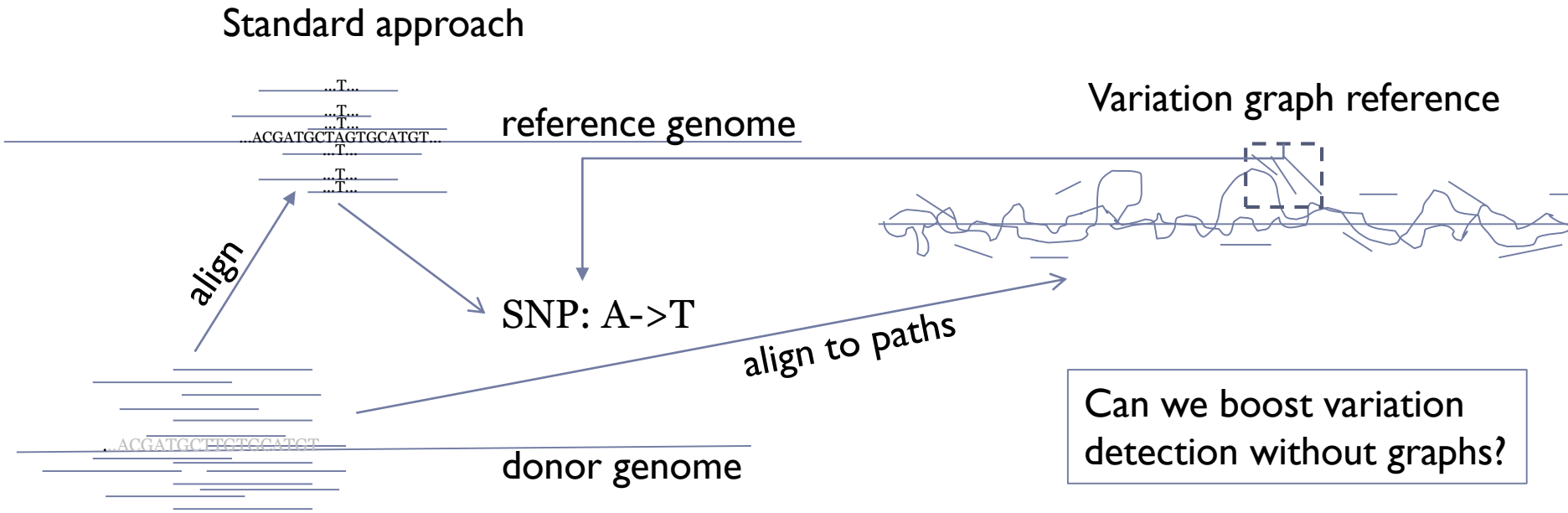


On enhancing variation detection through pan-genome indexing

Veli Mäkinen
HIIT, Dept. CS, University of Helsinki, Finland

Joint work with Daniel Valenzuela, Niko Välimäki, and Esa Pitkänen

Pan-genome data structures

- Index the extracted contexts around variations [Schneeberger et al. Genome Biology 2009; Huang et al. ISMB 2013; Yang et al. ICDE 2013]
- *Index multiple references* [Mäkinen et al. RECOMB 2009; Do et al. LNCS 7285, 2012; Ferrada et al. *Phil. Trans. R. Soc. A, 2013;* Wandelt et al. VLDB 2013]
- Index the automaton recognizing all variation combinations [Sirén et al. WABI 2011, TCBB 2014]

GenomeMapper

VG, HISAT2


Context extraction
ACGATCTCGTATAGCTACTAGTGCTGCTGTCTCGTC
ACGATCCCGTATAGCTAGTAGTGCTGCAGTCTCGTC
ACGATCTCGTATAGCTACTAGTGCTGCTGTCTCGTC #..#TCCCG#..#TAGTA#....
Multiple references
ACGATCTCGTATAGCTACTAGTGCTGCTGTCTCGTC
ACGATCCCGTATAGCTAGTAGTGCTGCAGTCTCGTC
Graph
ACGATCTCGTATAGCTACTAGTGCTGCTGTCTCGTC
C
G
A

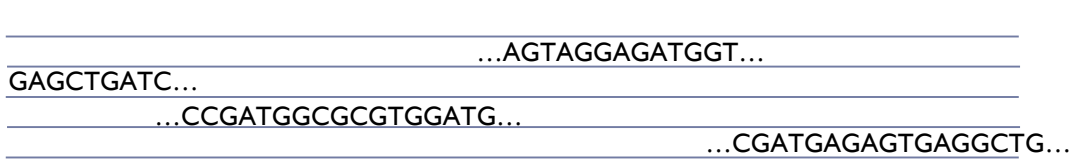
Graphs and extracted contexts in real use…

- … but indexes for multiple references have remained as a topic of algorithm engineering for time and space optimization for exact / approximate pattern matching.
- We show that one deploy them in a real genome analysis pipeline, offering scalability, accuracy, and compatibility beyond(?) other approaches.
- *Appendum*: Some motivating facts about why to study alternatives to graphs as pan-genome representation.

Overview of our proposal

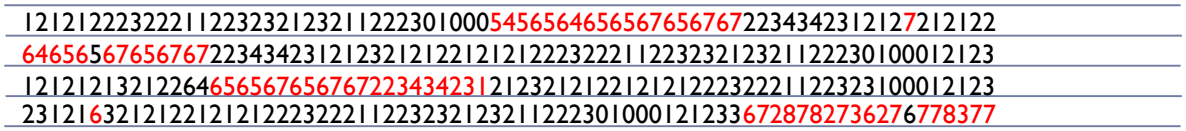
- Input 1: Multiple alignment of reference genomes
  - Or standard reference + set of vcf files
- Input 2: Reads from a donor
- Output 1: Predicted donor as an *ad hoc* reference
- Output 2: Metadata mapping the coordinate system of *ad hoc* reference to any of the original references.
- Running a best practices variation calling pipeline on the *ad hoc* reference produces a set of novel variants that can be projected back to the standard reference.
- We provide the first implementation of a space-efficient multiple references pan-genome index that supports plugging in any standard read aligner.

Predicting the donor



Multiple alignment of references

Reads from donor



heaviest path -> *ad hoc* reference

GAGCTGATGCGGATGGCGCGTGGATGAGTAGGAGACGGTCGATGAGACTGAGGCTG

reads, *ad hoc* reference, GATK

predicted donor

Coverage matrix resulting from aligning reads to multiple references pan-genome index

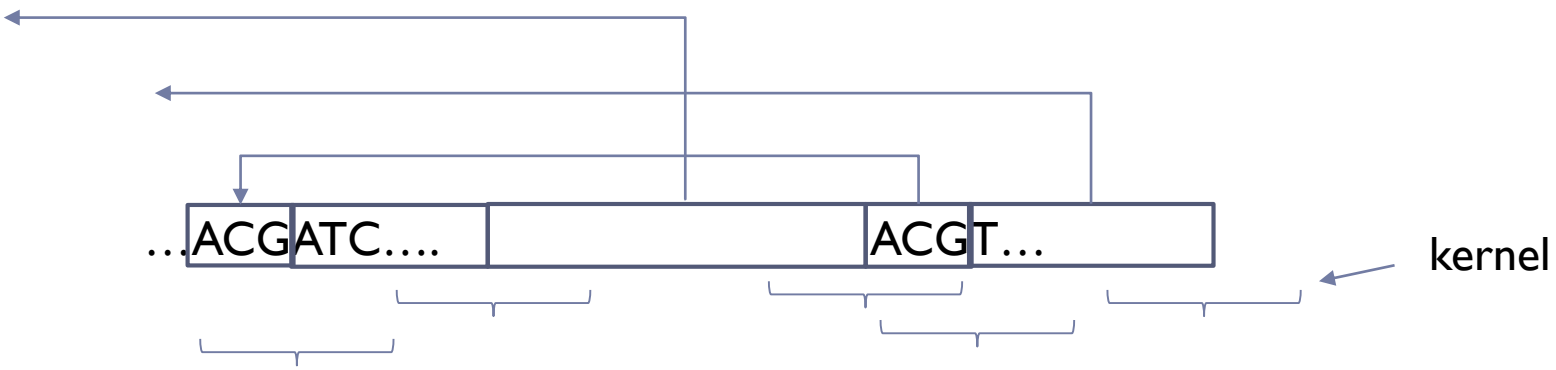
Components of our proposal

- Metadata (rank/select bitvectors) to map multiple alignent into a collection of sequences and back
- Space-efficient read aligner on the collection of similar sequences
  - + simple scripts to produce the coverage matrix
- Heaviest path with limited jumps dynamic programming on the coverage matrix
  - Predicted donor sequence as output
- Best practices variant calling on the predicted donor
  - Output a vcf file of novel variants (not yet seen in pan-genome)
- Normalizer routine to convert vcf file to the coordinate system of the standard reference using the metadata

Pan-genome indexing on multiple references

- Ferrada et al. *Phil. Trans. R. Soc. A, 2013;* Wandelt et al. VLDB 2013; Valenzuela SEA 2016.
- LZ77-based index on a concatenation of references
- Extract regions close to phrase boundaries, index them, and align using any standard read aligner
  - All primary occurrences
- Secondary occurrences found using a range query data structure
- Small space, fast query time… but limited to fixed length patterns

Pan-genome indexing on multiple references



OBSERVATION: Consider kernel string containing length M context before and after each phrase boundary. Pattern of length M has an occurrence in the kernel string if and only if it has an occurrence in the original string.

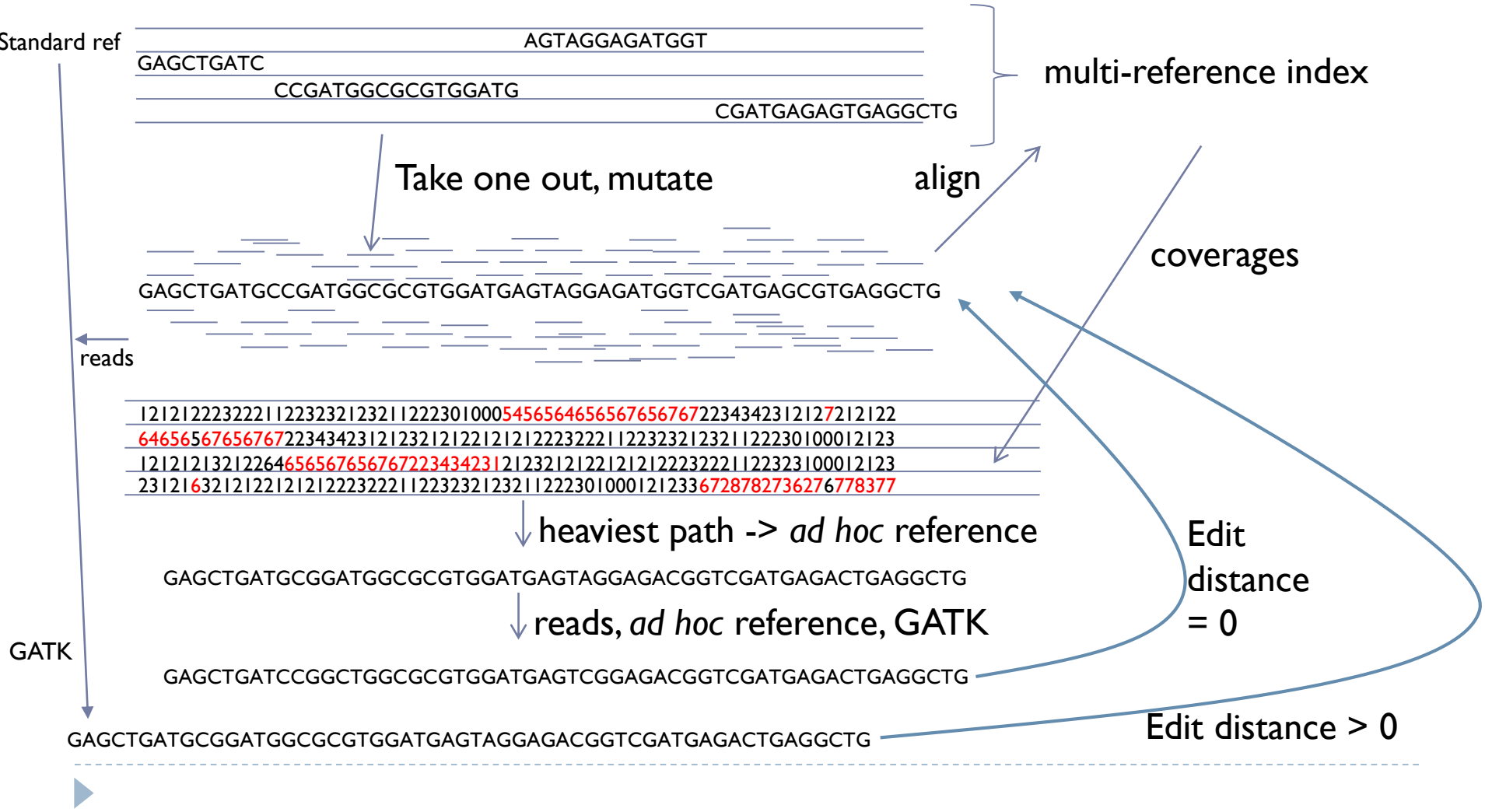
Pan-genome indexing on multiple references

- Kernel string can be indexed using any read aligned.
  - E.g. using standard wavelet tree based BWT index a primary exact occurrence can be found in $O(M \log \sigma + \log n)$ time.
  - $O(\log n)$ time per each secondary occurrence, when using wavelet tree to support range search.
  - Space is $O(z\,(M \log \sigma + \log n))$ bits, where **z** is the number of LZ77 phrases of a concatenation of the multiple references of length n, and σ is the alphabet size.
  - **z** ~ length of first reference sequence + total size of the variation included in other references compared to the first

CHIC Aligner

- Based on CHICO
  - Daniel Valenzuela: CHICO: A Compressed Hybrid Index for Repetitive Collections. SEA 2016: 326-338
  - CHICO supports only exact pattern matching
  - Supports also relative LZ: use one reference as the source of all phrases
- CHIC Aligner modifies CHICO to support plugging in any read aligner
  - We have experimented with BWA and Bowtie2

Experiment setup



Accuracy results

- Variation-rich regions, 186 Finnish genotypes, total 2,2 MB
- Averaging over random choice of donor
  - Average distance from standard reference to donor 95193,3

Distance to donor	1 ref	20 ref	50 ref	100 ref
GATK	74695,9			
GCSA+GATK		3230,4	3336,8	2706,9
CHICA+GATK		1346,7	1117,3	**1096,20**

CHICA		GATK	20 ref	50 ref	100 ref
SNP Precision		0.992161	0.997355	0.996913	0.996348
SNP Recall		0.904897	0.996721	0.997528	0.997554
Indel Precision		0.364853	0.994608	0.994906	0.994804
Indel Recall		0.0624981	0.961927	0.973369	0.982674

Scalability results

- BWA_x, $CHICA_x$, $BWBBLE_x$ index x different copies of the chromosome 21.

Aligner Index	Input size	Indexing time	Size	Aligning time	Mapped reads
BWA_1	48.1MB	48s	81MB	2m6s	143735
BWA_{193}	9.2GB	290m21s	16GB	31m17s	144050
$CHICA_{193}$	9.2GB	5m41s	118MB	2m18s	112777
$CHICA_{2001}$	96.3GB	11m33s	119MB	2m35s	**144694**
$BWBBLE_{193}$	-	1m8s	250MB	37m13s	28296
$BWBBLE_{2001}$	-	5m15s	250MB	39m37s	28296

?

Discussion

- Significant increase in accuracy with almost no slowdown or space explosion
- Only the index construction needs considerable resources for1000 complete human genomes
  - We are currently constructing ready-built index for download
- The tool uses standard file formats
  - Easy to plug into analysis pipelines
- Avoiding the burden of thinking about graphs
  - For really variation-rich cases graphs are still the only way

- CHICA: https://www.cs.helsinki.fi/en/gsa/chica/
- Preprocessor: https://www.cs.helsinki.fi/en/gsa/panvc/
- Manuscript in bioRxiv with the same title.

Thanks

- Future Perspectives in Computational Pan-Genomics, Leiden, Lorentz Center, June 2015
- Dagstuhl seminar on Next Generation Sequencing - Algorithms, and Software For Biomedical Applications, last week

- Academy of Finland grant for CoE in Cancer Genetics Research.
- Co-authors:



FAQ

- Is it possible to make this pan-genome representation dynamic?
  - Yes, in principle. Using relative LZ, one can quite easily remove and add reference genomes: dynamic bitvectors, BWT merging.
BWA_{193}	9.2GB	290m21s	16GB	31m17s	144050
$CHICA_{193}$	9.2GB	5m41s	118MB	2m18s	112777

  - Using BWA we could not add separators to kernel string… some alignments go over extracted contexts.
- Coverage matrix looks large…
  - Yes, chromosome size * #references * log n bits.
  - We can construct it using external sorting, online processing of heaviest path, storing samples of DP -> Version 1.1.

FAQ

- What about large variations?
  - Everything except chromosome fusions / fissions can be added to the multiple alignment.
  - The approach is almost agnostic to the exact representation of the variation. As long as the concatenation of sequences compresses well using LZ, and the multiple alignment is not ridiculously bad, the approach is practical.
  - We provide a simple script to convert vcf files into a multiple alignment: handling of large (overlapping) variations could probably be improved.
- How similar should the references be?
  - Try it out on your data!

FAQ

- You report only novel variants. How can I get the variants found in the pan-genome?
  - Our metadata encodes the alignment of *ad hoc* reference and standard reference. We will provide a script that converts this to a vcf file (under construction).
- Could you predict diploid genome by taking two heavy paths that best explain the coverage matrix
  - Possibly, we are working on this…
  - … this is related to transcript prediction and we have some recent work on this.

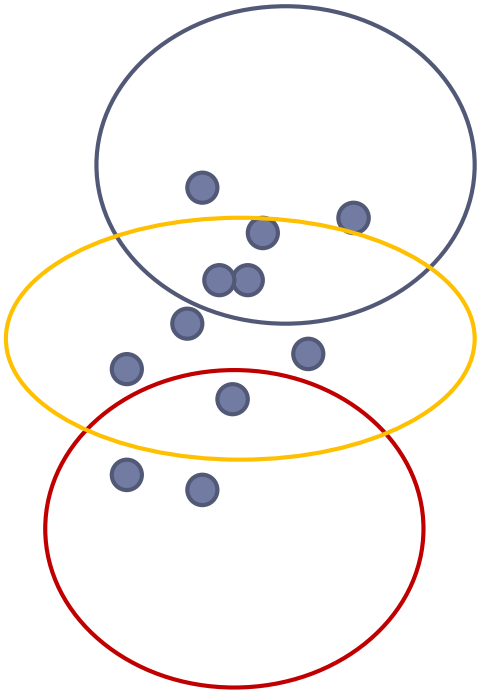
Addendum: Complexity of path queries on graphs

- Let G=(V,E) be a labeled DAG of |V| nodes and |E| edges. Nodes are labeled from alphabet of size σ. Let P be the path query.
- Exact and approximate matching basically as difficult:
  - O(|P||E|) time, without bitparallel speed-ups.
- Indexing graphs for path queries is *set intersection indexing* hard (see last slide).
- On a specific distribution of random labeled DAGs, there is a *Generalized Burrows-Wheeler transform* –based index (GBWT aka GCSA) with expected size O(|E| log σ) bits:
  - Worst case exponential on |E|.
  - Supports path query in O(|P| log σ) time.
    - Extends to approximate search with backtracking heuristics.
  - See WABI 2011/TCBB 2014 by Sirén et al.

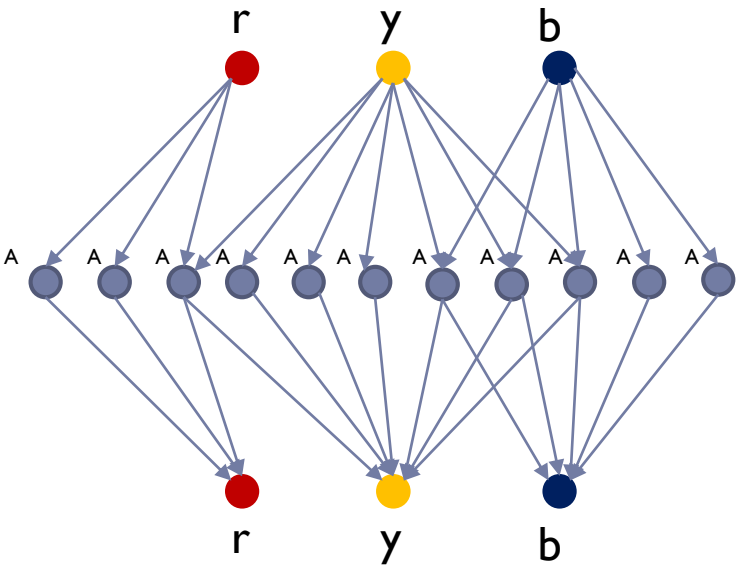
Addendum: Complexity of path queries on graphs

- Should we do theory first?
- Classical sequence analysis methods have been subject of serious theoretical study in stringology / combinatorial pattern matching
  - Lot's of back-and-forth influence between practical tool development and theory: e.g. filtering algorithms
- Perhaps the same could happen with path queries on graphs.

Appendum: Hardness of indexing graphs



If you can solve a path query (e.g. rAb) in linear time in path length, then you can find out if sets (e.g. r and b) intersect in constant time.



Only known constant time solution to set intersection query is to store quadratic size table with all the answers.

Personal communication with Philip Bille