PEMECAHAN MASALAH *EGYPTIAN FRACTION* MENGGUNAKAN ALGORITMA *CONFLICT RESOLUTION*

Dimas Yusuf Danurwenda

Sekolah Tinggi Elektro dan Informatika
Jl. Ganesha 10 Bandung
email: if15002@students.if.itb.ac.id

ABSTRAK

***Egyptian fraction* (pecahan Mesir) merupakan permasalahan yang telah dicoba untuk dipecahkan oleh para matematikawan di seluruh dunia. Inti dari pecahan Mesir adalah menyatakan suatu bilangan rasional $\frac{p}{q}$, $0 < \frac{p}{q} < 1$ sebagai jumlah dari beberapa bilangan harmonik yang berbeda. Beberapa metode untuk mencari bilangan-bilangan harmonik ini telah ditemukan, dan sekarang para matematikawan berusaha mencari metode yang paling efisien, dalam arti meminimalkan jumlah unit bilangan harmonik sesedikit mungkin (1).**

Algoritma yang paling mendasar dalam memecahkan masalah pecahan Mesir ini adalah algoritma *greedy*, dimana dalam setiap langkah kita mencari unit pecahan terbesar yang tidak lebih dari bilangan yang terisisa. Algoritma ini telah cukup banyak dikembangkan, termasuk salah satunya yang disebut *reverse greedy algorithm.* Namun algoritma ini memiliki masalah dalam memisahkan unit-unit pecahan yang sama, yang kemudian digunakan algoritma *conflict resolution* untuk memecahkan masalah ini.

Makalah ini akan membahas salah satu metode dalam memecahkan permasalahan pecahan Mesir, yaitu *conflict resolution*, implementasi dari algoritma tersebut, dan menganalisa performansi. Juga akan dibahas mengenai banyaknya unit pecahan yang diperlukan untuk menyatakan beberapa bilangan rasional dengan pembilang berupa bilangan yang kecil.

Kata kunci: *greedy*, *conflict resolution*, bilangan harmonik, bilangan rasional.

1. PENDAHULUAN

Pertama-tama untuk mengurangi keambiguan makna, dari paragraf ini hingga akhir makalah ini, kita sepakati bahwa **bilangan rasional** adalah bilangan yang berbentuk $\frac{p}{q}$, $0 < \frac{p}{q} < 1$, walaupun sebenarnya bilangan rasional tidak harus berada dalam rentang nol hingga satu.

Pada zaman modern ini ketika kita hendak menyatakan sebuah bilangan rasional, kita telah terbiasa dengan 2 format. Format pertama adalah format pecahan, di mana kita menyatakan bilangan rasional tersebut dalam bentuk $\frac{p}{q}$, di mana p dan q adalah bilangan bulat, dan q bernilai positif. Format lainnya yang sering dipakai adalah format penulisan desimal, yaitu menggunakan titik sebagai pemisah antara bagian bulat dan bagian pecahan, sebagaimana kita menyatakan $\frac{5}{8}$ sebagai 0.625. Format manapun yang kita gunakan, jika kita menggunakan komputer dalam mengolah bilangan rasional tersebut, komputer akan mengolahnya dalam bentuk biner. Tidak demikian halnya dengan masyarakat Mesir kuno. Masyarakat Mesir kuno menyatakan bilangan rasional tersebut sebagai penjumlahan dari beberapa unit pecahan yang berbeda. Unit pecahan-disebut juga bilangan harmonik-adalah bilangan rasional dalam bentuk $\frac{1}{n}$, dimana n adalah sebuah bilangan bulat positif. Sebagai contoh mereka menyatakan $\frac{2}{5}$ sebagai $\frac{1}{3}+\frac{1}{15}$, atau menyatakan $\frac{87}{100}$ sebagai $\frac{1}{2}+\frac{1}{5}+\frac{1}{11}$. Pada zaman modern ini, format penulisan seperti ini disebut sebagai *egyptian fraction* (pecahan Mesir). Tentu saja mereka tidak menggunakan angka arab dalam penulisan, melainkan menggunakan hierogliph.

Mudah untuk melihat bahwa beberapa bilangan rasional memiliki lebih dari satu representasi dalam pecahan Mesir. Sebagai contoh kita dapat menyatakan $\frac{8}{15}$ dalam bentuk $\frac{1}{3}+\frac{1}{5}$ atau $\frac{1}{2}+\frac{1}{30}$. Pada kenyataannya, setiap bilangan rasional memiliki **tak-hingga** banyaknya representasi dalam pecahan Mesir (2), walaupun demikian hanya berhingga banyaknya bilangan rasional yang dapat dinyatakan sebagai pecahan Mesir dalam banyak unit tertentu. Tidak diketahui bagaimana masyarakat Mesir menemukan sistem penulisan tersebut, namun saat ini beberapa algoritma telah ditemukan untuk memecahkan permasalahan ini. Setiap algoritma memiliki perbedaan dalam banyaknya unit pecahan yang dihasilkan, besarnya

pembilang dari unit-unit pecahan, dan tentu saja lamanya waktu untuk mendapatkan representasi yang memenuhi.

2. METODE

2.1. ALGORITMA *CONFLICT RESOLUTION*

Latar belakang munculnya algoritma *conflict resolution* adalah gagalnya algoritma *greedy* untuk membentuk unit-unit pecahan yang berbeda. Misalnya untuk bilangan rasional $\frac{2}{3}$, algoritma greedy akan memberikan $\frac{1}{3}$, lalu pada loop berikutnya memberikan unit yang sama yaitu $\frac{1}{3}$. Hal ini memaksa para matematikawan untuk mengmbangkan suatu algroritma yang dapat menjamin dihasilkannya unit-unit pecahan yang berbeda.

Ide dari algoritma *conflict resolution* cukup sederhana: dari sebuah pecahan $\frac{x}{y}$ kita cukup menerimanya sebagai penjumlahan x buah unit pecahan $\frac{1}{y}$. Namun karena pecahan Mesir menuntut unit-unit pecahan yang berbeda, sekarang kita mencari pasangan pecahan yang sama, menjumlahkan mereka, kemudian mencari representasi pecahan Mesir untuk jumlah tersebut. Metode ini memberikan hasil yang berbeda-beda tergantung dari pemilihan pecahan Mesir yang menggantikan pasangan pecahan. Secara umum mudah diterima bahwa algoritma ini akan memberikan hasil representasi yang memenuhi syarat, namun tidak mudah untuk melihat bahwa algoritma ini akan berhenti (tidak masuk dalam sebuah *loop* yang tak terbatas), atau menganalisa kinerja dari algoritma ini. Kita akan membahas 2 varian dari algoritma *conflict resolution*, yaitu metode pemasangan dan metode pemisahan.

2.2. METODE PEMASANGAN (*PAIRING METHOD*)

Metode ini menggunakan ide dari *conflict resolution* sebagaimana disebut di atas. Setiap kali kita mendapatkan pasangan unit pecahan yang sama, kita menggantinya dengan representasi Mesir yang memberikan nilai yang sama. Secara matematis, algoritma ini dapat dituliskan sebagai

$$\frac{1}{y}+\frac{1}{y}=\left\{\begin{array}{c}\frac{2}{y}, \text{jika } y \text{ genap} \\ \frac{2}{y+1}+\frac{2}{y(y+1)}, \text{jika } y \text{ ganjil}\end{array}\right\}$$

Perhatikan bahwa pada masing-masing kasus, dihasilkan unit-unit pecahan yang memenuhi syarat pecahan Mesir. Urutan pasangan pecahan yang mana yang harus diganti terlebih dahulu tidak menentukan representasi akhir. Dalam memilih struktur data, kita dapat memilih sebuah list yang berisikan unit-unit pecahan yang selalu terurut mengecil. Berikut pseudocode untuk algoritma ini.

```
function pairMethod(int x, int y)→list
deklarasi
      L:list;
begin
      if(x=1)
      begin
            buat list L, berisikan satu
            elemen yaitu x/y;
      end
      else
      begin
            buat list L, berisikan unit-unit
            pecahan 1/y sebanyak x buah;
            while(ada 2 elemen yang sama dari
            L) do
            begin
                  ambil 2 elemen yang sama,
                  ganti dengan representasi
                  Mesir;
            end
      end
      return L;
end;
```

Setiap penggantian $\frac{1}{y}+\frac{1}{y}$ dengan $\frac{2}{y}$ saat y genap akan mengurangi banyak elemen list sebesar satu buah, dan pengurangan ini paling banyak dapat dilakukan sebanyak x kali. Penggantian saat y ganjil tidak mengubah banyaknya elemen list tetapi memperbesar perbedaan di antara elemen-elemen list (perhatikan $y+1$ dengan $y(y+1)$), sehingga sekarang kita dapat melihat bahwa suatu saat algoritma ini pasti akan berhenti, yang berarti, pasti tercapai keadaan dimana tidak ada elemen list yang sama, dan memiliki paling banyak x elemen list.

Sekarang marilah kita menentukan penyebut terbesar yang mungkin muncul dalam elemen list. Salah satu elemen pasti bernilai sedikitnya $\frac{1}{y}$, dan secara umum dapat dilihat bahwa jika jumlah dari beberapa elemen awal adalah $\frac{x}{y}-\frac{a}{b}$, maka pecahan berikutnya yang mungkin muncul pasti sedikitnya $\frac{a}{xb}$. Jadi jika kita memisahkan unit-unit dari representasi akhir diurutkan berdasarkan ukuran unit, maka dalam setiap langkah penyebut dari unit yang kita ambil akan cenderung menuju ke penyebut sebelumnya pangkat $2x$, sehingga kita simpulkan penyebut terbesar yang mungkin muncul adalah $xy^{(2^x)}$. Nilai yang cukup besar ini terlihat terlalu besar, berdasarkan asumsi heuristik bahwa pecahan-pecahan yang sama tidak dihasilkan dari pasangan-pasangan unit yang berbeda, sehingga kita dapat mengurangi nilai tersebut mendekati y^x. Nilai ini masih dapat dikurangi lagi mengingat pasangan unit pecahan dengan penyebut

genap akan mengurangi jumlah pecahan, yang berarti unit-unit akan semakin besar, yang berarti penyebut-penyebut akan semakin kecil. Berikut diberikan sebuah contoh penggunaan metode pemasangan.

$$\frac{18}{23} = \frac{1}{2} + \frac{1}{6} + \frac{1}{12} + \frac{1}{35} + \frac{1}{276} + \frac{1}{2415}$$

Mengikuti metode pemasangan ini, muncul sebuah teorema yang dibuktikan oleh Tenzo Takenochi (3).

Teorema

Misalkan q dapat direpresentasikan sebagai jumlah dari t buah unit pecahan (tidak harus berbeda), maka q memiliki representasi Mesir dengan paling banyak t unit.

2.3. METODE PEMISAHAN (*SPLITTING METHOD*)

Beda halnya dengan metode pemasangan, pada metode pemisahan akan dihasilkan banyak elemen yang lebih banyak. Hal ini disebabkan metode pemisahan akan membiarkan salah satu unit dari suatu pasangan unit tetap ada, kemudian memecah unit yang satunya lagi menjadi suatu representasi Mesir. Secara matematis algoritma ini dapat ditulis sebagai

$$\frac{1}{y} + \frac{1}{y} = \frac{1}{y} + \frac{1}{y+1} + \frac{1}{y(y+1)}.$$

Terlihat jelas bahwa algoritma ini menghasilkan elemen list yang lebih banyak, karena setiap kali ditemukan pasangan unit, elemen dari list unit akan bertambah satu buah. Juga algoritma ini tidak memiliki kejelasan apakah algoritma ini tidak akan masuk kedalam sebuah *loop* tak terbatas, tapi bagaimanapun Graham dan Jewett telah membuktikan bahwa algoritma ini pasti berhenti (4). Jika kita gunakan asumsi kasus terbaik dimana tidak terdapat dua unit yang sama dan muncul dalam dua cara yang berbeda (salah satu muncul sebagai $\frac{1}{y+1}$ dan yang lain muncul sebagai $\frac{1}{y(y+1)}$), maka kita dapat menganalisa bahwa dengan input $\frac{x}{y}$ akan terjadi $x-1$ level pemisahan, dimana setiap level akan menggandakan jumlah elemen list. Sehingga kita simpulkan total elemen list yang dihasilkan adalah dalam orde $O(2^x)$, dengan penyebut terbesar berada dalam orde $O(y^{(2^x)})$. Sekali lagi, analisis ini didasarkan pada kasus terbaik, pada kenyataanya orde-orde tersebut dapat menjadi lebih besar. Sebagai contoh perhatikan output dari metode pemisahan dengan input $\frac{5}{6}$:

$$\begin{aligned}\frac{5}{6} = \frac{1}{6} &+ \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \frac{1}{42} + \frac{1}{43} + \frac{1}{44} + \frac{1}{45} + \frac{1}{56} \\ &+ \frac{1}{57} + \frac{1}{58} + \frac{1}{72} + \frac{1}{73} + \frac{1}{90} + \frac{1}{1806} \\ &+ \frac{1}{1807} + \frac{1}{1808} + \frac{1}{1892} + \frac{1}{1893} \\ &+ \frac{1}{1980} + \frac{1}{3192} + \frac{1}{3193} + \frac{1}{3306} \\ &+ \frac{1}{5256} + \frac{1}{3263442} + \frac{1}{3263443} \\ &+ \frac{1}{3267056} + \frac{1}{3581556} + \frac{1}{10192056} \\ &+ \frac{1}{10650056950806}\end{aligned}$$

2.4. APLIKASI *CONFLICT RESOLUTION* DALAM ALGORITMA LAIN

Setelah memiliki algoritma *conflict resolution*, para matematikawan mulai mengembangkan algoritma-algoritma lain yang menggunakan *conflict resolution*. Seperti yang telah dibahas pada awal makalah, kegagalan dari algoritma greedy dapat ditutupi dengan adanya *conflict resolution.* Berikut akan diulas secara singkat sebuah algoritma yang menggunakan *conflict resolution*, yaitu algoritma *Reverse Greedy Algorithm.*

2.4.1. *Reverse Greedy Algorithm (RGA)*

Ide utama dari RGA adalah membangkitkan unit-unit pecahan secara terbalik, dalam arti kita menentukan unit yang memiliki penyebut terbesar terlebih dahulu. Pada setiap tahapan, kita menyatakan bilangan rasional $\frac{x}{y}$ sebagai $\frac{x'}{y'} + \frac{1}{u}$, dengan tujuan membentuk y' sekecil mungkin. Hal ini bertentangan dengan algoritma greedy yang biasa dimana tujuan pembentukan unit adalah membentuk u sekecil mungkin. Dengan menggunakan pengetahuan mengenai *continued fraction*, dapat dilihat bahwa kita selalu bisa mendapatkan nilai y' sehingga $y' < y$.

Pertama-tama kita menentukan sebuah nilai d, dimana nilai ini kemudian akan sama dengan $xy' - x'y$. Sedikit analisis cukup untuk membatasi pencarian kita terhadap d. Misalkan terdapat bilangan prima p yang membagi d namun tidak membagi y. Karena $u = \frac{yy'}{d}$, maka kita ketahui bahwa p habis membagi y'. Selanjutnya, karena $x' = \frac{xy'-d}{y}$, p membagi x' dan $\frac{x'}{y'}$ tidak berada dalam bentuk efektif (y' yang sekecil mungkin), sehingga kita dapat mengabaikan d sedemikian. Dengan cara yang serupa dapat dilihat bahwa jika p^c membagi y, dan p^{2c+k} membagi d, kita dapatkan fakta bahwa p^{c+k} membagi y' dan p^k membagi x'. Jadi untuk pilihan terbaik kita ambil d sebagai pembagi dari y^2. Saat pembentukan unit

pecahan, setiap faktor dari y^2 akan menjadi unit pecahan $\frac{1}{u}$, tetapi untuk setiap faktor tersebut, beberapa akan mengakibatkan $\frac{x'}{y'}$ bernilai negatif atau bahkan membuat y' bernilai nol. Kita mengabaikan faktor-faktor sedemikian sehingga d yang kita pilih dapat membawa kita pada representasi Mesir yang valid.

Conflict resolution digunakan untuk menghindari terjadinya duplikasi unit pecahan, sebab jika tidak digunakan, misalnya pada kasus input $\frac{23}{231}$, algoritma ini akan menghasilkan $\frac{1}{11}+\frac{1}{231}+\frac{1}{231}$, yang bukan merupakan representasi Mesir yang valid. Dengan algoritma ini, banyaknya unit pecahan berada pada orde $O(y)$.

3. KESIMPULAN

a. Algoritma *conflict resolution* mampu menjamin terbentuknya unit-unit pecahan yang berbeda dalam representasi Mesir suatu bilangan rasional.
b. Metode Pemasangan (*Pairing Method*) memberikan hasil yang lebih efektif dibandingakan Metode Pemisahan(*Splitting Method*), dimana untuk bilangan rasional $\frac{x}{y}$, representasi Mesir paling banyak memerlukan x unit pecahan.
c. Jika suatu bilangan rasional q dapat dinyatakan sebagai jumlah dari t buah unit pecahan, maka terdapat representasi Mesir untuk q dengan banyak unit pecahan paling banyak t unit.
d. Algoritma *conflict resolution* dapat digunakan dalam algoritma-algoritma mengenai representasi pecahan Mesir yang lain.

4. REFERENSI

1. **David, E.** *ICS official website.* [Online] http://www.ics.uci.edu/.akses 22 May 2007 20.30
2. **I.Stewart.** The Riddle of The Vanishing Camel. *Scientific American.* 1992.
3. **T.Takenochi.** On An Indeterminate Equation. *Proc. Physico-Mathematical Soc. of Japan.* 1921, Vol. III.
4. **Wagon, S.** *Mathematica in Action.* s.l. : W.H. Freeman, 1991.