BLAKE THOMPSON

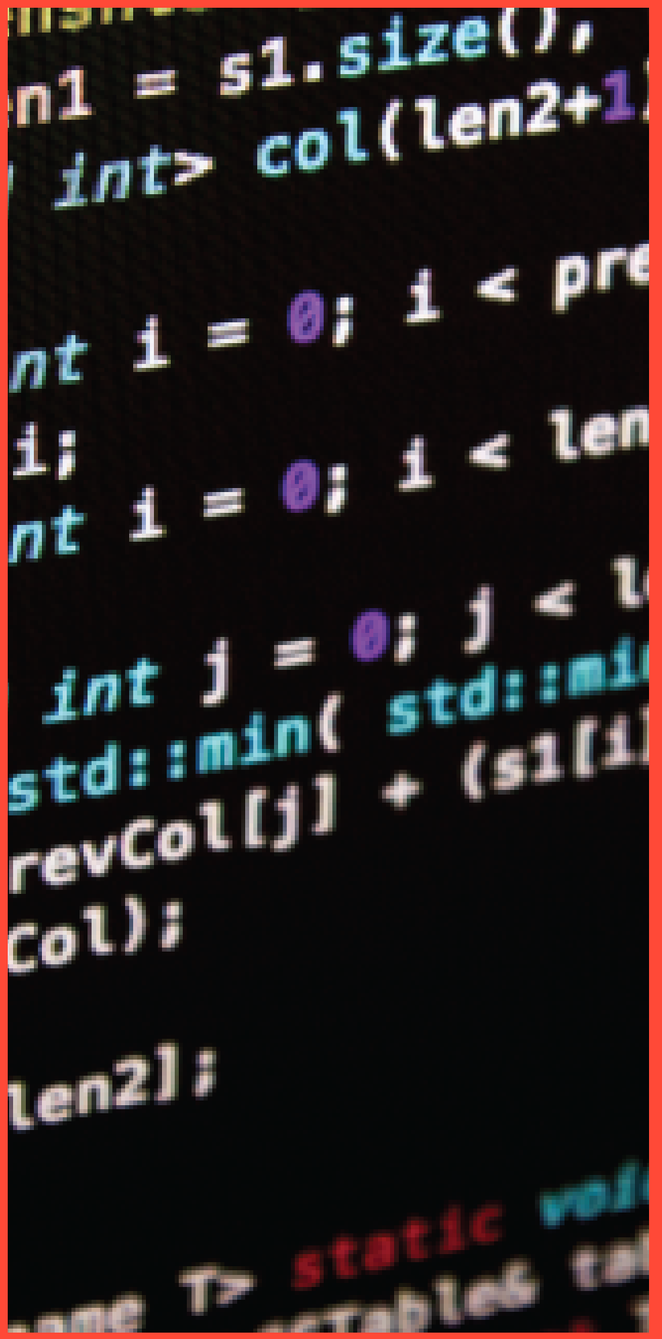
MAHATHIR MOSTAFA

Key	List							
3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8

SORTING COMPETITION GROUP 2

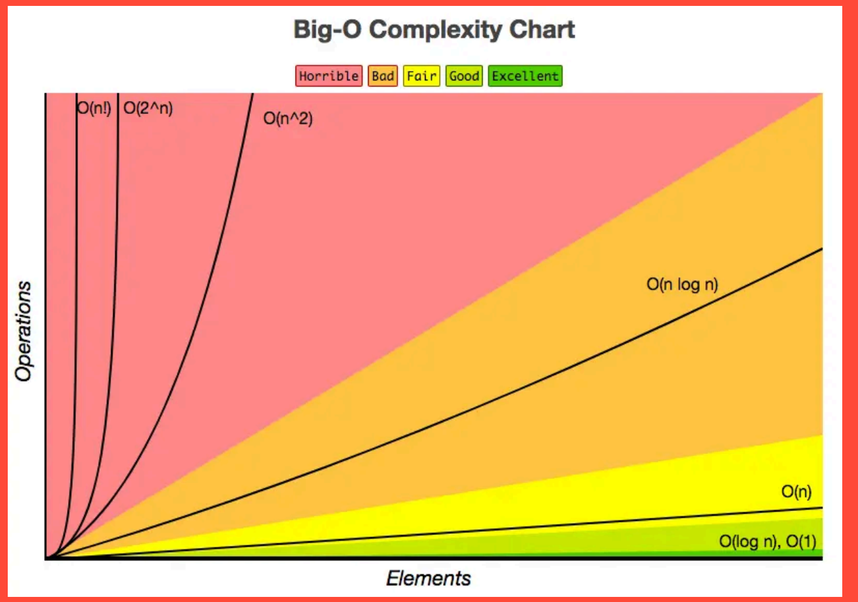
PRESENTATION FOR THE SORTING COMPETION LAB. GROUP 2 MAHATHIR AND BLAKE

SCORE AND TIME(01)
CORRECTNESS ISSUES(02)
DESCRIPTION OF THE ALGORITHM(03)
DATA STORAGE (04)
FURTHER DETAILS (05)



MAHATHIR BLAKE

SCORE AND TIME



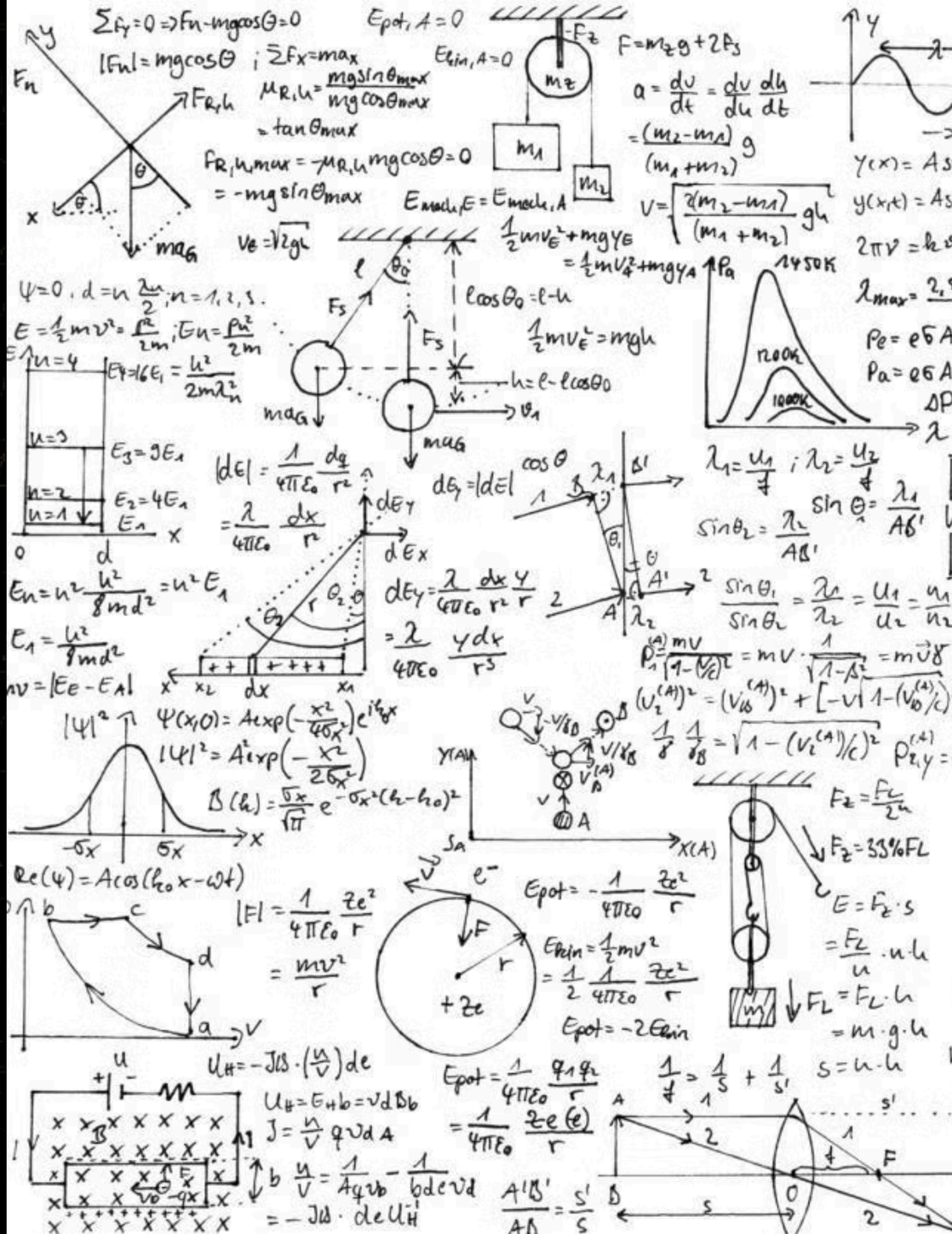
WE GOT 8TH PLACE FOR PRELIM 1.
WE GOT 9TH PLACE FOR PRELIM 2.
WE GOT 4TH PLACE FOR THE FINAL ROUND (INCLUDING THE EXTERNAL GROUPS)

CORRECTNESS ISSUES

NO CORRECTNESS ISSUES WERE BROUGHT UP FOR OUR GROUP, NEITHER DID WE FIND ANY CORRECTNESS ISSUES FOR THE ASSIGNED GROUP.

DESC. OF THE ALGORITHM

(03)



ORIGINAL GROUP 0

03

THE ORIGINAL GROUP ZERO USED A VERY INEFFICIENT CODE FOR THE SORTING ALGORITHM.

THE DISTANCE CALCULATION WAS DONE BY EVERY COMPARISON RECOMPUTING HAMMING DISTANCE BETWEEN TWO STRINGS INSIDE THE COMPARATOR.

THE BINARY VALUE COMPARISON USED BIGINTEGER CONVERSION FOR COMPARING BINARY VALUES WHEN DISTANCES TIED.

ARRAYS.SORT() WAS USED WITH A CUSTOM COMPARATOR WHICH DOES A COMPARISON BASED SORT O(N LOG N) AND RECOMPUTES WORK CONSTANTLY.

GROUP 2

03

OUR IMPLEMENTATION GROUP 2

WE PRECOMPUTED THE HAMMING DISTANCE FROM EACH STRING TO THE TARGET ONCE AND STORED IT IN AN ARRAY.

REPLACED WITH DIRECT LEXICOGRAPHIC COMPARISON OF CHARACTERS, WHICH IS MUCH FASTER.

USED COUNTING SORT BY DISTANCE (LINEAR TIME) AND THEN QUICKSORT ONLY INSIDE SMALL EQUAL-DISTANCE BUCKETS.

IF STRING LENGTH $\leqslant$ 64 BITS, CONVERTED STRINGS TO LONG AND USED XOR + BITCOUNT() TO COMPUTE HAMMING DISTANCE IN HARDWARE.

03

ALGORITHM OVERVIEW

PRECOMPUTE DISTANCES:

FOR EACH STRING, COMPUTE HAMMING DISTANCE TO TARGET USING XOR

COUNTING SORT:

GROUP STRINGS BY DISTANCE (LINEAR TIME)

QUICKSORT WITHIN GROUPS:

SORT EACH BUCKET BY BINARY VALUE

OUTPUT FINAL SORTED ARRAY

Comparing The Two

FEATURE	GROUP 0	GROUP 2
DISTANCE COMPUTATION	RECOMPUTED EVERY COMPARISON O(NLOGN*L)	PRECOMPUTED ONCE PER STRING(O(N * L))
BINARY NUMERIC COMPARISON	USES BIGINTEGER CONVERSION (SLOW)	DIRECT LEXICOGRAPHIC COMAPRISON(FAST)
SORTING METHOD	USES TIMSORT (COMPARISON BASED)	COUNTING SORT BY DISTANCE (LINEAR)
COMPARATOR CALLS	IN THE MILLIONS	ONLY SMALL BUCKETS
MEMORY USAGE	STRING ONLY	STRING + INT[N] + TEMP ARRAY
EXPECTED RUNTIME	O(N LOG(N) * L)	O(N * L)

BIGGEST DIFFERENCES

- PRECOMPUTING VS RECOMPUTING THE HAMMING DISTANCE

GROUP 0 (RECOMPUTING): DISTANCE COMPUTED INSIDE THE COMPARATOR
CALLED: N*LOG_2(N)
IF N = 1,000,000 L = 100
1,000,000*(LOG_2(1,000,000)) = 20,000,000 CALLS
20,000,000 * 100 = 2B TOTAL OPERATIONS
VS
GROUP 2(PRECOMPUTE):
CALLED: O(**N**) = 1,000,000

- USING COUNTING SORT VS COMPARISON BASED SORTING

GROUP 0 (TIMSORT):
CALLED: N*LOG_2(N)
IF N = 1,000,000 L = 100
1,000,000*(LOG_2(1,000,000)) = 20,000,000 CALLS
20,000,000 * 100 = 2B TOTAL ITERATIONS
VS
GROUP 2(COUNTING SORT):
CALLS: N + L = 1,000,000 + 120

COMPLEXITY & DATA STRUCTURES

04

DESCRIPTION

WORST CASE RUNTIME: O(NL + N LOG N)
EXPECTED RUNTIME: O(NL)

MEMORY USAGE: O(N+L) (COUNTING SORT)

OPTIMIZATION FOCUS WAS ON AVOIDING RECOMPUTING DISTANCE AND AVOIDING HEAVY OBJECTS LIKE BIGINTEGER.

05

REFLECTION

EARLY RUNS FAILED DUE TO MISSING CODE AND INCORRECT SORTING.

OUR FINAL VERSION FIXED CORRECTNESS AND OPTIMIZED PERFORMANCE.

AVOIDING UNNECCESARY CONVERSIONS WAS AN IMPORTANT PART IN FIXING CORRECTNESS.

THANK YOU