VARIATIONS OF BOYER-MOORE STRINGMATCHING ALGORITHM: A COMPARATIVE ANALYSIS

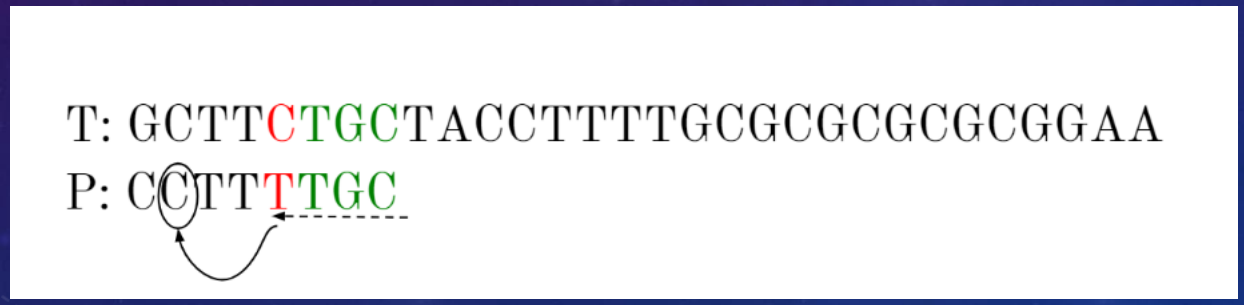
GENOME INFORMATICS(13M111GI)
UNIVERSITY OF BELGRADE , SCHOOL OF ELECTRICAL ENGINEERING
Lazar Beloica 17/3214
Milan Zafirović 17/3207

AGENDA

- About Boyer Moore algorithm
- Good Suffix heuristic
- Bad Character heuristic
- Boyer-Moore-Horspool
- Improved BoyerMoore-Horspool-Sundays
- Project Structure
- Performance analysis

ABOUT BOYER MOORE ALGORITHM

- an efficient string searching algorithm
- developed by Robert S. Boyer and J Strother Moore in 1977
- compares pattern to the text, starting from the rightmost character in pattern



ABOUT BOYER MOORE ALGORITHM - PSEUDOCODE

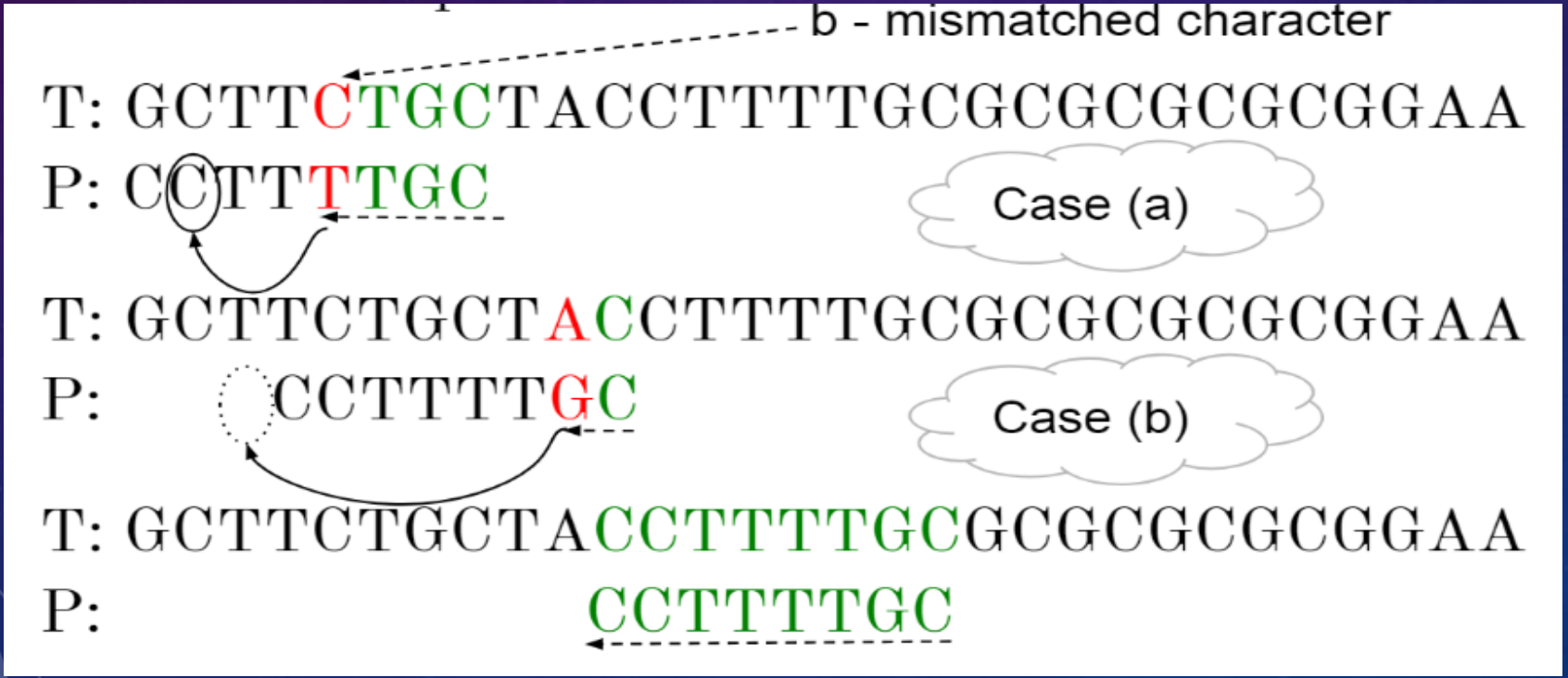
- preprocessing(pattern)
- align pattern to the text beginning
- while(not text end):
  - match pattern and text from right to left
  - if a mismatch occurs
    - shift pattern to the right (distance of shift depends on heuristic)
  - else(pattern is fully matched):
    - print the occurrence and shift pattern to the right (distance of shift depends on heuristic)

GOOD SUFFIX HEURISTIC

- find t + mismatched letter in the rest of the pattern
  - t = the suffix matched so far

BAD CHARACTER HEURISTIC

- look for the mismatched character in the rest of the pattern



BOYER-MOORE-HORSPOOL

- BMH algorithm only uses the bad characters shift
- no matter the location of mismatching, the distance of shift to right is determined by the character in the text string which is aligned to the last one of pattern string

	S	T	R	I	N	G	M	A	T	C	H	I	N	G	I	S	T	O	F	I	N	D	T	H	E	P	A	T	T	E	R	N
1	P	A	T	T	E	R	N																										
2								P	A	T	T	E	R	N																			
3															P	A	T	T	E	R	N											
4																						P	A	T	T	E	R	N				
5																									P	A	T	T	E	R	N	
6																										P	A	T	T	E	R	N

BOYER-MOORE-HORSPOOL

- advantages
  - the concept of Good-suffix is removed, so easy to implement
  - in case of mismatch ,the shift value is determined by the bad char value of last character instead of character that caused mismatch so more jump is archived using bad char than in BM
- disadvantages
  - the removal of Good-Suffix sometime may not give shift as much as in BM with Good-Suffix

IMPROVED BOYERMOORE-HORSPOOL-SUNDAYS

- idea: on mismatch watch the next two letters in the text
- preprocessing:
  - similar to what is done for the badcharacter heuristic
  - another table is needed to keep track how many times each letter occures in the pattern

	S	T	R	I	N	G	M	A	T	C	H	I	N	G	I	S	T	O	F	I	N	D	T	H	E	P	A	T	T	E	R	N
1	P	A	T	T	E	R	N																									
2							P	A	T	T	E	R	N																			
3																P	A	T	T	E	R	N										
4																									P	A	T	T	E	R	N	
5																										P	A	T	T	E	R	N

IMPROVED BOYERMOORE-HORSPOOL-SUNDAYS

- mismatch on the first compared character or on complete match:
  - calculate move for the first and second letter
    - note: if the first letter is not in the text move len(pattern) + 1
    - note: if the second letter is not in the text move len(pattern) + 2
  - if the calculated move for the first letter is 1, do it
  - else move for the max value of calculated moves for first and second letter

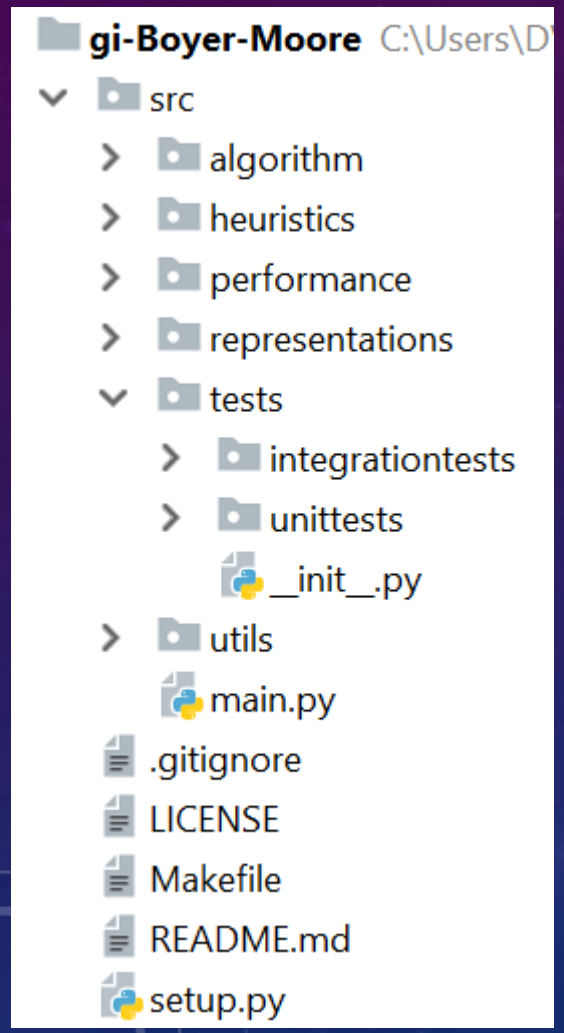
IMPROVED BOYERMOORE-HORSPOOL-SUNDAYS

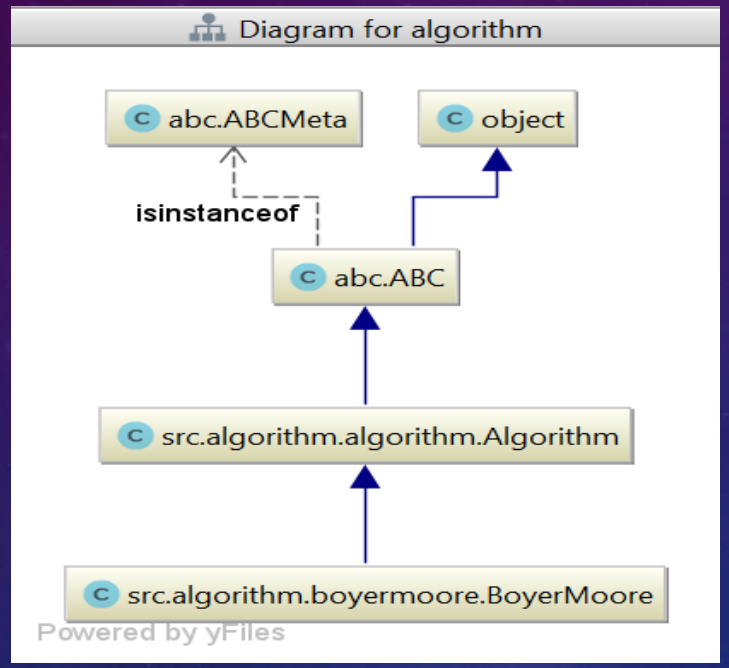
- mismatch on other than first compared character:
  - calculate move for the firs and second letter
    - note: if the first letter is not in the text move len(pattern) + 1
    - note: if the second letter is not in the text move len(pattern) + 2
  - if the calculated move for the first letter is 1 and numbber if the appearances of the letter in pattern we didn't match is 1, move for max value of calculated moves for the second letter and len(pattern) + 1
  - else move for the max value of calculated moves for the first and second letter

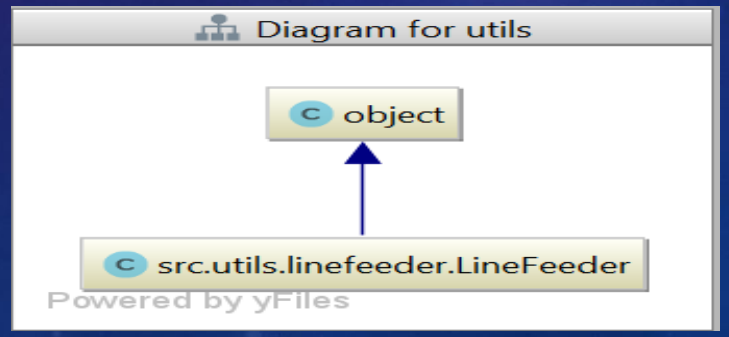
IMPROVED BOYERMOORE-HORSPOOL-SUNDAYS

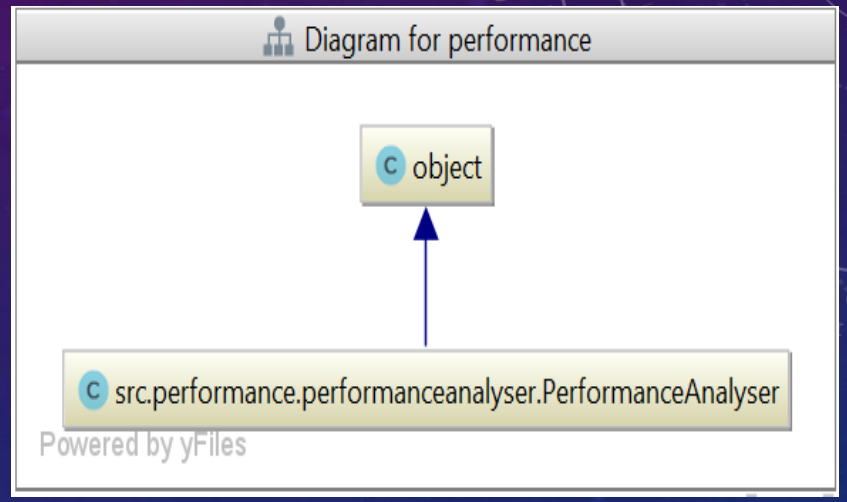
- advantages
  - maximum shift that can be achieved using this algorithm is pattern length + 2
- disadvantages
  - calculation of shift using Next-to-Last and Next-toNext-to-Last character increase searching over head and for that preprocessing of Num[ ] is done which increases preprocessing overhead

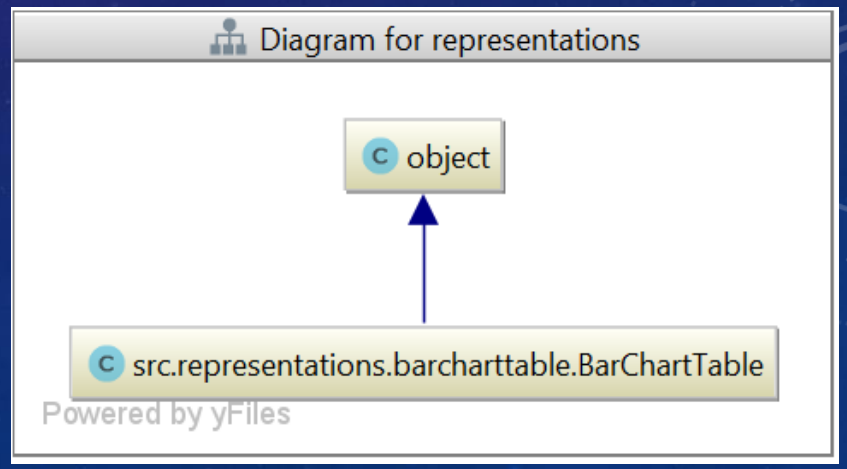
PROJECT STRUCTURE



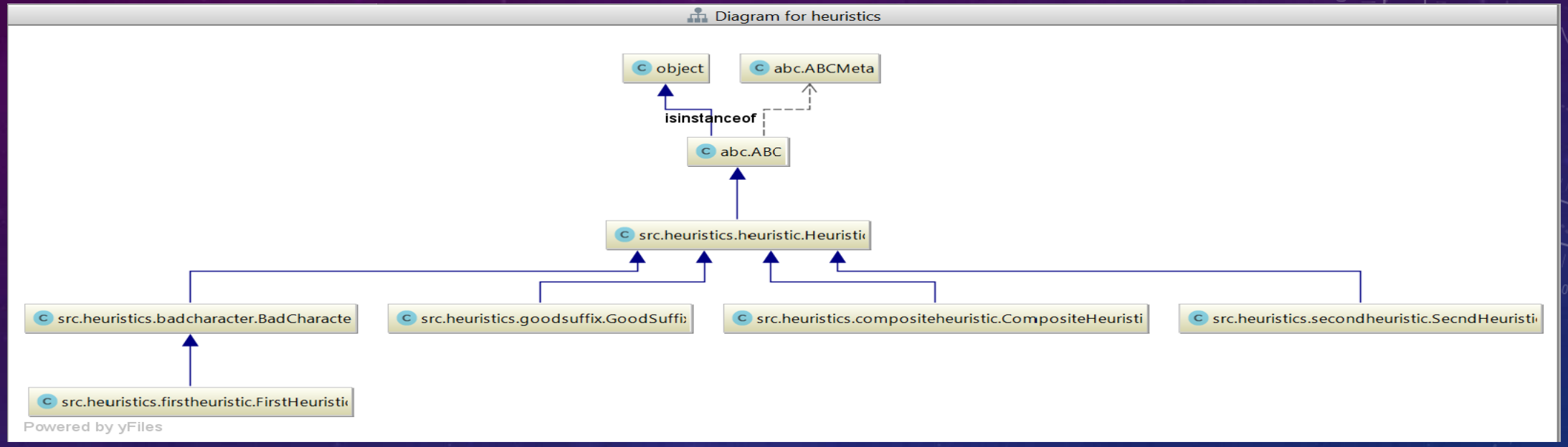


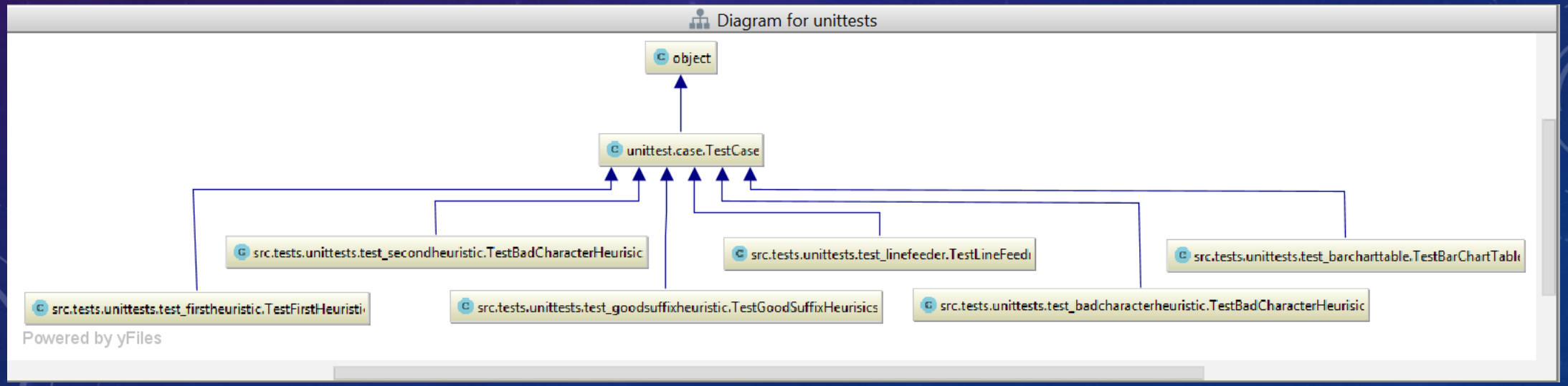




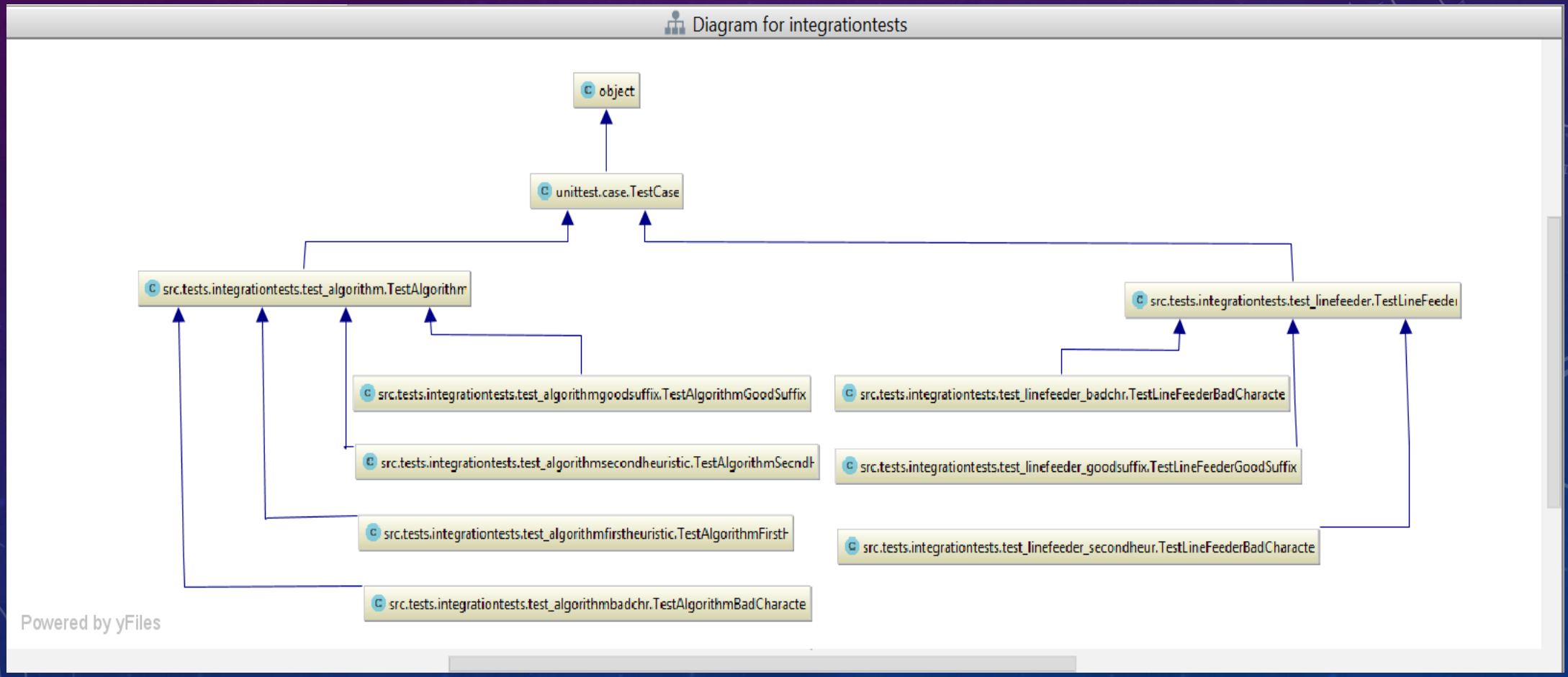


PROJECT STRUCTURE





PROJECT STRUCTURE



PERFORMANCE ANALYSIS

i	Pattern	File
1	ATGATG	Canis_lupus Chromosome 1
2	CTCTCTA	Canis_lupus Chromosome 1
3	TCACTACTCTCA	Canis_lupus Chromosome 1
4	ATGATG	Phoenix_dactylifera Genome
5	CTCTCTA	Phoenix_dactylifera Genome
6	TCACTACTCTCA	Phoenix_dactylifera Genome
7	ATGATG	Camelina_sativa Chromosome 1
8	CTCTCTA	Camelina_sativa Chromosome 1
9	TCACTACTCTCA	Camelina_sativa Chromosome 1

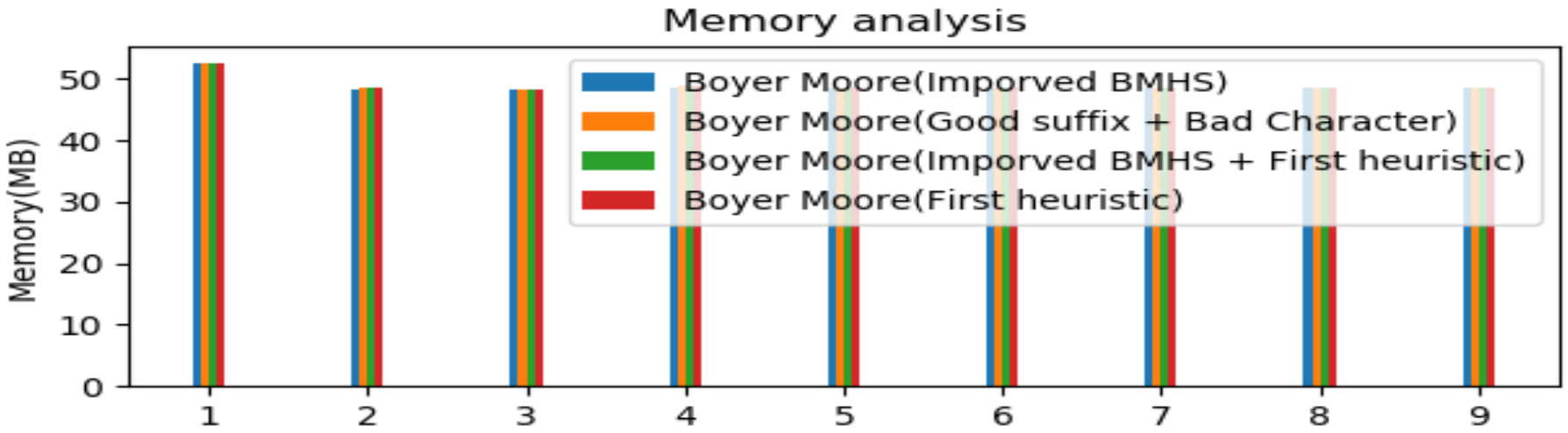
flag	meaning
-h	Heuristic list
-c	Pattern file list

abbreviation	heuristic
bc, BC	BadCharacter
gs, GS	GoodSuffix
fh, FH	FirstHeuristic (BMH)
sh, SH	SecondHeuristic (IBMHS)

Example:

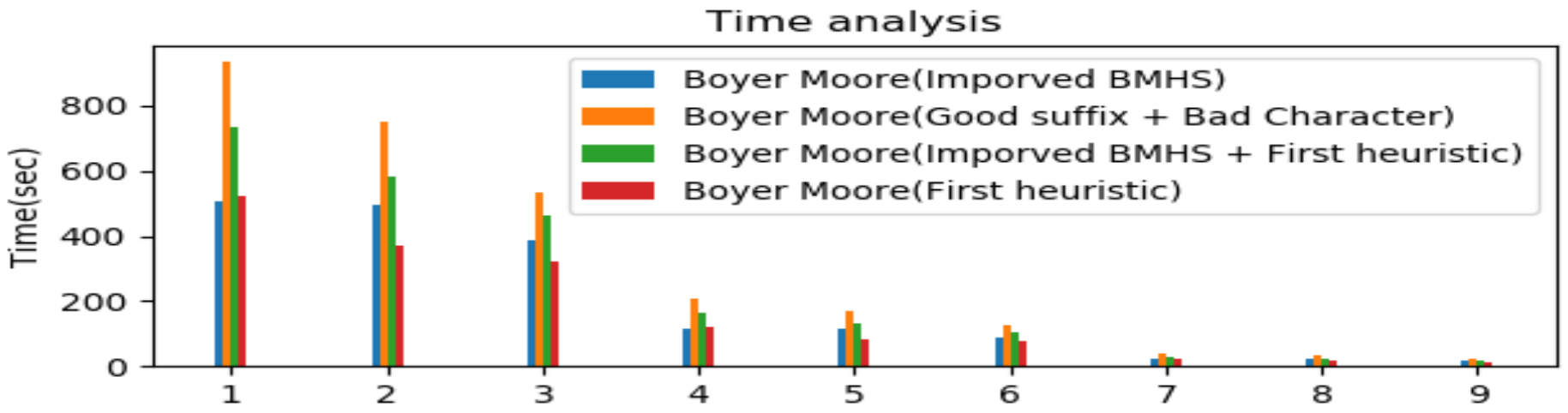
python main.py -h fh+sh fh sh bc+gs -c C:\Canis_lupus_chr_1 ATGATG

PERFORMANCE ANALYSIS



	1	2	3	4	5	6	7	8	9
	52.55	48.18	48.37	48.66	48.64	48.64	48.63	48.64	48.63
	52.58	48.68	48.37	48.77	48.63	48.63	48.63	48.64	48.63
	52.6	48.68	48.37	48.77	48.64	48.63	48.64	48.64	48.63
	52.56	48.7	48.37	48.77	48.64	48.63	48.64	48.64	48.63

PERFORMANCE ANALYSIS



	1	2	3	4	5	6	7	8	9
	506.39	496.48	385.02	113.49	112.58	87.19	21.33	21.59	16.71
	935.36	750.15	530.68	208.04	168.63	125.36	39.77	31.68	22.35
	732.02	584.2	461.6	161.26	130.09	104.39	30.2	24.3	19.79
	521.6	372.21	319.15	118.5	83.01	76.18	22.03	15.38	13.3

FOR MORE INFORMATION

- Project:
  https://github.com/lazarbeloica/gi-Boyer-Moore
- Email:
  zafirovicmilan2@gmail.com
  lazarbeloica@gmail.com