Journal of Theoretical and Applied Information Technology
15th October 2020. Vol.98. No 19

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

COMPARISON OF STEEPEST ASCENT HILL CLIMBING ALGORITHM AND BEST FIRST SEARCH ALGORITHM IN DETERMINING THE SHORTEST ROUTE FOR MEDAN TOURISM

[1]DIAN RACHMAWATI, [2]HANDRIZAL, [3]RIZALI AHMAD BATUBARA

[1,2,3]Departemen Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi,
Universitas Sumatera Utara
Jl. Universitas No. 9-A, Kampus USU, Medan 20155, Indonesia
E-mail: [1]dian.rachmawati@usu.ac.id

ABSTRACT

Medan is the capital of North Sumatra and one of the major cities in Indonesia. In this city there are several tourist destinations that can be visited by the people. When visitors want to travel by visiting several destinations, it will be difficult for visitors to determine the shortest route that can be traversed. And to find out the shortest travel route, a system is required to find the shortest route that visitors can travel through. In this study developed a route search system for the nearest tourist trip to Medan. In this system there is a menu to find routes by providing several tourist destinations in Medan. And in support of the shortest route search process on this system used the Steep Ascent Hill-Climbing algorithm and the Best First Search algorithm. The comparison process will then be carried out between the two algorithms, both from running time and the distance weight generated by the algorithm used. After both algorithms are implemented into the system and tested both algorithms, then obtained the complexity of the Steepest Ascent Hill Climbing algorithm is $\theta(n^5)$, and the running time value is 59.3856 ms. On the contrary, obtained the Best First $\theta(n^3)$ algorithm, and the running time value is 54.6669 ms. And from the testing of both algorithms, it can be concluded that the search using the Best First Search algorithm produces a better running time value than the Steepest Ascent Hill Climbing algorithm.
Keywords: *Best First Search, Steepest Ascent Hill Climbing, The Shortest Route*

1. INTRODUCTION

Medan is one of the major cities in North Sumatra and is one of Indonesia's major cities. In the city, several tourist attractions can be visited, and visitors usually take a tour to visit several tourist destinations. Due to the many tourist destinations that you want to visit will make it difficult for visitors to determine the route you want to pass to save time in making your tour. To make it easier for visitors to travel, it requires a system to determine the shortest route that connects several tourist destinations that you want to visit.

Traveling Salesman Problem is one method to solve problems in finding the shortest route or minimum distance for visitors from the starting point to another point and right back to the initial departure point. The Traveling Salesman problem is applied using weighted graphs where the weight is the distance between one point and another. In determining the shortest route, we need an algorithm in other research to mention that as an intelligent search optimization technique, a genetic algorithm has some internal weaknesses such as premature convergence and low computation efficiency [13]. Therefore this research explores another heuristic method, the Steepest Ascent Hill Climbing Algorithm, and the Best First Search Algorithm, to solve the TSP problem.

The Best First Search algorithm in finding the best route begins by representing the problem in the form of a complete graph, then looking for each node's value to another node. If the best node is found, it is used as the current state to be re-selected until all points are visited. After all, vertices have passed, they must return to the initial node.

The Steepest Ascent Hill Climbing algorithm in finding the shortest route compares the heuristic values of all trajectories and chooses the smallest heuristic value to be used as a new state. Then the new state is re-evaluated until the best route is obtained.

From the background that has been explained, the writer wants to compare the two algorithms by

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

taking a research topic entitled "Comparative Analysis of the Steepest Ascent Hill Climbing Algorithm and the Best First Search Algorithm in Determining the Shortest Route for Medan Tourism."

2. GRAPH

A graph is a pair (V, E), where V is a set of vertices, and E is a set of edges. The Graph can be interpreted as a collection of points connected by lines or, in other words, each side on E connects two points from V[6]. The following are some types of graphs.

a. Simple Graf
   A simple Graph is a graph that does not have a direction and a double edge and a ring (loop).
b. Not a Simple Graph
   Not a Simple Graph is a graph that has two directions and a loop.
c. Directed Graph
   Directional graphs are graphs in which each side only has a direction and does not have two opposite sides.
d. Non-directed Graph
   A non-directed Graph is a graph that has no direction on each side.

3. TRAVELLING SALESMAN PROBLEM

The Traveling Salesman Problem problem is when Salesmen, when traveling, must stop in all cities, and the Salesman can only visit the same city exactly once [2]. And the journey of a salesman must end right in the city of departure[8]. The aim is to obtain the route of travel with the minimum distance and cost [7].

In brief, the characteristics of the Traveling Salesman Problem are as follows [5]:

- The journey begins and ends in the same city
- each city can only be visited once
- The trip is complete when all cities have been visited

The example of the Traveling Salesman Problem and the shortest route from all points through the A-B-D-F-E-C-A route weights 29.

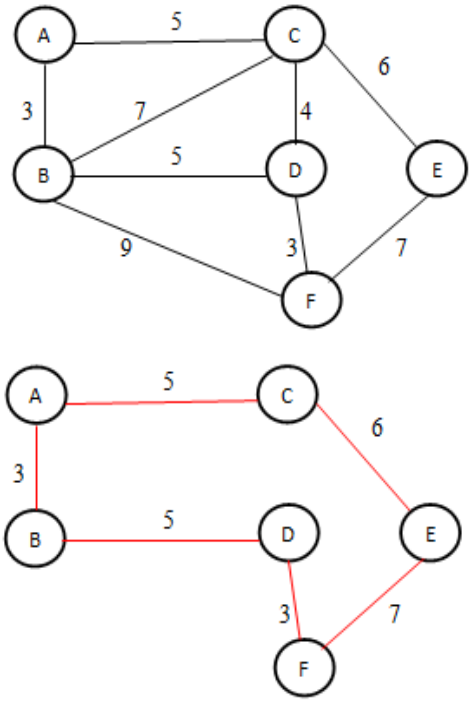


Figure 1. Example Of Travelling Salesman Problem

4. STEEPEST ASCENT HILL CLIMBING ALGORITHM

Steepest Ascent Hill Climbing is an algorithm method that is widely used for optimization problems. One application is to find the shortest route by minimizing the value of the existing optimization function [3].

The Steepest Ascent Hill Climbing method when determining the next state is to compare the value of the current state with the successors connected to it so that the next state is the successor that is the best or most close to the goal [3].

The workings of the Steepest Ascent Hill Climbing algorithm are as follows, do these steps until you get a goal state or until there are no more operators (states) that you want to use [12]:

1. Select a free operator to use as the current state. Make this operator search for a new state.
2. Evaluate the current state and get all successors to be made as to the next state. Then evaluate each successor and rate it.
3. If there is one successor that is found to have the best value of the current stat, then use the successor to be the new current state. Perform operations 1 to 3 until the current state is found to be the same as the goal state.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The following is an example of the application of Steepest Ascent Hill Climbing to the Traveling Salesman Problem.

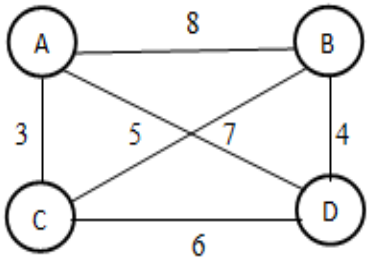


Figure 2. Illustration Of The Problem

From the example of the problem obtained, the search process results by using the Steepest Ascent Hill Climbing algorithm. The free participant used as a current state is ABCDA, with a weight of 26.

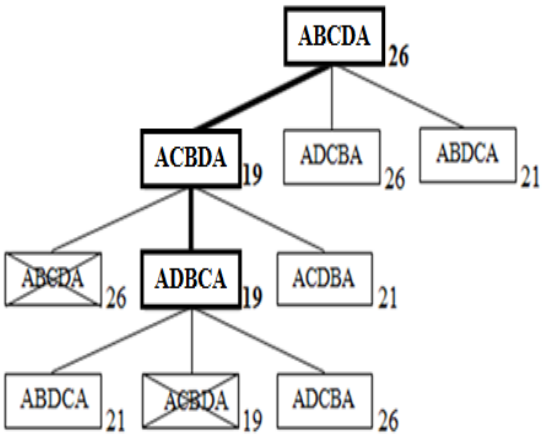


Figure 3. Steepest Ascent Hill Climbing Algorithm Searching Process

- Iteration 1
  Each successor was evaluated, and the ACBDA successor was better than ABCDA (current state). ACBDA is made the current state for the next iteration with a weight of 19.
- Iteration 2
  Every successor is evaluated, and the same ADBCA successor is obtained with ACBDA (current state). ADBCA is made the current state for the next iteration with a weight of 19. Whereas the previous successor obtained was not evaluated anymore.
- Iteration 3
  An evaluation was performed on each successor, not finding the best value from ADBCA (current state). Then the ADBCA route is the optimum solution for this example with a weight of 19. Whereas the previous successor that was obtained was no longer evaluated.

5. BEST FIRST SEARCH ALGORITHM

The Best First Search algorithm is assumed in the Process of finding the closest route, where the best value is a value that is more minimum than the results of comparison with other values [4]. After the point that has the best value is found and is not a goal state, checking is carried out at the next point at the same depth to obtain the best value. Then the point is opened and checked if it is the goal stat. If the goal state has not been received, do the same Process at the next point [6].

To implement this algorithm, two indexes are needed, OPEN and CLOSE. Where the OPEN index is functioning to accommodate points that have not been evaluated, and the CLOSE index serves to accommodate points that have been compared. The workings of the Best First Search algorithm are as follows [6].

1. Determine the starting point and place it in the OPEN queue.
2. Repeat until the goal state is obtained or until the OPEN queue is empty.
   i. Choose the best point from inside OPEN.
   ii. If that point is the same as the goal state, the Process stops. If not, enter that point in the CLOSE queue.
3. Generate point 3 For each work point:
   i. If that point has never been raised, then evaluate that point and enter OPEN.
   ii. If the point has been raised previously, change the path to a new route that is better than the previous way. Remove that point from the OPEN queue.

The following is an example of the application of Best First Search to the Traveling Salesman Problem.

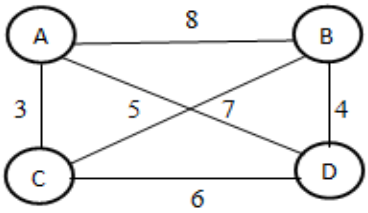


Figure 4. Illustration Of The Problem

The example questions obtained the search process results by using the Best First Search algorithm, where the starting point of the journey starts from point A.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

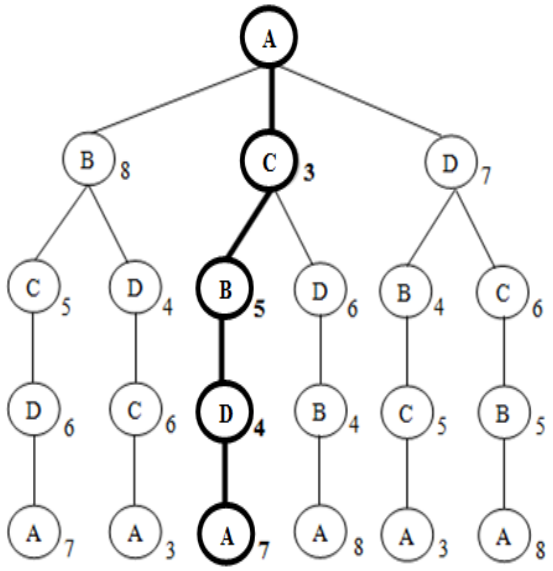


Figure 5. Best First Search Algorithm Searching Process

- Iteration 1
  The initial state A is entered into the OPEN queue and is made the best node because only point A is in the OPEN queue, and then it is moved to the CLOSE queue. Then all successors are raised, namely B, C, and D. Then, it produces the OPEN queue = [B, C, D] and the CLOSE queue = [A]. The selection process is carried out, and the best point is C = 3 and selected as the best node. Then C is moved to the CLOSE queue, and an A-C path is obtained. Then the queue CLOSE = [A, C] is obtained and the OPEN queue = [B, D].
- Iteration 2
  Then a new CLOSE queue is obtained, point C, then the successor is raised, then B and D. The selection process is carried out, and the best point is B = 5 and selected as the best node. Then B is moved to the CLOSE queue, and route A-C-B is obtained. Then obtained CLOSE = [A, C, B] and in the OPEN queue = [D].
- Iteration 3
  A new value is obtained from the CLOSE queue, which is point B, then the successor is raised, then obtained is D. Because D is the only OPEN queue, D = 6 is chosen as the best node. From this step, the queue OPEN = [] and the queue CLOSE = [A, C, B, D] are obtained. The iteration process will stop because all points in the OPEN queue have been visited, and the OPEN queue has been empty. Because the trip must end at the initial point, the search tree uses the Best First Search algorithm to get the best route is the point A-C-B-D-A, with a weight of 3 + 5 + 4 + 7 = 19.

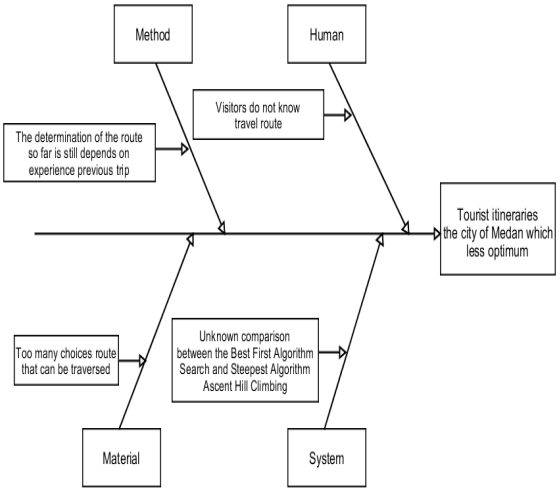
6. ALGORITHM COMPLEXITY

An algorithm is not only judged to be accurate but also felt by its level of efficiency. An algorithm's efficiency is measured by how much time and memory space is needed to run the algorithm [11]. An efficient algorithm is an algorithm that minimizes the need for time and space and the memory used [10]. Asymptotic notation states the complexity of an algorithm and is divided into three, namely:

1. Big-O (O)
2. Big Theta (Θ)
3. Big Omega (Ω)

7. ISHIKAWA DIAGRAM

The problem in this study is how one can determine the shortest route for a tour in the city of Medan. As for identifying the problem, and Ishikawa Diagram is needed. Ishikawa diagram is a diagram that shows the specific cause and effect of a problem. Using this diagram, we can identify and describe an issue along with the cause and effect of the problem.



.

Figure 6. Ishikawa Diagram

8. GENERAL ARCHITECTURE

General architecture represents a system that describes the Process, flow, and interaction between components in order.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

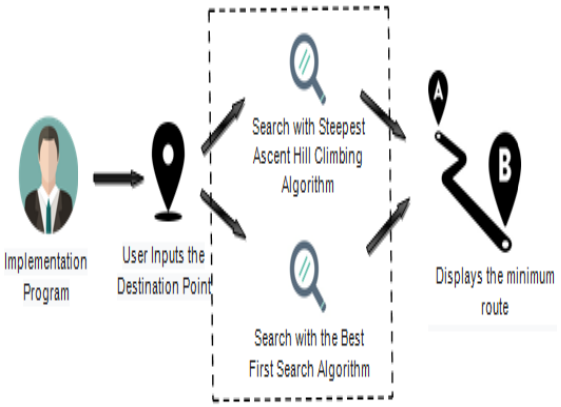


Figure 7. General Architecture

1. start the program implementation
2. the user enters the destination point that you want to visit
3. the user selects one of the algorithms
4. the system displays the best route that can be passed by the user

9. FLOWCHART

1. System Flowchart

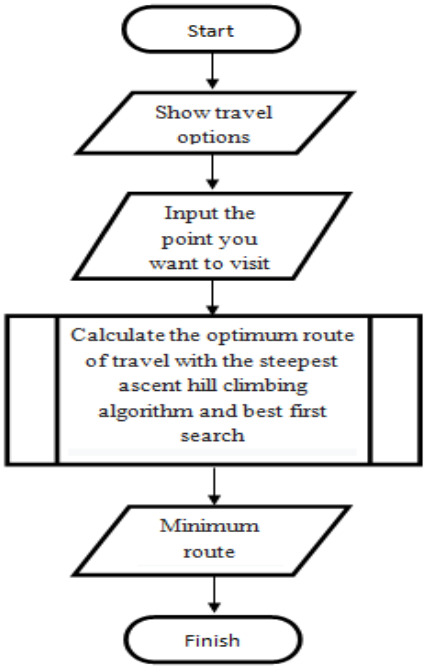


Figure 8. Flowchart of the System

2. Steepest Ascent Hill Climbing Flowchart

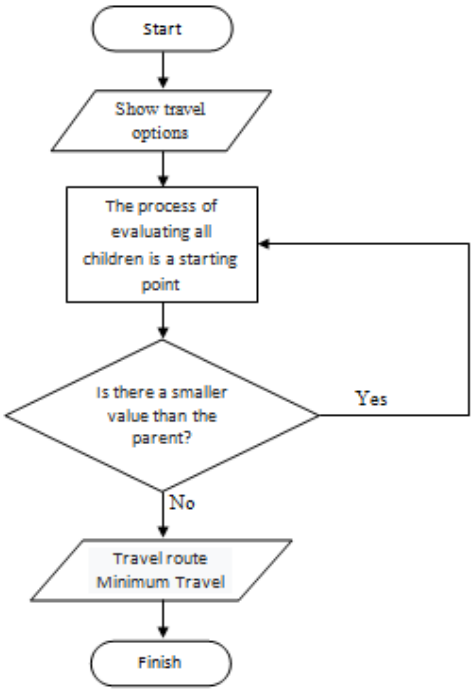


Figure 9. Flowchart of Steepest Ascent Hill Climbing

3. Best First Search Flowchart

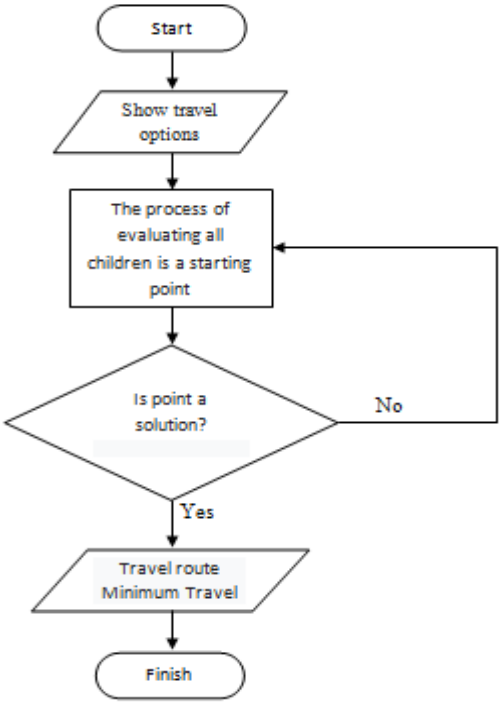


Figure 10. Flowchart of Best First Search

Journal of Theoretical and Applied Information Technology
15th October 2020. Vol.98. No 19

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

10. PROCESS PAGE

Display Page Process is a page that functions as a place for users to search for input tour routes. In the Process, the Pages section contains a list of tourist destinations that can be visited in accordance with the user's choice [1]. After the user chooses the goal he wants to visit, then the user selects one of the two algorithm keys provided. After the algorithm is selected, the system will display the best route, mileage, and running time algorithm used. The process page display can be seen in the following picture.

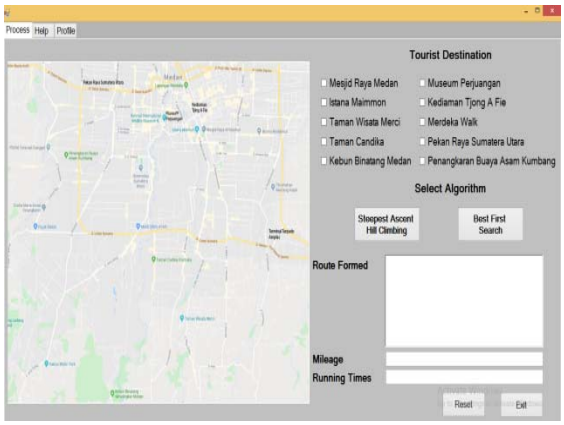


Figure 11. The Application

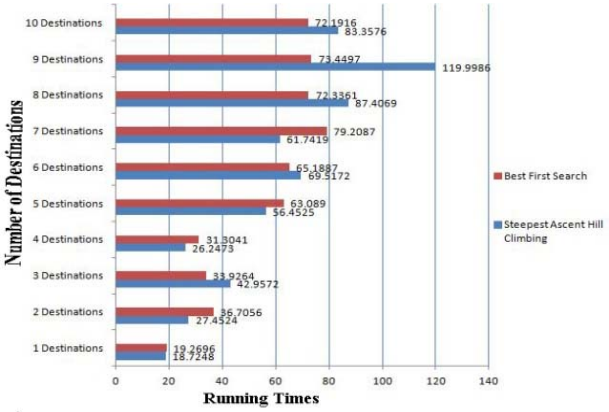
11. TEST RESULTS AND COMPARISON OF STEEPEST ASCENT HILL CLIMBING ALGORITHM AND BEST FIRST SEARCH ALGORITHM

Table 1. The Comparison Result

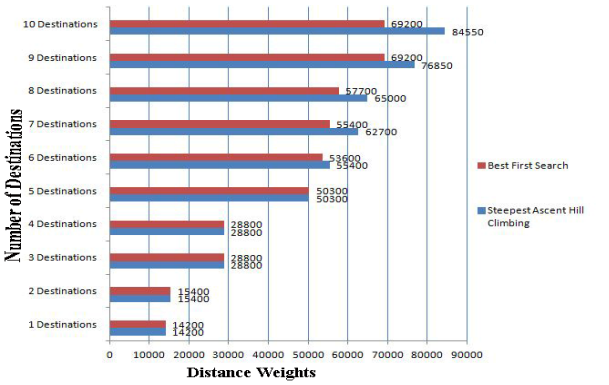
No	Selected destination	Comparison of Routes Formed		Comparison of distances (meters)		Comparison Running time (ms)	
		Steepest Ascent Hill Climbing	Best First Search	Steepest Ascent Hill Climbing	Best First Search	Steepest Ascent Hill Climbing	Best First Search
1	B	A-B-A	A-B-A	14200	14200	18.7248	19.2696
2	B-C	A-B-C-A	A-B-C-A	15400	15400	27.4524	36.7056
3	B-C-D	A-B-C-D-A	A-B-C-D-A	28800	28800	42.9572	33.9264
4	B-C-D-E	A-B-C-E-D-A	A-B-C-E-D-A	28800	28800	26.2473	31.3041
5	B-C-D-E-F	A-B-C-E-D-F-A	A-B-C-E-D-F-A	50300	50300	56.4525	63.089
6	B-C-D-E-F-G	A-B-C-G-E-D-F-A	A-B-C-G-E-D-F-A	55400	53600	69.5172	65.1887
7	B-C-D-E-F-G-H	A-B-C-E-D-F-H-G-A	A-B-C-H-G-E-D-F-A	62700	55400	61.7419	79.2087
8	B-C-D-E-F-G-H-I	A-B-C-E-D-F-I-H-G-A	A-B-C-H-I-G-E-D-F-A	65000	57700	87.4069	72.3361
9	B-C-D-E-F-G-H-I-J	A-B-C-E-D-F-G-H-I-J-A	A-B-C-H-I-G-J-E-D-F-A	76850	69200	119.9986	73.4497
10	B-C-D-E-F-G-H-I-J-K	A-B-C-E-D-F-G-H-I-J-K-A	A-B-C-H-I-G-J-K-E-D-F-A	84550	69200	83.3576	72.1916
Average				**48200**	**44260**	**59.3856**	**54.6669**

After the testing process, a graph that displays the running time algorithm comparison to the number of destinations chosen by the user, the running time comparison graph of the two algorithms is presented in the picture.

i. Running time comparison to the number of destinations.



ii. Comparison of distance weights to the number of destinations



Journal of Theoretical and Applied Information Technology
15th October 2020. Vol.98. No 19

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Figure 12. The Comparison Chart

12. STEEPEST ASCENT HILL CLIMBING ALGORITHM COMPLEXITY

No	Program Code	C	#	C*#
1	List<int> pilihan = new List<int>();	C1	1	C1
2	pilihan.Add(0);	C1	1	C1
3	if (checkBox1.Checked) pilihan.Add(1);	C2	1	C2
4	if (checkBox2.Checked) pilihan.Add(2);	C2	1	C2
5	if (checkBox3.Checked) pilihan.Add(3);	C2	1	C2
6	if (checkBox4.Checked) pilihan.Add(4);	C2	1	C2
7	if (checkBox5.Checked) pilihan.Add(5);	C2	1	C2
8	if (checkBox6.Checked) pilihan.Add(6);	C2	1	C2
9	if (checkBox7.Checked) pilihan.Add(7);	C2	1	C2
10	if (checkBox8.Checked) pilihan.Add(8);	C2	1	C2
11	if (checkBox9.Checked) pilihan.Add(9);	C2	1	C2
12	if (checkBox10.Checked) pilihan.Add(10);	C2	1	C2
13	int pilih = 0;	C3	1	C3
14	int[] sahc = new int[pilihan.Count];	C4	1	C4
15	foreach (int haha in pilihan) {	C5	n	nC5
16	sahc[pilih] = haha;	C4	n	nC4
17	pilih++; }	C3	n	nC3
18	fungsisahc(sahc);	C6	1	C6
19	void fungsisahc(int[] tujuan) {	C7	1	C7
20	int permutasi = 0;	C8	1	C8
21	for (int brp = 0; brp < tujuan.Length - 1; brp++) {	C9	n	nC9
22	for (int i = 1; i < tujuan.Length - 1; i++) {	C9	n^2	n^2C9
23	for (int j = i; j < tujuan.Length - 1; j++) { }	C9	n^2	n^2C9
24	permutasi++; } }	C8	n^2	n^2C8
25	permutasi = permutasi / 2 + 1;	C8	1	C8
26	string[] child = new string[permutasi];	C10	1	C10
27	int[] jarake = new int[permutasi];	C11	1	C11
28	int[] abisal = new	C12	1	C12
29	int[tujuan.Length];	C13	1	C13
30	int tanker = 0;	C14	1	C14
31	foreach (int x in tujuan) {	C5	n	nC5
32	abisal[tanker] = x;	C12	n	nC12
33	tanker++; }	C14	n	nC14
34	for (int brp = 0; brp < 2; brp++) {	C9	n	nC9
35	int temper = 1;	C15	n	nC15
36	for (int i = 1; i < tujuan.Length - 1; i++) {	C9	n^2	n^2C9
37	for (int j = i; j < tujuan.Length - 1; j++) {	C9	n^2	n^2C9
38	int invoker = 0;	C16	n^2	n^2C16
39	foreach (int xy in abisal) {	C5	n^4	n^4C5
40	tujuan[invoker] = xy;	C16	n^4	n^4C16
41	invoker++; }	C16	n^4	n^4C16
42	invoker = 0;	C16	n^2	n^2C16
43	string abysperl = "";	C17	n^2	n^2C17
44	foreach (int gg in abisal) {	C5	n^4	n^4C5
45	abysperl += gg; }	C17	n^4	n^4C17
46	child[0] = abysperl;	C10	n^2	n^2C10
47	int costarica2 = 0;	C18	n^2	n^2C18
48	int temperatursuhu = 0;	C19	n^2	n^2C19
49	for (int m = 0; m < tujuan.Length - 1; m++) {	C9	n^4	n^4C9
50	foreach (jarak x in edge[tujuan[m]]) {	C5	n^5	n^5C5
51	if (x.hubungan == (tujuan[m + 1])) {	C2	n^5	n^5C2
52	costarica2 += x.cosjarak; } }	C18	n^5	n^5C18
53	if (tujuan[m + 1] == tujuan[tujuan.Length - 1]) {	C2	n^4	n^4C2
54	foreach (jarak x in edge[tujuan[m + 1]]) {	C5	n^5	n^5C5
55	if (x.hubungan == 0) {	C2	n^5	n^5C2
56	temperatursuhu = x.cosjarak; } } } }	C19	n^5	n^5C19
57	jarake[0] = costarica2 + temperatursuhu;	C11	n^4	n^4C11
58	int temp = 0;	C20	n^4	n^4C20
59	temp = tujuan[i];	C7	n^4	n^4C7
60	tujuan[i] = tujuan[j + 1];	C7	n^4	n^4C7
61	tujuan[j + 1] = temp;	C7	n^4	n^4C7
62	string xamp = "";	C21	n^5	n^5C21
63	foreach (int x in tujuan) {	C5	n^6	n^6C5
64	xamp += x + ""; }	C21	n^6	n^6C21
65	int costarica = 0;	C22	n^5	n^5C22
66	for (int m = 0; m < tujuan.Length - 1; m++) {	C9	n^6	n^6C9
67	foreach (jarak x in edge[tujuan[m]]) {	C5	n^5	n^5C5
68	if (x.hubungan == (tujuan[m + 1])) {	C2	n^5	n^5C2
69	costarica += x.cosjarak; } }	C22	n^5	n^5C22
70	if (tujuan[m + 1] == tujuan[tujuan.Length - 1]) {	C2	n^6	n^6C2
71	foreach (jarak x in edge[tujuan[m + 1]]) {	C5	n^5	n^5C5
72	if (x.hubungan == 0) {	C2	n^5	n^5C2
73	temperatursuhu = x.cosjarak; } } } }	C19	n^5	n^5C19
74	child[temper] = xamp;	C10	n^2	n^2C10
75	jarake[temper] = costarica + temperatursuhu;	C11	n^2	n^2C11
76	if (temper == permutasi - 1){	C2	n^2	n^2C2
77	for (int kale = 0; kale < permutasi; kale++) {	C9	n^4	n^4C9
78	for (int jale = 0; jale < permutasi - 1; jale++) {	C9	n^4	n^4C9
79	if (jarake[jale] > jarake[jale + 1]) {	C2	n^4	n^4C2
80	int tenk = 0;	C23	n^2	n^2C23
81	tenk = jarake[jale];	C23	n^2	n^2C23
82	jarake[jale] = jarake[jale + 1];	C11	n^2	n^2C11
83	jarake[jale + 1] = tenk;	C11	n^2	n^2C11
84	string tenk2 = "";	C24	n^2	n^2C24
85	tenk2 = child[jale];	C24	n^3	n^3C24
86	child[jale] = child[jale + 1];	C10	n^3	n^3C10
87	child[jale + 1] = tenk2; } } }	C10	n^3	n^3C10
88	int blank = 0;	C25	n^2	n^2C25
89	for (int tes = 0; tes < child[0].Length - 1; tes++) {	C9	n^4	n^4C9
90	if (child[0][tes] == '1' && child[0][tes + 1] == '0') {	C2	n^4	n^4C2
91	abisal[tes] = 10;	C12	n^4	n^4C12
92	blank++; }	C25	n^4	n^4C25
93	else {	C26	n^4	n^4C26
94	abisal[tes] = Convert.ToInt32(new string(child[0][blank], 1)); }	C12	n^4	n^4C12
95	blank++; } }	C25	n^4	n^4C25
96	temper++;	C15	n^2	n^2C15

From the calculation of the complexity of the Steepest Ascent Hill Climbing algorithm in the table, the value of T (n) is obtained as follows:

$$T(n) = 2C_1 + 10C_2 + C_3 + C_4 + 2nC_5 + nC_4 + nC_3 + C_6 + C_7 + 2C_8 + 2nC_9 + 2n^2C_9 + 2n^3C_9 + n^2C_8 + C_{10} + C_{11} + C_{12} + C_{13} + C_{14} + nC_{12} + nC_{14} + nC_{15} + 3n^4C_5 + 2n^4C_{16} + 2n^3C_{16} + n^3C_{17} + n^4C_{17} + 2n^3C_{10} + n^3C_{18} + n^3C_{19} + 4n^4C_9 + 4n^5C_5 + 6n^5C_2 + 5n^4C_8 + 4n^4C_2 + 2n^5C_{19} + 2n^3C_{11} + n^3C_2 + n^3C_{20} + 3n^3C_7 + n^3C_{21} + n^4C_{21} +$$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$n^3C_{22} + n^5C_{22} + n^5C_9 + 2n^5C_{23} + 2n^5C_{11} + 2n^5C_{24} + 2n^5C_{10} + n^3C_{25} + 2n^4C_{12} + 2n^4C_{25} + n^4C_{26} + n^3C_{15}$

$T(n) = (2C_1 + 10C_2 + C_3 + C_4 + C_6 + C_7 + 2C_8 + C_{10} + C_{11} + C_{12} + C_{13} + C_{14} + C_{22} + C_{25})\ n^0 + (C_3 + C_4 + 2C_5 + +C8 + 2C_9 + C_{12} + C_{14} + C_{15})\ n^1 + (2C_9)\ n^2 + (3C_7 + 2C_9 + 2C_{10} + 2C_{11} + C_{15} + 2C_{16} + C_{17} + C_{18} + C_{19} + C_{20} + C_{21} + C_{22})\ n^3 + (4C_2 + 3C_5 + 5C_8 + 4C_9 + 2C_{12} + 2C_{16} + C_{17} + C_{21} + C_{24} + 2C_{25})\ n^4 + (7C_2 + 4C_5 + C_9 + 2C_{11} + 2C_{10} + 2C_{19} + C_{22} + 2C_{23} + 2C_{24})\ n^5$

$T(n) = \theta(n^5)$

In the calculation of complexity based on the table, the Steepest Ascent Hill Climbing algorithm complexity value is $\theta(n^5)$.

13. BEST FIRST SEARCH ALGORITHM COMPLEXITY

No	Program Code	C	#	C*#
1	List<int> pilihan2 = new List<int>();	C1	1	C1
2	pilihan2.Add(0);	C1	1	C1
3	if (checkBox1.Checked) pilihan2.Add(1);	C2	1	C2
4	if (checkBox2.Checked) pilihan2.Add(2);	C2	1	C2
5	if (checkBox3.Checked) pilihan2.Add(3);	C2	1	C2
6	if (checkBox4.Checked) pilihan2.Add(4);	C2	1	C2
7	if (checkBox5.Checked) pilihan2.Add(5);	C2	1	C2
8	if (checkBox6.Checked) pilihan2.Add(6);	C2	1	C2
9	if (checkBox7.Checked) pilihan2.Add(7);	C2	1	C2
10	if (checkBox8.Checked) pilihan2.Add(8);	C2	1	C2
11	if (checkBox9.Checked) pilihan2.Add(9);	C2	1	C2
12	if (checkBox10.Checked) pilihan2.Add(10);	C2	1	C2
13	int pilih = 0;	C3	1	C3
14	int[] bfs = new int[pilihan2.Count];	C4	1	C4
15	foreach (int haha in pilihan2){	C5	n	nC5
16	bfs[pilih] = haha;	C4	n	nC4
17	pilih++; }	C3	n	nC3
18	fungsibfs(bfs);	C6	1	C6
19	void fungsibfs(int[] tujuss){	C6	1	C6
20	int[] tuju = new int[tujuss.Length];	C7	1	C7
21	int mlers = 0;	C8	1	C8
22	foreach (int ml in tujuss) {	C5	n	nC5
23	tuju[mlers] = ml;	C7	n	nC7
24	mlers++; }	C8	n	nC8
25	int[] tujas = new int[tuju.Length];	C9	n	nC9
26	tujas[0] = tuju[0];	C9	n	nC9
27	int zaxuang = 0;	C10	n	nC10
28	for (int males = 0; males < tuju.Length; males++){	C11	n	nC11
29	int[] cos = new int[tuju.Length];	C12	n	C12
30	foreach (jarak x in edge[tuju[0]]) {	C5	n^2	n^2C5
31	tuju[0] = 11;	C7	n^2	n^2C7
32	for (int i = 0; i < tuju.Length; i++) {	C11	n^2	n^2C11
33	if (x.hubungan == tuju[i]) {	C2	n^2	n^2C2
34	cos[i] = x.cosjarak;}}}	C12	n^2	n^2C12
35	for (int i = 0; i < cos.Length; i++) {	C11	n^2	n^2C11
36	for (int j = 0; j < cos.Length - 1; j++) {	C11	n^2	n^2C11
37	if (cos[j] > cos[j + 1]) {	C2	n^2	n^2C2
38	int temp = cos[j];	C13	n^2	n^2C13
39	cos[j] = cos[j + 1];	C12	n^2	n^2C12
40	cos[j + 1] = temp;	C12	n^2	n^2C12
41	int temp2 = tuju[j];	C14	n^2	n^2C14
42	tuju[j] = tuju[j + 1];	C7	n^2	n^2C7
43	tuju[j + 1] = temp2; }}}	C7	n^2	n^2C7
44	if (zaxuang != tuju.Length - 1) {	C2	n	nC2
45	tujas[zaxuang] = tuju[0];	C9	n	nC9
46	zaxuang++; }}	C10	n	nC10
47	int[] finalresult = new int[tuju.Length];	C15	n	nC15
48	for (int i = 1; i < tuju.Length; i++) {	C11	n	nC11
49	finalresult[i] = tujas[i - 1]; }	C15	n	nC15
50	finalresult[0] = 0;	C15	1	C15
51	int[] arsytemp = new int[tuju.Length - 1];	C16	1	C16
52	for (int i = 1; i < tuju.Length; i++) {	C11	n	nC11
53	arsytemp[i - 1] = finalresult[i]; }	C16	n	nC16

From the calculation of the complexity of the Best First Search algorithm in the table, the value of T (n) is obtained as follows:

$T(n) = 2C_1 + 10C_2 + C_3 + C_4 + 2nC_5 + nC_4 + nC_3 + 2C_6 + C_7 + C_8 + nC_7 + nC_8 + 3nC_9 + 2nC_{10} + 3nC_{11} + C_{12} + n^2C_5 + n^2C_7 + 2n^3C_{11} + 2n^3C_2 + 3n^3C_{12} + n^2C_{11} + n^3C_{13} + n^3C_{14} + n^3C_7 + nC_2 + C_{15} + 2nC_{15} + C_{16} + nC_{16}$

$T(n) = (2C_1 + 10C_2 + C_3 + C_4 + C_6 + C_7 + C_8 + C_{12} + C_{15} + C_{16})\ n^0 + (2C_5 + C_4 + C_3 + C_7 + C_8 + 2C_{10} + 3C_{11} + C_5 + C_2 + C_9 + C_{10} + 2C_{15} + C_{16})\ n^1 + (C_5 + C_7 + C_{11})\ n^2 + (3C_9 + 2C_{11} + 2C_2 + 3C_{12} + C_{13} + C_{14} + 2C_7)\ n^3$

$T(n) = \theta(n^3)$

In the calculation of complexity based on the table, the complexity of the Best First Search algorithm is $\theta(n^3)$.

14. CONCLUSION

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

This system can be used in finding the shortest route for travel in the city of Medan by using the Steepest Ascent Hill-Climbing algorithm and the Best first search algorithm. From the calculation of the complexity of the Steepest Ascent Hill Climbing algorithm, the complexity value θ (n^5) is obtained. For the Best First algorithm, the complexity value θ (n^3) is obtained. From the results of the Steepest Ascent Hill Climbing Algorithm got the average value of running time 59.3856 ms, whereas the Best First Search algorithm received an average running time value of 54.6669 ms. Based on the results of testing the two algorithms, it is known that the running time value of the Best first search algorithm is faster than the amount of the running time of the Steepest Ascent Hill Climbing algorithm. The Best First Search algorithm produces better distance weights than the Steepest Ascent Hill Climbing algorithm for some cases based on the chosen tourist destination. The number of tourist destinations visited in the system influences the running time value of the two algorithms used. The more number of tourist destinations you want to visit, the greater the chosen algorithm's running time value.

REFERENCES

[1] Abdillah Baraja. (2009). Implementasi Sistem Informasi Akademik Universitas Surakarta. *Jurnal Speeed*, *Volume 1(2),*10-19.

[2] Anca Elena Iordan. (2012). A Comparative Study Of Meta-Heuristics Methods For Travelling Salesman Problem. *Journal Of Engineering*, *3*, 243-246.

[3] Eka Vickraien Dangkua, Vincencius Gunawan dan Kusworo Adi. (2015). Penerapan Metode Hill Climbing Pada Sistem Informasi Geografis Untuk Mencari Lintasan Terpendek. *Jurnal Sistem Informasi Bisnis, 01*, 19-25.

[4] Faozi. (2016). *Penerapan Algoritma Recursive Best First Search Dalam Penyelesaian Traveling Salesman Problem di Pt. Bintang Service Management*. Skripsi. Universitas Negeri Semarang: Semarang

[5] Hari Santoso. (2017). Penyelesaian Traveling Salesman Problem (TSP) Menggunakan Algoritma Recursive Best First Search (RBFS). *Unnes Jpurnal of Mathematics, Volume 6(2),* 211-220.

[6] Muchammad Abrori dan Rike Nur Setiyani. (2015). Implementasi Algoritma Best First Search (BeFS) Pada Penyelesaian Traveling Salesman Problem (TSP). *Jurnal* Fourier, *Volume 4(2),* 93-111.

[7] Muhammad Irfan. (2017). Penyelesaian Travelling Salesman Problem (TSP) Menggunakan Algoritma Hill Climbing dan MATLAB. *Jurnal Matematika*, *Volume 16(2),*13-20.

[8] N.Sathya, Dr.A.Muthukumaravel. (2016). A Survey of Travelling Salesman Problem Using Heuristic Search Techniques. *International Journal of Innovative Research in Computer, Volume 4(1),* 847-851.

[9] Pemerintah Kota Medan Dinas Pariwisata. (2016). Destinasi Wisata Kota Medan di http://www.pariwisata.pemkomedan.go.id/ (Diakses 2 Februari 2019).

[10] Rismon Alexantro. (2017). *Analisis Perbandingan Algoritma Generate And Test Dengan Hill Climbing Pada Penyelesaian Traveling Salesman Problem Untuk Kunjungan Wisata Di Kabupaten Tapanuli Tengah.* Skripsi. Universitas Sumatera Utara: Medan

[11] Ritayani. (2016). Pengantar Algoritma dan Pemrograman. *Jurnal*. *Volume 4,* 72-79.

[12] Siby Abraham, Imre Kiss, Sugata Sanyal, Mukund Sanglikar. (2010). Steepest Ascent Hill Climbing For A Mathematical Problem. *Jurnal of Applied Management*.

[13] Hussain, A., Muhammad, Y.S. *Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator*. Complex Intell. Syst. 6, 1–14 (2020). https://doi.org/10.1007/s40747-019-0102-7