PERFORMANCE EVALUATION OF PRINTED CHARACTER RECOGNITION SYSTEM

Dr. Pramod Kumar Rai,
Head, Computer Centre, A. P. S. University, Rewa (M.P.) India

ABSTRACT

In spite of today's powerful computer systems and intensive research for last five decades, it might be surprising to note that the machines are still far from the final frontier i.e. human's performance in case of reading.

The character recognition systems for various applications are classified based upon two major criteria – (i) according to the data acquisition process as, on-line or off-line (ii) according to the text type, as machine-printed or hand-written character recognition.

Although there are numerous elaborated and mature recognition techniques, only few systems for restricted applications are working quite satisfactorily. However, machine's recognition performance is still considerably lower than that of humans especially for unconstrained on-line and off-line handwriting. This inspired researchers to focus not only on the development of novel recognition algorithms, but also on the improvement of other aspects of recognition systems.

In the present paper we have made experiments to evaluate the recognition rate of printed character. The performance evaluation of printed character recognition has been made to assess how well the recognizer performed on the particular page. In the experiment the average recognition rate 94.97 has been obtained.

1. INTRODUCTION

Character Recognition or Optical Character Recognition (OCR) is the process of converting scanned image of machine printed or handwritten text into a computer processable format, such as, ASCII. In the last decade researchers have made significant efforts, both in terms of

technological supports and in software products to make available computerized document analysis systems. Character recognition (OCR) contributes to this progress by providing techniques to convert large volumes of data automatically. With survey of the literature in this field we find that there are so many papers and patents advertising and claiming recognition rates very high. The claimed recognition rates gives the impression that automation problems in this field seem to have been solved. However, in real life the failure of some real applications show that performance problems subsist on composite and degraded documents (i.e., noisy characters, tilt, mixing of fonts, etc.) and that there is still need for progress in this area. To increase the accuracy of optical character recognizers, researchers have proposed various methods. There is a parallel analogy between the various stages of evolution of OCR systems and those of pattern recognition. The classical approach focusing on isolated characters, to overcome the recognition deficiency. To improve the recognition rate researchers are exploring and using contextual techniques The opening of OCR domain to document recognition leads to combination of many strategies such as document layout handling , font identification , dictionary checking , word recognition , integration of several recognition approaches etc.

Intensive research has made OCR an efficient means of entering data directly into the computer and capturing information from data sheets, books and other machine printed or handwritten materials. Such capabilities greatly widen the applications of computers in areas like automatic reading of texts and data, man-computer communications, language processing and machine translation. OCR has been a subject of great interest to many computer scientists, engineers and people from other disciplines. A block diagram of typical OCR system is shown in Fig. 1.

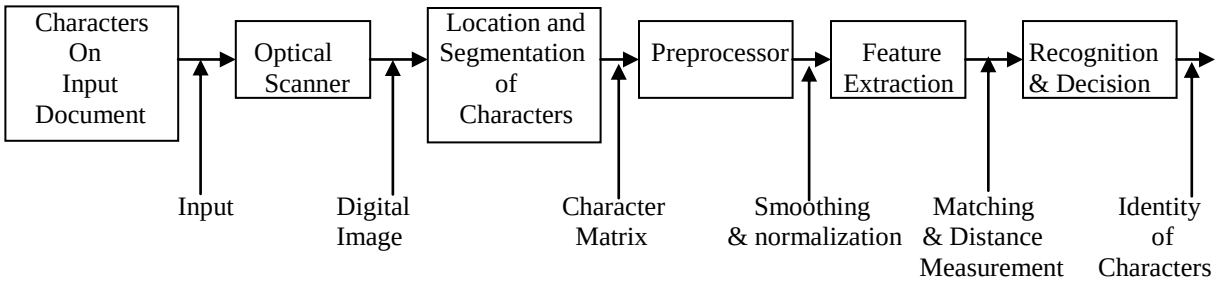


Figure 1 : Block diagram of a typical OCR system

There exists hundreds of type fonts and thousands of print fonts in the world, each having its own distinctive style and peculiarities such as serifs, shapes, curvatures, sizes, pitch, line, thickness etc. Due to the great variety of fonts, machine recognition of multifont and handwritten characters, have many problems.

Variations in handwritten characters are even greater than those in type fonts because they can be written in an immensely different number of ways. Since each person has his/her own ways and styles of writing and character samples written by the same hand are never identical in shape orientation and in size, there are an infinite number of possible character shapes. Actually, the problem of handprint recognition is of great interest and challenge to researchers because even human beings, who possess the best trained optical readers (their eyes) and interpreters (their minds) would make about 4% mistakes when reading hand printing in the absence of context. Due to the similar topological structures, the characters 6/G. D/O, I/1, S/5, 2/Z and U/V are the most confusing pairs, especially when they are written sloppily.

The great variability in hand printing may be attributed to writing habits, style and ease in writing, education, region of origin, mood, health and other conditions of the writer. Apart from these human factors, writing instruments, writing surfaces and scanning equipment and methods, as well as machine recognition algorithms also play an important role in explaining the recognition rate. In their paper [SS13] have given an overview of OCR and reviewed advantages and limitations of Character Recognition.

2. Handwriting Recognition

The final endeavor of handwriting recognition research is to make computers able to read human written texts, with a performance comparable to or even better than that of humans[L99]. The use of handwriting is involved in many of our day-to-day activities, such as note taking, form filling and letter addressing. During the past two decades, there have been increasing demands for the applications to automatically process the content of these handwritten documents with the recent advent of many hand-held devices that accept handwriting inputs.

The performance of handwriting recognition systems has improved dramatically over the past decade, especially in some specific tasks, such as handwritten address reading and the recognition of amounts on bank cheques [B03]. However, the performance of general handwritten word and sentence recognition is rather low and still not comparable to that achieved by humans.

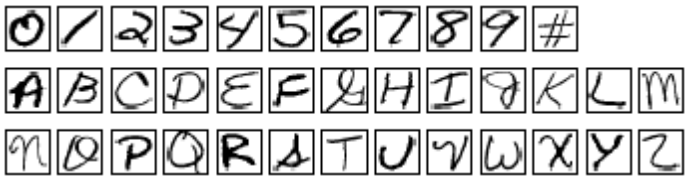
Researchers divide the field of handwriting recognition into off-line and on-line recognition. The off-line recognition systems recognize the characters after they are written on a piece of paper, scanned using the computer and stored in the image format. Whereas, the on-line systems can access dynamic information of handwriting strokes while the characters are being written on a tablet or a digitizer.

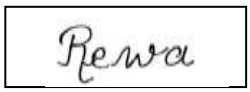
Although, the recognition processes of both systems are different, the four fundamental sequential stages are (i) pre-processing (ii) feature extraction (iii) classification and (iv) post-processing.

In pre-processing handwriting inputs are prepared to be suitable for later recognition stages. The goal of the feature extraction stage is to obtain a compact description (a feature vector) that can be used to uniquely represent the character. The main decision making stage is classification in which the extracted features are classified into one of several categories. Modern handwriting recognition research is dominated by the use of statistical methods for classification, i.e. statistical classifiers, as seen in a recent survey [VBB04]. Post-processing typically forms a verification step, such as the use of language models and contextual information to verify the recognized characters or words.

Here we describe some important terminology. Units of meaningful handwriting can be categorized into the four types : character, word, phrase and sentence [S93]. (See Figure 2.1).

(a)





(b)

(c)

Figure 2.1 : Example of handwritten (a) characters (b) words, and (c) phrases

3. Non-Discriminative and Discriminative Methods

In order to classify the unseen samples efficiently, some training algorithms are required to train the classifiers according to the presence of the training samples. Broadly classifiers are divided into two groups based on their training approaches: non-discriminative and discriminative classifiers.

Non-discriminative classifiers, sometimes referred to as generative or informative classifiers, aim at building a model to represent the training samples in each class. Given the unseen character, classification is done by choosing the model that best explains the data. Examples of non-discriminative classifiers are Hidden Markov Models (HMMs) and Gaussian Mixture Models. These classifiers usually rely on non-discriminative training methods such as Maximum Likelihood (ML) estimation in which the model of each class is trained separately by using the samples that belong to the class. Discriminative classifiers, on the other hand, only centre on the classification decisions, and hence do not attempt to construct a model over the training samples. They focus directly on determining the class decision boundaries, which is equivalent to learning a direct mapping from the training samples to their class labels. Examples include neural networks and Support Vector Machines (SVMs). The training process requires simultaneous considerations of the training samples of all competing classes through the use of discriminative training methods, which involve the optimization of some discriminative training criteria.

4. Novel Trends of Development

Despite the existence of the numerous elaborated and mature recognition techniques, machine's reading performance is still considerably lower than that of humans. This inspired researchers to focus not only on the development of novel recognition algorithms, but also on the improvement of other aspects of handwriting recognition systems. To overcome the difficulties and inherent limitations of collecting a large number of human written samples, the researchers are investigating the possibility of generating synthetic handwriting.

Another approach which has become a very active and popular research area is the combination of classifiers [KHDM98]. The basic idea is to use several different classifiers (experts) to classify an input pattern. The advantage of the approach is that errors made by an individual classifier can be corrected by the others, for example if we decide for the pattern class which is suggested by the majority of the recognizers. For further details see [K04].

Nowadays it seems that human reading performance at general unconstrained texts cannot be achieved by using merely the information extracted from the ink pattern. There is a demand for

shifting from the pattern recognition framework to a paradigm that emphasizes the utilization of much more a-priori knowledge [L99]. According to the new framework described in [L99], ambiguities occurring in difficult handwriting would be resolved mainly by applying the available linguistic knowledge, while using only general knowledge about the handwriting ink, namely its invariants common to a large variety of handwriting styles. This way the operation of the system would be transparent, so its errors could be analyzed and corrected more efficiently.

5. Hidden Markov Models

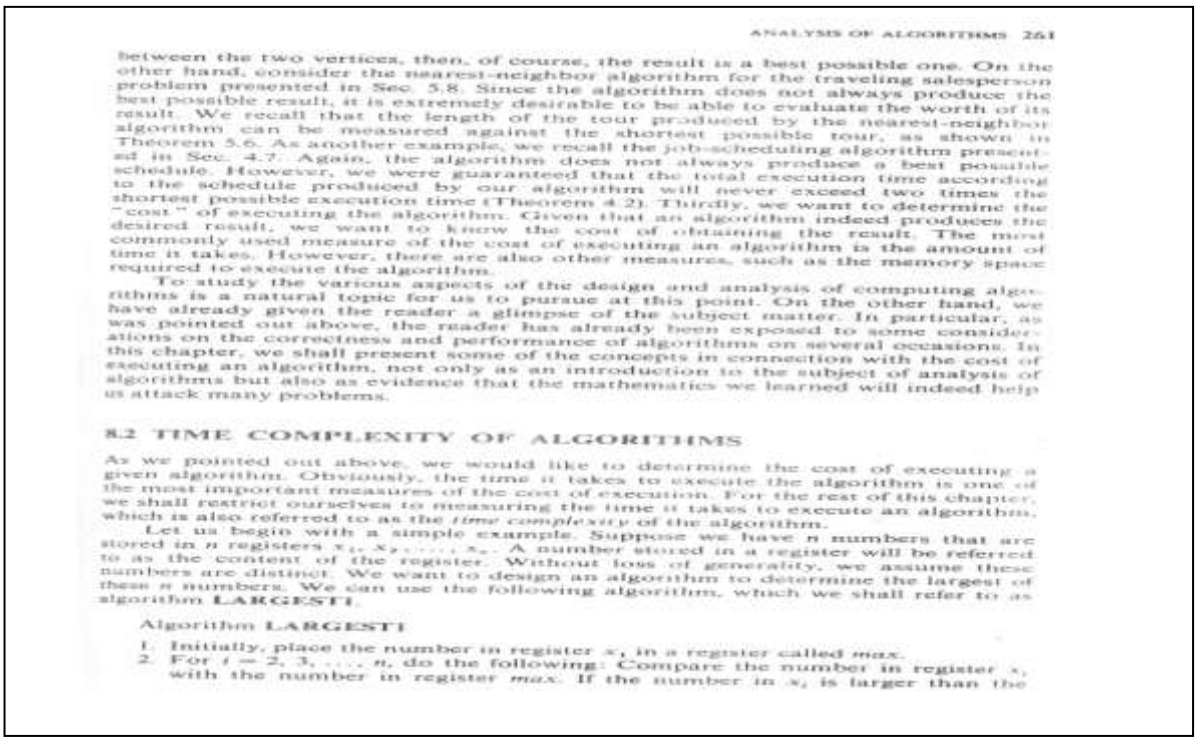
For a wide spectrum of applications in many fields of pattern recognition the Hidden Markov Modeling (HMM) methods have been found to be extremely useful. Earlier the HMM technique was applied in the field of speech recognition later on the HMM have been used for hand-writing recognition application. Researchers have shown, the benefits of their application to different forms of OCR, in particular is that of handwriting recognition. In case of On-line handwriting recognition, the temporal information of how characters or words are constructed by the writer is available to recognizer, is a very good example of the use of HMM approach. Rabiner and Juang results [RJ86], [R89] have shown promising results for off-line handwriting using HMM recognition.

6. Performance Evaluation of Printed Character Recognition

For performance evaluation of printed character recognition, we scanned 11 simple printed pages of different books and magazines. We used three different scanners and recognized these pages with the commercially available recognition software bundled with the scanners (HP Scanjet 1150 (sc1) , Canon Lide 25 (sc2), and HP Scanjet 8350 (sc3)). Considering one page only, the output of the recognizer is a sequence of words, which has to be compared with the correct transcription of the page image to assess the quality of the recognition result, that is, to assess how well the recognizer performed on that particular page. Then, certain related statistics are calculated, such as number of words of both the correct transcription and the recognition result. From this statistics, the recognition rate, which is the percentage of the correctly recognized words of the page are calculated as –

Recognition Rate = (Correct Words/Number of words) * 100

The statistics related to this experiment is summarised in the Table-6.1. One of the sample printed page is given in Figure -6.1. The recognition result pages are shown in Figure 6.2 for sc1, Figure 6.3 for sc2 and Figure 6.4 for sc3. Recognition rate of printed page and overall recognition rates are shown in Figure 6.5 and Figure 6.6.



ANALYSIS OF ALGORITHMS 261

between the two vertices, then, of course, the result is a best possible one. On the other hand, consider the nearest-neighbor algorithm for the traveling salesperson problem presented in Sec. 5.8. Since the algorithm does not always produce the best possible result, it is extremely desirable to be able to evaluate the worth of its result. We recall that the length of the tour produced by the nearest-neighbor algorithm can be measured against the shortest possible tour, as shown in Theorem 5.6. As another example, we recall the job-scheduling algorithm presented in Sec. 4.7. Again, the algorithm does not always produce a best possible schedule. However, we were guaranteed that the total execution time according to the schedule produced by our algorithm will never exceed two times the shortest possible execution time (Theorem 4.2). Thirdly, we want to determine the "cost" of executing the algorithm. Given that an algorithm indeed produces the desired result, we want to know the cost of obtaining the result. The most commonly used measure of the cost of executing an algorithm is the amount of time it takes. However, there are also other measures, such as the memory space required to execute the algorithm.

To study the various aspects of the design and analysis of computing algorithms is a natural topic for us to pursue at this point. On the other hand, we have already given the reader a glimpse of the subject matter. In particular, as was pointed out above, the reader has already been exposed to some considerations on the correctness and performance of algorithms on several occasions. In this chapter, we shall present some of the concepts in connection with the cost of executing an algorithm, not only as an introduction to the subject of analysis of algorithms but also as evidence that the mathematics we learned will indeed help us attack many problems.

8.2 TIME COMPLEXITY OF ALGORITHMS

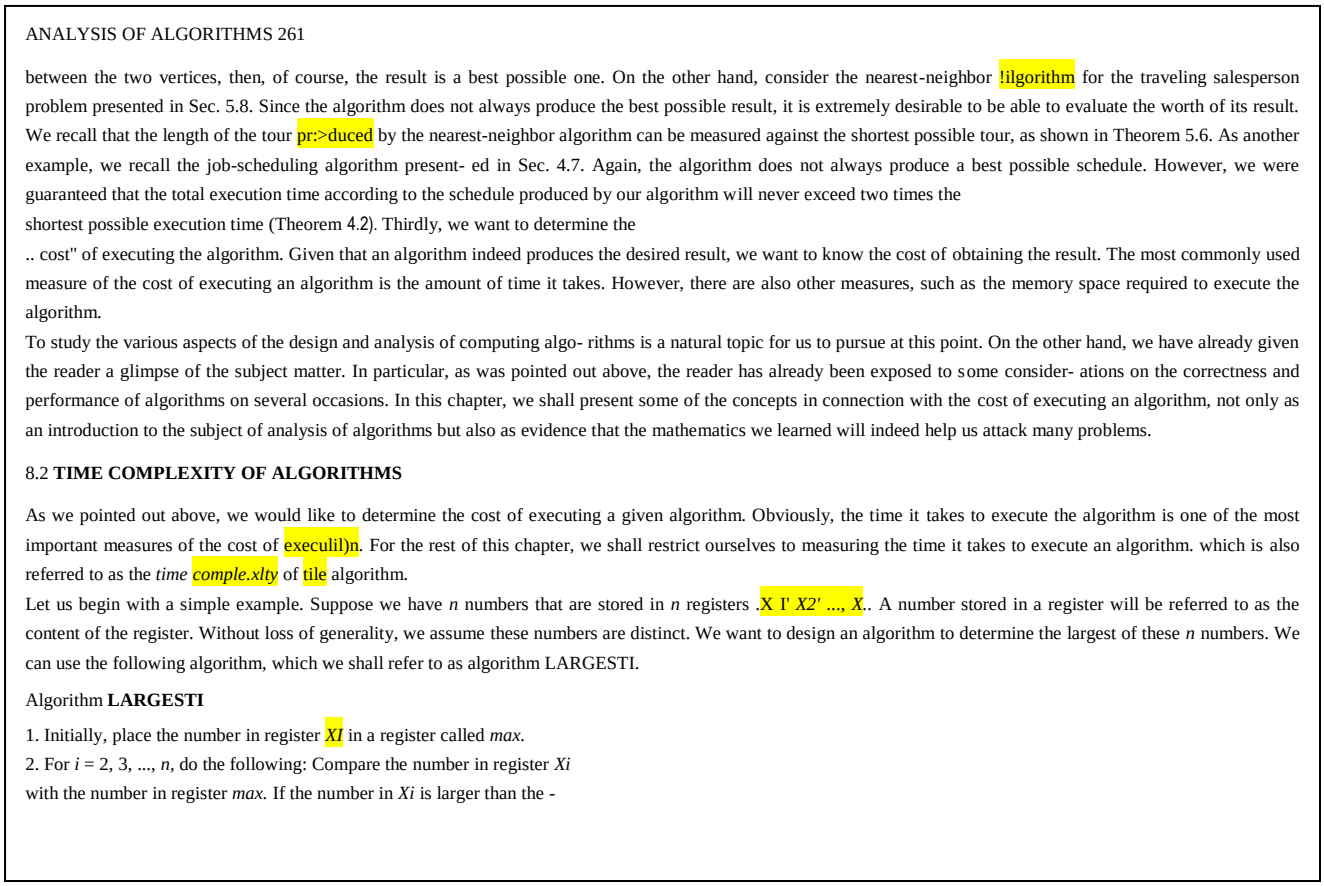
As we pointed out above, we would like to determine the cost of executing a given algorithm. Obviously, the time it takes to execute the algorithm is one of the most important measures of the cost of execution. For the rest of this chapter, we shall restrict ourselves to measuring the time it takes to execute an algorithm, which is also referred to as the *time complexity* of the algorithm.

Let us begin with a simple example. Suppose we have n numbers that are stored in n registers $x_1, x_2, \ldots, x_n$. A number stored in a register will be referred to as the content of the register. Without loss of generality, we assume these numbers are distinct. We want to design an algorithm to determine the largest of these n numbers. We can use the following algorithm, which we shall refer to as algorithm **LARGEST1**.

Algorithm **LARGEST1**

1. Initially, place the number in register x_1 in a register called *max*.
2. For $i = 2, 3, \ldots, n$, do the following: Compare the number in register x_i with the number in register *max*. If the number in x_i is larger than the

Figure 6.1 : Sample scanned printed page



ANALYSIS OF ALGORITHMS 261

between the two vertices, then, of course, the result is a best possible one. On the other hand, consider the nearest-neighbor !ilgorithm for the traveling salesperson problem presented in Sec. 5.8. Since the algorithm does not always produce the best possible result, it is extremely desirable to be able to evaluate the worth of its result. We recall that the length of the tour pr:>duced by the nearest-neighbor algorithm can be measured against the shortest possible tour, as shown in Theorem 5.6. As another example, we recall the job-scheduling algorithm present- ed in Sec. 4.7. Again, the algorithm does not always produce a best possible schedule. However, we were guaranteed that the total execution time according to the schedule produced by our algorithm will never exceed two times the
shortest possible execution time (Theorem 4.2). Thirdly, we want to determine the
.. cost" of executing the algorithm. Given that an algorithm indeed produces the desired result, we want to know the cost of obtaining the result. The most commonly used measure of the cost of executing an algorithm is the amount of time it takes. However, there are also other measures, such as the memory space required to execute the algorithm.
To study the various aspects of the design and analysis of computing algo- rithms is a natural topic for us to pursue at this point. On the other hand, we have already given the reader a glimpse of the subject matter. In particular, as was pointed out above, the reader has already been exposed to some consider- ations on the correctness and performance of algorithms on several occasions. In this chapter, we shall present some of the concepts in connection with the cost of executing an algorithm, not only as an introduction to the subject of analysis of algorithms but also as evidence that the mathematics we learned will indeed help us attack many problems.

8.2 **TIME COMPLEXITY OF ALGORITHMS**

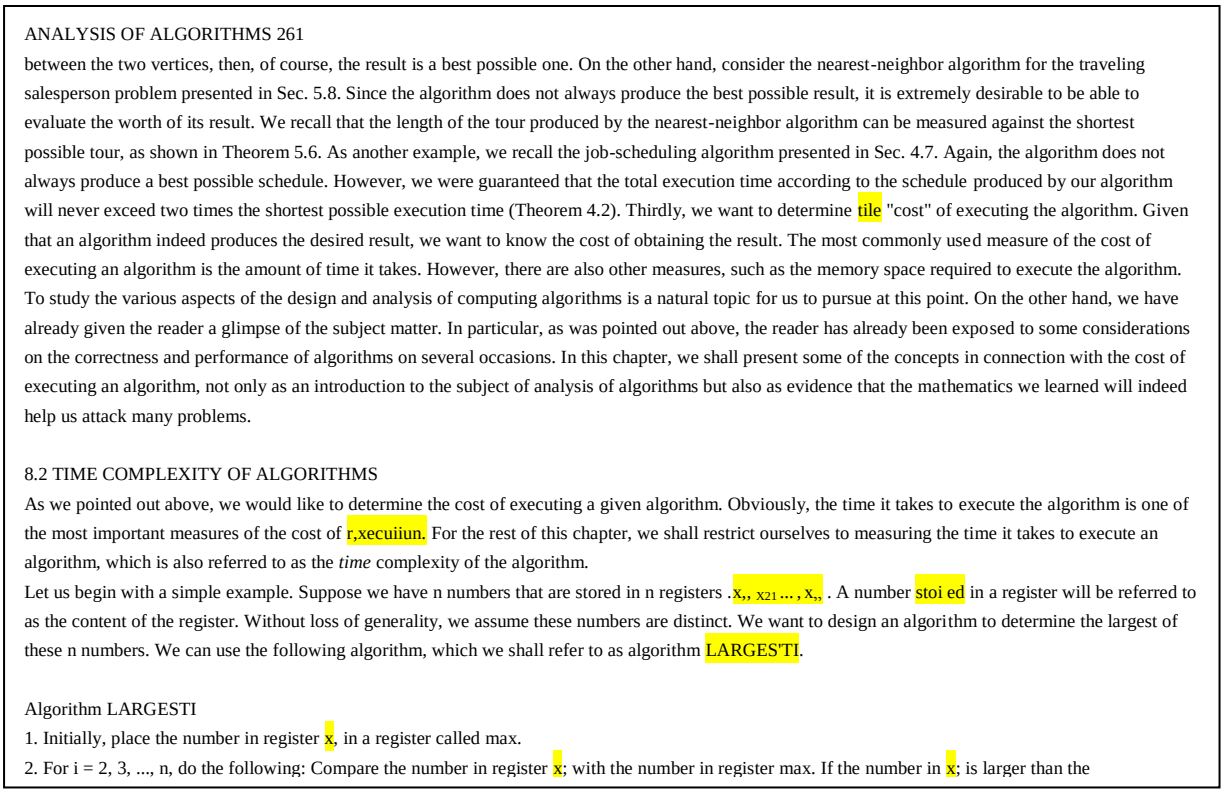
As we pointed out above, we would like to determine the cost of executing a given algorithm. Obviously, the time it takes to execute the algorithm is one of the most important measures of the cost of execulil)n. For the rest of this chapter, we shall restrict ourselves to measuring the time it takes to execute an algorithm. which is also referred to as the *time comple.xlty* of tile algorithm.
Let us begin with a simple example. Suppose we have *n* numbers that are stored in *n* registers .X I' *X2'* ..., *X*.. A number stored in a register will be referred to as the content of the register. Without loss of generality, we assume these numbers are distinct. We want to design an algorithm to determine the largest of these *n* numbers. We can use the following algorithm, which we shall refer to as algorithm LARGESTI.

Algorithm **LARGESTI**

1. Initially, place the number in register *XI* in a register called *max.*
2. For *i* = 2, 3, ..., *n,* do the following: Compare the number in register *Xi*
with the number in register *max.* If the number in *Xi* is larger than the -

Figure 6.2 : Recognition Result of printed page using sc1

Figure 5.3 : Recognition Result of printed page using sc2



ANALYSIS OF ALGORITHMS 261

between the two vertices, then, of course, the result is a best possible one. On the other hand, consider the nearest-neighbor algorithm for the traveling salesperson problem presented in Sec. 5.8. Since the algorithm does not always produce the best possible result, it is extremely desirable to be able to evaluate the worth of its result. We recall that the length of the tour produced by the nearest-neighbor algorithm can be measured against the shortest possible tour, as shown in Theorem 5.6. As another example, we recall the job-scheduling algorithm presented in Sec. 4.7. Again, the algorithm does not always produce a best possible schedule. However, we were guaranteed that the total execution time according to the schedule produced by our algorithm will never exceed two times the shortest possible execution time (Theorem 4.2). Thirdly, we want to determine tile "cost" of executing the algorithm. Given that an algorithm indeed produces the desired result, we want to know the cost of obtaining the result. The most commonly used measure of the cost of executing an algorithm is the amount of time it takes. However, there are also other measures, such as the memory space required to execute the algorithm. To study the various aspects of the design and analysis of computing algorithms is a natural topic for us to pursue at this point. On the other hand, we have already given the reader a glimpse of the subject matter. In particular, as was pointed out above, the reader has already been exposed to some considerations on the correctness and performance of algorithms on several occasions. In this chapter, we shall present some of the concepts in connection with the cost of executing an algorithm, not only as an introduction to the subject of analysis of algorithms but also as evidence that the mathematics we learned will indeed help us attack many problems.

8.2 TIME COMPLEXITY OF ALGORITHMS

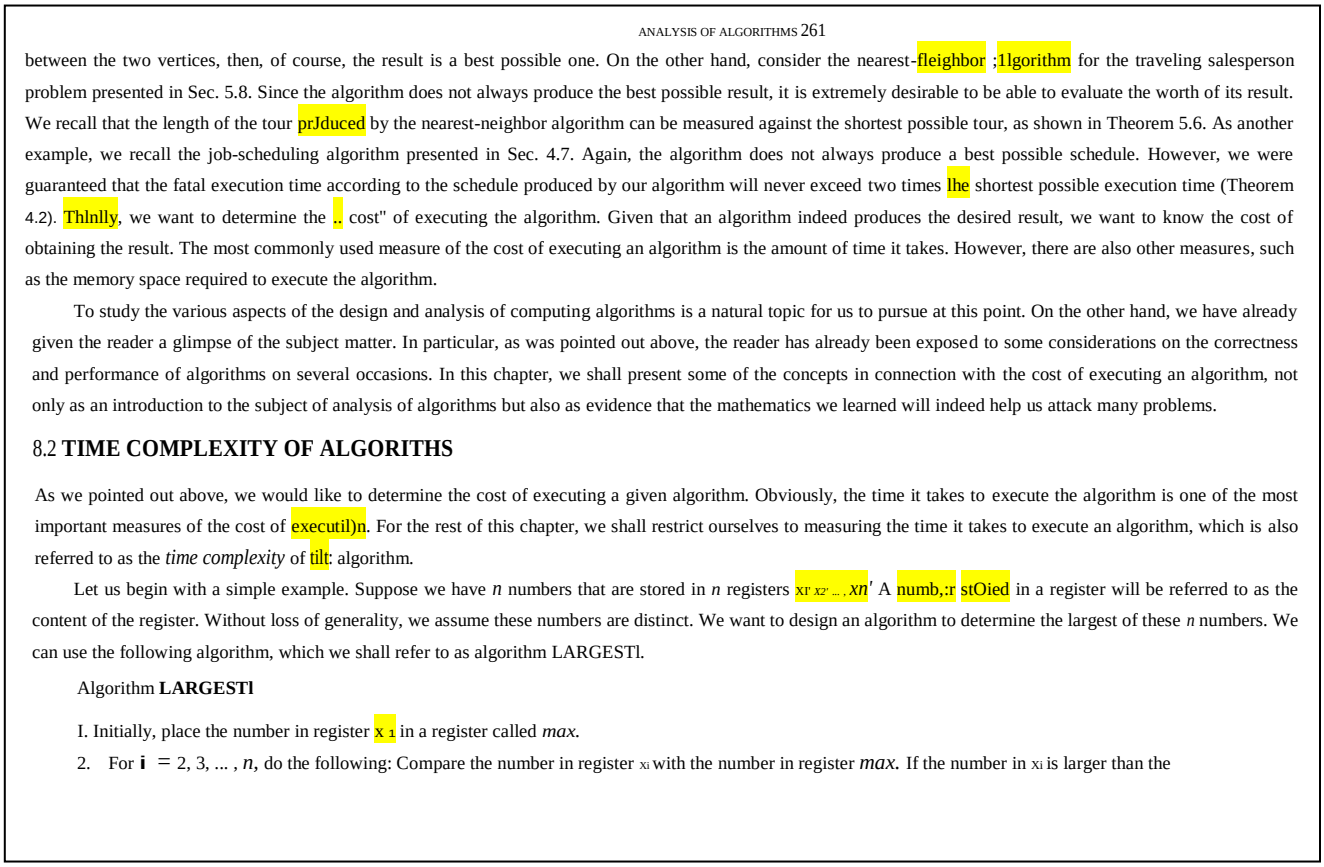
As we pointed out above, we would like to determine the cost of executing a given algorithm. Obviously, the time it takes to execute the algorithm is one of the most important measures of the cost of r,xecuiiun. For the rest of this chapter, we shall restrict ourselves to measuring the time it takes to execute an algorithm, which is also referred to as the *time* complexity of the algorithm.

Let us begin with a simple example. Suppose we have n numbers that are stored in n registers .x,, x21 ... , x,, . A number stoi ed in a register will be referred to as the content of the register. Without loss of generality, we assume these numbers are distinct. We want to design an algorithm to determine the largest of these n numbers. We can use the following algorithm, which we shall refer to as algorithm LARGESTI.

Algorithm LARGESTI

1. Initially, place the number in register x, in a register called max.
2. For i = 2, 3, ..., n, do the following: Compare the number in register x; with the number in register max. If the number in x; is larger than the

Figure 6.3 : Recognition Result of printed page using sc2



ANALYSIS OF ALGORITHMS 261

between the two vertices, then, of course, the result is a best possible one. On the other hand, consider the nearest-fleighbor ;1lgorithm for the traveling salesperson problem presented in Sec. 5.8. Since the algorithm does not always produce the best possible result, it is extremely desirable to be able to evaluate the worth of its result. We recall that the length of the tour prJduced by the nearest-neighbor algorithm can be measured against the shortest possible tour, as shown in Theorem 5.6. As another example, we recall the job-scheduling algorithm presented in Sec. 4.7. Again, the algorithm does not always produce a best possible schedule. However, we were guaranteed that the fatal execution time according to the schedule produced by our algorithm will never exceed two times lhe shortest possible execution time (Theorem 4.2). Thlnlly, we want to determine the .. cost" of executing the algorithm. Given that an algorithm indeed produces the desired result, we want to know the cost of obtaining the result. The most commonly used measure of the cost of executing an algorithm is the amount of time it takes. However, there are also other measures, such as the memory space required to execute the algorithm.

To study the various aspects of the design and analysis of computing algorithms is a natural topic for us to pursue at this point. On the other hand, we have already given the reader a glimpse of the subject matter. In particular, as was pointed out above, the reader has already been exposed to some considerations on the correctness and performance of algorithms on several occasions. In this chapter, we shall present some of the concepts in connection with the cost of executing an algorithm, not only as an introduction to the subject of analysis of algorithms but also as evidence that the mathematics we learned will indeed help us attack many problems.

8.2 TIME COMPLEXITY OF ALGORITHS

As we pointed out above, we would like to determine the cost of executing a given algorithm. Obviously, the time it takes to execute the algorithm is one of the most important measures of the cost of executil)n. For the rest of this chapter, we shall restrict ourselves to measuring the time it takes to execute an algorithm, which is also referred to as the *time complexity* of tilt: algorithm.

Let us begin with a simple example. Suppose we have n numbers that are stored in n registers $x_1, x_2, \ldots, x_n$. A numb,:r stOied in a register will be referred to as the content of the register. Without loss of generality, we assume these numbers are distinct. We want to design an algorithm to determine the largest of these n numbers. We can use the following algorithm, which we shall refer to as algorithm LARGESTl.

Algorithm **LARGESTl**

I. Initially, place the number in register x_1 in a register called *max*.

2. For $i = 2, 3, \ldots, n$, do the following: Compare the number in register x_i with the number in register *max*. If the number in x_i is larger than the

Figure 6.4 : Recognition Result of printed page using sc3

Page ↓	Total words	Correctly recognised words by sc1	Correctly recognised words by sc2	Correctly recognised words by sc3	Recognition Rate sc1	Recognition Rate sc2	Recognition Rate sc3
1	369	361	364	354	97.8	98.6	95.9
2	493	475	474	457	96.3	96.1	92.7
3	345	322	324	293	93.3	93.9	84.9
4	524	510	504	502	97.3	96.2	95.8
5	400	357	363	334	89.3	90.8	83.5
6	305	300	304	297	98.4	99.7	97.4
7	439	416	427	418	94.8	97.3	95.2
8	509	502	501	497	98.6	98.4	97.6

9	268	255	259	239	95.1	96.6	89.2
10	460	454	458	452	98.7	99.6	98.3
11	621	560	585	567	90.2	94.2	91.3
overall	4733	4512	4563	4410	95.3	96.4	93.2

Table 6.1 : Statistics of printed page recognition

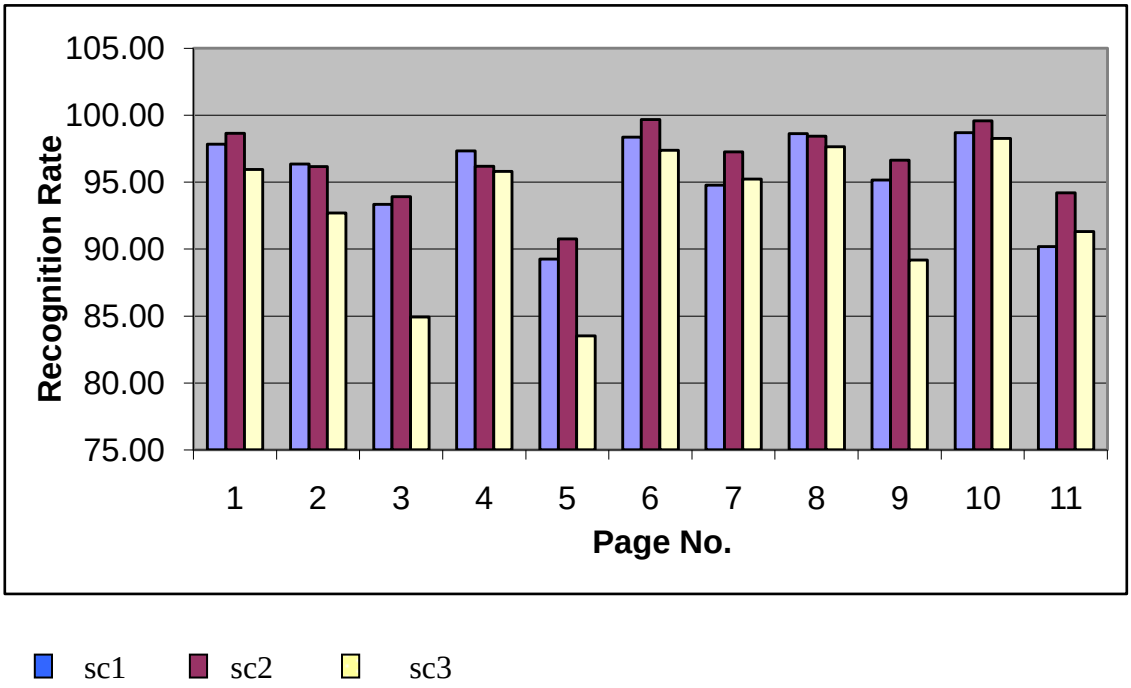


Figure 6.5 : Recognition Rate of printed pages

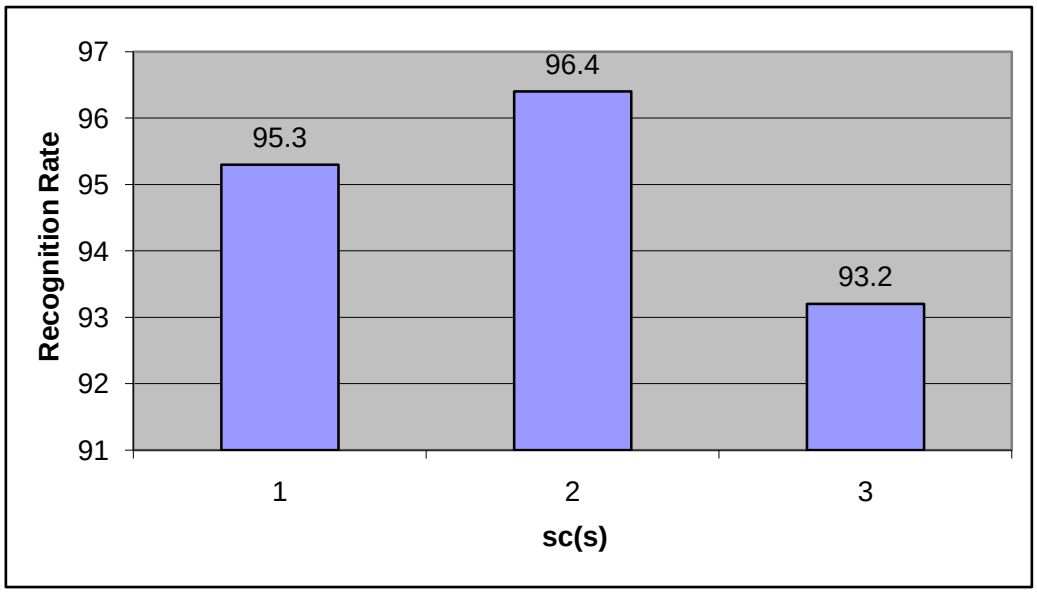


Figure 6.6 : Overall Recognition Rates

In the experiment on printed page recognition using commercial recognition systems, the average recognition rate have been obtained 94.97%. The sc1 could not recognize text printed in columns, correctly. The recognition system was not able to detect properly, the table data and the mathematical symbols.

The experiment clearly shows that the impression created by many papers and patents that recognition problems in case of printed matter recognition have been solved is not fully correct. There is still enough room for progress and further research for improvement in the character recognition is required, particularly in the public domain.

9. Summary and Conclusions

The experiment clearly shows that the impression created by many papers and patents that the recognition problem in case of printed character recognition have been solved is not fully correct. There is still enough room for progress and further research for improvement in the character recognition is required, particularly in the public domain.

In the field of CR several interesting issues still remain open and can be the subject of further research. In future, the focus of researchers will be on the recognition of texts rather than the recognition of single words. The use of language modeling, that has been neglected in the field of offline character recognition, still leaves open many possibilities for further improvement, with particular emphasis through a large body of public research institutions.

REFERENCES

[B03] Bunke, H., "Recognition of Cursive Roman Handwriting – Past, Present and Future", *In Proceedings of the 7th International Conference on Document Analysis and Recognition (ICDAR'03)*, volume 1, pages 448–459, Edinburgh, Scotland, 2003.

[K04] Kuncheva, L. I., "Combining Pattern Classifiers: Methods and Algorithms", *Wiley-Interscience*, 2004.

[KHDM98] Kittler, J., Hatef, M, Duin, R. P. W. and Matas, J., "On combining classifiers", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.20, no. 3, pp. 226–239, 1998.

[L99] Lorette, G., " Handwriting Recognition or Reading? – What is the Situation at the Dawn of the 3rd Millenium?", *Int. Journal on Document Analysis and Recognition*, vol. 2, no. 1, pp. 2-12, 1999.

[R89] Rabiner, L. R., "A tutorial on hidden Markov models and selected applications in speech recognition", *Proceedings of IEEE*, vol.10, no. 2, pp. 257–286, 1989.

[RJ86] Rabiner, L. R., and Juang B. H., "An introduction to Hidden Markov Models", *IEEE ASSP Magazine*, pp. 4-16, 1986.

[SS13] Singh R., and Sharma P. "English Character Recognition System : Digitization of Printed Documents", *International Journol of Computer& Organization Trends*, vol. 3, issue 6, pp. 231-234, 2013.

[S93] Srihari, S.N., "From pixels to Paragraph : the use of Contextual Models in Text Recognition", In *International Conference on Document Analysis and Recognition*, Japan, pp 416-423, 1993.

[VBB04] Vinciarelli, A., Bengio, S., and Bunke, H., "Offline Recognition of Unconstrained Handwritten Texts Using HMMs and Statistical Language Models", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 709-720, June 2004