LexicMap: efficient sequence alignment against millions of prokaryotic genomes

Wei Shen[1,2*], Zamin Iqbal[2,3,*]

[1] Department of Infectious Diseases, Key Laboratory of Molecular Biology for Infectious Diseases (Ministry of Education), Institute for Viral Hepatitis, the Second Affiliated Hospital, Chongqing Medical University, Chongqing, China

[2] European Molecular Biology Laboratory - European Bioinformatics Institute, Hinxton, UK

[3] Milner Centre for Evolution, University of Bath, UK

[*]Correspondence: shenwei356@cqmu.edu.cn (W.S.) and zi245@bath.ac.uk (Z.I.)

Abstract

Alignment against a database of genomes is a fundamental operation in bioinformatics, popularized by BLAST. However, the rate at which microbial genomes are sequenced has continued to increase, and there are now datasets in the millions, far beyond the abilities of existing alignment tools. We introduce LexicMap, a nucleotide sequence alignment tool for efficiently querying moderate length sequences (> 500 bp) such as a gene, plasmid or long read against up to millions of prokaryotic genomes. A key innovation is to construct a small set of probe *k*-mers (e.g. n = 40,000) which "window-cover" the entire database to be indexed, in the sense that every 500 bp window of every database genome contains multiple seed *k*-mers each with a shared prefix with one of the probes. Storing these seeds, indexed by the probes with which they agree, in a hierarchical index enables fast and low-memory variable-length seed matching, pseudoalignment, and then full alignment. We show that LexicMap is able to align with higher sensitivity than Blastn as the query divergence drops from 90% to 80% for queries ≥ 1 kb, and then benchmark on small (GTDB) and large (AllTheBacteria and Genbank+RefSeq) databases. We show that LexicMap achieves higher sensitivity and speed and lower memory compared to the state-of-the-art approaches. Alignment of a single gene against 2.34 million prokaryotic genomes from GenBank and RefSeq takes 36 seconds (rare gene) to 15 minutes (16S rRNA gene). LexicMap produces output in standard formats including that of BLAST and is available under MIT license at https://github.com/shenwei356/LexicMap.

Keywords: sequence alignment, prokaryotic genomes, microbial genomes, large-scale data

Introduction

Alignment of sequences to a single reference genome is a well-studied problem[1-4]. For specified gap and mismatch costs, dynamic programming is guaranteed to obtain the optimal solution[5-7], but processing time scales as the produce of the reference genome size and the query size. This is too slow in practice, so the challenge is to find faster shortcuts that ideally are still guaranteed to give the optimal solution. Rapid progress has recently been made on this front[8-12]. However, in this problem, the reference genome is relatively small, so it is possible to index all or most of the *k*-mers within it or to create a full-text index such as the FM index[13].

If seeking to align to a large database of bacterial genomes, we are confronted with an enormous scaling problem. The genome diversity of all bacteria is vast orders of magnitude beyond what one needs to handle for a single genome. Many bacterial species have “open pangenomes”, which means that every new genome of the species reveals hitherto unknown genes and sequence[14,15] - this is a completely different type of problem to indexing the pangenome of a recent and relatively inbred species with little diversity, such as *H. sapiens*[16]. Therefore, creating a *k*-mer index or an FM index does not appear to be a sustainable solution. We could, through hyper-efficient implementations and ruthless performance optimization, perhaps develop such an index for today's data, but the approach appears doomed. Data in the DNA archives is doubling every 2 years or so, and the breadth of the tree of life continually expands as we discover new species from metagenomes[17]. It does not seem credible that indexing technologies improve at a comparable rate.

Unlike the map-to-a-single-reference use-case, where you try to find the single most likely source of a read (and handle ambiguity), here we fully expect a sequence could truly come from multiple genomes – either orthologous hits in many genomes of one species, or different places in one species due to rearrangements, or multiple species. Typically, all alignments (potentially filtered above some quality threshold) would be wanted by the user, which creates a further scaling challenge. Even if we deduplicate (for example there is a vast over-representation of *E. coli* and *Salmonella* in the ENA, many of which are resequencing the same outbreak strains), as the tree of (known bacterial) life grows, so will the total set of *k*-mers and genes, and therefore the indexing problem will continue to explode. We conclude that straightforward alignment against large databases of genomes (e.g. GenBank[18]+RefSeq[19], or AllTheBacteria[20]) may not be feasible if based on indexing everything, or a random subset as data continues to grow exponentially. One caveat to this pessimism is the possibility that a solution might be found to the problem of large-scale Burrows-Wheeler Transform construction[21] – this would automatically handle the redundancy of over-sampling certain species (e.g., AllTheBacteria contains >10,000 species, but 90% of the genomes are from just 25 species), although it is unclear how the long tail of less common species would impact index size or query speed.

To our knowledge, Phylign[22] is the only tool capable of aligning sequences against prokaryotic genomes on the scale of millions of genomes. It is a Snakemake[23] workflow which compresses genomes leveraging phylogenetic information and employs a *k*-mer based method, COBS[24], to prefilter genomes before performing base-level alignment with minimap2[4]. However, prefiltering based on matching 31-mers is most effective for highly similar sequences; as the divergence of the query increases beyond 10%, it becomes very likely to fail the prefilter[25]. It would be satisfying to avoid this limitation.

In this study, we look at the problem from a fresh perspective. While the diversity of bacterial genomes is huge, it is nevertheless constrained, occupying a minuscule fraction of the space of all possible strings of A, C, G, and T. So, we ask - can we use this constraint somehow? In order to be able to do alignment, we need to be able to find anchoring matches shared between a query and a genome, between which we can do straightforward (Smith-Waterman[6], Wavefront alignment algorithm[8], or BlockAligner[9]) alignment. We hypothesize that there is a very low dimensional subspace of sequence space, such that bacterial genomes frequently spend time in this space.

If this is the case, first, we only need to look in this subspace for anchors and can align between them, and second, the subspace does not continually grow with the number of samples, rendering our approach scalable. However, this is a motivational idea, rather than a mathematical formulation. We solve the problem by constructing such a subspace. We create a relatively small set of probes (40,000 *k*-mers) that "cover" the genomes in the database (e.g. 2 million genomes), in the sense that every 500 bp window contains several (median around 10) *k*-mers, each with a 7 bp prefix-match with one of our probes. This is the foundation of LexicMap. Building on this with a range of innovations, detailed below, we are able to develop a standalone alignment tool which is able to align a gene to millions of genomes in seconds.

Our contributions include: 1) modifying LexicHash[26] to compute alignment seeds and support variable-length substring matches, 2) solving the "seed desert" problem of LexicHash to provide a window guarantee for seeds, 3) adding support for suffix matching to increase tolerance to mutations, 4) using a hierarchical index to enable fast and low-memory variable-length seed matching.

Results

Accurate seeding algorithm

In LexicMap, we reimplement the sequence sketching method LexicHash[26], with the support for variable-length substring matches (prefix matching) rather than fixed-length *k*-mers, and use this to compute alignment seeds. We outline the approach here and give full details in the Methods. First, forty thousand 31-mers (called "probes") are generated (Fig.1a), which can "capture" any DNA sequence via prefix matching, as the probes contain all possible 7-bp prefixes. Then, for every reference genome, each probe captures one *k*-mer across the genome as a seed – this is done via the LexicHash, which chooses the *k*-mer which shares the longest prefix with the probe (Fig.1b, Fig.S1).

However, since the captured *k*-mers (seeds) are randomly distributed across a genome, the distances between seeds vary, and there is (initially) no distance guarantee for successive seeds (Fig.S2a). As a result, some genome regions might not be covered with any seed, creating "seed deserts" or sketching deserts[27], where sequences homologous to these regions could fail to align. Generally, seed deserts' sizes and numbers increase with larger genome sizes (Fig.S2a). To address this issue, a second round of seed capture is performed for each seed desert region longer than 200 bp. New seeds spaced about 50 bp apart are added to the seed list of the corresponding probe (Fig.1c and Fig.1d). After filling these seed deserts, a seed distance of 200 bp is guaranteed (Fig.S2b), ensuring that all 500-bp sliding windows contain a minimum of 2 seeds, with a median of 10 in practice (Fig.S2c). Additionally, the seed number of a genome may in practice be lower or higher than the number of probes; the seed number is generally linearly correlated with genome size (Fig.S3). Finally, by allowing variable length prefix-matches, we significantly increase sensitivity compared with a *k*-mer-exact-matching approach, but the method is still vulnerable to variation within the

prefix region; we therefore extended LexicMap to support suffix-matching also (see Methods).

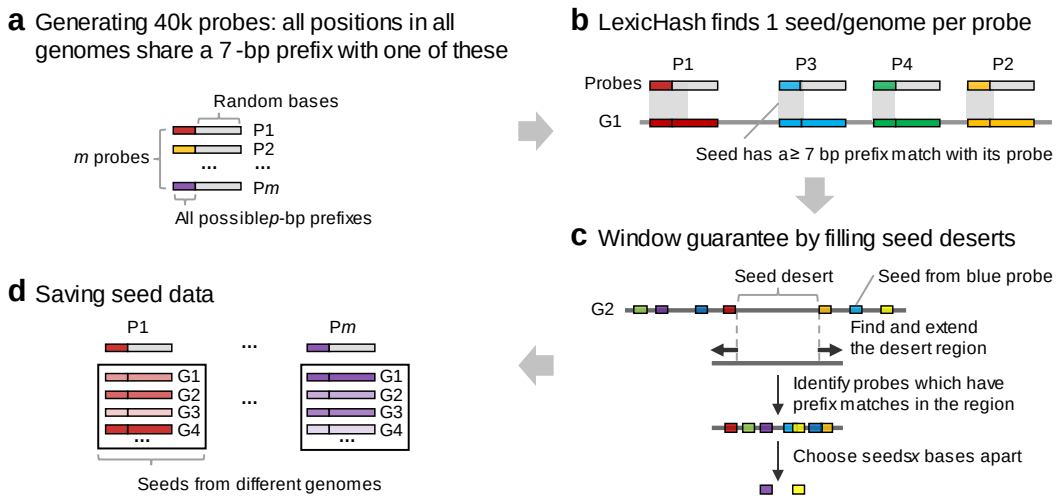

Fig.1 | Seeding scheme of LexicMap for reference database. **a)** A fixed set of 40,000 31-mers (called "probes") are generated, ensuring that their prefixes include every possible 7-mer. Seeds, each prefix-matching one of these, will be found distributed across all database genomes, and chosen in such a way as to have a window guarantee. **b)** LexicHash creates one hash function per probe and when applied to a genome, it finds the *k*-mer with the longest prefix match, which is then stored as a seed. **c)** Each genome is scanned to find seed deserts (regions longer than 200 bp with no seed); every *k*-mer within this region has a 7-mer prefix match with at least one probe (because the probes cover all possible 7-mers), and so seeds can be chosen with spacing of *x* bp (50 by default). **d)** Seeds are stored in a hierarchical index. In fact, although not shown here for simplicity, the number of seeds is doubled to support both prefix and suffix matching (details in Methods).

Scalable indexing strategies

To scale to millions of prokaryotic genomes, input genomes are indexed in batches to limit memory consumption, with all batches merged at the end (Fig.S4a). Within each batch, multiple sequences (contigs or scaffolds) of a genome are concatenated with 1-kb intervals of N's to reduce the sequence scale for indexing. Original coordinates and sequence identifiers are restored after sequence alignment. The complete genome or the concatenated contigs are then used to compute seeds using the generated probes as described above, with intervals and gap regions skipped. Genome sequences are saved in a bit-packed format along with genome information for fast random subsequence extraction. After indexing all reference genomes, each probe captures up to millions of *k*-mers, including position information (genome ID, coordinate, and strand). A scalable hierarchical index compresses and stores seed data for all probes (see Methods and Fig.S4a) and supports fast, low-memory variable-length seed matching, including both prefix and suffix matching (Methods and Fig.S4b).

Efficient variable-length seed matching and alignment

In the searching step, probes from the LexicMap index are used to capture *k*-mers from the query sequence (Fig.2a). Each captured *k*-mer is then searched in the seed data of the corresponding probe to identify seeds that share prefixes or suffixes of at least 15 bp, using a fast and low-memory approach (Methods, Fig.2b). The common prefix or suffix of the query and target seed, along with

the position information, constitutes an anchor. These anchors are grouped by genome ID before chaining. The minimum anchor length of 15 bp ensures search sensitivity, while longer anchors (up to 32 bp) provide higher specificity. Unlike minimizer-based methods such as minimap2, which use fixed-length anchors with a small window guarantee between anchors, LexicMap employs variable-length anchors and does not guarantee a fixed window size between anchors. Consequently, the chaining function (Function 1) assigns more weight to longer anchors and does not consider anchor distance (see Methods). Next, a pseudo-alignment is performed to identify similar regions from the extended chained regions (Fig.2d). Finally, the Wavefront alignment algorithm is used for base-level alignment (Fig.2e). LexicMap's output is a tab-delimited table providing alignment details (Table S1) for filtration or further analysis, and it also supports an intuitive Blast-style format (Fig.S5).

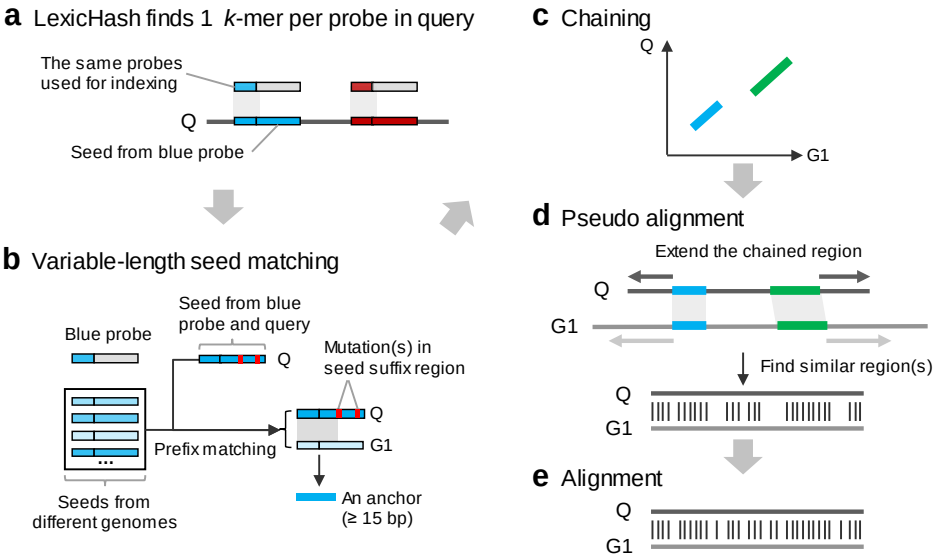

Fig.2 | LexicMap alignment workflow. **a)** The same LexicHash hash functions (one per probe) used in the indexing step, are used here, applied to the query to capture one prefix-matching *k*-mer per probe. **b)** For each probe, scan the seed data for that probe, to find prefix or suffix matches ≥ 15bp. The common prefix or suffix constitutes an anchor. **c)** Chain the variable-length anchors using a modified version of the minimap2 algorithm. **d)** Fast pseudoalignment is followed by **e)** base-level alignment using the WaveFront Alignment algorithm. Note that in (b), only prefix matching is illustrated and suffix matching is not shown for simplicity.

Robustness to sequence divergence

LexicMap supports variable-length seed matches through prefix and suffix matching, allowing greater tolerance to mutations compared to fixed-length seeding methods. To evaluate LexicMap's robustness to sequence divergence, ten bacterial genomes from common species with sizes ranging from 2.1 to 6.3 Mb (Table S2) were used to simulate queries of varying lengths and similarities by introducing SNPs and indels with Badread[28] (see Methods). Blastn[1] was employed as the gold-standard alignment tool, while minimap2 was included in the comparison because it is designed for error-prone long reads and was used in the alignment step of Phylign. COBS was compared as it was used in the prefilter step of Phylign, with a high sensitivity setting (minimum fraction of aligned *k*-mers: 0.33).

Generally, as query identity increases, the alignment rates of the four tools improve, reaching nearly

100% for query identities ≥ 95% when query length is ≥ 500 bp (Fig.3). For queries of 1000 bp and 2000 bp, minimap2 consistently achieves the highest alignment rates at lower query identities, followed by LexicMap and Blastn. COBS shows a steeper drop-off in alignment rates at query identities below 95%, which is expected given that it relies on comparing fractions of matched *k*-mers (k = 31). An exception is observed for 250 bp queries, where the alignment rate of COBS is higher than other tools for query identities lower than 83%, likely due to the high false positive rate setting of COBS. For 500 bp queries, LexicMap's alignment rates are slightly lower than those of Blastn when query identities range from 90% to 93%. However, LexicMap outperforms Blastn at query identities below 90% and surpasses minimap2 at query identities below 83%. A similar trend is observed for 250 bp queries. Overall, LexicMap demonstrates higher sensitivity than Blastn as query divergence decreases from 90% to 80% for queries ≥ 1 kb.

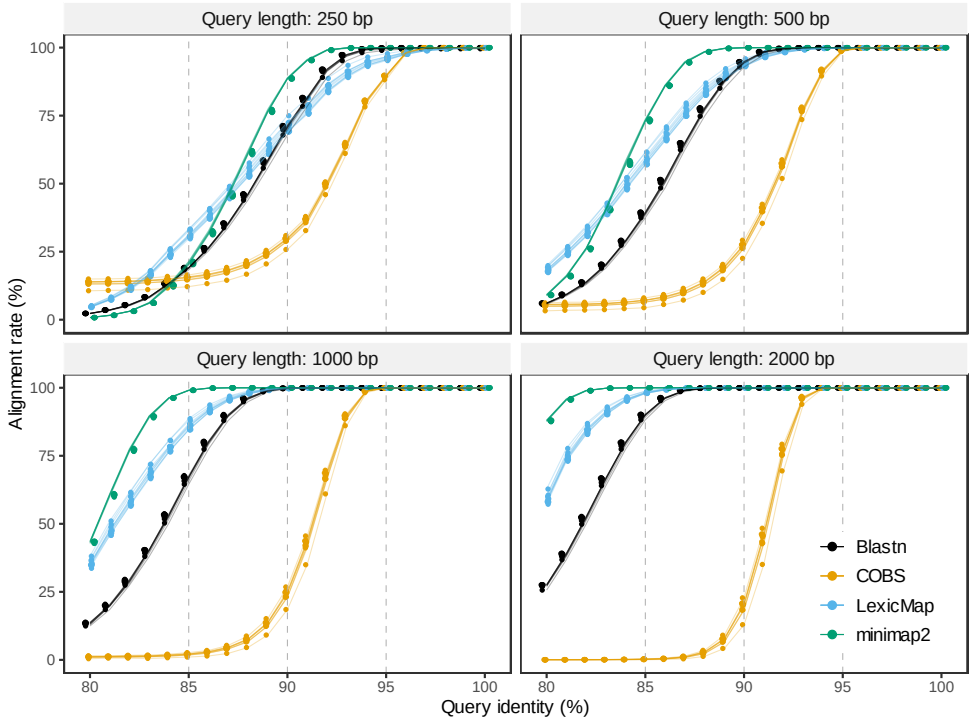

Fig.3 | Robustness of aligners to sequence divergence. This is measured by simulating 250, 500, 1000, and 2000 bp reads with coverage of 30× from 10 bacterial genomes, adding mutations to achieve sequence divergence between 0 and 20%, and then aligning back to the source genome. COBS (yellow) is a *k*-mer index, so here we show what proportion of the "reads" was detected as being present in the source genome (rather than being aligned); this falls off very rapidly as similarity drops because each base disagreement loses an entire 31-mer. For the other tools, we measure the proportion of reads that are correctly aligned back to the source genome.

Indexing performance

To evaluate the performance of LexicMap, we compared it with the widely used classic sequence alignment tool Blastn and the state-of-the-art approach Phylign. We used three datasets to create the databases (see Methods): the GTDB r214 complete dataset[29] with 402,538 prokaryotic assemblies, the AllTheBacteria v0.2 high-quality (HQ) dataset with 1,858,610 bacterial assemblies, and the GenBank+RefSeq with 2,340,672 prokaryotic assemblies (downloaded on February 15, 2024).

Since the indexing procedure of Phylign is not generalized, we used the pre-built index for AllTheBacteria.

In terms of index sizes, Phylign has the smallest size (Table 1) while LexicMap has index sizes approximately 2.5 times larger than Blastn. For indexing time and memory usage, Blastn is more than 3× faster than LexicMap and uses significantly less memory. LexicMap uses almost twice as much memory for the GenBank+RefSeq dataset compared to the AllTheBacteria HQ dataset, as the genomes in the former are more diverse within each batch.

Table 1 | LexicMap indexing performance on three datasets. Tbp: terabase; MB: megabyte (base 1000); GB: gigabyte (base 1000). LexicMap built indexes with the genome batch size of 5,000 for GTDB and 25,000 for the AllTheBacteria dataset and the GenBank+RefSeq dataset.

Dataset	Assemblies	Bases	Tool	Index size	Time	RAM
GTDB complete	402,538	1.5 Tbp	LexicMap	973 GB	10 h 36 m	63.3 GB
			Blastn	387 GB	3 h 11 m	718 MB
AllTheBacteria HQ	1,858,610	7.5 Tbp	LexicMap	4.26 TB	48 h 08 m	88.6 GB
			Blastn	1.93 TB	14 h 03 m	2.9 GB
			Phylign	248 GB	/	/
GenBank+RefSeq	2,340,672	9.2 Tbp	LexicMap	5.43 TB	54 h 33 m	178.3 GB
			Blastn	2.37 TB	14 h 04 m	4.3 GB

Alignment performance

Four datasets representing different types of queries were used to evaluate alignment performance (see Methods): a comparatively rare gene (with thousands of genome hits) - SecY from *Enterococcus faecalis*, a 16S rRNA gene from *Escherichia coli* with a large number of targets, a plasmid, and 1033 different AMR genes (batch queries).

First, we searched the queries with LexicMap and Blastn against the GTDB complete index, which includes 402,538 archaea and bacterial assemblies (see Table 2) - this is a set of genomes that cover the diversity of known bacterial and archaeal genomes, with some redundancy but far less so than AllTheBacteria. Generally, Blastn returns more genome hits than LexicMap, except for the 16S rRNA gene. In terms of search time, LexicMap is significantly faster than Blastn for single queries, being 125×, 10×, and 4.5× faster, respectively. For batch queries, LexicMap produces a comparable number of genome hits with query coverage >50%, and its search time is slightly shorter than that of Blastn. Regarding memory usage, LexicMap requires less than 5 GB for single queries and 16.3 GB for 1033 AMR genes, while Blastn consumes more than 350 GB across all types of queries.

Next, we compared LexicMap with Phylign on the AllTheBacteria dataset, which contains 1,858,610 bacterial genomes, including some species that are highly over-sampled. Blastn could not be run due to its requirement of more than 2000 GB of memory. Across all queries, LexicMap returns significantly more genome hits than Phylign. For single queries, LexicMap takes much less time than Phylign both in local or cluster mode (100 nodes), while also using less memory. Regarding batch querying, LexicMap is slightly faster than Phylign in local mode and slightly

slower than Phylign in cluster mode, although in both cases LexicMap finds around 12× more alignments.

LexicMap achieves similar performance on the GenBank+RefSeq dataset (234 million prokaryotic genomes) as it does on the AllTheBacteria dataset (Table S3). For the SecY gene, LexicMap returns 37,164 genome hits in 36 seconds with 4.1 GB memory. For the 16S rRNA gene, it returns 1,949,496 genome hits in 10 minutes and 41 seconds with 14.1 GB memory. For the plasmid, 544,619 genome hits are returned in 19 minutes and 20 seconds with 19.3 GB memory.

Table 2 | Alignment performance benchmarks on GTDB complete dataset. Sequence identifiers are in Methods. qcov: query coverage (the percentage of aligned query bases in an alignment).

Query	Query length	Tool	Genome hits	Genome hits (qcov > 50%)	Time	RAM
a rare gene	1,299 bp	LexicMap	5,170	5,143	**17 s**	**1.4 GB**
		Blastn	**7,121**	**6,177**	2,171 s	351.2 GB
a 16S rRNA gene	1,542 bp	LexicMap	**303,925**	**278,141**	235 s	4.4 GB
		Blastn	301,197	277,042	2,353 s	378.4 GB
a plasmid	52,830 bp	LexicMap	63,108	1,190	**499 s**	**4.6 GB**
		Blastn	**69,311**	**2,308**	2,262 s	364.7 GB
1033 AMR genes	1 kb (median)	LexicMap	3,867,003	2,228,339	**4,350 s**	**16.3 GB**
		Blastn	**5,357,772**	**2,240,766**	4,686 s	442.1 GB

Table 3 | Alignment performance benchmarks on AllTheBacteria high-quality dataset*. Sequence identifiers are in Methods. qcov: query coverage (the percentage of aligned query bases in an alignment).

Query	Query length	Tool	Genome hits	Genome hits (qcov > 50%)	Time	RAM
a rare gene	1,299 bp	LexicMap	**27,963**	27,953	**31 s**	**3.4 GB**
		Phylign_local	7,936	/	30 m 48 s	77.6 GB
		Phylign_cluster	7,936	/	28 m 33 s	/
a 16S rRNA gene	1,542 bp	LexicMap	**1,857,761**	1,740,000	**9 m 36 s**	**14.9 GB**
		Phylign_local	1,017,765	/	130 m 33 s	77.0 GB
		Phylign_cluster	1,017,765	/	86 m 41 s	/
a plasmid	52,830 bp	LexicMap	**468,821**	3,618	**15 m 55 s**	**15.7 GB**
		Phylign_local	46,822	/	47 m 33 s	82.6 GB
		Phylign_cluster	46,822	/	39 m 34 s	/
1033 AMR genes	1 kb (median)	LexicMap	**21,288,000**	12,148,64	138 m 55 s	**49.9 GB**
		Phylign_local	1,135,215	/	156 m 08 s	85.9 GB
		Phylign_cluster	1,135,215	/	**133 m 49 s**	/

* Query coverage is not available in the Phylign output result. Alignment memory can not be accurately measured for Phylign running in multiple cluster nodes.

Discussion

It is a worthwhile exercise to consider LexicMap both from the perspective of the data, and how that is likely to change over the coming years, and then from the perspective of algorithms and data structures.

The vast majority of the (cellular) tree of life consists of bacteria and archaea. We continue to expand (i.e. discover more of) the tree through metagenomic sequencing, and although the rate of discovery of new phyla has dropped, this is not the case for lower taxa. Taken together with the prevalence of horizontal gene transfer in prokaryotes and a high number of mobile genetic elements and their cargo, the levels of diversity are extremely high and continue to grow. Further, the amount of clinical sequencing and deposition in archives continues to grow our collection of pathogen sequence data, providing a year-by-year perspective on real-time evolution and horizontal gene transfer. Together, these lead to a progressively more challenging situation for alignment to representative (e.g. RefSeq, GTDB) or complete but considerably redundant (e.g. Genbank, AllTheBacteria) databases[17]. The primary use cases of alignment to such an index are: determining the host range of a mobile element or gene, locating probable orthologs for further analysis, or determining where a specific sequence has been seen before. More broadly, the power to do this alignment against all prior prokaryotic sequences will enable a wide range of specific analyses, just as BLAST has done with smaller datasets. Some concrete examples were seen in how the approximate (*k*-mer based) matching to the 661k dataset[30] has been used to search for plasmids[31-36], adhesins[36], diversity of vaccine targets[37], mutations of interest[31] and phages[38].

Alignment is a classical bioinformatics problem, with recent developments focusing on increasing speed and scaling with sequence length, motivated by the long reads from modern sequence technologies, and from the desire to do pairwise alignments of multiple genomes of a species when building pangenomes[9,39], However the problem of scaling alignments to large databases of bacterial/archaeal genomes remains open – NCBI Blast searches an exponentially dropping proportion of all bacterial data[22], and scalable alternatives either abandon alignment in favour of *k*-mer approximations[24,40-46], or use a *k*-mer filter to choose which genomes to obtain from a phylogenetically compressed archive[22], with an associated cost in sensitivity. The approaches used for single reference alignment – full-text compressed indexes such as the BWT, or minimizer-based methods – appear challenging at this scale. LexicMap takes an unusual approach, constructing a fixed set of 40000 probes (*k*-mers) which are guaranteed to have multiple (around 10 in this study) prefix matches in every 500bp window of every genome in the database. The *k*-mer with the best prefix match for each probe in each genome is (called a seed and) stored in a hierarchical index, which can be used for alignment. This use of variable-length prefix-matching enables sensitive nucleotide alignment for queries above 500 bp long (a tunable threshold). This approach is, we believe, novel and of independent interest – there is room for innovation in designing smaller sets of probes and understanding the nature of the space they define[6,47-51].

Benchmarking is always tricky and dependent on parameter choices, but we have tried to use it to understand performance differences. In the evaluation of alignment rate of diverged sequences (i.e. how successful an aligner is at remapping a mutated sequence back to the original genome) on a test set of 10 genomes, minimap2 shows the highest sensitivity, followed by LexicMap and Blastn.

Interestingly, LexicMap has higher sensitivity than Blastn as query divergence decreases from 90% to 80% for queries ≥ 1 kb. The seed length and density might explain the performance difference - Minimap2 and LexicMap both require a minimum seed length of 15 bp while Blastn's default task mode, megablast, uses a word size of 18 bp. Regarding the seed density, minimap2 chooses minimizers within windows of 10 bp, while LexicMap's seed distances vary with a big window guarantee of 200 bp, whereas Blastn checks all 18-mers and extends them. However, once you scale up to the largest databases (RefSeq/GenBank and AllTheBacteria), the only other tool which could index and align to the data was Phylign, compared with which LexicMap was 3-60× faster and 1.8-10× more sensitive.

Finally, in terms of user experience, the main advantages of LexicMap compared with current approaches include: 1) It provides direct alignment without a lossy prefilter step. 2) All possible matches, including multiple copies of genes, are returned. 3) The alignment is fast and memory-efficient. 4) Unlike minimizers-based methods, seeds in LexicMap are interpretable and several utility commands are available to interpret the probe and seed data. 5) LexicMap is easy to install on multiple operating systems and can be used as a standalone tool without needing a workflow manager or compute cluster. LexicMap mainly supports small genomes including complete or partially assembled prokaryotic, viral, and fungi genomes. The maximum supported sequence length is 268,435,456 bp (2^{28}), so it can in principle also be applied to bigger genomes like human genomes with a maximum chromosome size of 248 Mb.

In terms of limitations, LexicMap achieves a low memory footprint by storing a very large index on disk (5.43 TB for 2.34 million prokaryotic assemblies in GenBank+RefSeq); it would be desirable in future to reduce the size of this index. Finally, LexicMap is optimized for a small number of queries, batch searching speed is to be improved in future work. Nevertheless, in providing the means to do full alignment to the full corpus of bacterial/archaeal genomes, we believe LexicMap answers a growing need, and we hope it will prove useful to many researchers.

Methods

Probe generation

Probes, also referred to as "masks" in the LexicHash paper, consist of a fixed number (m = 40,000 by default) of k-mers ($k \leq 32$, k = 31 by default), which "capture" DNA sequences via prefix matching. A probe consists of two parts – the p-bp prefix and the remaining bases. To enable probes to match all possible reference and query sequences via prefix matching, all permutations of p-mers are generated as the base prefix set. The length of the prefix (p) is calculated to ensure that the total number of possible prefixes (4^p) does not exceed the total number of probes (m). For instance, when m = 40,000, p equals 7. Then, the base prefixes are duplicated to reach the number of probes m. Next, the suffixes of probes are randomly generated. The p'-bp ($p' = p + 1$) prefixes are required to be distinct to enable fast locating of the probe via the prefix of a k-mer through an array data structure – this accelerates multiple computations in the seed desert filling step (next section). Finally, these m k-mers serve as probes that can match all potential sequence regions in both reference genomes and query sequences.

Seed computation

K-mers are encoded as 64-bit unsigned integers, with a binary coding scheme of A=00, C=01, G=10, and T=11. Following the original implementation of LexicHash, the hash value between a probe and a *k*-mer is computed using a bitwise XOR operation. The value of this hash is that smaller hash values indicate longer common prefixes between the probe and the *k*-mer, while a hash value of zero means the probe and the *k*-mer are identical.

For each reference genome or query sequence, all *k*-mers from both forward and backward strands are compared with probes that share the same *p*-bp prefix. Each probe retains only the *k*-mer with the minimum hash value, which might in principle be located at more than one position. A 64-bit integer encodes each position's information, including the genome batch index (17 bits, see "indexing" section), genome index (17 bits), position (28 bits), the strand (1 bit), and the seed direction flag (1 bit, see "Seed data storage and variable-length seed matching" section). Ultimately, each probe captures one *k*-mer (as a seed) across the entire reference genome or query sequence, which shares the longest prefix with the probe.

However, due to the random distribution of captured *k*-mers (seeds), the distances between these seeds vary, and there is no guarantee of consistent distance between successive seeds. Consequently, some regions of the sequence might remain uncovered by seeds, leading to what are known as "seed deserts". These deserts are problematic because they can cause sequences homologous to the regions to fail to align. To address this issue, regions identified as seed deserts, which are longer than a certain threshold (200 bp by default), are extended by 1 kb both upstream and downstream. A second round of seed capture is then performed in these extended regions, and new seeds spaced about *x* (50 by default) bp apart within the region are added to the index of the corresponding probes. After filling these seed deserts, the total number of seeds may exceed the initial value of *m*.

Indexing

The input of LexicMap is a list of microbial genomes, with the sequences of each genome stored in separate FASTA files. These files can be in plain or compressed formats such as gzip, xz, zstd, or bzip2. Each file must have a distinct genome identifier in the file name. To limit memory consumption, genomes are indexed in batches, and all batches are merged at the end (Fig.S4a).

In each batch, genomes larger than a genome size threshold (15 Mb by default), such as non-isolate assemblies, are skipped. Additionally, unwanted sequences within genomes, such as plasmids, can be optionally discarded using regular expressions to match sequence names. Multiple sequences (contigs or scaffolds) of a genome are concatenated with 1-kb intervals of N's, to reduce the scale of sequences to be indexed. The original coordinates will be restored after sequence alignment. The complete genome or the concatenated contigs are then used to compute seeds with generated probes, as described above. Before this, any degenerated bases are converted to their corresponding alphabet-first bases (e.g., N is converted to A). The genome sequence is then saved in a bit-packed format (2 bits per base), along with associated genome information (genome ID, size, and sequence IDs and lengths of all contigs) to enable fast random subsequence extraction. Simultaneously, seeds and their positional information are appended to their corresponding probes and saved as seed files. After processing all genomes in the batch, these seed files are merged using an external sorting

method once all batches have been indexed.

Seed data storage and variable-length seed matching

After indexing with all *n* reference genomes, each probe captured *n* or more seeds (*k*-mers, encoded as 64-bit integers), with each seed potentially having one or more positions (also represented as 64-bit integers), and there are *m* probes in total (40,000 by default). To scale to millions of prokaryotic genomes, the storage of seed data needs to be both compact and efficient for querying. Since the seeds of different probes are independent, the seed data are saved into *c* chunk files to enable parallel querying (Fig.S4a). In each chunk file, the seed data of approximately *m*/*c* probes are simply concatenated. For each probe, all seeds are sorted in alphabetical order, and the varint-GB[52] algorithm is used to compress every two seeds along with their associated position counts. Since seeds captured by the same probe share common prefixes, the differences between two successive seeds are small, as are the position counts. As a result, these values can be stored using as little as two bytes instead of 16 bytes.

Unlike the approach in the LexicHash paper, where captured *k*-mers from each probe are stored in a prefix tree in main memory, here they are alphabetically sorted and saved in a list-like structure within files. To enable efficient variable-length prefix matching of seeds, an index is created for each seed data file (Fig.S4b). This index functions similarly to a table of contents in a dictionary, storing a list of marker *k*-mers along with their offsets (pages) in the seed data file. Each marker *k*-mer is the first one with a specific *p”*-bp subsequence (*p”* = 5 by default) following the *p*-bp prefix (all seeds of a probe share the same *p*-bp prefix). For a query sequence, one *k*-mer is captured by each probe and searched within the corresponding probe’s seed data to return seeds that share a minimum length of prefix with the query *k*-mer. For example, searching with CATGCT for seeds (with *p* = 2, *p”* = 1) that have at least 4 bp common prefixes is equivalent to finding seeds in the range of CATGAA to CATGTT. The process starts by extracting the *p”*-bp subsequence from CATGAA (in this case, T) to locate the marker *k*-mer (e.g., CATCAC) with the same *p”*-bp subsequence in the same region. The offset information (page) of this marker *k*-mer is then used as the starting point for scanning seeds within the *k*-mer range.

The index structure described above is extended to support the suffix matching of seeds. During the indexing phase, after a seed *k*-mer is saved into the seed data of its corresponding probe, the *k*-mer is reversed and added to the seed data of the probe that shares the longest prefix with the reversed *k*-mer. Additionally, the last bit of each position data is used as a seed direction flag, indicating that the seed *k*-mer is reversed. As a result, all seeds are doubled: there are “forward seeds” for prefix matching and “reversed seeds” for suffix matching, which is achieved through prefix matching of the reversed seeds. In the seed matching process, two rounds of matching are performed. The first round involves prefix matching (as described above) in the forward seed data. In the second round, the query *k*-mer is reversed and searched in the reversed seed data of the probe that shares the longest prefix with the reversed *k*-mer.

Chaining

After searching all captured *k*-mers from a query in the seed data of all probes, seed pairs (anchors) with different matched prefix/suffix lengths *l* are returned. These anchors indicate matched regions

between the query sequences [y, $y + l$ - 1] on the strand s_y and reference genome [x, $x + l$ - 1] on strand s_x. First, the anchors are grouped by the genome IDs. Within each group, the anchors are sorted by the following criteria: 1) Ascending order of start positions in the query sequence. 2) Descending order of end positions in the query sequence. 3) Ascending order of start positions in the target genome. Only one anchor is kept for duplicated anchors, and any inner anchors that are nested within other anchors are removed. Next, a chaining function modified from Minimap2 is applied to chain all possible colinear anchors:

$$f(i) = \max\{ \max_{\substack{i>j\geq 1 \\ x_i - G < x_j \leq x_i \\ y_i - G < y_j \leq y_i \\ d_{ji} \leq D}} \{f(j) + w_i - \gamma(g_{ji})\}, w_i\} \quad (1)$$

Here, $f(i)$, $f(j)$ are scores for anchors a_i and a_j, respectively. The anchor weight $w_i = 0.1 \times l_i^2$, where l_i is the length of the anchor a_i and $p' \leq l_i \leq k$, with p' being the minimum anchor length (15 by default). The seed distance $d_{ji} = \max\{|y_i - y_j|, |x_i - x_j|\}$ and D is the maximum seed distance (1,000 by default). The seed gap $g_{ji} = \big||y_i - y_j| - |x_i - x_j|\big|$, and G is the maximum gap (200 by default). The gap penalty is calculated as $\gamma(g) = 0.1 \times g + 0.5 \log_2(g)$. Additionally, a filter is applied to avoid non-monotonic increase or decrease in coordinates. For cases where there is only one anchor, the length of the anchor must be no less than a threshold p'' = 17.

Alignment

In contrast to the case of minimizers or syncmers[53], seeds in LexicMap do not have a small window guarantee. Hence a pseudo alignment algorithm is further used to find similar regions from the chained region, extended by 1-kb in the upstream and downstream directions. *K*-mers (k = 31) on both strands of the query sequence are stored in a prefix tree, and *k*-mers of a target region are queried in the prefix tree to return matches of $l_i{}'$ $(11 \leq l_i{}' \leq 31)$ in the extended chained region. After removing duplicated anchors as above, the anchors are chained with the score function:

$$f'(i) = \max\{ \max_{\substack{i>j\geq 1 \\ x_i - G' < x_j \leq x_i \\ y_i - G' < y_j \leq y_i}} \{f'(j) + l'_i - g'_{ji}\}, l'_i\} \quad (2)$$

where G' is the maximum gap (20 by default), seed distance $g'_{ji} = \big||y_i - y_j| - |x_i - x_j|\big|$. And the chaining is banded (50 by default). Besides, chained regions across the interval regions between contigs are split.

Similar regions between the reference and query sequences are further aligned with a reimplemented Wavefront alignment algorithm (https://github.com/shenwei356/wfa, v0.3.0), with a gap-affine penalty and the adaptive reduction heuristic.

Simulating mutations

Ten bacterial genomes (more details in Table S2) with genome sizes ranging from 2.1~6.3 Mb were used for simulating queries with Badread[28] v0.4.1 and SeqKit[54] v2.8.2. The command, "badread simulate --seed 1 --reference $ref --quantity 30x --length $qlen,0 --identity $ident,$ident,0 --

error_model random --qscore_model ideal --glitches 0,0,0 --junk_reads 0 --random_reads 0 --chimeras 0 --start_adapter_seq “” --end_adapter_seq “” | seqkit seq -g -m $qlen | seqkit grep -i -s -v -p NNNNNNNNNNNNNNNNNNNNN -o $ref.i$ident.q$qlen.fastq.gz”, simulated reads with a genome coverage of 30×, a mean length of 250, 500, 1,000 or 2,000 bp and mean percentage identities ranging from 80% to 100% with SNPs and indels. LexicMap v0.4.0, Blast++ v2.15.0, Minimap2 v2.26-r1175, COBS (https://github.com/iqbal-lab-org/cobs, iqbal-lab-org fork v0.3.0) were used to search simulated reads against the reference genomes. Each tool built an index for one genome and searched/aligned reads with the corresponding index. LexicMap built indexes and searched with default parameters (k = 31, m = 40,000). Blast built indexes and searched with default parameters (megablast). Minimap2 searched with the “map-ont” mode. COBS built compact indexes with k = 31 and a false positive rate of 0.3 and searched reads with a threshold of 0.33.

Benchmarking

LexicMap v0.4.0, Blast++ v2.15.0 and Phylign (AllTheBacteria fork https://github.com/AllTheBacteria/Phylign, commit 9fc65e6) were used for benchmarking. GTDB r214 complete dataset with 402,538 prokaryotic assemblies was downloaded with genome_updater (https://github.com/pirovc/genome_updater, v0.6.3). GenBank+RefSeq dataset with 2,340,672 prokaryotic assemblies was downloaded with genome_updater on February 15, 2024. AllTheBacteria v0.2 high-quality genome dataset with 1,858,610 bacteria genomes and the Phylign index were downloaded from https://ftp.ebi.ac.uk/pub/databases/AllTheBacteria/Releases/0.2/. All files were stored on a server with HDD disks. All tests were performed in cluster nodes running with Intel(R) Xeon(R) Gold 6336Y CPU @ 2.40GHz with 500GB of RAM.

For indexing, LexicMap built indexes with k = 31, m = 40,000, and the genome batch size was 5,000 for GTDB and 25,000 for the AllTheBacteria dataset and the GenBank+RefSeq dataset. Blastn databases were made with default parameters. The Phylign index was built with default parameters, including k = 31 and a false-positive rate of 0.3 for the COBS index; since the index building involved three workflows with multiple steps in multiple cluster nodes, the memory and time could not be measured accurately.

Four query datasets were used for evaluating alignment performance: a preprotein translocase subunit SecY gene CDS sequence from an *Enterococcus faecalis* strain (NZ_CP064374.1_cds_WP_002359350.1_906), a 16S rRNA gene from *Escherichia coli* str. K-12 substr. MG1655 (NC_000913.3:4166659-4168200), a plasmid from a *Serratia nevei* strain (CP115019.1), and 1033 AMR genes from the Bacterial Antimicrobial Resistance Reference Gene Database (PRJNA313047). LexicMap and Blastn both used all 48 CPUs and returned all possible matches with a minimum percentage identity of 70%. The main parameters of Phylign included, threads = 48, cobs_kmer_thres = 0.33, minimap_preset = "asm20", nb_best_hits = 5,000,000, max_ram_gb = 100. For the cluster mode, the maximum number of Slurm jobs was set to 100.

Code availability

LexicMap is an open-source standalone tool implemented in Go under the MIT license at https://github.com/shenwei356/LexicMap, with freely available statically linked executable binary

files for common operating systems and CPU types. Two main subcommands *index* and *search* are used to create an index and perform searching, respectively, and several utility subcommands are available for interpreting the index data and extracting indexed sequences.

Acknowledgements

This study was supported by grants from the National Natural Science Foundation of China (82341112), Chinese Scholarship Council scholarship (202308500105), EMBL Visitor/Sabbatical Programme fellowship, Remarkable Innovation-Clinical Research Project and Joint Project of Pinnacle Disciplinary Group of The Second Affiliated Hospital of Chongqing Medical University.

We thank Shuyi Wang (Peking University People's Hospital, China), Leah Roberts (Queensland University of Technology, Australia), Songbai Cai and Liuyang Zhao (Chongqing Medical University, China), and Rachel Colquhoun (Edinburgh University, UK) for using LexicMap and giving valuable feedback. We thank Daniel Anderson for suggesting test datasets. We thank Pengfei Wang (University of Montpellier, France) for comments on the manuscript and visualization. We thank Daniel Anderson, Martin Hunt, Daria Frolova, and John Lees for fruitful discussions.

Author contributions

W.S. and Z.I. designed the project. Z.I. managed the project. W.S. implemented the software. Both authors contributed to the data interpretation and writing of the manuscript. Both authors reviewed and approved the manuscript.

Competing interests

The authors declare no competing interests.

References

1 Camacho, C. *et al.* BLAST+: architecture and applications. *BMC Bioinformatics* **10**, 421, doi:10.1186/1471-2105-10-421 (2009).

2 Li, H. & Durbin, R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* **25**, 1754-1760, doi:10.1093/bioinformatics/btp324 (2009).

3 Langmead, B. & Salzberg, S. L. Fast gapped-read alignment with Bowtie 2. *Nat Methods* **9**, 357-359, doi:10.1038/nmeth.1923 (2012).

4 Li, H. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* **34**, 3094-3100, doi:10.1093/bioinformatics/bty191 (2018).

5 Needleman, S. B. & Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol* **48**, 443-453, doi:10.1016/0022-2836(70)90057-4 (1970).

6 Smith, T. F. & Waterman, M. S. Identification of common molecular subsequences. *J Mol Biol* **147**, 195-197, doi:10.1016/0022-2836(81)90087-5 (1981).

7 Gotoh, O. An improved algorithm for matching biological sequences. *J Mol Biol* **162**, 705-708, doi:10.1016/0022-2836(82)90398-9 (1982).

8 Marco-Sola, S., Moure, J. C., Moreto, M. & Espinosa, A. Fast gap-affine pairwise alignment using the wavefront algorithm. *Bioinformatics* **37**, 456-463, doi:10.1093/bioinformatics/btaa777 (2021).

9 Liu, D. & Steinegger, M. Block Aligner: an adaptive SIMD-accelerated aligner for sequences and position-specific scoring matrices. *Bioinformatics* **39**, doi:10.1093/bioinformatics/btad487 (2023).

10 Groot Koerkamp, R. & Ivanov, P. Exact global alignment using A* with chaining seed heuristic and match pruning. *Bioinformatics* **40**, doi:10.1093/bioinformatics/btae032 (2024).

11 Bzikadze, A. V. & Pevzner, P. A. UniAligner: a parameter-free framework for fast sequence alignment. *Nat Methods* **20**, 1346-1354, doi:10.1038/s41592-023-01970-4 (2023).

12 Shao, H. & Ruan, J. BSAlign: A Library for Nucleotide Sequence Alignment. *Genomics, Proteomics & Bioinformatics*, qzae025, doi:10.1093/gpbjnl/qzae025 (2024).

13 Ferragina, P. & Manzini, G. in *Proceedings 41st Annual Symposium on Foundations of Computer Science.* 390-398.

14 Colquhoun, R. M. *et al.* Pandora: nucleotide-resolution bacterial pan-genomics with reference graphs. *Genome Biol* **22**, 267, doi:10.1186/s13059-021-02473-1 (2021).

15 Tonkin-Hill, G., Corander, J. & Parkhill, J. Challenges in prokaryote pangenomics. *Microb Genom* **9**, doi:10.1099/mgen.0.001021 (2023).

16 Liao, W. W. *et al.* A draft human pangenome reference. *Nature* **617**, 312-324, doi:10.1038/s41586-023-05896-x (2023).

17 Hug, L. A. The ever-changing tree of life. *Nat Microbiol* **9**, 1906-1908, doi:10.1038/s41564-024-01768-w (2024).

18 Sayers, E. W. *et al.* GenBank 2024 Update. *Nucleic Acids Res* **52**, D134-D137, doi:10.1093/nar/gkad903 (2024).

19 Haft, D. H. *et al.* RefSeq and the prokaryotic genome annotation pipeline in the age of metagenomes. *Nucleic Acids Res* **52**, D762-D769, doi:10.1093/nar/gkad988 (2024).

20 Hunt, M., Lima, L., Shen, W., Lees, J. & Iqbal, Z. AllTheBacteria - all bacterial genomes assembled, available and searchable. *bioRxiv*, 2024.2003.2008.584059, doi:10.1101/2024.03.08.584059 (2024).

21 Boucher, C. *et al.* Prefix-free parsing for building big BWTs. *Algorithms Mol Biol* **14**, 13, doi:10.1186/s13015-019-0148-5 (2019).

22 Břinda, K. *et al.* Efficient and Robust Search of Microbial Genomes via Phylogenetic Compression. *bioRxiv*, 2023.2004.2015.536996, doi:10.1101/2023.04.15.536996 (2024).

23 Molder, F. *et al.* Sustainable data analysis with Snakemake. *F1000Res* **10**, 33, doi:10.12688/f1000research.29032.2 (2021).

24 Bingmann, T., Bradley, P., Gauger, F. & Iqbal, Z. in *String Processing and Information Retrieval.* (eds Nieves R. Brisaboa & Simon J. Puglisi) 285-303 (Springer International Publishing).

25 Shen, W. *et al.* KMCP: accurate metagenomic profiling of both prokaryotic and viral populations by pseudo-mapping. *Bioinformatics* **39**, doi:10.1093/bioinformatics/btac845 (2023).

26 Greenberg, G., Ravi, A. N. & Shomorony, I. LexicHash: sequence similarity estimation via lexicographic comparison of hashes. *Bioinformatics* **39**, btad652, doi:10.1093/bioinformatics/btad652 (2023).

27 Guillaume, M., Elder, C. S. & Carl, K. k-nonical space: sketching with reverse complements. *bioRxiv*, 2024.2001.2025.577301, doi:10.1101/2024.01.25.577301 (2024).

28 Wick, R. R. Badread: simulation of error-prone long reads. *Journal of Open Source Software* **4**, 1316, doi:10.21105/joss.01316 (2019).

29 Parks, D. H. *et al.* GTDB: an ongoing census of bacterial and archaeal diversity through a phylogenetically consistent, rank normalized and complete genome-based taxonomy. *Nucleic Acids Res* **50**, D785-D794, doi:10.1093/nar/gkab776 (2022).

30 Blackwell, G. A. *et al.* Exploring bacterial diversity via a curated and searchable snapshot of archived DNA

sequences. *PLOS Biology* **19**, e3001421, doi:10.1371/journal.pbio.3001421 (2021).

31 Smith, T. M. *et al.* Rapid adaptation of a complex trait during experimental evolution of Mycobacterium tuberculosis. *Elife* **11**, doi:10.7554/eLife.78454 (2022).

32 Lassalle, F. *et al.* Genomic epidemiology reveals multidrug resistant plasmid spread between Vibrio cholerae lineages in Yemen. *Nat Microbiol* **8**, 1787-1798, doi:10.1038/s41564-023-01472-1 (2023).

33 Mason, L. C. E. *et al.* The evolution and international spread of extensively drug resistant Shigella sonnei. *Nat Commun* **14**, 1983, doi:10.1038/s41467-023-37672-w (2023).

34 Hu, Y., Moran, R. A., Blackwell, G. A., McNally, A. & Zong, Z. Fine-Scale Reconstruction of the Evolution of FII-33 Multidrug Resistance Plasmids Enables High-Resolution Genomic Surveillance. *mSystems* **7**, e0083121, doi:10.1128/msystems.00831-21 (2022).

35 Smits, W. K. *et al.* Sequence-Based Identification of Metronidazole-Resistant Clostridioides difficile Isolates. *Emerg Infect Dis* **28**, 2308-2311, doi:10.3201/eid2811.220615 (2022).

36 Malaka De Silva, P. *et al.* A tale of two plasmids: contributions of plasmid associated phenotypes to epidemiological success among Shigella. *Proc Biol Sci* **289**, 20220581, doi:10.1098/rspb.2022.0581 (2022).

37 Croucher, N. J. Immune interface interference vaccines: An evolution-informed approach to anti-bacterial vaccine design. *Microb Biotechnol* **17**, e14446, doi:10.1111/1751-7915.14446 (2024).

38 Lassalle, F. *et al.* Genomic epidemiology of the cholera outbreak in Yemen reveals the spread of a multi-drug resistance plasmid between diverse lineages of <em>Vibrio cholerae</em&gt. *bioRxiv*, 2022.2008.2024.504966, doi:10.1101/2022.08.24.504966 (2022).

39 Marco-Sola, S. *et al.* Optimal gap-affine alignment in O(s) space. *Bioinformatics* **39**, btad074, doi:10.1093/bioinformatics/btad074 (2023).

40 Bradley, P., den Bakker, H. C., Rocha, E. P. C., McVean, G. & Iqbal, Z. Ultrafast search of all deposited bacterial and viral genomic data. *Nat Biotechnol* **37**, 152-159, doi:10.1038/s41587-018-0010-1 (2019).

41 Marchet, C. & Limasset, A. Scalable sequence database search using partitioned aggregated Bloom comb trees. *Bioinformatics* **39**, i252-i259, doi:10.1093/bioinformatics/btad225 (2023).

42 Alanko, J. N., Vuohtoniemi, J., Mäklin, T. & Puglisi, S. J. Themisto: a scalable colored k-mer index for sensitive pseudoalignment against hundreds of thousands of bacterial genomes. *Bioinformatics* **39**, i260-i269, doi:10.1093/bioinformatics/btad233 (2023).

43 Lemane, T. *et al.* Indexing and real-time user-friendly queries in terabyte-sized complex genomic datasets with kmindex and ORA. *Nat Comput Sci* **4**, 104-109, doi:10.1038/s43588-024-00596-6 (2024).

44 Shiryev, S. A. & Agarwala, R. Indexing and searching petabase-scale nucleotide resources. *Nat Methods* **21**, 994-1002, doi:10.1038/s41592-024-02280-z (2024).

45 Fan, J., Khan, J., Singh, N. P., Pibiri, G. E. & Patro, R. Fulgor: a fast and compact k-mer index for large-scale matching and color queries. *Algorithms for Molecular Biology* **19**, 3, doi:10.1186/s13015-024-00251-9 (2024).

46 Karasikov, M. *et al.* Indexing All Life's Known Biological Sequences. *bioRxiv*, 2020.2010.2001.322164, doi:10.1101/2020.10.01.322164 (2024).

47 Gotoh, O. An improved algorithm for matching biological sequences. *Journal of Molecular Biology* **162**, 705-708, doi:https://doi.org/10.1016/0022-2836(82)90398-9 (1982).

48 Altschul, S. F. & Erickson, B. W. Optimal sequence alignment using affine gap costs. *Bulletin of Mathematical Biology* **48**, 603-616, doi:https://doi.org/10.1016/S0092-8240(86)90010-8 (1986).

49 Myers, E. W. & Miller, W. Optimal alignments in linear space. *Bioinformatics* **4**, 11-17, doi:10.1093/bioinformatics/4.1.11 (1988).

50 Needleman, S. B. & Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* **48**, 443-453, doi:https://doi.org/10.1016/0022-

2836(70)90057-4 (1970).

51 Vintsyuk, T. K. Speech discrimination by dynamic programming. *Cybernetics* **4**, 52-57, doi:10.1007/BF01074755 (1968).

52 Dean, J. Challenges in building large-scale information retrieval systems: invited talk. *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, 1, doi:10.1145/1498759.1498761 (2009).

53 Edgar, R. Syncmers are more sensitive than minimizers for selecting conserved kmers in biological sequences. *PeerJ* **9**, e10805, doi:10.7717/peerj.10805 (2021).

54 Shen, W., Sipos, B. & Zhao, L. SeqKit2: A Swiss army knife for sequence and alignment processing. *Imeta* **3**, e191, doi:10.1002/imt2.191 (2024).

Supplementary materials

Table S1. LexicMap alignment output format details.

#	Column	Description
1	query	Query sequence ID
2	qlen	Query sequence length
3	hits	Number of subject genomes
4	sgenome	Subject genome ID
5	sseqid	Subject sequence ID
6	qcovGnm	Query coverage (percentage) per genome: $(aligned bases in the genome)/$qlen
7	hsp	Nth HSP in the genome
8	qcovHSP	Query coverage (percentage) per HSP: $(aligned bases in a HSP)/$qlen
9	alenHSP	Aligned length in the current HSP
10	pident	Percentage of identical matches in the current HSP
11	gaps	Gaps in the current HSP
12	qstart	Start of alignment in query sequence
13	qend	End of alignment in query sequence
14	sstart	Start of alignment in subject sequence
15	send	End of alignment in subject sequence
16	sstr	Subject strand
17	slen	Subject sequence length
18	cigar	CIGAR string of the alignment. (optional with -a/--all)
19	qseq	Aligned part of query sequence. (optional with -a/--all)
20	sseq	Aligned part of subject sequence. (optional with -a/--all)
21	align	Alignment text ("|" and " ") between qseq and sseq. (optional with -a/--all)

Table S2. Details of 10 bacterial genomes for the alignment rate comparison.

Assembly	TaxID	Species	Contigs	Genome size
GCF_000005845.2	562	*Escherichia coli*	1	4,641,652
GCF_000240185.1	573	*Klebsiella pneumoniae*	7	5,682,322
GCF_000013425.1	1280	*Staphylococcus aureus*	1	2,821,361
GCF_000006945.2	28901	*Salmonella enterica*	2	4,951,383
GCF_001457635.1	1313	*Streptococcus pneumoniae*	1	2,110,968
GCF_000195955.2	1773	*Mycobacterium tuberculosis*	1	4,411,532
GCF_000006765.1	287	*Pseudomonas aeruginosa*	1	6,264,404
GCF_008632635.1	470	*Acinetobacter baumannii*	2	3,980,230
GCF_018885085.1	1496	*Clostridioides difficile*	2	4,095,894
GCF_022869705.1	1351	*Enterococcus faecalis*	1	2,866,948

Table S3. Alignment performance of LexicMap on GenBank + RefSeq dataset. Sequence identifiers are in Methods. qcov: query coverage (the percentage of aligned query bases in an alignment).

Query	Query length	Tool	Genome hits	Genome hits (qcov > 50%)	Time	RAM
a rare gene	1,299 bp	LexicMap	37,164	37,082	36 s	4.1 GB
a 16S rRNA gene	1,542 bp	LexicMap	1,949,496	1,381,974	10 m 41 s	14.1 GB
a plasmid	52,830 bp	LexicMap	544,619	6,563	19 m 20 s	19.3 GB
1033 AMR genes	1 kb (median)	LexicMap	25,702,419	14,692,624	187 m 40 s	55.4 GB

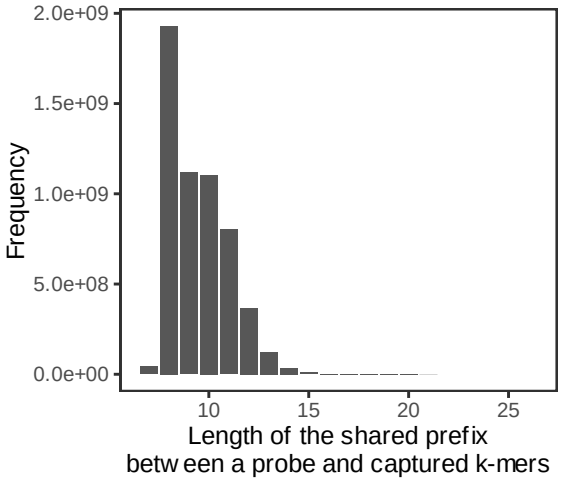

Fig.S1 | Histogram of shared prefix lengths between probes and the captured *k*-mers.

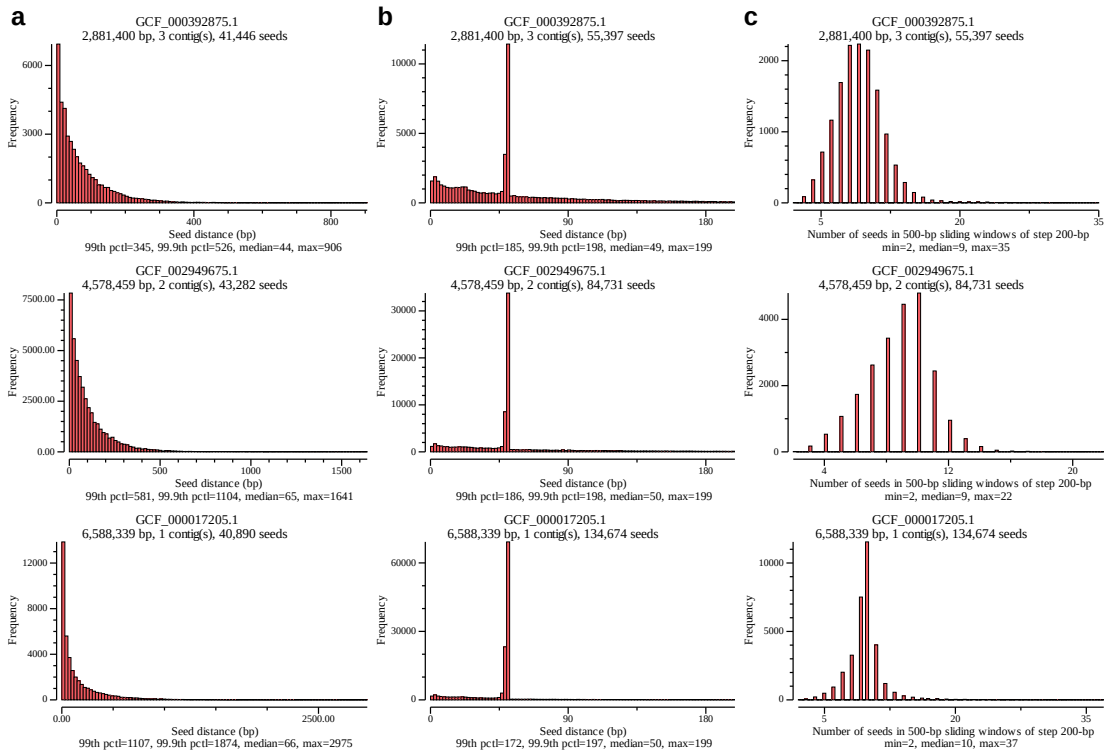

Fig.S2 | Histograms of seed distance and seed number in sliding windows. Three genomes of different genome sizes are used for illustration. a) Histograms of seed distance before filling deserts. b) Histograms of seed distance after filling deserts. c) Histograms of seed number in 500-bp sliding windows after filling deserts.

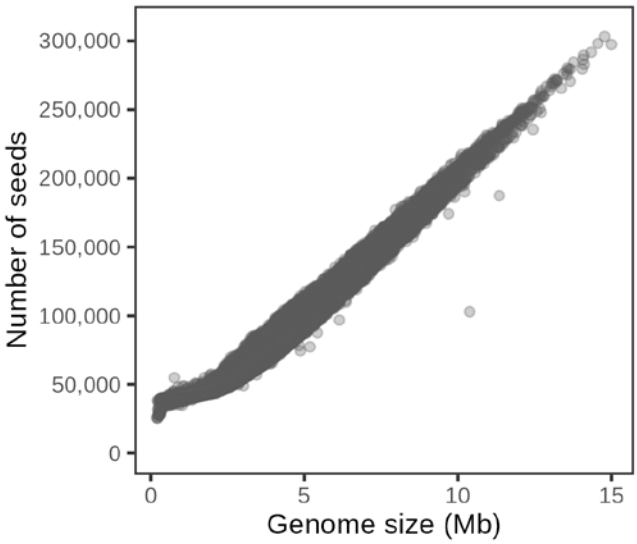

Fig.S3 | Number of seeds vs genome size. A LexicMap index is built from 85,205 GTDB r214 representative genomes with default parameters and “--save-seed-pos". Genome sizes are the total bases of assemblies.

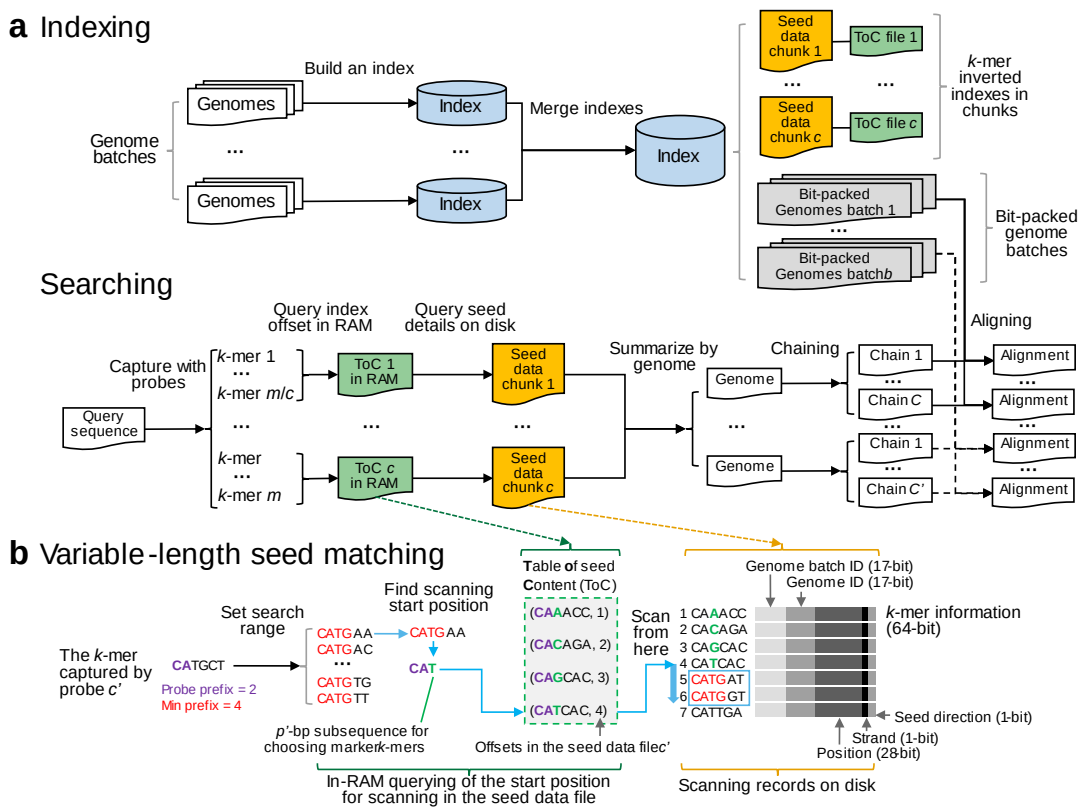

Fig.S4 | Architecture of LexicMap.

```
Query = GCF_006742205.1_r100
Length = 431

[Subject genome #1/1] = GCF_006742205.1 Staphylococcus epidermidis
Query coverage per genome = 92.575%

>NZ_AP019721.1
Length = 2422602

HSP #1
Query coverage per seq = 92.575%, Aligned length = 402, Identities = 98.507%, Gaps = 4
Query range = 33-431, Subject range = 1321677-1322077, Strand = Plus/Minus

Query  33       TAAAACGATTGCTAATGAGTCACGTATTTCATCTGGTTCGGTAACTATACCGTCTACTAT  92
                ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  1322077  TAAAACGATTGCTAATGAGTCACGTATTTCATCTGGTTCGGTAACTATACCGTCTACTAT  1322018

Query  93       GGACTCAGTGTAACCCTGTAATAAAGAGATTGGCGTACGTAATTCATGTG-TACATTTGC  151
                ||||||||||||||||||||||||||||||||||||||||||||||||| |||||||||
Sbjct  1322017  GGACTCAGTGTAACCCTGTAATAAAGAGATTGGCGTACGTAATTCATGTGATACATTTGC  1321958

Query  152      TATAAAATCTTTTTTCATTTGATCAAGATTATGTTCATTTGTCATATCACAGGATGACCA  211
                ||||||||||||||||||||||||||||||||||||||||||||||||| |||||||||
Sbjct  1321957  TATAAAATCTTTTTTCATTTGATCAAGATTATGTTCATTTGTCATATCAC-GGATGACCA  1321899

Query  212      TGACAATACCACTTCTACCATTTGTTTGAATTCTATCTATATAACTGGAGATAAATACAT  271
                ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  1321898  TGACAATACCACTTCTACCATTTGTTTGAATTCTATCTATATAACTGGAGATAAATACAT  1321839

Query  272      AGTACCTTGTATTAATTTCTAATTCTAA-TACTCATTCTGTTGTGATTCAAATGGTGCTT  330
                |||||||||||||||||||||||||||| |||||||||||||||||||||||||| |||||
Sbjct  1321838  AGTACCTTGTATTAATTTCTAATTCTAAATACTCATTCTGTTGTGATTCAAATGTTGCTT  1321779

Query  331      CAATTTGCTGTTCAATAGATTCTTTTGAAAAATCATCAATGTGACGCATAATATAATCAG  390
                |||||||||||||||||||||||||||||||||||||||||||||||||||| |||||
Sbjct  1321778  CAATTTGCTGTTCAATAGATTCTTTTGAAAAATCATCAATGTGACGCATAATATCATCAG  1321719

Query  391      CCATCTTGTT-GACAATATGATTTCACGTTGATTATTAATGC  431
                |||||||||| |||||||||||||||||||||||||||||||
Sbjct  1321718  CCATCTTGTTTGACAATATGATTTCACGTTGATTATTAATGC  1321677
```

Fig.S5 | Blast-style output format.