Annual IEEE Symposium on Foundations of Computer Science 2024

Matej Belšak[2,*], Nik Čadež[2], Klemen Kavčič[2], Tomaž Leonardis[1], Bor Pangeršič[1], Marko Rozman[2], Pia Sotlar[2], Vanja Stojanović[1]

[1]Faculty of Mathematics and Physics, University of Ljubljana
[2]Faculty of Computer and Information Science, University of Ljubljana

Abstract

We will write this at the end

Introduction

The FOCS conference or IEEE Annual Symposium on Foundations of Computer Science is an academic conference that coveres a broad range of theoretical computer science. It is sponsered by the IEEE Computer Science Technical Committe on the Mathematical Foundations of Computing (TCMF) **ieeefocs**.

The FOCS 2024 took place in Chicago - Voco Chicago Downtown, from October 27-30, 2024 **focs2024**. It covered a variety of topics, for submissions the following were mentioned **focs2024**:

- Communication complexity
- Circuit complexity
- Average-case algorithms and complexity
- High-dimensional algorithms
- Online algorithms
- Parametrized algorithms
- Spectral methods
- Streaming algorithms
- Randomized algorithms
- Cryptography
- Computational complexity
- Algorithms and data structures
- Quantum computing
- Foundations of machine learning
- Algorithmic coding theory
- Sublinear algorithms
- Algorithmic graph theory
- Continuous optimization
- Foundations of fairness, privacy and databases
- Pseudorandomness and derandomization
- Markov chains
- Analysis of Boolean functions
- Economics and computation
- Combinatorial optimization
- Algebraic computation
- Approximation algorithms
- Parallel and distributed algorithms
- Computational learning theory
- Computational geometry
- Algorithmic game theory
- Combinatorics

The official welcome message of FOCS 2024 states that nearly 500 papers were submitted, but does not specify the exact number, out of which 133 were accepted and 131 were presented as talks during the event.

In the following three sections we present three curated papers from the conference... (dopisemo na koncu ko vemo katere)

O(1) Insertion for Random Walk d-ary Cuckoo Hashing up to the Load Threshold

Random walk d-ary cuckoo hashing

Cuckoo hashing algorithm's basic idea is to resolve collisions by using two hash functions instead of one. When inserting a new object x into the table, if the slot $h_1(x)$ is occupied, the existing object x' is replaced by x, and then x' is inserted into slot $h_2(x')$. If the number of iterations exceeds a threshold, the whole table is rehashed with new hash functions. Random Walk d-ary Cuckoo Hashing generalizes the idea by using d hash functions and using a random walk to choose the next hash function in case of a collision. Standard cuckoo hashing, equivalent to the $d = 2$ case, has a load threshold of 0.5, meaning it can use up to 50% of the hash table space. The d-ary hashing improves this threshold. For example, the $d = 3$ case has the threshold at approximately 0.9, while the insertion time increases linearly with d. The insertion algorithm guarantees $\mathcal{O}(1)$ lookup and deletion time, as the object can be retrieved by checking its d positions.

*Corresponding author: mb4390@student.uni-lj.si
Ljubljana, March 2025

Insertion time

This paper shows that for any $d \geq 4$ hashes and load factor $c < c * d$, the expectation of the random walk insertion time is constant. It shows that the expected number of reassignments during insertion does not depend on the size of the hash table m, but only on d and c. The article uses bipartite graphs as a representation for the hash functions and objects. In the graph, the objects in set X are connected to their d possible locations in the hash table Y. In this representation, a valid perfect matching corresponds to a valid assignment of objects to slots. The existence of such a matching is subject to Hall's Theorem, which states that a perfect matching exists if and only if every subset $W \subseteq X$ is smaller than its neighborhood in Y. Neighborhood meaning all nodes connected to at least one vertex in W. The paper shows that when the load factor is below the threshold $c * d$, such perfect matchings exist with high probability as the graph exhibits strong expansion properties. The article identifies "bad" sets which have few connections to the rest of the graph. In these sets, the walk might get stuck, but the authors prove that the random walk is unlikely to hit such a set in the first $\mathcal{O}(i^{999})$ steps, and that any random walk which avoids it for the first $\mathcal{O}(i^{999})$ steps is likely to finish in $\mathcal{O}(i)$ steps.

Super-polynomial tail bounds

The paper also provides super-polynomial tail bounds on the insertion time, showing that the probability of a walk exceeding ℓ steps decays exponentially in ℓ, with the exponent approaching 1 as d increases. They relate their findings to previous work on BFS-based insertion algorithms, noting that random walk insertion achieves comparable or better performance without the computational overhead of BFS path searches.

Computing the 3-Edge-Connected Components of Directed Graphs in Linear Time

Abstract

The paper describes a significant improvement of a *randomized* (Monte-Carlo) algorithm for computing the 3-edge-connected components of a digraph with m edges in polylogarithmic time $\widetilde{O}(m^{3/2})$. The algorithm described bests the previous one by being deterministic and computable in linear time.

Preliminaries and primary problem

This algorithm solves the problem of finding **3-edge-connected components in directed graphs**, for preliminaries; Let $G = (V, E)$ be a strongly connected directed graph with $|V(G)| = n$ and $|E(G)| = m$. Generally a set of edges $C \subseteq E$ is a **cut** if $G \setminus C$ is not strongly connected i.e. there does not exist a directed path between every pair of vertices, if $|C| = k$ we refer to C as a **k-sized cut** of G. Hence a digraph G is k-edge-connected if it has no $(k-1)$ cuts.

We say that two vertices v and w are k-edge-connected, and we denote this relation by $v \leftrightarrow_k w$, if there are k-edge-disjoint directed paths from v to w and k-edge-disjoint directed paths from w to v. We define a **k-edge-connected component** of a digraph G as a maximal subset $U \subseteq V(G)$ such that $u \leftrightarrow_k v, \forall u, v \in U$.

How they achieved this improvement

The new method relies on a substructure of digraphs, known as 2**-connectivity-light graph** (denoted 2CLG). This is because the decomposition of digraphs into a collection of 2CLGs exists in linear time, and maintains the 3-edge-connected components of the original graph. They also define the minimal 2-in and -out sets, which contain vertices with out- or in-degree of 2. Formally, we define both here.

Definition 1. *A* 2***-connectivity-light*** *graph G is a strongly connected digraph that contains two types of vertices;* ***ordinary*** *and* ***auxiliary****, that satisfy the following conditions:*

1. *Any two ordinary vertices are* 2*-edge-connected,*
2. *each auxiliary vertex has an in- or out-degree of 1,*
3. *for every vertex u with out-degree > 1 and every vertex v with in-degree > 1, there are 2 edge-disjoint paths from u to v,*
4. *for each auxiliary vertex v with out-degree (resp. in-degree) of one, all paths from v to any vertex in G (resp. from any vertex in G to v), we have exactly one common edge, the unique out-edge (resp. in-edge).*

Definition 2. *For a strongly connected digraph G we arbitrarily choose a start vertex s. For any vertex $v \neq s$ we define $M(v)$ as a* ***minimal*** *2**-in set** that contains v, i.e. a minimal set of vertices which contains v, does not contains s, and has two incoming edges from $V(G) \setminus M(v)$, we denote by $M_R(v)$ the analogous sets in G^R, which is the reverse graph of G.*

The technique is then based on the following proposition and theorem.

Proposition I.3. *Let G be a 2GLC with a fixed ordinary start vertex s. Then for any two ordinary vertices u and*

v, we have $v \leftrightarrow_k u$ if and only if $M(u) = M(v)$ and $M_R(u) = M_R(v)$.

Theorem II.5. *Let G be a strongly connected digraph. In linear time, we can construct a collection $H_1, ..., H_t$ of 2CLG graphs, such that:*

- *For every vertex of G there is exactly one graph among $H_1, ..., H_t$, that contains it as an ordinary vertex.*
- *Every two vertices u and v of G are 3-edge-connected if and only if there is an $i \in \{1, ..., t\}$ such that u and v are 3-edge-connected.*

Paper 3

Verifying Groups in Linear Time

Ta članek obravnava temeljni algoritemski problem odločanja, ali podana $n \times n$ tabela opisuje grupo. Deterministični pristop k tej težavi je že dolgo temeljil na Lightovi ugotovitvi (1949), ki zahteva $\mathcal{O}(n^2 \log n)$ časa, medtem ko sta Rajagopalan in Schulman (FOCS 1996) s svojo naključno metodo znižala zapletenost v $\mathcal{O}(n^2 \log(1/\delta))$, ob majhni verjetnosti za napako δ. Glavni doprinos tega dela je nov deterministični algoritem, ki deluje v $\mathcal{O}(n^2)$ času, kar je optimalno glede velikost vhodne tabele $n \times n = n^2$. Ključni izziv je preverjanje asociativnosti: ker je treba za asociativnost preveriti $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ za vse trojice a, b, c, bi naivni postopek obsegal $\Theta(n^3)$ preverjanj. Delna znana izboljšava uporabi množico generatorjev velikosti $\log(n)$ (Lightov pristop) in s tem zmanjša število preverjanj, a še vedno pri $\mathcal{O}(n^2 \log n)$. Novi algoritem to oviro preseže z vpeljavo koncepta "4-asociativnosti" na skrbno izbranem majhnem podmnožičnem sistemu $S \subseteq G$. Množica S se imenuje "osnova" (basis), če lahko vsakega elementa grupe G izrazimo kot produkt dveh elementov iz S. Avtorji dokažejo, da je, če taka osnova izpolnjuje pogoje asociativnosti (posebno 4-asociativnost) za vse četverice iz S, potem asociativnost velja za celotno tabelo (tj. celo grupo). Ker lahko $|S|$ izberemo reda $\sqrt{n}$, preverjanje 4-asociativnosti zahteva $\mathcal{O}(|S|^4) = \mathcal{O}(n^2)$ časa, kar točno sovpada z najboljšim možnim časom, glede na velikost vhoda. Sama konstrukcija takšne majhne osnove temelji na globljih grupno-teoretskih idejah. Avtorji izrabijo dejstvo, da ima vsaka končna grupa, ki ni praštevilske velikosti, "veliko podgrupo" (large subgroup) z indeksom največ $\sqrt{n}$. Prisotnost takih podgrup omogoča učinkovito dekompozicijo grupe v dve podmnožici, katerih velikosti se množita v največ konstanto $|G|$. S postopnim iskanjem takih velikih podgrup se nato sestavi zahtevana osnova (base). V tem procesu se identificirajo pomembne normalne podgrupe (če jih grupa ima) s postopkom, ki dinamično sledi konjugacijskim razredom in preverja, ali ostaja unija teh razredov zaprta za množenje. Če se izkaže, da grupa ni preprosta, se algoritem posveti ustrezni normalni podgrupi ali komplementarnemu kvocientu in nadaljuje rekurzivno. Če je grupa preprosta, avtorji uporabijo klasifikacijski izrek o končnih preprostih grupah in znane konstrukcije velikih podgrup v tem okviru. Ko je osnova enkrat določena, je preverjanje, da obstaja iskani identitetni element in da ima vsak element skupine svoj inverz, enostavno izvedljivo v $\mathcal{O}(n^2)$ z neposrednim preverjanjem. Ključen korak je tako "čiščenje" dragega preverjanja asociativnosti po celotni tabeli: zadostuje preverjanje 4-asociativnosti na manjši "osnovi". Poleg tega vgradijo še pregled normalnih podgrup in iskanje velikih podgrup, vse skupaj v $\mathcal{O}(n^2)$ ali $\mathcal{O}(n^{1+\epsilon})$ pri izvedbi dodatnih dekompozicij. Celota postopkov zagotavlja, da se končno preverjanje, ali vhodna tabela dejansko predstavlja grupo, izvede v $\mathcal{O}(n^2)$. To pomeni nov deterministični mejnik in dokončno odgovarja na dolgo odprto vprašanje o natančni časovni zahtevnosti preverjanja grupne strukture iz Cayleyjeve tabele.

Paper 4

Lempel–Ziv (LZ77) Factorization in Sublinear Time

Lempel–Ziv (LZ77) factorization is a string processing technique and the main component of most data compression algorithms, such as ZIP, PDF, and PNG. It separates a given string greedily from left to right into phrases $T = f_1 f_2 \cdots f_z$, so that each phrase is either the first occurrence of a character or the longest prefix of the remaining suffix that has already appeared earlier in the text. Each phrase is encoded either as a that same character or as a pair (l, i), where l is the length of the phrase and i is the position of its earlier occurrence. For example, for the string $T = b \cdot b \cdot a \cdot ba \cdot aba \cdot bababa \cdot ababa$, the LZ77 representation is $(0, b), (1, 1), (0, a), (2, 2), (3, 3), (6, 7), (5, 10)$.

In the RAM model, the theoretical lower bound for LZ77 factorization is $O(n/\log_\sigma n)$, which matches the maximum number of phrases. Despite this, no algorithm achieved this optimal bound until recently. Earlier algorithms were based on suffix trees or suffix arrays, achieving $O(n \log \sigma)$ time and $O(n)$ space. Later improvements reduced space usage to the optimal $O(n/\log_\sigma n)$, but the algorithm still required $\Omega(n)$ time in the worst case.

Kempa and Kociumaka introduced the first sublinear-time algorithm for LZ77 factorization, running in $O(n/\sqrt{\log n})$ time for binary alphabets and

$O((n \log \sigma)/\sqrt{\log n})$ time for larger alphabets, while using optimal $O(n/\log_\sigma n)$ space. They developed a novel index structure that can be built in sublinear time and efficiently finds the leftmost previous occurrence of a substring. This index makes the computation of the Longest Previous Factor (LPF) for each position in the string fast and answers substring occurrence queries in $O(\log^\varepsilon n)$ time.

Instead of relying on the classical method based on Range Minimum Queries (RMQ) over the suffix array—which is too slow for sublinear-time construction—they use a sampling-based approach. The idea is that instead of looking through all the text positions they select a small, representative sample (S) of positions in non-periodic regions (size $O(n/\log_\sigma n)$). The rule is that if two substrings are equal their starting positions are either both included or both excluded from S.

To check if a substring has appeared before, the problem is turned into a special kind of Range Minimum Query (RMQ). Prefix RMQ finds the smallest value in a range, but only if the prefix matches a given pattern. The authors create a fast and memory-efficient data structure to perform these prefix RMQ queries.

For periodic regions, the algorithm uses two different approaches. If the pattern is only partially periodic, it uses sorted runs and special range queries (3-sided RMQ) to handle them. If the pattern is fully periodic, the algorithm marks the starting positions with a bitvector (a sequence of 0 and 1, where 1 defines the starting postions of the pattern). It then uses fast rank/select queries on this bitvector to quickly find the relevant patterns.

So by using this sampling-based approach with prefix RMQ structure, the authors achive an index that maintains optimal $O(n/\log_\sigma n)$ space complexity, sublinear preprocessing time, and efficient query performance.

Paper 5

Hi, I'm paper 5!

Paper 6

Hi, I'm paper 6!