Modification of Common Web Crawler Using Greedy Best First Search and Its Implementation in The Wiki Game Solver

Steve Bezalel Iman Gustaman - 13518018
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
stevebezalel215@gmail.com 13518018@std.stei.itb.ac.id

***Abstract*—The Wiki Game is a simple game designed to work with Wikipedia which only requires a web browser and Internet access. The goal of this game is to navigate from a randomly selected Wikipedia article to another in least number of clicks or the least time. In this paper, the author will discuss about the concepts of web crawler, greedy best first search algorithm, web crawler modification using the algorithm, and its usage in solving The Wiki Game.**

Keywords*—*Web crawler, web spider, greedy best first search, Wikipedia, The Wiki Game

I. Introduction

A game is defined as a form of play that is usually competitive and is played according to certain rules. Game players play their games to obtain enjoyment or sometimes to achieve certain rewards. There exists many varieties of games played in different kinds of media, all of which shares the common properties of having challenges, rules, and interactions to the player.

The Wiki Game is one example of a game which uses the Wikipedia, an online free encyclopedia as a media. It is a simple game which only requires Internet access and a web browser. It is played with the goal of reaching a certain preselected article from another. The challenge is to do so in the least number of clicks or least time.

To reach the goal of the game, one strategy that the players can use is to repeatedly scan through the available articles and click on or navigate to the link that they think will lead to the destination article. These series of steps can be automated with a web crawler that uses greedy best first search algorithm. Web crawlers are used to scan and navigate through the articles while greedy best first search algorithm takes care of the article choosing part. This paper will discuss the concepts of web crawlers, greedy best first search algorithm, and the modified web spider that implements greedy best first search algorithm, also its usage in the solver of The Wiki Game.

II. Web Crawler

A web crawler or spider is a computer program, usually distributed, to index content from all over the Internet it is used to collect information. The Internet is a large network that can be modeled down to a directed graph, with the nodes or vertices being pages on the network and the edges being the link between two pages. One important thing to notice here is that a certain page may have a link to another specific page but not vice-versa, hence the graph is directed. Web crawlers traverse this network in a specific manner that follows a selection policy. The policy determines the page to index or download next

The selection policy of a web crawler is a rule that selects which page on the network to index next, or in other words the selection policy selects which node on the graph to visit next. There are many selection policy that can be used in a web crawler. Because the model is a graph, the selection policy can be said to be the graph traversing algorithm used in traversing the network. This algorithm can be focused or non-focused. Non-focused examples for the algorithm would be simple breadth-first traversal or depth-first traversal. While for the focused algorithms examples include academic-focused crawler, this kind of crawler algorithm focuses more on crawling educational sites and research papers.

Other than the selection policy, web crawlers follow other policies including a revisit policy which determines when the crawler needs to revisit an outdated page. Politeness policy that states how the crawler can avoid overloading websites, and a parallelization policy that controls a distributed web crawler.

A common web crawler has modules including a fetcher to fetch preselected links, a link extractor to extract links to be fetched later from the currently fetched page, and link filter to filter links that should not be fetched. The selection policy is usually implemented in the fetcher module. One design of a web crawler can be observed in the figure 1 below.

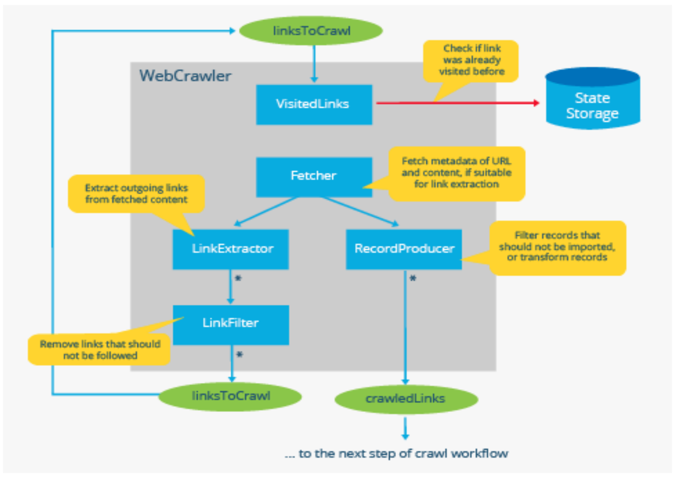

linksToCrawl
WebCrawler
Check if link was already visited before
VisitedLinks
State Storage
Fetcher
Fetch metadata of URL and content, if suitable for link extraction
Extract outgoing links from fetched content
LinkExtractor
RecordProducer
Filter records that should not be imported, or transform records
LinkFilter
Remove links that should not be followed
linksToCrawl
crawledLinks
... to the next step of crawl workflow

Figure 1

Source: https://en.ryte.com/wiki/Crawler

III. Greedy Best First Search Algorithm

Node searching on a graph or tree is one of the most commonly used algorithm in computer science, greedy best first search is a variation of it. Greedy best first search algorithm is a type of informed search algorithm. An informed search algorithm means that the algorithm has information on the goal node. With this knowledge, the algorithm can cut off some search space and find the goal node more efficiently.

The informed search algorithm is especially useful in searching in a large search space. In this type of graph or tree searching algorithm, a heuristic function is used on the nodes to determine how promising an intermediate node is to reach the goal node. The function is denoted as $h(n)$. Examples of this search algorithm includes A* algorithm and greedy best first search algorithm.

Greedy best first search algorithm is also known as pure heuristic search. This algorithm works by expanding the best, most promising node only according to the result of the heuristic function $h(n)$, hence the name pure heuristic search.

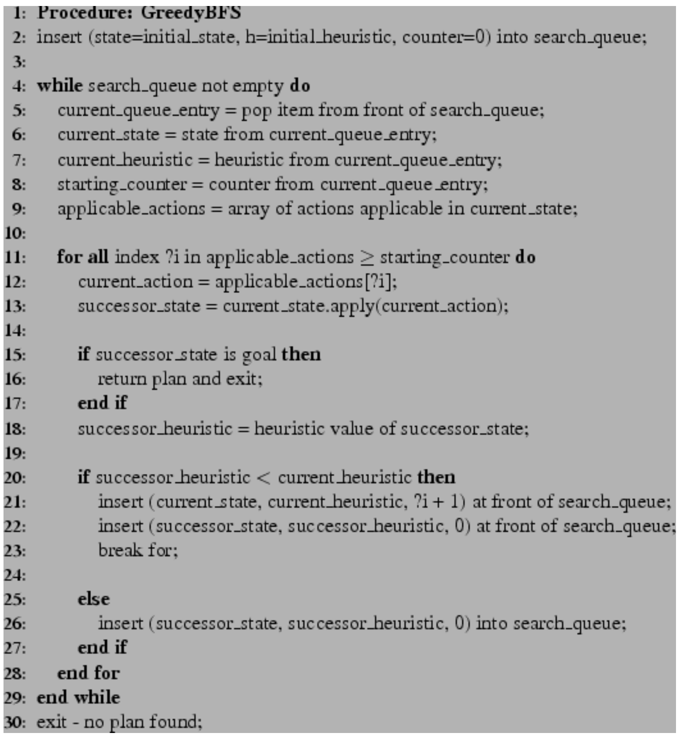
The algorithm itself can be explained as the following steps:

1. Place the starting node on the to-be-expanded list.
2. If the list is empty, stop the search. This means that the search does not find the goal node.
3. Find the node with the lowest value of the heuristic function defined, remove it from the list.
4. Expand the node found by generating its successors of that node.
5. Check if any of the successors are the goal node. If so, stop the search. This means that the search has found the goal node.
6. For every successor node, place it on the to-be-expanded list.
7. Go back to step 2.

For this algorithm, the time and space complexity in Big O notation is $O(b^m)$ with b being the branching factor or the average number of successors per state, and d being the depth of the solution. Some disadvantages of this algorithm is that the algorithm is not complete nor optimal. This algorithm can also get stuck in a loop even if the search space does not contain any loops. This can be handled by noting the already visited nodes so that such nodes would not have to be visited again.

Even so, it is quite efficient in in searching through a large network compared to simple breadth-first or depth-first search. Especially when the network has little to no disconnected nodes or dead-ends.

The pseudo-code for this algorithm can be seen in the figure 2 below. The code uses the algorithm explained earlier but uses a priority queue instead of list to optimize the time complexity. The priority of the queue corresponds with the priority of a node which is the heuristic function value for that node.



```
Procedure: GreedyBFS
insert (state=initial_state, h=initial_heuristic, counter=0) into search_queue;

while search_queue not empty do
  current_queue_entry = pop item from front of search_queue;
  current_state = state from current_queue_entry;
  current_heuristic = heuristic from current_queue_entry;
  starting_counter = counter from current_queue_entry;
  applicable_actions = array of actions applicable in current_state;

  for all index ?i in applicable_actions $\geq$ starting_counter do
    current_action = applicable_actions[?i];
    successor_state = current_state.apply(current_action);

    if successor_state is goal then
      return plan and exit;
    end if
    successor_heuristic = heuristic value of successor_state;

    if successor_heuristic $<$ current_heuristic then
      insert (current_state, current_heuristic, ?i + 1) at front of search_queue;
      insert (successor_state, successor_heuristic, 0) at front of search_queue;
      break for;

    else
      insert (successor_state, successor_heuristic, 0) into search_queue;
    end if
  end for
end while
exit - no plan found;
```

Figure 2

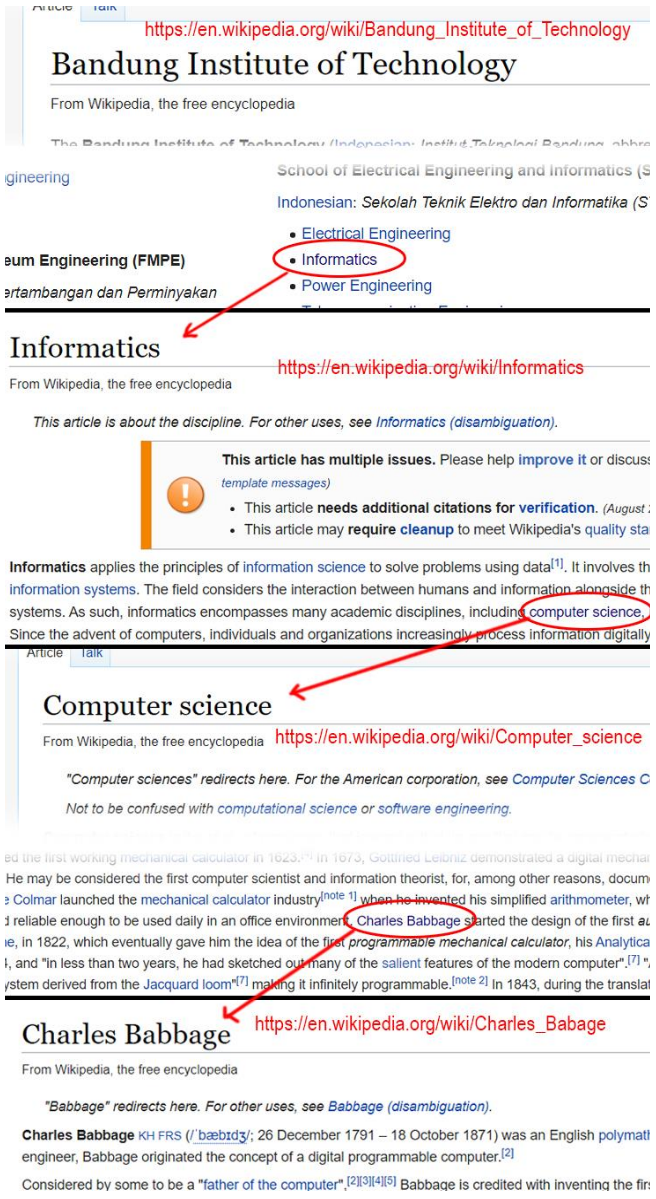
Source: https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume28/coles07a-html/node11.html

IV. The Wiki Game

The Wiki Game, also known as Wikiracing is a simple online game that can be played with only Internet access and a web browser. The goal of the game is to navigate from one Wikipedia article to another using intermediate articles. Intermediate articles here, are the Wikipedia pages that are not the initial article nor the goal article but connects those two articles. The starting and the destination page can be selected or chosen randomly.

The challenge of this game is to accomplish the goal with the least number of pages or least time. Navigating backwards (using the back feature of the web browser), using the Wikipedia search box, and typing directly the destination URL is considered to be a violation to the game rule. In addition to the traditional fewest click and shortest time rule, there are also other varieties of the game rule such as banning the use of several Wikipedia pages or limiting the use of such articles.

One example of The Wiki Game instance is to navigate from the Wikipedia page for Bandung Institute of Technology (https://en.wikipedia.org/wiki/Bandung_Institute_of_Technology) as the starting page to the Wikipedia page for Charles Babbage (https://en.wikipedia.org/wiki/Charles_Babbage) as the destination page. This can be done in only 3 clicks, with the pages: Informatics, Computer Science, and Charles Babbage consecutively, see figure 3 for the details.


Article
Talk
https://en.wikipedia.org/wiki/Bandung_Institute_of_Technology
https://en.wikipedia.org/wiki/Informatics
https://en.wikipedia.org/wiki/Computer_science
https://en.wikipedia.org/wiki/Charles_Babage

Bandung Institute of Technology

From Wikipedia, the free encyclopedia

igineering

School of Electrical Engineering and Informatics (S

Indonesian: *Sekolah Teknik Elektro dan Informatika (S*

eum Engineering (FMPE)

ertambangan dan Perminyakan

- Electrical Engineering
- Informatics
- Power Engineering

Informatics

From Wikipedia, the free encyclopedia

This article is about the discipline. For other uses, see Informatics (disambiguation).

This article has multiple issues. Please help **improve it** or discuss

template messages)

- This article **needs additional citations for verification**. *(August*
- This article may **require cleanup** to meet Wikipedia's quality sta

Informatics applies the principles of information science to solve problems using data[1]. It involves th
information systems. The field considers the interaction between humans and information alongside th
systems. As such, informatics encompasses many academic disciplines, including computer science,
Since the advent of computers, individuals and organizations increasingly process information digitally

Computer science

From Wikipedia, the free encyclopedia

"Computer sciences" redirects here. For the American corporation, see Computer Sciences C

Not to be confused with computational science or software engineering.

ed the first working mechanical calculator in 1623.[4] In 1673, Gottfried Leibniz demonstrated a digital mechar
He may be considered the first computer scientist and information theorist, for, among other reasons, docum
e Colmar launched the mechanical calculator industry[note 1] when he invented his simplified arithmometer, wh
d reliable enough to be used daily in an office environment. Charles Babbage started the design of the first *au*
ie, in 1822, which eventually gave him the idea of the first *programmable mechanical calculator*, his Analytica
, and "in less than two years, he had sketched out many of the salient features of the modern computer".[7] "
ystem derived from the Jacquard loom"[7] making it infinitely programmable.[note 2] In 1843, during the translat

Charles Babbage

From Wikipedia, the free encyclopedia

"Babbage" redirects here. For other uses, see Babbage (disambiguation).

Charles Babbage KH FRS (/ˈbæbɪdʒ/; 26 December 1791 – 18 October 1871) was an English polymat
engineer, Babbage originated the concept of a digital programmable computer.[2]

Considered by some to be a "father of the computer".[2][3][4][5] Babbage is credited with inventing the firs

Figure 3

The path represented in figure 3 is one of the solutions for such instance of the game. Note that there may very possibly be many paths other than this one which can lead from the starting page to the destination page. Those paths are also considered to be a valid solution for the game instance.

Another thing to consider is that an instance of the game might not be solvable if there exists no path from the starting article to the destination article. This usually happens when the starting article has no outbound link or the destination article has no inbound link. Even so, the common Wikipedia pages can be reached from any Wikipedia articles.

V. Web Crawler Using Greedy Best First Search

The aim of the common web crawler is to crawl through a network to collect information from nodes on the network. Which is why a graph traversing algorithms are used. In this paper, the author is interested not in the information collection but in the path finding from a certain node to another. To do so, graph searching algorithm, that is greedy best first search, is used for the selection policy instead of graph traversal algorithm.

To design a web crawler, its components must first be defined. In this case, the main parts of the crawler which are the fetcher, link extractor, and filter is defined in one function that takes the URL to be fetched as a parameter. This function aims to get all available outbound links from the current URL. The detail of the algorithm can be described as the following series of steps:

1. Fetch the URL.
2. Parse the fetched URL to find all links with the HTML tag *"<a>"* and *"</a>"*.
3. For all links in the parsed list, do step 4.
4. If the in the parsed list passes the filter, add it to the final list of links.
5. Return the final list of links.

The filter module of a crawler can be made using all sorts of technique. In this paper, the author will be using regex to match the available links for the filter module of the crawler. A link represented in URL string is said to be available if it matches the regex filter.

The selection policy of this crawler uses the greedy best first search algorithm. As explained before, this algorithm works by expanding the best, most promising node only according to the result of the heuristic function. The heuristic function used may vary according to one's needs.

The integration of greedy best first search to the selection policy of the crawler makes use of priority queue and set data structures, the algorithm can be described as the following series of steps:

1. Place the starting URL and the value of its heuristic function as its priority on the queue.
2. If the queue is empty stop the search, this means that the search does not find the goal URL.

3. Choose the URL with the highest priority from the queue, remove it from the list.

4. If the URL has not been visited before, expand it by generating all valid URLs present using the fetcher function. Else repeat step 2.

5. If any of the successors are the destination URL, stop the search. This means that the search has found the destination URL.

6. For every successors, if the URL has not been visited yet, place it and the value of its heuristic function as its priority on the queue.

7. Go back to step 2.

With this design, a modified web crawler that finds a path from one URL to another is almost complete. What is left is the definition of the filter regex and the heuristic function. This should be defined according to the needs of the crawler.

VI. Implementation of the Web Crawler in The Wiki Game Solver

The Wikipedia (https://www.wikipedia.org/) consists of many articles and links coming into and out of said articles. Thus, the articles can be modeled as nodes and links as edges on a directed graph. By March of 2008, it was observed that there were 2,301,486 articles and 55,550,003 links on the website with the average of around 25 links coming out of each article. This number is still growing. As of April 2020, there are 6,066,409 articles in the Wikipedia. This shows that the Wikipedia is very large and so is the graph model of it. The figure 4 below is a simplified and incomplete model of the Wikipedia article for Bandung Institute of Technology.

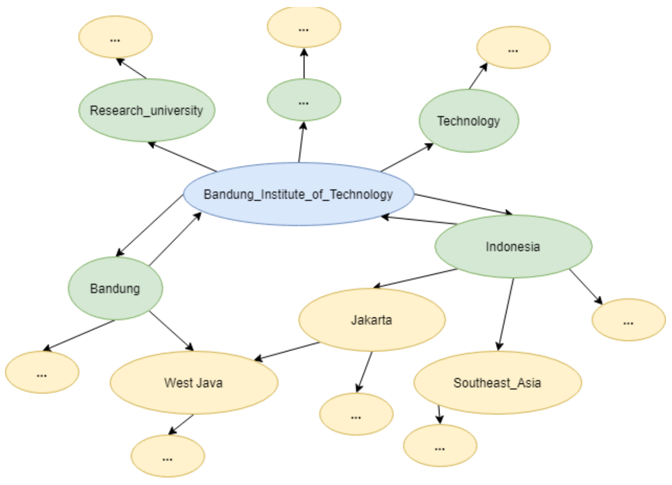


Figure 4

To design a solver for The Wiki Game, a path from the starting node to the destination node in the form of links and articles must be found. To do so, the Wikipedia must be traversed. One way to do so is using a web crawler. The right selection policy for the crawler must also be chosen. One alternative is to use the breadth-first search or uniform-cost search. These algorithms would give an optimal result for the game, which is the shortest path. But, as discussed above, for the size of the Wikipedia, the execution time for these algorithms would be very slow. Thus, the greedy best first search algorithm is used. The design of the web crawler using this algorithm has been discussed previously.

To use this web crawler, the filter regex and the heuristic function for the greedy best first search algorithm must first be defined. The filter regex is used to match the available links. In this case the regex is used to match all Wikipedia pages that are articles. Wikipedia articles has the prefix of *"/wiki/"* after *"https://en.wikipedia.org"* and suffix not containing special annotations such as *"File"*, *"Special"*, etc. To match this rule, the regex in the figure 5 below is used.

```
^/wiki/(?!File|Special|Wikipedia|Help|Portal|Template
Category|Talk)
```

Figure 5

Next, the heuristic function for the greedy best first search algorithm has to be defined. It is critical to choose a good heuristic function because a good heuristic function can lead to a better performance, on the other hand, a bad heuristic function can slow down the performance. The heuristic function must be able to give the most promising node the highest priority value while giving unpromising nodes low priority. This way, there will be less nodes need to be visited to reach the goal.

The heuristic function used here is the prediction of the number of clicks or steps to reach the final node. To define this prediction, several supporting points must be addressed. First, it takes an average of 4.573 clicks to get from one Wikipedia article to another, this is based on the data of Wikipedia on March 2008. Second, to determine the relatedness of an article to another, the number of common links pointing to same articles can be used. The more links the two article share, the more related they are.

With A being the set of links present in the first article and B being the set of links present in the second. The formula to calculate the relatedness from article A to B (R_{AB}) can be written as follows.

$$R_{AB} = \frac{|A \cap B|}{|B|} \times 100\%$$

To better visualize this, some examples are chosen, see the figure 6 below.

Article A (Number of links)	Article B (Number of links)	Number of Common Links	Relatedness from A to B
Bandung (383)	Bandung (383)	383	100%
Surabaya (462)	Bandung (383)	138	36%
Seoul (1114)	Bandung (383)	50	13%
Banana (427)	Bandung (383)	10	2%
Coronavirus (354)	Bandung (383)	6	1%

Figure 6

With these information, a heuristic function that predicts the number of clicks from an article to another can be designed. This prediction or heuristic function has the following formula:

$$h(n) = (1 - R_{nDest}) * 4.573$$

With $h(n)$ being the heuristic function, n being the start point of the prediction, R_{nDest} being the relatedness from n to the destination node. This means that a 100% relatedness will result in the prediction of 0 click and a 0% relatedness will result in the average number of clicks, which is 4.573.

Note that this heuristic function is not admissible. There is no guarantee that this function always returns a prediction that never overestimates the real value. The figure 7 below contains several randomly selected example of pairs of articles, along with their relatedness, heuristic function, and admissibility.

Article A (/wiki/…)	Article B (/wiki/…)	R_{AB}	$h(n)$	Actual	Admissible?
Riencourt	Mers-les-Bains	99.63%	0.01	1	Yes
Oholiab	Uzzah	71.21%	1.31	1	No
Hunting_H. 126	Honda_MH02	26.3%	3.36	2	No
X_With_U	Kim_Heungsou	6.25%	4.28	5	Yes
Monolopia_congdonii	Wish_Egan	3.84%	4.39	4	No
Sourdough	Navillera	1.67%	4.49	4	No

Figure 7

With both the heuristic function and the filter function defined, the solver of The Wiki Game is complete.

VII. Case Studies

This chapter will show the Wiki Game solver in work for some case studies in the form of game instances. The solver will display the heuristic function for every node or articles and which article it will expand in the next iteration. At the end of the program, it will output the path from the source article to the destination article, number of articles visited (clicks needed), and the number of secondary articles visited (to calculate the heuristic function of each node).

The first example shows the solver solving the game with the starting point being the article with the URL /wiki/Algorithm and the destination being the article with the URL /wiki/Electron. The optimal solution is two clicks. Figure 8 shows the output of the program.

```
#1 loading /wiki/Algorithm
/wiki/Stephen_C._Kleene 0/392 h(n)=4.506186688311689
/wiki/Formulation_1 1/392 h(n)=4.558152597402597
/wiki/Flowchart 2/392 h(n)=4.543305194805195
/wiki/Analytical_engine 3/392 h(n)=4.528457792207793
/wiki/Graph_theory 4/392 h(n)=4.424525974025975
/wiki/Latin 5/392 h(n)=4.491339285714286
/wiki/Binary_search 6/392 h(n)=4.506186688311689
... [100 lines truncated]
/wiki/Control_flow 107/392 h(n)=4.543305194805195
/wiki/J._Barkley_Rosser 108/392 h(n)=4.535881493506494
```

```
Found with path:
/wiki/Algorithm ->
/wiki/State_University_of_New_York_at_Stony_Brook ->
/wiki/Electron
In 53.37142038345337 seconds
found in 2 clicks
and 110 secondary visits
```

Figure 8

The second example shows the solver solving the game with the start being the article with the URL /wiki/Web_crawler and the destination being the article with the URL /wiki/Nasi_goreng. The optimal solution is 3 clicks, the solver did not find the optimal solution, but instead it found the solution with 4 clicks. The output of the program is shown on the figure 9 below.

```
#1 loading /wiki/Web_crawler
/wiki/Apache_Hadoop 0/197 h(n)=4.563686354378819
... [196 lines truncated]
#2 loading /wiki/Yahoo!_Search from /wiki/Web_crawler
h(n)=4.549715885947047
/wiki/Portuguese_language 0/257 h(n)=4.535745417515275
... [256 lines truncated]
#3 loading /wiki/Indonesian_language from
/wiki/Yahoo!_Search h(n)=4.49849083503055
/wiki/Moken_language 0/1377 h(n)=4.554372708757638
... [83 lines truncated]

Found with path:
/wiki/Web_crawler -> /wiki/Yahoo!_Search ->
/wiki/Indonesian_language -> /wiki/Culture_of_Indonesia
-> /wiki/Nasi_goreng
In 309.6990053653717 seconds
found in 4 clicks
and 539 secondary visits
```

Figure 9

The last example shows the solver solving the game with the start being the article with the URL /wiki/Bezalel and the destination being the article with the URL /wiki/Yuju_(singer). Optimally this can be solved in 4 clicks, the solver did not find the optimal solution, but instead, as before, it found the solution with 5 clicks. The output of the program is shown on the figure 10 below.

```
#1 loading /wiki/Bezalel
/wiki/Gaza_City 0/70 h(n)=4.534571428571429
... [68 lines truncated]
/wiki/Levite 69/70 h(n)=4.534571428571429
#2 loading /wiki/Solomon from /wiki/Bezalel
h(n)=4.342428571428572
/wiki/Kingdom_of_Israel_(Samaria) 0/505
h(n)=4.534571428571429
... [504 lines truncated]
#3 loading /wiki/National_Library_of_Korea from
/wiki/Solomon h(n)=4.227142857142858
/wiki/Biblioth%C3%A8que_nationale_de_France 0/91
h(n)=4.342428571428572
... [90 lines truncated]
#4 loading /wiki/Seoul from
/wiki/National_Library_of_Korea h(n)=4.073428571428572
... [67 lines truncated]

Found with path:
/wiki/Bezalel -> /wiki/Solomon ->
/wiki/National_Library_of_Korea -> /wiki/Seoul ->
/wiki/Goyang -> /wiki/Yuju_(singer)
In 423.9701063632965 seconds
found in 5 clicks
and 734 secondary visits
```

Figure 10

VIII. Conclusion

Web crawler is a type of program that crawls through a network to collect information, it uses a selection algorithm to choose which page or node to visit next. Greedy best first search algorithm is a graph searching algorithm that uses purely heuristics to choose which node to expand next. The modification of the web crawler using graph searching algorithm switches the functionality of the common web crawler to search for a page on a network. Greedy best first search is one alternative of the algorithm that is quite fast and reliable although the optimality is not guaranteed. One of the implementation of this modified web crawler using greedy best first search is in the solver of The Wiki Game.

Video Link at YouTube

The video about this paper can be accessed in the following YouTube link: https://youtu.be/eKskiVFCCO8.

Acknowledgment

The author would like to thank God, for only by His blessings and grace the author is able to finish this paper. The author would also like to express gratitude to the author's family, friends, and everyone that has supported the author in process of writing this paper. Lastly, the author would like to thank Dr. Ir. Rinaldi, M. T. as the author's lecturer in the IF2211 course, algorithm strategies.

References

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/A-Star-Best-FS-dan-UCS-(2018).pdf

[2] https://www.cloudflare.com/learning/bots/what-is-a-web-crawler/

[3] https://en.ryte.com/wiki/Crawler

[4] https://www.javatpoint.com/ai-informed-search-algorithms

[5] http://mu.netsoc.ie/wiki/

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 30 April 2020

[illegible]

Steve Bezalel Iman Gustaman - 13518018