IMPLEMENTATION OF SIMPLIFIED MEMORY-BOUNDED A* ALGORITHM FOR SEARCHING WORD ON WORD SEARCH PUZZLE GAME

Asih Joko Purnomo[1]

[1] Program Studi Teknik Informatika
Fakultas Teknik dan Ilmu Komputer Universitas Komputer Indonesia
Jl. Dipatiukur No 112-116 Bandung - Indonesia
E-mail : jokopurnomoa@gmail.com

ABSTRACT

Word search puzzle game is a game to find the hidden words on the board game that it arranged in the form of a matrix. The words can be arranged horizontally, vertically or it can be arranged in more than one segment which are connected horizontally and vertically. Search word is compiled more than one line segment which has the same characteristics as pathfinding problems, so it requires a pathfinding algorithm to perform a search.

Simplified Memory-Bounded A (SMA*) Algorithm is one of the pathfinding algorithms that can be used to perform a word search on the word search puzzle game. SMA* algorithm has an advantage in uses of less memory, this thing because the memory usage is limited to a certain number of nodes.*

Based on test results that the longer of characters on the search terms so the search time will be longer and also memory usage will increase although. More nodes are available to search so it will be faster, wherein the percentage of search speed increase with adding nodes 100% it can be up to 21.99% than it does not add the nodes.

Keywords : *Pathfinding, Word Search Puzzle, Simplified Memory-Bounded A* (SMA*)*

1. INTRODUCTION

Word search puzzle games or word search game is a puzzle based game to find words that are arranged in a matrix form. Words can be arranged horizontally, vertically or arranged in more than one segment are connected horizontally and vertically. Completion of word search puzzle game is to find all the hidden word on the board in the form of a matrix. The problem is how the system can find all the words hidden in the puzzle that has been arranged in random either horizontally, vertically or arranged in more than one segment are connected horizontally and vertically.

Previously been research on comparative analysis Knuth-Morris-Pratt algorithm with Boyer-Moore algorithm on a word search puzzle game [1]. From these research it is known that the Boyer-Moore algorithm is efficient for searching horizontally and vertically, while the algorithm Knuth-Morris-Pratt is more efficient in the search diagonally. However, these algorithms can not be used in the completion of this game because the rules of search on the word that is composed not only horizontal, vertical and diagonal.

Therefore in this research used Simplified Memory-Bounded A* algorithms for searching word on word search puzzle game. This algorithms have been seeing a word search puzzle problem characteristics are similar to the problems in pathfinding. This algorithm is an algorithm that is often used on the shortest path search and a heuristic search algorithm. This algorithm has the advantage of requiring a smaller memory than the A* algorithm [2].

Based on the explanation that has been described above, it is expected that the solution to the word search puzzle game can be found and it is known that the selected algorithm is an effective algorithm to perform searches on the word game word search puzzles.

The purpose of which is expected to be reached in this research are:

1. Find out whether the Simplified Memory-Bounded A* algorithm can be used to find solutions on word search puzzle game.
2. Find out the performance on speed and memory usage of Simplified Memory-Bounded A* algorithm to searching words on word search puzzle game.

2. RESEARCH CONTENTS

2.1 Word Search Puzzle Game

Word search puzzle game known as word find game, which is famous for helping students to recognize words. At this game the letters of a word lies in the grid and usually has a square shape. To play this game, players find and mark all the words hidden in the grid. In the game of word search puzzles, hidden word list is provided. Usually a lot of related words easy for players to find. Registered word can be set in the horizontal direction, vertical or diagonal grid. The faster completion of each

level, the higher score will be obtained. In searching for the words, users read and memorize the words as they play games that help them learn the words and spelling, letter by letter, the puzzle [5]. Here is a picture of the word search puzzle game:



Image 1 Word Search Puzzle [5]

2.2 Heuristic Search Methods (Informed Search)

The word heuristic comes from a Greek verb, heuriskein, which means 'search' or 'find'. In the programming world, some people use the word heuristics as opposed to the word of algorithmic, wherein said heuristic is defined as 'a process that may be able to solve a problem but there is no guarantee that a solution is sought can always be found'. In studying the methods of this search, the word heuristic is defined as a function that gives a value in the form of cost estimates (estimates of a solution) [2].

The methods included in search techniques based on heuristic functions are: Generate and Test, Hill Climbing (Simple Hill Climbing dan Steepest-Ascent Hill Climbing), Simulated Annealing, Best First Search(Greedy Best First Search and A* with different variations, such as Simplified Memory-Bounded A*).

2.3 Simplified Memory-Bounded A* (SMA*) Algorithm

For certain problems, which limited the computer's memory, the A* algorithm might not be able to find a solution because it is not available memory to store the generated nodes. IDA* algorithm can be used for conditions such as IDA* only requires a little memory. However, one drawback is that the IDA* searches performed iteratively will take a long time because they have raised repeatedly node [2].

Contrary to IDA * only remember one f-limit, the algorithm SMA * given f-cost of each iteration until the number of nodes that exist in memory. Due to memory limitations in a certain amount, it can limit the search to only the nodes that can be reached from the root along a path that memory is still insufficient. Then restore a best among these routes that exist within the limits of the number of nodes. If the computer memory is only capable of storing 100 node, it can limit the search process until level 99.

In the SMA* algorithm there is a list that is used to manipulate the queue node sorted by f-cost. Here is the f-cost is the combined cost of the actual and the estimated cost, which is expressed mathematically in equation 2.1 as follows:

$$f(n) = g(n) + h(n) \qquad (2.1)$$

With:

n = current node
g(n) = cost from first node to *n* node along the search path
h(n) = estimated cost of the node *n* to the goal node (heuristic value)
f(n) = total cost from *n* node to goal node

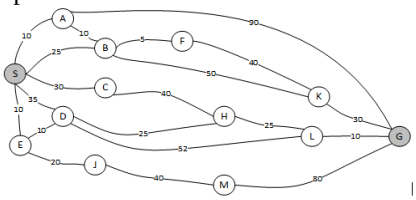
Here is the Simplified Memory-Bounded A* algorithm [2]:

function SMA*(masalah) returns solusi
inputs : masalah, sebuah masalah
local variables : Queue, antrian nodes yang terurut berdasarkan f-cost
Queue ← MAKE-QUEUE({MAKE-SIMPUL(INITIAL-STATE[masalah])})
loop do
if Queue kosong **then** return gagal
n ← simpul di Queue yang memiliki f-cost terkecil dan level terdalam
if GOAL-TEST(n) **then** return sukses
suk ← NEXT-SUCCESSOR(n)
if suk bukan goal dan levelnya sudah maksimum **then**
f(suk) ← INFINITE
else
f(suk) ← MAX(f(n), g(n) + h(n))
end
if semua suksesor dari n sudah dibangkitkan **then**
Ganti f-cost pada n dengan nilai f(suk) yang terkecil. Gantikan nilai f(suk) terkecil ini ke semua ancestors dari n(ayah, kakek, dan seterusnya keatas) kecuali ancestors yang memiliki f-cost lebih kecil daripada f(suk)terkecil itu.
if semua SUCCESSOR(n) sudah di memori **then**
Keluarkan n dari Queue (tetapi tidak dihapus secara fisik di memori)
if memori penuh **then**
if suk = Goal and f(suk) = f(start) **then** return sukses dan exit
Hapus simpul terburuk di dalam Queue yang memiliki f-cost terbesar dan level terdangkal.
Keluarkan simpul terburuk tersebut dari daftar suksesor

parent-nya.
Masukkan parent dari simpul terburuk tersebut ke Queue jika parent tersebut tidak ada di Queue.
end
insert suk in Queue

Image 2 Simplified Memory-Bounded A* Algorithm

For example there is the problem of finding the shortest path as shown below:



n	S	A	B	C	D	E	F	G	H	J	K	L	M
h(n)	80	80	60	70	85	74	70	0	40	100	30	20	70

Image 3 Path finding problem

Here are the steps to resolve the use Simplified Memory-Bounded A* algorithm:

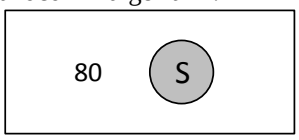


Image 4 Step 1 Path Finding

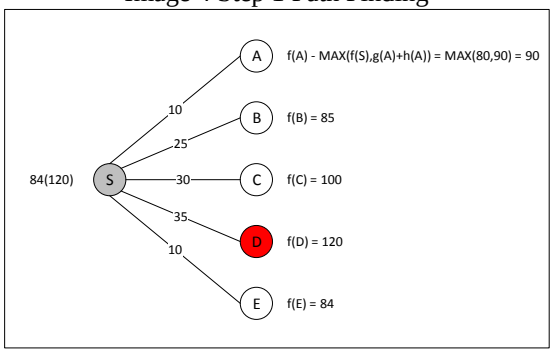


Image 5 Step 2 Path Finding

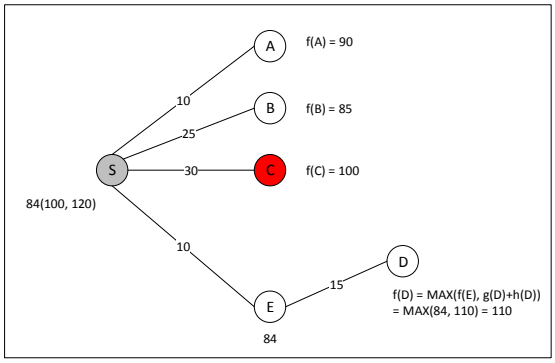


Image 6 Step 3a Path Finding

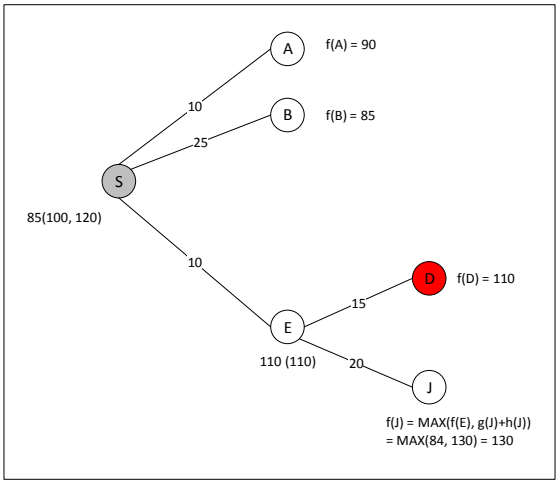


Image 7 Step 3b Path Finding

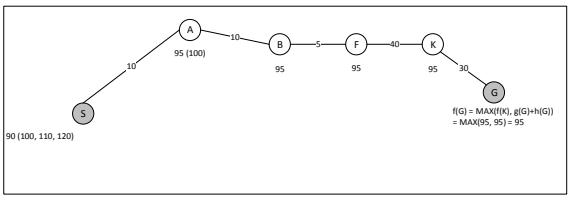


Image 8 Step 8 Path Finding (the final results search)

2.4 System Analysis

Word search puzzle game consists of two processes, namely word randomization and word searching. The randomization process said that fill the game board with the word of a random arrangement of random letters and fill in the blank matrix. The word search process is doing a word search on the game board that has been filled with word of a random arrangement. The following is a flowchart of a word search puzzle game:

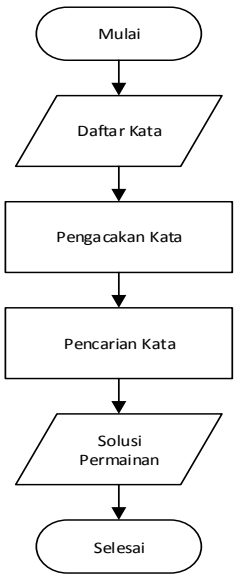


Image 9 Word Search Puzzle Flowchart

In the word search puzzle game there are a few rules that differ from one game to the other games. For the existing rules in this game are as follows:

a. Word search can be done horizontally and vertically, and can not diagonally.

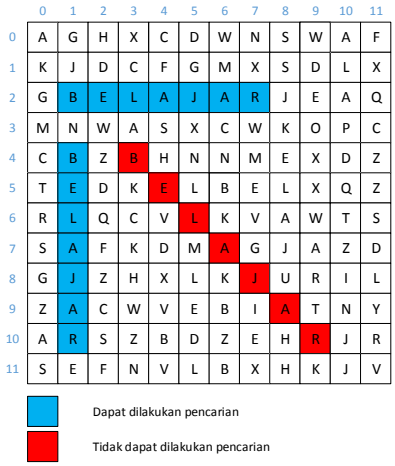


Image 10 Word Search Rules

b. Word search can also be performed by zigzagged up, down, left and right, but between the characters to one another to be connected horizontally or vertically.

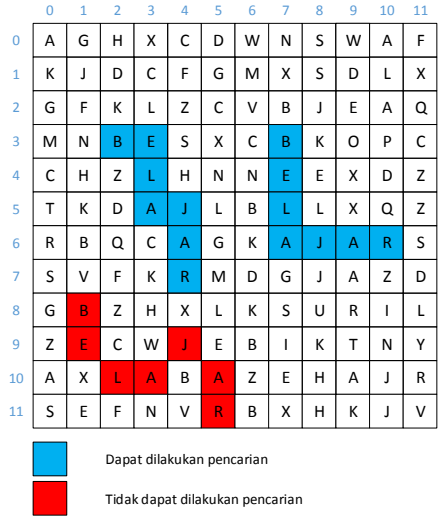


Image 11 Word Search Rules (2)

c. Searches can be started from anywhere between coordinates (0,0) to coordinates (19.19) in the matrix.

d. Length of characters on the search terms maximum 30 characters.

2.5 Problem Analysis

The problems that arise in the word search puzzle game is how the system can find all the words hidden in the matrix. The first is search first letter of word to be searched, and then searched a second letter by looking up, down, left and right which around the first letter. The method is repeated until the search terms found.

To perform a word search on the word search puzzle game can use path finding algorithms. There are some path finding algorithms that can be used. The algorithm used is Simplified Memory-Bounded A* algorithm. Simplified Memory-Bounded A* Algorithm (SMA*) is a development of the A* algorithm which overcomes a problem to do a word search on the word search puzzle game. This algorithm overcomes the weakness that is owned by the A* algorithm that large memory usage.

Therefore, to be studied to measure the effectiveness of the use of the SMA* algorithm is as follows:

1. The number of node generated and removed during the search process.
2. The execution time for finding a solution.

2.6 Word Search Puzzle Game Analysis

Analysis on the word search puzzle game conducted to see the effectiveness in terms of execution time and the number of nodes that are raised and removed. In implementing the algorithm SMA* will be built a prototype with a picture of a word search puzzle game is as follows:

1. The game consists of a matrix with a minimum size is 3 rows x 3 columns and 20 rows x 20 columns.
2. Implementation of the chosen algorithm is an SMA* algorithm.
3. The value of the input parameter is the maximum number of nodes.
4. The word searched randomly placed in the matrix.
5. Searching can only be done horizontally, vertically, and a combination of horizontal and vertical that is connected to more than one segment.

Game board on a word search puzzle made with a long-scale pixel size of 500 pixels wide by 500 pixels. In the game board are stored several grid size 25 pixels long and 25 pixels wide, obtained 20 grid of horizontal and 20 vertical grid. If the sum total of its grid to 400 grid.

The game board as shown below:

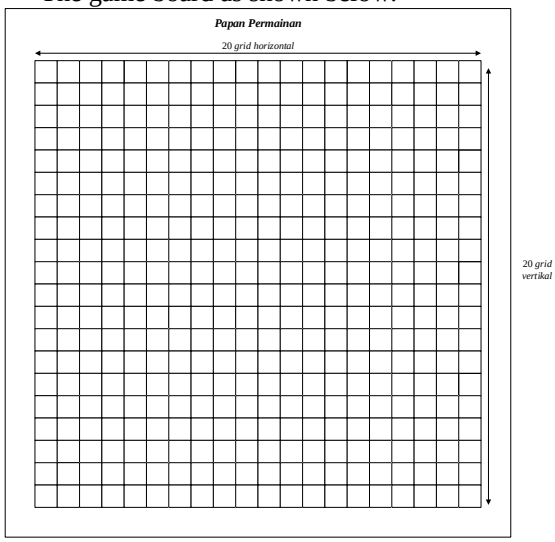


Image 12 Game Board

In the game, there are two major processes are performed, among others:

1. Word Randomization
2. Word Searching

Word randomization process is done by filling the matrix with the words being searched for according to the rules described above, then fill the empty matrix with random letters. Word search process performed by doing a word search in accordance with the rules described above, after a search is found, the stored coordinates of the search terms.

For more details, the following is a flowchart of the process of randomization words:

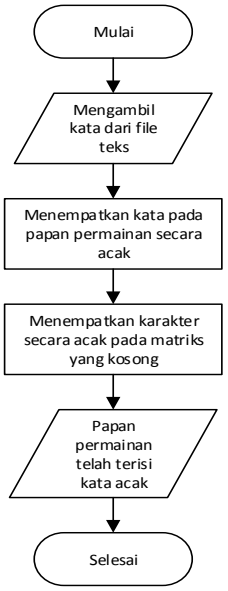


Image 13 Word Randomization Flowchart

In this study the process will be discussed more in is the process of searching word using the Simplified Memory-Bounded A* algorithm. In the searching word process there are several steps, among others, as in the following flowchart:



Image 14 Searching Word Flowchart

2.7 Input Analysis

Input data analysis the required in the SMA* algorithm is the position of the first letter of the word you are looking (initial state), the search terms (goal state). The position of the first letter searched sequentially from the first column and the first row to the last column and last row or until the position is found. In addition, the search SMA* algorithm, the search needed to calculate the value of *f (n)*.

Word search performed on the board game size 20x20. Board games have filled the search terms are arranged randomly. The game board is filled with word with a maximum length of 30 characters. The word that will be inserted into the matrix is stored in a text file, and drawn at random to be inserted into the matrix.

After the placement of words on the game board, then the game board has been filled with word are arranged randomly. Game board that has been filled words as shown below:

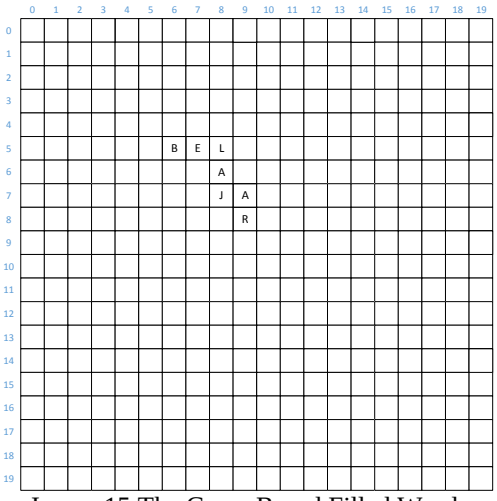


Image 15 The Game Board Filled Word

Then after the placement of the word, the next process is to fill the empty matrix with random letters. After matrix blank is random letters, then the result is as shown below:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	A	G	H	X	C	D	W	N	S	W	A	F	G	P	O	I	T	F	V	S
1	K	J	D	C	F	G	M	X	S	D	L	X	K	Q	Y	T	J	E	R	R
2	G	H	J	D	R	E	D	H	J	E	A	Q	D	F	R	E	F	J	J	M
3	M	N	W	A	S	X	C	W	K	O	P	C	V	F	G	X	G	H	G	N
4	C	V	Z	N	H	N	N	M	E	X	D	Z	F	Q	I	U	Q	T	D	Y
5	T	Y	D	K	D	L	B	E	L	X	Q	Z	F	R	Y	K	Z	X	D	S
6	R	W	Q	C	V	N	K	V	A	W	T	S	D	K	D	L	O	P	E	Q
7	S	J	F	K	D	M	K	G	J	A	Z	D	A	X	M	X	N	C	B	V
8	G	N	Z	H	X	L	K	S	U	R	I	L	U	L	D	Y	H	P	Y	R
9	Z	Q	C	W	V	E	B	I	R	T	N	Y	M	U	I	T	O	P	S	A
10	A	W	S	Z	B	D	Z	E	H	H	J	R	L	J	B	N	J	X	K	L
11	S	E	F	N	V	L	B	X	H	K	J	V	X	B	F	T	V	N	F	D
12	D	G	S	Z	C	N	M	R	X	X	J	V	B	B	K	N	M	X	Z	J
13	D	V	X	B	C	R	C	C	X	Z	Z	C	V	N	B	T	Y	L	M	U
14	V	M	B	B	N	Z	A	N	Z	Z	D	T	T	M	V	N	K	C	X	G
15	F	N	Q	R	X	W	R	E	V	R	T	T	F	U	D	S	I	J	O	A
16	C	X	V	C	B	B	Z	X	N	W	T	W	T	M	T	X	E	Z	R	H
17	H	N	V	X	Z	C	L	L	B	B	M	N	L	K	M	L	L	J	C	P
18	A	B	Z	N	D	H	X	M	J	I	C	O	O	L	P	V	X	N	C	M
19	Z	A	B	S	N	D	M	Z	F	N	Q	G	W	E	H	R	T	J	Y	K

Image 16 The Game Board Filled Random Letter

2.8 Algorithm Analysis

Algorithm analysis performed in this study is to analyze the workings of the SMA* (Simplified Memory-Bounded A*) algorithm to the problem of word search puzzles to word searches. This section will explain the process manually to generate output.

In the SMA* (Simplified Memory-Bounded A*) algorithm to search the *f-cost* calculation for each node generated. To calculate the *f-cost* mathematically expressed in the following equation 2.1:

$$f(n) = g(n) + h(n)$$

With:

n = current node
g(n) = cost from first node to *n* node along the search path
h(n) = estimated cost of the node *n* to the goal node (heuristic value)
f(n) = total cost from *n* node to goal node

To calculate the value heuristic, then searched using the total of letters in the search terms reduced total correct letters of the search. Heuristic value calculation is as follows:

h(n) = Total letter word searched - total letters correctly

The steps of manually searching SMA* is as follows:

a. Select the node with the least cost and the deepest level
b. Generate the next successor of *n*

suk node = *NEXT-SUCCESOR(n)*

c. Calculate the *f-cost* of a successor who has been raised

f(suk) = MAX(f(n), g(n) + h(n))

d. If all the node selected successor has been raised, then replace the *f-cost n* with the smallest value *f (suk)*.
e. If all *n* successor is already in memory, then remove *n* from Queue (but not deleted physically in memory).
f. If the memory is full, then remove the worst in the Queue node that has the greatest f-cost and shallowest level. Remove it from the worst node successor to its parent's worst Enter the parent of the node to the queue if the parent is not in Queue.
g. Insert the successor node to Queue

With:

n = current node
suk = successor node from *n*
f(suk) = *f-cost n* successor node
MAX = maximum value
NEXT-SUCCESSOR(n) = next successor node from *n*

For instance in this case will searched the word "BELAJAR" on the matrix which has been arranged at random. The first process is done is to find the position of the first letter of words to be searched. To perform these searches, the search must be done sequentially from column 0 and row 0. Here the **B** character of a search performed on the matrix. The possibility of the letter **B** in the matrix will be more than one, so it will take the letter was first discovered. When the position is found, the search will be performed with the initial position of node is the position that has been found. For example in the case of the maximum number of node is 8 node.

Here is an example of a matrix with the first character of the word have been found:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	A	G	H	X	C	D	W	N	S	W	A	F	G	P	O	I	T	F	V	S
1	K	J	D	C	F	G	M	X	S	D	L	X	K	Q	Y	T	J	E	R	R
2	G	H	J	D	R	E	D	H	J	E	A	Q	D	F	R	E	F	J	J	M
3	M	N	W	A	S	X	C	W	K	O	P	C	V	F	G	X	G	H	G	N
4	C	V	Z	N	H	N	N	M	E	X	D	Z	F	Q	I	U	Q	T	D	Y
5	T	Y	D	K	D	L	B	E	L	X	Q	Z	F	R	Y	K	Z	X	D	S
6	R	W	Q	C	V	N	K	V	A	W	T	S	D	K	D	L	O	P	E	Q
7	S	J	F	K	D	M	K	G	J	A	Z	D	A	X	M	X	N	C	B	V
8	G	N	Z	H	X	L	K	S	U	R	I	L	U	L	D	Y	H	P	Y	R
9	Z	Q	C	W	V	E	B	I	R	T	N	Y	M	U	I	T	O	P	S	A
10	A	W	S	Z	B	D	Z	E	H	H	J	R	L	J	B	N	J	X	K	L
11	S	E	F	N	V	L	B	X	H	K	J	V	X	B	F	T	V	N	F	D
12	D	G	S	Z	C	N	M	R	X	X	J	V	B	B	K	N	M	X	Z	J
13	D	V	X	B	C	R	C	C	X	Z	Z	C	V	N	B	T	Y	L	M	U
14	V	M	B	B	N	Z	A	N	Z	Z	D	T	T	M	V	N	K	C	X	G
15	F	N	Q	R	X	W	R	E	V	R	T	T	F	U	D	S	I	J	O	A
16	C	X	V	C	B	B	Z	X	N	W	T	W	T	M	T	X	E	Z	R	H
17	H	N	V	X	Z	C	L	L	B	B	M	N	L	K	M	L	L	J	C	P
18	A	B	Z	N	D	H	X	M	J	I	C	O	O	L	P	V	X	N	C	M
19	Z	A	B	S	N	D	M	Z	F	N	Q	G	W	E	H	R	T	J	Y	K

Keterangan :
B Simpul Awal

Image 17 Initial Conditions of Searching Words

After the first letter is found, the search performed using the SMA* algorithm with measures such as the following:

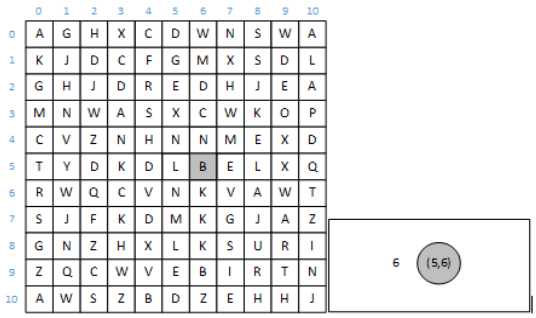


Image 18 Step 1 Searching Word

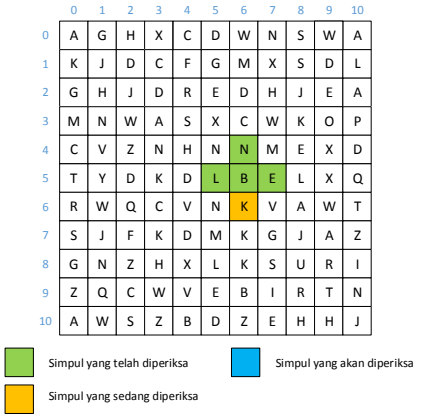


Image 19 Step 2 Searching Word Matrix

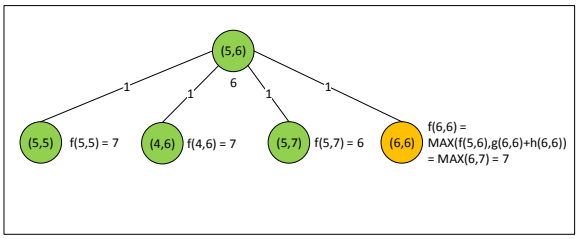


Image 20 Step 2 Searching Word

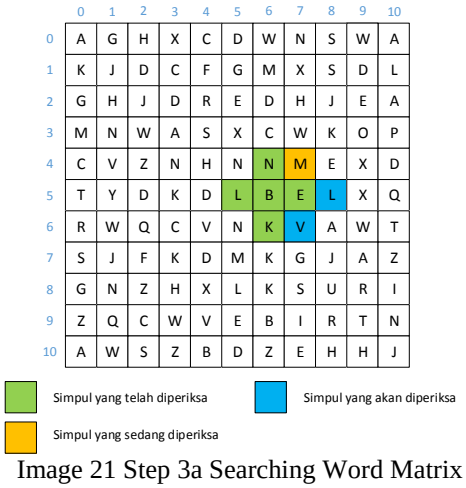


Image 21 Step 3a Searching Word Matrix

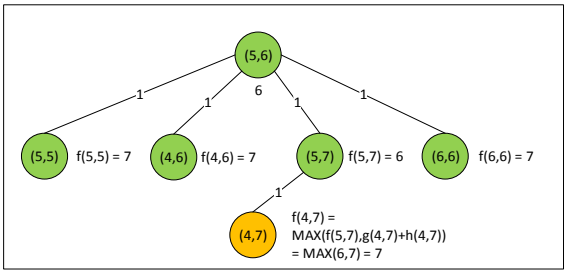


Image 22 Step 3a Searching Word

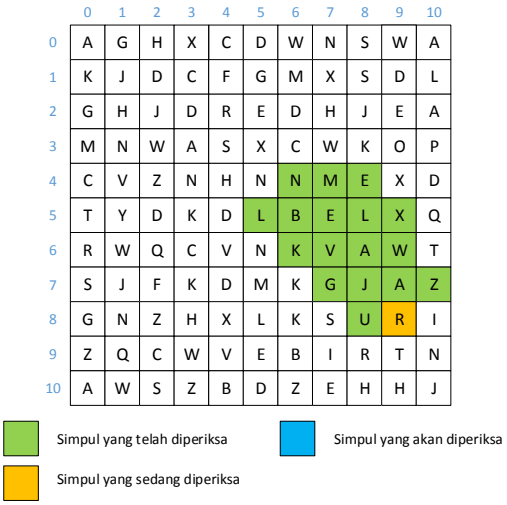


Image 23 Step 7b Searching Word Matrix

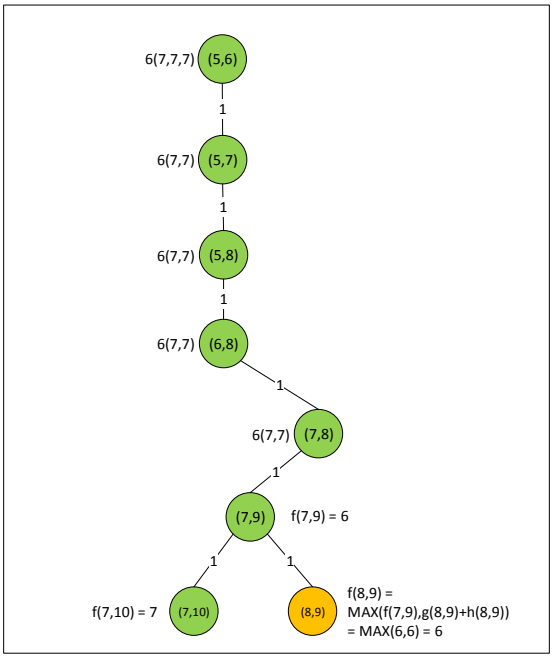


Image 24 Step 7b Searching Word (Solution)

In the picture above shows the search has been completed, the path found is **(5,6) – (5,7) – (5,8) – (6,8) – (7,8) – (7,9) – (8,9)**. So if these nodes are combined will formed word “BELAJAR”.

2.9 Functional Requirements Analysis

Functional requirements analysis aims to analyze the process to be applied in the system and explains the necessary requirements for this application to run properly. Functional requirements analysis include user requirements, modeling using the Unified Modeling Language (UML), interface design, and the design method.

2.9.1. Use Case Diagram

Use case diagram illustrates the functions of a system from a user perspective, so as to facilitate in formulating the requirements specification system to be built and developed. From the analysis of the game is built, then the use case diagram as shown below:

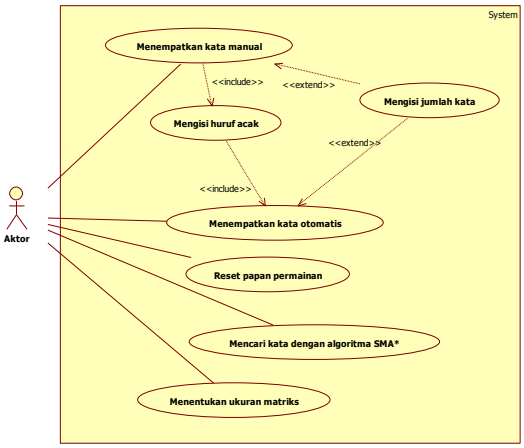


Image 25 Use Case Diagram

2.9.2. Activity Diagram

Activity diagram illustrates various activities in the system designed. Activity diagrams can also illustrate the parallel processes that may occur in several executions. The following is an activity diagram look for searching words with the SMA* algorithm:

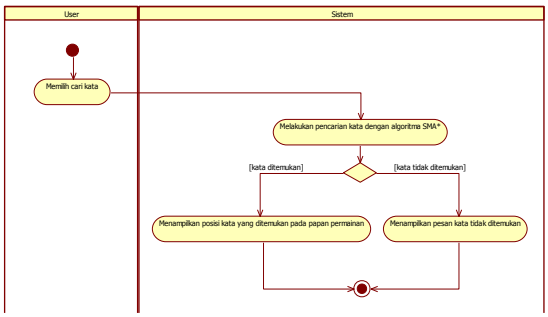


Image 26 Searching Word With SMA* Algorihtm Activity Diagram

2.9.3. Class Diagram

Class diagram is a diagram illustrating the structure of the defining aspects of the system of classes that are made to the system. Class diagram of the game is as shown below:

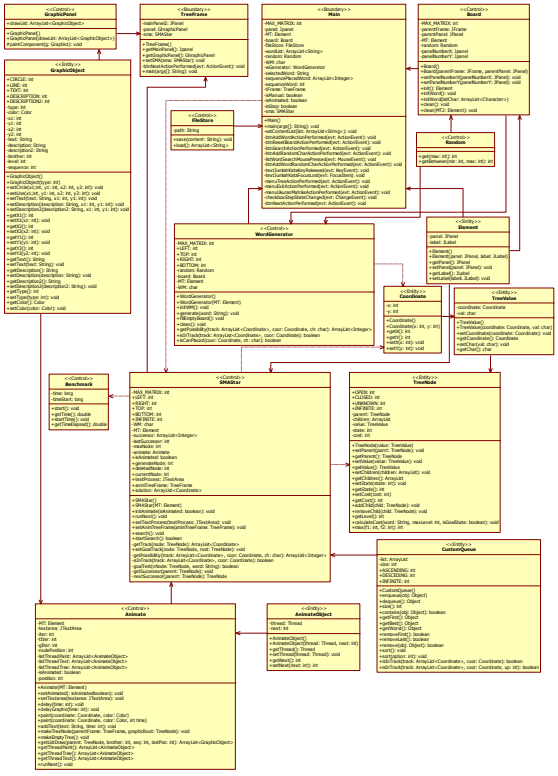


Image 27 Class Diagram

2.10 Interface Implementation

The main interface is the initial view which is testing the display when the user determine the position of the word to be searched by the Simplified Memory-Bounded A* algorithm. In this interface consists of a game board, a list of words, the search process, and the other button. View of the main interface can be seen in the following figure:

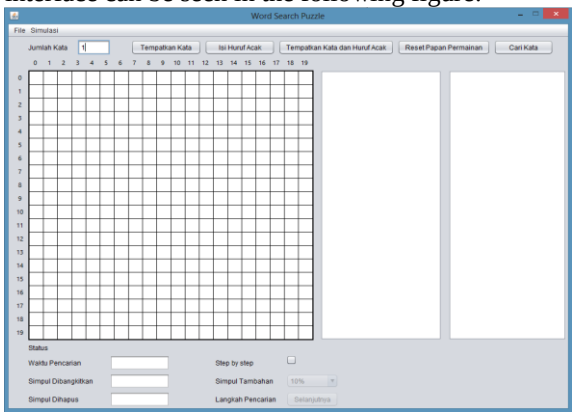


Image 28 Main Interface

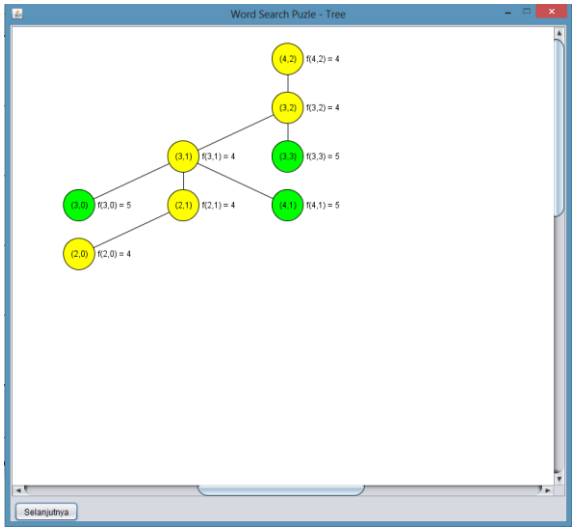


Image 29 Tree Search Interface

3 CLOSING

3.1 Conclusion

Based on research on implementation of Simplified Memory-Bounded A* algorithm to searching words on the word search puzzle game, it can be concluded:

1. Simplified Memory-Bounded A* algorithm can be used to perform a word search on the word search puzzle game.
2. Based on test results that longer of characters on the search terms so the search time will be longer and also memory usage will be increase. More nodes are available to search so it will be faster, wherein the percentage of search speed increase with adding nodes 100% it can be up to 21.99% than it does not add the nodes.

3.2 Suggestions

The suggestions that can be given to further development, among others things, are as follows:

1. Determine the percentage of node additions are optimal for each case of search.
2. The addition of the word list can be done automatically by the system by taking words of Indonesian dictionary.

DAFTAR PUSTAKA

[1] A. Rojali, "Analisis Perbandingan Algoritma Knuthmorris-Pratt dengan Algoritma Boyer-Moore Pada Permainana Word Search Puzzle," *Skripsi Teknik Informatika, Universitas Komputer Indonesia*, 2014.

[2] Suyanto, Artificial Intelligence Searching, Reasoning, Planning dan Learning, Bandung: Informatika Bandung, 2014.

[3] M. Nazir, Metodologi Penelitian, Bogor: Ghalia Indonesia, 2005.

[4] R. S. Pressman, Rekayasa Perangkat Lunak Pendekatan Praktis, Yogyakarta: Andi, 2012.

[5] A. Sukstrienwong and P. Vongsumedh, "Software Development of Word Search Game on Smart Phones in English Vocabulary Learning," *International Conference on Education and Modern Educational Technologies*, 2013.

[6] S. Russel and P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall International, Inc, 1995.

[7] B. Hariyanto, Esensi-esensi Bahasa Pemrograman Java, Bandung: Informatika, 2014.

[8] A. Nugroho, Rational Rose Untuk Pemodelan Berorientasi Objek, Bandung: Informatika, 2005.

[9] Munawar, Pemodelan Visual dengan UML, Yogyakarta: Graha Ilmu, 2005.

[10] K. Hamilton and R. Miles, Learning UML 2.0, Sebastopol: O'Reilly Media, Inc., 2006.

[11] Ayuliana, "Teknik Pengujian Perangkat Lunak," *Jurnal Teknik Informatika, Universitas Gunadarma*, 2009.