Modifikasi Algoritma Pencarian *Decrease and Conquer* pada Akinator

Kania Azrina 13510058[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13510058@std.stei.itb.ac.id

***Abstrak*— Para *programmer* tertantang untuk membuat algoritma sehingga *software* yang menggunakan algoritma tersebut seolah-olah memiliki kemampuan *mind-reading*. *Decrease and conquer* adalah metode desain algoritma dengan mereduksi persoalan menjadi beberapa sub-persoalan yang lebih kecil, tetapi selanjutnya hanya memproses satu sub-persoalan saja. Akinator adalah sebuah game berbasis web yang dapat menebak tokoh yang ada pada pemikiran penggunanya. Analisis solusi dari masalah ini berpusat pada pemilihan pertanyaan, pilihan jawaban dan pemilihan solusi. Ada faktor-faktor eksternal yang dapat membuat program tidak lebih efektif. Walaupun program Akinator terlihat seolah-olah dapat membaca pikiran manusia, hal ini tidak lepas dari strategi algoritma dan integritas dari basis data yang ada di baliknya**

***Indeks*—*Psychic, Decrease and Conquer*, Akinator, Strategi Algoritma**

I. PENDAHULUAN

Psychic atau dalam bahasa Indonesia lebih dikenal dengan sebutan cenayang merupakan orang-orang yang dapat merasakan sesuatu yang tidak dapat dirasakan oleh panca indera biasa dengan menggunakan *extrasensory perception* (ESP) atau indera keenam. Bentuk dari indera keenam ini pun bermacam-macam, seperti menggerakan sesuatu tanpa menyentuhnya (*psychokinesis*), rmelihat tembus pandang / sangat jauh (remote viewing), membaca sesuatu dari orang lain (*psychometry*), dan lain-lain.

Karena kemampuan-kemampuan diatas yang terdengar 'super' dan cenderung menentang logika, banyak orang yang masih meragukan kebenaran dari indera keenam ini. Bagi masyarakat yang percaya, banyak isu-isu yang beredar, misalnya, ada sumber yang mengatakan bahwa kemampuan tersebut merupakan kemampuan bawaan dari lahir. Selain itu, ada juga pendapat bahwa manusia biasa dapat memiliki kemampuan tersebut dengan serangkaian latihan-latihan khusus.

Hal ini menjadi inspirasi bagi para *programmer* untuk membuat algoritma sehingga *software* yang menggunakan algoritma tersebut seolah-olah memiliki kemampuan yang setara dengan *psychic,* dalam hal ini adalah kemampuan *mind reading* atau membaca pikiran manusia. Adalah Akinator, sebuah program berbasis web yang dapat menebak apa yang ada di dalam pikiran *user* setelah memberikan sejumlah pertanyaan.

Bagaimana cara algoritma dibaliknya sehingga Akinator memiliki kemampuan fungsional yang mendekati *mind reading* menjadi rahasia utama dari para *develope*rnya. Banyak teori-teori yang bermunculan, tetapi tidak ada satupun yang dapat membuktikan keabsahannya, karena selain algoritma, basis data yang digunakan akinator tidak dibuka ke publik.

Dalam makalah ini, penulis akan mencoba melakukan modifikasi pada algoritma *decrease and conquer* untuk menyelesaikan persoalan Akinator yang merupakan pengembangan dari permainan yang pernah populer pada era '90-an, yaitu permainan *20 questions*. Walaupun begitu, kebenaran dari analisis pemecahan masalah yang ditemukan terlihat tidak dapat dibuktikan pada program sebenarnya.

II. DASAR TEORI

A. Algoritma Divide and Conquer.

Ide utama dari algoritma ini adalah membagi masalah menajdi beberapa bagian, menyelasaikan masalah per bagian tersebut dan menggabungkannya kembali. Ada tida tahapan dalam algoritma ini, yaitu :

a) *Divide*: membagi masalah menjadi beberapa sub-masalah yang memiliki kemiripan dengan masalah semula. Keluaran dari tahapan ini adalah sub masalah yang ukurannya lebih kecil.
b) *Conquer*: memecahkan atau menyelesaikan permasalahan pada masing-masing sub-masalah secara rekursif.
c) *Combine:* mengabungkan solusi dari masing-masing sub masalah sehingga membentuk solusi masalah semula.

Obyek permasalahan yang dibagi adalah masukan/input/*instances* yang berukuran *n,* seperti:

- Tabel (larik)
- Matriks
- Eksponen

```
procedure DIVIDE_and_CONQUER(input n : integer)
{ Menyelesaikan masalah dengan algoritma D-and-C.
  Masukan: masukan yang berukuran n
  Keluaran: solusi dari masalah semula
}
Deklarasi
     r, k : integer
Algoritma
  if n ≤ n0 then  {ukuran masalah sudah cukup kecil }
     SOLVE upa-masalah yang berukuran n ini
  else
     Bagi menjadi r upa-masalah, masing-masing berukuran n/k
     for masing-masing dari r upa-masalah do
        DIVIDE_and_CONQUER(n/k)
     endfor
     COMBINE solusi dari r upa-masalah menjadi solusi masalah semula }
  endif
```

Gambar 2-1 Skema Umum Algoritma Divide and Conquer. Sumber : Bahan Kuliah IF3051 Strategi Algoritma.

Jika pembagian selalu menghasilkan dua upa-masalah yang berukuran sama, maka kompleksitas waktu dari algoritmanya adalah :

$$T(n) = \begin{cases} g(n) & ,n \le n_0 \\ 2T(n/2) + f(n) & ,n > n_0 \end{cases}$$

Algoritma ini juga dapat menyelesaikan berbagai macam persoalan, diantaranya :

- Persoalan Minimum dan Maksimum (MinMaks)
- Mencari Pasangan Titik yang Jaraknya Terdekat (*Closest Pair*)
- *Sorting* atau pengurutan, dengan metode-metode:
  - *Merge Sort*
  - *Insertion Sort*
  - *Quick Sort*
  - *Selection Sort*
- Persoalan Pemasangan Ubin
- Perpangkatan a^n
- Perkalian matriks dengan menggunakan metode :
  - *Default Divide and Conquer*
  - *Strassen*
- Perkalian dua buah bilangan bulat yang besar dengan menggunakan metode :
  - *Default Divide and Conquer*
  - *A. A. Karatsuba*

B. Algoritma Decrease and Conquer

Strategi algoritma ini memiliki cara dengan mereduksi persoalan menjadi beberapa sub-persoalan yang lebih kecil. Perbedaannya dengan *divide and conquer* adalah metode ini tidak memproses semua sub-persoalan dan menggabung semua solusi setiap sub-persoalan.

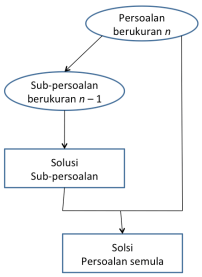
Decrease and conquer terdiri dari dua tahapan:

1. *Decrease*: mereduksi persoalan menjadi beberapa persoalan yang lebih kecil (biasanya dua sub-persoalan).
2. *Conquer*: memproses satu sub-persoalan secara rekursif.

Perlu digarisbawahi bahwa tidak ada tahap *combine* dalam *decrease and conquer.*

Ada tiga varian *decrease and conquer*:

1. *Decrease by a constant*: Ukuran instans dari persoalan direduksi sebesar konstanta yang sama setiap iterasi algoritma. Biasanya konstanta = 1.

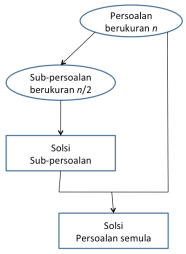


Gambar 2-2 Flowchart decrease by a constant. Sumber : Bahan Kuliah IF3051 Strategi Algoritma.

Ada beberapa permasalahan yang dapat diselesaikan dengan tipe ini, antara lain :

a) *Insertion sort* : Untuk melakukan pengurutan pada larik A[0..*n*-1], lakukan pengurutan pada A[0..*n*-2] secara rekursif lalu masukkan A[*n*-1] pada tempat yang benar pada larik A[0..*n*-2] yang sudah terurut. Biasanya diimplementasikan dengan skema non-rekursif (*bottom up*).
b) *Selection sort*
c) Algoritma Graf Traversal (DFS dan BFS)
d) *Topological sorting* : Mengurutkan simpul simpul yang ada di sebuah rantai, dengan menggunakan metode
   - *DFS-based Algorithm*, atau
   - *Source Removal Algorithm*

2. *Decrease by a constant factor*: Ukuran instans dari sebuah persoalan direduksi sebesar faktor konstanta yang sama setiap iterasi. Biasanya, faktor konstanta bernilai 2.



Gambar 2-3 Flowchart decrease by a constant factor. Sumber : Bahan Kuliah IF3051 Strategi Algoritma.

Ada beberapa permasalahan yang dapat diselesaikan dengan tipe ini, antara lain :

a) *Binary search* dan Metode *bisection*
b) *Multiplication à la russe* : Melakukan komputasi pada produk dari 2 integer positif
c) *Interpolation Search*
d) Mencari koin palsu : Diberikan *n* buah koin yang identik, tetapi salah satu diantaranya merupakan koin palsu

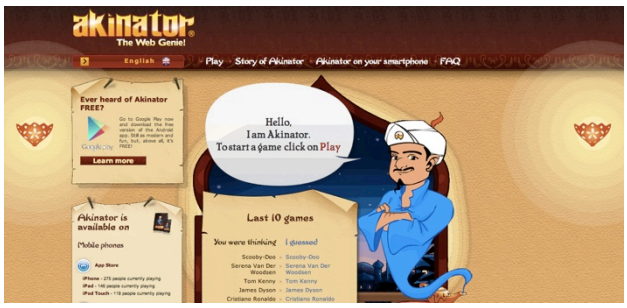
3. *Decrease by a variable size*: Ukuran instans persoalan direduksi bervariasi pada setiap iterasi algoritma.

1. Algoritma Euclid untuk *Greatest Common Divisor* (GCD)
2. Algoritma *Partition-based* untuk masalah seleksi
3. Beberapa algoritma pada *Binary Search Tree* (BST).
4. Menghitung median dan *Selection Problem* :

Mencari median dari *unsorted array* namun tidak perlu mengurutkannya terlebih dahulu.

C. Akinator

Akinator adalah sebuah game berbasis web yang dapat menebak tokoh yang ada pada pemikiran penggunanya. Saat ini akinator juga tersedia pada *App Store*, Google *Play,* WP *Marketplace* dan *Blackberrty*. Pada awal permainan ini, *user* akan diminta untuk memasukan nama, *gender*, dan umur. Selain itu, *user* juga diminta untuk memikirkan salah satu tokoh yang terkenal. Akinator dapat menebak tokoh baik fiksi maupun non fiksi. Akinator akan memberikan beberapa pertanyaan dan akan menebak tokoh yang ada dalam pikiran *user*, yang tergantung dari jawaban *user*. Basis data dari Akinator cukup lengkap, dari tokoh yang terkenal secara internasional, maupun lokal seperti Luna Maya, ataupun Presiden Susilo Bambang Yudhoyono.



Gambar 2-4 Halaman Awal Akinator. Sumber : http://en.akinator.com/

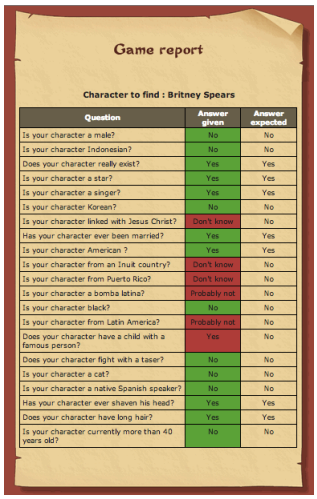
Pada saat penulis melakukan percobaan pada Akinator versi *desktop browser*, penulis memasukan nama, umur (20), *gender (female*) dan memikirkan Britney Spears sebagai tokohnya. Setelah itu, Akinator akan memberikan serangkaian pertanyaan yang bertujuan untuk mengeliminasi kemungkinan solusi yang ada.

Gambar 2-5 Cara Akinator melakukan reduksi. Sumber : http://en.akinator.com/

Setelah melalui beberapa serangkaian pertanyaan, Akinator berhasil menjawab pertanyaan penulis dan menampilkannya pada layar.

Gambar 2-6 Hasil percobaan apda Akinator. Sumber : http://en.akinator.com/

User pun dapat melakukan *validasi* apakah tokoh yang Akinator tebak merupakan tokoh yang seding user pikirkan. Apabila benar, Akinator dapat menampilkan *game report* yang berisi daftar pertanyaan beserta jawaban yang *user* masukan.


Game report

Character to find : Britney Spears

Question	Answer given	Answer expected
Is your character a male?	No	No
Is your character Indonesian?	No	No
Does your character really exist?	Yes	Yes
Is your character a star?	Yes	Yes
Is your character a singer?	Yes	Yes
Is your character Korean?	No	No
Is your character linked with Jesus Christ?	Don't know	No
Has your character ever been married?	Yes	Yes
Is your character American ?	Yes	Yes
Is your character from an Inuit country?	Don't know	No
Is your character from Puerto Rico?	Don't know	No
Is your character a bomba latina?	Probably not	No
Is your character black?	No	No
Is your character from Latin America?	Probably not	No
Does your character have a child with a famous person?	Yes	No
Does your character fight with a taser?	No	No
Is your character a cat?	No	No
Is your character a native Spanish speaker?	No	No
Has your character ever shaven his head?	Yes	Yes
Does your character have long hair?	Yes	Yes
Is your character currently more than 40 years old?	No	No

Gambar 2-7 Game Report dari Akinator. Sumber : http://en.akinator.com/

Dan apabila tokoh yang Akinator tampilkan tidak sesuai dengan tokoh yang user pikirkan, maka Akinator akan melakukan prosedur sebagai berikut :

1. Apabila ditemukan kemiripan pada database tokoh yang dimiliki, Akinator akan menampilkan beberapa tokoh yang mungkin, lalu,
   - Apabila tokoh yang dimaksud *user* muncul satu kali dalam daftar, *user* dapat memilihnya.
   - Apabila tokoh yang dimaksud *user* muncul lebih dari satu kali dari daftar, *user* harus memilih pilihan "*My character appears several times"*.
   - Apabila tokoh yang dimaksud *user* tidak ada pada daftar yang ditampilkan oleh Akinator, *user* harus memilih pilihan "*My character is not in the list*" dan memasukan

nama dan deskripsi singkat tokoh yang dimaksud.

2. Apabila tidak ditemukan kemiripan pada database tokoh yang dimiliki, *user* dapat memasukan nama dan deskripsi singkat tokoh yang dimaksud.

Data yang dimasukan oleh *user* dapat langsung diimplementasikan pada permainan, tetapi ada kemungkinan akan dihapus oleh moderator.

Selain itu, *user* juga dapat memasukan pertanyaan baru. Pada fitur ini, user akan memasukan konten utama pertanyaan yang ingin dibuat, lalu Akinator akan menampilkan pertanyaan-pertanyaan yang mungkin memiriki kemiripan dengan query user. Apabila tidak ada kesamaan, user dapat memasukkan pertanyaannya

III. Analisis Pemecahan Masalah

Shannon's *Entropy Statistic* mengatakan bahwa informasi yang dibutuhkan untuk mengindentifikasi sebuah objek yang belum jelas adalah maksimal sebanyak 20 bit. Apabila teori ini diemplementasikan pada 20 pertanyaan yang hanya memiliki jawaban ya dan tidak, dan pada setiap jawaban dapat mengeliminasi setengah dari kemunginan solusi, maka secara matematis rangkaian pertanyaan ini akan mampu mengindentifikasi 2^{20} atau 1,048,576 objek yang berbeda. Namun sayangnya, teori ini tidak sepenuhnya berlaku pada akinator.

Hal ini terjadi terutama karena Akinator memiliki 5 pilihan jawaban. Dalam kasus sempurna, dimana *user* benar-benar mengetahui jawaban yang pasti dari pertanyaan yang akinator berikan sehingga ia hanya akan menjawab "*Yes*"/1 atau "*No*"/0 dengan tingkat kepercayaan 100%, sehingga setiap pertanyaan yang terjawab akan mengeliminasi rata-rata ½ dari jumlah solusi sebelum pertanyaan tersebut. Tetapi pada praktiknya, rata-rata *user* akan memasukkan jawaban "*Probably Yes",* "*Probably No*" atau "*Don't Know*" yang mungkin tidak akan mereduksi jumlah solusi. Apabila hal ini dilakukan secara berulang, jumlah solusi yang tersisa akan tetap banyak, sedangkan Akinator harus mengambil keputusan tentang solusi mana yang akan dipilihnya pada saat pertanyaan ke 20. Kalau hal ini sampai terjadi, sangat kecil persentase keberhasilan Akinator untuk mendemostrasikan kemampuan '*mind reading'* nya.

Maka dari itu, analisis solusi dari masalah ini adalah :

1. Dari segi pemilihan pertanyaan, program harus membuat Akinator memberikan pertanyaan yang seefektif mungkin sehingga dapat mereduksi jumlah solusi kira-kira sampai ½ dari jumlah solusi awal.
2. Dari segi pilihan jawaban, program harus membuat pilihan absolut maupun non-absolut yang dipilih oleh *user* benar-benar mengurangi jumlah kemungkinan solusi dan membantu Akinator mengarah ke solusi yang paling tepat.
3. Dari segi pemilihan solusi, pada kasus dimana tidak mengarah ke sebuah solusi sampai 20 pertanyaan telah terjawab, program harus dapat membuat Akinator memilih solusi yang paling tepat.

Dari analisis tersebut, dibutuhkan struktur data sebagai berikut :

Gambar 3-1 Skema Basis Data Analisis Pemecahan Masalah.

Keterngan : Pada setiap permainan, variabel mTrue dan variabel mFalse diinisialisasi dengan false.

Kelas solusi merupakan tokoh-tokoh yang ada pada basis data Akinator. Sedangkan kelas atribut merupakan fakta yang ada pada masing-masing tokoh. Seperti *gender*, status, dan lain-lain. Untuk mTrue dan mFalse akan dibahas lebih lanjut pada strategi pemecahan masalah.

Dan berikut adalah penjelasan strateginya secara bertahap :

a. Strategi Pemilihan Pertanyaan

1. Untuk setiap atribut dari solusi yang tersisa, ambil *value* dari setiap atributnya (misalnya untuk atribut *gender*, tokoh dengan jenis kelamin laki-laki akan bernilai *true*, tokoh dengan jenis kelamin perempuan akan bernilai *false*).
2. Hitung setiap *value* pada setiap atribut yang ada pada solusi, dan simpan hasilnya.
3. Cari atribut yang memiliki jumlah nilai 0 hampir sama atau sama dengan jumlah nilai 1, dan gunakan atribut tersebut sebagai pertanyaan yang akan diberikan kepada *user.*

Dengan strategi ini, kemungkinan untuk melakukan reduksi pada solusi sebanyak mungkin sejak awal akan permainan dimulai semakin besar.

b. Strategi Pilihan Jawaban

1. *Yes*/1: Apabila atribut (pertanyaan) yang diberikan oleh Akinator dijawab dengan *true,* maka program akan langsung menghapus semua kemunkinan solusi dengan dengan solusi dengan nilai atribut yang sama bernilai *false.*
2. *No*/0 : Apabila atribut (pertanyaan) yang diberikan oleh Akinator dijawab dengan *false,* maka program akan langsung menghapus semua kemunkinan solusi dengan dengan solusi dengan nilai atribut yang sama bernilai *true*
3. *Don't Know :* Apabila atribut (pertanyaan) yang diberikan oleh Akinator dijawab dengan '*Don't Know*', maka tidak ada reduksi solusi yang dilakukan.
4. *Probably Yes :* Apabila atribut (pertanyaan) yang diberikan oleh Akinator dijawab dengan 'Probably Yes', maka untuk sesi permainan tersebut, atribut yang berkaitan nilai *mTrue*-

nya bernilai *true.*

5. *Probably Not :* Apabila atribut (pertanyaan) yang diberikan oleh Akinator dijawab dengan '*Probably Not*, maka untuk sesi permainan tersebut, atribut yang berkaitan nilai *mFalse*-nya bernilai *true*.

Reduksi terhadap solusi yang memiliki atribut mTrue dan mFalse yang bernilai 1 akan dilakukan pada tahap pemilihan solusi, dimana pada tahap ini Akinator belum menuju solusi tunggal. Sehingga, atribut yang memiliki mTrue = *true* akan mereduksi solusi lain yang memiliki atribut yang sama dengan nilai *false*. Begitupun pada mFalse.

c. Strategi Pemilihan Solusi

Setiap solusi memiliki *PopularCount*, sebuah variabel yang tidak diinisalisasi setiap kali permainan dimulai karena variabel ini akan menyimpan seberapa banyak tokoh ini dimainkan. Apabila hasil reduksi dari tahap-tahap sebelumnya belum valid, Akinator akan menggunakan *PopularCount* ini sebagai acuan.

IV. Batasan Strategi

Walaupun dari segi strategi analisis pemecahan masalah diatas terlihat dapat memberikan solusi sesuai yang dipikirkan oleh usernya, namun ada faktor-faktor eksternal yang dapat membuat strategi tidak lebih efektif. Faktor tersebut antara lain :

1. User dapat berbohong. Pada saat Akinator memberikan pertanyaan yang user tahu betul jawabannya, user memberikan jawaban yang tidak benar. Dan tidak konsisten. Tetapi saat Akinator memberikan solusi yang salah, user mengkoreksi kesalahan itu dengan memasukan solusi yang seharusnya. Hal ini dapat mengacaukan basis data dari Akinator sendiri.
2. User dapat memasukan informasi palsu saat user memasukan tokoh baru ataupun memasukan pertanyaan. Sama halnya dengan faktor pertama, hal ini dapat mengacaukan isi dari basis data Akinator.
3. Karena tokoh yang dapat ditebak oleh Akinator tidak hanya tokoh pada dunia nyata namun juga tokoh fiksi, ada beberapa hal atribut dari tokoh tersebut yang bersifat absolut. Misalnya pada tokoh Ranma pada serial kartun Jepang Ranma ½, tokoh ini dapat berganti-ganti jenis kelamin. Hal ini akan mengurangi reduksi pada solusi, membuat strategi ini tidak berjalan dengan efektif.

V. Kesimpulan dan Saran

Walaupun program Akinator terlihat seolah-olah dapat membaca pikiran manusia, hal ini tidak lepas dari strategi algoritma dan integritas dari basis data yang ada di baliknya. Strategi yang ada pada tulisan ini merupakan modifikasi dari algoritma *decrease and conquer* yang tereduksi dengan variabel yang berbeda-beda dari tiap iterasi (*decrease by variable size*). Penentuan besar setiap variabel apda setiap tahap decrease merupakan inti dari strategi ini.

Keabsahan dari strategi ini belum dapat dipastikan karena tidak diimplementasikan pada kelas data yang dimiliki oleh Akinator. Namun, seharusnya algoritma ini dapat diimplementasikan pada kelas data yang terdapat pada kelas analisis.

Untuk saran, program Akinator ini dapat menjadi salah satu kandidat yang baik untuk tugas besar mata kuliah IF3051 Strategi Algoritma. Selain para mahasiswa dapat mengasah kemampuan untuk menemukan solusi terbaik, program yang mereka rancang juga dapat dilombakan antar kelompok untuk menambahkan nilai bonus, asalkan para asisten menyediakan kelas data yang sama untuk setiap kelompok. Tokoh yang menjadi bahan uji juga harus sama untuk setiap kelompok, tetapi tidak diberitahukan dari awal.

References

[1] Levitin, Anany. Introduction to The Design & Analysis of Algorithms, 2nd Edition. Pearson Addison-Wesley. 2007. http://www.cs.berkeley.edu/~vazirani/algorithms/chap2.pdf

[2] http://en.akinator.com/

[3] Munir, Rinaldi. Bahan Kuliah IF3051 Strategi Algoritma. Program Studi Teknik Informatika, Sekolah Teknik Elektro dan In formatika ITB. 2011.

[4] Gerard, Benoit. Shannon entropy: a generic tool for analyzing statistical attacks. http://cca.saclay.inria.fr/Data/Benoit.Gerard-27-05-2011.pdf

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, 20 Desember 2012

Kania Azrina 13510058