**PRESENTATIONS**

**Welcome:**

 **Amnon Albeck;** Rector, Bar Ilan University.

 **Zvi Galil;** the one who coined "Stringology"

 **Yoav Kish;** Israel's minister of education.

**Dan Gusfield:**

 **Title:** Some thoughts on Unique Decipherability Problems (and disease)

 **Abstract:**

The Unique Decipherability Problem (UD) in stringology (finding a linear time algorithm for determining whether a code is UD) was independently listed as one of the major open questions by Zvi Galil more than 30 years ago; by Alberto Apostolico about 20 years ago; and by Franco Preparata. It was part of Shimon Even's Phd thesis, and one of the books by Maxime also discusses the UD problem. Lothaire (or was it Satoshi Nakamoto?) lso discuss it. Independently I homed in on it as a major open problem before knowing that others had. I come back to the problem every several years -and have made NO progress on it (but have some ideas I like). So, why should I give a talk about it? To infect more people who might then go on to cure the disease! There are also several interesting stories (regarding false proofs; and early conjectures by Schutzenberger, and by Karp, resolved by a young Peter Schor - who knew he ever did stringology?); and lesser open UD questions besides the main one. Maybe at least we can make some progress on those.

PS: As far as I can tell, the UD problem has no practical importance - it's pure stringology.

**Steve Skiena:** joint work with Tanzir Pial.

 **Title:** Sequence Alignment for Narrative Texts

 **Abstract:**

Algorithmic sequence alignment identifies similar segments shared between pairs of documents, and is fundamental to many NLP tasks. But it is difficult to recognize similarities between distant versions of narratives such as translations and retellings, particularly for summaries and abridgements which are much shorter than the original novels.

We develop a general approach to narrative alignment coupling the Smith-Waterman algorithm from bioinformatics with modern text similarity metrics. We show that the background of alignment scores fits a Gumbel distribution, enabling us to define rigorous p-values on the significance of any alignment. We apply and evaluate our general narrative alignment tool (GNAT) on four distinct problem domains differing greatly in both the relative and absolute length of documents, namely summary-to-book alignment, translated book alignment, short story alignment, and plagiarism detection---demonstrating the power and performance of our methods.

Finally, we apply our alignment methods to study how novels get adapted into feature films. We use novel-script alignments to perform an automated analysis of 40 adaptations, revealing insights into the screenwriting process concerning (i) faithfulness of adaptation, (ii) importance of dialog, (iii) preservation of narrative order, and (iv) gender representation issues reflective of the Bechdel test.

**Andrei Broder**:

 **Title:** The DNA of MinHash & the MinHash of DNA: A Personal Journey

 **Abstract:**

**Zvi Galil:**

**Title:** Georgia Tech's Revolutionary Online Program and the future of online learning

 in higher education

**Abstract:**

In May 2013, Georgia Tech together with its partners, Udacity and AT&T, announced a new online master's degree in computer science delivered through the platform popularized by massively open online courses (MOOCs). This new online MS in CS or OMSCS for short costs less than $7,000 total, compared to a price tag of $40,000 for an MS CS at comparable public universities and upwards of $70,000 at private universities.

The first-of-its-kind program was launched in January 2014 and has sparked a worldwide conversation about higher education in the 21st century. President Barack Obama has praised OMSCS by name twice, and over 1,200 news stories mentioned the program. It's been described as a potential "game changer" and "ground zero of the revolution in higher education". Harvard University researchers concluded that OMSCS is "the first rigorous evidence showing an online degree program can increase educational attainment" and predicted that OMSCS will single-handedly raise the number of annual MS CS graduates in the United States by at least 7 percent.

OMSCS started in 2014 with small enrollment of 380; in fall 2024 semester enrollment is over 15,000; OMSCS is apparently the biggest MS in CS program in the world. So far over 12,000 students have graduated from OMSCS, over 2,000 graduated in 2022 and in 2023. The number of applications to OMSCS keeps rising. In the 2023-24 academic year there were over 10,000 applications, 34% higher than the record in the year before. The program has also paved the way for more than 70 similar, MOOC-based affordable online MS programs. In November 2023, a Forbes article described OMSCS as the best degree program ever ([The Greatest Degree Program Ever (forbes.com)](https://www.forbes.com/sites/brandonbusteed/2023/11/10/the-greatest-degree-program-ever/?sh=46a10467186c). There is a shortage of one million computing professionals in the US. Therefore, OMSCS is satisfying a great national need. Starting in 2017, Georgia Tech expanded its online offerings to its undergraduate computer science students.

The talk will describe the OMSCS program, how it came about, its first ten years, and what Georgia Tech has learned from the OMSCS experience. It will also discuss the speaker's vision of the future of higher education with much larger role for online learning.

**Tomasz Kociumaka:**

**Title:** Dynamic Algorithms for Edit Distance

**Abstract:**

The dynamic version of the edit distance problem asks to maintain the edit distance of two strings that change dynamically, with each update modeled as a character edit. A folklore adaptation of the Landau‒Vishkin [JCSS'88] algorithm supports updates in Õ(k²) time, where k is the maintained edit distance. A few years ago, with Panagiotis Charalampopoulos and Shay Mozes [CPM'20], we showed an algorithm with update time Õ(n), which is optimal under OVH in terms of n. The complexity parameterized by the distance k remained open – Õ(k²) update time was still the state of the art for k ≤ √n. In this talk, I will present my very recent result with Egor Gorbachev [arXiv'24]: a dynamic edit distance algorithm that supports updates in Õ(k) time, which is optimal under OVH in terms of n and k.

Our solution relies on tools originating from weighted edit distance, where the weight of each edit depends on the edit type and the characters involved. The textbook algorithm supports weights, but the Landau-Vishkin approach does not, and a simple O(n·k)-time procedure long remained the fastest for bounded weighted edit distance. Only recently, an O(n+k⁵)-time algorithm [STOC'23] and a conditionally optimal Õ(n + √{nk³})-time solution [FOCS'23] have been discovered. Our new results also imply that, for integer edit weights between 0 and W, weighted edit distance can be computed in Õ(n+W·k²) time and maintained dynamically in Õ(W²·k) time per update.

**Peter Van Emde Boas:**

 **Title:** The history of the van Emde Boas trees

**Abstract:**

The stratified tree (better known as van Emde Boas trees) is a data I invented in the  fall of 1974 during a short postdoc residence at Cornell.

The structure implements a number of operations on  subsets of the finite universe *{1,...,N}*  in time  *O(log log(N))*. This time bound *seems* to violate the logarithmic lower bound on sorting operations but actually there is no violation of any kind.

will give some information on the historical background of this work. More relevantly I will explain why the version of the structure as originally designed is entirely different from the by now traditional recursive description one finds in several standard texts on algorithms. I conclude with some of the more recent developments about this topic since 1974.

**Paolo Ferragina:**

 **Title:** From compression to learning, one more step in data structure design.

**Abstract:**

Key-value stores and search engines are posing a continuously growing need to efficiently store, retrieve and analyze massive sets of keys under the many and different requirements posed by users, devices, and applications. Such a new level of complexity could not be properly handled by known data structures, so that academic and industrial researchers started recently to devise new approaches that integrate classic data structures with various kinds of advanced techniques drawn from data compression, computational geometry and machine learning, hence originating what are currently called “Learned Data Structures”. In this talk, I’ll survey the evolution of these kinds of data structures, discuss their theoretical and experimental performance, and point out new challenges worth of future research.

**Uzi Vishkin:**

 **Title:** Reversing the decline of general-purpose CPUs: where can parallel algorithms

 help?

 **Abstract:**

The transition from single core general-purpose commodity CPUs to parallel multicore ones required figuring out a coherent design for both new parallel system architecture and cost-effective parallel algorithms and programming. Given a benchmark suite of application code and respective workloads, hardware designers are at their best when tasked with optimizing hardware design. Given a computation cost model, algorithm designers specialize in designing algorithms for optimizing such costs. However, since the transition to multiple core CPUs around 2005, hardware designers followed a build-first-figure-out-how-to-program-later approach. Algorithms could not do much to remedy suboptimal hardware. It is now clear that this attempt did not yield the desired results. As one of several faults: too few program today’s multicore CPU for parallelism. In the same vein, hardware design leaders concluded in 2019 that it is “unlikely that simple multicore scaling will provide cost-effective path to growing performance”.

I long believed that: (i) for general parallel computing to become ubiquitous, algorithm and hardware design communities need to pull together their respective strengths, and (ii) parallel algorithms and programming need to provide an extensive benchmark suite of codes that meet two main requirements:

- The benchmark suite should reflect scalable algorithm design framework that will

 attract massive participation of programmers, once the framework is supported by

 commodity manycore CPUs.

- The framework should be supportable by cost-effective hardware, including

 constant factors.

The second requirement clearly exceeds the comfort zone of most algorithm designers. Still, the most important objective for research in general and parallel algorithms in particular is maximizing impact. Therefore, I decided to bite the bullet and devote more than two decades to accomplishing the following. Starting with the PRAM parallel algorithmic theory, a vertically integrated framework known as XMT was developed at UMD, centered around integrating a highly parallel

accelerator with a state-of-the-art CPU core. Extensive hardware and software prototyping demonstrated the feasibility and performance of this PRAM-based approach.

However, the job is not yet done. Originating from computer graphics app, GPUs have provided the most successful form factor of parallel computing so far, “eating the CPUs lunch”. The availability of GPUs led applications such as machine learning to be developed for them, much in the same often-serendipitous way that CPU applications emerged in prior decades. Relating such application-minded thinking to the title of this presentation will conclude my talk.

**Travis Gagie:** joint work with Lore Depuydt, Ben Langmead, Giovanni Manzini and Nicola

 Prezza

 **Title:** Suffixient Sets

 **Abstract:**

We define a suffixient set for a text *T [1..n]* to be a set *S* of positions between 1 and *n* such that, for any edge descending from a node *u* to a node *v* in the suffix tree of *T*, there is an element *s Є S* such that *u*'s path label is a suffix of *T [1..s - 1]* and *T [s]* is the first character of *(u, v)*'s edge label.  We first show there is a suffixient set of cardinality at most *2ṝ*, where *ṝ* is the number of runs in the Burrows-Wheeler Transform of the reverse of *T*.  We then show that, given a straight-line program for *T* with *g* rules, we can build an *O(* *ṝ + g)*-space index with which, given a pattern *P [1..m]*, we can find the maximal exact matches (MEMs) of *P* with respect to *T* in *O (m* log *(σ /* log *n + d* log  *n)$* time, where *σ* is the size of the alphabet and *d* is the number of times we would fully or partially descend edges in the suffix tree of *T* while finding those MEMs.

**Rivi Shalom:** joint work with Itai Boneh, Shay Golan, Avivit Levy and Ely Porat.

**Title:** Burst Edit Distance

**Abstract:** We define two types of burst edit errors, generalizing the traditional Levenshtein distance, occurring in Text Editing scenarios when the communication speed is unstable within a wireless keyboard usage:

1. A Burst of Errors (BE) involves a sequence of erroneous identical symbols and allows a single edit operation applied to a sequence of identical symbols;
2. A Burst of Operations (BO) involves a sequence of erroneous symbols that are not necessarily identical and allows a single edit operation applied to a sequence of symbols.

In both burst types, every burst operation has a penalty, which is a cost function F(k), where k is the burst length.

The Burst Edit Distance of two strings S and T is:

1. The minimum cost of a sequence of BE operations that transforms S into T in the bursts of errors variant (EDBE);
2. The minimum cost of a sequence of BO operations that transforms S into T in the bursts of operations variant (EDBO);

We describe solutions to both problems for general natural penalty functions families. A conditional lower bound for the EDBE problem is also given.

**Bill Smyth:** Joint with: Jacqueline W. Daykin and Neerja Mhaskar

 **Title:** V-Order: Non-Trivial But Useful & Efficient

 **Abstract:**

V-order is a total order on strings that, over 20 years of research, has rather surprisingly been shown to be competitive with the standard lexicographical order

(lexorder) as a basis for string processing of all kinds: pattern-matching, computation of the suffix array, the Burrows-Wheeler transform, and the Unique Maximal

Factorization Family (UMFF) -- a generalization of Lyndon Words. In this talk we survey these results and provide suggestions for future research: in certain processing contexts, V-order may become the ordering of choice.

**Stefano Lonardi:**

 **Title:** RAmbler Resolves Complex Repeats in Human Chromosomes 8, 19 and X

 **Abstract:**

Despite significant algorithmic and technological advancements in genome sequencng and assembly over the past three decades, modern de novo assemblers still struggle to accurately reconstruct highly repetitive regions. Here, we introduce RAmbler (Repeat Assembler), a new assembler that can resolve complex repetitive
regions exclusively on PacBio HiFi reads. RAmbler (i) identifies repetitive regions by detecting unusually high coverage regions after mapping HiFi reads to the draft genome assembly, (ii) computes the k-mers that are expected to occur only once in the genome, (iii) uses the relative location of single copy k-mers to barcode each HiFi read, (iv) clusters HiFi reads based on their shared barcodes, (v) generates contigs
by assembling the reads in each cluster, and (vi) generates a consensus assembly from the overlap graph of the assembled contigs.
We show that RAmbler can reconstruct human centromeres and other complex repeats to a quality comparable to the manually-curated telomere-to-telomere human genome assembly.

**Frantisek Franek**

 **Title:** *d*-step aspect of periodicities in strings.

 **Abstract:**

Periodicities in strings, in particular tandem repetitions, has been of interest to researchers in the string algorithms and data structures and combinatorics on words areas from the beginning . The pioneering work of Corchemore in 1981 showed that the optimal bound for the number of maximal repetitions in a string of length $n$ is of $O(n\log(n))$ complexity and attained by Fibonacci strings, followed closely by the seminal work by Apostolico and Preparata. In 1989, Main introduced an $O(n\log(n))$ algorithm for detection of maximal repetitions where the $\log(n)$ factor represented the size of the alphabet, so for a constant size alphabet, it was a linear algorithm and a linear number of repetitions.

From this, two subsequent lines of research crystallized over several years. The first, dealing with the generalization of maximal repetitions in the form of runs and determining the maximum number of runs in a string, the second line of research focusing on determining the maximum number of distinct squares in a string. These two lines of research culminated in corresponding conjectures: the ***runs conjecture***: *the maximum number of runs in string is bounded by the length of the string*, and the ***distinct squares conjecture***: *the maximum number of distinct squares in string is bounded by the length of the string*. Intense research on these problems were initiated by the pioneering work of Kolpakov & Kutcherov (1999) for runs and by

Fraenkel & Simpson (1998) for distinct squares. Before both conjectures were settled, Deza, myself, and our graduate students Jiang and Baker strengthened both conjecture to hypothesize the bound to be the length of the string less the size of the alphabet of the string. The bounds $n-d$ were based on the $d$-step approach where the $d$ stands for the size of the alphabet and $n$ the length of the string.

The runs conjecture was settled by Banai et al. in 2015 (published in 2017). The $d$-step conjecture for runs was proven by Deza et al. in 2017. The $d$-step conjecture for distinct squares (and hence the distinct squares conjecture ) was recently proven by Brlek & Li (not published yet, though the paper in arXiv has a lot of defects). The Brlek & Li approach shows why the size of the alphabet naturally occurs in the bound. The $d$-step aspects of both problems investigated by Deza et al. have some additional consequence beyond the size of the bound, and in both cases they are shown to be optimal as strings of length $n$ with $n-d$ runs or distinct squares are computed.

**Kuba Radoszewski**

**Title:** Fast and Compact Representation of Key Positions Across All Windows in a

Text

**Abstract:**

We consider an augmented text consisting of n words, where each word is assigned an integer value (e.g. a hash value or a color encoding some useful information). A window in the text is a range of consecutive words that represent a text chunk. For a parameter k < n, we want to find the k key positions for each of the θ(n^2) windows in the text. The key positions in a window correspond, for example, to the k smallest hash values. This basic problem has been introduced for the bottom-k sketches for Jaccard index to detect near-duplicate passages in a document collection by Deng et al. (SIGMOD '21, '22). Their solution takes O(nk^2) space and O(nk^2 + n log n + nkf) time, where f is the maximum frequency of a word in the source document. We propose a compact way to represent the same information as above using O(nk) space, taking O(nk log n) time and O(nk + n log n) space for the computation, independently of f. Our experiments on real-world datasets show that our solution makes a major step towards scalability for large document repositories.

**Michael Itshaki:**

 **Title:** The Universal Compression of Palindromes.

 **Abstract:** Data compression and palindromic analysis in strings are two orthogonal

fields of study in the pattern matching endeavor. Widely adopted and general-purpose data structures and algorithms, such as the suffix tree and suffix array influenced both fields of research. For a given string S, we want to compress the palindromic information of S. Previous works discuss the compressibility of palindromes and palindromes generator and provide some asymptotic bounds on the number of bits needed to represent only the palindromes of a string.

Under the restriction of only preserving the pattern while disregarding the content of the string itself, most text preprocessing transformations and techniques either lose information or don’t simply work. However, we take well known algorithms, such as the BWT, MFT and LZ\* to its variants, and adapt them to the domain of palindromes compression. Using the variants and other studied results in stringology, we construct a universal lossless palindromes compressor for general strings .

**Avivit Levy:** joint with Dana Shapira

**Title:** Computation over APT Compressed Data

**Abstract:**

The Arithmetic Progressions Tree (APT) is an encoding of a monotonic sequence L in [1..n]. Previous work on APT coding focused on its theoretical and experimental compression guarantees. This work is the first to consider computations over APT compressed data. In particular:

1. We show how to perform a search for any sub-sequence/a set from the monotone sequence L in time proportional to the query sub-sequence length/set size multiplied by the size of the APT compressed representation of L.
 2. We show how, given the APT compressed representation of the monotone sequence L, we can find a minimum run-length of L in constant time, a maximum run-length of L in O(\log n) time, and all runs of L in constant time plus the output size.
3. We show how, given the APT compressed representation of the monotone sequence L, we can answer whether a consecutive periodic pattern P is represented by an APT-node in O(\log n) time and report occurrences of P in L within the output size time.
 4. In addition, we improve the APT construction algorithm time and space complexity.

**Gary Benson:**

**Title:** A Deep Learning Approach for Filtering Sequencing Reads for Tandem

 Repeats

**Abstract:**

The Tandem Repeats Finder (TRF) program was introduced 25 years ago (1999), before the publication of the first draft of the human genome (2001). At that time, a sequence set of 3 billion bases was an extremely large dataset and the several hours required for TRF analysis of such enormous data was considered acceptable. Over the years, processors have gained speed, but that gain has been swamped by the growth in whole genome sequencing (WGS) datasets. Today, a single human sample of 100X coverage is not uncommon, and millions of samples are being made available as a result of large sequencing projects. At that scale, TRF is unable to keep up with the computational demands.

In recent years, my group developed a computational pipeline, VNTRseek, that uses TRF to detect copy-number variant tandem repeats (variable number tandem repeats) in WGS datasets and employed it to analyze over 2800 human samples. We found over 35,000 VNTRs, 1,096 population specific alleles (which could confound GWAS analysis), and 187 genes with expression correlated with proximal VNTR genotypes (eQTLs). These results signal the importance of further human VNTR analysis.

However, as stated above, computing time is a limiting factor. For example, one typical dataset from the 1000 Genomes Project contains over 700 million reads, yet only 4 million (<6%) are informative for VNTR genotyping. This suggests that development of a rapid filtering method, to identify reads containing tandem repeats, would benefit the VNTRseek pipeline and facilitate further high-throughput genome analysis.

The release of ChatGPT (Chat Generative Pre-trained Transformer) in November 2022 riveted public attention and illuminated the predictive power of AI deep learning models, the vast computational speed of graphical processing units (GPUs), and the utility of AI software tools like Tensorflow and PyTorch Standard genome analysis will likely be vanquished by these new approaches, in particular, because the huge volumes of genomic data that are required to train AI models are readily available.

In this talk, I will discuss the challenges involved in developing a deep learning, Convolutional Neural Network (CNN) classification model which, when given a read sequence, decides (yes or no), rapidly and with very high accuracy, whether it is likely to contain a tandem repeat. The work was comprised of three equally important parts: data curation, model construction and testing, and prediction (data pipelining into and out of a trained model). Preliminary results show that a simple model, if trained with reliable data, can obtain an accuracy of ~98% on data not used in training.

**Moshe Vardi:**

 **Title:** Linear-Time Model Checking -- Automata Theory in Practice

 **Abstract:**

 In automata-theoretic model checking we compose the design under

 verification with a Büchi automaton that accepts traces violating the

specification. We then use graph algorithms to search for a counterexample

trace. The basic theory of this approach was worked out in the 1980s, and

the basic algorithms were developed during the 1990s. Both explicit and

symbolic implementations, such as SPIN and and SMV, are widely used. I

will covers the fundamentals of automata-theoretic model checking and

review the reduction of the theory to practice.

**Jeff Vitter:** Joint with Hongwei Huo of Xidian University in Xian, China

**Title:** Text Indexing in Optimal High-Order Entropy Space, With Application

**Abstract:**

A pivotal problem in the stringology community is to design and implement entropy-compressed self-indexes for massive text data strings that support fast search. For a text string $\idcal{T}$ of $t$ symbols over an alphabet of size $\sigma$, we propose a novel text self-index that for the first time for arbitrary alphabets achieves the space bound $t\idcal{H}\_k + o(t)$ bits, where $\idcal{H}\_k = \idcal{H}\_k(\idcal{T})$ is the $k$th-order empirical entropy of text~$\idcal{T}$ and $k \leq \alpha \log\_\sigma t - 1$ for some constant $0 < \alpha < 1$. The bound also holds for the more stringent finite-set entropy. Previous approaches have an extraneous space term of at least either $o(t \log \sigma)$ or $o(t \idcal{H}\_k) $ bits or larger. \emph{A counterintuitive basis for our improvements is that we avoid using wavelet trees}, which for over 20 years has been ubiquitous in text indexing.

We also discuss applications to labeled property multidigraphs that efficiently support typical interactive complex and business intelligence queries, as well as BFS and PageRank. Experimental results on arge LDBC SNB data demonstrate substantial improvements in practice.

**Dana Shapira:** Joint with: Tomi Klein

 **Title:** Extending Forward Adaptive Coding

 **Abstract:**

Forward coding is a recent method of modeling the data source based on predictions of what is yet to come. It is the best dynamic variant known to date, and we extend it by proposing new adaptive coding methods, which we call positional and weighted coding. We then show how to apply it to word-based alphabets, increasing compression efficiency also in practice. Finally, we combine the above with a PPM scheme and demonstrate the effectiveness of the proposed approach by experimental results.

**Igor Zavadskyi**

 **Title:** Fast practical adaptive encoding on large alphabets

 **Abstract:**

We present two new, simple algorithms for adaptive coding that perform well even on large alphabets like those used in word-level natural language text compression. The first is based on re-indexing a universal codeword set of an arbitrary structure. The more evolved second algorithm assumes that the underlying codeword set can be optimized and re-generated efficiently. Experiments show that this algorithm, with the use of the Binary Mixed-digit codes, produces about 1% bigger compressed files than the classical Vitter's algorithm adapted to large alphabets, being orders of magnitude faster in encoding and decoding.