Penyelesaian Persoalan Penukaran Uang dengan Program Dinamis

Albert Logianto - 13514046
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514046@std.stei.itb.ac.id

***Abstrak*—Persoalan penukaran uang atau koin adalah permasalahan yang umum kita temui di kehidupan sehari-hari, seperti pada *vending machine*. Pokok permasalahan ini adalah, diberikan uang senilai *N*. Tukarlah *N* dengan sejumlah set koin-koin uang yang disediakan. Berapa jumlah minimum koin yang diperlukan untuk penukaran tersebut? Persoalan ini bisa dipandang juga sebagai permasalahan *integer knapsack*. Permasalahan *decision problem* pada penukaran uang ini apakah mungkin melakukan penukaran jika diberikan denominasi sedemikian rupa merupakan permasalahan NP-Complete (NPC). Permasalahan perhitungan dan optimasi pada masalah penukaran uang ini adalah permasalahan NP-Hard. Terdapat beberapa cara atau solusi dalam menyelesaikan masalah ini yaitu menggunakan algoritma *greedy* atau *dynamic programming* (Program Dinamis), walaupun algoritma tersebut belumlah mangkus. Pada makalah ini akan dibahas penyelesaian persoalan ini menggunakan algoritma *dynamic programming*.**

Keywords—persoalan penukaran uang; *change problem*; *dynamic programming*; NPC Problem

I. PENDAHULUAN

Di kehidupan sehari-hari tentulah kita selalu berurusan dengan masalah keuangan, salah satunya adalah bagaimana kasir yang memberikan uang kembalian. Salah satu variasi cara yang biasa dilakukan orang adalah dengan menggunakan "algoritma *greedy*", namun dengan cara ini solusi yang dihasilkan tidaklah selalu optimal (jumlah koin yang dikembalikan tidaklah minimum). Untuk negara yang menggunakan denominasi *1-2-5 series*, algoritma *greedy* akan selalu optimal global, namun untuk mata uang yang tidak menggunakan denominasi tersebut maka algoritma *greedy* tidaklah akan selalu optimal global. Maka dari itu dibutuhkan sebuah algoritma lain yang akan selalu menghasilkan solusi yang optimal global, yaitu algoritma *dynamic programming*.

Dynamic Programming (Program Dinamis) pertama kali dikembangkan oleh seorang ilmuwan benama Richard Bellman, seorang professor di Universitas Princeton pada tahun 1957. Apabila pada algoritma atau metode lain memiliki formulasi standar untuk memecahkan masalah, maka dalam program dinamis ini tidak ada formulasi yang standar seperti pada pemrograman linier, artinya setiap masalah dalam program dinamis memerlukan pola pendekatan atau penyelesaian yang berbeda satu dengan lainnya.

Program Dinamis ini adalah suatu algoritma atau metode dalam menyelesaikan masalah yang kompleks dengan menguraikan cara penyelesaiannya dalam beberapa tahapan dengan membagi-bagi masalah tersebut ke masalah yang lebih sederhana. Sehingga setiap tahapan penyelesaian berkaitan satu sama lain sampai dihasilkan solusi akhir. Ide dari program dinamis ini sangatlah sederhana, jika telah diselesaikan dengan *input* sedemikian rupa, solusi tersebut disimpan untuk referensi di masa depan dimana jika ditemukan masalah yang sama maka tidaklah perlu melakukan komputasi ulang, singkatnya prinsip dari program dinamis adalah "*Remember your Past"*. Jika diberikan persoalan yang dibagi dalam beberapa sub-persoalan dan sub-persoalan tersebut *overlap* dengan persoalan yang pernah diselesaikan sebelumnya maka tidak perlu dilakukan pencarian solusi pada sub-persoalan tersebut. Terdapat dua cara dalam program dinamis, yaitu secara *Top-Down* yang dikenal juga sebagai *memoization* dan *Bottom-Up*.

Program dinamis ini biasa digunakan pada bidang matematika, bioinformatika, *computer* science, dan ekonomi. Tidaklah semua persoalan dapat diselesaikan dengan program dinamis, persoalan-persoalan yang biasa diselesaikan oleh program dinamis adalah persoalan optimasi. Selain digunakan untuk mencari solusi persoalan, program dinamis juga biasa digunakan untuk mencari jumlah solusi optimal yang dapat dihasilkan seperti pada persoalan penukaran uang dan *integer knapsack*. Beberapa contoh persoalan umum yang dapat diselesaikan menggunakan program dinamis adalah pencarian tur terpendek dari sebuah graf, persoalan *Travelling Salesman Person* (TSP), masalah penganggaran modal (Capital Budgeting), *integer knapsack*, *Sequence alignment*, *Fibonacci sequence*, dan berbagai persoalan lainnya.

.

II. DASAR TEORI

A. Program Dinamis

Program Dinamis merupakan suatu metode yang biasannya digunakan untuk membuat suatu keputusan dari serangkaian

keputusan yang saling berkaitan. Tujuan utama dari model ini adalah untuk mempermudah penyelesaian persoalan optimasi yang mempunyai karakteristik tertentu. Inti dari program dinamis ini adalah membagi satu persoalan atas beberapa bagian persoalan yang dalam program dinamis disebut tahap (*stage*), kemudian memecahkan tiap tahap dengan mengoptimalkan keputusan atas tiap tahap sampai seluruh persoalan telah terpecahkan. Istilah "program dinamis" muncul karena perhitungan solusi menggunakan tabel yang ukurannya dapat berubah pada setiap tahapannya. Kata "program" pada program dinamis tidak ada berhubungan dengan kode program (*source code*), tetapi yang berarti perencanaan pada penyelesaian masalah. Prinsip dari program dinamis mirip dengan algoritma *greedy* dimana perbedaannya adalah pada algoritma *greedy* hanya satu rangkaian keputusan yang dihasilkan tanpa memperdulikan keputusan yang dibuat sebelumnya sehingga bersifat optimum lokal, sedangkan pada program dinamis terdapat lebih dari satu rangkaian keputusan yang dipertimbangkan sehingga keputusan yang dibuat pada setiap tahapan akan menghasilkan hasil solusi yang optimum global.

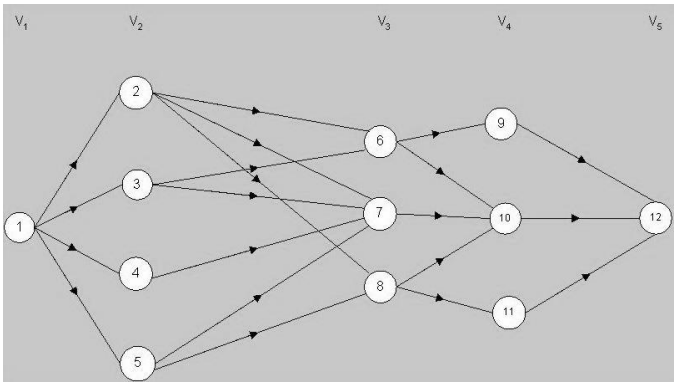
Pada program dinamis, terdapat karakteristik penyelesaiannya sebagai berikut:

1. Terdapat sejumlah pilihan berhingga yang mungkin dipilih.
2. Solusi diselesaikan dengan bertahap; setiap tahapan solusi dibangun dari hasil solusi tahap yang sebelumnya.
3. Untuk membatasi pilihan yang harus dipertimbangkan pada setiap tahap, digunakan persyaratan optimasi dan kendala.

Rangkaian keputusan yang dibuat pada program dinamis sedemikian hingga optimal ditentukan dengan prinsip optimalitas. Pada dasarnya prinsip optimalitas berbunyi sebagai berikut: "jika solusi total optimal, maka bagian solusi sampai tahap ke-*n* juga optimal". Dengan prinsip optimalitas, maka jika kita bekerja dari tahapan *n* ke tahapan *n* + 1, kita dapat menggunakan hasil optimal pada tahapan ke *n* tanpa melakukan komputasi ulang dari tahapan awal. Dengan prinsip optimalitas ini dijamin bahwa pengambilan keputusan pada suatu tahap adalah keputusan yang benar dan optimal untuk tahap-tahapan selanjutnya. Ongkos pada tahapan *n* + 1 adalah ongkos yang dihasilkan dari tahap *n* ditambah dengan ongkos tahap *n* ke tahap *n* + 1.

Karakteristik persoalan yang terdapat pada program dinamis yaitu:

1. Persoalan dapat dibagi menjadi beberapa tahapan (*stage*), dengan pada setiap tahapan hanya diambil satu keputusan.
2. Pada masing-masing tahapan, terdiri dari sejumlah status (*state*) yang berhubungan dengan tahap tersebut. Secara gambaran umum, status merupakan berbagai macam kemungkinan *input* atau masukan yang ada pada tahap tersebut.



Gambar 1. Graf multitahap (*mutistage graph*)
Sumber: http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2014-2015/Program%20Dinamis%20(2015).ppt

Tiap simpul di dalam graf tersebut menyatakan status, sedangkan V1, V2, V3, V4, V5 menyatakan tahap.

3. Hasil dari keputusan yang diambil pada setiap tahap ditransformasikan dari status yang bersangkutan ke status berikutnya pada tahap berikutnya.
4. Ongkos (*cost*) pada suatu tahap meningkat secara teratur (*steadily*) dengan bertambahnya jumlah tahapan.
5. Ongkos pada suatu tahap bergantung pada ongkos tahap-tahap yang sudah berjalan dan ongkos pada tahap tersebut.
6. Keputusan terbaik pada suatu tahap bersifat independen terhadap keputusan yang dilakukan pada tahap sebelumnya.
7. Adanya hubungan rekursif yang mengidentifikasikan keputusan terbaik untuk setiap status pada tahap *n* memberikan keputusan terbaik untuk setiap status pada tahap *n* + 1.
8. Prinsip optimalitas berlaku pada persoalan tersebut.

Dalam menyelesaikan masalah menggunakan program dinamis, terdapat dua pendekatan yaitu maju (*forward* atau *top-down*) dan mundur (*bacward* atau *bottom-up*). Misalkan terdapat x_1, x_2, …, x_n yang menyatakan peubah keputusan (*decision variable*) yang harus dibuat masing-masing untuk tahap 1, 2, …, *n*. Maka pada program dinamis maju, program dinamis bergerak dimulai dari tahapan 1, terus maju ke tahap 2, 3, dan seterusnya hingga tahap ke-*n*. Urutan dari peubah keputusannya adalah x_1, x_2, …, x_n. Sedangkan pada program dinamis mundur, program dinamis bergerak mulai dari tahap ke-*n*, terus mundur ke tahap $n - 1$, $n - 2$, dan seterusnya hingga tahap 1. Urutan peubah keputusannya adalah x_n, x_{n-1}, …, x_1.

Pada program dinamis maju, prinsip optimalitasnya adalah ongkos pada tahap *n* + 1 adalah ongkos yang dihasilkan pada tahap *k* ditambah dengan ongkos dari tahap *k* ke tahap *k* + 1 dengan nilai *k* = 1, 2, …, *n* – 1. Sedangkan pada program dinamis mundur, prinsip optimalitasnya adalah ongkos pada tahap k adalah ongkos yang dihasilkan pada tahap *k* + 1 ditambah dengan ongkos dari tahap *k*+1 ke tahap *k*.

Langkah-langkah pengembangan algoritma program dinamis secara umumnya adalah:

1. Karakteristikkan struktur solusi optimal
2. Definisikan secara rekursif nilai solusi optimal.
3. Hitung nilai solusi optimal secara maju atau mundur.
4. Konstruksi solusi optimal.

Keuntungan dari menggunakan program dinamis untuk menyelesaikan masalah adalah tidak akan terjadi komputasi ulang untuk penyelesaian sub-masalah yang sama, salah satu contohnya adalah deret Fibonacci, jika menggunakan algoritma *naïve* maka implementasinya adalah sebagai berikut

```
function fib(n)
       if n <= 1 return n
       return fib(n - 1) + fib(n - 2)
```

Jika dipanggil fib(5), maka komputasi yang akan dilakukan adalah:

1. fib(5)
2. fib(4) + fib(3)
3. (fib(3) + fib(2)) + (fib(2) + fib(1))
4. ((fib(2) + fib(1)) + (fib(1) + fib(0))) + ((fib(1) + fib(0)) + fib(1))
5. (((fib(1) + fib(0)) + fib(1)) + (fib(1) + fib(0))) + ((fib(1) + fib(0)) + fib(1))

Jika dilihat, terdapat tiga kali komputasi terhadap fib(2), pada persoalan yang lebih besar maka akan banyak sub-masalah yang akan dilakukan rekalkulasi, hal ini tentulah tidak efisien, namun jika digunakan program dinamis dengan memanfaatkan *map object* sederhana yang berisi hasil kalkulasi dari *fib*, maka tidak akan terjadi komputasi ulang terhadap sub-masalah yang sama. Sehingga kompleksitas waktu yang dihasilkan lebih mangkus.

B. Persoalan Penukaran Uang

Persoalan penukaran uang atau koin adalah suatu persoalan yang umum ditemukan di kehidupan sehari-hari kita. Pada dasarnya inti dari masalah ini adalah: "jika kita ingin menukarkan untuk uang atau koin sebesar *N*, dan kita memiliki jumlah tidak terbatas dari denominasi uang atau koin $S = \{S_1, S_2, \ldots, S_n\}$, maka berapa jumlah koin atau uang minimal yang dibutuhkan?". Secara matematis, kita diminta untuk meminimasi $\sum x_k \; untuk \; N = \sum_{k=1 \ldots m} x_k \; S_k \; dimana \; x_k \geq 0, k \in \{1 \ldots m\}$. Masalah ini terdiri dari dua sub-masalah yaitu masalah optimasi, yaitu bagaimana menukar uang dengan jumlah koin seminimal mungkin dan masalah perhitungan, yaitu mencari ada berapa jumlah cara untuk menukarkan uang. Persoalan ini merupakan kasus umum dari *Integer Partition* yang bisa diselesaikan dengan berbagai macam algoritma. Aplikasi lain yang hampir sama dengan persoalan ini adalah komputasi cara untuk menghasilkan *nine dart finish* pada permainan *dart.*

Persoalan penukaran uang ini mirip dengan *integer knapsack*, dimana pada *integer knapsack* terdapat kapasitas maksimum yang tidak harus dicapai, sedangkan pada persoalan ini terdapat jumlah uang yang harus ditukar dan harus tepat dipenuhi. Tahap pada *integer knapsack* menggunakan program dinamis adalah proses memasukkan barang ke dalam *knapsack*, sedangkan pada persoalan ini tahapnya adalah proses memilih koin untuk dipakai dalam penukaran uang. Status pada *integer knapsack* menyatakan kapasitas muat karung yang tersisa setelah memasukkan barang pada tahap sebelumnya, sedangkan status pada persoalan ini adalah jumlah uang tersisa yang setelah dikurangi dengan koin yang digunakan pada tahap sebelumnya.

Persoalan penukaran uang ini termasuk dalam kelompok kelas *NP*-Complete (*Non-Deterministic Polynomial Complete*) atau biasa disebut NPC. Hal ini dikarenakan masalah penukaran uang termasuk ke dalam kelompok kelas NP dan setiap persoalan di dalam kelas NP dapat direduksi menjadi persoalan penukaran uang dalam waktu polinom. Persoalan ini masih termasuk sebagai persoalan yang sukar karena belum ada persoalan dalam NPC yang dapat diselesaikan dalam waktu polinomial. Tidak semua algoritma yang diajarkan pada mata kuliah Strategi Algoritma dapat digunakan untuk menyelesaikan permasalahan ini, beberapa algoritma yang dapat menyelesaikan masalah ini yaitu algoritma *greedy*, rekursif, dan program dinamis. Pada algoritma *greedy*, maka kita akan selalu mengambil denominasi koin terbesar yang mungkin untuk diambil pada setiap langkahnya, lalu mengurangi jumlah uang yang ingin ditukar dengan nilai koin yang diambil, langkah-langkah ini dilakukan sampai jumlah uangnya sama dengan 0. Dengan menggunakan cara rekursif, terdapat beberapa pengulangan kalkulasi pada beberapa sub-masalah, hal ini dapat dihindari menggunakan program dinamis.

Sebagai contoh, kita diminta untuk menukar uang sejumlah sebesar 8 dan kita memiliki koin: 5, 4, 3, 1 maka solusi optimalnya adalah 8 = 5 + 3 (2 koin) yang juga merupakan hasil yang didapat menggunakan algoritma *greedy*. Namun tidak semua solusi yang dihasilkan oleh algoritma *greedy* selalu optimal. misalkan kita diminta untuk menukarkan uang sejumlah 7, maka algoritma *greedy* tidak menghasilkan solusi yang optimal yaitu 7 = 5 + 1 + 1 (3 koin) dimana solusi optimalnya adalah 7 = 4 + 3 (2 koin). Pada beberapa jenis denominasi seperti 1-2-5 *series*, maka algoritma *greedy* akan selalu menghasilkan solusi optimal, namun pada jenis denominasi lain, algoritma *greedy* tidak akan selalu menghasilkan solusi yang optimal. Maka dari itu, agar selalu didapatkan solusi optimal, digunakan metode rekursif atau program dinamis. Dimana kompleksitas waktu program dinamis lebih baik dibandingkan dengan metode rekursif karena tidak terdapat rekalkulasi atau komputasi ulang untuk sub-masalah yang telah diselesaikan.

III. IMPLEMENTASI PROGRAM DINAMIS DALAM MENYELESAIKAN PERSOALAN PENUKARAN UANG

Pada persoalan penukaran uang ini, kita hanya tertarik pada nilai dari solusi dan ongkos (*cost*) yang dibutuhkan. Pencarian solusi dilakukan dengan optimasi terhadap suatu nilai tertentu. Untuk mencari solusi, kita dapat membagi persoalan tersebut menjadi beberapa tahap. Oleh karena itu, persoalan penukaran uang dapat diselesaikan dengan menggunakan program dinamis. Hal ini disebabkan oleh persoalan penukaran uang memiliki karakteristik yang sama dengan karakteristik masalah yang dapat diselesaikan pada program dinamis.

Pada persoalan ini,

1. Tahap (k) adalah proses memilih koin yang digunakan untuk menukar uang.
2. Status (y) menyatakan jumlah uang tersisa setelah dikurangi dengan koin yang dipakai atau dipilih pada tahapan sebelumnya.

Langkah awal yang diperlukan dalam menyelesaikan persoalan ini menggunakan program dinamis adalah dengan menentukan karakter struktur dari solusi optimal. Misalkan S adalah solusi optimal untuk menukar uang sejumlah n, maka $S' = S - c$, untuk $c \in S$ sehingga S' merupakan solusi optimal untuk sub-masalah dalam menukarkan uang sejumlah $n - c$.

Langkah kedua yang dilakukan adalah menentukan basis dan relasi rekurens dari permasalahan ini, yaitu

$f_0(y) = 0, y = 0, 1, 2, ..., M$	(basis)
$f_k(y)$ tidak ada untuk $y < 0$	(basis)
$f_k(y) = \min\{f_{k-1}(y), x + C_k(y)$	(rekurens)
$C_k(y)$ *tidak ada* jika $y - p_k < 0$	
$C_k(y) = 1 + f_{k-1}(y - p_k)$, jika $p_k \le y < 2 * p_k$	
$C_k(y) = 1 + C_k(y - p_k)$, jika $y \ge 2 * p_k$	

- $f_k(y)$ adalah jumlah koin minimum dari persoalan penukaran uang pada tahap k untuk jumlah uang sebesar y.
- Nilai k merupakan tahapan dalam pengerjaan solusi.
- $f_0(y) = 0$ adalah nilai dari persoalan penukaran uang dengan tidak ada koin yang dipilih dan jumlah uang sebesar y.
- $f_k(y)$ tidak ada jika persoalan penukaran uang dengan jumlah uang negatif.
- Solusi optimum dari persoalan penukaran uang adalah $f_n(M)$.

Misalkan kita diminta menukarkan uang (M) sebesar 11 dan kita memiliki denominasi koin yaitu $S = \{1, 4, 5, 7, 8\}$ sehingga $n = 5$. Dibutuhkan lima tahapan untuk menyelesaikan masalah ini. Pada penyelesaian ini, digunakan program dinamis maju (*top-down*).

- ***TAHAP 1***:

  $f_1(y) = \min\{f_0(y), C_1(y)\}$

y	$f_0(y)$	$C_1(y)$	*Solusi Optimum*	
			$f_1(y)$	$(x_1, x_2, x_3, x_4, x_5)$
0	0	-	0	{0, 0, 0, 0, 0}
1	0	1	1	{1, 0, 0, 0, 0}
2	0	2	2	{2, 0, 0, 0, 0}
3	0	3	3	{3, 0, 0, 0, 0}
4	0	4	4	{4, 0, 0, 0, 0}
5	0	5	5	{5, 0, 0, 0, 0}
6	0	6	6	{6, 0, 0, 0, 0}
7	0	7	7	{7, 0, 0, 0, 0}
8	0	8	8	{8, 0, 0, 0, 0}
9	0	9	9	{9, 0, 0, 0, 0}
10	0	10	10	{10, 0, 0, 0, 0}
11	0	11	11	{11, 0, 0, 0, 0}

- ***TAHAP 2***:

  $f_2(y) = \min\{f_1(y), C_2(y)\}$

y	$f_1(y)$	$C_2(y)$	*Solusi Optimum*	
			$f_2(y)$	$(x_1, x_2, x_3, x_4, x_5)$
0	0	-	0	{0, 0, 0, 0, 0}
1	1	-	1	{1, 0, 0, 0, 0}
2	2	-	2	{2, 0, 0, 0, 0}
3	3	-	3	{3, 0, 0, 0, 0}
4	4	1	1	{0, 1, 0, 0, 0}
5	5	2	2	{1, 1, 0, 0, 0}
6	6	3	3	{2, 1, 0, 0, 0}
7	7	4	4	{3, 1, 0, 0, 0}
8	8	2	2	{0, 2, 0, 0, 0}
9	9	3	3	{1, 2, 0, 0, 0}
10	10	4	4	{2, 2, 0, 0, 0}
11	11	5	5	{3, 2, 0, 0, 0}

- ***TAHAP 3***:

  $f_3(y) = \min\{f_2(y), C_3(y)\}$

y	$f_2(y)$	$C_3(y)$	*Solusi Optimum*	
			$f_3(y)$	$(x_1, x_2, x_3, x_4, x_5)$
0	0	-	0	{0, 0, 0, 0, 0}
1	1	-	1	{1, 0, 0, 0, 0}
2	2	-	2	{2, 0, 0, 0, 0}
3	3	-	3	{3, 0, 0, 0, 0}
4	1	-	1	{0, 1, 0, 0, 0}
5	2	1	2	{0, 0, 1, 0, 0}
6	3	2	2	{1, 0, 1, 0, 0}
7	4	3	3	{2, 0, 1, 0, 0}
8	2	4	2	{0, 2, 0, 0, 0}
9	3	2	2	{0, 1, 1, 0, 0}
10	4	2	2	{0, 0, 2, 0, 0}
11	5	3	3	{1, 0, 2, 0, 0}

- ***TAHAP 4***:

  $f_4(y) = \min\{f_3(y), C_4(y)\}$

y	$f_3(y)$	$C_4(y)$	*Solusi Optimum*	
			$f_4(y)$	$(x_1, x_2, x_3, x_4, x_5)$
0	0	-	0	{0, 0, 0, 0, 0}
1	1	-	1	{1, 0, 0, 0, 0}
2	2	-	2	{2, 0, 0, 0, 0}
3	3	-	3	{3, 0, 0, 0, 0}
4	1	-	1	{0, 1, 0, 0, 0}
5	2	-	2	{0, 0, 1, 0, 0}
6	2	-	3	{1, 0, 1, 0, 0}
7	3	1	1	{0, 0, 0, 1, 0}
8	2	2	2	{1, 0, 0, 1, 0}
9	2	3	2	{0, 1, 1, 0, 0}
10	2	4	2	{0, 0, 2, 0, 0}
11	3	2	2	{0, 1, 0, 1, 0}

- ***TAHAP 5***:

  $f_5(y) = \min\{f_4(y), C_5(y)\}$

y	$f_4(y)$	$C_5(y)$	*Solusi Optimum*	
			$f_5(y)$	$(x_1, x_2, x_3, x_4, x_5)$
0	0	-	0	{0, 0, 0, 0, 0}
1	1	-	1	{1, 0, 0, 0, 0}
2	2	-	2	{2, 0, 0, 0, 0}
3	3	-	3	{3, 0, 0, 0, 0}
4	1	-	1	{0, 1, 0, 0, 0}
5	2	-	2	{0, 0, 1, 0, 0}
6	3	-	3	{1, 0, 1, 0, 0}
7	1	-	1	{0, 0, 0, 1, 0}
8	2	1	1	{0, 0, 0, 0, 1}
9	2	2	2	{1, 0, 0, 0, 1}
10	2	3	2	{0, 0, 2, 0, 0}
11	2	4	2	{0, 1, 0, 1, 0}

Didapatkan himpunan solusi optimal dari persoalan ini adalah {0, 1, 0, 1, 0}. Pada implementasi program dinamis dalam menyelesaikan persoalan penukaran uang ini digunakan *array* satu dimensi berukuran *n* dengan n adalah jumlah denominasi koin. Sehingga kompleksitas ruangnya adalah O(n). Berikut adalah pseudocode dari program dinamisnya.

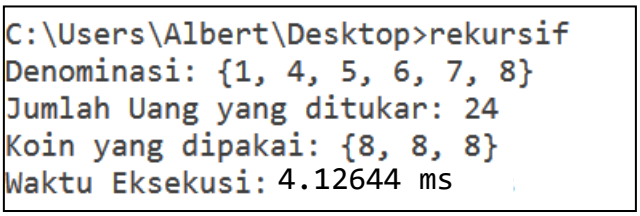
```
func count( amount: integer, d: array of integer,
size: integer, C: array of integer, s: array of
integer ) → array of integer
//d adalah denominasi koin
//C adalah jumlah koin yang dibutuhkan
//s adalah index denominasi koin terbesar yang
digunakan
  C[0] = 0;
    for j ← 1 to amount do
        //inisialisasi tabel temp
        C[j] ← 999999;
        for i ← 0 to size-1 do
            if(j >= d[i] && 1 + C[j-d[i]] < C[j] )
                then C[j] ← 1 + C[j-d[i]]
                // denominasi ke i yang digunakan
                untuk jumlah uang (amount) j
                s[j] ← i;

return C[amount]
```

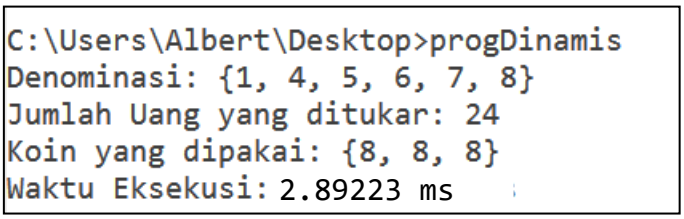
Kompleksitas waktunya adalah: O(Mn) dengan *M* adalah jumlah uang yang ditukar dan *n* adalah jenis denominasi koin yang terdapat.

IV. Analisis

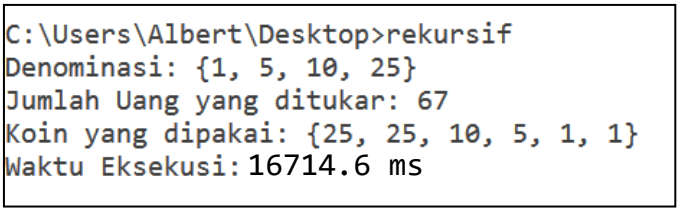
Pada program dinamis, kompleksitas waktu yang dibutuhkan lebih efisien jika dibandingkan dengan metode rekursif. Waktu eksekusi dari program dinamis terbukti lebih cepat setelah dilakukan pengujian yang bisa dilihat di bawah.



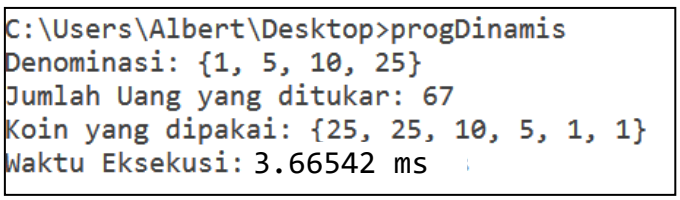
Gambar 2. Output program rekursif 1



Gambar 3. Output program dinamis 1



Gambar 3. Output program rekursif 2



Gambar 2. Output program dinamis 2

Terlihat bahwa untuk persoalan yang masih sederhana dengan jumlah uang yang ditukar masih sedikit, kecepatan eksekusi dari kedua algoritma tidak berbeda jauh dengan keunggulan masih dipegang oleh program dinamis kurang lebih sebesar 1,5 kali lebih cepat. Namun untuk persoalan dengan jumlah uang yang semakin besar seperti pada gambar 4 dan 5, terlihat metode rekursif sangatlah tidak efisien, terlihat dengan metode rekursif dibutuhkan waktu sekitar 16 detik sedangkan dengan program dinamis hanya dibutuhkan waktu 3.7 ms. Dengan menggunakan program dinamis, kecepatan eksekusinya 4500 kali lebih cepat dibandingkan dengan menggunakan metode rekursif biasa. Ini dikarenakan pada metode rekursif, terdapat sub-masalah yang *overlap* satu sama lain, sehingga terjadi rekalkulasi atau komputasi ulang. Banyaknya sub-masalah yang *overlap* ini akan berkembang secara eksponensial seiringnya meningkatnya jumlah uang yang ditukar. Hal ini tidak terjadi pada program dinamis, karena sub-masalah yang telah diselesaikan akan disimpan nilainya pada suatu tabel untuk digunakan untuk memecahkan sub-masalah pada tahap selanjutnya, sehingga tidak akan terjadi sub-masalah yang *overlap*. Hal ini membuat waktu program dinamis menjadi jauh lebih cepat dibandingkan metode rekursif biasa.

V. Kesimpulan

Persoalan penukaran uang ini adalah persoalan yang umum dijumpai pada kehidupan sehari-hari. Terdapat beberapa algoritma yang dapat menyelesaikan persoalan ini, antara lain berupa algoritma *greedy*, rekursif, dan program dinamis. Pada penyelesaian menggunakan algoritma *greedy*, solusi yang dihasilkan tidaklah akan selalu optimal. Algoritma *greedy* akan selalu optimal pada beberapa macam denominasi saja. Dengan metode rekursif biasa, akan selalu dicapai hasil solusi yang optimal namun waktu yang dibutuhkan untuk mendapatkannya masihlah kurang efisien karena masih terdapat rekalkulasi atau komputasi ulang sub-masalah yang *overlap*, dengan menggunakan program dinamis, pengulangan komputasi dapat dihindari dengan menggunakan tabel sementara yang menampung hasil atau solusi yang diselesaikan pada sub-masalah dalam tahap tertentu. Sehingga pada tahap selanjutnya, solusi pada tahap sebelumnya dapat dipakai dengan melihat pada tabel yang telah dibuat. Sehingga demikian, program dinamis akan berjalan dengan waktu eksekusi yang lebih cepat. Walaupun masih belum bisa diselesaikan dalam waktu polinomial, program dinamis terbukti lebih cepat dibandingkan algoritma lain dalam menyelesaikan persoalan penukaran uang.

VI. Ucapan Terima Kasih

Penulis ingin mengucapkan puji syukur kepada Tuhan Yang Maha Esa karena oleh rahmat-Nya penulis dapat menyelesaikan makalah ini dengan baik dan tepat waktu. Penulis juga menyampaikan banyak terima kasih kepada Bapak Rinaldi Munir dan Ibu Nur Ulfa Maulidevi yang telah membimbing penulis selama satu semester ini dan membantu dalam menyelesaikan makalah strategi algoritma ini.

Referensi

[1] Munir, Rinaldi. 2009. ”Diktat Kuliah IF 2251 Strategi Algoritmik”. Bandung:Program Studi Teknik Informatika STEI ITB.

[2] http://www.columbia.edu/~cs2035/courses/csor4231.F07/dynamic.pdf, diakses pada 6 Mei 2016.

[3] http://www.geeksforgeeks.org/dynamic-programming-set-7-coin-change/, diakses pada 6 Mei 2016.

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, 6 Mei 2016

Albert Logianto
13514046