THREE-GLASSES PROBLEM

M. Haekal Izmanda Pulungan

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknolohi Bandung
Jl. Ganesha No. 10, Bandung 40132
e-mail: if17020@students.if.itb.ac.id

ABSTRAK

Makalah ini membahas tentang pencarian solusi optimum pada sebuah permainan logika bernama *three-glasses problem*. Algoritma-algoritma yang dapat digunakan dalam memecahkan masalah pada permainan ini antara lain algoritma *brute force, greedy,* dan runut-balik, *n*amun algoritma yang akan dianalisis dalam makalah ini dibatasi hanya pada algoritma runut-balik saja. Analisis yang akan dilakukan adalah bagaimana solusi demi solusi dibangkitkan dengan menggunakan algoritma runut-balik dan bagaimana sebuah solusi dapat ditemukan.
Pembahasan dengan algoritma runut-balik dilakukan karena algoritma ini selalu dapat menemukan solusi dari sebuah permasalahan yang melibatkan logika dalam skala kecil maupun besar yang tidak lagi dapat dipikirkan langsung oleh otak manusia, selama solusi tersebut memang ada.
Permainan "Bridge Crossing" merepresentasikan sebuah kondisi di mana diperlukan sebuah pemikiran yang tepat untuk mengambil satu keputusan yang tepat dari berbagai peluang yang mungkin. Permainan ini hanya sebuah contoh kasus dalam kehidupan sehari-hari yang tidak luput dari masalah yang serupa dengan permainan ini. Dalam permainan ini, suatu langkah akan sangat mempengaruhi langkah selanjutnya. Suatu langkah bisa langsung mengarahkan ke solusi atau malah membuat kita berputar-putar pada langkah yang sama. Oleh karena itu, ada baiknya untuk mengetahui pemikiran atau algoritma yang baik untuk menyelesaikannya.

Kata kunci: *three-glasses problem*, runut-balik, solusi, optimal.

1. *THREE-GLASSES PROBLEM*

Banyak sekali permainan yang dapat menstimulus kemampuan otak kita. Kadangkala dibutuhkan pemikiran yang mendalam untuk menemukan penyelesaian dari permainan tersebut. Tidak hanya pemikiran, namun sebuah algoritma juga dapat membantu kita. Oleh karena itu, permainan semacam ini tidak hanya menjadi sekedar hiburan, akan tetapi juga digunakan sebagai sarana pembelajaran.

Salah satunya adalah *three-glasses problem* (permasalahan tiga gelas).

Penulis menyebutnya sebagai salah satu permainan karena mengetahuinya untuk pertama kalinya dari sebuah forum yang berisi permainan-permainan logika.

Secara singkat, berikut adalah penjelasannya.

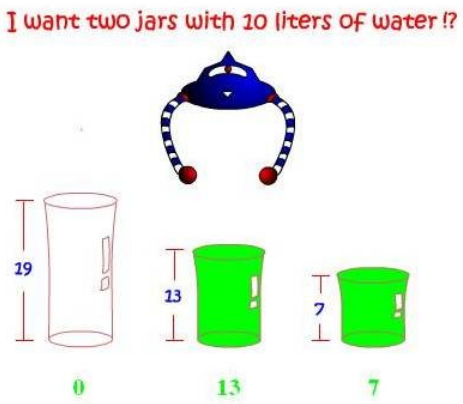
Terdapat tiga buah gelas dengan volume maksimal yang berbeda. Masing-masing berukuran 7 liter, 13 liter, dan 19 liter.

Saat ini gelas berukuran 7 liter dan 13 liter terisi penuh dengan air, sedangkan gelas terakhir (gelas berukuran 19 liter) kosong. Tujuan dari permainan ini adalah memperoleh **tepat** 10 liter air pada gelas berukuran 13 liter dan 19 liter.

Dalam menyelesaikan permainan ini, hal yang diperbolehkan adalah memindah-mindahkan isi satu gelas ke gelas lainnya. Gelas yang digunakan hanyalah ketga gelas yang telah disebutkan sebelumnya.

Air yang digunakan hanyalah air dari gelas berukuran 7 liter dan 13 liter pada awal permainan (total 20 liter). Air tidak boleh dipindahkan kecuali ke salah satu dari dua gelas lainnya, atau dengan kata lain, air tidak boleh dibuang.

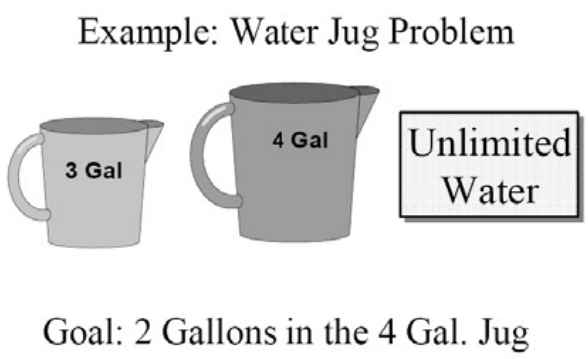
Dari bentuk masing-masing gelas, kita sama sekali tidak bisa mengetahui dengan tepat (tanpa menggunakan alat bantu) volume air yang ada pada saat itu kecuali pada saat suatu gelas terisi penuh, yang berarti bahwa volume air adalah volume maksimal dari gelas penampungnya.



Gambar 1. Ilustrasi dari *three-glasses problem*

Terdapat variasi dari permainan ini berkaitan dengan volume maksimal gelasnya (3, 5, dan 8 liter, dengan tujuan untuk mendapatkan 4 liter).

Selain itu, terdapat juga *water jug probem*.



Gambar 2. Ilustrasi dari *water jug problem*

Perbedaannya adalah bahwa pada *water-jug problem*, wadah yang digunakan hanya dua buah dengan volume maksimal masing-masing 3 liter dan 4 liter. Tujuannya adalah untuk mendapatkan air **tepat** 2 liter. Air yang digunakan (diasumsikan) tidak terbatas dan gelas dapat dikosongkan tanpa harus mengisikan air yang ditampungnya ke gelas lainnya.

Dengan adanya perbedaan ini, permainan *three-glasses problem* akan terasa lebih sulit dan menantang.

2. METODE PENYELESAIAN

Seperti yang disebutkan pada bagian pendahuluan, dalam menyelesaikan permainan ini dapat digunakan suatu algoritma, yang dalam hal ini algoritma yang digunakan adalah algoritma *backtracking*.

2.1 Algoritma Runut-balik

Algoritma runut-balik (*backtracking*) sangat erat hubungannya dengan pembentukan pohon ruang status. Pencarian solusi akan dilakukan dengan menelusuri simpul-simpul dalam pohon tersebut.

Algoritma runut-balik melibatkan algoritma *DFS* (*depth-first search*) dalam penelusuran simpul-simpul pada pohon sehingga penelusuran dilakukan sampai simpul terdalam terlebih dahulu.

Secara umum, langkah-langkah dalam algoritma runut-balik adalah sebagai berikut.

1. Simpul yang sudah dibangkitkan dinamai **simpul hidup,** sedangkan simpul yang sedang diperluas dinamai **simpul-E**. Pencarian solusi dilakukan dengan membentuk simpul akar sebagai **simpul-E** terlebih dahulu (simpul akar merupakan keadaan awal).
2. Perluas setiap simpul-E dengan menelusuri simpul-simpul anaknya sampai ditemukan simpul yang tidak mengarah ke solusi. Simpul tersebut dinamakan **simpul mati**. Sebuah fungsi yang digunakan untuk menandai sebuah simpul adalah simpul mati bernama **fungsi pembatas**. Simpul mati tidak akan diperluas lagi.
3. Jika menemukan simpul mati, proses perluasan dilakukan dengan membangkitkan simpul anak pada simpul bapak dari simpul mati yang ditemukan. Jika pada awal tahap 3 ini tidak ada lagi simpul yang dapat dibangkitkan, maka perluasan dilakukan pada simpul pada aras setingkat di atasnya.
4. Pencarian selesai jika ditemukan sebuah solusi atau seluruh simpul sudah dibangkitkan.

Berikut adalah penjabarannya dalam notasi algoritma.

```
procedure RunutBalikI(input n:integer)
{Mencari semua solusi persoalan dengan
algoritma backtracking-iteratif.
Input: n (panjang vektor solusi)
Output: solusi = (x[1], x[2],…, x[n])}
Kamus
k : integer
Algoritma
k ← 1
while (k > 0) do
   if (x[k] belum dicoba sedemikian
   sehingga x[k]←T(k)) and
   (B(x[1], x[2], ... ,x[k])= true)
   then
      if (x[1],x[2],...,x[k]) adalah
      lintasan dari akar ke daun then
        CetakSolusi(x)
      {endif}

      k←k+1
   else {x[1], x[2], …, x[k] tidak
```

```
   mengarah ke simpul solusi}
      k←k-1 {runut-balik ke anggota
      tupple sebelumnya}
   {endif}
{endwhile; k = 0 }
```

2.2 Implementasi Algoritma Runut Balik Untuk Penyelesaian *Three-Glasses Problem*

Untuk memudahkan, selanjutnya gelas dengan volume maksimal 7 liter disebut dengan *gelas pertama*, gelas 13 liter *gelas kedua*, dan gelas 19 liter dengan *gelas ketiga*.

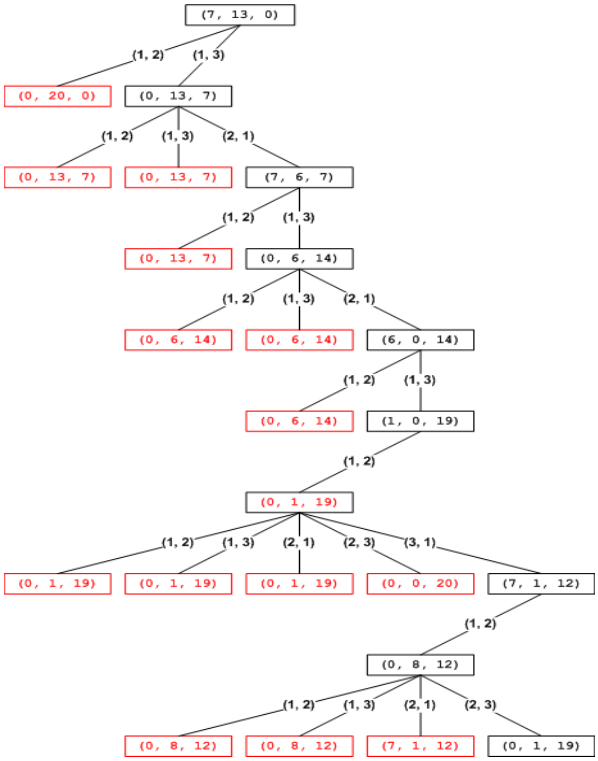
Status pada *state space tree* menyatakan volume air yang ditampung masing-masing gelas pada saat itu. Status tersebut dituliskan dengan format (v_1, v_2, v_3), di mana v_1 menunjukkan volume air pada gelas pertama, v_2 pada gelas kedua, dan v_3 pada gelas ketiga. Sebagai contoh, (0, 10, 10) berarti pada saat itu gelas pertama berisi 0 liter (kosong), gelas kedua dan ketiga berisi 10 liter.

Dalam menentukan fungsi pembatas, penulis menemukan bahwa selama algoritma ini "bekerja", sangat mungkin nantinya ditimbulkannya kembali suatu status (*repeated-state*). Status ini tentunya tidak dibangkitkan (atau kita anggap sebagai simpul mati). Oleh karena itu, dalam hal ini, fungsi pembatas akan memeriksa apakah status yang akan dibangkitkan sudah pernah dibangkitkan sebelumnya. Berarti, adanya penanganan untuk menampung status apa saja yang sudah pernah dibangkitkan.

Selain itu, fungsi pembatas juga harus memeriksa apakah status yang dibangkitkan menunjukkan volume air yang melebihi dari volume maksimal masing-masing gelas. Terakhir, fungsi ini juga memeriksa apakah *gelas-dari* pada saat itu kosong.

Busur (*edge*) pada pohon dilabeli dengan pemindahan isi air yang dilakukan, yang dituliskan dengan format (c_s, c_f), di mana c_s menunjukkan gelas mana yang isinya akan dipindahkan (atau *gelas-dari*) dan c_f menunjukkan ke gelas mana isi dari gelas-dari akan dipindahkan (atau *gelas-ke*). Sebagai contoh, (1, 3) berarti dilakukan pemindahan isi gelas pertama ke gelas ketiga.

Berikut adalah ilustrasi dalam bentuk graf yang menunjukkan bagaiman algoritma runut-balik bekerja.



Gambar 3. Ilustrasi proses pada saat algoritma "bekerja"

Simpul berwarna merah adalah simpul mati.

Salah satu perluasan dari simpul (0, 1, 19) adalah (0, 1, 19). Ternyata ini adalah simpul mati karena telah dibangkitkan sebelumnya. Selanjutnya, perluasan dilanjutkan sampai dibangkitkannya simpul (7, 1, 12). Simpul inilah selanjutnya yang diperluas, hingg didapatkan simpul (0, 8, 12).

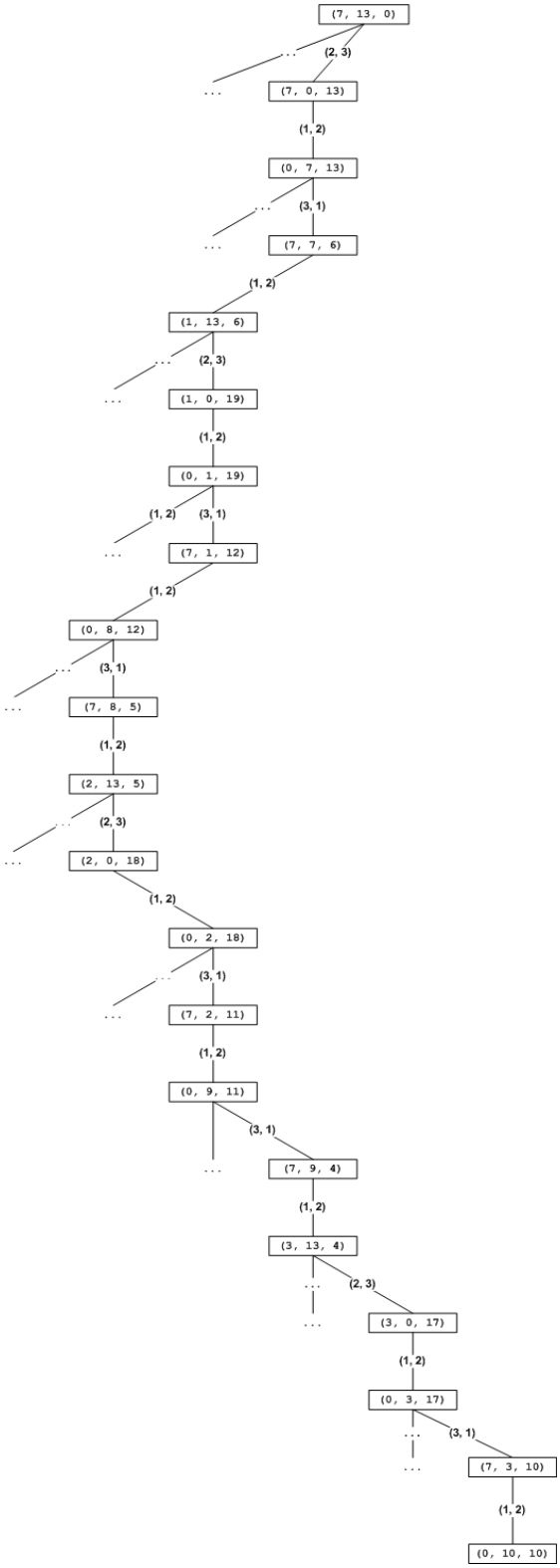
Solusi yang ingin dicapai adalah (0, 10, 10).

Dalam perluasan simpul-simpulnya, harus ditentukan prioritas busurnya. Pada contoh di atas, prioritasnya adalah (terurut dari busur dengan prioritas tertinggi) (1,2)-(1,3)-(2,1)-(2,3)-(3,1)-(3,2).

Prioritas busur ini juga akan menentukan bagaimana proses yang akan berlangsung, baik itu mempengaruhi kedalaman solusi maupun banyaknya simpul yang harus dibangkitkan.

Penentuan prioritas busur ini juga akan menentukan keoptimalan solusi yang akan dihasilkan

Berikut adalah ilustrasi proses sampai tercapainya solusi. Ilustrasi berikut tidak menampilkan percabangan yang berujung pada simpul mati.



Gambar 4. Ilustrasi hingga diperolehnya solusi

3. KESIMPULAN

Algoritma runut-balik atau *backtracking* cukup baik untuk digunakan dalam menyelesaikan *three-glasses problem* dan masalah-masalah lainnya yang serupa. Penyelesaian permainan ini sebenarnya masih dapat diselesaikan dengan "coba-coba", namun penggunaan algoritma seperti yang telah dijabarkan di atas akan menghasilkan solusi yang lebih atau bahkan paling optimal.

Optimalnya solusi yang ditunjukkan bergantung pada prioritas busur yang kita tentukan.

Akan tetapi, banyak hal yang masih bisa dilakukan sebagai pengembangan, seperti optimasi algoritma pada pengkodean (seperti penanganan akan perlunya rekam status yang telah dibangkitkan, dll).

Dengan sedikit modifikasi, entah itu dengan menggunakan pengaturan prioritas busur yang berbeda-beda atau membiarkan proses berlangsung sampai tidak ada lagi simpul yang bisa dibangkitkan, kita akan memperoleh banyak solusi yang mungkin, kecuali solusi yang mengandung pengulangan status. Inilah salah satu kelebihan dari penggunaan algoritma ini.

REFERENSI

[1] ttp://www.digitalspy.co.uk/forums/showthread.php?t=706770 (diakses pada 5 Januari 2010)

[2] ttp://img413.imageshack.us/i/wateret8.jpg/ (diakses pada 5 Januari 2010)

[3] unir, Rinaldi, "Diktat Kuliah Strategi Algoritma", Program Studi Teknik Informatika ITB, 2006