Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IMPACT FACTOR: 6.017

IJCSMC, Vol. 6, Issue. 4, April 2017, pg.259 – 266

A System for Searching, Extracting & Copying for Algorithm, Pseudocodes & Programs in Data

Mr. Vinod S. Jadhav[1], Dr. Rekha Rathore[2]

[1]PG Student RKDF SOE Indore & RGPV Bhopal, India

[2]Associate Professor, RKDF SOE Indore & RGPV Bhopal, India

[1] v23jadhav@gmail.com; [2] rekharathore23@gmail.com

Abstract: Algorithm, Programs, Pseudocodes is most widely used in software development. Many Search Engines (SEs) available but it is still hard to locate and concentrate only on material specific task. Software algorithm, program is a process to change functionality and effectiveness of the software. AlgorithmSeer, One type of search engine is a use to search for algorithm, has investigated as a part of CiteSeerX with the purpose of providing a large database of algorithms. . It's the function of automatically find and extract these algorithm in this increasingly vast collection of scholarly large digital documents. The system purpose a novel set of scalable technique used by AlgorithmSeer to identify and external algorithm representation in a different pool of scholarly document.

Keywords— AlgorithmSeer, Algorithm Procedure (AP's), Pseudo codes (PC's), Machine Learning and Regular expression algorithm

I. INTRODUCTION

Algorithm is a important role in many software projects. For example an algorithm is help to improve the performance of the software used for indexing and searching billions of documents. Hence it's important for software developer to keep abreast of latest algorithmic developments related to their projects. The system is basically to work in past to help software developer search for old source code, algorithm, pseudo code , however to best of our knowledge , no effort has been mode to develop tools and technique that can help software developer search for literature about the algorithm used in source code in documents that describe new design developments.

The as per my knowledge algorithm procedure and pseudo code are not easily available with their analysis and it required more efforts and time to search them. Numbers of algorithm procedure, pseudo codes, and programs are being to publish every year national and international journal. So searching this code is difficult as effort are needed to compare and analysis them to determine most efficient one. So, searching this code is difficult as efforts are needed to compare and analyze them to determine the most efficient one. These techniques involve knowledge discovery from web and available research paper from national and international journal. To achieve this input request is accepted, indexing id done using proper mechanism and most relevant algorithm procedure, pseudo code, programs are listed. Also user is provided with downloading facility for algorithm procedure and pseudo codes. Hence, algorithm procedures and pseudo code extraction and analysis become an important part of this implementation.

II. BACKGROUND & RELATED WORK

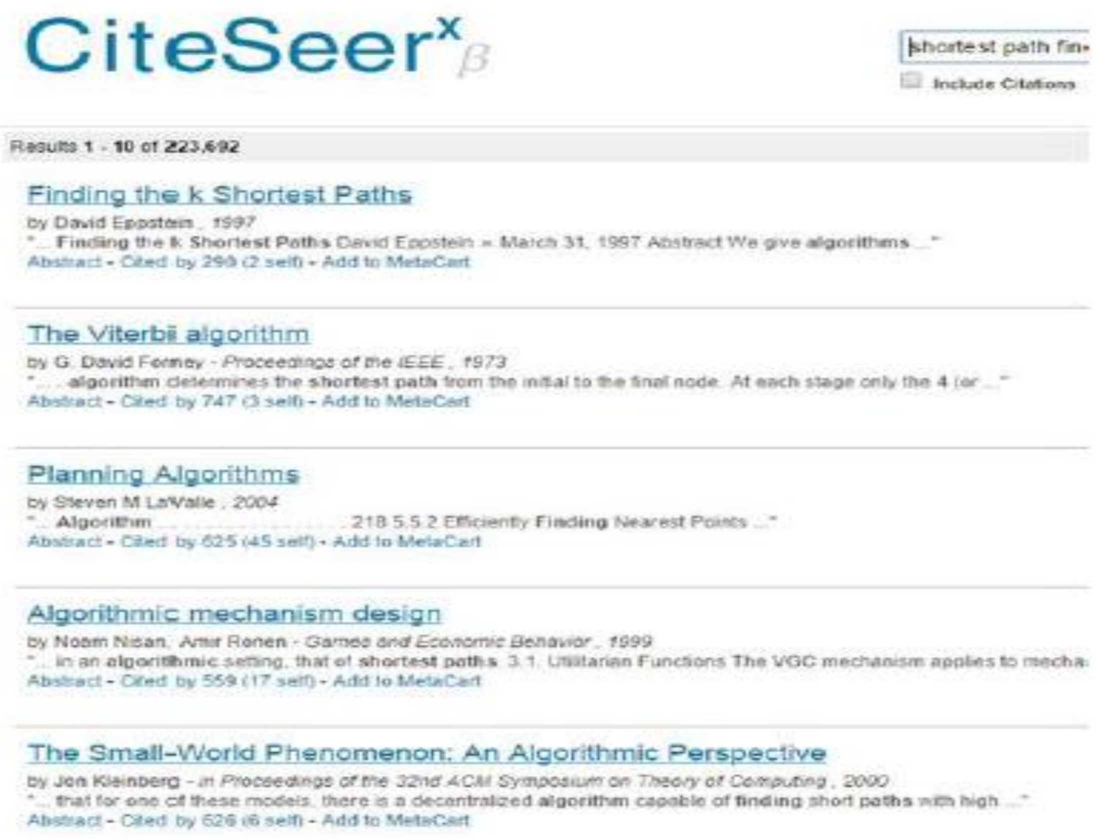
Every year many more algorithm and pseudo code are being publish in national and international journals [1]. So searching for efficient and relevant algorithm procedure and pseudo code become difficult as effort are needed to compare and analyze them to determine the most efficient one.

II.I Extraction of element in scholarly documents

So many more information is identified and extraction entities such as mathematical expressions, tables, figure and contents of tables form documents have been studied. They also proposed a method to index and search for the information, which is extracted. Sojka and Liska proposed Math Indexer and prescribed. The user may note peculiarities. For example, the head margin in this template measures proportionately more than is Customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and Searcher that interprets and collects mathematical expressions.

II.II System to search for Scholarly Information

CiteSeer is a one of digital library and one of the search engine for all academic documents. Basically the focus of all CiteSeer has been on computer science, information science and related category; however, in recent years there has been an expansion to include other academic fields also. Features of this algorithm are that it automatically performs information extraction on all documents added to the collection and automatic indexing, which allows for citations and other information among papers to be linked[1].



Results 1 - 10 of 223,692

Finding the k Shortest Paths
by David Eppstein , 1997
"... Finding the k Shortest Paths David Eppstein = March 31, 1997 Abstract We give algorithms ..."
Abstract - Cited by 299 (2 self) - Add to MetaCart

The Viterbi algorithm
by G. David Forney - *Proceedings of the IEEE* , 1973
"... algorithm determines the shortest path from the initial to the final node. At each stage only the 4 (or ..."
Abstract - Cited by 747 (3 self) - Add to MetaCart

Planning Algorithms
by Steven M LaValle , 2004
"... Algorithm .... 218 5.5.2 Efficiently Finding Nearest Points ..."
Abstract - Cited by 625 (45 self) - Add to MetaCart

Algorithmic mechanism design
by Noam Nisan, Amir Ronen - *Games and Economic Behavior* , 1999
"... in an algorithmic setting, that of shortest paths. 3.1. Utilitarian Functions The VGC mechanism applies to mecha
Abstract - Cited by 559 (17 self) - Add to MetaCart

The Small-World Phenomenon: An Algorithmic Perspective
by Jon Kleinberg - *in Proceedings of the 32nd ACM Symposium on Theory of Computing* , 2000
"... that for one of these models, there is a decentralized algorithm capable of finding short paths with high ..."
Abstract - Cited by 626 (6 self) - Add to MetaCart

Fig. 1. Sample search results for the query “shortest path finding algorithm” on CiteseerX.

Fig. 1 illustrates an example of a search session (top five results) for shortest path finding algorithms using the query “shortest path searching algorithm” on CiteseerX search engine that hosts 5 million academic publications whose major proportion is computer science and related disciplines[4]. According to the figure, only two out of five top results are relevant (actual papers that describe shortest path algorithms), while the other three results are simply documents containing those search terms. These search engines would return the whole documents (papers or websites) as results, requiring the users to invest additional, unnecessary effort to read the entire documents to find the desired algorithms. This system is the first to explore the possibility of building a search engine specifically for algorithms extracted from scholarly digital documents.

III. PROPOSED SYSTEM

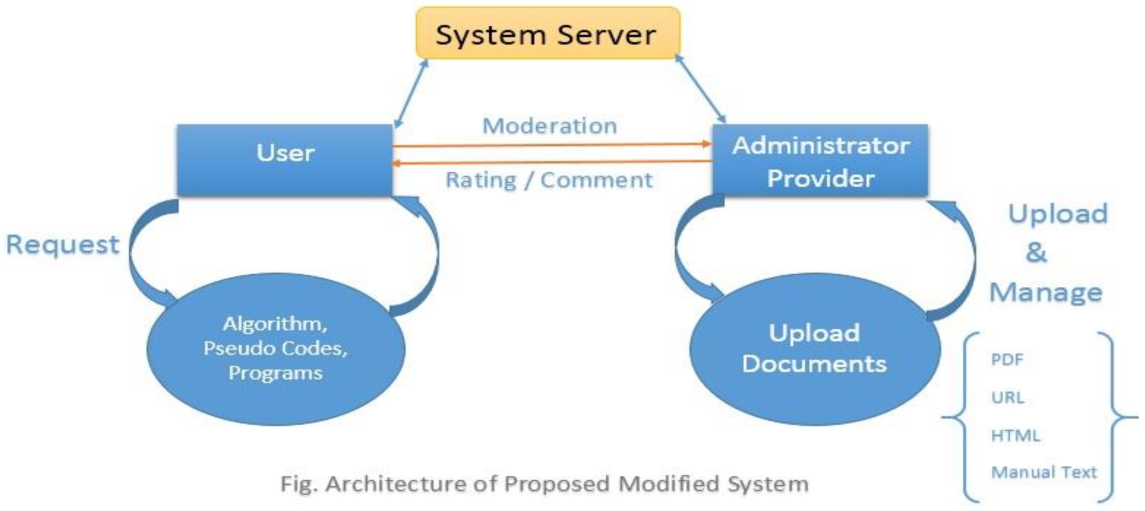


Fig.2 Architecture of the proposed system

In this system, a prototype of an algorithm search engine, AlgorithmSeer, is presented. Fig. 2 explains the high level of the proposed system. The proposed system analyzes a document to identify any algorithms that may be pin the document. If an algorithm is found in a document, the document text is further analyzed to extract additional information about the algorithm. All the algorithms thus found with their associated metadata are indexed and made available for searching through a text query interface. For a given user query, the system utilizes evidence from multiple sources to assign relevance scores to algorithms and results are presented to the user in decreasing order of relevance. First, scholarly documents are processed to identify algorithm representations. Then, the textual metadata that provides relevant information about each detected algorithm representation is extracted. The extracted textual metadata is then indexed, and made searchable.

The document segmentation module identifies sections in the document. The PC detection module detects PCs in the parsed text file. The AP detector first cleans extracted text and repairs broken sentences

IV. WORKFLOW

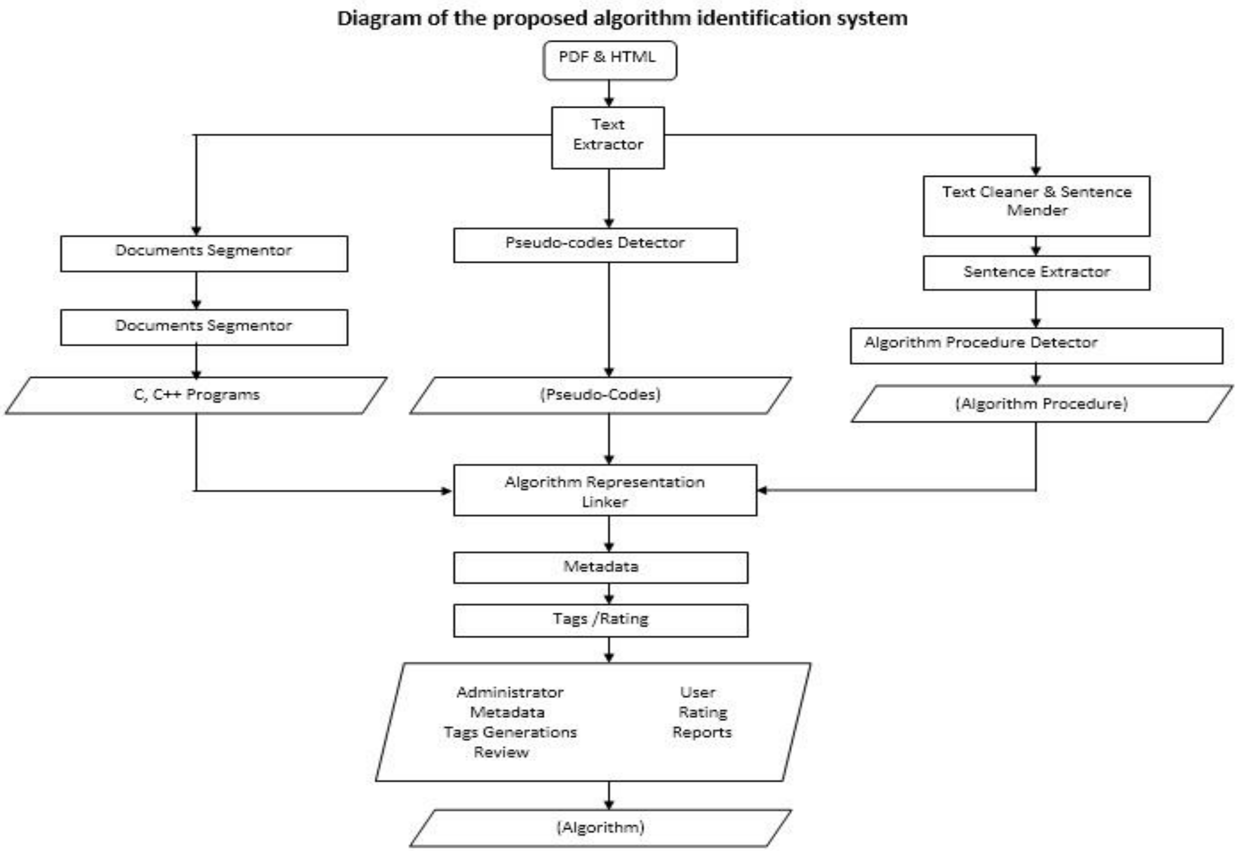


Fig.3 Data Flow diagram

In this figure 3 is show the flow structure in various process. Such as uploading pdf, URL after that how to separate of program. How to generate report. User requests, tags, Reports.

V. Algorithm Identification

In this section discusses of methods for automatic discover of PCs and APs in scholarly documents. The system specifically handles PDF documents since a majority of articles in modern digital libraries including CiteseerX are in PDF format. First, plain text is extracted from the PDF file. Inspired by Hassan, we use PDFBox13 to extract text and modify the package to also extract object information such as font and location information from a PDF document. Then, three sub-processes operate in parallel, including document segmentation, PC detection, and AP detection.

The document segmentation module identifies sections in the document. The PC detection module detects PCs in the parsed text file. The AP detector first cleans extracted text and repairs broken sentences (since the method assumes that a document is represented with a sequence of sentences), then identifies APs. After PCs and APs are identified, the final step involves linking these algorithm representations referring the same algorithms together. The final output would then be a set of unique algorithms [5].

VI.I Detecting Pseudo-Codes (PCs)

Since PCs can appear anywhere in a document, these identifiers usually serve the purpose of being anchors which can be referred to by context in the running text. Here, three approaches for detecting PCs in scholarly documents are presented: a rule based method (PC-RB), an ensemble machine learning based method (PC-ML), and a combined method (PC-CB)[4]. Note that though OCR based techniques (where each page of the document is first converted into an image, where image processing based techniques are used to locate the boundary of the PCs, then an OCR technique is used to extract the content within each candidate region) have been explored, textual content can be directly and quite accurately extracted from most PDF files.

VI.I.I Rule Based Method (PC-RB)

Recently, Bhatia et al. proposed a rule based PC detection approach. In which utilizes a grammar for document-element captions to detect the presence of PC captions. The methods are referred as our baseline (PC-BL) for the PC detection task.

VI.I.II Machine Learning Based Method (PC ML)

The PC-RB contains a high precision, however it still suffers from a low coverage resulting in a poor recall. It is found that 25.8 percent of PCs in dataset do not have accompanied captions. The PC-ML first detect and extracts these sparse boxes, then classifies each box whether it is a PC box or not. A PC box is a sparse box that contains at least 80 percent content of a PC. The following sections explain how sparse boxes are identified, the feature sets, and the classification models. The output of the PC-ML is a tuples of start; ending line numbers of the detected PC.

VI.I.III Combined Method (PC-CB)

Though the PC-ML method can capture PCs even though they do not have accompanied captions, some PCs are not first captured in a sparse box would still remain undetected. such PCs cannot be captured using the sparse box extraction. However, 27 (out of 35) of these undetected PCs have accompanied captions and hence might still be detected using the PC-RB method. This proposed method combines method PC-CB of the (PC-RB) and the PC- using a simple heuristic as follows:

STEP1 For a given document, run both PC-RB and PC-ML methods.

STEP2 For each PC box detected by PC-ML, if a PC caption detected by PC-RB is in proximity, then the PC box and the caption should combined.

VI.II Detecting Algorithmic Procedures (APs)

Two methods are developed for detecting AP indication sentences: a rule based method (AP-RB) and a machine learning based method (AP-ML). Both methods rely on the sentences which correctly extracted from the document. However, most scholarly documents are multi-columned and contain document elements such as tables and diagrams.

Hence, before extracting sentences, garbage text fragments are removed from the document. The heuristic is also developed that stitches up an incomplete sentence which are broken into multiple lines.

VI.II.I Rule Based Method (AP-RB)

Often, AP indication sentences exhibit certain common properties:

- The sentences usually end with follows:, steps:, algorithm:, follows:, following:, follows., steps:, below:.
- The sentences usually contain at least an algorithm keyword.

We create a set of regular expressions to capture sentences according to the rules above.

VI.II.II Machine Learning Based Method (AP-ML)

Some AP indication sentences do not conform to the rules described in Section 4.2.1. Therefore, an alternative approach based on machine learning is proposed to directly learn to capture the characteristics of APs.

VI. III Linking Algorithm Representations

A simple heuristic is implemented to link algorithm representations referring to the same algorithm together. Specifically, two algorithm representations are linked if:

1) They represent the same algorithm. For Ex. an Algorithm Procedure AP's may be used to provide more detail to a PC.
2) They are part of the same algorithm. For Ex. an algorithm may be broken into sub-parts, each is represented using a different algorithm representation.

VI.III.I Identifying Sections in Scholarly Documents

A machine learning based approach is used to identify section boundaries by detecting section headers. The algorithm was first proposed by Turbojet almost scholarly documents have following common properties:
1) Each section has a section header and section number.
2) Section headers have distinct font styles from the surrounding content.
3) A most of sections are common sections such as Abstract, Introduction, Background, Conclusions, and References.

VI.III.II Using Document Sections for Linking Algorithm

Representations Assigning an AP to a section is easy, since it is part of the running text. Unlike APs, PCs are normally located separately from the running text; hence, they could appear outside the sections.. To get around this problem, a PC is mapped to the section that contains the largest number of reference sentences that refer to it. Once each algorithm representation is assigned to a section, algorithm representations which are mapped to the same sections are then linked.

VII. Algorithm

We have basically used in regular expression algorithm. In concept is used to separation of program, algorithms many mores. In this concept is select program and directly to separate to from PDF, Web page.

```
$plain_text = strip_tags(nl2br($content),"<br><p><br />");
$plain_text = html_entity_decode($plain_text, ENT_QUOTES, 'UTF-8');
$plain_text = str_ireplace("&nbsp;", " ", $plain_text);
$plain_text = preg_replace('/(<[^>]+) style=".*?"/i', '$1', $plain_text);
$plain_text = preg_replace('/(<[^>]+) class=".*?"/i', '$1', $plain_text);
$plain_text = str_ireplace("<p>", "<br />", $plain_text);
$array_java_prog = array ('int','public','for','if','system','print','while');
$array_algo = array('end','for','repeat','begin');
matchRegExp($plain_text, "int function", "end;",$array_algo);
matchRegExp($plain_text, "procedure", "end;",$array_algo);
matchRegExp($plain_text, "#include", "}",$array_c_prog);
matchRegExp($plain_text, "void ", "}",$array_c_prog);
//matchRegExp($plain_text, "void function", "}",$array_algo);
matchRegExp($plain_text, "void function", "end;",$array_algo);
$plain_text = str_replace('<stdio.h>', '&lt;stdio.h&gt;', $plain_text);
$plain_text = str_replace('</conio.h>', '', $plain_text);
$plain_text = str_replace('<n', '&lt;n', $plain_text);
```

VIII. ANALYSIS

VIII.IEXISTING SYSTEM

Number of algorithm procedures and pseudo codes are being published every year in national and international journal. So searching for efficient and relevant algorithm procedures and pseudo code become difficult as efforts are needed to compare and an analyse them to determine the most efficient one.

In Existing system, For searching algorithms, user have to go on different sites and search the codes, ands algorithms. From PDF Documents we have to search the algorithms manually.

URL based searching is not provided in Existing system.

VIII.II THE PROPOSED SYSTEM

In this system, a prototype of an algorithm search engine, Algorithm Seer, is presented. Fig. 2 explains the high level of the proposed system. The proposed system analyses a document to identify any algorithms that may be present in the document. If an algorithm is found in a document, the document text is further analyzed to extract additional information about the algorithm.

All the algorithms thus found with their associated metadata are indexed and made available for searching through a text query interface. For a given user query, the system utilizes evidence from multiple sources to assign relevance scores to algorithms and results are presented to the user in decreasing order of relevance.

First, scholarly documents are processed to identify algorithm representations. Then, the textual metadata that provides relevant information about each detected algorithm representation is extracted. The extracted textual metadata is then indexed, and made searchable.

URL based searching method is added. User can search the C, Cpp codes from URL.

IX. Difference between Existing and Proposed System

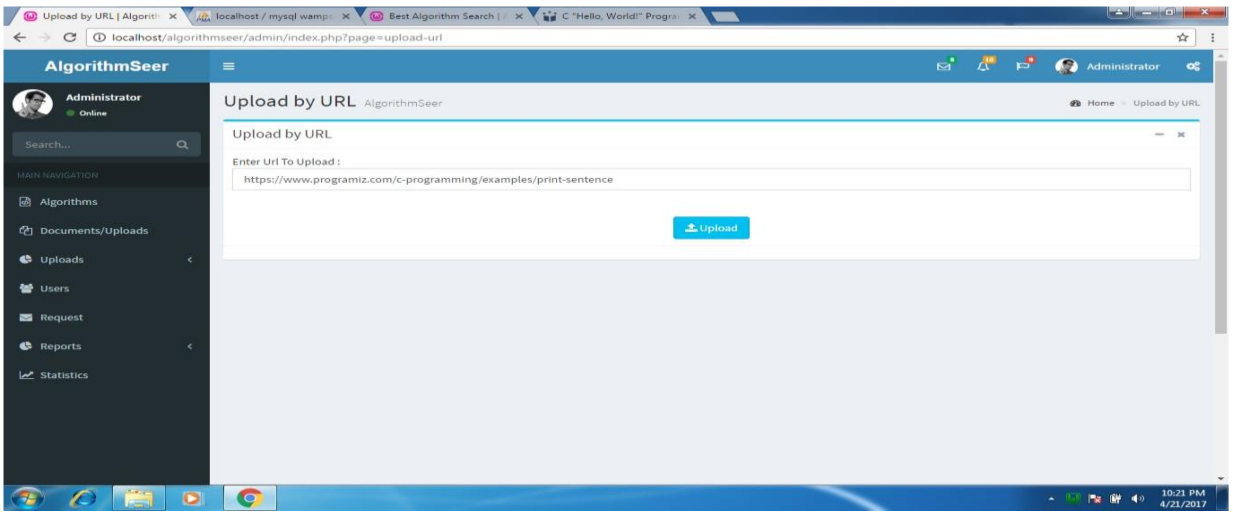
Sr.No.	Existing System	Proposed System
1	User have to on different sites for searching algorithms.	Algorithms can be easily searched.
2	Manual searching of algorithms from PDF.	Algorithms are easily Extracted from the PDF.
3	URL based searching is not Provided.	Efficient searching is Done by using URL based Searching.
4	Algorithms was not search according to user choice.	User can request for any algorithm he want.
5	User have to search algorithms details manually.	User can know all the details of the algorithm. Ex. Time complexity, Space complexity.
6	In existing there is no facility for searching algorithms by Author or type.	User can also search any algorithm by type and author.
7.	No option for adding different algorithms.	Admin can add different algorithms in the system.

Fig. Table Difference OLD & NEW System

X. Mathematical Model

Let S be the system design for Multicloud Architecture.
S = (s, e, x, y, sucess case, failure case, DD, NDD)
Where,
s = Start State
e = End State
x = Input
y = Output
DD = Deterministic Data
NDD = Non-Deterministic Data
1. **Start State**: Admin select the pdf and upload it.
2. **End State:** Algorithms are extracted from pdf.
3. **Input:**
Let x be the set of inputs such as x ={x1,x2,......xn}
x1=PDF document.
x2= URL.
4. **Output:**
Let y be the set of inputs such as y={y1,y2,y3,.....yn}
y1= Extracted algorithms from PDF.
y2= Extracted C,C++ codes from the URL.
y3= Detail report of algorithm with time and space complexity.
5**. Deterministic Data:** PDF document.
6. **Non-Deterministic Data:** URL

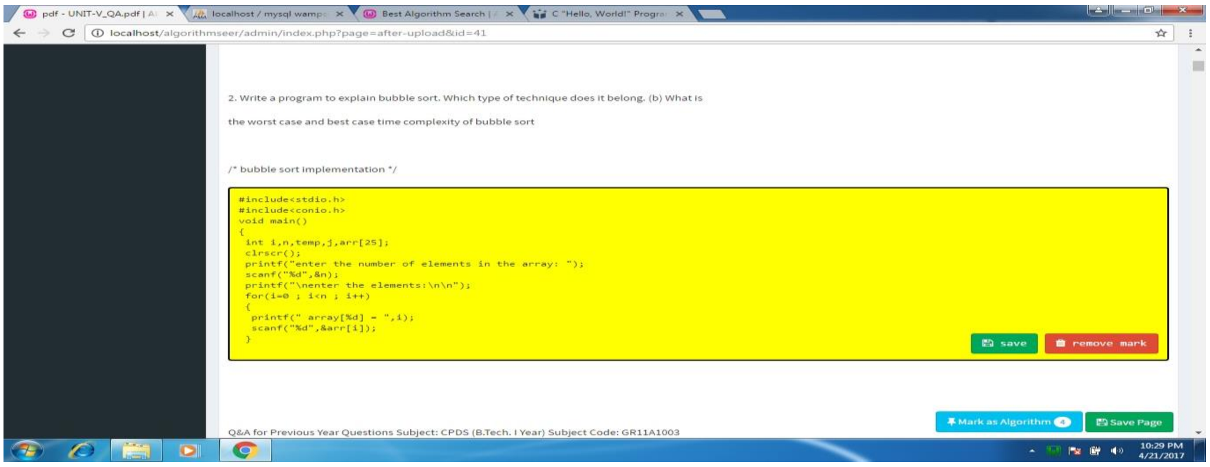
XI. Report Result



localhost/algorithmseer/admin/index.php?page=upload-url

Upload by URL

Enter Url To Upload : https://www.programiz.com/c-programming/examples/print-sentence


localhost/algorithmseer/admin/index.php?page=after-upload&id=41
save
remove mark
Mark as Algorithm 4
Save Page

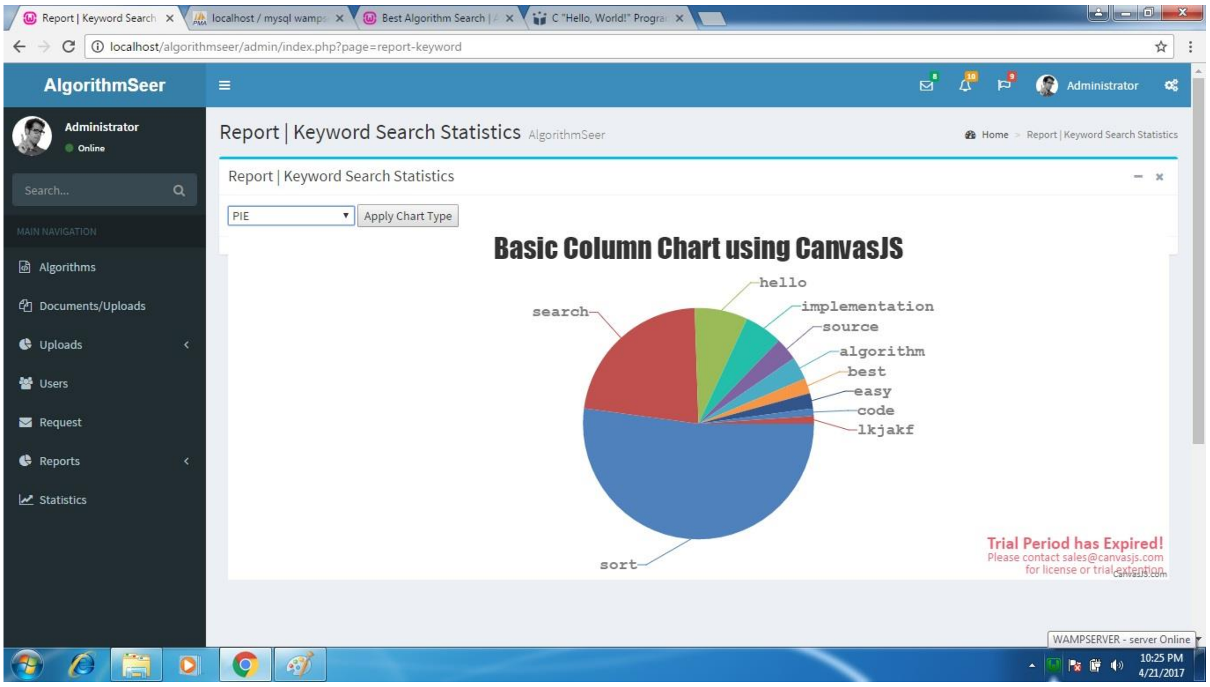
2. Write a program to explain bubble sort. Which type of technique does it belong. (b) What is

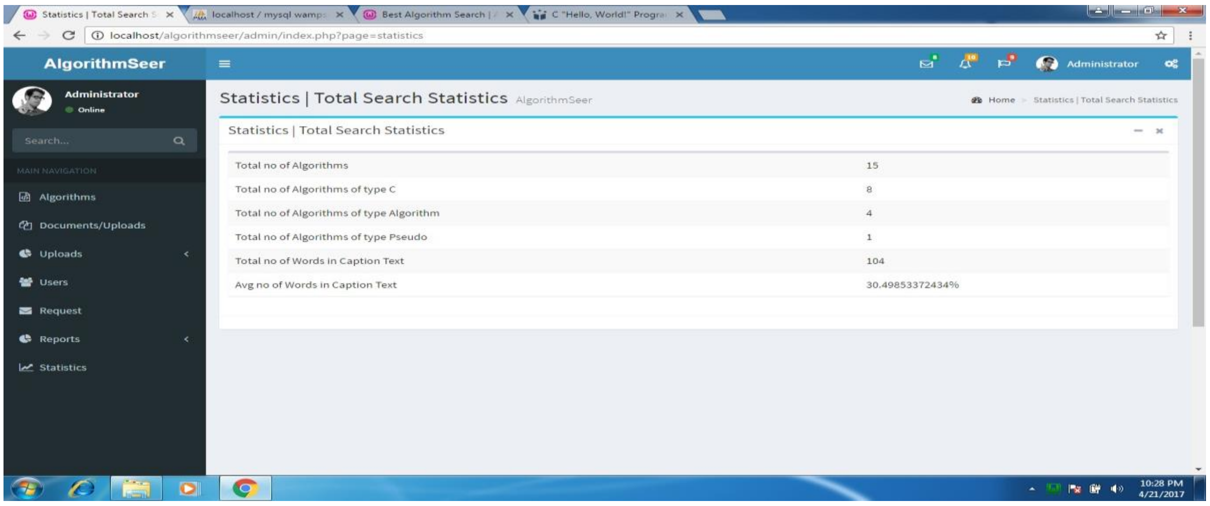
the worst case and best case time complexity of bubble sort

/* bubble sort implementation */

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int i,n,temp,j,arr[25];
 clrscr();
 printf("enter the number of elements in the array: ");
 scanf("%d",&n);
 printf("\nenter the elements:\n\n");
 for(i=0 ; i<n ; i++)
 {
  printf(" array[%d] = ",i);
  scanf("%d",&arr[i]);
 }
```

Q&A for Previous Year Questions Subject: CPDS (B.Tech. I Year) Subject Code: GR11A1003


localhost/algorithmseer/admin/index.php?page=report-keyword
AlgorithmSeer
Administrator
Online
Search...
MAIN NAVIGATION
Algorithms
Documents/Uploads
Uploads
Users
Request
Reports
Statistics
Report | Keyword Search Statistics AlgorithmSeer
Home > Report | Keyword Search Statistics
PIE
Apply Chart Type
Basic Column Chart using CanvasJS
hello
implementation
source
algorithm
best
easy
code
lkjakf
search
sort
Trial Period has Expired!
Please contact sales@canvasjs.com for license or trial extention.



Statistics | Total Search Statistics AlgorithmSeer

Statistics | Total Search Statistics

Total no of Algorithms	15
Total no of Algorithms of type C	8
Total no of Algorithms of type Algorithm	4
Total no of Algorithms of type Pseudo	1
Total no of Words in Caption Text	104
Avg no of Words in Caption Text	30.49853372434%

XII. Conclusion

Algorithms are very important in solving research problems. Scientific publications host a tremendous amount of such high-quality algorithms developed by professional researchers. In digital libraries, being able to extract and catalogue these algorithms will introduce a number of exciting applications including algorithm searching, discovering, and analyzing. Specifically, proposed system is a set of scalable machine learning based methods to detect algorithms in scholarly documents. The annotation methods for documents are extract textual metadata for pseudo-codes are discussed, and how algorithms are indexed and made searchable.

XIII. Future Work

Future work would be the semantic analysis of algorithms, programs, their trends, and how algorithms influence each other over time. Such analyses would give rise to multiple applications that could improve algorithm search. We can add document files also; Robustness of the system will increase. In this system basically used in future separation of algorithm, programs in C, C++ , JAVA , VB , PHP , HTML etc.

References

[1] Suppawong Tuarob, Member, IEEE, Sumit Bhatia, Prasenjit Mitra, Senior Member," AlgorithmSeer: A System for Extracting and Searching for Algorithms in Scholarly Big Data" IEEE TRANSACTIONS ON BIG DATA, VOL. 2, NO. 1, JANUARY-MARCH 2016.

[2] Sumit Bhatia, Suppawong Tuarob, Prasenjit Mitra and C. Lee Giles"An Algorithm Search Engine for Software Developers" The Pennsylvania StateUniversity Park, PA-16802, USA.

[3] Yves Petinot1, C. Lee Giles1,2,3, Vivek Bhatnagar2,3, Pradeep B. Teregowda1, Hui Han1"Enabling Interoperability For Autonomous.

[4] Suppawong Tuarob, Prasenjit Mitra and C. Lee Giles" Improving Algorithm Search Using the Algorithm Co-Citation Network"

[5] Suppawong Tuarob, Sumit Bhatia, Prasenjit Mitra, C. Lee Giles " Automatic Detection of Pseudocodes in Scholarly Documents Using Machine Learning".

[6] Kyle Williamsz, Jian Wuy, Sagnik Ray Choudhuryz, Madian Khabsay, C. Lee Gilesy"Scholarly Big Data Information Extraction and Integration in the CiteSeer_ Digital Library"