

Penerapan Algoritma *Exhaustive Search* dalam Penyelesaian Persoalan Pola Angka di Bangun Datar pada Soal Ujian Tulis Berbasis Komputer

Raka Wirabuana Ninagan - 13520134
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13520134 @std.stei.itb.ac.id

Abstract—UTBK merupakan tes masuk perguruan tinggi yang diselenggarakan oleh LTMPT. Materi yang diujikan pada UTBK tentu bermacam-macam, yang salah satunya adalah menebak pola urutan beberapa angka yang disusun pada suatu bangun datar tertentu kemudian pola tersebut diterapkan ke gambar yang menjadi pertanyaannya (biasanya masuk ke dalam subtopik figural). Soal yang jenisnya seperti ini memiliki tingkat kesulitan yang sangat beragam, bahkan ada kemungkinan ketika satu persoalan memiliki pola penyelesaian yang sangat banyak. Pada umumnya, pelajar mempelajari pola-pola tertentu yang biasanya keluar sehingga tidak menghabiskan waktu untuk melakukan pengecekan satu-satu saat ujian. Makalah ini akan membahas mengenai penerapan *exhaustive search* melalui tenaga program computer sehingga bisa diselesaikan dengan cepat dan akurat.

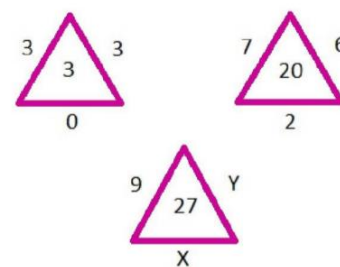
Keywords—*figural; exhaustive search; UTBK*

I. PENDAHULUAN

UTBK merupakan tes masuk perguruan tinggi di Indonesia yang diselenggarakan oleh LTMPT (Lembaga Tes Masuk Perguruan Tinggi). Sebagai gerbang yang menguji kelayakan calon mahasiswa untuk masuk ke kampus impiannya, UTBK mengujikan berbagai jenis materi yang sudah diajarkan selama pendidikan dasar hingga sekolah menengah (SD - SMA), seperti Matematika, Fisika, Biologi, Kimia, Bahasa Inggris, dan lain-lain.

Namun, ada beberapa materi yang tidak diajarkan secara eksplisit selama 12 tahun, tetapi diharapkan agar bisa dikuasai oleh peserta sebagai salah satu tolak ukur kesiapan menjalani kegiatan akademik di tingkat perguruan tinggi. Materi-materi tersebut diantaranya adalah penarikan kesimpulan (terimplementasikan di berbagai mata pelajaran, tetapi lebih condong ke mata pelajaran Bahasa Indonesia), penentuan mengenai pola dari sebuah gambar ataupun angka (condong ke mata pelajaran Matematika), dan yang lainnya. Salah satu persoalan yang diujikan di UTBK adalah persoalan mencari pola yang terbentuk oleh angka-angka pada bangun datar.

18. Perhatikan gambar di bawah ini!



Nilai X dan Y secara berturut-turut adalah

- A. 3 dan 1
- B. 1 dan 3
- C. 0 dan 3
- D. 3 dan 0
- E. 2 dan 7

Gambar 1.1 Contoh soal pola angka pada bangun datar

sumber : <https://www.marikuliah.com/2022/01/soal-asli-utbk-2021-dan-kunci-jawabannya.html>

Persoalan jenis ini bisa diselesaikan dengan berbagai cara, misalnya melakukan pemeriksaan pola aritmatika satu per satu, mengira-ngira apakah sebuah bilangan bisa didapat dari angka sekitarnya, bahkan mungkin saja melakukan pemeriksaan kecocokan karakter antara gambar-gambar yang ada sehingga bisa menemukan pola yang menjadi solusi dari bilangan yang ditanyakan (berdasarkan Gambar 1.2, ada kemungkinan bahwa gambar segitiga kiri atas dengan segitiga kanan atas tidak memiliki pola independen, tetapi saling memberikan kaitan).

Sudah disebutkan sebelumnya bahwa salah satu cara untuk menemukan pola dari soal-soal seperti ini adalah dengan memeriksa pola aritmatika satu persatu. Namun, tentu ini sangatlah bermasalah apabila diterapkan tanpa menggunakan heuristik apapun karena bisa saja peserta tidak beruntung dalam melakukan pengecekan, yaitu terlalu jauh untuk mencapai pengecekan menggunakan pola yang benar.

Salah satu solusi yang bisa menyelesaikan masalah tersebut adalah dengan membuat program yang menerapkan *brute force* dalam pencarian pola yang tepat sehingga peserta tidak perlu kerepotan melakukan pengecekan satu-persatu yang kemungkinan besar akan menghabiskan terlalu banyak waktu. Namun, program ini tentu saja hanya bisa digunakan sebagai

guide untuk pembelajaran karena saat ujian hampir tidak mungkin digunakan (dilarang menggunakan alat bantu). Setidaknya, program ini bisa dijadikan alat bantu apabila soal-soal latihan yang dikerjakan memiliki kunci jawaban tetapi tidak ada pembahasannya.

II. TEORI DASAR

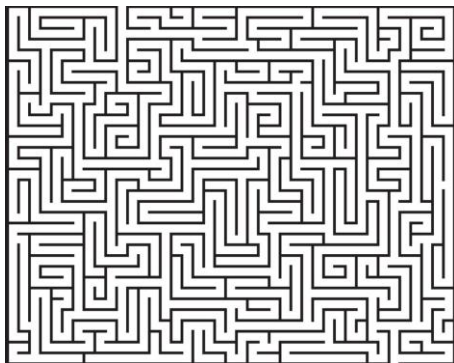
A. Algoritma Brute Force

1. Brute Force

Algoritma Brute Force merupakan algoritma yang menggunakan pendekatan secara lempang, yang artinya secara langsung, caranya jelas, dan sangat sederhana. Misalkan ada sebuah persoalan mencari bilangan terbesar dari larik sebuah bilangan, maka tinggal kita bandingkan saja satu persatu setiap bilangan hingga didapatkan bilangan dengan nilai terbesar. Contoh lainnya misalkan mengurutkan larik bilangan menjadi terurut mengecil, maka elemen larik dibandingkan satu persatu dan digeser ke sebelah kiri untuk bilangan yang lebih besar. Langkah tersebut diulangi terus menerus sehingga tidak ada lagi bilangan yang lebih besar berada di sebelah kanan bilangan lainnya. Menentukan cara menyelesaikannya hampir semudah mengatakannya.

Hal yang menjadi kelebihan utama dari algoritma brute force adalah jawaban yang didapatkan akan selalu optimal. Alasannya tentu karena brute force tidak terlalu mengandalkan asumsi maupun teknik khusus. Oleh karena itu, brute force sering disebut juga sebagai algoritma naif.

Kekurangan terbesar algoritma brute force adalah prosesnya lambat. Misalkan ada sebuah *maze game* yang harus diselesaikan. Apabila diterapkan brute force, setiap jalan buntu harus dilakukan lagi penjelajahan rute baru dari awal (ada cara mempercepat proses ini dengan cara menerapkan algoritma backtracking yang tidak dibahas di makalah ini). Makin banyak percabangan dan makin dalam rute, makin banyak juga rute yang harus diperiksa, sehingga memakan waktu terlalu lama hingga bisa saja terjadi *exhausted* (baik manusia maupun program komputer).



Gambar 2.1 Contoh ilustrasi dari sebuah maze

sumber : <https://www.vectorstock.com/royalty-free-vector/big-maze-labyrinth-vector-31897262>

Meskipun memiliki kekurangan yang cukup “tidak efisien”, brute force tetap sering digunakan karena

penerapannya yang sederhana. Brute force akan memiliki performa yang lebih baik apabila ukuran masukannya kecil, sehingga keuntungannya selain sederhana, efisien pula.

Ada permasalahan-permasalahan yang hanya ditemukan solusinya oleh brute force, contohnya adalah mencari elemen terbesar pada sebuah larik bilangan (berkaitan dengan pernyataan bahwa hampir semua persoalan bisa diselesaikan oleh brute force).

2. Exhaustive Search

Exhaustive Search merupakan teknik pencarian solusi yang menerapkan algoritma brute force untuk menyelesaikan permasalahan-permasalahan objek kombinatorik seperti permutasi, kombinasi, atau himpunan bagian dari sebuah himpunan.

Exhaustive search memiliki 3 tahap dalam penerapannya, yaitu

1. Enumerasi

Persoalan-persoalan objek kombinatorik memungkinkan munculnya banyak solusi yang memenuhi persoalan, sehingga dilakukan enumerasi (list) kemungkinan solusi.

2. Evaluasi

Setiap solusi hasil enumerasi dievaluasi yang kemudian solusi terbaik sementara disimpan.

3. Pemilihan solusi

Setelah pencarian berakhir, solusi yang disimpan menjadi solusi terbaik sehingga bisa disimpulkan sebagai jawaban / solusi final.

Sama seperti brute force, exhaustive search juga secara teoritis akan menghasilkan solusi optimal, tetapi permasalahannya adalah setiap solusi yang memungkinkan harus disimpan terlebih dahulu (tahap enumerasi) sehingga membutuhkan alokasi memori yang besar.

3. Teknik Heuristik

Besarnya volume komputasi dan alokasi memori yang dibutuhkan membuat exhaustive search cukup mahal untuk digunakan dan waktu yang terbuang cukup banyak (sesuai namanya, melelahkan). Oleh karena itu, ada teknik khusus yang dapat memperbaiki kinerja algoritma sehingga tidak semua solusi harus dieksplorasi, yaitu Teknik Heuristik.

Teknik Heuristik adalah teknik yang memberikan efek khusus kepada algoritma untuk tidak mengeksplorasi semua kemungkinan dengan cara pemasangan batasan seperti penilaian intuitif, terkaan, ataupun akal sehat. Teknik ini memang tidak teruji secara matematis, tetapi apabila terkaan yang digunakan tepat, bisa sangat memperbaiki kinerja algoritma. Penentuan teknik heuristik yang diterapkan pada algoritma biasanya berdasarkan pengalaman dari

proses pembelajaran, misalnya pengambilan rute jalan selalu melalui jalan yang paling lebar demi keamanan. Dasar dari penggunaan heuristik tersebut adalah biasanya rute-rute yang tidak aman berasal dari rute jalan yang sempit.



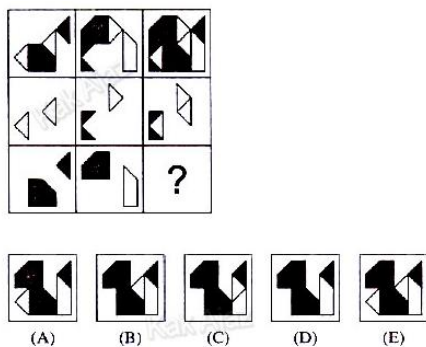
Gambar 2.2 Dua contoh status pada 8-puzzle

sumber : [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-\(2022\)-Bag2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag2.pdf)

Contoh lain dari penggunaan teknik heuristik adalah menjadikan jumlah ubin yang tidak berada pada posisi seharusnya sebagai minimum langkah untuk mencapai status akhir yang valid. Dengan heuristik ini, tentu program tidak perlu repot-repot untuk mengenumerasi langkah-langkah yang jumlahnya dibawah nilai minimum hasil dari teknik heuristik tersebut.

B. Persoalan Figural pada Soal UTBK

Persoalan figural yang muncul di Ujian Tulis Berbasis Komputer umumnya berfokus pada mencari pola aritmatika dari angka-angka yang tersusun pada bangun datar. Pada zaman ujian ini masih bernama SBMPTN (sekarang ujiannya UTBK dan proses seleksi masuk perguruan tinggi memiliki beberapa jalur termasuk UTBK, diberi nama SBMPTN), persoalan figural lebih condong ke arah memilih pola mana yang cocok untuk mengisi urutan dari pola-pola yang ditampilkan.



Gambar 2.3 Persoalan figural saat tes tertulis masuk perguruan tinggi masih bernama SBMPTN

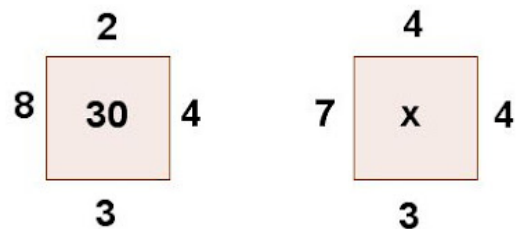
sumber : <https://kakajaz.blogspot.com/2018/02/pembahasan-figural-41-45-tkpa-sbmptn-2017.html>

Persoalan UTBK figural yang sekarang memiliki kecenderungan memiliki pola berupa urutan operasi aritmatik, misalnya terdapat kumpulan angka $[1, 2, 4, 6]$ dan $[1, x, 8, 12]$, maka x bisa didapatkan berdasarkan pola yang diambil dari kumpulan angka pertama, yaitu $6 * 1 - 4$ (ini hanya salah satu solusi) sehingga $x = 12 * 1 - 8 = 4$. Tipe soal di atas merupakan tipe soal yang muncul di UTBK 2020. UTBK 2021 sedikit

berbeda, karena nilai yang ditanyakan tidak 1 variabel, tetapi 2 variabel (contohnya ada pada Gambar 1.1).

Makalah ini akan berfokus pada tipe yang menanyakan 1 variabel. Biasanya jumlah angka yang dilibatkan sebagai pola berjumlah 3 sampai 6. Memang tidak menutup kemungkinan lebih besar dari 6, tetapi jarang sekali keluar (dasar ini berasal dari penulis yang ikut berbagai *try out* UTBK 2020 dari berbagai lembaga *try out*).

1. Perhatikan Gambar dan Tentukan Nilai x !



Gambar 2.4 Soal figural tipe 1 variabel

sumber : <https://www.aiairyn.com/2020/04/soal-try-out-tps-kemampuan-penalaran.html>

Meskipun soal tipe ini tidak bisa dipastikan kemunculannya, tetapi pada dasarnya tipe ini sangat memungkinkan untuk muncul lagi karena sudah disebutkan sejak awal bahwa soal ini bisa melahirkan ratusan tipe soal, hanya prinsipnya yang akan selalu sama.

III. IMPLEMENTASI

A. Pemetaan masalah

Dalam mencari nilai yang ditanyakan oleh soal, langkah pertama adalah menentukan dulu pola dari contoh yang kemudian dicoba ke persoalan. Ada dua kemungkinan, yang pertama adalah hanya ada satu contoh, yang kedua adalah lebih dari satu contoh, bisa dua bahkan lebih. Fungsi dari banyaknya contoh sebenarnya hanya untuk memberikan kepastian pola yang dipakai (biasanya termasuk yang cukup rumit). Artinya, ada dua hal yang perlu diproses, yaitu kumpulan bilangan pertama sebagai contoh dan kumpulan bilangan kedua dan variabel yang ditanyakannya. Alasan dipilihnya kemungkinan yang satu contoh akan dikemukakan di bagian implementasi pada program.

Permasalahan kedua yang perlu diselesaikan adalah menentukan bilangan pada contoh untuk dijadikan target. Pada persoalan tentu sudah jelas target yang ditanyakan adalah yang berbentuk variabel atau simbol tanda tanya (simbol pada Gambar 2.4 adalah x). Contoh dimunculkan sebagai acuan atau panduan untuk mengerjakan persoalan, sehingga akan lebih mudah apabila menganggap posisi tiap bilangan dari contoh dan persoalan saling berkorespondensi, sehingga target untuk contoh adalah bilangan yang berkorespondensi dengan variabel pada persoalan.

Permasalahan yang ketiga adalah jumlah masukan. Penulis menggunakan teknik heuristik yaitu jumlah masukan hanya boleh di atas 2 (dari 3 sampai seterusnya), karena jarang sekali diminta mencari pola hanya dari 2 angka. Makin besar masukan tentu akan makin menyulitkan, tetapi tidak ada dasar yang dapat

membatasi jumlah masukan yang besar selain ketidakmampuan program dalam memproses (*exhausted*).

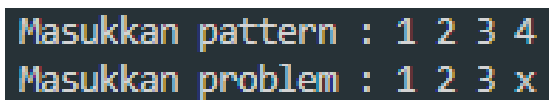
Permasalahan yang keempat adalah menentukan teknik heuristik yang cocok. Mayoritas pertanyaan figural ini jarang menggunakan operasi akar, logaritma, perpangkatan, dan operasi-operasi lain di luar penjumlahan (+), pengurangan (-), perkalian (*), dan pembagian (/). Atas dasar inilah, heuristik yang diambil adalah penerapan exhaustive search hanya akan berfokus pada empat operasi utama yang sudah disebutkan sebelumnya. Berdasarkan teknik heuristik ini, maka tidak ada operator unary (misalnya simbol akar) sehingga minimal harus ada 2 bilangan yang akan menghasilkan satu bilangan baru (sehingga masukan minimal adalah 3 bilangan).

B. Implementasi Program

Pada dasarnya, apa yang akan dilakukan oleh program hampir sama persis dengan apa yang dilakukan oleh peserta ujian, yaitu mencoba-coba pola yang memungkinkan. Keunggulan yang perlu diperhatikan dari program adalah kemampuan dalam mensimulasikan percobaan sangat cepat, sehingga persoalan bisa diselesaikan dengan cepat, berbeda dengan peserta ujian yang merupakan manusia, pada umumnya tidak mungkin kecepatan mensimulasikannya melebihi kecepatan program sehingga cenderung selesai melakukan simulasi ketika salah satu pola ditemukan.

Sejak awal, program ini tidak dibuat untuk mempermudah peserta ujian dalam mengerjakan soal saat ujian, tetapi sebagai alat bantu peserta ujian dalam pembelajaran, sehingga akan lebih baik apabila program bisa memberikan semua kemungkinan pola yang dapat diterapkan (enumerasi yang sudah divalidasi).

Program yang dibuat akan menerima 2 kelompok masukan, yaitu kumpulan angka dari contoh dan kumpulan angka beserta variabelnya dari persoalan. Permasalahan mengenai soal yang memiliki contoh lebih dari satu sudah ditangani dengan konsep program yang diterapkan karena pengguna tinggal memasukkan contoh dua kali, yaitu contoh pertama dengan persoalan dan contoh kedua dengan persoalan.



Gambar 3.1 Format masukan yang diimplementasikan

sumber : penulis

Setelah menerima kedua masukan tersebut, program akan memilih bilangan untuk menjadi target dari pola pada contoh. Langkah-langkah rinci dalam penerapan algoritma exhaustive search dan exhaustive search pada program :

1. Mencatat urutan bilangan selain target menggunakan permutasi. Bilangan yang dijadikan target akan dihapus sebelum proses permutasi dimulai, sehingga kompleksitasnya adalah $(n - 1)!$
2. Melakukan permutasi urutan operator sebanyak 1 kurangnya dari jumlah bilangan non target. Operator yang digunakan pada program ini hanya empat (+, -, *, /) sehingga kompleksitasnya adalah (4^{n-1}) .

3. Mengaplikasikan setiap urutan bilangan dengan urutan operator (enumerasi). Setiap hasil aplikasi yang nilainya sama dengan target dianggap sebagai kandidat solusi pada persoalan (evaluasi). Proses ini melibatkan hasil permutasi bilangan-bilangan dan operator sehingga kompleksitasnya menjadi $(n - 1)! * (4^{n-1})$.
4. Menerapkan urutan bilangan dan urutan operator yang sudah dikumpulkan ke persoalan. Hasil akan ditampilkan dan pengguna harus mencocokkan nilainya dengan jawaban dari kunci jawaban ataupun hasil berpikir. Langkah ini tentu hanya akan terjadi sekali (satu pasangan urutan bilangan dan operator akan diaplikasikan sekali ke persoalan), sehingga kompleksitas tetap seperti sebelumnya.

Hasil akhir dari kompleksitas implementasi algoritma exhaustive search pada program ini adalah $O(n! * 4^n)$. Terlihat dari kompleksitas yang didapatkan bahwa algoritma ini memberikan *cost* yang sangat besar ketika kedua permutasi dilakukan (tidak terlalu banyak batasan selain jumlah operator yang dibatasi). Namun, solusi akhir yang didapatkan akan sesuai harapan ketika n bilangan yang dimasukkan memang memiliki hubungan yang bisa dibentuk dari 4 operator tersebut.

IV. PENGUJIAN DAN ANALISIS

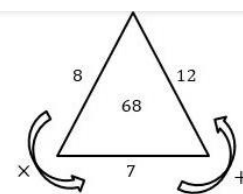
Penyusunan solusi yang sudah dijabarkan di bagian III diimplementasikan pada program sesungguhnya. Penulis menggunakan bahasa pemrograman Java untuk mengimplementasikan program ini. Kode program bisa dilihat di sini

<https://github.com/rkvilena/FiguralUTBKTools.git>

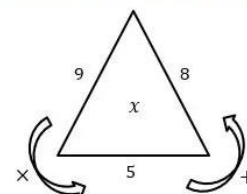
Pada bagian ini, akan ditampilkan 4 contoh hasil pengujian beserta analisisnya. Hasil pengujian akan menampilkan permasalahan yang ingin diselesaikan, urutan bilangan dan operatornya secara berurutan, dan nilai yang didapat dari menerapkan operator ke bilangan-bilangan persoalan.

A. Pengujian

1. Soal pertama



Dengan pola yang sama, kita dapatkan



Jadi, nilai x yang memenuhi adalah $9 \times 5 + 8 = 45 + 8 = 53$.

Gambar 4.1 Soal pertama dengan perhitungan 4 bilangan

sumber : <https://www.ruangguru.com/blog/latihan-soal-utbk-sbmptn-2021-penalaran-umum>

Soal pada Gambar 4.1 memiliki contoh [8,7,12,68] dan persoalan [9,5,8,x]. Kebetulan soal yang satu ini sudah ada pembahasannya, jadi sekarang hanya perlu periksa hasil dari program.

```
Masukkan pattern : 8 7 68 12
Masukkan problem : 9 5 x 8

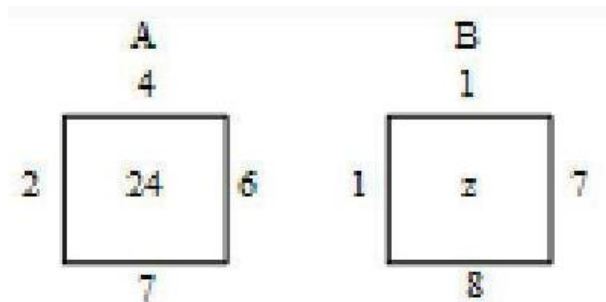
Hasil Pencocokan Pola
Urutan 1 : [9.0, 5.0, 8.0]
Operasi : [*, +]
Hasil : 53.0

Hasil Pencocokan Pola
Urutan 2 : [5.0, 9.0, 8.0]
Operasi : [*, +]
Hasil : 53.0
```

Gambar 4.2 Hasil pengujian soal 1
sumber : penulis

Bisa dilihat dari hasil pengujian bahwa ada dua cara untuk mendapatkan hasil yang sama (53), yaitu $9 \times 5 + 8$ dan $5 \times 9 + 8$. Tidak ditemukan hasil selain 53, yang artinya tidak mungkin ada cara lain untuk mendapatkan 53 dengan hanya menggunakan 4 operator utama yang dijadikan pendekatan non matematis pada program ini (teknik heuristik).

2. Soal kedua



Gambar 4.3 Soal kedua dengan perhitungan 5 bilangan
sumber : <https://brainly.co.id/tugas/38246962>

Soal pada Gambar 4.3 memiliki contoh [2,4,6,7,24] dan persoalan [1,1,7,8,x] (pada program akan ditulis x karena pada implementasi sudah diatur untuk menerima simbol (x)). Hasil perhitungan manual penulis adalah 2, dengan urutan operasi $8 * 1 - 7 + 1$. Pada contoh, pola ini didapatkan dengan proses $7 * 4 - 6 + 2 = 28$.

```
Masukkan pattern : 2 4 6 7 24
Masukkan problem : 1 1 7 8 x

Hasil Pencocokan Pola
Urutan 1 : [8.0, 1.0, 1.0, 7.0]
Operasi : [*, +, +]
Hasil : 16.0

Hasil Pencocokan Pola
Urutan 2 : [7.0, 1.0, 8.0, 1.0]
Operasi : [-, *, -]
Hasil : 47.0

Hasil Pencocokan Pola
Urutan 3 : [8.0, 1.0, 7.0, 1.0]
Operasi : [*, +, +]
Hasil : 16.0

Hasil Pencocokan Pola
Urutan 4 : [8.0, 1.0, 7.0, 1.0]
Operasi : [*, -, +]
Hasil : 2.0

Hasil Pencocokan Pola
Urutan 5 : [1.0, 8.0, 1.0, 7.0]
Operasi : [*, +, +]
Hasil : 16.0

Hasil Pencocokan Pola
Urutan 6 : [1.0, 8.0, 1.0, 7.0]
Operasi : [*, +, -]
Hasil : 2.0
```

```
Hasil Pencocokan Pola
Urutan 7 : [1.0, 8.0, 7.0, 1.0]
Operasi : [*, +, +]
Hasil : 16.0

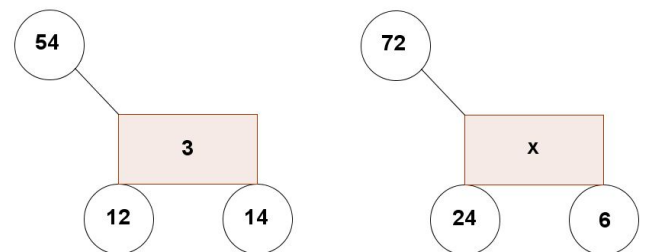
Hasil Pencocokan Pola
Urutan 8 : [8.0, 1.0, 1.0, 7.0]
Operasi : [*, +, -]
Hasil : 2.0

Hasil Pencocokan Pola
Urutan 9 : [1.0, 8.0, 7.0, 1.0]
Operasi : [*, -, +]
Hasil : 2.0
```

Gambar 4.4 Hasil pengujian soal 2
sumber : penulis

Bisa dilihat dari hasil pengujian bahwa ada 9 pola yang ditemukan dari contoh. Pada hasil pencocokan pola, ada 4 pola yang memberikan jawaban bernilai 2 (sama seperti yang dilakukan penulis). Kebetulan tidak bisa dipastikan jawaban sesungguhnya dari pembuat soal, tetapi soal ini membuktikan bahwa akan ada pola yang sesuai dengan jawaban yang diminta oleh soal.

3. Soal ketiga



Gambar 4.5 Soal ketiga dengan perhitungan 4 bilangan

sumber : <https://www.aiairyn.com/2020/04/soal-try-out-tps-kemampuan-penalaran.html>

Soal pada Gambar 4.5 memiliki contoh [3,54,12,14] dan persoalan [x,72,24,6]. Berdasarkan kunci jawaban, nilai x adalah 8, dengan cara $8 = (72 - 24) / 6$. Pola tersebut didapat dari contoh dengan bukti proses perhitungan $3 = (54 - 12) / 14$.

```

Masukkan pattern : 3 54 12 14
Masukkan problem : x 72 24 6

Hasil Pencocokan Pola
Urutan 1 : [72.0, 24.0, 6.0]
Operasi : [-, /]
Hasil : 8.0

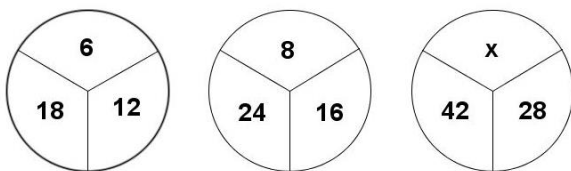
```

Gambar 4.6 Hasil pengujian soal 3

sumber : penulis

Bisa dilihat dari hasil pengujian bahwa hanya ada 1 pola yang ditemukan dari contoh untuk bisa diterapkan pada persoalan. Pada hasil pencocokan pola, bisa dilihat bahwa pola yang dihasilkan oleh program sama persis dengan kunci jawaban dari pembuat soal.

4. Soal keempat



Gambar 4.7 Soal keempat dengan perhitungan 3 bilangan

sumber : <https://www.aiairyn.com/2020/04/soal-try-out-tps-kemampuan-penalaran.html>

Soal pada Gambar 4.7 memiliki contoh [6,18,12], [8,24,16], dan persoalan [x,72,24,6]. Berdasarkan kunci jawaban, nilai dari x adalah 14. Pola yang digunakan oleh kunci jawaban sesungguhnya tidak ditangani oleh program yang hanya mengkali angka-angka yang ada pada soal ($42 / 3 * 2$).

```

Masukkan pattern : 6 18 12
Masukkan problem : x 42 28

Hasil Pencocokan Pola
Urutan 1 : [42.0, 28.0]
Operasi : [-]
Hasil : 14.0

```

Gambar 4.8 Hasil pengujian soal 4

sumber : penulis

Bisa dilihat dari hasil pengujian bahwa hanya ada 1 pola yang ditemukan dari contoh untuk bisa diterapkan pada persoalan. Memang pola yang diajukan oleh program tidak sesuai dengan yang diminta oleh kunci jawaban, tetapi kejadian ini justru menunjukkan bahwa brute force benar-benar menguji segala kemungkinan sehingga didapat pola dan hasil yang paling masuk akal. Selain itu, hasil pengujian soal 4 juga menunjukkan bahwa teknik heuristik yang diambil tidak melenceng.

B. Analisis

Berdasarkan hasil ujicoba, jumlah masukan tidak memberikan kepastian akan banyak pola yang layak diterapkan.

Kemungkinan besar alasannya adalah makin banyak masukan, sesungguhnya perhitungan makin kompleks sehingga kesempatan mendapatkan jawaban yang diinginkan makin kecil. Program ini didesain untuk memaksa setiap angka terlibat dalam perhitungan (tidak ada yang dibiarkan).

Selain itu, hasil ujicoba ini memberikan kepastian mengenai perhitungan yang menggunakan 4 operator dasar saja. Apabila program tidak memberikan solusi pola, artinya tidak ada kombinasi 4 operator dasar yang bisa memberikan hasil yang diinginkan (terpaksa pengguna harus memikirkan lagi jalan lain).

Kekurangan utama dari program ini adalah terlalu sempitnya jenis permasalahan yang dihadapi, salah satunya adalah penggunaan operator unary dan perhitungan yang melibatkan angka tambahan dari luar (seperti yang diharapkan oleh kunci jawaban pada persoalan ke-4).

V. KESIMPULAN

Algoritma brute force sebagai algoritma yang hampir selalu bisa memecahkan semua masalah ternyata mampu menyelesaikan permasalahan mencari pola dari kumpulan angka yang dimunculkan di soal UTBK. Dengan algoritma exhaustive search yang menerapkan brute force, semua kemungkinan urutan pola yang didapat akan diuji sehingga hasil akan selalu optimal. Kelebihan utama dari penerapan algoritma ini pada sebuah program adalah proses yang cepat (terutama untuk masukan yang jumlahnya sedikit) sehingga sebagai pengguna tidak perlu memakan waktu lama untuk memeriksa semua kemungkinan pola. Kekurangan utama dari penerapan algoritma ini pada sebuah program adalah besarnya kompleksitas program dan kurangnya model-model persoalan figural UTBK yang lainnya seperti menentukan nilai dari 2 variabel, menghitung hubungan setiap bilangan dengan memasukkan bilangan-bilangan tambahan, dan menghitung bilangan baru dari n bilangan yang dimasukkan. Implementasi algoritma exhaustive search pada soal-soal figural UTBK masih sangat bisa dikembangkan kedepannya untuk bisa menangani kekurangan dan menyempurnakan kelebihan yang sudah ada.

LINK VIDEO MAKALAH

Berikut link youtube untuk melihat video dari demonstrasi makalah ini : <https://youtu.be/cqzboFI7UPg>

PENUTUP

Penulis mengucapkan puji dan syukur kepada Tuhan YME karena telah mengizinkan makalah ini untuk tuntas secara tepat waktu. Selain itu, penulis juga berterima kasih yang sebesar-besarnya kepada:

1. Seluruh pengajar mata kuliah IF2211 Strategi Algoritma Semester II Tahun 2021/2022, terutama Ibu Dr. Nur Ulfa Maulidevi S.T., M.Sc. sebagai pengajar K-02.
2. Keluarga penulis yang bersedia mendukung dan memberi inspirasi dalam proses pembuatan makalah ini.

REFERENCES

- [1] [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-\(2022\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf), diakses pada 21 Mei 2022.
- [2] [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-\(2022\)-Bag2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag2.pdf), diakses pada 21 Mei 2022.
- [3] <https://www.geeksforgeeks.org/java-program-to-print-all-permutations-of-a-given-string/>, diakses pada 20 Mei 2022.
- [4] <https://www.aiairyn.com/2020/04/soal-try-out-tps-kemampuan-penalaran.html>, diakses pada 21 Mei 2022.

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, 22 Mei 2022



Raka Wirabuana Ninagan - 13520134