

Optimalisasi Algoritma Greedy dalam Permasalahan Penukaran Uang

Muhammad Ibnu Prayogi 120140152 (*Author*)

Program Studi Teknik Informatika

Prodi Teknik Informatika

Institut Teknologi Sumatera, Jalan Terusan Ryacudu, Lampung Selatan

E-mail (gmail): muhammadibnuprayogi24@gmail.com

Abstract—

Algoritma greedy merupakan algoritma paling mangkus dalam penyelesaian masalah optimasi baik menentukan solusi maksimum, maupun minimum. Masalah penukaran uang adalah salah satu masalah yang bisa diselesaikan dengan efektif oleh algoritma greedy. Namun, algoritma tersebut ternyata memiliki kelemahan dimana solusi yang didapatkan tidak selalu merupakan solusi terbaik dari setiap permasalahan. Hal yang sama terjadi pada masalah penukaran uang. Hal ini disebabkan karena algoritma greedy tidak membangkitkan seluruh kemungkinan solusi karena keterbatasan kemampuan dalam mencari solusi permasalahan secara kombinatorikal. Akan tetapi, kelemahan tersebut dapat diminimalisasi dengan optimalisasi berupa modifikasi algoritma greedy sehingga meningkatkan kemungkinan mendapatkan solusi optimum global pada tiap permasalahan

Kata kunci : Algoritma Greedy, penukaran uang, brute force, exhaustive search

I. PENDAHULUAN

Permasalahan optimasi merupakan permasalahan yang sering terjadi dalam kehidupan sehari-hari. Permasalahan ini tidak sekedar mencari solusi dalam setiap proses penyelesaiannya, tetapi solusi yang ditemukan haruslah solusi terbaik. Dalam hal ini, ada dua konteks solusi terbaik yang dimaksud, yaitu solusi maksimum dan solusi minimum. Solusi maksimum merupakan himpunan solusi dari sebuah permasalahan yang memiliki value paling besar, panjang, ataupun banyak. Sedangkan, solusi minimum merupakan himpunan solusi yang memiliki nilai paling kecil, sedikit, atau pendek dalam sebuah permasalahan.

Algoritma yang dianggap paling efektif dalam penyelesaian algoritma ini adalah algoritma greedy. Algoritma greedy merupakan algoritma yang sangat mangkus dalam menyelesaikan permasalahan optimasi. Algoritma greedy bekerja bertahap dimana tiap tahapnya akan diambil sebuah solusi optimum lokal yang diharapkan akan menuju solusi optimum global pada akhirnya. Algoritma ini bekerja tanpa memikirkan konsekuensi kedepannya setelah pengambilan optimum lokal. Beberapa permasalahan yang dapat diselesaikan oleh algoritma ini, yaitu pencarian jarak terpendek, persoalan knapsack, dan pengurutan menggunakan selection sort

Penukaran uang merupakan salah satu permasalahan optimasi yang dapat diselesaikan dengan algoritma greedy, solusi optimum dari permasalahan ini adalah jumlah pecahan uang paling sedikit yang dapat ditukar dengan nilai tertentu. Akan tetapi, penyelesaian permasalahan ini dengan algoritma greedy tidak selalu menemukan solusi optimal. Oleh karena itu, penelitian ini dilakukan untuk mengoptimisasi algoritma greedy untuk menemukan solusi global untuk permasalahan ini. Hal ini dapat dilakukan dengan cara melakukan kombinasi antara algoritma greedy dengan algoritma lainnya guna mengantisipasi solusi lokal yang tidak mewakili solusi global dalam permasalahan penukaran uang. Harapan dari ujian ini adalah tercipta sebuah algoritma yang menyempurnakan algoritma greedy dalam pemecahan masalah penukaran uang [1]

II. TEORI DASAR

A. Algoritma

Algoritma adalah sebuah urutan logis yang berisi langkah langkah dalam penyelesaian masalah yang tersusun secara terstruktur dan sistematis. Pada awal kemunculan istilah ini, algoritma merujuk pada aturan aturan aritmetis untuk menyelesaikan permasalahan. Akan tetapi, pada abad ke-18 istilah algoritma berkembang sehingga makna meluas menjadi suatu langkah atau prosedur yang diperlukan untuk menyelesaikan suatu permasalahan dalam kehidupan sehari-hari. Kata algoritma sendiri berasal dari latinisasi nama seorang tokoh ahli matematika bernama AL-Khawarizmi. [2]



Sumber: <https://suaramuhammadiyah.id/>

Ada lima ciri-ciri penting yang harus dimiliki sebuah algoritma, yaitu berupa finiteness, definiteness, masukan, keluaran, dan efektivitas.

1. Finiteness, hal ini menyatakan bahwa algoritma harus memiliki kondisi akhir untuk semua opsi langkah yang mungkin
2. Definiteness, hal ini menyatakan bahwa tiap langkah harus dideskripsikan dengan cara yang jelas
3. Masukan, hal ini merupakan besaran yang diberikan pada awal algoritma, sebuah algoritma mungkin memiliki satu atau lebih masukan, bahkan tidak memerlukan masukan sama sekali
4. Keluaran, hal ini merupakan produk dari sebuah algoritma. Algoritma dapat memiliki satu atau lebih keluaran pada akhir setiap langkahnya
5. Efektivitas, Hal ini berarti bahwa setiap proses yang dijalankan dalam algoritma harus efektif dan efisien sehingga memakan waktu sesingkat mungkin dengan keluaran yang optimal

Terdapat beberapa klasifikasi algoritma yang dibagi berdasarkan alasan tersendiri. Salah satu cara dalam melakukan pengklasifikasian tersebut adalah berdasarkan paradigma dan metode yang digunakan dalam perancangan algoritma tersebut. Beberapa paradigma yang digunakan untuk menyusun suatu algoritma antara lain:

1. Divide and Conquer, merupakan paradigma untuk membagi suatu permasalahan yang besar menjadi permasalahan-permasalahan yang kecil. Pembagian masalah ini dilakukan secara terus-menerus sampai ditemukan bagian masalah yang kecil dan mudah untuk dipecahkan.
2. Dynamic programming, paradigma pemrograman dinamik akan sesuai jika digunakan pada suatu masalah yang mengandung sub-struktur yang optimal dan mengandung beberapa bagian permasalahan yang tumpang tindih. Paradigma ini sekilas terlihat mirip dengan paradigma divide and conquer, sama-sama mencoba untuk membagi permasalahan menjadi sub permasalahan yang lebih kecil, tapi secara intrinsic ada perbedaan dari karakter permasalahan yang dihadapi.
3. Algoritma Greedy, merupakan paradigma yang mirip dengan pemrograman dinamik, namun jawaban dari setiap submasalah tidak perlu diketahui dari setiap tahap, dan menggunakan pilihan apa yang terbaik pada saat itu.
4. Search and enumeration, merupakan paradigma pemodelan yang memberikan aturan tertentu dalam pemecahan masalah dan optimalisasi.

B. Algoritma Greedy

Algoritma greedy merupakan salah satu metode yang paling populer dalam penyelesaian masalah optimasi. Dalam masalah optimasi, terdapat dua jenis persoalan, yaitu minimasi dan maksimasi. Solusi maksimum merupakan himpunan solusi dari sebuah permasalahan yang memiliki value paling besar, panjang, ataupun banyak. Sedangkan, solusi minimum merupakan himpunan solusi yang memiliki nilai paling kecil, sedikit, atau pendek dalam sebuah permasalahan. [3]

Algoritma greedy memiliki beberapa komponen penting yang berperan dalam proses pemecahan masalah optimasi secara lebih mangkus. Berikut ini penjelasan dari beberapa komponen tersebut:

1. Himpunan Kandidat C

Himpunan ini berisi elemen-elemen yang akan menjadi calon pembentuk solusi optimum di akhir proses penyelesaian. Pada setiap langkah, akan dipilih satu kandidat sebagai calon solusi optimum

2. Himpunan Solusi S

Himpunan ini berisi elemen-elemen kandidat yang terpilih pada tiap tahap sebagai solusi persoalan. Dengan kata lain, himpunan ini merupakan *sub-set* (Himpunan Bagian) dari himpunan kandidat

3. Fungsi Seleksi

Fungsi ini berguna untuk melakukan pemilihan terhadap salah satu elemen dari himpunan kandidat yang paling memungkinkan dan mendekati solusi optimum. Kandidat yang terpilih pada suatu tahap tidak akan pernah dipertimbangkan lagi pada Fungsi kelayakan (dieksekusi)

4. Fungsi Kelayakan

Fungsi ini bekerja dengan melakukan pemeriksaan kelayakan terhadap himpunan kandidat yang dipilih sebagai solusi. Apabila kandidat tersebut memenuhi syarat atau batasan, maka kandidat tersebut dimasukkan dalam solusi. Sedangkan, apabila elemen tersebut tidak memenuhi syarat atau constrain, maka elemen tersebut akan dibuang.

5. Fungsi Objektif

Fungsi objektif ini adalah suatu fungsi yang memaksimalkan atau meminimalkan nilai solusi. Dengan kata lain, algoritma greedy berisi pencarian untuk serangkaian himpunan bagian. S, dari set kandidat, Himpunan S ini harus memenuhi beberapa kriteria Solusi yang ditentukan. S kemudian dioptimalkan oleh fungsi objektif. Ada kalanya solusi optimum global yang diperoleh dari algoritma greedy yang diharapkan sebagai solusi optimum dari persoalan, belum tentu merupakan solusi optimum (terbaik), tetapi solusi sub-optimum atau pseudo-optimum. Hal ini

dikarenakan algoritma greedy tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada dan terdapat beberapa fungsi seleksi yang berbeda, yaitu jika fungsi seleksi tidak identik dengan fungsi obyektif karena fungsi seleksi biasanya didasarkan pada fungsi obyektif. Sehingga harus dipilih fungsi yang tepat jika menginginkan algoritma menghasilkan solusi optimal atau nilai yang optimum.

Algoritma greedy memiliki kelebihan dan kelemahan dalam proses penyelesaiannya. Kelebihan algoritma greedy terdapat dalam kecepatannya dalam penyelesaian masalah karena algoritma ini tidak perlu membangkitkan seluruh kemungkinan solusi dan hanya berusaha untuk menemukan solusi optimum lokal yang diharapkan berakhir pada solusi maksimum global. Algoritma ini juga tidak mempertimbangkan konsekuensi dari setiap opsi pilihan pada tahapan selanjutnya sehingga proses pemilihan solusi berjalan lebih cepat dan ringkas [4]

Sementara itu, kekuarngannya terletak pada kemampuannya dalam penemuan solusi optimum yang tidak mencapai 100%. Ada beberapa kasus yang memungkinkan algoritma greedy tidak menampilkan solusi yang optimum global seperti yang pasti ditemukan menggunakan algoritma brute force. Hal ini merupakan konsekuensi dari pembangkitan solusi permasalahan pada algoritma greedy yang tidak menyeluruh sehingga ada kemungkinan solusi yang tidak terseleksi sebagai opsi dalam algoritma greedy

Berikut adalah contoh *pseudo-code* Algoritma greedy :

Procedure greedy(input C : himpunan – kandidat , output S : himpunan-solusi)

{ Menentukan solusi optimum dari persoalan optimasi dengan algoritma greedy

Masukan : himpunan kandidat C

Keluaran : himpunan solusi S }

Deklarasi

x : kandidat;

Algoritma

S $\leftarrow \{\}$ {inisialisasi S dengan kosong}

while (belum SOLUSI(S)) and (C $\neq \{\}$) do

 x \leftarrow SELEKSI (C); {pilih sebuah kandidat dari C}

 C \leftarrow C-{x} {elemen himpunan kandidat berkurang satu}

if LAYAK(S \cup {x}) then

 S \leftarrow S \cup {x}

endif

endwhile

{solusi(S) sudah direroleh or C = { } }

C. Optimasi

Optimasi menurut KBBI diartikan sebagai pengoptimalan, yaitu proses, cara, pembuatan untuk menghasilkan yang paling baru. Analisis optimasi merupakan proses pengelolaan data awal dengan me

Optimasi adalah usaha untuk mendapatkan hasil yang maksimal atau optimal dalam setiap hal yang dikerjakan dengan proses tertentu. Secara matematis, optimasi merupakan suatu cara untuk mendapatkan nilai ekstrim maksimum maupun minimum dari suatu fungsi tertentu dengan menggunakan faktor-faktor pembatas sebagai acuannya

Penyelesaian masalah dengan menggunakan metode optimasi merupakan pengambilan keputusan optimal sehingga didapatkan hasil yang sesuai dengan kondisi yang harus dipenuhi. Banyak metode yang digunakan untuk menyelesaikan permasalahan optimasi antara lain *Algoritma Greedy, Calculus, Dynamic Programming, Linier Programming, Geomtry, dan Inventory Theory*.

Dari penjelasan diatas, dapat disimpulkan bahwa optimalisasi merupakan suatu proses atau kegiatan untuk meningkatkan atau mengoptimalkan suatu solusi dari sebuah permasalahan sehingga menjadi lebih atau sepenuhnya sempurna, fungsional, atau lebih efektif serta mencari solusi terbaik (optimum global) dari beberapa masalah agar tercapai tujuan sebaik baiknya sesuai dengan Batasan-batasan atau kriteria tertentu [3]

D. Penukaran Uang

Penukaran uang merupakan masalah yang sering terjadi dalam kehidupan sehari-hari di masyarakat. Dalam kasus optimasi, penukaran uang dilakukan dari suatu nilai mata uang yang lebih besar menjadi lebih kecil dengan menggunakan jumlah uang pecahan seminimal mungkin. Maka dari itu, semakin sedikit jumlah uang pecahan dalam pemukaran akan membuat suatu solusi tersebut semakin optimal.

Masalah ini dapat diselesaikan dengan menggunakan beberapa algoritma, salah satunya adalah exhaustive search. Exhaustive search merupakan algoritma pencarian dengan mengumpulkan atau membangkitkan seluruh kemungkinan solusi dalam bentuk kombinatorikal. Dengan menggunakan metode ini, solusi optimum pasti akan ditemukan. Akan tetapi, metode ini membutuhkan banyak waktu sehingga dapat membuang sumber daya dan efisiensi dari suatu pekerjaan.

Selain exhaustive search, algoritma yang dianggap paling efektif dan mangkus dalam menyelesaikan masalah ini adalah algoritma greedy. Penyelesaian masalah penukaran uang dengan algoritma greedy dilakukan dengan pengurutan uang mulai dari pecahan terbesar sampai terkecil. Kemudian dilakukan penyeleksian mulai dari koin terbesar sampai terkecil dimana poin yang diambil sebagai solusi optimum lokal adalah pecahan dengan nilai terbesar dan tidak melebihi jumlah penukaran yang ditentukan. Apabila pada tahap berikutnya bilangan yang sama masih memenuhi *constrain*, maka pecahan yang sama diambil sebagai solusi di tahap tersebut. Akan tetapi, jika uang yang sama sudah melebihi jumlah penukaran

yang ditentukan, maka pecahan yang diambil adalah pecahan berikutnya dengan value terbesar kedua. Begitu seterusnya sampai uang pecahan jumlahnya sesuai atau mendekati *constrain* yang ditentukan [5]

III. ANALISIS DAN PEMBAHASAN

A. Simulasi Penukaran Koin

```
function CoinExchange (input C : himpunan_koin, A : integer) → himpunan_koin
{ mengembalikan koin-koin yang total nilainya = A, tapi jumlah koinnya
minimum}
Deklarasi
    S : himpunan_koin
    x : koin
Algoritma
    S ← {}
    while (Σ(nilai semua koin didalam S) ≠ A) and (C ≠ {}) do
        x ← koin yang mempunyai nilai terbesar
        C ← C - {x}
        if (Σ(nilai semua koin didalam S) + nilai koin x ≤ A)
            then
                S ← S U {x}
            endif
        endwhile
    if (Σ(nilai semua koin didalam S) = A)
        then
            return S
        else
            write ('tidak ada solusi')
        endif
    endfunction
```

Sumber: <https://algorithmsanalysis.blogspot.com/>

Gambar diatas merupakan pseudo-code dari penyelesaian masalah penukaran uang dengan menggunakan algoritma greedy. Algoritma tersebut sudah mencakup segala komponen yang diperlukann dalam algoritma greedy, diantaranya himpunan kandidat, himpunan solusi, fungsi seleksi, fungsi kelayakan, dan fungsi obyektif.

Berikut contoh kasus penyelesaian penukaran uang dengan algoritma greedy:

Akan dilakukan Penukaran uang sebesar 32 satuan dengan menggunakan uang pecahan 1, 5, 10, dan 25. Solusi minimum dari masalah ini adalah jumlah pecahan paling sedikit yang bisa menghasilkan jumlah koin mencapai 32

N = Variabel penampung jumlah total pecahan

25	10	5	1
----	----	---	---

N = 25, N < 32 (valid)

25	10	5	1
----	----	---	---

N = 25 + 25, N > 32 (tidak valid)

25	10	5	1
----	----	---	---

N = 25 + 5, N < 32 (valid)

25	10	5	1
----	----	---	---

N = 25 + 5 + 5, N > 32 (tidak valid)

25	10	5	1
----	----	---	---

N = 25 + 5 + 1, N < 32 (valid)

25	10	5	1
----	----	---	---

N = 25 + 5 + 1 + 1, N = 32 (Program Selesai)

Solusi = 25 + 5 + 1 + 1 (4 pecahan)

Berikut ini merupakan ilustrasi penyelesaian masalah diatas dengan paradigma greedy, uang pecahan akan di urutkan terlebih dahulu mulai dari yang terkecil sampai yang terbesar. Kemudian, variable temporary digunakan untuk melakukan increament elemen-elemen kandidat yang akan dipilih dan menjadi himpunan solusi dimana dalam ilustrasi tersebut digunakan N sebagai temporary. Algoritma ini akan melakukan assign nilai terbesar dari list tersebut kedalam nilai N yang mulanya 0, kemudian dilakukan pengecekan kelayakan. Apabila nilai N setelah penambahan kurang dari 32, maka elemen kandidat tersebut layak dimasukan dalam himpunan solusi. Kemudian program akan melakukan assign dengan menambahkan nilai N yang sebelumnya dengan bilangan yang sama (terbesar). Jika setelah penambahan nilai N masih kurang dari 32, maka bilangan tersebut dimasukan Kembali dalam himpunan solusi. Akan tetapi, jika jumlah N melebihi 32, bilangan yang diambil akan bergeser pada bilangan terbesar kedua dan kemudian dilakukan pengecekan. Langkah tersebut berulang sampai nilai memenuhi jumlah yang diinginkan (32).

Pada ilustrasi tersebut, kandidat yang di seleksi ditandai dengan warna merah, kemudian akan di assign pada nilai N sebagai bentuk fungsi seleksi yang kemudian akan diuji kelayakannya oleh fungsi kelayakan untuk dimasukan dalam himpunan solusi. Pada kasus ini, solusi yang didapatkan untuk menukar uang sebesar 32 satuan adalah dengan pecahan 25 (1 buah), 5 (1 buah), dan 1 (2 buah) sehingga menghasilkan total 4 buah uang pecahan.

Solusi tersebut nilainya sama dengan solusi yang dihasilkan oleh algoritma brute force, yakni exhaustive search. Dengan ini, berarti algoritma brute force berhasil untuk membangkitkan solusi optimum global dari kasus penukaran uang tersebut. Akan tetapi, bukan berarti algoritma ini selalu menemukan solusi optimum global dari sebuah permasalahan. Ada beberapa kasus yang membuktikan bahwa algoritma greedy tidak menghasilkan solusi optimum global dalam masalah penukaran uang.

Contohnya adalah penukaran uang sebesar 15 satuan dengan uang pecahan 10, 7, dan 1. Berikut penyelesaiannya menggunakan algoritma greedy:

N=0

10	7	1
----	---	---

N=10, N<15 (valid)

10	7	1
----	---	---

N=10+10, N>15 (tidak valid)

10	7	1
----	---	---

N=10+7, N>15 (tidak valid)

10	7	1
----	---	---

N=10+1, N<15 (valid)

10	7	1
----	---	---

N=10+1+1, N<15 (valid)

10	7	1
----	---	---

N=10+1+1+1, N<15 (valid)

10	7	1
----	---	---

N=10+1+1+1+1, N<15 (valid)

10	7	1
----	---	---

N=10+1+1+1+1+1, N<15 (valid)

10	7	1
----	---	---

N=10+1+1+1+1+1+1, N<15 (valid)

10	7	1
----	---	---

N=10+1+1+1+1+1+1+1, N=15 (Selesai)

Solusi = 10+1+1+1+1+1+1+1 (8 pecahan)

Dengan algoritma greedy, solusi yang ditemukan adalah menukarkan uang sebesar 15 satuan dengan uang pecahan 10 (1 buah) dan 1 (7 buah) sehingga total menghasilkan 8 pecahan. Apabila dibandingkan dengan solusi optimal yang dihasilkan dengan metode brute force, solusi ini bukanlah solusi optimum global. Penyelesaian kasus ini dengan exhaustive search yang mendapatkan solusi pecahan 7 (2 buah) dan 1 (1 buah) sehingga menghasilkan jumlah pecahan yang lebih minimum, yaitu tiga uang pecahan. Hal ini membuktikan bahwa algoritma greedy memang tidak selalu menghasilkan solusi optimum global dalam setiap permasalahan penukaran uang. Akan tetapi, solusi optimum lokal yang didapatkan dengan algoritma greedy dapat digunakan sebagai dasar pengembangan algoritma yang lebih efektif dalam menemukan optimum global

B. Optimalisasi Algoritma Greedy

Optimalisasi ini dilakukan untuk meningkatkan kemungkinan algoritma greedy dalam menemukan solusi optimum global dalam menyelesaikan masalah. Proses optimalisasi ini dapat dilakukan dengan mengombinasikan algoritma greedy dengan algoritma lain ataupun memodifikasi algoritma greedy itu sendiri. Optimalisasi dilakukan dengan membangkitkan lebih banyak elemen elemen kandidat yang akan menjadi solusi dengan menggunakan solusi optimum lokal pada algoritma greedy yang dilakukan sebelumnya sebagai dasar untuk mengeliminasi beberapa solusi yang tidak diperlukan sehingga algoritma yang tercipta tetap akan lebih cepat dibandingkan algoritma brute force.

Berikut ini contoh perbandingan hasil solusi masalah penukaran uang menggunakan algoritma greedy dan solusi optimum global dengan exhaustive search:

- Koin: 5, 4, 3, dan 1
Uang yang ditukar = 7.
Solusi greedy: $7 = 5 + 1 + 1$ (3 koin) → tidak optimal
Solusi optimal: $7 = 4 + 3$ (2 koin)
- Koin: 10, 7, 1
Uang yang ditukar: 15
Solusi greedy: $15 = 10 + 1 + 1 + 1 + 1 + 1$ (6 koin)
Solusi optimal: $15 = 7 + 7 + 1$ (hanya 3 koin)
- Koin: 15, 10, dan 1
Uang yang ditukar: 20
Solusi greedy: $20 = 15 + 1 + 1 + 1 + 1 + 1$ (6 koin)
Solusi optimal: $20 = 10 + 10$ (2 koin)

Dapat dilihat bahwa pada ketiga kasus tersebut, algoritma greedy tidak menghasilkan solusi optimum. Dan pada ketiga kasus tersebut, terdapat kesamaan pola pada solusi optimal yang dihasilkan, yaitu nilai terbesar pada solusi optimal bukanlah nilai pecahan terbesar dari himpunan kandidat. Pada contoh pertama (a), pecahan terbesar yang menjadi solusi optimum adalah 4. Pada contoh kedua, nilai pecahan terbesar pada solusi adalah 7. Sedangkan pada contoh ketiga (c), pecahan terbesar pada solusi adalah 10. Kesamaan pola tersebut menimbulkan kesimpulan bahwa pengambilan nilai pecahan terbesar tidak akan menghasilkan solusi optimum global, sedangkan pengambilan nilai terbesar kedua justru menghasilkan solusi yang optimal. Berdasarkan hal tersebut, optimalisasi algoritma greedy dapat dilakukan

Optimalisasi tersebut adalah dengan melakukan perulangan algoritma greedy. Contoh pada kasus sebelumnya dengan penukaran uang sebesar 15 dengan pecahan 10, 7, dan 1.

- Pada tahap pertama, list yang dipakai sebagai calon kandidat adalah 10, 7, dan 1 sehingga menghasilkan solusi optimum lokal dengan 8 buah pecahan ($10+1+1+1+1+1+1+1$) seperti pada gambar diatas. Kemudian, solusi optimum lokal tersebut di jadikan Batasan dalam penentuan solusi di tahap berikutnya

2. Pada tahap kedua, nilai pecahan terbesar pada list sebelumnya dihilangkan dari himpunan kandidat sehingga himpunan kandidat hanya berisi pecahan 7 dan 1, sehingga akan menghasilkan solusi 3 buah pecahan ($7+7+1$)

Solusi Optimum Lokal = 8 buah pecahan

7	1
---	---

$N=7$, $N < 15$ (valid)

7	1
---	---

$N=7+7$, $N < 15$ (valid)

7	1
---	---

$N=7+7+7$, $N > 15$ (tidak valid)

7	1
---	---

$N=7+7+1$, $N=15$ (Selesai)

Solusi = $7+7+1$ (3 buah pecahan)

3. Apabila jumlah pecahan yang didapatkan pada solusi sebelumnya lebih sedikit (minimum) dari solusi optimum lokal pada tahap pertama, maka solusi pada tahap kedua menjadi optimum lokal yang baru dan akan digunakan pada tahap berikutnya
4. Pada optimalisasi ini dapat ditambahkan sebuah konstrain yaitu, Nilai Pecahan terbesar dalam list kandidat pada tiap tahap harus lebih besar dari jumlah uang yang akan ditukarkan dibagi dengan solusi optimum pada tahap sebelumnya.

Contoh:

Pada kasus sebelumnya, Jumlah uang yang ingin ditukarkan adalah 15

Pada tahap 1, didapatkan solusi optimum = 8 buah pecahan

Pada tahap 2, nilai pecahan yang terbesar yang dapat ditukar adalah 7

Oleh karena 7 lebih besar dari $15/8$, maka tahap 2 dapat dilanjutkan.

Pada tahap 2, didapatkan solusi optimum 3 buah pecahan

Pada tahap 3, pecahan terbesar yang dapat ditukar adalah 1,

Karena 1 tidak lebih besar dari $15/3$, maka tahap 3 tidak perlu dilanjutkan karena sudah pasti tidak bisa menghasilkan solusi yang lebih optimum dari tahap berikutnya

A = jumlah uang yang ingin ditukar

N = Solusi optimum pada tahap sebelumnya

B = Nilai pecahan terbesar yang dapat ditukar pada tahap sekarang

$$\text{Fungsi kelayakan} = B > A/N$$

Kemudian, cara lain yang bisa dilakukan adalah menempatkan solusi optimum lokal yang diperoleh algoritma greedy sebagai Batasan dalam melakukan exhaustive search sehingga dapat mengeliminasi beberapa kemungkinan solusi yang tidak diperlukan. Hal ini tidak termasuk heuristic search karena kandidat yang dieliminasi dapat dipastikan tidak layak untuk menjadi solusi. Metode ini cenderung lebih tidak mangkus dibandingkan optimalisasi sebelumnya dengan mengganti nilai pecahan koin terbesar. Akan tetapi, tingkat keberhasilannya bisa mencapai 100%

IV

KESIMPULAN

Berdasarkan pengujian dan analisis terhadap penerapan algoritma greedy dalam masalah penukaran uang, dapat disimpulkan bahwa algoritma greedy merupakan algoritma paling mangkus dalam penyelesaian masalah tersebut. Akan tetapi, solusi yang dihasilkan tidak selalu merupakan solusi optimum global sebagaimana yang pasti ditemukan dengan algoritma brute force. Akan tetapi, solusi yang didapatkan adalah solusi optimum lokal dengan harapan bahwa solusi tersebut merupakan solusi terbaik dalam pemecahan masalah ini.

Optimalisasi dapat dilakukan dengan melakukan modifikasi terhadap algoritma greedy dengan melakukan perulangan kemudian memanipulasi list kandidat dengan menghilangkan nilai paling besar pada himpunan kandidat pada setiap tahap perulangan dengan menyimpan solusi optimum lokal pada tiap tahap guna dipakai sebagai Batasan sekaligus pembanding terhadap solusi optimum lokal yang akan didapatkan pada tahap berikutnya.

Link Video Youtube

<https://youtu.be/rAhr0LMu-Q>

Ucapan Terimakasih

Alhamdulillah, puji syukur penulis panjatkan kehadirat Allah swt karena atas berkat dan rahmatnya tugas strategi algoritma tentang pembuatan technical report dapat terselesaikan. Technical report berjudul "Optimalisasi Algoritma Greedy dalam penyelesaian masalah Penukaran Uang" ini merupakan salah satu bentuk pemenuhan tugas kecil strategi algoritma pada semester 4 prodi Teknik informatika. Oleh karena itu, tidak lupa saya menyampaikan terimakasih kepada Bapak Imam Eko Wicaksono S.Si., M.Si selaku dosen pengampu mata kuliah strategi algoritma yang sudah membimbing saya dalam mempelajari Strategi Algoritma

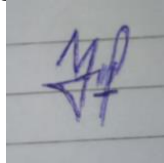
REFERENSI

- [1] H. Sunandar and Pristiwanto, "Optimalisasi Algoritma Greedy dalam Fungsi Penukaran Uang," *Jurnal Teknik Informatika Unika St Thomas*, p. Vol 4 No 2, 2019.
- [2] Mesran, "Implementasi Algoritma Brute Force dalam Pencarian Data," 2014.
- [3] S. Oktaviana and A. Naufal, "Algoritma Greedy untuk Menyusun Ruangan dan Penjadwalan Kuliah," *Jurnal Multinetics*, p. Vol 3 No 1, 2017.
- [4] Paryati, "Optimalisasi Algoritma Greedy dalam Penyelesaian Masalah Knapstack 0-1," in *Seminar Informatika Nasional*, Yogyakarta, 2009.
- [5] A. Y. Husodo, "TECHNICAL REPORT PENGGUNAAN ALGORITMA PENCOCOKAN STRING BOYER-MOORE DALAM MENEMUKAN SITUS TERLARANG," Institut Teknologi Bandung, Bandung, 2009.

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.

Lampung, 29 Maret 2021



Muhammad Ibnu Prayogi 120140152