DOI 10.29042/2017-1783-1789

Helix Vol. 7(4): 1783-1789

Exact and Approximate Repeats in DNA Sequences using Suffix Array and Divide & Conquer

***[1]Raju Bhukya, [2]Priyanka Agrawal, [3]Rajmani Arya, [4]Shekhar Prasad**
NIT Warangal
Email: drrajunitw@gmail.com

Received: 10th June 2017, Accepted: 20th June 2017, 1st July 2017

Abstract
Microsatellites or Simple Sequence Repeats are short motifs, for the most part 2-8 base pairs long that are repeated numerous times in genomic DNA. Here, motifs are short, repeating patterns in DNA that have an organic capacity. These SSRs are responsible for sequence-binding locations for proteins like Transcription factor. Such groupings are extremely normal over the eukaryotic genomes and are found in plenitude with variable recurrence of event. In the event that the rates of transformations at the repeat loci is observed to be high contrasted with different locales of genomic DNA it can often lead to generation of inter and intra-specific hereditary variety, and thus permit their hereditary marker exploitation. Recent evidences demonstrate the auxiliary and utilitarian essentialness of microsatellites and have consequently made it an imperative subject in contemporary research in addition to their time–proven use as an efficient molecular marker system. Taking after the transformation in sequencing advances, even the extent that non model life forms are concerned, the whole genomes can now be possibly screened utilizing bio tools to locate the presence of SSRs.

I. INTRODUCTION

Repeats in DNA successions comprise of short arrangements or examples that repeat. These repeats can be either scattered or tandem. Blended repeats are long, less incessant and far separated while tandem rehashes are continuous events of an example that are contiguous each other. These repeats examples are either precise or there might be inclusion or deletion or substitution of a base, because of transformations, as needs be there are correct and surmised pair rehashes. Pair repeats are additionally grouped into microsatellites, mini-satellites and satellite DNA in light of the time of the repeat. Rehashes comprising of under 1 to 6 nucleotides are alluded to as microsatellites. Small scale satellites are those that have a period estimate more prominent than 6 nucleotides and satellite DNA have more noteworthy than 100 nucleotides. Tandem repeats are identified with numerous human illnesses brought about by hereditary issue, for example, Huntington's infection, delicate X mental impediment, 'myotonic dystrophy', strong decay and "Friedreichs ataxia". These repeats additionally help in deciding the parentage and individual legacy qualities. Certain adjustments in the recurrence of these repeats can bring about tumor. Blemished preservation of examples and the perplexing structure makes it hard to distinguish tandem repeats. Also the yield gotten for a given information shifts from technique to strategy. Most tandem repeats finders create distinctive, frequently non overlapping surmising, reflecting attributes of the fundamental calculations as opposed to the genuine circulation of Tandem Repeats (TRs) in DNA sequences.

Two types of sequence repeats are -

Perfect Sequence Repeats: A exact tandem repeat is a sequence A = (B) t which is a continuous copies of two or more sequence *B* of length *p* where $p \geq 2$ is repeating ***t*** times. The sequence B is called the repeating pattern of A.

Approximate Sequence Repeats: Due to gene mutation, sequence repeats in DNA generally don't consist of perfectly repeated patterns. Each repeating unit are different from other or it may be the same and different. Since there is more than multiple way to distinguish between repeated units (e.g edit distance, hamming distance or statistical methods), multiple definitions of approx. tandem repeats exist.

A. Sequence Repeat Searching tools

Given the considerable interest in tandem epeats, it is hardly surprising to find a rich abundance of bioinformatics tools developed for detecting and characterizing them. Two surveys have founded more than 30 tools TRF, MREPS , Sputnik, ATR Hunter, Repeat Masker etc.

B. Repeat Detection

The majority of the search tools embrace a two-stage way to deal with identifying "tandem repeats": search and filtering. In the main stage, either one of the two sorts of algorithms is ordinarily actualized:

1. A combinatorial search algorithm that comprehensively areas the whole sequence into subsequence's and compares about each contiguous subsequence to recognize potential sequence repeats for all positions and all period sizes.

2. A measurable or heuristic based algorithm that uses little section windows to look over the sequence to first identify the conceivable repetitive loci (regularly little exact repeats) before combine them to decide the more extended repeats. The runtime for the combinatorial algorithm grows exponentially, particularly when the search includes imperfect and covering repeats.

C. Significant repeats filtering

In next step, different types of filtering techniques are utilized to recognize and remove the naturally noteworthy repeats. It can be a straightforward length channel or a more intricate execution of heuristic-based, (for example, k-mean clustering) or measurements based models. Likewise, arrangement algorithms, for example, wraparound dynamic programming or star-focused algorithm are additionally utilized to decide the imperfect repeats in view of the given insertion and deletion (indels) or mismatch with limit settings. For the vast majority of these tools, the clients are permitted to change the filtering parameters to satisfy their articular application however the announced outcomes are exceedingly delicate to these settings. In addition, distinctive developers utilize or prescribe diverse settings to identify what they consider as "biologically significant repeats", subsequently making it troublesome for clients to comprehend their complicated impacts.

II. Related Work

A. Algorithmic performance evaluation systematically

This is compulsory to minimize the biases caused by the different parameter settings of individual tools to assess the algorithmic performance of the tools and understand their inherent constraints. A standard approach is applied for which we first compute the tool's' performance under default settings, sequentially filtering for non-redundant and exact repeats. In a second run, to maximize the detection of exact repeats, we valuate the tools' performance using parameter settings. Exact repeats, being precise and not subjective to varying interpretations, rovide a good model for studying the intrinsic tool performance. For example, there is only one precise interpretation of this repetitive sequence (CGGTACGGTACGGTA) for an exact repeat, which is (CGGTA)3 . This makes it simple to control and standardize the parameter settings when making tool comparisons, which are normally limited to repeat length and period size. Alternatively, the interpretation of approx. repeats is highly variable and depends also on the mismatch and indel parameter settings. It is difficult to find suitable settings among so many arameter to minimize their influence for approximate repeat search so as to have an objective comparison.

B. Extraction repeats and comparison

A filtering process is used to retrieve only the required repeats from the collected results for comparison. In the case of TRF, to extract the perfect repeats, the results are filtered with a indel percentage of zero and mismatch percentage of 100% and. Similarly, for ATR Hunter, only repeats with the score of one which indicates perfect match are considered. Since it is difficult to establish a good benchmark for the amount of appropriate overlap, and not all the tools handle the overlapping repeats in the same manner, we only considered non-overlapping epeats to retain consistency for software comparison. Here, non-overlapping repeats refer only to repeats ending position of the previous repeat lesser than with starting position of current repeat.

C. Maximized perfect repeat detection Settings

For the next run, for mreps, the exact repeat search was maximised by simply setting the resolution parameter to zero; for T-reks, the similarity level was set to 100% with indel percentage set to zero; and for iMEx, INVERTER and Sputnik , the respective option to perform exact repeat search was selected(for Sputnik simply include '-r 0' in the command line argument). But not all the tools have the option to select exact repeat search directly. For those that could not, the match score, mismatch and insertion and deletion(indels) penalties and the minimum score were adjusted to maximize the search for exact repeat with lowest sieving of the exact repeats. In TRF, this was set by adjusting the alignment weights to the extreme values of 2, −50 and −50 for match score, mismatch penalty and indels penalty, respectively, and the lowest score was pushed down to zero to allow discovering of all repeats detected indiscriminately . Similarly, the match score, mismatch penalty, indel penalty and minimum score for ATRHunter were set to 1, −50, −50 and 50, respectively.

III. Proposed Work

A. Suffix Array Based Approach for Exact Repeats

Suffix Array(SA)

It is an array of integers, which provides the starting positions of suffixes if arranged in lexicographical or sorted order. Suffix array can replace suffix tree with help some additional information. Manber & Myers (1990) introduced the suffix array first time, to improve over the suffix trees: Suffix arrays contains M integers for M length string or text . If integer type size is 4 bytes then a suffix array need 4*M bytes memory.

Longest Common Prefix(LCP)

Integer array consists of no. of common prefix between two consecutive suffixes according to

"suffix array". For efficient implementation see Kasai et. al .

Locating Sequence Repeats using SA and LCP

We described a simple algorithm for finding SSRs in a nucleotide sequence in linear time and space using suffix and longest common prefix arrays. The algorithm makes no difference between micro-satellites or "minisatellites" and can find sequence repeats of any length. It can also find maximal length repeat. To find the SSRs parameters k, R and P from the suffix array(SA) and longest common prefix array(LCP) is used, where R = Total number of times sequence is repeating (after the 1st occurrence), k = the length of an SSR repeating unit or period size, P = The index position of the 1st occurrence of the SSRs.

Algorithm 1. Algorithm for Finding SSRs for a given DNA sequence

Input: dna_sequence is a char sequence of {A, C, G, T} characters

Output: Positions and length of exact repeats

1. SA = **SuffixArray**(dna_sequence)
2. LCPA = **LCP**(SA, dna_sequence)
3. **for** i:= 1 to length(dna_sequence) **do**
4. K_i = abs (SA_i – SA_{i-1})
5. R_i = floor ($LCPA_i$ / K_i)
6. P_i = min (SA_i, SA_{i-1})
7. **end**
8. **for** i := 1 to length(dna_sequence) **do**
9. **if** R_i > 0 **then**
10. print P_i, R_i + 1, K_i // position, number of repeats , motif length
11. **end**
12. **end**

For a given sequence suffix array is construct in linear time[5]. With SA and sequence, LCP can be created also in linear time Kasai et. al. After computing SA and LCP, we are calculating difference of adjacent value of SA which gives us a unit motif length. Repeating frequency is calculated by taking floor of LCP[i] to K[i]. Starting position is determined by taking minimum of two adjacent SA values. With K, R, P array repeats are calculated as if R is greater than 0, it is a repeat of length (R[i]+1) * K[i] whose repeating unit length is K[i].

B. Divide and Conquer Based approach with Hamming Distance for Approx. Repeats

Hamming Distance

Hamming Distance between a pair of sequences is defined when there length are equals. The Hamming distance is the number of mismatches when the two strings are aligned character by character. We applied hamming distance on divide and conquer algorithm. While calculating largest common extension. There are others alternatives for hamming distance are edit distance, Euclidean distance and mahalanobis distance.

Now we are at center of sequence, we assume that every repeat always goes through a center of sub problems so we consider two cases- first, repeating unit length 'l' is left side of center then for given repeat length range we check longest common extension to left side = l1 and longest common extension right side = l2, now total repeat length would be l1 + l2 + l. second case, when repeating unit is considered right side of center then for given repeating unit length we perform the same backward and forward longest common extension to calculate total repeating length.

The repeats that span XY are classified into two groups according to where their centers lie. A right repeat has its center at or to the right of the boundary between X and Y. A left repeat has its center to the left of this boundary. Following we describe the procedure to find all right repeats at the highest level, for S=s1...sn. By symmetry, all left repeats can be found. To find ATRs We used the same algorithm as described with an additional tolerance factor. While calculating the longest common backward and forward extensions, we take into account the amount of mismatch between the motifs. If no. of mismatches t, total length of the string being considered k and set tolerance level then $t/k < t$, and at least half length of adjacent motif should matches then the repeats are considered. It is same as hamming distance between adjacent pair is less than a given amount and overall mismatch is given as tolerance factor.

Algorithm 2. Search the Approx. repeats using divide and conquer approach

Input: Any sequence w.

Output: All the tandem repeats in the DNA sequence

1. repeats = [] // set of repeats will be stored
2. **procedure** findreps(w) begin
3. **if** (|w| <= 1) then w is repetition-tree
4. **else** begin
5. left_sequence = w[1,n/2]
6. right_sequence = w[n/2,n]
7. findreps (left_sequence)
8. findreps (right_sequence)
9. **for** len := 2 to max_motif_length do
10. len1 = backwardLCE (left_sequence, len)
11. len2 = forwardLCE (right_sequence, len)
12. **if** len1 + len2 > len then
13. repeats += Repeat(len1+ len2+len)
14. **end**
15. **end**
16. **end**

It is proved that every repeats always cross centre of the sub-problems and so Algorithm 2 finds all

repeats over centre of the sequence. Now here two cases occur that is, from which side motif is to be taken, we consider both scenarios left and right side. For left side we take a unit motif (length is given in a range) and calculates longest common extension in both left and right direction. If sum of length of left LCE and right LCE is greater than unit motif length it is a repeat, this repeat is added to repeat set. Similarly for right side, repeat can be found. Algorithm 2 considers a sequence of length n, W = s_1 s_2…s_n, if |W| is less than 1 it has no repeats and otherwise it divides problem statement into smaller problem as left sub-problem X = s_1…$s_{n/2}$ and right sub-problem as Y = $s_{n/2}$…..s_n.

IV. Experimental Results

Our implementation EAR(exact) is based on "Suffix Array" and EAR(approx.) is based on Divide and Conquer and accepts DNA sequence input format. Fasta files contains base pair in single character format with sequence numbers and 'N' (not known bp). For nucleotide sequences it consists of {A,C,G,T} symbols only where 'A' - Adenine, 'G' - Guanine, 'T' -Thymine and 'C' - Cytosine and 'N' is "not sure" base pair. Before taking input as fasta file our program remove sequence number and 'N' symbols from file so modified file contains {A,C,G,T} symbol only.

The implementation is tested on i3-4200U 1.7 GHz 4 Core Intel processor with 4 GB RAM. The Program is tested on human chromosome (Homo sapiens, GRCh38.p7 Primary Assembly) 38 downloaded from NCBI (https://www.ncbi.nlm.nih.gov) available for experimental purpose.

We also evaluated performance of our algorithm compare to some existing applications on Arabidopsis thaliana (chromosome 4) and Escherichia coliand Zaire ebolavirus genomes (GenBank association: NC_003075.7, NC_002549.1 respectively) of length ranges from 4500 to 1859000 (18 MB) bps. Some of the implementation couldn't evaluate this much amount because of memory and runtime limited capacity.

Suffix Array based Exact Repeats and Divide and Conquer based Approx. Repeats

We tested our implementation on different Homo Sapiens Chromosome 1-22, GenBank sequence number of different chromosomes are given in Table 1. Every sequence file is preprocessed and sequence number and 'N' symbols are removed, modified sequence file is used for repeat detection. As we know human chromosome size is very huge approx. 50-250 MB, So on 4 GB System we were not able to process that amount data that's why we are processing whole sequence in chunks of custom size ~1000000 bps. So actual repeats count may differ because of chunk file processing. For larger memory system, our program will be able to process the sequence file as is and there will be no requirement to divide it into chunks. Thus, we are limited only by the capabilities of the system and not by the algorithm. Table 1 is arranged such that Chromosome 22 occupies the first row, Chromosome 21 the second and so on upto Chromosome 1 in the last row.

Table. 1. Exact and Approx. repeats using Suffix Array and Divide and Conquer method on Homo Sapiens Chromosome

GenBank Sequence	*Sequence Size*	*Exact Repeats using Suffix Array*		*Approx. Repeats using Divide and Conquer*	
		Repeats Count	*Runtime*	*Repeats Count*	*Runtime*
NC_000002 2.11	50 MB	259611	9.209s	5310296	3s
NC_000002 1.9	46 MB	247930	8.611s	5553298	3.5s
NC_000002 0.11	63 MB	421634	12.78s	8629471	5.15s
NC_000001 9.10	57 MB	444267	12.67s	8165114	5.321s
NC_000001 8.10	78 MB	472118	14.565s	10886573	7s
NC_000001 7.11	81 MB	573720	17.219s	11253462	7.1s
NC_000001 6.10	88 MB	572640	15.495s	11166134	6.624s

GenBank Sequence	*Sequence Size*	*Exact Repeats using Suffix Array*		*Approx. Repeats using Divide and Conquer*	
		Repeats Count	*Runtime*	*Repeats Count*	*Runtime*
NC_0000015.10	99 MB	514938	15.063s	11520993	6.96s
NC_0000014.9	104 MB	544511	16.327s	12411831	7.117s
NC_0000013.11	111 MB	596930	18.108s	13621741	7.556s
NC_0000012.12	129 MB	845946	25.349s	18365870	10.425s
NC_0000011.10	131 MB	806627	22.462s	18272514	10.00s
NC_0000010.11	130 MB	844268	23.872s	18227441	11.188s
NC_000009.12	134 MB	738536	25.80s	16680171	10.250s
NC_000008.11	141 MB	890415	26.903s	19829848	12.337s
NC_000007.14	155 MB	999264	29.048s	21977362	12.747s
NC_000006.12	166 MB	1034057	31.018s	23455690	13.783s
NC_000005.10	176 MB	1081118	29.643s	24985319	15.163s
NC_000004.12	184 MB	1130200	33.67s	26398723	15.151s
NC_000003.12	192 MB	1168487	31.827s	27205229	16.362s
NC_000002.11	236 MB	1473320	38.734s	32785738	18.054s
NC_000001.11	241 MB	1440611	36.866s	31493659	19s

Table 1 summarizes the number of repeats decreases from Chromosome 1 to Chromosome 22 which is due to the fact that the input file size increases progressively as we move from Chromosome 1 to Chromosome 22. The number of approximate repeats found in all chromosomes of Homo Sapiens (GRCH38.p7) along with the runtimes taken in our system for our program to execute. As it can be observed, the number of approx. repeats found for each chromosome was about one and half times that of the number of exact repeats found for the same chromosome. Also, the runtime was observed to be more for the approximate repeats. The cause of this difference in numbers is due to the fact that if we allow approximate repeats there will clearly be more matches between consecutive sequences if some degree of mismatch (tolerance factor) is allowed. The number of repeats found will also depend on the number of base pairs allowed in a small motif for which the sequence is being considered. As size increases, number of repeats also increases almost at same rate.

Table. 2. Comparison of different Sequence Repeats searching tools

Software	*CPU Time (ss)*	*Real Time (ss)*	*SSRs Reported*	*SSRs in Range*	*Correct*	*Percent Correct*
EAR(Exact)	840	840	90777	90777	90777	100
EAR(Approx.)	840	840	27224260	242260	242260	100

Software	*CPU Time (ss)*	*Real Time (ss)*	*SSRs Reported*	*SSRs in Range*	*Correct*	*Percent Correct*
GMATo	3 29	3 29	727138 58	15 284	66 17	43 .29
SA-SSR	2 882	2 416	38088	38 088	38 088	10 0
MREPs	3 93	3 93	75552	37 076	37 076	10 0
ProGeRF	3 194	3 194	545712 9	22 78	22 68	99 .56
QDD	2 4	2 4	53248	17 418	17 418	10 0
SSR-Pipeline	1 411	1 411	603440 67	36 398	36 398	10 0
SSRIT	2	2	13217	1 3217	1 3217	1 00
TRF	1 2	1 2	20357 15	1 47284	3 3876	2 3

From Table 2 we can perform a comparative analysis of our tools EAR(Exact) and EAR(Approx.) with respect to other existing tools. Maximum length repeat found for Arabidopsis thaliana is 9777 bps and unit motif length is 3259 bps then it reduces to unit motif length 498, 356, 306 bps and so on. Repeats of length 2-8 bps repeats more frequently. Our implementation considers also branching repeats.. The difference between the number of SSRs in range and reported is due exclusively to period size (greater than 7). Our tool EAR(Exact) reports 90777 repeats which compares well with most of the tools we use for comparison. Specifically Mreps and and QDD. The difference in the counts is due to various biases of the algorithms used and the difference in the approaches-statistical or heuristic and the various parameter settings. Our tool locates all the repeats found in the data set without any length restrictions. To compare with other similar tools, we set some parameters like tolerance factor. Some of the results that we obtained were as follows -

1. When Tolerance is 10 % i.e, for 10 length motif 1 mismatch can be tolerated: no. of repeats reported in Arabidopsis thaliana is 3130652.

2. When Tolerance is 30 % i.e, for 10 length motif 3 mismatch can be tolerated: no. of repeats reported in Arabidopsis thaliana is 21510292.

3. When Tolerance is 50 % i.e, for 10 length motif 5 mismatch can be tolerated: no. of repeats reported in Arabidopsis thaliana is 29268906.

4. When Tolerance is 70 % i.e, for 10 length motif 7 mismatch can be tolerated: no. of repeats reported in Arabidopsis thaliana is 25387683

V. Conclusion

Our program can find repeats of motif length in a given range with given maximum period limitation. We have modified the algorithm proposed by Main and Lorentz to find approximate tandem repeats by introducing a tolerance factor. Further this algorithm can be improved by applying Euclidean distance or edit distance because hamming distance has many drawbacks it only calculates character to character mismatch while edit distance can calculate score based on insertion, replacement or deletion by giving weight to each type of operation. The algorithm can also be useful in DNA compression (e.g. replacing '*xxxx'* as '*x4'* where 'x' is a repeating unit) and in DNA pattern recognition. For large DNA sequences greater than 100 MB can't be processed on simple system so parallel programming concept can enhance the memory and time efficiency of this program. Parallel programming concept can be applied easily to divide and conquer method.

BIBLIOGRAPHY

[1] Abouelhoda M.I. et al. (2004) Replacing suffix trees with enhanced suffix arrays. J. Discrete Algorithms, 2, 53–86.

[2] Kurtz S. (1999) Reducing the space requirement of suffix trees. Softw. Pract. Exp., 29, 1149–1171.

[3] Lim K.G. et al. (2013) Review of tandem repeat search tools: a systematic approach to evaluating algorithmic performance. Brief. Bioinf., 14, 67–81.

[4] Madesis P. et al. (2013) Microsatellites:

Evolution and contribution In: Microsatellites. Springer, pp. 1–13.[PubMed]
[5] Manber U., Myers G. (1993) Suffix arrays: a new method for on-line string searches. SIAM J. Comput., 22, 935–948.
[6] McCreight E.M. (1976) A space-economical suffix tree construction algorithm. J. ACM (JACM), 23, 262–272.
[7] Schlotterer C., Tautz D. (1992) Slippage synthesis of simple sequence DNA. Nucleic Acids Res., 20, 211–215.
[8] Ukkonen E. (1995) On-line construction of suffix trees. Algorithmica, 14, 249–260.
[9] Weiner P. (1973) Linear pattern matching algorithms. Switching and Automata Theory, 1973. SWAT'08. In: IEEE Conference Record of 14th Annual Symposium on IEEE, pp. 1–11.
[10] Benson,G. (1999) Tandem repeats finder: a program to analyze DNA sequences. Nucleic Acids Res., 27, 573–580.
[11] R. Pokrzywa, Application of the Burrows-Wheeler Transform for searching for tandem repeats in DNA sequences, Int. J. Bioinform. Res. Appl. vol. 5 (4) (2009) 432–446.
[12] Merkel A, Gemmell N. Detecting microsatellites in genome data:Briefings in bioinformatics CS . VOL 9. NO 5. 355,366
[13] Adalberto T. Castelo, Wellington Martins and Guang R. Gao. Tandem Repeat Occurrence Locator: Bioinformatics (2002) 18 (4): 634-636. doi: 10.1093/bioinformatics/18.4.634
[14] Pamela Vinitha Eric, Kusum Rajput, Gopakumar G ,An Improved Method to Identify Exact and Approximate Tandem Repeats in DNA Sequences using Biclustering, International Journal of Computer Applications (0975 - 8887) Volume 146 - No.9, July 2016
[15] Wexler Y, Yakhini Z, Kashi Y, et al. Finding approximate tandem repeats in genomic sequences. J Comput Biol 2005;12:928–42.
[16] B. D. Pickett, S. M. Karlinsey, C. E. Penrod, M. J. Cormier,M. T. W. Ebbert, D. K.Shiozawa, C. J. Whipple and P. G. Ridge,SA-SSR: a suffix array-based algorithm for exhaustive and efficient SSR discovery in large genetic sequences ,Bioinformatics, 32(17), 2016, 2707–2709
[17] Kasai, T.; Lee, G.; Arimura, H.; Arikawa, S.; Park, K. (2001). Linear-Time Longest-Common-Prefix Computation in Suffix Arrays and Its Applications. 181–192. doi:10.1007/3-540-48194-X_17. ISBN 978-3- 540-42271- 6.
[18] Barsky, Marina; Stege, Ulrike; Thomo, Alex; Upton, Chris (2009), Suffix trees for very large genomic sequences;, CIKM Proceedings of the 18th ACM Conference on Information and Knowledge Management, New York, NY, USA: ACM.
[19] Abouelhoda, Mohamed Ibrahim; Kurtz, Stefan; Ohlebusch, Enno (2002). The Enhanced Suffix Array and Its Applications to Genome Analysis. Algorithms in Bioinformatics. Lecture Notes in Computer Science. 2452. p. 449. doi:10.1007/3-540- 45784-4_35. ISBN 978-3- 540-44211-0.
[20] Juha Kärkkäinen, Peter Sanders, and Stefan Burkhardt. 2006. Linear work suffix array construction. J. ACM 53, 6 (November 2006), 918-936. doi:10.1145/1217856.1217858
[21] Tao Tao (2011-08- 24). Single Letter Codes for Nucleotides". [NCBI Learning Center]. National Center for Biotechnology Information. Retrieved 2012-03- 15.
[22] Delgrange, O. and Rivals, E. STAR: An Algorithm to Search for Tandem Approximate Repeats. Bioinformatics, 20:2812–2820, 2004
[23] Kolpakov, R. and Kucherov, G. Finding Approximate Repetitions under Hamming Distance. Theoret. Comput. Sci., 303, 2003.
[24] Landau, G., Schmidt, J. and Sokol, D. An Algorithm for Approximate Tandem Repeats. J. Comput. Biol., 8(1):1–18, 2001.
[25] Milosavljevic, A. and Jurka, J. Discovering simple DNA sequences by the algorithmic significance method. Comput. Appl. Biosci., 9(4):407–411, 1993.
[26] Wexler Y, Yakhini Z, Kashi Y, Geiger D (2004) Finding approximate tandem repeats in genomic sequences. Proc. 8th Annual Int Conf Res Comput Mol Biol (RECOMB04) pp 223–232
[27] Roman Kolpakov, Ghizlane Bana, Gregory Kucherov: Mreps: efficient and flexible detection of tandem repeats in DNA, Nucleic Acid Res. 2003 Jul 1; 31(13): 3672–3678.
[28] Main M.G, Lorentz R.J :An O(nlogn) algorithm for finding all repetitions in a string, Journal of Algorithms pg. 422-32

