2 - Approximate Matching

Sunday, 10 October 2021 17:06

BOYER-MOORE VERY POPULAR ALGORITHM

USED FOR QUERIES, WEB & DB SEARCH

MAIN IDEA: CAN WE LEARN FROM PAST COMPARISONS ?

WE ARE GOING TO LEARN FROM OUR
CHARACTER-COMPARISON AS MUCH AS WE
CAN TO SKIP ALIGNMENTS THAT WILL NOT
RESULT IN A MATCH

ALIGNMENTS IN LEFT-TO-RIGHT
CHAR COMPARISON IN OPPOSITE ORDER (RIGHT-TO-LEFT)

IF WE SHIFT P BY x WE DON'T CHECK $x-1$ ALIGNMENTS

THE BAD CHARACTER RULE

UPON MISMATCH SKIP ALIGNMENTS UNTIL:

a) MISMATCH BECOMES A MATCH

b) P MOVES PAST MISMATCHED CHARACTER

GOOD SUFFIX RULE

t = SUBSTRING MATCHED BY INNER LOOP, SKIP UNTIL:

a) THERE ARE NO MISMATCHES BETWEEN P AND t

b) P MOVES PAST t

BOYER-MOORE

USE BAD-CHAR OR GOOD-SUFFIX
WHICHEVER SKIPS MORE

PREPROCESSING USING TABLES

REAL GENOMES ARE REALLY DIFFERENT FROM RANDOM ONES

REPETITIVE ELEMENTS ?

EVOLUTIONARY PROCESS INTRODUCES PATTERNS

HUMAN GENOME IS EXTREMELY REPETITIVE

GENOMES GETS INVADED BY TRANSPOSONS

45% OF HUMAN GENOME IS COVERED BY
TRANSPOSABLE ELEMENTS

Alu, L1, SVA & OTHER

WHY SHOULD WE CARE ?

THEY CREATE AMBIGUITY

LIKE A PUZZLE WHERE HALF OF THE PIECES
ARE FEATURELESS BLUE SKY

WHERE DOES THE PIECE COME FROM ?

REPETITIVE ELEMENTS CREATE PROBLEMS
FOR OUR ALGORITHMS

BOYER-MOORE PRE-PROCESS PATTERN P IN ORDER
TO BUILD LOOK-UP TABLES THAT HELP TO USE
THE GOOD-SUFFIX & BAD-CHARACTER RULES

PREPROCESSING

IF WE WANT TO MATCH THE SAME PATTERN P AGAINST
DIFFERENT TEXTS T, I CAN REUSE MY PREPROCESSING
FOR THE PATTERN

THE COST OF PREPROCESSING CAN BE AMORTIZED
OVER MANY PROBLEMS

OFFLINE VS OFFLINE

PREPROCESSING THE TEXT T (ONLINE)

BOTH NAÏVE & BOYER-MOORE ARE 'ONLINE' ALGORITHMS
WHEREAS WEB SEARCH ENGINE IS OFFLINE

WHEN WE SEARCH ON THE WEB, THE ENGINE
ALREADY PREPROCESSED LOADS OF INFO.
THIS MAKES QUERIES FASTER

READ ALIGNMENTS ? WE CAN USE AN OFFLINE ONE
IT WOULD BE HELPFUL FOR A REFERENCE GENOMES

PREPROCESSING TO MACH MANY READS QUICKLY

WHY REF-BASE ASSEMBLY IS OFFLINE ?

REFERENCE GENOMES DOES NOT CHANGE
BETWEEN EXPERIMENTS, IT MAY BE
UPDATED ONCE A YEAR

- **INDEXING** (LIKE IN BOOKS)

LIST OF KEY-TERM & PLACES ON THE
BOOK WHERE IT IS MENTIONED

- **GROUPING** (LIKE IN SUPERMARKETS)

HOW CAN WE ORGANIZE QUERIES IN ORDER
TO MAKE THEM EASY TO RETRIEVE ?

HOW CAN WE INDEX DNA ?

TAKE EVERY SUBSTRING OF TEXT T
OF A CERTAIN LENGTH, TREAT EACH OF
THOSE SUBSTRING AS A WORD

K-MER SUBSTRING OF LENGTH K

WE CAN SOLVE THE **EXACT-MATCHING PROBLEM**

THESE K CHAR OF P (PATTERN) MATCH K CHAR
OF T (TEXT) AT INDEX i

FINDING THE INDEX-HITS

MULTI-MAP

MAPS EACH K-MER TO LIST OF ALL
OCCURRENCES IN TEXT

(K-MER → OFFSET IN GENOME)

BINARY SEARCH $O(\log n)$

FASTER QUERY SINCE MAP ORDERED ALPHABETICALLY

HASH-TABLE

USEFUL TO IMPLEMENT MULTI-MAPS
(OFTEN AS LINKED LISTS)

PYTHON'S DICTIONARY IS AN IMPLEMENTATION
OF A HASH TABLE

SUBSEQUENCE ≠ SUBSTRING (NOT CONTIGOUS)

USING SUBSEQUENCE TENDS TO INCREASE
THE SPECIFICITY OF THE INDEX

SUBSTRINGS ARE SUBSEQUENCES BUT NOT THE
OTHER WAY AROUND

GENOME INDEXES USED IN RESEARCH

INDEXING IS AT THE CORE IN A LOT OF METHODS
TO ANALYZE READING-DATA

HOW CAN WE FIT GENOMES INTO A COMPACT INDEX ?

VARIATIONS (DIFFERENT IMPLEMENTATIONS)

EXTRACT ONLY SOME OF THE K-MER
(BOTH ODD & EVEN OFFECTS)

SUBSEQUENCE-INDEX
(MORE SPECIFIC)

SUFFIX-INDEX (TREE, ARRAY, FM)

FM-INDEX (BASED ON BURROWS-WHEELER)
ORIGINALLY DEVELOPED FOR COMPRESSION

BIO SEQUENCES ANALYSIS

APPROXIMATE MATCHING PROBLEM

ACCOUNTING FOR:

- SEQUENCING ERRORS
- NATURAL VARIANCE

DISTANCE

HOW 2 STRINGS DIFFER FROM EACH OTHER

- **HAMMING DISTANCE**

MINIMUM # OF <u>SUBSTITUTIONS</u> TO TURN
ONE STRING INTO ANOTHER

$|x| = |y|$ (SAME LENGTH)

- **EDIT DISTANCE**

MINIMUM # OF <u>EDITS</u> TO TURN
ONE STRING INTO ANOTHER

- SUBSTITUTION
- INSERTION
- DELETION

$|x|, |y|$ ANY LENGTH

PIGEONHOLE PRINCIPLE

IF WE HAVE 10 HOLES BUT 11 PIGEONS
ONE HOLE WILL HAVE 2 PIGEON.

THERE MUST EXIST SUCH A HOLE

THE EDIT-DISTANCE PROBLEM

DYNAMIC PROGRAMMING

METHOD TO SOLVE BIO-SEQ ANALYSIS PROBLEMS
NEW FAMILY OF TOOLS

H-DIST EASY
E-DIST HARD

DISTANCE RELATIONSHIP

$E(x,y) \leq H(x,y)$

INSERTIONS & DELETION ARE VERY POWERFUL,
WE CAN GET FROM x TO y IN FEWER CHANGES

$E(x,y) \geq \big| |x| - |y| \big|$

WE AVOID REPETITIONS IN DYNAMIC PROGRAMMING