dan membentuk solusi yang paling optimal.

Program dinamis dapat diterapkan pada persoalan yang memiliki karakteristik sebagai berikut:

1. Persoalan dapat dibagi menjadi beberapa tahap (*stage*), yang pada setiap tahap hanya diambil satu keputusan.
2. Masing-masing tahap terdiri dari sejumlah status (*state*) yang berhubungan dengan tahap tersebut. Secara umum, status merupakan bermacam kemungkinan masukan yang ada pada tahap tersebut. Gambar berikut memperlihatkan perbedaan antara tahap dan status diberikan pada graf multitahap (*multistage graph*). Tiap simpul di dalam graf tersebut meyatakan status, sedangkana V1, V2, ... menyatakaan tahap.



Gambar 1.2.1: Graf yang menyatakan tahap dan status

3. Hasil dari keputusan yang diambil pada setiap tahap ditransformasikan dari status yang bersangkutan ke status berikutnya pada tahap berikutnya.
4. Ongkos (*cost*) pada suatu tahap meningkat secara teratur (*steadily*) dengan bertambahnya jumlah tahapan.
5. Ongkos pada suatu tahap bergantung pada ongkos tahap-tahap yang sudah berjalan dan ongkos pada tahap tersebut.
6. Keputusan terbaik pada suatu tahap bersifat independen terhadap keputusan yang dilakukan pada tahap sebelumnya.
7. Adanya hubungan rekursif yang mengidentifikasikan keputusan terbaik untuk setiap status pada tahap k memberikan keputusan terbaik untuk setiap status pada tahap k + 1.
8. Prinsip optimalitas berlaku pada persoalan tersebut.

Solusi yang dihasilkan oleh program dinamis dapat lebih dari satu buah.

II. METODE

2.1 Permasalahan Menara Hanoi

Penyelesaian Menara hanoi memiliki 4 parameter yaitu H(n,s,d,h). N adalah nomor piringan, s adalah berlian awal, d adalah berlian tujuan, dan h adalah berlian yang menjadi bantuan.

Memindahkan piringan dari berlian awal ke berlian tujuan mengggunakan berlian antuan. Pada masalah ini, kita menyebutkan berlian tujuan sebagai a, berlian tujuan sebagai b, dan berlian bantuan sebagai c, sehingga kita bisa membuat notasi Hmin(n,a,b,c).

Untuk memudahkan, kita akan membuat suatu notasi. Notasi itu adalah sebagai berikut.

- a→b : memindahkan suatu piringan dari a ke b
- a >> b : memindahkan piringan paling atas dari a ke b

Utuk mencari solusi dengan langkah minimum, piringan paling besar dipindahkan dari a ke b dengan minimum langkah yaitu a→b. Hal ini mengharuskan sisa piringan yang lain sudah dialihkan dari berlian a ke berlian c (a >> c). Setelah itu, sisa piringan dialihkan legi dari c ke b (c >> b).

Dari penjelasan di atas, kita bisa menemuka langkah untuk penyelesaian secara rekursif yaitu,

Saat n = 1, maka Hmin(n,a,b,c) = a → b
Saat n > 1, maka Hmin (n,a,b,c) = Hmin(n-1,a,c,b); a→b; Hmin(n-1,c,b,a)

Apabila M(n) adalah jumlah untuk penyelesaian masalah hanoi dengan jumlah piringan sama dengan n, maka kita bisa menyimpulkan seperti di bawah.

M(1) = 1;
$M(n) = 2M(n-1)+1 = 2(2M(n-2)+1)+1 = 2^2M(n-2)+2+1$
....
$M(n) = 2^{n-1}M(1)+2^{n-2}+. . .+2^2+2+1$
$M(n)= 2^{n-1}$

Dari persamaan diatas, jumlah maksimum untuk penyelesaian menara hanoi dengan langkah minimum adalah 2^n-1 langkah.

2.2 Penyelesaian dengan Algoritma Greedy

Untuk penyelesaian dengan algoritma greedy, kita mengaplikasikan keputusan greedy untuk men-*traverse* pohon BFS. Kita menggunakan greedy untuk memilih dua solusi (anak) yang paling minimum.

Pada pengaplikasiannya, kita butuh merekam semua pergerakannya. Kita coba untuk mengodekan a→b, a→c, b→a, c→b, a→c, b→a dengan 0,1,2,3,4,5. Untuk permasalahan 64 piringan, maka kita harus menyimpan pergerakan dalam 2^{64}-1 digit!

Oleh karena itu, dalam algoritma ini kita merekam dalam array sebesar q[0 ..(2^n-1)-1]. Q menyimpan data mengenai s,d,h, l dan r untuk mencatat segmen pergerakan saat itu.

Setiap pergerakan, algoritma greedy memulai dari elemen tengah dari array q. Algoritma ini memilih 2 sub-problem untuk ditelusuri yaitu dari elemen pertama ke elemen tengah dan elemen tengah ke elemen akhir.

Berikut pseudocode untuk algoritma greedy untuk penyelesaian menara hanoi.

```
void P greedy min(int n, char a, char
b, char c, int* langkahs, item* q,
int(*code)[3])
{
```

```
  int min_step_nr =(1<<n){1;
  int first = 0;
  q[0].s = a; q[0].d = b; q[0].h = c;
  q[0].l = 0; q[0].r = min_step_nr- 1;
  int last = 1;
  int m;
  while (first < last)
  {
    m = (q[first].l + q[first].r)>> 1;
    langkahs[m]       =code[q[first].s-
'a'][q[first].d-'a'];
      if (q[first].l < q[first].r)
      {
      q[last].s = q[first].s;
      q[last].d = q[first].h;
      q[last].h = q[first].d;
      q[last].l = q[first].l;
      q[last].r = m - 1;
      ++last;
      q[last].s = q[first].h;
      q[last].d = q[first].d;
      q[last].h = q[first].s;
      q[last].l = m + 1;
      q[last].r =  q[first].r;
      ++last;
      }
    ++first;
    }
  for(i = 0; i < min step nr; ++i)
  printf("%d, ", langkahs[i]);
}
```

2.3 Penyelesaian dengan Program Dinamis

Seperti yang dijelaskan sebelumnya, program dinamis adalah membuat solusi-solusinya dari solusi sebelumnya dengan prinsip optimalitas. Untuk permasalahan ini, sub-problem Hmin(k,c,a,b) yang didapat dari solusi optimal dari Hmin(k,a,b,c) sub problem dengan menggani huruf a dengan huruf c, huruf b dengan huruf a, dan huruf c dengan huruf a.

Kesulitan dari program dinamis adalah susah untuk memodelkan sub-problem dari problem yang akan diselesaikan. Jika dalam program dinamis, solusi optimal dari Hmin(3,a,b,c) didapat dari solusi optimal dari Hmin(2,a,b,c) dan Hmin(2,c,b,a).

Sangat sulit untuk membuat algoritma yang iteratif yang menjelaskan tentang alur dari dynamic programming. Oleh karena itu, lebih mudah memodelkan program dinamis menjadi algoritma yang rekursif.

Akan tetapi, saat kita melakukan algoritma secara rekursif seperti biasa, menghasilkan hasil yang tidak efisien dari divide dan conquer. Mengapa? Karena standar dari program dinamis adalah mengelompokkan masalah menjadi sub-problem yang terkadangan berpotongan dan dengan pendekatan divide and conquer, menghasilkan hasil yang berulang untuk sub-problem yang sama. Oleh karena itu digunakan *memoization. Memoization* adalah salah satu teknik dimana hasil yang sudah dihasilkan sebelumnya disimpan dan saat algoritma rekursif bertemu subproblem yang sama, dengan sangat mudah solusi tadi bisa diperoleh tanpa harus mengkomputasi ulang.

Akan tetapi, terdapat kelemahan. Mengatur *memoization* seperti menyimpan, mendapatkan hasil, mengkomputasi dari solusi lain, dan printing memiliki langkah yag eksponensial yang memiliki kompleksitas waktu yang sama seperti menciptakan memoization itu sendiri. Oleh karena itu, teknik ini tidak mengurangi kompleksitas waktu dari algoritma,

Untuk permasalahan ini, terdapat dua sub-problem yaitu Hmin(k,a,b,c) dan Hmin(k,a,c,b). Untuk sub problem Hmin(k,a,b,c), perpindahan piringan terbesar adalah a→b(0). Lalu, setelah solusi optimal dari anakdari sub-problem kiri disimpan yaitu Hmin(k-1,a,c,b), maka langkah optimum untuk anak dari sub-problem kanan (Hmin(k - 1; c; b; a)) didapatkan dengan cara mengganti huruf a dengan huruf c, huruf b dengan huruf a, dan huruf c dengan huruf b.

Dari pergantian huruf di atas, dapat didapatka pergantian pergerakan yaitu langkah 0 dengan langkah 4, langkah 1 dengan langkah 3, langkah 2 dengan langkah 0, langkah 3 denga langkah 5, langkah 4 dengan langkah 2 dan langkah 5 dengan langkah 1.

Sedangkan untuk subproblem (Hmin(k; a; c; b)) pergerakan piringan terbesar adalah a→c(1). Sehingga pergantiannya yaitu langkah 0 dengan langkah 2, langkah 1 dengan langkah 5, langkah 2 dengan langkah 4, langkah 3 dengan langkah 1, langkah 4 dengan langkah 0, langkah 5 dengan langkah 3.

Procedure P memoization min mengaplikasika teknik memoization yaitu menyediakan array langkahchanging yaitu untuk mengetahui pergantian langkah untuk setiap subproblem.

```
void P_memozation_min(int k, char s,
char d, char h, int n, int *langkahs)
{
  int i, p, pattern;
  pattern = ((n+k)&1);
  if(k==1)
    langkahs[0]=pattern;
  else
  {
  P_memozation_min(k-1, s, h, d, n,
langkahs);
  p=(1<<(k-1))-1;   //the number of
langkahs in the son-sub-problems
langkahs[p]=pattern; //the langkah of
the largest disc
  for(i=0;i<p;++i)   //generating the
right-son-solution
  langkahs[p+1+i]       =       langkah
changing[pattern][langkahs[i]];
  }
}
```

III. PERBANDINGAN

Setelah kita ulas diatas, kita akan membandingkan waktu yang dibutuhkan untuk menyelesaikan permasalahan dalam mikrodetik. Kita menguji untuk k = 1,2,3,4,5,6,7,8,9,10,15, dan 20. Berikut tabel nya.

Jumlah Piringan	Greedy	Program Dinamis
1	1	1
2	1	1
3	1	1
4	2	2
5	3	2
6	4	2
7	8	3
8	15	4
9	30	5
10	63	10
15	2071	250
20	70112	10511

IV. KESIMPULAN

Prinsip utama dari penyelesaian masalah menara hanoi adalah greedy yaitu memindahkan piringan terbesar terlebih dahulu. Akan tetapi solusi optimal yang dihasilkan dibuat dengan cara pikir program dinamis. Algoritma greedy menghasilkan haya satu keputusan. Sedangkan program dinamis menghasilkan keputusan dari solusi optimal dari keputusan sebelumnya.

REFERENSI

Munir, R. 2009. "Diktat Kuliah IF 3051 Strategi Algoritma", Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

http://www.cambridgebrainsciences.com/browse/planning/test/hampshire-tree-task, 5 Desember 2010 17:30

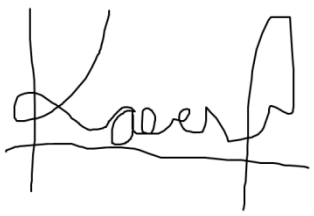
http://www.psychology.mcmaster.ca/3vv3/chapter9.htm 5 Desember 2010 17:24

http://id.wikipedia.org/wiki/Menara_Hanoi 5 Desember 2010 17:14.

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, 29 April 2010



Karina Novita Suryani – 13508048