Penerapan Dynamic Programming dalam Penentuan Pengambilan Job dalam Euro Truck Simulator 2

Putu Arya Pradipta - 13515017
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13515017@std.stei.itb.ac.id

***Abstract*—Perkembangan permainan elektronik saat ini sangatlah pesat. Hal ini diakibatkan oleh teknologi yang semakin canggih. Dalam bermain, terkadang kita menggunakan konsep informatika secara tidak sadar untuk mengakselerasi perkembangan permainan kita. Dalam hal ini, penulis membahas penggunaan algoritma program dinamis dan juga persoalan integer knapsack untuk mempercepat naiknya level seorang pemain di Euro Truck Simulator 2.**

Keywords—program dinamis, knapsack, level up

I. PENDAHULUAN

Permainan berbasis elektronik saat ini sudah menjamur di seluruh dunia. Permainan ini lebih digandrungi dibandingkan dengan permainan di dunia nyata karena kemudahannya. Selain itu, adanya fitur multiplayer juga menunjang perkembangan dari permainan elektronik ini. Salah satu genre dalam permainan elektronik ini adalah genre simulasi. Pada genre ini, permainan meyimulasikan hal – hal yang sebenarnya ada di dunia nyata, namun sulit, seperti simulator pesawat tempur, dan juga permainan yang akan penulis bahas kali ini, yaitu permainan simulasi truk.

Euro Truck Simulator 2 merupakan permainan yang menyimulasikan kehidupan supir truk di Eropa. Permainan ini merupakan perbaikan dari Euro Truck Simulator 1 sebelumnya. Permainan ini terkenal akan realismenya yang bisa dibilang hampir mendekati aslinya. Keberadaan kotanya pun sama dengan di dunia nyata. Permaian ini pertama kali dirilis secara luas ke publik pada tanggal 12 Oktober 2012. Dalam permainan ini, kita ditugaskan untuk mengendarai truk keliling Benua Eropa dan membeli garasi di seluruh kota di Eropa. Permainan ini sudah terjual sekitar 3.5 juta kopi hanya melalui platform Steam saja. Permainan ini pun semakin berkembang dengan menambah *Downloadable Content* (DLC) ke negara – negara lain seperti Swedia, Prancis, dan juga Hungaria yang sebelumnya tidak termasuk dalam versi aslinya.

Dalam Euro Truck Simulator 2 ini, pertama - tama pemain diwajibkan untuk membuat sebuah profil dan memilih sebuah kota sebagai kantor pusat mereka. Pada awalnya player tidak memiliki uang untuk membeli sebuah truk, sehingga mereka harus melakukan *quick jobs* untuk menabung uang dan kemampuan sampai mereka dapat membeli sebuah truk ataupun mengambil utang. Setiap selesai seorang pemain menyelesaikan job, kemampuan player juga bertambah. Ketika seorang pemain naik level, maka player tersebut dapat menginvestasikan *skillpoint* tersebut ke salah satu dari 6 kemampuan yang tersedia, sehingga mereka dapat membawa barang – barang yang lebih berharga dan juga jarak yang lebih jauh. Pemain juga dapat membeli garasi – garasi di kota lainnya dan membayar *Non Playable Character* untuk bekerja untuknya. Dengan hal ini player mendapatkan uang secara lebih cepat.

Gambar 1 : Dunia Euro Truck Simulator 2

Sumber gambar :
https://eurotrucksimulator2.com/images-/hp_carousel/15.jpg

Pemilihan dan pengambilan job sangat penting dalam permainan ini, karena jika job yang diambil tepat maka level driver kita akan semakin cepat naik. Jika kita asal dalam mengambil job, maka kita akan sering tertahan di level bawah. Selain itu, untuk mensimulasikan dengan aslinya, maka Euro Truck Simulator 2 hanya memperbolehkan pengendara melakukan perjalanan selama 12 jam tanpa henti, yang juga berdasarkan aturan Uni Eropa. Dalam hal pemilihan ini penulis mencoba untuk melakukan pemilihan job menggunakan program

dinamis dengan sedikit memodifikasi persoalan integer(0/1) knapsack.

II. DASAR TEORI

A. *Program Dinamis*

Program dinamis adalah salah satu metode dalam menyelesaikan masalah dalam lingkup pemrograman. Metode ini mengumpulkan solusi dan menguraikannya menjadi berbagai tahapan (*state*). Tahapan tersebut disusun sedemikian sehingga solusi dari persoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan. *State* atau keadaan ditentukan berdasarkan situasi-situasi yang mungkin terjadi dalam setiap tahapan pencarian solusi dari permasalahan yang ada.

Syarat pertama agar suatu permasalahan dapat diselesaikan dengan program dinamis adalah permasalahan tersebut harus dapat dipecah-pecah menjadi keputusan - keputusan dengan jumlah yang terbatas.

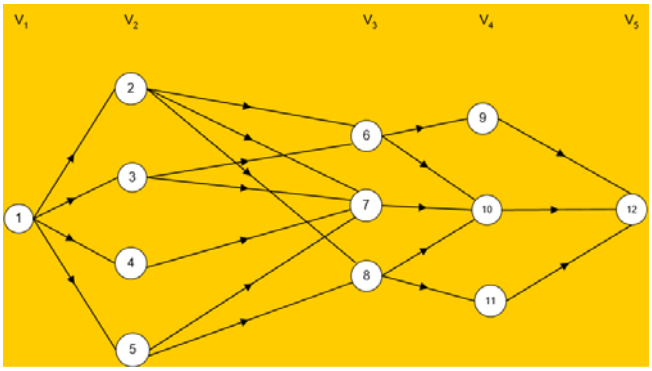
Ketika subproblem persoalan yang sama muncul, program cukup melihat ke perhitungan sebelumnya alih – alih menghitung kembali. Diharapkan ini dapat mengurangi waktu komputasi dari sebuah problem dengan mengorbankan sedikit memori untuk menyimpan data. Teknik ini sering disebut dengan memoisasi. Teknik ini dapat menghindarkan kita dari jebakan minima lokal di greedy dengan menghitung pilihan – pilihan lainnya.

Dalam program dinamis ini, program dijalankan dengan sifat rekurens, dimana hasil sebelumnya menjadi dasar dari hasil selanjutnya. Terdapat 2 pendekatan dalam program dinamis, yaitu top-down dan bottom-up. Perbedaannya adalah jika di top down kita menghitung dari dasar atau basis, sedangngkan di bottom up kita menghitung dari subproblemnya terlebih dahulu hingga sampai ke basis. Kedua pendekatan ini sama baiknya dibandingkan dengan satu sama lain.

Selain itu terdapat beberapa karakteristik dari persoalan – persoalan yang mungkin dapat diselesaikan dengan program dinamis, yaitu

1. Terdapat sejumlah berhingga pilihan yang mungkin.
2. Solusi dibangun secara bertahap.
3. Solusi pada setiap tahap dibangun dari hasil solusi tahap sebelumnya.
4. Menggunakan persyaratan optimasi dan kendala untuk membatasi sejumlah pilihan yang harus dipertimbangkan pada suatu tahap.

Program dinamis ini memanfaatkan konsep graf multi tahap. Graf multitahap ini adalah sebuah graf dimana tiap simpul di dalam graf tersebut menyatakan status, sedangkan V1, V2 menyatakan tahap.



Gambar 2 : Graf Multi Level

Sumber :
http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2014-2015/Program%20Dinamis%20(2015).pdf

Meskipun terlihat mirip dengan greedy, namun terdapat perbedaan diantara program dinamis dan greedy, diantaranya dalam greedy, hanya satu rangkaian keputusan yang dihasilkan. Sedangkan dalam program dinamis, lebih dari satu rangkaian keputusan yang dipertimbangkan.

Secara matematis, biaya pada tahap k+1 dapat dituliskan sebagai berikut.

$F_{k+1}(x) = c_k+1 + F_k(x)$

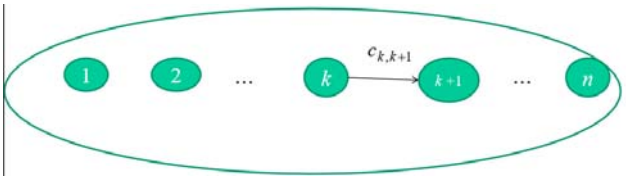
C_k+1 adalah biaya optimum tahap ke k+1

$F_{k+1}(x)$ adalah total biaya optimum hingga tahap ke k +1

Pada program dinamis, rangkaian keputusan yang optimal dibuat dengan menggunakan prinsip optimalitas. Prinsip optimalitas berbunyi sebagai berikut,

> Jika solusi total optimal, maka bagian solusi sampai tahap ke-k juga optimal.

Prinsip optimalitas berarti bahwa jika kita bekerja dari tahap k ke tahap k+1, kita dapat menggunakan hasil optimal dari tahap k tanpa harus kembali ke tahap awal. Ongkos pada tahap k+1 didefinisikan sebagai ongkos yang dihasilkan pada tahap k ditambah ongkos dari tahap k ke tahap k+1.



Gambar 3 : Prinsip Optimalisasi

Sumber :
http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2014-2015/Program%20Dinamis%20(2015).pdf

Karakteristik persoalan program dinamis adalah sebagai berikut :

1. Persoalan dapat dibagi menjadi beberapa tahap (*stage*), yang pada setiap tahap hanya diambil satu keputusan

2. Masing – masing tahap terdiri dari sejumlah status (*state*) yang berhubungan dengan tahap tersebut. Secara umum status merupakan bermacam kemungkinan masukan yang ada pada tahap tersebut.
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 k memberikan keputusan terbaik untuk setiap status pada tahap k+1
8. Prinsip optimalitas berlaku pada persoalan tersebut

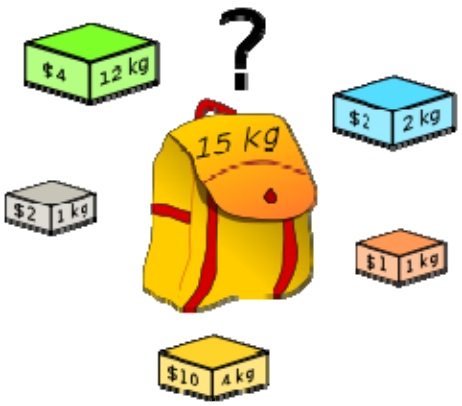
Beberapa langkah langkah pengembangan program dinamis untuk menyelesaikan suatu masalah informatika adalah sebagai berikut :

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

Program dinamis dapat menyelesaikan beberapa macam masalah, seperti pencarian lintasan terpendek, *travelling salesman problem*, dan juga *knapsack problem*. Dalam bagian selanjutnya, akan dibahas pemecahan masalah pengambilan job dengan algoritma program dinamis

B. *Knapsack problem*

Knapsack problem ini adalah persoalan optimasi dimana kita dituntut untuk memasukkan barang dengan berat dan profit tertentu ke dalam suatu tas. Tujuan yang ingin dicapai adalah terdapat profit maksimal yang didapat, namun berat seluruh benda tersebut tidak boleh lebih dari berat maksimal tas. Terdapat 2 macam persoalan knapsack, yaitu 0/1 knapsack dan fractional knapsack. Disebut 0/1 karena barang yang bisa dimasukkan di dalam tas pada persoalan ini tidak dapat direpresentasikan dalam bentuk pecahan. Sedangkan dalam *fractional knapsack* benda dapat dipecah – pecah sehingga dengan memilih profit/berat tertinggi saja sudah cukup untuk menyelesaikan masalah ini.



Gambar 4 : Ilustrasi Knapsack problem

Sumber :
https://en.wikipedia.org/wiki/Knapsack_problem

Secara formal, dapat dideskripsikan knapsack problem adalah: Diberikan n buah objek dan sebuah knapsack dengan maksimal bobot sebesar K. Setiap objek yang akan dimasukkan ke dalam knapsack memiliki properti bobot (weight) w_i dan keuntungan (profit) p_i. Persoalannya adalah bagaimana memilih objek – objek yang akan dimasukkan ke dalam knapsack sehingga tidak melebihi kapasitas knapsack namun memaksimalkan total keuntungan yang diperoleh. Solusi persoalan ini dinyatakan sebagai vektor n-tuple:

$X = \{x_1, x_2, \ldots, x_n\}$

Yang dalam hal ini, $x_i = 1$ jika objek ke-$_i$ dimasukkan ke dalam knapsack, atau $x_i = 0$ jika objek ke-i tidak dimasukkan. Atas dasar itulah maka persoalan ini disebut 0/1 *knapsack*.

Secara matematis persoalan ini dapat dirumuskan sebagai berikut.

Maksimasi $F = \sum p_i x_i$

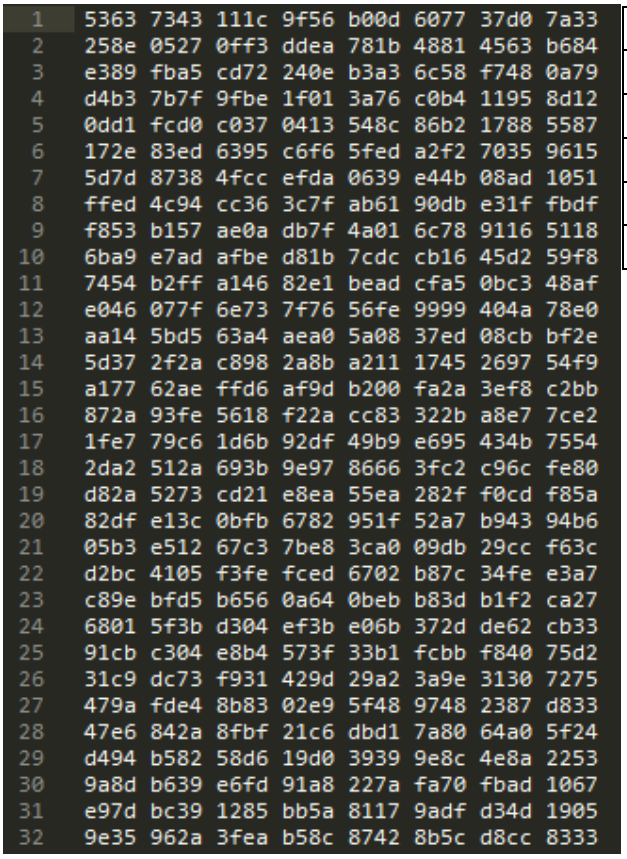
Dengan kendala : $\sum w_i \leq W$

Masalah pengambilan job ini sangatlah mirip dengan 0/1 *knapsack problem*, karena memiliki batasan tertentu dan ingin mendapatkan profit (dalam hal ini experience) tertinggi.

III. Pembahasan

A. *Save game dalam Euro Truck Simulator 2*

Sebelum mulai membahas algortima program dinamis yang digunakan, penulis ingin membahas format yang digunakan oleh Euro Truck Simulator 2 untuk menyimpan *savegame*. Tidak seperti permainan lainnya yang menutup secara rapat *savegame* mereka agar tidak ada yang mengutak - atiknya, Euro Truck Simulator 2 memberikan akses kepada *savegame*, namun data tersebut telah dienkripsi sedemikian rupa.



Gambar 5 : Cuplikan *Savegame*

Sumber : Dokumentasi Pribadi

Oleh karenanya, kita membutuhkan sebuah perangkat lunak untuk melakukan dekripsi file tersebut. Penulis menggunakan perangkat lunak yang dibuat oleh seorang member dalam *scsforum* yang berfungsi untuk melakukan dekripsi *savegame*. Hasil keluaran dari program tersebut adalah *savegame* yang sudah didekripsi dan dapat dibaca oleh manusia.

B. Penentuan EXP dalam job

Di dalam dunia Euro Truck Simulator 2, EXP didapat tidak hanya dari pengambilan job, namun juga dari *free roaming* atau hanya berkeliling tanpa pekerjaan. Namun, persentasenya sangat kecil jika dibandingkan dengan exp dari job. Pada dasarnya, setiap job akan mendapatkan exp sejauh jarak darat dari kedua tempat tersebut. Misalnya jika kita mengambil job dengan jarak 542 KM, maka kita akan mendapatkan exp sebesar 542 juga. Jika job kita mengharuskan kita untuk menaiki ferry ataupun kereta, maka exp yang kita dapatkan hanyalah dari jarak daratnya saja. Misalkan jaraknya adalah 1300 km namun harus menaiki kapal sejauh 700 km, maka EXP yang kita dapatkan hanyalah 600 km. Selain itu, skill juga sangat mempengaruhi besarnya EXP yang kita dapatkan.

Berikut ini merupakan daftar dari kemampuan yang mendapatkan bonus untuk EXP serta besarannya

Tabel 1 Daftar Skill dan Bonus

Nama Skill	EXP Bonus
ADR	+21%
Fragile Cargo	+22%
Important Delivery	+20%
Urgent Delivery	+30%
High Value Cargo	+18%
Long Distance Cargo	+25%

Untuk Long Distance Cargo, jarak yang dihitung adalah jarak darat dan juga jarak ferry. Jika total jaraknya lebih dari 250 km, maka job tersebut juga akan mendapatkan *long distance cargo bonus*. Kombinasi terbesar adalah jika kita mengambil job yang merupakan Long Distance Cargo, ADR, Fragile Cargo, dan Urgent Delivery. Kita sudah mendapatkan bonus sebesar 98% dari kombinasi tersebut.

Selain dari skill, kita dapat mendapatkan EXP tambahan dari memakirkan trailer yang kita bawa. Terdapat 2 pilihan, yaitu memakirkan trailer secara langsung ke tempatnya yang mendapatkan 40 EXP, memakirkan di sisi jalan yang lebih mudah dan mendapat 15 EXP, dan tidak memakirkan dan langsung diberikan ke perusahaan dan tidak mendapatkan bonus EXP. Di dalam program ini penulis mengasumsikan bahwa semua trailer akan dipakirkan secara langsung ke tempatnya sehingga mendapatkan bonus yang besar.

C. Jenis - jenis kargo

Ada banyak sekali jenis – jenis kargo dalam Euro Truck Simulator 2, dan berikut ini merupakan kargo – kargo yang memiliki bonus seperti *fragile*, *high value*, maupun ADR. Attribut Important dan Urgent Cargo dilihat dalam pengajuan job dan tidak tergantung pada jenis kargonya, sedangkan *long distance cargo* hanya bergantung pada jarak kargo. Penulis mengambil sampel sebanyak 30 jenis kargo karena masih banyak kargo lainnya yang memiliki bonus.

Tabel 2 Daftar Kargo Sampel

Jenis Kargo	Fragile	High Value	ADR
cargo.helicopter	1	1	0
cargo.pot_flowers	1	0	0
cargo.comp_process	1	1	0
cargo.med_equip	1	1	0
cargo.yacht	1	1	0
cargo.cars_fr	1	1	0
cargo.mason_jars	1	0	0
cargo.electronics	1	0	0
cargo.digger500	0	1	0
cargo.digger500	0	1	0
cargo.nitrogen	1	0	1
cargo.dryers	1	0	0
cargo.beverages	1	0	0
cargo.petrol	0	0	1

cargo.aircond	1	1	0
cargo.fireworks	1	0	1
cargo.forklifts	0	1	0
cargo.acetylene	1	0	1
cargo.fueltanker	0	0	1
cargo.neon	1	0	1
cargo.hipresstank	0	1	0
cargo.transmis	1	0	0
cargo.phosphor	1	0	1
cargo.tractors	0	1	0
cargo.live cattle	1	0	0
cargo.volvo tr	1	0	0
cargo.tableware	1	0	0
cargo.diggers	0	1	0
cargo.nitrocel	1	0	1
cargo.cyanide	0	0	1

Keterangan : 1 menandakan ya, dan 0 menandakan tidak.

D. Algoritma

Pseudocode dari algoritma yang penulis gunakan adalah sebagai berikut

1. Input jumlah menit kerja maksimal yang diinginkan
2. Input jumlah jarak minimal job
3. Lakukan pembacaan data kargo
4. Lakukan pembacaan data *savegame*. Jika data merupakan job_offer_data, maka kalkulasi exp dan simpan data company, destination, cargo, dan juga jarak beserta EXP nya ke dalam larik.
5. Tutup semua file eksternal
6. Lakukan program dinamis knapsack
7. Cetak exp maksimal yang bisa didapatkan

Untuk algoritma program dinamis, penulis menggunakan matriks berukuran w+1 dan n+1, dimana n merupakan jumlah job yang tersedia dan w merupakan beban maksimal. Beban maksimal disini dihitung dari jumlah menit kerja maksimal yang diasumsikan bahwa setiap menit kerja dapat menempuh 1 kilometer. Pseudocode program dinamis adalah sebagai berikut

Untuk setiap job I {

Untuk setiap beban w {

Jika job pertama atau beban = 0 maka matriks diisi 0

Jika job ke i lebih kecil atau sama dengan w maka hasil iterasi ke I adalah cari maksimal dari exp job ke i ditambah dengan hasil iterasi sebelumnya berbeban w dikurang beban job ke I dengan hasil iterasi sebelumnya dengan beban w

Jika job ke I lebih besar dari w maka hasil iterasi ke I adalah hasil iterasi sebelumnya

}

}

Jawaban adalah cell terakhir dari matriks (baris ke n dan kolom ke w)

Implementasi penyelesaian persoalan integer knapsack dalam program dinamis dalam bahasa c++ adalah sebagai berikut :

```
int knapsack(int W, vector<Job>& array, int n) {
    int i, w;
    int K[n+1][W+1];
   for (i = 0; i <= n; i++) {
       for (w = 0; w <= W; w++) {
           if (i == 0 || w == 0) {
               K[i][w] = 0;
           } else if (array[i-1].length <= w) {
                 K[i][w] = max(array[i-1].exp + K[i-
1][w-(array[i-1].length)],  K[i-1][w]);
           } else {
                 K[i][w] = K[i-1][w];
           }
       }
   }
   return K[n][W];
}
```

Sedangkan dalam pembacaan file eksternal, penulis tidak menggunakan library apapun dan diparse secara hard-coded.

Hasil dari program adalah sebagai berikut

```
aryapradipta9@ARYAPRADIPTA:/mnt/c/Users/lenovo/Downloads/scsc$ ./stima
Masukkan jumlah jam kerja yang diinginkan (maks 720) : 300
Masukkan minimum jarak yang diinginkan (min 0) : 0
Jumlah EXP maksimal yang didapat adalah 557
aryapradipta9@ARYAPRADIPTA:/mnt/c/Users/lenovo/Downloads/scsc$ ./stima
Masukkan jumlah jam kerja yang diinginkan (maks 720) : 600
Masukkan minimum jarak yang diinginkan (min 0) : 20
Jumlah EXP maksimal yang didapat adalah 1158
aryapradipta9@ARYAPRADIPTA:/mnt/c/Users/lenovo/Downloads/scsc$ ./stima
Masukkan jumlah jam kerja yang diinginkan (maks 720) : 300
Masukkan minimum jarak yang diinginkan (min 0) : 100
Jumlah EXP maksimal yang didapat adalah 557
aryapradipta9@ARYAPRADIPTA:/mnt/c/Users/lenovo/Downloads/scsc$ ./stima
Masukkan jumlah jam kerja yang diinginkan (maks 720) : 300
Masukkan minimum jarak yang diinginkan (min 0) : 300
Jumlah EXP maksimal yang didapat adalah 0
aryapradipta9@ARYAPRADIPTA:/mnt/c/Users/lenovo/Downloads/scsc$ ./stima
Masukkan jumlah jam kerja yang diinginkan (maks 720) : 300
Masukkan minimum jarak yang diinginkan (min 0) : 150
Jumlah EXP maksimal yang didapat adalah 557
aryapradipta9@ARYAPRADIPTA:/mnt/c/Users/lenovo/Downloads/scsc$ ./stima
Masukkan jumlah jam kerja yang diinginkan (maks 720) : 720
Masukkan minimum jarak yang diinginkan (min 0) : 0
Jumlah EXP maksimal yang didapat adalah 1411
```

Gambar 6 Contoh hasil program

Sumber : Dokumentasi pribadi

IV. Kesimpulan

Program dinamis merupakan salah satu alat untuk menyelesaikan masalah dalam dunia informatika. Selain itu, tanpa disadari banyak dari permasalahan di dunia nyata juga dapat dimodelkan dengan permasalahan informatika seperti integer knapsack ini. Oleh karenanya, penting bagi kita untuk memodelkan masalah – masalah di dunia ini dengan pemodelan informatika agar ilmu ini semakin bermanfaat di dunia.

Ucapan Terima Kasih

Penulis berterima kasih kepada Tuhan Yang Maha Esa karena atas berkat dan karunia-Nya, karya tulis ini dapat penulis selesaikan tanpa rintangan yang berarti. Penulis juga ingin berterima kasih kepada Ibu Dr. Nur Ulfa Maulidevi selaku dosen K2 IF2211 Strategi Algoritma yang telah memberikan ilmu tentang program dinamis dan integer knapsack yang menjadi dasar dari penulisan makalah ini.

Referensi

[1] Munir, Rinaldi, Diktat Kuliah IF2211, Strategi Algoritma, Program Studi Teknik Informatika, STEI, ITB, 2009

[2] Referensi Standard Template Library tentang Vector, http://www.cplusplus.com/reference/vector/vector/, diakses tanggal 17 Mei 2017

[3] http://truck-simulator.wikia.com/, diakses tanggal 16 Mei 2017

[4] https://en.wikipedia.org/wiki/Euro_Truck_Simulator_2, diakses tanggal 15 Mei 2017.

[5] http://forum.scssoft.com/viewtopic.php?f=34&t=164103&start=200, diakses tanggal 16 Mei 2017

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, 18 Mei 2017

Putu Arya Pradipta
13515017