Penerapan Algoritma Branch and Bound dalam Menentukan Kegiatan Olahraga dengan Pembakaran Kalori yang Optimal

Kadek Dwi Bagus Ananta Udayana - 13519057
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13519057@std.stei.itb.ac.id

Abstract*—Untuk mendapatkan tubuh yang ideal, tidak hanya didapatkan dengan mengurangi asupan makanan saja. Output yang dikeluarkan tubuh kita juga harus ditambah. Output yang dikeluarkan tidak hanya keluar di akhir pencernaan kita saja, tetapi output bisa keluar dari keringat. Ya, olahraga merupakan salah satu solusi yang tepat untuk mendapatkan tubuh ideal. Akan tetapi tidak semua orang mengetahui cara olahraga dengan tepat dan mendapatkan hasil yang tepat juga. Dengan kata lain, tidak semua orang mengetahui cara efektif mengeluarkan kalori secara optimal. Pada makalah ini akan dibahas mengenai penentuan olahraga yang efektif untuk dilakukan dengan menggunakan algoritma *Branch & Bound.

Keywords—Branch & Bound, Olahraga, Kalori. Knapsack (0/1) Problem.

I. PENDAHULUAN

Olahraga merupakan salah satu cara yang paling mudah dan praktis untuk kita implementasikan agar tubuh kita menjadi bugar. Olahraga masih saja dipandang sebelah mata oleh kebanyakan orang. Padahal berolahraga dengan rutin dapat membuat tubuh kita menjadi sehat. Tak hanya berguna bagi kesehatan secara fisik, tetapi juga baik untuk kesehatan mental.

Berbicara tentang olahraga, terdapat olahraga yang mudah dan *fleksible* untuk kita lakukan dimanapun. Olahraga tersebut adalah *kalistenik* dan *cardio. Kalistenik* adalah olahraga yang menggunakan atau mengandalkan berat badan untuk membakar kalori dan melatih otot-otot. Sehingga olahraga ini bisa dilakukan tanpa menggunakan alat. Adapaun contoh olahraga ini adalah *push-up, pull-up, sit-up, lunges, jumping jack* dan masih banyak lagi. Olahraga kedua yang bisa dilakukan dengan mudah adalah *cardio*. Kata *cardio* sebenarnya berhubungan dengan jantung. Lebih lengkapnya olahraga *cardio* merupakan olahraga yang dilatih untuk meningkatkan asupan oksigen ke dalam jantung. *Cardio* merupakan olahraga berjenis *aerobic* yang memerlukan banyak oksigen sehingga akan terjadi pembakaran lemak yang cukup banyak. Akan tetapi perlu diingat juga bahwa olahraga *cardio* harus diimbangi dengan olahraga yang menggunakan beban. Karena pada olahraga cardio sebenarnya selain membakar kalori juga mengurangkan massa otot. Oleh karenanya diperlukan olahraga menggunakan beban, untuk menjaga massa otot.

Gambar 1. Macam-macam olahraga (Sumber: https://hengkyharwanto.wordpress.com/2015/03/21/latihan-kardio/)

II. LANDASAN TEORI

A. Algoritma *Branch and Bound*

Algoritma *Branch & Bound* atau B&B merupakan algoritma yang digunakan untuk persoalan optimasi. Optimasi yang dimaksud adalah meminimalkan atau memaksimalkan suatu fungsi objektif yang tidak melanggar suatu batasan atau constraint persoalan. Algoritma *Branch & Bound* merupakan penggabungan algoritma *Breadth First Search* (BFS) ditambah dengan *Least Cost Search*. Sehingga pada algoritma ini simpul berikutnya juga akan diekspansikan berdasarkan urutan pembangkitannya (FIFO). Pada algoritma *Branch & Bound*, setiap simpul akan diberikan sebuah nilai *cost* atau nilai sebesar ĉ(i) yang definisinya adalah nilai taksiran lintasan termurah ke simpul status tujuan yang melalui simpul status i. Lalu Simpul berikutnya yang akan di-expand tidak lagi berdasarkan urutan pembangkitannya, tetapi simpul yang memiliki *cost* atau nilai yang paling kecil (*least cost search*)

pada kasus minimasi. Sedangkan yang memiliki *cost* yang paling besar pada kasus maksimasi.

Algoritma *Branch & Bound* juga memiliki fungsi pembatas yang menerapkan pemangkasan pada jalur yang tidak lagi mengarah ke solusi. Secara umum kriteria pemangkasan adalah

1. Nilai simpul tidak lebih baik dari nilai terbaik sejauh ini
2. Simpul tidak merepresentasikan solusi yang layak karena ada batasan yang dilanggar
3. Solusi pada simpul tersebut hanya terdiri atas satu titik tidak ada pilihan lain lalu bandingkan nilai fungsi objektif dengan solusi terbaik saat ini dan ambil yang terbaik

B. *Knapsack (0/1) Problem*

Gambar 2. *Knapsack (0/1) problem* (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branchand-Bound-2021-Bagian4.pdf)

Knapsack Problem (KP) adalah masalah penempatan item atau barang kedalam suatu tempat yang biasa disebut *Knapsack* yang mempunyai kapasitas tertentu, dimana setiap item memiliki berat dan nilai, sehingga total berat dari item-item yang ditempatkan tidak melebihi kapasitas *Knapsack* dan nilai yang didapatkan maksimum. Lebih jelasnya, pada persoalan ini diberikan n buah objek dan sebuah *knapsack* dengan kapasitas K. Setiap objek memiliki properti bobot (weight) w_i dan keuntungan (profit) p_i. Disebut *knapsack (0/1) problem*, karena suatu objek hanya dapat dimasukkan ke dalam *knapsack* yang bernilai 1 atau tidak dimasukkan yang bernilai 0

Knapsack 0/1 Problem dapat kita pandang sebagai mencari himpunan bagian atau *subset* dari himpunan n objek yang dapat dimuat ke dalam knapsack dan memberikan total keuntungan terbesar.

C. Kegiatan Olahraga

Seperti yang telah dibahas sebelumnya. Kegiatan olahraga yang paling mudah dilakukan dimana saja dan tidak memerlukan banyak alat adalah olahraga *kalistenik* dan olahraga *cardio*. Berikut adalah contoh kegiatan olahraga *kalistenik* dan *cardio*.

1. *Push Up*

   Push Up adalah olahraga yang bermanfaat untuk membangun kekuatan tubuh bagian atas. Fungsi *push up* adalah untuk melatih trisep, otot dada, dan bahu. Jika dilakukan dengan bentuk yang tepat, olahraga ini juga dapat memperkuat punggung bawah dengan menarik otot perut. Diperkirakan jika melakukan *push up* selama 15 menit, akan mengeluarkan kalori sekitar 150 kalori

Gambar 3. *Push up* (Sumber : https://hellosehat.com/)

2. *Pull Up*

   Pull Up merupakan latihan olah otot yang dilakukan dengan memanfaatkan sebuah palang sebagai pegangan atau daya angkatnya. Otot tubuh yang terlatih ketika melakukan gerakan *pull up* di antaranya adalah otot punggung dan otot bisep. Diperkirakan jika melakukan *pull up* selama 15 menit, akan mengeluarkan kalori sekitar 150 kalori

Gambar 4. *Pull up* (Sumber : https://www.idntimes.com/)

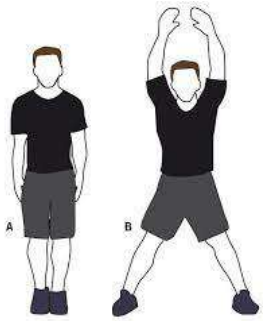
3. *Sit Up*

   Sit up adalah salah satu latihan otot yang paling sering dilakukan. Olahraga satu ini berguna untuk melatih otot perut. *Sit up* adalah latihan penguatan perut yang dapat dilakukan tanpa peralatan apa pun. Diperkirakan jika melakukan *sit-up* selama 20 menit, akan mengeluarkan kalori sekitar 200 kalori

Gambar 5. *Sit up* (Sumber : https://www.gq.com/)

4. *Jumping jack*

Jumping jack adalah satu gerakan yang bertujuan untuk meningkatkan daya tahan tubuh, menyehatkan dan memadatkan tulang, serta menguatkan otot seluruh badan. Olahraga ini juga bisa membantu meningkatkan lompatan kalian saat bertanding di lapangan. Diperkirakan jika melakukan *jumping-jack* selama 20 menit, akan mengeluarkan kalori sekitar 200 kalori



Gambar 6. *Jumping jack* (Sumber : http://femaleez.com/)

5. *Running*

Running atau Lari merupakan olahraga yang mudah untuk dilakukan. Olahraga lari adalah langkah cepat yang pada saat dilakukan, membuat tubuh jadi memiliki kecenderungan melayang akibat hanya ada satu kaki yang menjejak tanah dalam satu waktu. Diperkirakan jika melakukan *running* selama 30 menit, akan mengeluarkan kalori sekitar 300 kalori

Gambar 7. *Running* (Sumber : http://healthline.com/)

6. *Skipping*

Olahraga *skipping* atau secara sederhana bisa disebut olahraga lompat tali ini dilakukan dengan menggunakan bantuan tali yang diputar atau digerakkan sebagai rintangan dari lompatan yang dilakukan dengan menggunakan kedua tangan kita sebagai porosnya. Diperkirakan jika melakukan *skipping* selama 25 menit, akan mengeluarkan kalori sekitar 300 kalori

Gambar 8. *Skipping* (Sumber : http://doktersehat.com/)

7. *Plank*

Plank adalah latihan yang bisa Anda lakukan di mana pun dan tanpa menggunakan alat apa pun. Meski demikian, penting bagi Anda untuk memahami gerakan dan manfaat *plank* sebelum mempraktikan latihan ini ke dalam olahraga harian. Dalam hal ini kalori yang dikeluarkan plank penulis anggap sama dengan *corepower* yoga. Diperkirakan jika melakukan *plank* selama 10 menit, akan mengeluarkan kalori sekitar 60 kalori

Gambar 9. *Plank* (Sumber : http://kompasiana.com/)

8. *Lunges*

Lunges adalah latihan kekuatan yang sering dilakukan ketika seseorang ingin memperkuat, membentuk, dan mengencangkan tubuh mereka, sekaligus meningkatkan kebugaran tubuh secara keseluruhan, serta meningkatkan kinerja atletik. Dalam hal ini kalori yang dikeluarkan lunges, penulis anggap sama dengan olahraga aerobic secara umum. Diperkirakan jika melakukan *lunges* selama 20 menit, akan mengeluarkan kalori sekitar 160 kalori

Gambar 10. *Lunges* (Sumber : http://24life.com/)

Selain delapan contoh diatas, sebenarnya masih banyak lagi kegiatan olahraga *kalistenik* dan olahraga *cardio*. Akan tetapi pada makalah ini, hanya delapan olahraga diatas saja yang penulis gunakan untuk diselesaikan dengan algoritma *Branch & Bound*.

III. PEMBAHASAN

Pada pemilihan kegiatan olahraga berikut, olahraga yang dianggap ideal adalah olahraga yang dilakukan kurang lebih 60 menit. Sehingga penulis membuat nilai K = 60 menit atau olahraga akan dilakukan selama 60 menit

Tabel 1. Daftar Olahraga Beserta Durasi dan Kalori

No	Olahraga	Durasi (W) (menit)	Kalori (P) (kal)
1	*Push Up*	15	148
2	*Pull Up*	15	142
3	*Sit Up*	20	197
4	*Jumping jack*	20	207
5	*Running*	30	300
6	*Skipping*	25	312
7	*Plank*	10	62
8	*Lunges*	20	162

Bentuk *source code* pengisian tabel adalah sebagai berikut.

```
# Fungsi Isi Tabel
def isiTable(table):
    jumlah = len(table)
    for i in range(jumlah):
        print("OLAHRAGA", i + 1)
        table[i][0] = input("masukkan nama olahraga = ")
        table[i][1] = int(input("masukkan waktu olahraga = "))
        table[i][2] = int(input("masukkan nilai kalori yang dihasilkan = "))
        table[i][3] = table[i][2]/table[i][1]
    return table
```

Agar pencarian solusi lebih mangkus, maka objek-objek diurutkan berdasarkan p_i/w_i yang menurun (dari besar ke kecil) sebagai berikut:

Tabel 2. Daftar Kegiatan Olahraga Beserta Nilai p_i/w_i

No	Olahraga	Pi/Wi (kal/menit)
1	*Skipping*	12.48
2	*Jumping Jack*	10.35
3	*Running*	10
4	*Push Up*	9.87
5	*Sit Up*	9.85
6	*Pull Up*	9.47
7	*Lunges*	8.1
8	*Plank*	6.2

Bentuk *source code* pengurutan tabel adalah sebagai berikut.

```
# Fungsi Urut Tabel
def urutTable(table):
    jumlah = len(table)
    for i in range(jumlah):
        for j in range(i + 1, jumlah):
            if table[j][3] > table[i][3]:
                temp = table[i]
                table[i] = table[j]
                table[j] = temp
  return table
```

Persoalan *knapsack* ini merupakan persoalan maksimasi. Algoritma *Branch & Bound* akan menyelesaikan persoalan ini dengan membentuk pohon ruang status. Pohon ruang statusnya berbentuk pohon biner. Cabang kiri menyatakan objek i dipilih ($x_i = 1$), cabang kanan menyatakan objek i tidak dipilih ($x_i = 0$).

Tiap simpul diisi dengan total bobot knapsack yang sudah terpakai (W) dan total keuntungan yang sudah dicapai (F). Cost atau batas atas simpul i dihitung sebagai penjumlahan total keuntungan yang sudah dicapai (F) ditambah dengan perkalian sisa kapasitas knapsack (K – W) dengan rasio keuntungan per bobot objek yang tersisa berikutnya ($p_i + 1/w_i + 1$), atau dengan rumus:

$$\hat{c}(i) = F + (K - W)p_{i+1}/w_{i+1}$$

Jika implementasi program dibuat dalam source code, tidak efektif jika harus divisualisasikan dalam pohon ruang status. Terdapat ide lain dengan cara menggunakan *Prio Queue*. Sebenarnya hal ini memiliki ide yang sama dengan pohon ruang status, yaitu simpul yang diekspansi adalah simpul yang memiliki nilai ĉ(i) terbesar.

Elemen pada *Prio Queue* ini adalah sebuah simpul yang bertipe list dengan elemen pertama simpul adalah total bobot yang telah digunakan, elemen kedua simpul adalah total keuntungan yang telah diperoleh, elemen ketiga simpul adalah tingkat kedalaman pohon ruang status, elemen keempat simpul adalah nilai ĉ(i), dan elemen kelima simpul adalah jalur yang dilalui dari akar sampai ke simpul tersebut.

Penambahan elemen ke dalam *Prio Queue* harus memperhatikan besaran nilai ĉ(i), sehingga *Prio Queue*

akan selalu terurut berdasarkan nilai ĉ(i). Dengan demikian simpul yang di pop akan selalu elemen pertama.

Berikut merupakan implementasi berupa source code yang telah dibuat secara modular.

```
# Fungsi cost ĉ(i)
def cost(nilaiF, nilaiK, nilaiW, nilaipw):
  return nilaiF + (nilaiK - nilaiW)*nilaipw
```

```
# Fungsi Print Hasil
def printHasil(simpul, table):
    print("Olahraga yang diambil adalah : ")
    for i in range(len(simpul[4])):
        if simpul[4][i] == 1:
          print(table[i][0])
```

```
# Fungsi Branch & Bound
def branchAndBound(table):
    queue = []
    K = 60
    F = 0
    W = 0
    i = 0
    Ci = cost(F, K, W, table[i][3])
    jalur = []
    queue.append([W, F, i, Ci, jalur])
    while len(queue) != 0 and W != K:
        cut = True
        stop = False
        simpul = queue.pop(0)
        # apakah harus di block atau tidak
        while cut:
            if simpul[0] > K:
                if len(queue) != 0:
                    simpul = queue.pop(0)
                else:
                    stop = True
                    cut = False
            else:
                cut = False

        # append simpul anak
        if not cut and not stop and i < len(table) - 1:
            W = simpul[0]
            F = simpul[1]
            i = simpul[2]
            jalur = simpul[4]
            # untuk kegiatan yang di ambil
            Fambil = F + table[i][2]
            Wambil = W + table[i][1]

            i = i + 1
            Ci = cost(Fambil, K, Wambil, table[i][3])

            jalur.append(1)
            newJalur = []
            for z in range(len(jalur)):
                newJalur.append(jalur[z])
            queue.append([Wambil, Fambil, i, Ci, newJalur])
            jalur.pop()

            # untuk kegiatan yang tidak di ambil
            Ci = cost(F, K, W, table[i][3])
            jalur.append(0)
            queue.append([W, F, i, Ci, jalur])
            # urutkan queue
            queue = urutTable(queue)
    return simpul
```

```
# Fungsi Main Program
def mainProgram():
    jumlah = int(input("Masukka berapa Jumlah Olahraga = "))
    table = [[0 for j in range(4)] for i in range(jumlah)]
```

```
        table = isiTable(table)
        table = urutTable(table)
        simpul = branchAndBound(table)
        printHasil(simpul, table)
mainProgram()
```

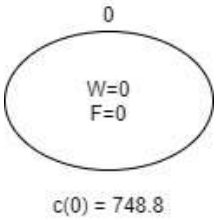
Agar penjelasan terasa lebih jelas, berikut adalah tahap-tahap menyelesaikan persoalan ini dengan menggunakan visualisasi pohon ruang status.

Tahap 1

Bangkitkan simpul akar (simpul 0), W = 0. F = 0 karena belum ada yang dipilih.

$\hat{c}(0) = F + (K - W)p_1/w_1 = 0 + (60 - 0)(12.48)$

$= 748.8$



Gambar 11. Simpul Akar(Sumber : milik pribadi)

Tahap 2

Bangkitkan simpul anak kiri (simpul 1) dan simpul anak kanan (simpul 2) dari simpul akar

1. Simpul 1 (Olahraga *skipping* dipilih)

   W = 0 + 25 = 25

   F = 0 + 312 = 312

   $\hat{c}(1) = F + (K - W)p_2/w_2 = 312 + (60 - 25)(10.35)$

   $= 674.25$

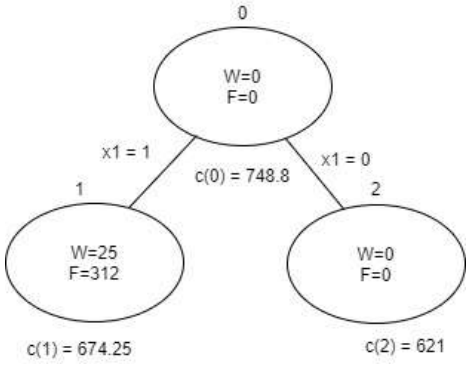
2. Simpul 2 (Olahraga *skipping* tidak dipilih)

   W = 0

   F = 0 + 0 = 0

   $\hat{c}(2) = F + (K - W)p_2/w_2 = 0 + (60 - 0)(10.35)$

   $= 621$



Gambar 12. Ekspansi Tahap 2 (Sumber : milik pribadi)

Simpul hidup adalah 1 dan 2. Karena simpul 1 memiliki cost paling besar, maka simpul 1 selanjutnya yang akan diekspansi.

Tahap 3

Bangkitkan simpul anak kiri (simpul 3) dan simpul anak kanan (simpul 4) dari simpul 1

1. Simpul 3 (Olahraga *jumping jack* dipilih)

   W = 25 + 20 = 45

   F = 312 + 207 = 519

   $\hat{c}(3) = F + (K - W)p_3/w_3 = 519 + (60 - 45)(10)$

   $= 669$

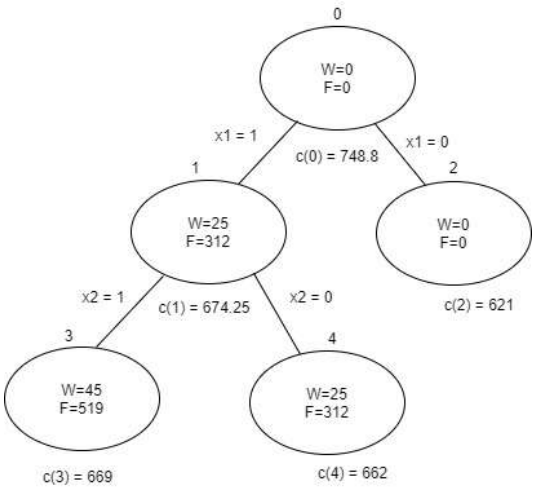
2. Simpul 4 (Olahraga *jumping jack* tidak dipilih)

   W = 25 + 0 = 25

   F = 312 + 0 = 312

   $\hat{c}(4) = F + (K - W)p_2/w_2 = 312 + (60 - 25)(10)$

   $= 662$



Gambar 13. Ekspansi Tahap 3 (Sumber : milik pribadi)

Simpul hidup adalah 2, 3, dan 4. Karena simpul 3 memiliki *cost* paling besar, maka simpul 3 selanjutnya yang akan diekspansi.

Tahap 4

Bangkitkan simpul anak kiri (simpul 5) dan simpul anak kanan (simpul 6) dari simpul 3

1. Simpul 5 (Olahraga *running* dipilih)

   W = 25 + 20 + 30 = 75 > 60 (lebih besar dari kapasitas knapsack yaitu 60)

   Sehingga simpul 5 langsung dimatikan

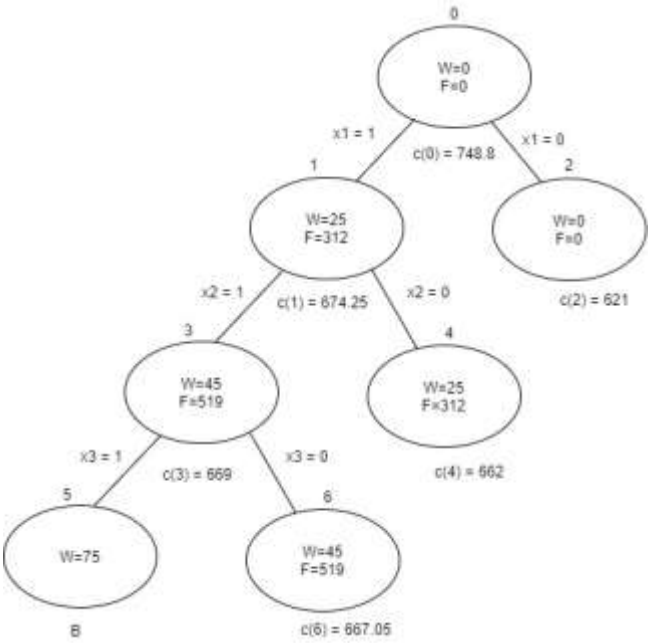
2. Simpul 6 (Olahraga *running* tidak dipilih)

   W = 25 + 20 + 0 = 45

   F = 312 + 207 = 519

   $\hat{c}(6) = F + (K - W)p_4/w_4 = 519 + (60 - 45)(9.87)$

   = 667.05



Gambar 14. Ekspansi Tahap 4 (Sumber : milik pribadi)

Simpul hidup adalah 2, 4, dan 6. Karena simpul 6 memiliki *cost* paling besar, maka simpul 6 selanjutnya yang akan diekspansi.

Tahap 5

Bangkitkan simpul anak kiri (simpul 3) dan simpul anak kanan (simpul 4) dari simpul 1

1. Simpul 7 (Olahraga *push up* dipilih)

   W = 25 + 20 + 15 = 60

   F = 312 + 207 + 148 = 667

   $\hat{c}(7) = F + (K - W)p_5/w_5 = 667 + (60 - 60)(9.85)$

   = 667

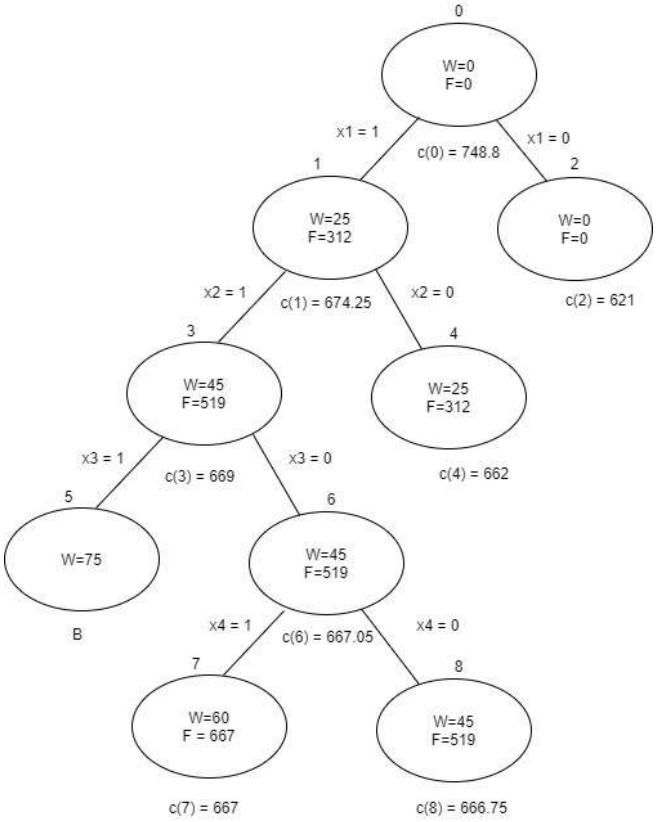
2. Simpul 8 (Olahraga *push up* tidak dipilih)

   W = 25 + 20 + 0 = 45

   F = 312 + 207 = 519

   $\hat{c}(8) = F + (K - W)p_5/w_5 = 519 + (60 - 45)(9.85)$

   = 666.75



Gambar 15. Ekspansi Tahap 5 (Sumber : milik pribadi)

Simpul hidup adalah 2, 4, 7, dan 8. Karena simpul 7 memiliki *cost* paling besar, maka simpul 7 selanjutnya yang mungkin akan diekspansi. Akan tetapi simpul 7 telah mempunyai W yang sama dengan K sehingga algoritma ini berhenti. Karena jika diekspan terus simpul 7 dan anak anaknya, maka x akan selalu 0 yang artinya kegiatan olahraga tidak dipilih dan nilai *cost* juga akan tetap sama yaitu 667.

Sehingga didapatlah solusi optimal X = {1,1,0,1,0,0,0,0}. Oleh karenanya untuk permasalahan *knapsack 0/1 problem* pemilihan kegiatan olahraga yang dapat mengahasilkan kalori optimal adalah dengan melakukan *skipping* atau lompat tali, *jumping jack*, dan *push up*.

IV. PENUTUP

A. *Kesimpulan*

Setelah dilakukan implementasi *knapsack 0/1 problem* yang dalam hal ini adalah pemilihan jenis olahraga yang tepat untuk mendapatkan kalori yang optimal. Dapat disimpulkan bahwa algoritma *Branch & Bound* dapat mennyelesaikan persoalan ini. Algoritma *Branch & Bound* juga dapat memberikan persoalan yang optimal sehingga kalori yang dihasilkan oleh seluruh olahraga yang dilakukan dapat memberikan hasil yang maksimal pada durasi tertentu. Penyelesaian permasalahan ini dapat membantu seseorang yang memiliki waktu terbatas, contohnya pekerja kantoran. Jika dibandingkan dengan pencarian solusi secara manual, penyelesaian dengan algoritma *Branch & Bound* tergolong cepat, karena terdapat *cost* atau nilai untuk menentukan langkah selanjutnya dan juga terdapat fungsi pembatas untuk membatasi setiap langkah selanjutnya. Sebenarnya selain algoritma *Branch & Bound* terdapat banyak algoritma lain yang bisa digunakan untuk menyelesaikan *knapsack 0/1 problem* ini. Seperti algoritma *Brute Force*, algoritma *Greedy*, algoritma *Backtracking*, dan *Dynamic Programming*.

B. *Saran*

Hal yang mungkin bisa dilakukan kedepannya untuk memyelesaikan persoalan ini lebih baik lagi adalah dengan menambah pilihan olahraga yang ingin dilakukan dan lebih memvariasikan waktunya. Sehingga penentuan olahraga yang dilakukan akan lebih tepat. Selain itu, jika pilihan olahraganya hanya sedikit, akan lebih baik jika mencarinya secara manual saja.

PRANALA VIDEO YOUTUBE

Berikut merupakan pranala video yang berisi tentang penjelasan materi makalah ini.

https://youtu.be/Xl5Lp1HrrIU

UCAPAN TERIMA KASIH

Pertama-tama penulis mengucapkan terima kasih yang sebesar-besarnya kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmatnya, penulis bisa menyelesaikan makalah ini sesuai waktu yang telah diberikan. Tidak lupa juga penulis mengucapkan terima kasih kepada kedua orang tua yang selalu mendukung dan mendidik penulis sehingga bisa melanjutkan pendidikan di Institut Teknologi Bandung. Tentu juga penulis mengucapkan terima kasih kepada Ibu Dr. Nur Ulfa Maulidewi, ST., M.Sc yang telah memberikan begitu banyak ilmu kepada penulis selama 1 semester ini. Terakhir penulis juga mengucapkan terima kasih kepada teman-teman yang mendukung saat pembuatan makalah ini. Besar harapan saya agar makalah ini bisa berguna bagi banyak orang.

REFEREMCE

[1] R. Munir, Teori Graf Matematika Diskrit. Bandung: Departemen Teknik Informatika Institut Teknologi Bandung, 2005. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian1.pdf (diakses 8 Mei 2021)

[2] R. Munir, Teori Graf Matematika Diskrit. Bandung: Departemen Teknik Informatika Institut Teknologi Bandung, 2005. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branchand-Bound-2021-Bagian4.pdf (diakses 8 Mei 2021)

[3] https://www.unisbank.ac.id/ojs/index.php/ft1/article/download/1110/654 (diakses 9 Mei 2021)

[4] https://hengkyharwanto.wordpress.com/2015/03/21/latihan-kardio/(diakses 9 Mei 2021)

[5] https://www.alodokter.com/beragam-manfaat-olahraga (diakses 9 Mei 2021)

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.

Denpasar, 11 Mei 2021

Kadek Dwi Bagus Ananta Udayana
13519057