Penerapan Algoritma Runut-Balik dalam Pencarian Solusi Permainan *Calculator: The Game*

Putu Gde Aditya Taguh Widiana / 13517032
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517032@std.stei.itb.ac.id

***Abstrak*—Permainan adalah salah satu bentuk kegiatan yang menyenangkan, yang dilakukan untuk mengusir rasa jenuh bagi pemainnya. Permainan terdiri dari berbagai bentuk. Saat ini, permainan dapat berupa aplikasi pada ponsel pintar. Salah satu contoh aplikasi ini adalah *game puzzle* bernama *Calculator: The Game*. Solusi dari permainan ini dapat ditemukan dengan menggunakan algoritma runut balik.**

Kata Kunci— permainan, puzzle, runut balik, solusi

I. PENDAHULUAN

Manusia pada umumnya memiliki aktifitas yang sangat padat. Aktifitas-aktifitas tersebut terkadang membuat manusia merasa tertekan, bahkan stres. Oleh karena itu, manusia memerluka sesuatu untuk menyalurkan/ menghilangkan stresnya. Inilah alasan singkat munculnya permainan.

Permainan (*Game*) adalah suatu aktifitas yang dilakukan pemainnya untuk bersenang-senang dan mengisi waktu luang. Permainan terdiri dari berbagai jenis, salah satunya adalah permainan *puzzle*. *Puzzle* adalah jenis permainan yang menguji kemampuan berpikir pemainnya. Puzzle biasanya dibuat untuk menjadi hiburan, tetapi bisa juga dibuat dari permasalahan matematis ataupun logika.

Permainan masa kini terdiri dari berbagai bentuk, salah satunya yang paling baru adalah permainan berupa aplikasi pada ponsel pintar. Beragam jenis aplikasi permainan dapat diunduh dah dimainkan pada ponsel pintar, termasuk permainan berjenis *puzzle*. Salah satu permainan *puzzle* yang sudah banyak diunduh dan dimainkan adalah aplikasi *Calculator: The Game.*

Aplikasi ini adalah aplikasi permainan berjenis *puzzle* matematis. Pemain diberikan suatu angka, dan beberapa operasi matematis yang dapat diterapkan pada angka tersebut. Pemain diminta untuk menggunakan operasi-operasi matematis tersebut sehingga angka yang didapat akan sama dengan angka tujuan yang diminta (*goal)*. Pemain harus dapat menemukan solusi dari permainan tersebut dalam jumlah penggunaan operasi yang diiperbolehkan oleh aplikasi (*moves*). Jika pemain gagal menemukan solusinya, pemain dapat melakukan reset pada level tersebut, hingga pemain menemukan solusinya.

Permainan ini memiliki sangat banyak tingkatan (*level*). Semakin tinggi levelnya, maka semakin sulit teka-teki yang diberikan. Pada level-level tertentu, pemain akan diberikan lebih banyak pilihan operasi matematis, sehingga puzzle ini menjadi lebih sulit. Hal ini akan membuat pemain menjadi semakin tertantang untuk menyelesaikannya.

Gambar 1. Contoh tampilan dari permainan *Calculator:The Game*
Sumber : https://play.google.com/store/apps/details?id=-com.sm.calculateme&hl=en
Waktu akses : 25 April 2019 13.14 WIB

Jumlah kemungkinan solusi dari permainan ini terbatas, sehingga solusinya dapat dicari dengan mencoba berbagai kemungkinan kombinasi dari operasi matematis, hingga solusi ditemukan. Pencarian solusi ini dapat dibuat dengan algoritma runut-balik (*backtracking*).

II. LANDASAN TEORI

Algoritma runut-balik atau biasa disebut dengan algoritma *backtracking*, adalah suatu algoritma pencarian yang diterapkan pada struktur graf. Algoritma ini pertama kali diperkenalkan oleh D. H. Lehmer pada tahun 1950. Beberapa tahun kemudian, penyajian secara umum dari algoritma ini dipaparkan oleh R.J Walker, Golomb, dan Baumert. Algoritma ini berbasis pada algoritma DFS, yang melakukan pencarian secara lebih mangkus. Algoritma ini biasanya digunakan untuk menemukan solusi yang solusinya dibagun secara bertahap melalui proses yang berulang, hingga suatu kondisi terpenuhi.

Algoritma ini merupakan perbaikan dari algoritma *brute force* dan *exhaustive search*. Pada algoritma *exhausive search*, semua kemungkinan solusi dari ruang solusi ditelusuri secara satu per satu, sedangkan pada algoritma *backtracking*, pilihan tidak lagi mengarah ke kemungkinan solusi yang tidak memungkinkan untuk menjadi solusi, sehingga menjadi lebih efektif karena memangkas simpul-simpul yang tidak mengarah ke solusi. Hal ini biasanya disebut dengan *pruning*.

Saat ini algoritma backtracking banyak diterapkan untuk program games (seperti permainan tic-tac-toe, menemukan jalan keluar dalam sebuah labirin, catur, dll) dan masalah-masalah pada bidang kecerdasan buatan (artificial intelligence).

1. Properti Umum Metode Runut-Balik

   Untuk menerapkan algotitma backtracking, properti-properti berikut perlu untuk didefinisikan :

   a. Solusi Persoalan

      Solusi dinyatakan sebagai vektor dengan n-tuple: $X = (x_1, x_2, \ldots, x_n)$, x_i himpunan berhingga S*i*.

      Mungkin saja $S_1 = S_2 = \ldots = S_n$.

      Contoh: $S_i = \{0, 1\}$, $x_i = 0$ atau 1.

   b. Fungsi Pembangkit

      Dinyatakan sebagai:

      T(k).

      T(k) membangkitkan nilai untuk x_k, yang merupakan komponen vektor solusi.

   c. Fungsi Pembatas

      Dinyatakan sebagai: $B(x_1, x_2, \ldots, x_k)$.

      Fungsi pembatas menentukan apakah $(x_1, x_2, \ldots, x_k)$ mengarah ke solusi. Jika ya, maka pembangkitan nilai untuk x_k+1 dilanjutkan, tetapi jika tidak, maka $(x_1, x_2, \ldots, x_k)$ dibuang dan tidak dipertimbangkan lagi dalam pencarian solusi.

      Fungsi pembatas tidak selalu dinyatakan sebagai fungsi matematis, ia dapat dinyatakan sebagai predikat yang bernilai true atau false, atau dalam bentuk lain yang ekivalen.

2. Pengorganisasian Solusi

   Pada algoritma runut-balik, semua kemungkinan solusi dari persoalan yang sedang ingin diselesaikan disebut degan **ruang solusi** (*solution space*).

   Jika $x_i \in S_i$, maka $S1 \times S2 \times \ldots \times Sn$ disebut ruang solusi. Jumlah anggota di dalam ruang solusi adalah $|S_1| \times |S_2| \times \ldots \times |S_n|$. Tinjau persoalan Knapsack 0/1 untuk n = 3. Solusi persoalan dinyatakan sebagai vektor (x_1, x_2, x_3) dengan $x_i \in \{0,1\}$. Ruang solusinya adalah

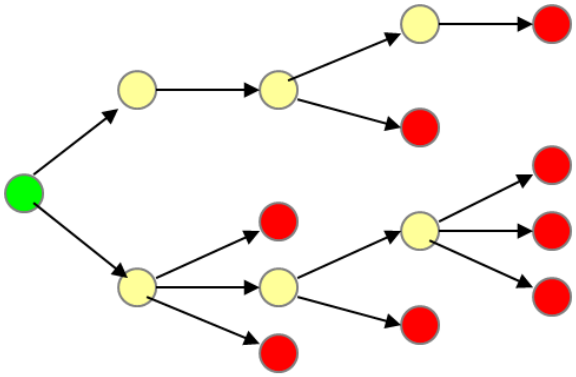
   $\{0,1\} \times \{0,1\} \times \{0,1\} = \{(0, 0, 0), (0, 1, 0), (0, 0, 1), (1, 0, 0), (1, 1, 0), (1, 0, 1), (0, 1, 1), (1, 1, 1)\}$.

   Pada persoalan Knapsack 0/1 dengan n = 3 terdapat $2^n = 2^3 = 8$ kemungkinan solusi, yaitu:

   (0, 0, 0), (0, 1, 0), (0, 0, 1), (1, 0, 0), (1, 1, 0), (1, 0, 1), (0, 1, 1), dan (1, 1, 1).

   Pada algoritma runut-balik ini, ruang solusi diorganisasikan ke dalam struktur pohon (*tree*). Tiap-tiap simpul pohon menyatakan suatu status (*state*) persoalan, sedangkan sisi (cabang) dari pohon ruang solusi tersebut dilabeli dengan nilai-nilai x_i. Lintasan dari akar ke daun pada pohon ruang status menyatakan solusi yang mungkin. Oleh karena itu, seluruh lintasan dari akar ke daun membentuk ruang solusi. Pengorganisasian pohon ruang solusi diacu sebagai **pohon ruang status** (*state space tree*).

   Berikut adalah ilustrasi dari sebuah pohon ruang status :

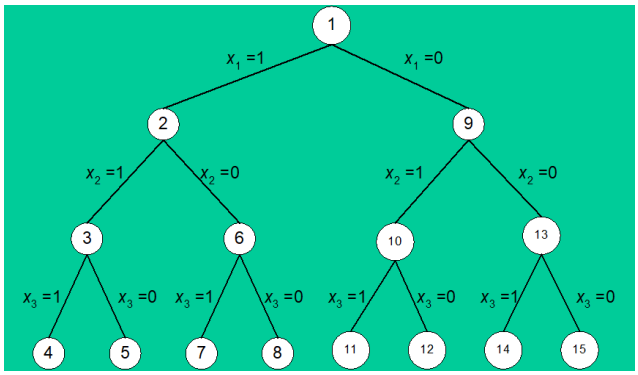


Gambar 2. Ilustrasi dari Sebuah Pohon Ruang Status
Sumber : Slide Perkuliahan IF2211 – Stategi Algoritma

Keterangan dari gambar di atas adalah sebagai berikut :

- Simpul akar, awal pembentukkan solusi
- Simpul dalam, solusi akan dibentuk melalui simpul ini
- Simpul daun, pencarian tidak dilanjutkan ke simpul yang lebih dalam jika tidak memenuhi fungsi pembatas yang sudah ditentukan sebelumnya

Tinjau kembali persoalan Knapsack 1/0 untuk n = 3. Pohon ruang status untuk permasalahan tersebut dapat dilihat pada **Gambar 3** di bawah ini.



Gambar 3. Pohon ruang status untuk persoalan Knapsack 0/1 dengan n = 3
Sumber : Slide Perkuliahan IF2211 – Stategi Algoritma

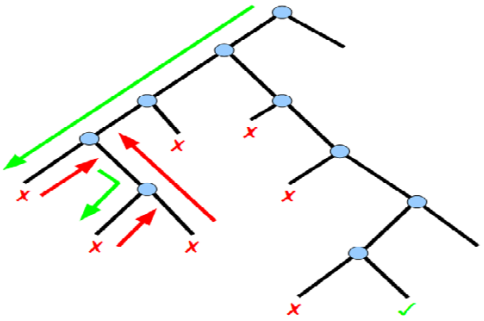
3. Prinsip Pencarian Solusi dengan Algoritma Runut-Balik

   Pada algoritma runut-balik, solusi dicari dengan membentuk lintasan dari akar ke daun. Aturan pembentukan yang dipakai adalah mengikuti metode pencarian mendalam (DFS). Simpul-simpul yang sudah dilahirkan dinamakan **simpul hidup** (*live*

node), sedangkan simpul hidup yang sedang diperluas dinamakan **simpul-E** (*Expand-node*). Simpul-simpul tersebut dapat diberi nomor sesuai dengan urutan kelahirannya sesuai dengan prinsip DFS.

Tiap kali simpul-E diperluas, lintasan yang dibangun olehnya juga bertambah panjang. Jika lintasan yang sedang dibentuk tidak mengarah ke solusi, maka simpul-E tersebut "dibunuh" sehingga menjadi **simpul mati** (*dead node*). Fungsi yang digunakan untuk membunuh simpul-E adalah dengan menerapkan **fungsi pembatas** (*bounding function*). Jika suatu simpul tidak memenuhi hasil yang diinginkan oleh fungsi pembatas, maka fungsi tersebut akan dibunuh. Simpul yang sudah mati tidak akan pernah diperluas lagi.

Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian *backtrack* ke simpul aras diatasnya. Lalu, teruskan dengan membangkitkan simpul anak yang lainnya. Selanjutnya simpul ini menjadi simpul-E yang baru. Pencarian dihentikan bila kita telah sampai pada node dengan keadaan yang sesuai dengan apa yang dicari (*goal node*).



Gambar 4. Ilustrasi pencarian solusi dengan menggunakan algoritma runut-balik
Sumber : Slide Perkuliahan IF2211 – Stategi Algoritma

Tinjau persoalan *Knapsack* 0/1 dengan instansiasi:

$n = 3$

$(w_1, w_2, w_3) = (35, 32, 25)$

$(p_1, p_2, p_3) = (40, 25, 50)$

$M = 30$

Dengan keterangan :

- n = banyaknya barang
- w = berat dari setiap barang
- p = keuntungan dari setiap barang
- M = beban maksimal yang dapat dibawa

Solusi dinyatakan sebagai $X = (x_1, x_2, x_3)$, $x_i \in \{0, 1\}$.

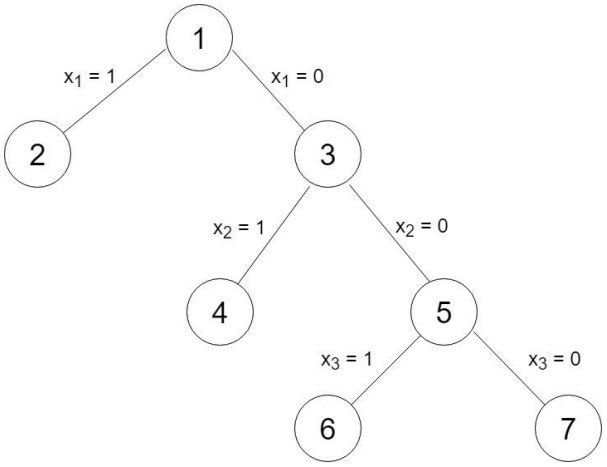
Fungsi konstrain (*bounding function*) yang digunakan adalah sebagai berikut :

$$\sum_{i=1}^{k} w_i x_i \leq M$$

Fungsi itu menyatakan bahwa untuk memperluas suatu simpul dan membangkitkan anak-anaknya, vektor solusi hingga x_i total beratnya harus lebih rendah dari berat maksimal yang dapat dibawa (M).

Algoritma penyelesaian :

Simpul yang pertama kali dibangkitikan adalah simpul akar, yang berarti belum ada barang yang dipilih, dinamakan simpul 1. Simpul 1 kemudian diperluas menjadi simpul 2 dan simpul 3. Simpul aktif sekarang berpindah dari simpul 1 mejadi simpul 2. Simpul 2 berisi bahwa barang pertama dibawa. Simpul ini tidak memenuhi fungsi pembatas, sehingga dibunuh ($w_1 \geq M$). Proses akan kembali ke simpul 1, lalu memperluas ke simpul 3. Simpul 3 berisi bahwa barang pertama tidak dibawa, dan simpul ini memenuhi fungsi pembatas ($0 \leq M$), sehingga simpul diperluas menjadi simpul 4 dan simpul 5. Simpul aktif berpindah ke simpul 4. Simpul 4 berisi bahwa barang kedua dibawa. Simpul ini tidak memenuhi fungsi pembatas, sehingga dibunuh ($w_4 \geq M$). Proses akan kembali ke simpul 3, dan diperluas ke simpul 5. Simpul aktif sekarang adalah simpul 5. Simpul 5 berisi bahwa barang kedua tidak dibawa, sehingga simpul ini memenuhi fungsi pembatas ($0 \leq M$). Simpul ini kemudian diperluas menjadi simpul 6 dan simpul 7. Simpul 6 berisi bahwa barang ketiga dibawa, dan simpul ini memenuhi fungsi pembatas ($w_3 \leq M$). Ini dapat menjadi salah satu solusi. Karena simpul 6 tidak dapat diperluas kembali, maka simpul aktif berpindah ke simpul 7, yang berisi bahwa barang ketiga tidak dibawa. Simpul ini memenuhi fungsi pembatas ($0 \leq M$), tetapi hasil keuntungannya (F) lebih kecil daripada simpul 6, sehingga yang dipilih jadi jawaban akhir adalah simpul 1, 3, 6. Solusi optimumnya adalah $X = (0, 0, 1)$ dan $F = 50$.



Gambar 5. Ilustrasi pencarian solusi dengan menggunakan algoritma runut-balik pada permasalahan knapsack dengan n = 3

4. Skema Umum Algoritma Runut-Balik (versi Rekursif)

Sebenarnya, skema umum algoritma runut-balik dapat dilakukan dengan melalui rekursif dan iterasi, tetapi makalah ini hanya membahas melalui rekursif. Berikut adalah algoritmanya dalam pseudocode.

```
procedure RunutBalikR(input k:integer)
{Mencari semua solusi persoalan dengan
metode runut-balik; skema rekursif
 Masukan: k, yaitu indeks komponen vektor
solusi, x[k]
 Keluaran: solusi x = (x[1], x[2], …,
x[n])}
Algoritma:
   for tiap x[k] yang belum dicoba
sedemikian sehingga
        ( x[k]←T(k)) and B(x[1], x[2],
... ,x[k])= true do
      if (x[1], x[2], ... ,x[k]) adalah
lintasan dari akar ke daun
      then
        CetakSolusi(x)
      endif
      RunutBalikR(k+1)     { tentukan
nilai untuk x[k+1]}
   endfor
```

III. *CALCULATOR: THE GAME*

Calculator: The Game adalah permainan *puzzle* matematis pada platform mobile yang dirilis oleh *Simple Machine*. Pemain diberikan suatu angka, dan beberapa operasi matematis yang dapat diterapkan pada angka tersebut. Pemain diminta untuk menggunakan operasi-operasi matematis tersebut sehingga angka yang didapat akan sama dengan angka tujuan yang diminta (*goal)*. Pemain harus dapat menemukan solusi dari permainan tersebut dalam jumlah penggunaan operasi yang diiperbolehkan oleh aplikasi (*moves*). Jika pemain gagal menemukan solusinya, pemain dapat melakukan reset pada level tersebut, hingga pemain menemukan solusinya.

Permainan ini memiliki sangat banyak tingkatan (*level*). Semakin tinggi levelnya, maka semakin sulit teka-teki yang diberikan. Pada level-level tertentu, pemain akan diberikan lebih banyak pilihan operasi matematis, sehingga puzzle ini menjadi lebih sulit. Hal ini akan membuat pemain menjadi semakin tertantang untuk menyelesaikannya.

Berikut adalah beberapa operasi matematis yang ada pada permainan *Calculator: The Game* :

- `+x` : Merupakan penambahan biasa. Angka awal akan dijumlahkan dengan x. Contoh : Misalkan angka awal adalah `3`, dan operasi yang dipilih adalah `+4`, maka hasilnya adalah `7`.
- `-x` : Merupakan pengurangan biasa. Angka awal akan dikurangkan dengan x. Contoh : Misalkan angka awal adalah `3`, dan operasi yang dipilih adalah `-4`, maka hasilnya adalah `-1`.
- `*x` : Merupakan perkalian biasa. Angka awal akan dikalikan dengan x. Contoh : Misalkan angka awal adalah `3`, dan operasi yang dipilih adalah `*4`, maka hasilnya adalah `12`.
- `/x` : Merupakan pembagian biasa. Angka awal akan dibagi dengan x. Contoh : Misalkan angka awal adalah `12`, dan operasi yang dipilih adalah `/4`, maka hasilnya adalah `3`.
- `a > b` : Mengubah karakter `a` menjadi `b`. Contoh : Misalkan angka awal adalah `153`, operasi yang dipilih adalah `5 > 7`, hasilnya adalah `173`.
- `x` : Menambahkan karakter terakhir pada input. Contoh : Misalkan angka awal adalah `56`, dan operasi yang dipilih adalah `8`, maka hasilnya adalah `568`.
- `^x` : Memangkatkan dengan x. Angka awal akan dipangkatkan dengan x. Contoh : Misalkan angka awal adalah `3`, dan operasi yang dipilih adalah `^2`, maka hasilnya adalah `9`.
- `+/-` : Mengganti tanda (sign). Angka awal akan diubah tandannya (nilai positif negatifnya). Contoh : Misalkan angka awal adalah `12`, dan operasi yang dipilih adalah `+/-`, maka hasilnya adalah `-12`.
- `<<` : Menghapus angka terakhir dari input. Contoh : Misalkan angka awal adalah `1312`, dan operasi yang dipilih adalah `<<` , maka hasilnya adalah `131`.
- `Reverse` : Menghapus angka terakhir dari input. Contoh : Misalkan angka awal adalah `1312`, dan operasi yang dipilih adalah `<` , maka hasilnya adalah `131`.

Gambar 6. Salah satu level lanjutan dari permainan *Calculator: The Game*
Sumber : https://play.google.com/store/apps/details?-id=com.sm.calculateme&hl=en

IV. PENERAPAN ALGORITMA RUNUT-BALIK DALAM PERMAINAN *CALCULATOR: THE GAME*

Dalam permainan *Calculator: The Game*, pemain diberikan beberapa operasi matematis yang dapat dipilih supaya dapat mencapai angka tujuan dari angka awal. Karena jumlah operasi yang dapat digunakan terbatas, dan *moves* yang diijinkan juga terbatas, maka pencarian solusi dapat dilakukan dengan menggunakan algoritma runut-balik.

Algoritma runut balik yang digunakan adalah algoritma runut-balik dalam bentuk rekursif. Bahasa pemrograman yang digunakan adalah bahasa python.

Properti algoritma runut balik yang didefinisikan adalah :

- Solusi Persoalan : Solusi dari persoalan ini adalah ketika hasil yang didapatkan sesuai dengan hasil yang diminta.
- Fungsi Pembangkit : Setiap operasi matematis yang didaftarkan pada suatu level memiliki indeksnya sendiri. Untuk setiap operasi matematis yang digunakan, indeksnya akan didaftarkan pada vektor solusi.
- Fungsi Pembatas : Jika operasi yang digunakan adalah reverse ataupun insert (menambahkan angka dari belakang) dan angka inputnya adalah desimal (bukan bilangan bulat), maka rekursif tidak akan dilanjutkan.

Berikut ini adalah kode program dalam bentuk bahasa python :

```
def solve(start,result,depth,arrTypeInput,arrJawaba
n):

  return solveRecursive(float(start),float(result),
int(depth),0,arrTypeInput,arrJawaban)
```

Fungsi solve akan memanggil fungsi solveRecursive, yang didefinisikan sebagai berikut :

```
def solveRecursive(start,result,depth,curDepth,arrT
peInput,arrJawaban):
            found = False
            if (depth>curDepth):
                curDepth+=1
                i = 0

 while(i<len(arrTypeInput) and not(found)):
                    error = False
                    hasil = 0

 if (arrTypeInput[i].getCmd() == "i" or arrTypeInpu
[i].getCmd() == "r"):
                        temp = start
                        while(temp>=1):
                            temp-=1
                        if (temp!=0):
                            error = True
                        else:

asil = arrTypeInput[i].calc(float(start))
                    else:
asil = arrTypeInput[i].calc(float(start))
                        arrJawaban[curDepth-1] = i

 if ((hasil != result) and not(error)):

ound = solveRecursive(hasil,result,depth,curDepth,a
rTypeInput,arrJawaban)
                    elif (hasil == result):
                        found = True
                    i+=1
            return found
```

Program juga akan meminta parameter start, goal, dan move dari pengguna. Masing-masing parameter itu berarti :

- Start : merupakan angka awal
- Goal : merupakan angka akhir yang hendak dituju
- Move : jumlah langkah yang diperbolehkan

Berikut adalah contoh penggunaan program :

Level 72

Isi input.txt

```
+ 6
- 3
r
<
```

Isi dari input.txt menyatakan bahwa operasi yang didaftarkan untuk digunakan adalah `+ 6`, `- 3`, `reverse`, dan `<<`. Operasi-operasi inilah yang akan digunakan pemain untuk menemukan solusi (28). Berikut adalah hasil program solver yang sudah dijalankan :

Hasil Program yang Dijalankan

```
Masukan Angka Awal : 0
Masukan Angka Tujuan : 28
Masukan Jumlah Move : 7

Solusi :
+ 6
+ 6
<
+ 6
+ 6
r
- 3
```

Dari hasil program yang dijalankan, dengan 7 operasi, didapatkan hasil 28 dengan langkah langkah sebagai berikut :

- `0 + 6 → 6`
- `6 + 6 → 12`
- `12 < → 1`
- `1 + 6 → 7`
- `7 + 6 → 13`
- `13 reverse → 31`
- `31 – 3 → 28`

Dengan ini, langkah-langkah yang ditentukan oleh program sudah terbukti kebenarannya.

V. KESIMPULAN

Algoritma runut-balik atau *backtracking* merupakan algoritma yang sangat berguna untuk menyelesaikan permasalahan, baik permasalahan yang kompleks maupun sekedar permasalahan sederhana seperti melakukan pencarian solusi terhadap permainan Calculator: The Game ini. Pada pencarian solusi dari permainan ini, algoritma runut-balik mampu menghasilkan solusi penyelesaian dari suatu level, sehingga pemain yang kesulitan pada suatu level dapat melanjutkan permainannya ke level selanjutnya.

VI. PENUTUP

Saya mengucapkan terima kasih kepada semua pihak yang secara langsung maupun tidak langsung telah membantu kelancara pembuatan makalah ini. Saya juga berharap bahwa makalah ini dapat berguna bagi pembaca. Mohon maaf jika ada informasi yang dirasa kurang tepat.

VII. REFERENSI

[1]. Donald E. Knuth (1968). The Art of Computer Programming. Addison Wesley.

[2]. Gurari, Eitan (1999). "CIS 680: DATA STRUCTURES: Chapter 19: Backtracking Algorithms". Archived from the original on 17 March 2007.

[3]. Rossi, Francesca; Beek, Peter Van; Walsh, Toby (August 2006). "Constraint Satisfaction: An Emerging Paradigm". Handbook of Constraint Programming. Foundations of Artificial Intelligence. Amsterdam: Elsevier. p. 14. ISBN 978-0-444-52726-4. Retrieved 2008 12-30. Bitner and Reingold credit Lehmer with first using the term 'backtrack' in the 1950s.

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, 26 April 2019

Putu Gde Aditya Taguh Widiana
13517032