MEMBANGUN PECAHAN MESIR (*EGYPTIAN FRACTION*) DENGAN METODE-METODE BERBASIS APROKSIMASI

Satrio Adi Rukmono

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Labtek V, Jalan Ganeça 10 Bandung
e-mail: r.satrioadi@yahoo.com

ABSTRAK

Pecahan Mesir (*Egyptian Fraction)* adalah penjumlahan dari beberapa pecahan yang berbeda di mana setiap pecahan tersebut memiliki pembilang 1 dan penyebut berupa bilangan bulat positif yang berbeda satu sama lain (yang disebut sebagai pecahan satuan atau *unit fraction*). Penjumlahan ini menghasilkan suatu bilangan rasional positif *a/b* di mana 0 < *a/b* < 1. Penjumlahan pecahan semacam ini berperan penting dalam matematika Mesir Kuno, karena notasi dalam matematika Mesir kuno hanya mengenal pecahan berpembilang 1 dengan perkecualian 2/3 dan 3/4. Pada zaman modern ini notasi pecahan dalam bentuk pecahan Mesir tidak lagi digunakan secara umum karena pecahan yang dapat menerima semua bilangan bulat sebagai pembilang tentunya lebih mudah untuk digunakan dalam berbagai komputasi dari pada harus menyatakan pecahan tersebut sebagai penjumlahan pecahan satuan. Selain itu, penggunaan notasi desimal dengan simbol pemisah antara bilangan bulat dan bagian pecahan (biasanya menggunakan simbol titik ('.') atau koma (',')) juga lebih memudahkan perhitungan. Akan tetapi, notasi pecahan Mesir tetap menarik untuk dipelajari, karena terdapat banyak metode untuk mengubah suatu pecahan biasa (*vulgar fraction* atau *common fraction*) *a/b* ke dalam bentuk pecahan Mesir. Makalah ini membahas tiga metode pembangkitan pecahan Mesir yang berdasarkan aproksimasi, yaitu metode *greedy*, *harmonic*, dan *odd greedy*.

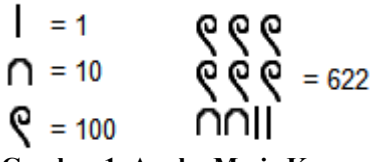
Kata kunci: *Egyptian Fraction*, Pecahan Mesir, Metode Aproksimasi, *Greedy*, *Odd Greedy*, *Harmonic*.

1. PENDAHULUAN

Pada masa Mesir Kuno, orang-orang Mesir menggunakan gambar sebagai media komunikasi tertulis, yaitu huruf yang dikenal dengan nama hieroglif. Demikian juga dengan angka. Bangsa Mesir Kuno mengenal bilangan berbasis sepuluh dengan simbol-simbol yang memiliki nilai 1, 10, 100, 1.000, 10.000, 100.000, dan 1.000.000 (simbol yang melambangkan bilangan 1.000.000 ini juga sering digunakan untuk melambangkan suatu bilangan yang sangat besar). Kelipatan masing-masing nomina tersebut didapatkan dengan cara mengulang simbol-simbol yang bersesuaian, misalnya untuk menuliskan 622 bangsa Mesir mengulang simbol yang melambangkan bilangan 100 sebanyak enam kali diikuti simbol yang melambangkan bilangan 10 sebanyak dua kali, kemudian simbol yang melambangkan bilangan 1 sebanyak dua kali (Gambar 1).

Permasalahan muncul ketika dibutuhkan penghitungan yang melibatkan pecahan. Untuk menyatakan pecahan, bangsa Mesir Kuno menambahkan satu simbol yang berbentuk seperti mulut di atas simbol angka untuk membaliknya. Jadi, untuk menyatakan bilangan 1/3 cukup dilakukan dengan menambahkan simbol "mulut" tersebut di atas simbol yang menyatakan bilangan 3 (Gambar 2). Perkecualian ada pada pecahan-pecahan 2/3 dan 3/4 yang memiliki simbol sendiri.

Dengan demikian, pecahan biasa dengan pembilang lebih besar dari pada satu harus ditulis sebagai penjumlahan dari pecahan-pecahan satuan. Bentuk inilah yang disebut dengan pecahan Mesir. Bentuk pecahan Mesir yang diinginkan tentunya adalah bentuk paling ringkas, yaitu menggunakan sesedikit mungkin pecahan satuan. Hal inilah yang mendorong timbulnya berbagai strategi algoritmik untuk membangun suatu pecahan Mesir dari pecahan biasa.



Gambar 1. Angka Mesir Kuno

Gambar 2: Pecahan Mesir Kuno

Papirus Rhind[1] yang dibuat sekitar tahun 1650 SM memuat sebuah tabel yang berisi representasi pecahan Mesir dari semua pecahan 2/*n* dengan *n* merupakan bilangan ganjil antara 5 dan 101. Pecahan 2/3 adalah satu-satunya pecahan yang tidak dapat dinyatakan dalam pecahan Mesir[2], oleh karena itu pecahan ini diberi simbol khusus. Pecahan 3/4 diberi simbol sendiri hanya untuk mempersingkat penulisan. Untuk menyederhanakan persoalan, pada makalah ini bilangan 2/3 dan 3/4 dianggap tidak memiliki simbol sendiri.

2. METODE

Terdapat banyak algoritma untuk membangun pecahan Mesir. Namun, hingga saat ini belum ditemukan algoritma yang selalu memberikan solusi optimal[3]. Metode-metode berdasarkan aproksimasi merupakan metode yang natural untuk menyelesaikan persoalan ini, yaitu dengan mencari satu pecahan satuan yang terdekat dengan bilangan yang ingin dicari pecahan Mesir-nya, kemudian mengaplikasikan metode yang sama untuk sisanya[4].

Sebagai contoh, pecahan satuan terbesar yang kurang dari pecahan biasa 5/6 adalah 1/2, menyisakan 1/3 (yaitu 5/6 dikurangi 1/2) , sehingga didapat representasi pecahan Mesir untuk 5/6, yaitu 1/2 + 1/3. Dari dasar pemikiran ini dapat diturunkan algoritma-algoritma untuk membangun representasi pecahan Mesir dari suatu bilangan rasional, antara lain yang akan dibahas pada makalah ini, yaitu algoritma *greedy*, *harmonic*, dan *odd greedy*.

2.1 Algoritma *Greedy*

Algoritma greedy untuk penyelesaian masalah ini merupakan algoritma paling sederhana yang berbasiskan aproksimasi. Oleh karena itu, hasil yang didapatkan pun kalah optimal dibandingkan dengan algoritma-algoritma lainnya.

Mendapatkan representasi pecahan Mesir dari suatu bilangan rasional *q* menggunakan metode ini dilakukan dengan mengambil pecahan unit pertama berupa pecahan satuan terbesar yang lebih kecil dari *q*, dan kemudian mengulang proses tersebut untuk hasil pengurangan *q* dengan pecahan satuan yang dipilih tersebut. Jika *q*>1, terlebih dulu dilakukan pemisahan bagian bulat dari q (menggunakan fungsi *floor*, $\lfloor q \rfloor$) kemudian memproses bagian pecahannya (yaitu $q - \lfloor q \rfloor$) menggunakan metode yang telah disebutkan di atas.

Pseudocode untuk algoritma ini adalah sebagai berikut:

```
function GreedyPart(input q : pecahan) ->
       pecahan;
       body
           if ((penyebut(q) = 1) or
       (pembilang(q) = 1)) then
                   -> 0
           else if (q < 0) or (q > 1)
       then
                   -> q - floor(q)
           else
                   -> q - (1 + (1 div q))
           end if
end function

function SubtractShifted(input ar : array
       of pecahan) -> array of
       pecahan;elemen ke-(i+1) }
       declarations
           i   : integer
           tmp : array of pecahan
       body
           for i <- 1 to (nbElmt(ar)-2)
       do
                tmp[i] <- ar[i] - ar[i+1]
           end for
           -> tmp
end function

function EgyptGreedy(input q : pecahan) -
       > array of pecahan;
       declarations
           i    : integer
           tmp  : array of pecahan
           ftmp : pecahan
       body
           i    <- 1
           ftmp <- q
           repeat
               tmp[i] <- ftmp
               ftmp   <- GreedyPart(ftmp)
               if (tmp[i] =
       GreedyPart(ftmp)) then
                    i <- i + 1
               end if
           until (tmp[i] =
       GreedyPart(ftmp))
           -> SubtractShifted(tmp)
end function
```

Untuk melihat kebenaran dari algoritma ini dan menganalisa performanya, misalkan masukan untuk fungsi `EgyptGreedy` ini adalah sebuah pecahan *x/y*. Setelah langkah pertama, sisa bilangan yang didapat adalah $(y \bmod x)/y\lfloor y/x \rfloor$ dengan pembilang yang lebih kecil dari sebelumnya. Begitu seterusnya, pembilang akan terus berkurang pada setiap langkah, sehingga tidak

akan terjadi kalang tanpa henti. Satu-satunya kesalahan yang mungkin terjadi adalah jika ada dua pecahan satuan yang sama, namun kasus ini tidak akan dijumpai karena penyebut dari pecahan satuan terus bertambah (sehingga pecahan semakin kecil).

Karena pembilang selalu berkurang, maka jumlah pecahan satuan maksimum yang diperlukan untuk representasi pecahan Mesir dari *x/y* adalah *x* buah. Sementara itu penyebut biasanya berpangkat dua dari langkah sebelumnya, sehingga penyebut terbesar yang mungkin adalah *y*^(2^*x*) atau lebih umum *y*^(2^*k*) dengan *k* adalah jumlah pecahan satuan yang digunakan. Jumlah operasi yang dilakukan dalam algoritma ini berbanding lurus dengan *k*, akan tetapi operasi tersebut dapat melibatkan bilangan-bilangan yang sangat besar. Secara umum algoritma ini memiliki kompleksitas O(log *x*).

Sebagai contoh kasus, dengan menggunakan algoritma ini pecahan biasa 18/23 akan direpresentasikan sebagai 1/2 + 1/4 + 1/31 + 1/2852. Pada kasus ini algoritma ini menunjukkan hasil yang cukup optimal, namun sebagai contoh pecahan biasa 31/311 dengan algoritma ini akan direpresentasikan dengan 10 buah pecahan satuan dengan pembilang terbesar merupakan bilangan ber-digit lebih dari 500. Metode-metode lain ditemukan dapat memberikan solusi yang lebih optimal untuk kasus ini[4].

2.1 Algoritma *Harmonic*

Algoritma *greedy* di atas memperlakukan bagian bulat dan bagian pecahan dari sebuah bilangan rasional secara berbeda. Dalam algoritma harmonic ini, kedua bagian tersebut tidak dipisahkan. Tanpa memisahkan kedua bagian ini pun akan selalu dapat ditemukan pecahan satuan yang lebih kecil dari pada *x/y* maupun pecahan satuan sebelumnya meskipun *x/y* > 1. Akan tetapi, metode ini membutuhkan tambahan larik, yaitu untuk menyimpan batas (*bound*) untuk penyebut. Fungsi EgyptHarmonic dengan demikian membutuhkan dua argumen, yaitu bilangan yang akan direpresentasikan dalam pecahan Mesir dan penyebut terbesar yang sudah terdapat dalam himpunan solusi.

Pseudocode untuk algoritma ini sebagai berikut:

```
function HarmonicPart(input <q : pecahan,
       d : integer>) -> <pecahan,
       integer>;
       body
           if ((q = 0) or (pembilang(q) =
       1)) then
               -> <0, d>
           else
               -> <(q-1) / max(d, 1 + (1
       div q)), max(d, 1 + (1 div q)) +
       1>
           end if
end function

function EgyptHarmonic(input <q :
       pecahan, d : integer>) -> array of
       pecahan;
       declarations
           i    : integer
           tmp  : array of pecahan
           ftmp : pecahan
           denom: array of integer
       body
           i    <- 1
           ftmp <- q
           repeat
               tmp[i]   <- ftmp
               denom[i] <- 1
               ftmp     <-
       HarmonicPart(ftmp, d)
               if (tmp[i] =
       Harmonicart(ftmp)) then
                    i <- i + 1
               end if
           until (tmp[i] =
       HarmonicPart(ftmp))
           ->
       transpose(SubtractShifted(tmp))
end function
```

Algoritma ini membangun deret harmonis 1/2+1/3+1/4+1/5+... hingga mendapatkan hasil penjumlahan yang lebih besar dari pada *q*, kemudian memproses sisanya dengan metode *greedy*. Kebenaran algoritma ini dapat dibuktikan dengan cara yang sama dengan algoritma sebelumnya. Kompleksitas algoritma ini adalah O(*x/y* + log *d*).

Sebagai contoh kasus, dengan masukan 18/23 metode ini akan menghasilkan 1/5 + 1/6 + 1/7 + 1/8 + 1/9 + 1/28 + 1/794 + 23010120.

2.1 Algoritma *Odd Greedy*

Telah diketahui bahwa untuk setiap pecahan biasa *x/y* dengan *y* ganjil dapat direpresentasikan dalam pecahan Mesir dengan setiap penyebutnya bernilai ganjil[4]. Demikian juga jika y genap, maka paling sedikit satu dari pecahan-pecahan satuan yang membentuk pecahan Mesirnya memiliki penyebut genap. Dengan demikian nampak bahwa cara paling *straightforward* untuk mencari representasi berpenyebut ganjil adalah dengan memodifikasi algoritma *greedy* untuk hanya menggunakan penyebut ganjil, akan tetapi metode ini belum dapat dibuktikan sepenuhnya benar.

Pseudocode sebagai berikut:

```
function OddGreedyPart(input <q :
       pecahan, d : integer>) ->
       <pecahan, integer>;
       body
           if ((q = 0) or (pembilang(q) =
       1)) then
               -> <0, d>
           else if (odd(max(d, 1 + (1 div
       q)))) then
               -> <(q-1) / max(d, 1 + (1
       div q)), max(d, 1 + (1 div q)) +
       1>
           else if (even(max(d, 1 + (1
       div q)))) then
               -> <(q-1) / (max(d, 1 + (1
       div q)) + 1), max(d, 1 + (1 div
       q)) + 2>
end function

function EgyptOddGreedy(input <q :
       pecahan, d : integer>) -> array of
       pecahan;
       declarations
           i    : integer
           tmp  : array of pecahan
           ftmp : pecahan
           denom: array of integer
       body
           i    <- 1
           ftmp <- q
           repeat
               tmp[i]   <- ftmp
               denom[i] <- 1
               ftmp     <-
       OddGreedyPart(ftmp, d)
               if (tmp[i] =
       OddGreedyPart(ftmp)) then
                   i <- i + 1
               end if
           until (tmp[i] =
       OddGreedyPart(ftmp))
           ->
       transpose(SubtractShifted(tmp))
end function
```

Tidak seperti algoritma *greedy* biasa, pembilang dari pecahan yang tersisa tidak selalu berkurang pada setiap langkah. Dua hal menjadi penyebabnya: pertama, seperti halnya algoritma *harmonic*, terdapat argumen *d* untuk memastikan setiap pecahan yang dihasilkan berbeda. Argumen *d* ini digunakan hingga deret 1/3+1/5+1/7+1/9+... lebih besar dari pada *q*. Pada tahap ini, pembilang secara umum akan bertambah. Kedua, algoritma ini akan menambahkan penyebut untuk mencegah adanya pecahan satuan yang sama.

Sebagai contoh, dengan algoritma ini masukan 10/39 akan menghasilkan keluaran 1/5 + 1/19 + 1/265 + 1/196365.

Masalah yang ada dalam algoritma ini adalah memastikan kalang berhenti. Suatu penjelasan heuristik menyatakan bahwa kalang akan selalu berhenti.

3. KESIMPULAN

Masih terdapat banyak sekali algoritma untuk membangkitkan pecahan Mesir dari pecahan biasa selain metode-metode yang telah diuraikan di atas. Akan tetapi belum dapat dibuktikan bahwa algoritma-algoritma tersebut memberikan hasil yang optimal.

Metode pertama dan kedua yang dibahas pada masalah ini, yaitu *greedy* dan *harmonic*, selalu memberikan solusi yang benar, namun pada banyak kasus nampak bahwa solusi tersebut tidak optimal. Metode ketiga yang dibahas, yaitu *odd greedy*, terbukti optimal dalam lebih banyak kasus dibandingkan dengan kedua metode sebelumnya, namun tidak terbukti benar untuk semua kasus, bahkan ada kemungkinan di mana kalang tidak berhenti.

REFERENSI

[1] Weisstein, Eric W. "Egyptian Fraction." From MathWorld – A Wolfram Web Resource. http://mathworld.wolfram.com/EgyptianFraction.html (tanggal akses 19 Mei 2008 pukul 17.00)

[2] Wells, D. *The Penguin Dictionary of Curious and Interesting Numbers.* Middlesex, England: Penguin Books, p. 29, 1986.

[3] Hoffman, P. The Man Who Loved Only Numbers: The Story of Paul Erdős and the Search for Mathematical Truth. New York: Hyperion, pp. 153-157, 1998.

[4] Eppstein, David. “Egyptian Fractions”. http://www.ics.uci.edu/~eppstein/numth/egypt/ (tanggal akses 19 Mei 2008 pukul 17.00)