Implementasi Program Dinamis dalam Optimasi Kapasitas Kargo Logistik

Renjira Naufhal Dhiaegana / 13516014
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung, Indonesia
13516014@std.stei.itb.ac.id

***Abstrak*—Dewasa ini, sudah banyak aliran barang yang terjadi, mulai dari belanja online, pemesanan makanan dengan ojek online, pengiriman surat, ataupun logsitik ke kota, provinsi, bahkan sampai ke luar negeri. Hal ini sudah tidak menjadi asing lagi bahwa kita sudah masuk ke jaman maju dimana semua urusan antar mengantar sudah menjadi sangat sederhana bagi kita. Untuk mengantar suatu barang ke orang lain, kita hanya perlu membuka gawai kita untuk mengakses aplikasi pengirim barang dan mengisikan detil-detil yang diperlukan. Atau kita bisa juga pergi ke cabang-cabang perusahaan logsitik terdekat untuk mengajukan permintaan pengantaran. Tetapi belum tentu mudah untuk perusahaan logistik tersebut karena secara kasar, setiap hari terdapat jutaan bahkan miliaran permintaan pengantaran barang. Tanpa penggunaan otomatisasi, dapat dikatakan manajemen pengantaran barang tidak dapat dilakukan oleh manusia biasa. Diperlukan sebuah program yang bisa menyelesaikan permasahan tersebut karena sebuah program sudah menjadi bagian dari semua aspek kehidupan modern.**

Kata Kunci—dynamic programming; greedy; optimal; memoization

I. PENDAHULUAN

Logistik adalah ilmu yang berkaitan dengan aliran barang atau jasa dari pemasok barang ke tujuan-tujuannya, yaitu penerima atau pembeli barang. Perusahaan logistik sudah banyak di indonesia, contohnya seperti JNE, TIKI, DHL, UPS, Fed-Ex, dan masih banyak lagi perusahaan yang membidangi logistik. Jasa logistik merupakan hal yang penting dalam dunia bisnis maupun logistik bantuan dalam kejadian bencana.

Perusahaan logistik mendapatkan keuntungannya dari perusahaan skala menenengah, rataan harga dapat dihitung dari jarak tempuh pengiriman barang dan berat total barang yang dikirim. Harga pengiriman barang bervariasi, semakin berat barang atau semakin jauh pengiriman maka semakin mahal juga harga pengiriman barang ke pemesan. Logistik merupakan pengoptimalan penggunaan modal pengantaran dalam mendapatkan barang yang tepat dalam kurun waktu yang singkat.

GAMBAR I. ILUSTRASI LOGISTIK

Sumber : https://suaracargo.com/paket-id-siap-jadi-marketplace-bagi-perusahaan-jasa-logistik/

Pengantaran barang logistik membutuhkan kendaraan yang tepat untuk setiap pengirimannya. Pengiriman barang yang banyak dengan menggunakan kendaraan yang kecil memungkinkan pengantaran yang tidak mangkus. Begitu pula pengiriman barang yang yang sedikit dengan menggunakan kendaraan yang memiliki kapasitas besar. Oleh karena itu dibutuhkan optimasi dalam pemerataan barang yang dibawa oleh pengangkut logistik.

Algoritma Program Dinamis(*dynamic programming*) adalah algoritma yang mangkus dalam masalah optimasi. Algoritma ini dapat diaplikasikan dalam pengoptimasian pengangkut logistik karena mirip dengan persoalan *dynamic programming*, yaitu *0-1 knapsack problem*.

Pada makalah ini, penulis akan memberikan contoh program menggunakan bahasa pemrograman python.

II. DASAR TEORI

A. Program Dinamis

Program Dinamis adalah sebuah algoritma penguraian solusi menjadi sekumpulan langkah-langkah sehingga solusi yang didapatkan dari sebuah persoalan dapat dilihat sebagai

serangkaian keputusan yang diambil dan berkaitan satu sama lain.

Program Dinamis pertamakali ditemukan oleh seorang matematikawan Amerika Serikat bernama Richard Bellman pada tahun 1957. Selain itu dia juga menyatakan *Bellman's Principle of Optimality* yang berbunyi

Algoritma program dinamis memiliki karakteristik persoalan sebagai berikut:

1. Persoalan memiliki sekumpulan pilihan berhingga yang mungkin,
2. Solusi yang diambil pada sebuah tahapan dipilih dengan memperhatikan solusi sebelumnya(berkaitan dengan solusi sebelumnya),
3. Persyaratan optimasi dan kendala digunakan untuk membatasi banyaknya pilihan dalam setiap tahapan.

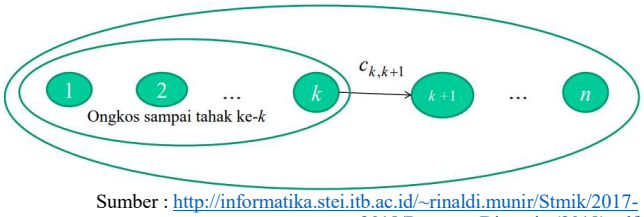
Prinsip optimalitas adalah prinsip yang menyatakan apabila solusi total optimal, maka dari bagian solusi sampai ke tahap-*n* juga optimal. Contohnya apabila pada tahap-*n* ke tahap-*n+1*, hasil yang optimal dari tahap-k dapat digunakan kembali tanpa harus kembali ke tahap awal.

Dengan prinsip optimalitas, dapat dijamin bahwa pengambilan keputusan pada suatu tahap adalah keputusan yang benar untuk tahapan-tahapan selanjutnya.

Jika pada setiap tahapan, kita menghitung ongkos (cost), maka dapat dirumuskan sebagai berikut:

$$cost_{k+1} = cost_k + cost_{k \to k+1} \quad (1)$$

GAMBAR II. ONGKOS PROGRAM DINAMIS



Sumber : http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Program-Dinamis-(2018).pdf

Sebenarnya Program dinamis adalah algoritma greedy yang dilakukan berulang ulang sampai menemukan solusi yang paling optimal. Karena pada algoritma *greedy*, hanya diambil satu rangkaian keputusan yang pernah dihasilkan saja.

Secara umum, terdapat empat langkah yang dilakukan dalam mengembangkan algoritma program dinamis, yaitu:

1. Karakteristikkan struktur solusi optimal,
2. Definisikan secara rekursif nilai solusi optimal,
3. Hitung nilai dari solusi optimal secara maju atau mundur (akan dijelaskan selanjutnya),
4. Konstruksi solusi optimal.

Terdapat dua jenis pendekatan pada Program Dinamis yaitu pendekatan maju (*forward* atau *up-down*) dan mundur (*backward* atau *bottom-up*).

1) Pendekatan Maju

Pada pendekatan maju, sama saja halnya seperti formulasi rekursif dari masalah apapun. Jika solusi dari sebuah masalah dapat diformulasikan secara rekursif dengan menggunakan solusi untuk upa-masalah, dan jika upa-permasalahannya berhimpit satu sama lain, maka sesorang dapat melakukan *memoization* atau dengan kata lain menyimpan solusi untuk sub-masalah dalam sebuah tabel. Setiap kali kita mencoba memecahkan sub-masalah baru, pertama-tama kita periksa tabel untuk melihat apakah itu sudah dipecahkan. Jika suatu solusi sudah dicatat, kita dapat menggunakannya secara langsung, jika tidak kita memecahkan sub-masalah dan menambahkan solusinya ke tabel.

Program dinamis mulai dari tahap 1, lalu maju ke tahap 2, 3, dan tahap seterusnya sampai ke tahap *n* (tahap akhir). Hasil keputusan yang diambil merupakan runutan peubah $x_1, x_2, x_{3,\ldots}, x_n$.

2) Pendekatan Mundur

Sama seperti pendekatan maju, pertama-tama kita harus memformulasikan pemecahan solusi untuk masalah dengan cara rekursif. Tetapi, sekarang kita bergerak dari tahapan yang paling akhir ke tahapan yang paling awal. Biasanya pada pendekatan mundur, tahapan yang diketahui nilainya adalah tahapan yang paling akhir. Baru dari tahapan tersebut, dilakukan pencarian solusi yang paling optimal.

Program dinamis mulai dari tahap *n*, lalu mundur ke tahap n-1, n-2, dan seterusnya sampai ke tahap 1 (tahap awal). Hasil keputusan yang diambil merupakan runutan peubah $x_n, x_{n-1}, x_{n-2,\ldots}, x_1$.

B. Contoh permasalahan yang akan dibahas dan cara penyelesaian permasalahan dengan menggunakan program dinamis (0/1 knapsack problem).

1) Permasalahan *0/1 knapsack*

Permasalahan 0/1 knapsack adalah salah satu dari sekian banyak permasalahan optimasi. Persoalan ini sebetulnya sangat sederhana, yaitu bagaimana cara sesorang dapat mengambil sejumlah barang yang memiliki nilai tertentu dan memiliki

berat masing-masing, tetapi tidak boleh melebihi kapasitas yang dapat dibawa pada kantong (knapsack) dan mendapatkan nilai total barang yang paling besar. Pada permasalahan 0/1 knapsack, semua benda diasumsikan memiliki jumlah 1, sehingga perhitungan yang dapat dilakukan untuk menentukan sebuah barang masuk atau tidak masuk ke dalam ransel hanya memilih 0 (tidak dibawa) atau 1 (dibawa).

Penyelesaian dengan Program Dinamis Maju

Pada persoalan ini,

1. Tahap adalah proses memasukkan barang kedalam kantung.
2. Status menyatakan kapasitas kantong yang tersisa setelah pemasukan barang pada tahapan sebelumnya.

Dari tahap pertama, kita masukkan objek ke-1 ke dalam kantung untuk setiap satuan kapasitas karung sampai batas maksimum yang bisa dibawa sudah tercapai. Karena kapasitas dari kantung adalah bilangan bulat, pendekatan ini praktis. Misalkan ketika memasukan objek pada tahap ke k, kapasitas muat kantung sekarang adalah $y - w_k$, karena y adalah kapasitas kantung setelah pemasukan barang pada tahap *k-1*.

Untuk mengisi kapasitas sisanya, kita menerapkan prinsip optimalitas dan mengacu pada nilai optimal dari tahapan sebelumnya untuk kapasitas sisa $y - w_k$, yaitu $f_{k-1}(y - w_k)$.

Selanjutnya, kita bandingkan nilai keuntungan dari objek pada tahap k (p_k) ditambah nilai $f_{k-1}(y - w_k)$ dengan keuntungan pengisian hanya *k-1* macam objek, $f_{k-1}(y)$.

Jika $p_k + f_{k-1}(y - w_k)$ lebih kecil dari $f_{k-1}(y)$,maka objek yang ke-*k* tidak dimasukkan ke dalam karung, tetapi lebih besar, maka objek ke-*k* dimasukkan.

Relasi rekurens untuk persoalan ini adalah:

$f_0(y) = 0, y = 0, 1, 2, \ldots, M$ (basis)(2)

$f_k(y) = -\infty , y < 0$ (basis)(3)

$f_k(y) = max\{f_{k-1}(y), p_k + f_k\text{-}1(y - w_k)\}$, (rekurens)(4)

$k = 1, 2, \ldots, n$

$f_k(y)$ adalah keuntungan optimum dari persoalan *0/1 knapsack* pada tahap *k* untuk kapasitas kantung sebesar *y*.

$f_0(y) = 0$ adalah nilai dari persoalan *knapsack* kosong (tidak ada barang untuk dimasukkan kedalam *knapsack*). Dengan kapasitas.

$f_k(y) = -\infty$ adalah nilai dari persoalan *knapsack* untuk kapasitas negatif.

Solusi optimum dari persoalan *0/1 knapsack* adalah $f_n(M)$.

2) Contoh persoalan 0/1 knapsack

Misalkan terdapat 3 barang yang masing masing memiliki berat 2, 3, dan 1 satuan berat, dan nilai 65, 80, dan 30 satuan nilai. Lebih jelasnya dapat dilihat pada tabel dibawah,

TABEL I. CONTOH PERSOALAN *0/1 KNAPSACK*

Barang ke-*i*	w_i	p_i
1	2	65
2	3	80
3	1	30

Karena terdapat tiga barang, maka terdapat juga 3 tahapan:

Tahap 1:

$$f_1(y) = \max\{f_0(y), p_1 + f_0(y - w_1)\}$$
$$= \max\{f_0(y), 65 + f_0(y - 2)\}$$

TABEL II. TAHAP 1

y			Solusi Optimum	
	$f_0(y)$	$65 + f_0(y-2)$	$f_1(y)$	(x_1*,x_2*,x_3*)
0	**0**	-∞	0	(0,0,0)
1	**0**	-∞	0	(0,0,0)
2	0	**65**	65	(1,0,0)
3	0	**65**	65	(1,0,0)
4	0	**65**	65	(1,0,0)
5	0	**65**	65	(1,0,0)

Tahap 2:

$$f_2(y) = \max\{f_1(y), p_2 + f_1(y - w_2)\}$$
$$= \max\{f_1(y), 80 + f_1(y - 3)\}$$

TABEL III. TAHAP 2

y			Solusi Optimum	
	$f_1(y)$	$80 + f_1(y-3)$	$f_2(y)$	(x_1*,x_2*,x_3*)
0	**0**	80+(-∞) = -∞	0	(0,0,0)
1	**0**	80+(-∞) = -∞	0	(0,0,0)
2	**65**	80+(-∞) = -∞	65	(1,0,0)
3	65	80+0 = **80**	80	(0,1,0)
4	65	80+0 = **80**	80	(0,1,0)
5	65	80+65 = **145**	145	(1,1,0)

Tahap 3:

$$f_3 y = \max\{f_2(y), p_3 + f_2(y - w_3)\}$$
$$= \max\{f_2(y), 30 + f_2(y - 1)\}$$

TABEL IV. TAHAP 3

y			Solusi Optimum	
	$f_2(y)$	$30 + f_2(y-1)$	$f_3(y)$	(x_1*,x_2*,x_3*)
0	**0**	30+(-∞) = -∞	0	(0,0,0)
1	**0**	30+0 = **30**	30	(0,0,0)
2	**65**	30+0 = 30	65	(1,0,0)
3	80	30+0 = **95**	95	(1,0,1)
4	80	30+0 = **110**	110	(0,1,1)
5	**145**	30+65 = 110	**145**	(1,1,0)

Dengan demikian, didapatkan solusi optimum X = (1,1,0) dengan $\sum p = f$ = 145.

III. IMPLEMENTASI PROGRAM DINAMIS

Permasalahan yang akan dibahas oleh penulis adalah kapasitas muat kargo logistik. Apabila pada sebuah waktu, terdapat beberapa barang yang sudah diatur tujuan pengantarannya ke satu tempat tujuan. Lalu, Akan dipilih dari sekian banyak barang, barang mana saja yang akan dibawa oleh kendaraan pengangkut barang logistik ke sebuah perusahaan yang membutuhkan barang tersebut. Sayangnya, tidak semua barang dapat dibawa secara bersamaan sehingga diperlukan optimasi pemilihan barang yang akan dibawa. Perusahaan memiliki sebuah daftar kebutuhan mana yang lebih penting dan memiliki nilai yang lebih dan menyerahkan masalah pengantaran kepada perusahaan logistik.

Tentu apabila tidak menggunakan bantuan komputer akan membutuhkan sumber daya manusia yang lebih sehingga butuh penggajian yang akan menambahkan cost perusahaan logistik. Hal ini dapat dihindari dengan otomatisasi pekerjaan optimalisasi barang yang akan dibawa kargo menggunakan program yang mengimplementasikan Program Dinamis, yang mana akan menampilkan solusi yang paling optimum dari semua kemungkinan solusi yang ada.

Dengan menggunakan program, maka juga akan terbantu dalam pemilihan barang kargo yang berjumlah banyak. Program ini tidak harus digunakan satu kali saja tapi bisa berulang-ulang dan memberikan solusi yang paling optimum.

Sebuah contoh permasalahan yang sederhana adalah sebagai berikut, apabila sebuah perusahaan logistik hanya memiliki satu sisa kendaraan pengangkut berupa truk yang memiliki kapasitas maksimum sebesar 6 ton. Terdapat 4 jenis bahan mentah yang masing-masing memiliki berat terurut 3 ton, 2 ton, 2 ton, dan 1 ton. Masing-masing bahan juga memiliki nilai kegunaan yang sudah diberikan detilnya dari perusahaan, yaitu 35 satuan, 80 satuan, 60 satuan, dan 95 satuan.

Optimalisasi dapat dilakukan secara sederhana dengan kode berikut:

```
def dpknapsack(K, list_barang, val, n):
  S = [[0 for x in range(K+1)] for x in
range(n+1)]
  #loop tahapan
  for i in range(n+1):
    #loop tiap kapasitas
    for w in range(K+1):
      if i==0 or w==0:
        S[i][w] = 0
      elif list_barang[i-1] <= w:
        S[i][w] = max(val[i-1] + S[i-
1][w-list_barang[i-1]],  S[i-1][w])
      else:
        S[i][w] = S[i-1][w]
  print("Barang yang
dibawa:",barangDibawa(list_barang ,S),
"\nNilai total:")
  return S[n][K]

def barangDibawa(list_barang,c):
  i = len(c)-1
  currentK =  len(c[0])-1
  dibawa = [0]*len(list_barang)
  while (i >= 1 and currentK >= 1):
    if (i==0 and c[i][currentK]>0) or
(c[i][currentK] != c[i-1][currentK]):
      dibawa[i-1] = 1
      currentK = currentK -
list_barang[i-1]
    i = i-1
  return dibawa
```

Pada kode ini, akan dicari nilai maksimum barang yang dapat dibawa dan barang apa saja yang harus dibawa untuk mendapatkan nilai maksimum tersebut. Fungsi dpknapsack akan mengembalikan nilai maksimum yang dapat dibawa oleh kargo. Sedangkan fungsi barangDibawa akan mengembalikan sebuah list yang berisi 0 atau satu pada masing-masing indeksnya dimana 0 menyatakan barang ke-*i* tidak dibawa dan 1 menyatakan barang ke-*i* dibawa.

Deskripsi Permasalahan:

Barang ke-i	w_i	p_i
1	35	3
2	80	2
3	60	2
4	95	1

TABEL V. PERSOALAN LOGISTIK

Apabila permasalahan sebelumnya ingin dimasukkan kedalam program, dapat diadaptasi dengan kode python sebagai berikut:

```
val = [35,80,60,95]
list_barang = [3,2,2,1]
K = 6
jumlah = len(val)
```

Dengan keterangan sebagai berikut:

- val adalah list yang berisi nilai masing masing barang yang bisa dibawa
- list_barang adalah list yang berisi berat masing - masing barang
- K adalah kapasitas maksimum kendaraan

Program bekerja persis dengan cara tabel, akan dicatat $f_k(y)$ dari masing-tahap 1 sampai n.

Berikut adalah hasil program yang sudah dijalankan:

```
Barang yang dibawa: [0, 1, 1, 1]
Nilai total:
235
[Finished in 0.4s]
```

Dapat dilihat bahwa program dapat dijalankan dengan benar.

IV. ANALISIS

Pada bagian ini, penulis ingin menguji kebenaran program dengan menyelesaikan permasalahan menggunakan Program Dinamis tanpa menjalankan program, atau bisa dikatakan manual tanpa bantuan komputer.

Karena terdapat empat barang, maka terdapat juga empat tahapan:

Tahap 1:

$$f_1(y) = \max\{f_0(y), p_1 + f_0(y - w_1)\}$$
$$= \max\{f_0(y), 35 + f_0(y - 3)\}$$

TABEL VI. TAHAP 1

y			Solusi Optimum	
	$f_0(y)$	$65 + f_0(y-3)$	$f_1(y)$	(x_1*,x_2*,x_3*, x_4*)
0	**0**	-∞	0	(0,0,0,0)
1	**0**	-∞	0	(0,0,0,0)
2	**0**	-∞	0	(0,0,0,0)
3	0	**35**	35	(1,0,0,0)
4	0	**35**	35	(1,0,0,0)
5	0	**35**	35	(1,0,0,0)
6	0	**35**	35	(1,0,0,0)

Tahap 2:

$$f_2(y) = \max\{f_1(y), p_2 + f_1(y - w_2)\}$$
$$= \max\{f_1(y), 80 + f_1(y - 2)\}$$

TABEL VII. TAHAP 2

y			Solusi Optimum	
	$f_1(y)$	$80 + f_1(y-2)$	$f_2(y)$	(x_1*, x_2*, x_3*, x_4*)
0	**0**	80+(-∞) = -∞	0	(0,0,0,0)
1	**0**	80+(-∞) = -∞	0	(0,0,0,0)
2	0	80+0 = **80**	80	(0,1,0,0)
3	35	80+0 = **80**	80	(0,1,0,0)
4	35	80+0 = **80**	80	(0,1,0,0)
5	35	80+35 = **115**	115	(1,1,0,0)
6	35	80+35 = **115**	115	(1,1,0,0)

Tahap 3:

$$f_3(y) = \max\{f_2(y), p_3 + f_2(y - w_3)\}$$
$$= \max\{f_2(y), 60 + f_2(y - 2)\}$$

TABEL VIII. TAHAP 3

y			Solusi Optimum	
	$f_1(y)$	$80 + f_1(y-2)$	$f_2(y)$	(x_1*, x_2*, x_3*, x_4*)
0	**0**	60+(-∞) = -∞	0	(0,0,0,0)
1	**0**	60+(-∞) = -∞	0	(0,0,0,0)
2	**80**	60+0 = 60	80	(0,1,0,0)
3	**80**	60+0 = 60	80	(1,1,0,0)
4	80	60+80 = **140**	140	(0,1,1,0)

5	115	60+80 = **140**	140	(0,1,1,0)
6	115	60+80 = **140**	140	(0,1,1,0)

Tahap 4:

$$f_3 y = \max\{f_2(y), p_3 + f_2(y - w_3)\}$$
$$= \max\{f_2(y), 95 + f_2(y - 1)\}$$

TABEL IX. TAHAP 4

y			Solusi Optimum	
	$f_3(y)$	$95 + f_3(y-1)$	$f_4(y)$	(x_1*, x_2*, x_3*, x_4*)
0	**0**	95+(-∞) = -∞	0	(0,0,0,0)
1	**0**	95+0 = 95	95	(0,0,0,1)
2	80	95+0 = 95	95	(0,0,0,1)
3	80	95+80 = **175**	175	(0,0,1,1)
4	140	95+80 = **175**	175	(0,0,1,1)
5	140	95+140 = **235**	235	(0,1,1,1)
6	140	95+140 = **235**	235	(0,1,1,1)

Dengan demikian, didapatkan solusi optimum X = (0,1,1,1) dengan $\sum p = f = 235$.

UCAPAN TERIMA KASIH

Penulis ingin mengucapkan terima kasih kepada Tuhan Yang Maha Esa. Karena dengan rahmat yang diberikan-Nya, penulis dapat menyelesaikan makalah yang berjudul "Implementasi Progam Dinamis dalam Optimasi Kapasitas Kargo Logistik" tepat pada waktunya. Penulis juga ingin berterima kasih kepada Dr. Nur Ulfa Maulidevi ST,M.Sc. dan Dr. Ir. Rinaldi Munir, M.T selaku dosen pengajar mata kuliah Strategi Algoritma IF2211 yang sudah memberikan ilmunya dan bekerja keras memberikan yang terbaik untuk anak didiknya supaya kelak dapat mengharumkan nama bangsa. Tidak lupa penulis juga ingin berterima kasih kepada orangtua penulis. Karena tanpa doa dan jasanya, penulis tidak mungkin dapat mengerjakan makalah ini dengan baik.

REFERENSI

[1] Rinaldi Munir, "Diktat Strategi Algoritmik" Departemen Teknik Informatika. Bandung, 2004.

[2] Program Dinamis , http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Program-Dinamis-(2018).pdf

[3] Permasalahan Knapsack, https://piptools.net/permasalahan-knapsack/

[4] *Dynamic Programming, https://en.wikipedia.org/wiki/Dynamic_programming*

[5] https://www.quora.com/Whats-the-difference-between-greedy-algorithm-and-dynamic-programming-Is-a-greedy-program-a-subset-of-dynamic-programming

[6] https://sutrisnoadityo.wordpress.com/2013/10/12/program-dinamis/

[7] http://masud.lecture.ub.ac.id/files/2013/05/Pertemuan-ke-20-Program-Dinamik.pdf

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, 14 Mei 2018

Renjira Naufhal Dhiaegana
13516014