Web Intelligence and Security
M. Last and A. Kandel (Eds.)
IOS Press, 2010

doi:10.3233/978-1-60750-611-9-83

Toward Real Time Internet Traffic Monitoring*

Benny Porat and Ely Porat
Bar-Ilan University Bar-Ilan University

Abstract. Pattern matching is one of the basic topics in computer science. There were two waves of tens research into pattern matching. Due to new problems and changes in the model as well as its practical application, we must now examine pattern matching problems in the context of new models. Such considerations give rise to many interesting research problems in this area. The main difference between the last two waves of pattern matching research and the new wave is the need to deal with more practical questions. Such problems are encountered in industry applications first, while in the past the researchers had suggested the problems themselves. Two papers [4, 16] could be already tagged as the beginning of the new wave. We discuss the new challenges that must be handled in these exciting future research directions.

Keywords. Pattern matching, string searching, streaming algorithms.

1. Brief History

1.1. First Wave

Pattern matching is one of the oldest areas in computer science. It started in the mid 60's as an interesting research topic on its own. Although hard drives had already existed then, the biggest hard drive size was about 10MB, and only few big companies could afford it. Thus, the input sizes for many problems did not exceed over several kilobytes, and the interest in pattern matching was purely theoretic. Research at that time includes works such as Levenshtein [**?**] definition of edit distance, Karp Miller Rosenberg [11] renaming technique, Fisher Paterson [10] algorithm for pattern matching with wildcards based on convolutions, Knuth Morsis Pratt [12] first linear time algorithm for exact pattern matching, Bird [6] and Baker [5] 2-dimensions pattern matching algorithm, Boyer Moore [7] first exact pattern matching which ran in sub-linear time on average, and Weiner [18] and McCreight [15] suffix tree. Six Turing award winners were among the researchers who worked in the area. Pattern matching was one of the "hottest" research areas in those days. However as time passed, most of the interesting problems were solved and new unsolved problems, were considered extremely hard to solve. Therefore most of the researchers had moved to other areas.

*This work was supported by BSF and ISF.

1.2. Second Wave

During the late '80s there was a new wave of interest in pattern matching, due to new applications. People began to buy personal computers, and the number of computer applications rose quickly. Businesses uploaded their data bases into electronic storage. This had led to the question of efficient indexing of the data. Ukkonen [17] presented his linear time construction for suffix tree and Manber and Myers [14] invented the suffix array. Storage was expensive, therefore people began to use compression. This opened the question of how to search in a compressed data, introduced by Amir and Benson [1]. Progress in computer vision led to the natural question of the theoretical foundation of low-level vision as 2-dimensional pattern matching [2]. The interest in parallel computing also gave rise to novel parallel algorithms for pattern matching [13]. Many good researchers put effort in pattern matching research. In addition to solving the problems inspired by the new technological development, they searched for more problems that could be solved by the new tools of the galvanized pattern matching field. It turned out that pattern matching was extremely useful for solving problems in biology introduced by the new sequencing techniques. Researchers studied these nascent biological problems in greater depth and this led to the new and, today, highly successful field of "computational biology". In its inception, computational biology was, indeed, almost a subfield of pattern matching. Its top research was presented in the pattern matching conferences, and many pattern matching researchers constituted the backbone of computational biology research. As computational biology developed, it gradually diverged from the pattern matching community. Two new computational biology conferences were opened (WABI and RECOMB) and this led again to a temporary waning of interest in pattern matching.

2. The New World

A new wave of pattern matching research is emerging today. The Internet brought us a variety of hard new challenges. New problems in areas such as Musicology, Audio and Video streaming, Audio and Video Indexing, Meteorology, or Economics have appeared and require new techniques. The data size of modern problems is bigger by several order of magnitude, therefore a lot of the old solutions need to be revised. Furthermore, these areas introduced new models of computation like the External Memory model, Streaming, the Constant Passes model, and more.

One major difference between the classical and new models is that in the current setting the size of the input is enormous relative to the available Random Access Memory. For example, the number of pages in the World Wide Web has already exceeded twenty billion, and is still growing exponentially. Major phone companies, like AT&T, collect tens of gigabytes of call data every day. The sequenced human genome consists of about 3 billion base pairs. Billions of dollars have been invested by the US defense department in techniques for searching massive data bases. Trends are towards storing and managing more types of data (audio, video, large texts, etc.) and massive amounts of each.

The vast amount of collected information is of little use without efficient methods which will facilitate the efficient processing and analysis of the data. The goal of designing algorithms which solve such data processing problems has been a focus of computer

science since its beginnings. However, the definition of what constitutes an "efficient" algorithm changes drastically, when one attempts to process terabytes of data. For example, the sorting algorithm that takes $O(n \log n)$ time makes too many passes over the data and is therefore expensive. Moreover, for many years, linear running time for solving a problem was considered the ultimate efficiency goal. Today it could easily mean that the algorithm might take several days to finish. Alternatively, the running time of a new algorithm, which requires only linear space, could easily become unacceptable if the required amount of memory is not available; this applies to cases when the generated data is much too voluminous to even store or archive, as in IP traffic analysis at the level of IP packets. In addition, it is beneficial to impose further restrictions on the way the computation must be executed. For example, in essentially all secondary storage devices, the cost of accessing a random element of a data set is much higher than the cost of accessing the "next" element.

Ideally, one would like to perform the computation in only one pass over the data. While it may not be prohibitively expensive to make two or even a small number of passes, it is nevertheless a good guiding principle to perform as much of the computation as possible in one pass. Thus there is a need to develop new methods for data processing and analysis, which use sub-linear amount of space or time resources and make as few passes over the data as possible. We call such algorithms sub-linear. These algorithms either can not afford operating at every item in the input (hence must use time sub-linear in the input size to do their computation) or can afford to process each input item, but can not afford to archive the entire input and post-process it (and hence must use space sub-linear in the input size). These models raise the need to develop entirely new methods.

3. Current State of the Art

3.1. Space Lower Bound for Deterministic Online Approximate Matching

It would seem desirable to reduce the space requirements even further in order to increase the practicality of processing data streams. Unfortunately, there is an $\Omega(m)$ communication complexity lower bound for “sum-type” functions which covers the additive distance functions we have been most interested in. The space lower bound for any deterministic approximate pattern matching algorithm follows directly from the communication complexity lower bound by a standard argument that we briefly summarize. Assuming the communication complexity lower bound, the proof of the space lower bound is by contradiction. If Alice can preprocess the pattern to use $o(m)$ space and then starts her online pattern matching algorithm, she could then transfer a snapshot of the current state to Bob who could then carry on running the algorithm on his string. Bob would then find the distance to his string having received only $o(m)$ items of data, thereby giving the desired contradiction.

3.2. Special Hardware

Todays problems are very big and hard, in most of the cases there are no efficient algorithmic solutions. Therefore it lead people in the industry to be creative and use special hardware. There are several examples of such special hardware machines: HP and Cray

released a hybrid computer which includes FPGA accelerators; IBM released a computer which contains cell processors; Intel is about to release some day their 80 core CPU; Several companies now use GPU's to accelerate their algorithms, and some companies use TCAM inside their computers.

Each hardware architecture has its own advantages. The main disadvantage of using a special hardware is the cost. Usually companies will use off shelf products and they won't develop there own ASIC. However it is still very costly to develop software for those special hardware. We need a different specialist for each different hardware. Section 3.4 discusses how such new hardware could be used to solve new pattern matching problems.

3.3. Algorithmic Point of View

There are two models which emphasis the problems in our current situation. The first one is the online model.

3.3.1. Online Pattern Matching Algorithms

Online pattern matching is a specific pattern matching model, where the input is received character by character, and the algorithm is required to generate the answer before receiving the next character. The state of the art in this field is my paper on a black box for online pattern matching algorithms [4]. In this paper we gave the following result: Given a pattern of length m and a streaming text of length n that arrives one character at a time, the task is to report the distance between the pattern and a sliding window of the text as soon as the new character arrives. Our solution requires $O(\Sigma_{j=1}^{\log_2 m} T(n, 2^{j-1})/n)$ time for each input character, where $T(n, m)$ is the total running time of the best offline algorithm. The types of approximation that are supported include exact matching with wildcards, matching under the Hamming norm, approximating the Hamming norm, k-mismatch and numerical measures such as the L_2 and L_1 norms. For these examples, the resulting online algorithms take $O(\log^2 m)$, $O(\sqrt{m \log m})$, $O(\log^2 m/\epsilon^2)$, $O(\sqrt{k \log k} \log m)$, $O(\log^2 m)$ and $O(\sqrt{m \log m})$ time per character respectively. The space overhead is $O(m)$ which we show is optimal.

Despite such problems being very important theoretically, and critical for many applications for processing massive data sets, little progress has been made in this field. The most prominent work in this field has been the KMP algorithm, which solves the exact search problem in the online model. However, almost no further progress has been made regarding similar problems, until my work has shown that almost any pattern matching algorithm can be converted to the online model, at the cost of an additional $\log m$ factor. Yet, there are many questions that are still unanswered, and many directions for research exist. Research in the new wave of pattern matching must continue to explore this field, and to research and develop principles and methods for online pattern matching. Such research should address still unsolved pattern matching problems, seek to reduce the $\log m$ factor incurred by the algorithm of [4] when moving to the online model, and should also prove several lower bounds. Future research must explore performing convolutions in the online model, and various pattern matching extensions in the online model, including searching with wildcards and building dictionaries with wildcards, and searching when allowing a small amount of edit operations. Some of the prominent problems to examine are:

Online convolutions Convolution is a basic tool in pattern matching. Therefore it is very important to find an optimal solution for it in the online case. Past research [4] shows how to perform an online convolution in $O(\log^2 m)$ time per character, while offline convolutions can be done in time $O(\log m)$, amortized per character. It is critical to derive a lower bound to online convolutions or a better upper bound.

Searching with wildcards Online exact string searching has been studied for over half a century. However solutions for online exact string searching with wildcard are very recent [4]. Exact searching with wildcard is one of the most important questions, both in theory and in practice. The current solution runs in time $O(\log^2 m)$ per character. Future research should aim to define the right complexity for exact string searching with wildcards, and should develop a better algorithm and prove some lower bounds.

Searching with small amount of edit operations Two algorithms deal offline with a small a distance. Cole et als algorithm [9] solves edit distance in linear time in the cases where we seek positions with a small enough distance, the algorithm [3] calculates the hamming distance in all of the positions where the distance is bound by a small number in linear time. Future research should attempt to design similar algorithms that work online.

Dictionary with wildcards Dictionary matching with wildcards is the one of the most important open question in pattern matching. It is defined as follows: given a set of patterns which might include the wildcard symbol, we must pre-process them and then scan a text character by character and raise a flag whenever we see one of the patterns appearing in the test. There exists a solution for the problem without wildcards, based on suffix trees. There is a non practical solution to the case that the number of wildcard is bound. Future research should develop better algorithm for this problem, possible using techniques for online pattern matching algorithms. Relaxed versions of regular expressions Regular expression searching is one of the oldest open problems that are not believed to be solved in the near future. A key direction for future research is to define relaxations to the problem and attempt to solve them in the online manner.

3.3.2. Streaming Pattern Matching

Research for the new wave of pattern matching algorithms must focus on newer models, that are better adapted to today's problems. Such research should focus on the **streaming model** itself. Novel research must fully define various alternative streaming models, formalizing possible restrictions on the algorithms and the operations they are allowed to perform. The problems that were handled in the past must be addressed in both standard streaming models, such as search and its extensions, dictionaries and convolutions, and also in alternative streaming models, such as a model allowing several passes over the input, augmentations with primitives and other new computational models. In order for the new research to provide fruitful results, it is not enough to examine models which reflect various restrictions inspired by applications. Strong contribution in this area must also take a theoretical approach in analyzing these restrictions. In other words, successful new work should also be viewed as a basic research of computational models and how they impact pattern matching problems. A key step in such research is researching and developing principles and methods in models related to streaming:

Streaming model this model is defined for cases where there is not enough memory (or storage space) to hold the entire input. Thus, the input is examined sequentially, character by character, and each character is processed as it is received. Algorithms in

this model have sub-linear storage, and typically use only poly-logarithmic space. The streaming model has been examined by some of today's most active researchers, but there are still many very fundamental open questions. One major of further research is to solve several fundamental problems in the streaming model. Specifically, a key step in such research is addressing problems of monitoring and searching in streams. The following problems are of particular interest:

Exact Search - Given an input text T of length n, and pattern P of length m, the goal is to find all occurrences of the P in T using memory that is only poly-logarithmic in the size of P. In a previous work on this problem we present in [16] a fully online randomized algorithm for the classical pattern matching problem that uses merely $O(\log m)$ space (i.e. streaming algorithm). this problem was very interesting from a theoretical point of view since for a long time it seemed impossible to break the $O(m)$ barrier. In addition, it can be used as a tool in many practical applications, including monitoring Internet traffic and firewall applications. In this model we first receive the pattern P of size m and preprocess it. After the preprocessing phase, the characters of the text T of size n arrive one by one in online fashion. For each index of the text input we wish to indicate whether the pattern matches the text at that location index or not. Clearly, for index i, an indication can only be given after all characters from index i till index $i+m-1$ have arrived. Our goal was to provide such answers using minimal space, and spending a minimal amount of time on each character (time and space which are in $O(poly\log(n))$).

We present an algorithm whereby both false positive and false negative answers are possible with a probability at most $\frac{1}{n^3}$. Thus, overall, the correct answer for all positions is returned with a probability of $\frac{1}{n^2}$. The time which our algorithm spends on each input character is bounded by $O(\log m)$, and the space complexity is $O(\log m)$ words.

In addition we present a solution in the same model for the pattern matching with k mismatches problem. In this problem, a *match* means allowing up to k symbol mismatches between the pattern and the subtext beginning at index i. We provide an algorithm in which the time spent on each character is bounded by $O(k^2 poly(\log m))$, and the space complexity is $O(k^3 poly(\log m))$ words.

We believe this is the best algorithm one could hope to achieve. Although this might actually be the lower bound for this problem, most inputs can be handled faster and with less memory consumption. An important open question is to prove that lower bound, and to find the better algorithms which can be run on most of the inputs.

Searching up to k mismatches - In this problem we are given an input text T of length n, pattern P of length m, and input number k. The aim is to find all occurrences of the P in T with up to k mismatches, using memory which is only poly-logarithmic in the size of P. Similarly to the above problem, previous work [16] suggested an algorithm that uses only $O(k^3 \log^5 m)$ bits of space and $O(k^2 \log^3 m)$ time per character time. However, several techniques may be used to to improve that algorithm: Filtering (as presented in [3]), Selectors, Sketching based coding theory [8]. Another important step is proving a lower bound for that problem, and an algorithm that will be better for most of the inputs, as described in the Exact Search problem above.

Search with wildcard - In this problem the pattern P can be given with special wildcard characters which match any letter in the text T. The output should be all occurrences of P in T, taking into account that special character. One can prove the $O(|P|)$ (where $|P|$ is the length of the pattern P) is a memory lower bound, which makes this problem irrelevant in the streaming model. However, there are several restricted cases where one

can solve that problem even with poly-logarithmic memory, hence enabling to solve this problem in the streaming model.

Dictionary Search - In this problem we are given a list of words. Under the streaming model definition, we want to be able to find all occurrences of any of the words using only one pass over the data and as little memory as possible. There are currently no known solutions for this problem in the streaming model.

Approximation for pattern matching - The problem is to approximate the hamming distance for any text location to the pattern. As in the previous problems, one can prove that a complete A possible direction is to develop a randomized solution, such as Karloff, for this problem. Convolution – As mentioned before, convolutions are fundamental tool in the pattern matching area. One can prove that $O(|P|)$ is a memory lower to calculate the convolution (similar to above lower bound). We plan to define the cases where this can be done even with less memory, which will enable the use of that tool in the streaming model, where possible.

3.3.3. Other Models

Some of the problems we might prove to be unsuitable to the streaming model, or sometimes we would prefer a better algorithm. Thus, the following models should be considered as well:

Few passes model – One extension of the streaming model is the few passes model. As opposed to the streaming model, where input is read only once, in the "few passes" model, input is read "few" times. The goal in this model is to reduce the number of required input accesses, and, as in the streaming model, to use as little memory as possible. The "few passes" model is well motivated from problems of reading tape devices (which are very slow). Only few systems use tape devices today, but there are many other similar examples, such as reading the data from a remote server when the communication channel is slow, etc. In such examples it is easy to examine the input sequentially (and in the tape example even read it again in reverse order), but a second pass over the input is as costly as reading it again, so it is desirable to read the input as few times as possible. Further work should study the limitation of this model with respect to pattern matching. The first basic question is to find when this model gets better result from the streaming model in terms of memory and time requirements. For example, it is known that if we allow constant memory, the performance in both models will be the same.

Streaming Model Augmented with a Primitive – Even though most of the model is known, in pattern matching this model has not been referred to. Proposing algorithms in the "few passes model is only a first step in designing a full solution. To examine better the difficulty of the problem, researchers have introduced the model of "few passes with sorting", i.e. with the ability to sort the input so in the next pass the input would be sorted in some chosen order. In this model the goal is to minimize the number of sort phases required to solve the problem. This model can become practical using the support of a special hardware accelerator, which would perform the sorting phase. Future research should use this sort primitive in our few passes pattern matching algorithms as well as seek for other primitives which could be supported in practice using optimized hardware.

Those models are very important for many application such as real-time Internet monitoring, Stock exchange automatic trading systems, Antiviruses and more. As I mentioned earlier in the online model there exist a novel way [4] to transfer almost all pattern

matching offline algorithm to work online, which give a feasible solution to some of the problems.

3.4. Fusion with Special Hardware

All of the models mentioned above would certainly lead to very interesting problems, which could be solved using methods of the new wave of pattern matching. Some of the models consider certain strong operations, that could be supported in practical applications by special hardware.

As mentioned above, such hardware is being currently developed in several industry firms, such as HP and Cray's hybrid computers (with FPGA accelerators), IBM's machines with many Cell processors, Intel's 80 core CPU, or GPUs used to accelerate certain algorithms.

Each of these hardware architectures has its own advantages; therefore there is a need to define appropriate models to produce the maximum efficiency from the algorithm written for each such specific architecture. For example FPGA is a programmable electronic circuit. It has the advantage of massive parallelism like other circuits, as well as the ability to change the program like a computer that other circuits do not. Therefore new models should consider hardware with a circuit model capable of changing once in a while (reprogramming takes time and that should be taken under consideration).

Some of the special hardwares will be come a standard. For example you can find GPU's in almost every computer, and there are companies which working on standards for GPU's. Given a specific special hardware we will have to develops models which use the advantages of the special hardware. After define those new models it will be simpler to develop efficient algorithms which will be able to use the advantages of the special hardware.

Since such special hardware is becoming more and more common, it should not be ignored, and computational models based on such operations should be examined, especially in the context of the new pattern matching problems, which could use such hardware to tractably solve problems that could not be handled before.

4. Measurement and Analysis Models

Although the model of permitted operations is clearly very important, various applications require different types of analysis of the quality of the proposed algorithms. Some approaches, for example, perform extremely well on almost all inputs, but perform very badly on few very rare inputs. If such inputs never occur in practical applications, the worst-case analysis may give the wrong results. The problem if measurement and analysis is very important to the new wave of pattern matching research, as these problems have specific characteristics that must be taken into account.

A core objective is to achieve algorithms with better upper bounds, as well as prove lower bounds. Proving lower bound is very important from theoretical point of view to the understanding of the limitation of each model, however it has no practical help when the need is to find a way for solving such problems. Therefore future research should not stop at proving lower bound and must also find way to bypass the lower bounds by using other measurements.

We believe that although the previously defined research directions are hard, in many cases the input to the problem could be restricted in a way that allows tractably solving the problem, while providing a solid basis for practical applications. Thus, research should not focus only on worse case analysis. The problems should be addressed using specific distribution analysis (like zipf) as well as the types of analysis:

Smooth analysis: There are some problems that are proven to be difficult to solve or very costly in terms of both computation time and storage space. Most of the hardness proofs use a specific input for which it is possible to show that any algorithm that one may conceive must use many resources (time or space) to solve the instance. However, in many cases these are extremely rare inputs, which are very unlikely to occur in practice. Thus, some algorithms may function very well in general, except for these extremely rare inputs. Future research should discover such cases and to develop new algorithms for "hard" problems that do well on almost all inputs.

One way to analyze the complexity of such algorithms is by using smoothed analysis. Smoothed analysis is a hybrid of worst-case and average-case analyses that inherits advantages of both. The smoothed complexity of an algorithm is the maximum, over its inputs, of the expected running time of the algorithm under slight random perturbations of that input, measured as a function of both the input length and the magnitude of the perturbations. If an algorithm has low smoothed complexity, then it should perform well on most inputs in every neighborhood of inputs. Future research should explore the use of the smoothing over massive graph problems. There are many massive graph problems, in various models like the streaming and external memory models that are considered very hard to analyze. As an example consider the problem of clustering internet blogs or finding connected components in an online graph where we are only allowed to use very limited (poly-logarithmic) space and time for processing. The worst case bounds for these problems under these models are not applicable. Using smoothed analysis techniques one might produce a different perspective for solving these problems or help to provide better results.

Competitive analysis: The current prevailing definitions of an optimal algorithm are those that consider how the algorithm works on the worst possible input, but in many cases such inputs are very unlikely to occur. One cannot conclude that Algorithm A is better than Algorithm B simply based on examining the worst case behavior of both.

An alternative analysis is competitive analysis, which can be used to design algorithms that perform well relative to the optimal algorithm designed specifically for that input. In some cases, it is impossible to achieve this. For example, consider the problem of searching for an item in a sorted array. Given a specific input, a search query, the best algorithm for that specific input would first examine the place where the item appears on this specific input, and only after that perform a linear probing (which is not optimal at all). This algorithm would retrieve the answer with a single probe. Thus, the time complexity per any query should be $O(1)$. However, it is easy to see that any algorithm requires at least $\frac{1}{2}\log n$ on most queries. This example explains why a new notion of competitive algorithms must be developed, in which we compare our algorithm to the best algorithm which has some information known about the input. Future works should develop such an analysis framework and use it to tackle very famous open problems, such as the Splay tree conjecture.

5. Conclusions

There are many open research questions in the area of pattern matching, inspired by practical problems that industry firms face. Such problems are becoming quite common, and cannot be ignored. Certain key pattern matching problems in the streaming models stop us from taking the next big leap in both theory of computer science and in leveraging such theory for practical applications. Although industry firms have managed to bypass the problems using special hardware, such steps are typically "patches" that only bypass the problem.

Future research for the new wave of pattern matching must methodically investigate such models, and possible solutions. The most prominent problems in this area were outline in this paper. They focus both on specific problems, algorithmic models, models of analysis and measurement, and fusion with special hardware solutions. This area is certainly full of exciting future research directions.

References

[1] Amihood Amir, Gary Benson: Efficient Two-Dimensional Compressed Matching. Data Compression Conference 1992: 279-288

[2] Amihood Amir, Gary Benson, Martin Farach: Alphabet Independent Two Dimensional Matching STOC 1992: 59-68

[3] Amihood Amir, Moshe Lewenstein, Ely Porat: Faster algorithms for string matching withkmismatches. SODA 2000: 794-803

[4] Rapheal Clifford, Klim Efremenko, Benny Porat, Ely Porat: A Black Box for Online Approximate Pattern Matching. In CPM 2008: 143-151

[5] Brenda S. Baker: A theory of parameterized pattern matching: algorithms and applications. STOC 1993: 71-80

[6] Richard S. Bird: Two Dimensional Pattern Matching. Inf. Process. Lett. 6(5): 168-170 (1977)

[7] Robert S. Boyer, J. Strother Moore: A Fast String Searching Algorithm. Commun. ACM 20(10): 762-772 (1977)

[8] Raphal Clifford, Klim Efremenko, Ely Porat, Amir Rothschild:k- Mismatch with Don't Cares. ESA 2007: 151-162

[9] Richard Cole, Ramesh Hariharan: Approximate String Matching: A Simpler Faster Algorithm. SODA 1998: 463-472

[10] M.J. Fischer, M.S. Paterson, String matching and other products, SIAM-AMS Proc. 7 (1973) 113-125.

[11] Richard M. Karp, Raymond E. Miller, Arnold L. Rosenberg: Rapid Identification of Repeated Patterns in Strings, Trees and Arrays STOC 1972: 125-136

[12] Donald E. Knuth, James H. Morris Jr., Vaughan R. Pratt: Fast Pattern Matching in Strings. SIAM J. Comput. 6(2): 323-350 (1977)

[13] Gad M. Landau, Uzi Vishkin: Introducing Efficient Parallelism into Approximate String Matching and a New Serial Algorithm STOC 1986: 220-230

[14] Udi Manber, Gene Myers: Suffix Arrays: A New Method for On-Line String Searches. SODA 1990: 319-327

[15] Edward M. McCreight: A Space-Economical Suffix Tree Construction Algorithm. Journal of the ACM (1976) 23 (2): 262–272. doi:10.1145/321941.321946.

[16] Benny Porat, Ely Porat: Exact And Approximate Pattern Matching In The Streaming Model. To appear in FOCS 2009

[17] E. Ukkonen: On-line construction of suffix trees. Algorithmica (1995) 14 (3): 249–260. doi:10.1007/BF01206331.

[18] Peter Weiner: Linear pattern matching algorithm. 14th Annual IEEE Symposium on Switching and Automata Theory.(1973) pp. 1-11.