

IF2211 Strategi Algoritma

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force

Laporan Tugas Kecil

Disusun untuk memenuhi tugas besar mata kuliah IF2211 Strategi Algoritma pada
Semester II Tahun Akademik 2023/2024



Oleh

Shabrina Maharani 13522134

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2024**

DAFTAR ISI

DAFTAR ISI	2
BAB 1 DESKRIPSI MASALAH	3
1.1 Algoritma Brute Force	3
1.2 Permainan Cyberpunk 2077 Breach Protocol	4
BAB 2 PENYELESAIAN	5
2.1 Langkah Penyelesaian Cyberpunk 2077 Breach Protocol dengan pendekatan Algoritma Brute Force	5
BAB 3 IMPLEMENTASI PROGRAM	7
3.1 Spesifikasi Teknis Program	7
3.1.1 Struktur Data	7
3.1.2 Fungsi dan Prosedur	7
3.1.3 Source Code Program	16
3.1.4 Library	22
BAB 4 ANALISIS DAN PENGUJIAN	23
4.1. Kasus Uji (screenshot)	23
4.1.1 Kasus Uji Berdasarkan Contoh pada Spesifikasi Tugas Kecil 1	23
4.1.2 Kasus Uji Mandiri	24
4.1.2.1 Masukan dari File	24
4.1.2.2 Masukan dari Keyboard	28
BAB 5 KESIMPULAN	33
BAB 6 LAMPIRAN	34
6.1. Github	34
6.2. Tabel Pemeriksaan	34
DAFTAR PUSTAKA	35

BAB 1 DESKRIPSI MASALAH

1.1 Algoritma Brute Force

Algoritma *Brute Force* adalah sebuah metode algoritma dengan pendekatan yang *straightforward* (lempang) untuk menyelesaikan suatu persoalan. Algoritma *brute force* biasanya didasarkan pada definisi atau konsep yang dilibatkan dan pernyataan pada persoalan (*problem statement*). Algoritma *brute force* memiliki ciri-ciri untuk memecahkan persoalan dengan sangat sederhana, langsung, langkah penyelesaiannya yang jelas (*obvious way*) dan dilakukan dengan prinsip *just do it!* atau *just solve it!*.

Pada umumnya, algoritma *brute force* bukan merupakan algoritma yang cerdas dan mangkus. Hal ini dikarenakan algoritma *brute force* membutuhkan waktu yang cukup lama dan komputasi dengan volume yang besar dalam menyelesaikan persoalan. Kata “*force*” dalam *brute force* memiliki makna konotasi yaitu tenaga dibandingkan otak. Terkadang, algoritma *brute force* ini disebut juga dengan *naive algorithm* (algoritma naif).

Algoritma *brute force* lebih sesuai digunakan untuk persoalan dengan ukuran masukannya kecil dengan pertimbangan algoritma *brute force* cocok dipakai untuk persoalan yang sederhana dan implementasinya mudah. Algoritma *brute force* juga marak digunakan untuk menjadi basis pembanding dengan algoritma lain yang lebih cerdas dan mangkus. Namun, walaupun algoritma *brute force* bukan merupakan metode menyelesaikan permasalahan dengan mangkus, semua persoalan hampir dapat diselesaikan dengan algoritma *brute force*. Kecil kemungkinannya untuk menemukan persoalan yang tidak dapat diselesaikan dengan metode *brute force*. Bahkan, terdapat persoalan yang hanya mampu diselesaikan dengan algoritma *brute force*.

Algoritma *brute force* memiliki aplikabilitas yang luas dalam menyelesaikan berbagai masalah, karena sederhana dan mudah dipahami. Pendekatan ini menghasilkan solusi yang layak untuk berbagai masalah penting, termasuk pencarian, pengurutan, pencocokan string, dan perkalian matriks. Selain itu, algoritma *brute force* seringkali menjadi standar dalam tugas-tugas komputasi seperti penjumlahan atau perkalian sejumlah besar bilangan, serta menemukan elemen minimum atau maksimum dalam sebuah daftar.

Meskipun memiliki kelebihan dalam aplikabilitas dan kemudahan pemahaman, algoritma *brute force* memiliki beberapa kelemahan yang perlu

dipertimbangkan. Pertama, algoritma ini jarang menghasilkan solusi yang optimal atau efisien. Kedua, kinerjanya umumnya lambat untuk masukan berukuran besar, membuatnya tidak praktis dalam situasi di mana efisiensi waktu menjadi faktor penting. Terakhir, pendekatan brute force cenderung kurang konstruktif atau kreatif dalam memecahkan masalah, karena hanya bergantung pada enumerasi seluruh kemungkinan solusi tanpa mempertimbangkan strategi yang lebih canggih atau pintar.

1.2 Permainan Cyberpunk 2077 Breach Protocol

Cyberpunk 2077 Breach Protocol adalah minigame meretas yang terdapat dalam permainan video Cyberpunk 2077. Dalam minigame ini, pemain akan menghadapi simulasi peretasan jaringan local dari ICE (Intrusion Countermeasures Electronics) dalam dunia Cyberpunk 2077. Komponen utama dalam permainan ini meliputi token, matriks, sekuens, dan buffer. Token merupakan karakter alfanumerik dua digit seperti E9, BD, dan 55. Matriks adalah susunan token yang harus dipilih pemain untuk menyusun urutan kode. Sekuens adalah rangkaian token dua atau lebih yang harus dicocokkan. Buffer adalah jumlah maksimal token yang dapat disusun secara sekuensial.

Aturan permainan Breach Protocol mengharuskan pemain untuk bergerak dengan pola horizontal, vertikal, horizontal, dan vertikal (bergantian) hingga semua sekuens berhasil dicocokkan atau buffer penuh. Pemain memulai dengan memilih satu token pada posisi baris paling atas dari matriks, kemudian sekuens dicocokkan pada token-token yang berada di buffer. Satu token pada buffer dapat digunakan pada lebih dari satu sekuens, dan setiap sekuens memiliki bobot hadiah atau reward yang variatif. Selain itu, sekuens juga memiliki panjang minimal berupa dua token.

Cyberpunk 2077 Breach Protocol menambahkan elemen kecerdasan dan strategi dalam permainan dengan meminta pemain untuk merencanakan langkah-langkah mereka dengan bijak untuk mencocokkan sekuens dengan efisien dan memperoleh hadiah maksimal. Selain itu, game ini juga menghadirkan tantangan dalam memanfaatkan token dengan cara yang optimal dalam batasan buffer yang diberikan.

BAB 2 PENYELESAIAN

2.1 Langkah Penyelesaian Cyberpunk 2077 Breach Protocol dengan pendekatan Algoritma Brute Force

Berikut adalah langkah-langkah penyelesaian Cyberpunk 2077 Breach Protocol dengan pendekatan Algoritma Brute Force:

1. Pengguna diminta memilih masukan dari *keyboard* atau dari file txt. Jika masukan dilakukan dari *keyboard*, pengguna akan diminta untuk memasukkan jumlah token, token-token yang akan digunakan, ukuran maksimal buffer, ukuran matriks (baris dan kolom), jumlah sekuens, dan jumlah maksimal token dalam satu sekuens. Sedangkan jika dari file, file tersebut harus berisi ukuran maksimal buffer, ukuran matriks (baris dan kolom), matriks, jumlah sekuens, serta sekuens beserta bobotnya, dan semuanya harus terurut per-barisnya.
2. Program akan menyimpan data yang sudah dimasukkan pengguna. Jika masukannya dari *keyboard*, program akan melakukan randomisasi pada nilai matriks dan pola sekuensnya. Jika masukan dari file, program akan menyimpan data-data tersebut ke dalam variabel atau objek yang sesuai.
3. Pendekatan *brute force* dimulai dengan mencari seluruh kemungkinan jalur sekuens yang dapat ditempuh oleh matriks tanpa membandingkan terlebih dahulu dengan pola sekuens yang ada. Program akan melakukan pencarian dengan selang-seling antara vertikal dan horizontal. Pencarian dilakukan secara rekursif.
4. Setelah menemukan semua kemungkinan jalur sekuens, program akan membandingkan satu per satu dengan pola sekuens yang dimasukkan pengguna. Jika terdapat jalur sekuens yang sesuai dengan aturan permainan dan pola sekuensnya, program akan melakukan perhitungan bobot dari sekuens terkait. Semua sekuens dan bobot yang ditemukan disimpan dalam bentuk objek *sequence*.
5. Objek-objek *sequence* yang ditemukan akan dimasukkan ke dalam sebuah *list of sequence*.

6. Dari *list of sequence*, program akan mencari bobot yang paling maksimal dengan menyimpan bobot dari indeks pertama dalam list ke dalam sebuah variabel dan juga menyimpan jalur sekuens (berupa token) ke dalam variabel lainnya. Setelah itu, program akan melakukan iterasi ke indeks-indeks selanjutnya. Jika ada bobot yang lebih besar, variabel maksimum diganti dengan bobot tersebut beserta dengan sekuensnya.

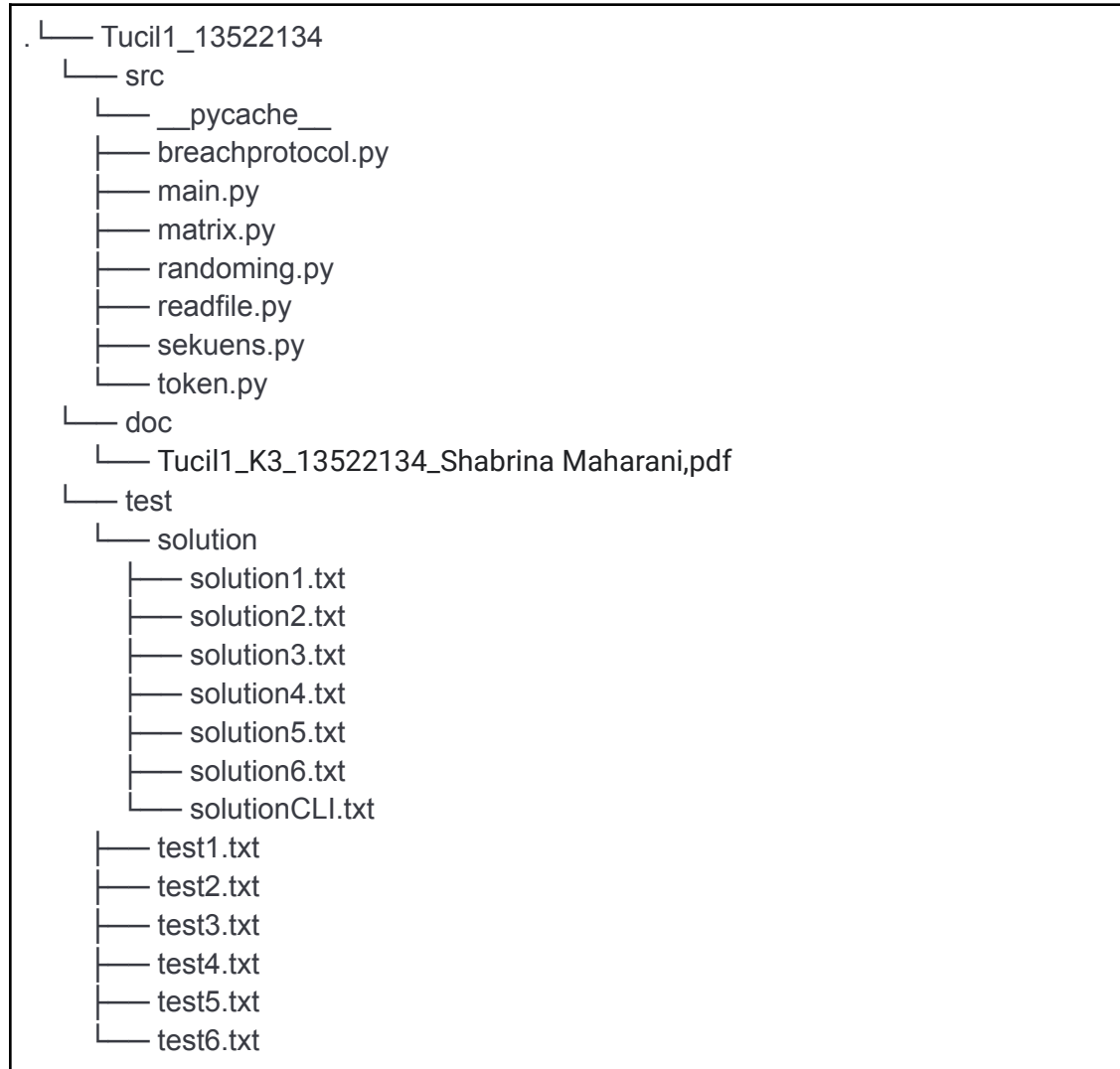
7. Setelah mendapatkan bobot maksimal, program akan mengeluarkan output berupa reward maksimal, sekuens(dalam bentuk token) yang memiliki reward maksimal, koordinat matriks dari sekuensnya, dan waktu eksekusi. Program juga dapat melakukan penyimpanan solusi ke folder test dalam format .txt jika pengguna menginginkan.

Dengan demikian, langkah-langkah di atas membantu penulis dalam menyelesaikan Cyberpunk 2077 Breach Protocol dengan menggunakan pendekatan algoritma *brute force*.

BAB 3 IMPLEMENTASI PROGRAM

3.1 Spesifikasi Teknis Program

3.1.1 Struktur Data



3.1.2 Fungsi dan Prosedur

Fungsi / Prosedur	Tujuan
Class Matriks	

<pre>def __init__(self, baris, kolom)</pre>	Fungsi inisialisasi untuk membuat objek Matriks.
<pre>def setVal(self, i, j, value)</pre>	Fungsi setVal dibuat untuk memungkinkan pengguna untuk mengatur atau mengubah nilai dalam setiap kotak matriks dengan koordinat tertentu. Tujuan utamanya adalah untuk menyediakan cara yang mudah dan aman bagi pengguna untuk memperbarui nilai dalam matriks tanpa harus khawatir tentang kesalahan indeks. Dengan memeriksa terlebih dahulu apakah koordinat yang diberikan berada dalam rentang yang valid untuk matriks, fungsi ini memastikan bahwa perubahan nilai dilakukan secara benar dan aman. Jika koordinat yang diberikan berada di dalam rentang yang valid, nilai dalam kotak matriks tersebut diubah sesuai dengan nilai yang disediakan. Jika tidak, fungsi akan mengembalikan IndexError yang memberi tahu pengguna bahwa indeks yang diberikan keluar dari rentang yang valid untuk matriks.
Class Token	
<pre>def __init__(self, token, c, r)</pre>	Fungsi inisialisasi untuk membuat objek Stack. Membuat atribut stack yang merupakan string untuk menyimpan elemen-elemen dalam tumpukan.
Class Sequence	
<pre>def __init__(self, tokens, weight)</pre>	Fungsi inisialisasi untuk membuat objek Sequence. Parameter `token`, `c`, dan `r` digunakan untuk menyimpan nilai token, kolom, dan baris dari token tersebut ke

	<p>dalam atribut-atribut objek yang sesuai. Dengan demikian, tujuan utama dari fungsi ini adalah memungkinkan pembuatan objek token dengan informasi yang diberikan secara langsung saat pembuatan objek, sehingga memudahkan penggunaan objek token tersebut dalam program secara lebih efisien dan terorganisir.</p>
Class FillMtx	
<pre>def __init__(self, tokens)</pre>	<p>Fungsi `FillMtx` dibuat dengan tujuan untuk menginisialisasi objek yang merepresentasikan matriks dengan menggunakan token yang diberikan sebagai argumen saat objek dibuat. Dengan cara ini, setiap instance dari kelas `FillMtx` memiliki akses ke token yang digunakan untuk mengisi matriksnya. Tujuannya adalah untuk menyediakan sebuah objek yang siap digunakan untuk memanipulasi dan mengoperasikan matriks dengan token yang telah ditentukan sebelumnya, memungkinkan penggunaan fungsi-fungsi lain dalam kelas ini untuk melakukan manipulasi matriks dengan lebih mudah dan intuitif.</p>
<pre>def filling_matrix(self, baris, kolom)</pre>	<p>Fungsi `filling_matrix` dalam kelas `FillMtx` bertujuan untuk menghasilkan sebuah matriks dengan ukuran baris dan kolom yang ditentukan, di mana setiap elemen dalam matriks diisi dengan token acak yang dipilih dari kumpulan token yang tersedia dalam objek kelas tersebut. Tujuan utamanya adalah untuk menyediakan sebuah mekanisme yang</p>

	memungkinkan pengguna untuk dengan mudah membuat matriks dengan isi yang bervariasi secara acak, sesuai dengan kebutuhan aplikasi atau permasalahan yang ingin diselesaikan.
matrix.py	
<pre>def print_matrix (matrix)</pre>	<p>Fungsi `print_matrix` diciptakan untuk mencetak matriks ke layar. Dengan menggunakan loop for, fungsi ini mengakses setiap baris dalam matriks dan kemudian mengonversi setiap elemen dalam baris tersebut menjadi string menggunakan fungsi `map(str, row)`. Hasilnya digabungkan menjadi satu string dengan spasi sebagai pemisah antara elemen-elemen tersebut menggunakan metode `join`. Akhirnya, string tersebut dicetak ke layar. Tujuannya adalah memudahkan penggunaan matriks dalam pemrograman dengan memberikan cara yang sederhana dan mudah dipahami untuk mencetak matriks ke layar.</p>
readfile.py	
<pre>def read_file (filename)</pre>	<p>Fungsi `read_file` bertujuan untuk membaca data dari sebuah file teks yang berisi informasi terkait ukuran matriks, data matriks, jumlah dan bobot dari beberapa urutan (sequences). Tujuan utamanya adalah untuk memproses data dari file tersebut dan mengembalikan informasi yang relevan dalam bentuk yang sesuai untuk digunakan dalam program, seperti ukuran buffer, ukuran matriks, data</p>

	matriks, jumlah total urutan, dan daftar urutan beserta bobotnya.
randoming.py	
<pre>def random_sequence (tokens, max_sequence)</pre>	Tujuan dari fungsi `random_sequence` adalah untuk menghasilkan urutan acak dari sekelompok token dengan panjang maksimum tertentu. Fungsi ini mengambil dua parameter, yaitu `tokens`, yang merupakan kumpulan token yang akan diacak, dan `max_sequence`, yang menentukan panjang maksimum urutan acak yang dihasilkan.
<pre>def random_weight ()</pre>	Fungsi `random_weight()` diciptakan dengan tujuan untuk menghasilkan bobot acak untuk sekuens dengan batas yang ditentukan. Bobot tersebut dapat digunakan dalam berbagai aplikasi, seperti dalam algoritma optimasi, pembelajaran mesin, atau simulasi. Dengan memperbolehkan nilai bobot menjadi negatif, fungsi ini memungkinkan variasi yang lebih luas dalam representasi data, sehingga meningkatkan fleksibilitas dalam penggunaannya.
<pre>def random_sequences (tokens,total_sequence, max_sequence</pre>	Fungsi `random_sequences` dibuat dengan tujuan untuk menghasilkan sejumlah urutan acak (sequences) berdasarkan sejumlah token yang tersedia. Tujuannya adalah untuk menghasilkan kumpulan urutan acak sebanyak `total_sequence` dengan panjang maksimum `max_sequence`. Fungsi ini mengambil token dan panjang maksimum urutan sebagai input, dan kemudian

	<p>menggunakan fungsi bantu <code>`random_sequence`</code> untuk menghasilkan urutan acak dengan panjang maksimum yang ditentukan. Setiap urutan kemudian diberi bobot (weight) acak dan disimpan dalam objek <code>`Sequence`</code>. Akhirnya,</p> <p>kumpulan urutan yang dihasilkan dikembalikan sebagai output dari fungsi ini.</p>
breachprotocol.py	
<pre>def check_token_validity (tokens,sequence)</pre>	<p>Fungsi <code>`check_token_validity`</code> dibuat untuk memeriksa apakah sebuah urutan (sequence) dapat ditemukan dalam daftar token yang diberikan. Tujuannya adalah untuk mengembalikan nilai True jika urutan tersebut ditemukan secara berurutan dalam daftar token, dan False jika urutan tersebut tidak ditemukan atau urutan token lebih panjang dari daftar token. Fungsi ini beroperasi dengan menggunakan pendekatan brute force, yaitu dengan memeriksa setiap kemungkinan awal dari urutan di dalam daftar token untuk mencocokkan urutan yang diberikan. Dengan cara ini, fungsi ini memberikan kemampuan untuk memverifikasi keberadaan urutan dalam daftar token dengan kompleksitas waktu yang cukup efisien.</p>
<pre>def check_move_validity (baris,kolom,stack)</pre>	<p>Fungsi ini bertujuan untuk memeriksa apakah suatu langkah (dinyatakan oleh <code>`baris`</code> dan <code>`kolom`</code>) valid dalam permainan atau tidak. Ini dilakukan dengan memeriksa apakah posisi yang</p>

	<p>dituju oleh langkah tersebut sudah ada di dalam `stack`, yaitu sebuah struktur data yang berisi token-token yang telah ditempatkan sebelumnya di papan permainan. Jika posisi tersebut sudah ada dalam `stack`, fungsi akan mengembalikan nilai `True`, menandakan bahwa langkah tersebut valid. Namun, jika tidak, fungsi akan mengembalikan nilai `False`, menandakan bahwa langkah tersebut tidak valid dan tidak dapat dilakukan. Dengan demikian, tujuan utama fungsi ini adalah untuk memvalidasi langkah-langkah yang diambil dalam permainan dengan memeriksa apakah langkah tersebut telah diambil sebelumnya atau tidak.</p>
<pre>def sum_weight(sequences,token)</pre>	<p>Fungsi `sum_weight` dibuat untuk menghitung total bobot dari urutan (sequences) yang valid, berdasarkan token-token yang valid yang diberikan. Fungsi ini menerima dua parameter: `sequences`, yang merupakan daftar urutan, dan `tokens`, yang merupakan daftar token. Tujuan utama dari fungsi ini adalah untuk memeriksa setiap urutan dalam `sequences` dan menambahkan bobot urutan tersebut ke dalam variabel `reward` jika urutan tersebut valid berdasarkan token-token yang valid dalam `tokens`. Untuk melakukan ini, fungsi menggunakan fungsi bantuan `check_token_validity`, yang memeriksa apakah setiap token dalam urutan ada dalam daftar token yang valid. Setelah memproses semua urutan, fungsi mengembalikan total bobot yang</p>

	diakumulasikan dari urutan-urutan yang valid.
<pre>def find_optimal_reward (possible_sequence,sequences)</pre>	<p>Fungsi `find_optimal_reward` dibuat untuk mencari urutan optimal dari suatu himpunan urutan yang mungkin (`possible_sequence`). Ini dilakukan dengan memeriksa setiap urutan dalam `possible_sequence` dan menghitung total bobotnya menggunakan fungsi `sum_weight`. Fungsi ini kemudian mengembalikan urutan dengan bobot total tertinggi beserta bobot total tersebut.</p>
<pre>def find_path(matrix, stack,row,column,buffer_size,optimal_path, horizontal)</pre>	<p>Fungsi `find_path` bertujuan untuk mengeksplorasi semua kemungkinan jalur dalam matriks yang diberikan, dimulai dari koordinat yang ditentukan. Ini dilakukan dengan menggunakan pendekatan rekursif. Pada setiap langkah rekursif, fungsi memeriksa kemungkinan langkah yang valid secara horizontal atau vertikal, tergantung pada parameter `horizontal`. Fungsi juga memperhitungkan buffer_size yang menentukan panjang maksimum jalur yang akan dieksplorasi. Saat buffer_size mencapai 1, jalur yang sedang dieksplorasi dianggap optimal dan ditambahkan ke daftar optimal_path. Dengan demikian, tujuan utama dari fungsi ini adalah untuk menemukan semua jalur yang mungkin dalam matriks dengan mempertimbangkan batasan panjang jalur yang diizinkan.</p>
<pre>def print_solution(opt_reward,opt path)</pre>	<p>Fungsi `print_solution` bertujuan untuk mencetak informasi tentang solusi optimal</p>

	<p>dari suatu masalah. Tujuan utamanya adalah untuk menghasilkan output yang terstruktur dan informatif yang mencakup bobot hadiah optimal, isi buffer optimal, dan koordinat dari setiap token dalam solusi. Fungsi ini menerima dua parameter: <code>`opt_reward`</code>, yang merupakan bobot hadiah optimal, dan <code>`opt_path`</code>, yang merupakan urutan token dalam solusi optimal. Jika tidak ada solusi yang memenuhi kriteria tertentu, fungsi akan mencetak pesan bahwa tidak ada solusi yang memenuhi. Dengan demikian, fungsi ini membantu dalam menampilkan hasil secara jelas kepada pengguna, memungkinkan mereka untuk memahami solusi optimal dari masalah yang diberikan.</p>
--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

3.1.3 Source Code Program

1. File main.py

File ini berperan sebagai *driver* utama dari program ini, dengan menyediakan menu utama dan mendeklarasikan variabel yang akan digunakan. Di dalamnya tidak terdapat implementasi fungsi-fungsi program, hanya fokus pada interaksi dengan pengguna dan persiapan variabel yang dibutuhkan oleh program.

```
src > % main.py
1 #Shabrina Maharani - 13522134
2
3 import os
4 import time
5 from matrix import Matriks, print_matrix
6 from sekuens import Sequence
7 from token import Token
8 from readFile import read_file
9 from randoming import FillMtx, random_sequences
10 from breachprotocol import find_optimal_reward, find_path, print_solution
11
12 print("\033[96m" + r""
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2
```



```

src > matrix.py
1 class Matriks:
2     # Fungsi untuk menmbetuk sebuah matriks
3     def __init__(self, baris, kolom):
4         self.baris = baris
5         self.kolom = kolom
6         self.data = [[0 for _ in range(kolom)] for _ in range(baris)]
7
8     # Fungsi untuk melakukan setting atau perubahan value dalam setiap kotak matriks
9     def setVal(self, i, j, value):
10        if 0 <= i < self.baris and 0 <= j < self.kolom:
11            self.data[i][j] = value
12        else:
13            raise IndexError(f"Index out of range")
14
15
16 #Fungsi untuk mencetak matriks
17 def print_matrix(matrix):
18     for row in matrix.data:
19         print(" ".join(map(str, row)))

```

3. File sekuens.py

File ini hanya berisi class dari Sequence.

```

src > sekuens.py
1 class Sequence:
2     def __init__(self, tokens, weight):
3         self.tokens = tokens
4         self.weight = weight #weight = bobot hadiah

```

4. File token.py

File ini hanya berisi class dari Token.

```


src > token.py
1 class Token :
2     def __init__(self, token, c, r):
3         self.token = token
4         self.row = r          # r = row
5         self.column = c       # c = column

```

5. File readfile.py

File ini berisi fungsi yang digunakan untuk membaca sebuah file dan mengembalikan data-data yang diperlukan dalam perhitungan.

```

src >  readfile.py
1  from sekuens import Sequence
2  def read_file(filename):
3      with open(filename, 'r') as file:
4          lines = file.readlines()
5          # Membaca line pertama
6          buffer_size = int(lines[0])
7
8          # Membaca line kedua dan menjadikannya sebagai tuple (kolom,baris)
9          matrix_size = tuple(map(int, lines[1].split()))
10
11         # Membaca dan menyimpan data matriks dari file
12         matrix_data = [line.split() for line in lines[2:2+matrix_size[1]]]
13
14         # Membaca dan menyimpan jumlah sekuens
15         total_sequence = int(lines[2+matrix_size[1]])
16
17         sequences = []
18         # Menyimpan sequence dalam sebuah list of Sequence
19         for i in range(2+matrix_size[1]+1, len(lines), 2):
20             sequence_tokens = lines[i].split()
21             weight = int(lines[i+1])
22             sequences.append(Sequence(sequence_tokens, weight))
23
24     return buffer_size, matrix_size, matrix_data, total_sequence, sequences

```

6. File randoming.py

File ini berisi class FillMtx dan fungsi - fungsi yang digunakan untuk melakukan randomisasi pada matriks dan sekuens jika pengguna memasukkan input melalui *keyboard*.

```

src > randoming.py
1  import random
2  from sekuens import Sequence
3
4  #Melakukan randomisasi komponen matriks
5  class FillMtx:
6      def __init__(self, tokens):
7          self.tokens = tokens
8
9      def filling_matrix(self, baris, kolom):
10         matrix = []
11         for _ in range(baris):
12             row = []
13             for _ in range(kolom):
14                 random_token = random.choice(self.tokens)
15                 row.append(random_token)
16             matrix.append(row)
17         return matrix
18
19 #Melakukan randomisasi untuk pola sekuens beserta bobotnya
20 def random_sequence(tokens, max_sequence):
21     sequence = random.sample(tokens, min(len(tokens), random.randint(2, max_sequence)))
22     return sequence
23
24 def random_weight():
25     # Asumsi : Bobot sekuens diberikan batas dari -100 sampai 100 karena di QNA dikatakan bisa saja bernilai negatif
26     return random.randint(-100,100)
27
28 def random_sequences(tokens, total_sequence, max_sequence):
29     sequences = []
30     for _ in range(total_sequence):
31         sequence = random_sequence(tokens, max_sequence)
32         weight = random_weight()
33         sequences.append(Sequence(sequence, weight))
34     return sequences
35

```

7. File breachprotocol.py

File ini berisi fungsi-fungsi yang digunakan dalam perhitungan solusi.

```

src > breachprotocol.py
1  from token import Token
2  from matrix import Matriks
3
4  def check_token_validity(tokens,sequence):
5      if len(sequence)>len(tokens):
6          return False
7      else:
8          for i in range(len(tokens)-len(sequence)+1):
9              if tokens[i] == sequence[0]:
10                 valid = True
11                 for j in range(1,len(sequence)):
12                     if tokens[i+j] != sequence[j]:
13                         valid = False
14                         break
15
16                 if valid:
17                     return True
18             return False
19
20 def check_move_validity(baris, kolom, stack):
21     if stack != []:
22         for token in stack:
23             if (token.row, token.column) == (baris, kolom):
24                 return True
25     return False
26

```

```

src > breachprotocol.py
26
27 def sum_weight(sequences,tokens):
28     reward = 0
29     for seq in sequences:
30         if check_token_validity([token.token for token in tokens],seq.tokens):
31             reward += seq.weight
32     return reward
33
34 def find_optimal_reward(possible_sequence,sequences):
35     maks = sum_weight(sequences,possible_sequence[0])
36     makspath = possible_sequence[0]
37     for i in range(1,len(possible_sequence)):
38         if sum_weight(sequences,possible_sequence[i]) > maks :
39             maks = sum_weight(sequences,possible_sequence[i])
40             makspath = possible_sequence[i]
41     return maks,makspath
42
43
44 def find_path(matrix, stack, row, column, buffer_size, optimal_path, horizontal):
45     if buffer_size != 1:
46         if not horizontal:
47             for i in range(matrix.baris):
48                 if not check_move_validity(i ,column, stack):
49                     stack.append(Token(matrix.data[i][column],column,i))
50                     find_path(matrix, stack, i, column, buffer_size-1, optimal_path, True)
51                     stack.pop()
52             else :
53                 for i in range(matrix.kolom):
54                     if not check_move_validity(row, i, stack):
55                         stack.append(Token(matrix.data[row][i],i,row))
56                         find_path(matrix, stack, row, i, buffer_size-1, optimal_path, False)
57                         stack.pop()
58         else :
59             optimal_path.append(list(stack))
60

```

```

def print_solution(opt_reward, opt_path):
    string_output = ""
    if opt_reward != 0:
        string_output += f"Bobot Hadiah Optimal : {opt_reward}\n"
        flag = True
        buffer_content = ""
        coordinate_content = ""
        for i in opt_path:
            buffer_content += i.token + " "
            coordinate_content += f"{i.column + 1}, {i.row + 1}\n"
        string_output += f"Isi dari Buffer Optimal : {buffer_content.strip()}\n"
        string_output += f"Koordinat dari Setiap Token :\n{coordinate_content.strip()}"
    else:
        string_output += "Tidak ada solusi yang memenuhi\n"
    print(string_output)
    return string_output

```

3.1.4 Library

1. Import os

Library os digunakan untuk berinteraksi dengan sistem operasi. Ini membantu program untuk melakukan berbagai operasi terkait sistem file, seperti membuat, mengubah, dan menghapus direktori atau file, mengelola variabel lingkungan, dan menavigasi struktur direktori. Dengan os, program dapat lebih fleksibel dan dapat beradaptasi dengan berbagai lingkungan sistem operasi.

2. Import time

Library time digunakan untuk bekerja dengan waktu dan penundaan dalam program Python. *Library* ini menyediakan fungsi-fungsi untuk mendapatkan waktu saat ini, mengonversi waktu antara format yang berbeda, serta mengukur durasi eksekusi suatu kode.

BAB 4 ANALISIS DAN PENGUJIAN

4.1. Kasus Uji (screenshot)

4.1.1 Kasus Uji Berdasarkan Contoh pada Spesifikasi Tugas Kecil 1

Isi File test1.txt

```
test > test1.txt
1 7
2 6 6
3 7A 55 E9 E9 1C 55
4 55 7A 1C 7A E9 55
5 55 1C 1C 55 E9 BD
6 BD 1C 7A 1C 55 BD
7 BD 55 BD 7A 1C 1C
8 1C 55 55 7A 55 7A
9 3
10 BD E9 1C
11 15
12 BD 7A BD
13 20
14 BD 1C BD 55
15 30
```

Solusi

```
test > solution > solution1.txt
1 Bobot Hadiah Optimal : 50
2 Isi dari Buffer Optimal : 7A BD 7A BD 1C BD 55
3 Koordinat dari Setiap Token :
4 1, 1
5 1, 4
6 3, 4
7 3, 5
8 6, 5
9 6, 3
10 1, 3
11
12
13 Waktu eksekusi program: 390.453577041626 ms
```

Analisis : Program penulis menunjukkan bobot hadiah optimal yang ditemukan sesuai dengan yang tercantum dalam spesifikasi. Namun, program penulis menghasilkan sekuens token yang berbeda. Ini disebabkan oleh pendekatan pencarian maksimum yang digunakan dalam program penulis, yang mencari kemungkinan sekuens token yang paling terakhir dengan yang sama-sama maksimal. Dengan demikian, solusi yang dihasilkan mencerminkan sekuens yang merupakan salah satu kemungkinan sekuens optimal terakhir dengan bobot maksimal yang ditemukan oleh algoritma pencarian. Meskipun hasilnya berbeda dalam urutan token, bobot hadiah yang diperoleh tetap sesuai dengan yang diharapkan, menunjukkan

keefektifan algoritma dalam mencapai bobot maksimal yang telah ditetapkan.

4.1.2 Kasus Uji Mandiri

4.1.2.1 Masukan dari File

1. Kasus Uji 1

Masukan file test2.txt :

```
test > test2.txt
1 5
2 4 4
3 BC BC AB AB
4 DA AB AB CD
5 BC DA BC CD
6 AB BC BC BC
7 4
8 DA AB CD
9 -12
10 DA AB CD
11 -26
12 AB DA
13 -17
14 CD AB
15 61
```

Solusi dari test2.txt :

```
test > solution > solution2.txt
1 Bobot Hadiah Optimal : 61
2 Isi dari Buffer Optimal : BC DA CD AB BC
3 Koordinat dari Setiap Token :
4 1, 1
5 1, 2
6 4, 2
7 4, 1
8 2, 1
9
10
11 Waktu eksekusi program: 3.9985179901123047 ms
12
```

Analisis : Dari pengujian di atas, kita dapat membuktikan kebenaran teori yang menyatakan bahwa algoritma *brute force* cocok digunakan untuk ukuran (dalam hal ini matriks) yang tergolong kecil, penyelesaian yang sederhana dan implementasinya mudah terbukti benar. Hal ini dapat dibuktikan dengan waktu eksekusi yang digunakan dalam proses penyelesaiannya hanya membutuhkan 4 ms (dibulatkan).

2. Kasus Uji 2

Masukan file test3.txt :

```
test > test3.txt
1 6
2 7 8
3 ST MN OP UV MN KL ST
4 UV KL QR KL QR QR MN
5 KL OP MN UV UV OP ST
6 OP QR ST KL ST OP MN
7 OP ST KL OP ST ST QR
8 UV QR UV OP KL UV OP
9 UV UV QR MN MN UV UV
10 KL ST KL UV QR QR MN
11 4
12 ST UV
13 50
14 ST QR
15 19
16 QR OP
17 78
18 MN ST OP
19 50
```

Solusi dari test3.txt :

```
test > solution > solution3.txt
1 Bobot Hadiah Optimal : 147
2 Isi dari Buffer Optimal : ST UV KL ST QR OP
3 Koordinat dari Setiap Token :
4 1, 1
5 1, 2
6 2, 2
7 2, 5
8 7, 5
9 7, 6
10
11
12 Waktu eksekusi program: 420.2098846435547 ms
13
```

Analisis : Dari pengujian di atas, kita dapat membuktikan kebenaran teori yang menyatakan bahwa algoritma *brute force* tidak terlalu sesuai digunakan untuk penyelesaian dengan ukuran (dalam hal ini matriks) yang cukup besar. Hal ini dapat dibuktikan dengan waktu eksekusi yang digunakan dalam proses penyelesaiannya membutuhkan 420 ms (dibulatkan). Waktu eksekusi yang sangat jauh dibandingkan dengan ukuran matriks yang kecil.

3. Kasus Uji 3

Masukan file test4.txt :

```
test > test4.txt
1 4
2 10 10
3 1C BD BD BD 7A 7A 7A BD 1C 1C
4 1C 7A BD 1C 1C 7A 1C BD BD 1C
5 1C BD 1C 7A BD BD BD BD 1C 1C
6 1C BD 1C BD BD 1C BD 7A 1C 1C
7 7A 1C BD 7A 7A 1C 7A 7A BD BD
8 7A 7A 1C 1C BD 1C 7A 1C BD 1C
9 7A 1C 7A 1C 1C 7A BD BD BD 7A
10 BD 7A 1C 1C BD 7A 1C 1C 1C BD
11 7A 7A 7A 7A BD 1C 1C 1C BD 1C
12 1C 1C BD 7A BD 1C BD 1C 1C 1C
13 3
14 7A BD
15 52
16 1C 7A BD
17 -49
18 BD 1C 7A
19 -12
```

Solusi dari test4.txt :

```
test > solution > solution4.txt
1 Bobot Hadiah Optimal : 52
2 Isi dari Buffer Optimal : 1C 7A 7A BD
3 Koordinat dari Setiap Token :
4 1, 1
5 1, 5
6 4, 5
7 4, 1
8
9
10 Waktu eksekusi program: 21.010160446166992 ms
11
```

Analisis : Dari pengujian di atas, penulis menyadari bahwa tidak hanya ukuran matriks yang memiliki pengaruh terhadap waktu eksekusi dari penggunaan algoritma *brute force*, tetapi jumlah ukuran buffer juga mempengaruhi waktu eksekusi. Hal ini lumrah terjadi karena semakin besar ukuran buffer nya, semakin banyak juga langkah yang harus dilakukan oleh program untuk memeriksa kemungkinan sekuens yang berarti semakin kompleks juga program yang akan dijalankan. Jadi, selain ukuran matriks, terdapat ukuran buffer yang juga mempengaruhi kecepatan kerja dari algoritma *brute force* yang digunakan pengguna.

4. Kasus Uji 4

Masukan file test5.txt :

```
test > test5.txt
1 7
2 10 9
3 DE EF HI AB GH BC CD DE CD CD
4 BC CD DE EF DE BC AB BC DE CD
5 BC DE CD CD EF BC HI BC EF EF
6 HI AB HI EF DE AB DE HI HI BC
7 BC BC HI CD AB CD BC GH DE HI
8 GH BC GH AB CD EF CD HI BC CD
9 AB HI DE BC GH AB EF BC DE AB
10 DE AB GH GH GH DE GH CD CD EF
11 DE AB EF DE HI CD EF GH CD DE
12 5
13 AB EF DE BC
14 5
15 CD AB HI GH EF
16 95
17 DE GH HI CD
18 21
19 BC GH AB DE HI CD
20 -36
21 AB DE GH
22 55
```

Solusi dari test5.txt :

```
test > solution > solution5.txt
1 Bobot Hadiah Optimal : 95
2 Isi dari Buffer Optimal : DE BC CD AB HI GH EF
3 Koordinat dari Setiap Token :
4 1, 1
5 1, 2
6 2, 2
7 2, 4
8 1, 4
9 1, 6
10 6, 6
11
12
13 Waktu eksekusi program: 34466.12310409546 ms
14
```

Analisis : Dari pengujian di atas, dapat dibuktikan bahwa masukan dengan ukuran matriks yang besar dan ukuran buffer yang besar akan sangat membutuhkan waktu yang cukup lama dibandingkan dengan ukuran matriks yang kecil dan ukuran buffer yang kecil. Hal

ini dapat dibuktikan dengan waktu eksekusi yang digunakan dalam proses penyelesaiannya membutuhkan 34466 ms (dibulatkan).

5. Kasus Uji 5

Masukan file test6.txt :

```
test > test6.txt
1 5
2 7 6
3 BD BD BD BD BD BD BD
4 BD BD BD BD BD BD BD
5 BD BD BD BD BD BD BD
6 BD BD BD BD BD BD BD
7 BD BD BD BD BD BD BD
8 BD BD BD BD BD BD BD
9 2
10 BD E9 1C
11 15
12 BD 7A BD
13 20
```

Solusi dari test6.txt :

```
test > solution > solution6.txt
1 Tidak ada solusi yang memenuhi
2
3 Waktu eksekusi program: 27.006864547729492 ms
4
```

Analisis : Dari pengujian di atas, jika tidak ada sekuens yang memenuhi, maka program akan mengeluarkan bobot hadiah optimalnya menjadi 0 dan akan mengeluarkan prompt seperti di atas.

4.1.2.2 Masukan dari Keyboard

1. Kasus Uji 1

```

Pilih Masukan
1. Masukan dari keyboard
2. Masukan dari file
Pilihan Anda: 1
Masukkan jumlah token unik: 5
Masukkan token yang boleh digunakan untuk mengisi matrix (Contoh : BD 1C 7A): BD E9 1C 7A 55
Masukkan ukuran buffer: 7
Masukkan ukuran matriks (Format : row column):
6 6
Masukkan jumlah sekuens: 3
Masukkan ukuran maksimal sekuens: 4
Matriks acak:
E9 E9 55 7A E9 BD
E9 E9 55 55 1C 55
BD 55 55 BD 55 1C
7A BD E9 55 E9 1C
BD 7A 55 E9 7A 55
E9 BD 7A 7A BD E9
sequences 1: ['7A', 'BD'], Reward: -29
sequences 2: ['BD', '7A', '1C', 'E9'], Reward: 79
sequences 3: ['55', '1C'], Reward: 61
-----Please Wait! Your Solution is On The Way-----

```

```

Bobot Hadiah Optimal : 140
Isi dari Buffer Optimal : E9 BD 7A 1C E9 55 1C
Koordinat dari Setiap Token :
1, 1
1, 5
5, 5
5, 2
2, 2
2, 3
6, 3

Waktu eksekusi program: 528.8519859313965 ms

Apakah Anda ingin menyimpan solusi(y/n): N

```

2. Kasus Uji 2

```

Pilih Masukan
1. Masukan dari keyboard
2. Masukan dari file
Pilihan Anda: 1
Masukkan jumlah token unik: 5
Masukkan token yang boleh digunakan untuk mengisi matrix (Contoh : BD 1C 7A): BD E9 1C 7A
Jumlah token yang dimasukkan tidak sesuai dengan jumlah token unik
Masukkan token yang boleh digunakan untuk mengisi matrix (Contoh : BD 1C 7A): BD E9 1C 7A 55
Masukkan ukuran buffer: 7
Masukkan ukuran matriks (Format : row column):
6 6
Masukkan jumlah sekuens: 8
Masukkan ukuran maksimal sekuens: 8
Matriks acak:
E9 1C BD 1C 1C 1C
7A 55 1C BD 1C 7A
BD BD 1C 55 BD 55
BD 55 E9 E9 1C 1C
55 55 BD E9 E9 BD
E9 BD 1C 7A 7A 7A
sequences 1: ['E9', '7A', '1C', '55', 'BD'], Reward: -65
sequences 2: ['1C', 'BD'], Reward: -68
sequences 3: ['55', 'BD', '7A', '1C', 'E9'], Reward: 37
sequences 4: ['7A', '1C', 'E9', 'BD', '55'], Reward: -83
sequences 5: ['55', '1C', '7A'], Reward: 19
sequences 6: ['E9', '55', 'BD', '7A', '1C'], Reward: 18
sequences 7: ['7A', 'BD', '1C', 'E9', '55'], Reward: -18
sequences 8: ['1C', 'BD', '55', '7A', 'E9'], Reward: -30
-----Please Wait! Your Solution is On The Way-----

```

```

Bobot Hadiah Optimal : 55
Isi dari Buffer Optimal : E9 55 BD 7A 1C E9 BD
Koordinat dari Setiap Token :
1, 1
1, 5
6, 5
6, 2
3, 2
3, 4
1, 4

Waktu eksekusi program: 859.4975471496582 ms

Apakah Anda ingin menyimpan solusi(y/n): n

```

Kasus di atas merupakan uji coba saat pengguna memasukkan token yang tidak sesuai jumlahnya dengan jumlah token unik yang pengguna input sebelumnya.

3. Kasus Uji 3

```

Pilih Masukan
1. Masukan dari keyboard
2. Masukan dari file
Pilihan Anda: 1
Masukkan jumlah token unik: 3
Masukkan token yang boleh digunakan untuk mengisi matrix (Contoh : BD 1C 7A): AA BB CC
Masukkan ukuran buffer: 3
Masukkan ukuran matriks (Format : row column):
2 2
Masukkan jumlah sekuens: 2
Masukkan ukuran maksimal sekuens: 2
Matriks acak:
BB AA
AA AA
sequences 1: ['AA', 'CC'], Reward: -64
sequences 2: ['CC', 'BB'], Reward: -12
-----Please Wait! Your Solution is On The Way-----
Tidak ada solusi yang memenuhi

Waktu eksekusi program: 0.0 ms

```

4. Kasus Uji 4

```

Pilih Masukan
1. Masukan dari keyboard
2. Masukan dari file
Pilihan Anda: 1
Masukkan jumlah token unik: 6
Masukkan token yang boleh digunakan untuk mengisi matrix (Contoh : BD 1C 7A): 11 22 33 44 55 66
Masukkan ukuran buffer: 4
Masukkan ukuran matriks (Format : row column):
7 8
Masukkan jumlah sekuens: 3
Masukkan ukuran maksimal sekuens: 4
Matriks acak:
44 44 22 66 33 55 33 66
11 33 44 11 11 33 11 55
33 33 66 11 55 44 22 22
22 66 44 66 66 22 44 33
22 66 33 22 44 55 33 44
11 44 11 44 33 22 44 66
55 66 33 11 55 66 33 33
sequences 1: ['11', '66', '44', '33'], Reward: 50
sequences 2: ['55', '33'], Reward: 99
sequences 3: ['11', '55', '44', '22'], Reward: -38
-----Please Wait! Your Solution is On The Way-----

```

```

Bobot Hadiah Optimal : 99
Isi dari Buffer Optimal : 44 11 55 33
Koordinat dari Setiap Token :
1, 1
1, 2
8, 2
8, 4

Waktu eksekusi program: 10.072469711303711 ms

Apakah Anda ingin menyimpan solusi(y/n): y
Masukkan nama berkas untuk menyimpan solusi (Format : filename.txt) : solutionCLI.txt

Solusi telah disimpan dalam berkas: ..\test\solution\solutionCLI.txt

```

Kasus di atas merupakan kasus uji apabila pengguna yang menginput masukan dari keyboard ingin melakukan penyimpanan solusi dalam bentuk file.

BAB 5 KESIMPULAN

Berdasarkan analisis yang telah dilakukan terhadap implementasi algoritma brute force pada penyelesaian permainan Cyberpunk 2077 Breach Protocol, dapat disimpulkan bahwa algoritma ini berhasil mencapai bobot hadiah optimal yang sesuai dengan yang diharapkan dalam spesifikasi permainan. Meskipun demikian, hasilnya dapat berbeda dalam urutan sekuens token yang dihasilkan, karena pendekatan brute force mencari kemungkinan sekuens terakhir dengan bobot maksimal. Waktu eksekusi algoritma brute force menunjukkan performa yang baik untuk ukuran masukan yang kecil, namun menjadi tidak praktis untuk ukuran masukan yang besar karena kompleksitasnya yang meningkat secara eksponensial. Selain itu, ukuran buffer juga mempengaruhi waktu eksekusi, dengan ukuran yang lebih besar menyebabkan peningkatan kompleksitas dan waktu yang diperlukan oleh algoritma.

Meskipun algoritma brute force cenderung kurang efisien dan praktis untuk masukan berukuran besar, ia tetap menjadi pilihan yang valid untuk masukan dengan ukuran yang kecil dan untuk masalah-masalah sederhana yang memerlukan solusi yang langsung dan jelas. Namun, untuk masalah yang lebih kompleks dan memerlukan efisiensi waktu yang tinggi, diperlukan pendekatan algoritma yang lebih canggih dan pintar. Dengan demikian, pemahaman tentang kekuatan dan batasan algoritma brute force membantu dalam pemilihan dan penggunaan yang tepat dari algoritma tersebut dalam menyelesaikan berbagai masalah permasalahan.

BAB 6 LAMPIRAN

6.1. Github

https://github.com/Maharanish/Tucil1_13522134

6.2. Tabel Pemeriksaan

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	✓	
2. Program berhasil dijalankan	✓	
3. Program dapat membaca masukan berkas .txt	✓	
4. Program dapat menghasilkan masukan secara acak	✓	
5. Solusi yang diberikan program optimal	✓	
6. Program dapat menyimpan solusi dalam berkas .txt	✓	
7. Program memiliki GUI		✓

DAFTAR PUSTAKA

Munir, Rinaldi. (2024). Algoritma Brute Force (Bagian 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 9 Februari 2024).

GreenBerg, Frank. (2020). How to Breach Protocol in Cyberpunk 2077. <https://gamerjournalist.com/how-to-breach-protocol-in-cyberpunk-2077/#:~:text=In%20Cyberpunk%202077%2C%20attempting%20to%20Breach%20Protocol%20will,equipment%20to%20apply%20RAM%20buffs%20to%20your%20quickhacks> (diakses pada 9 Februari 2024).

Toms, Ollie. (2020). How does Breach Protocol work in Cyberpunk 2077. <https://www.rockpapershotgun.com/cyberpunk-2077-hacking-minigame-breach-protocol-explained> (diakses pada 10 Februari 2024).

Cyberpunk 2077 Hacking Minigame Solver (cyberpunk-hacker.com) (diakses pada 8 Februari 2024).