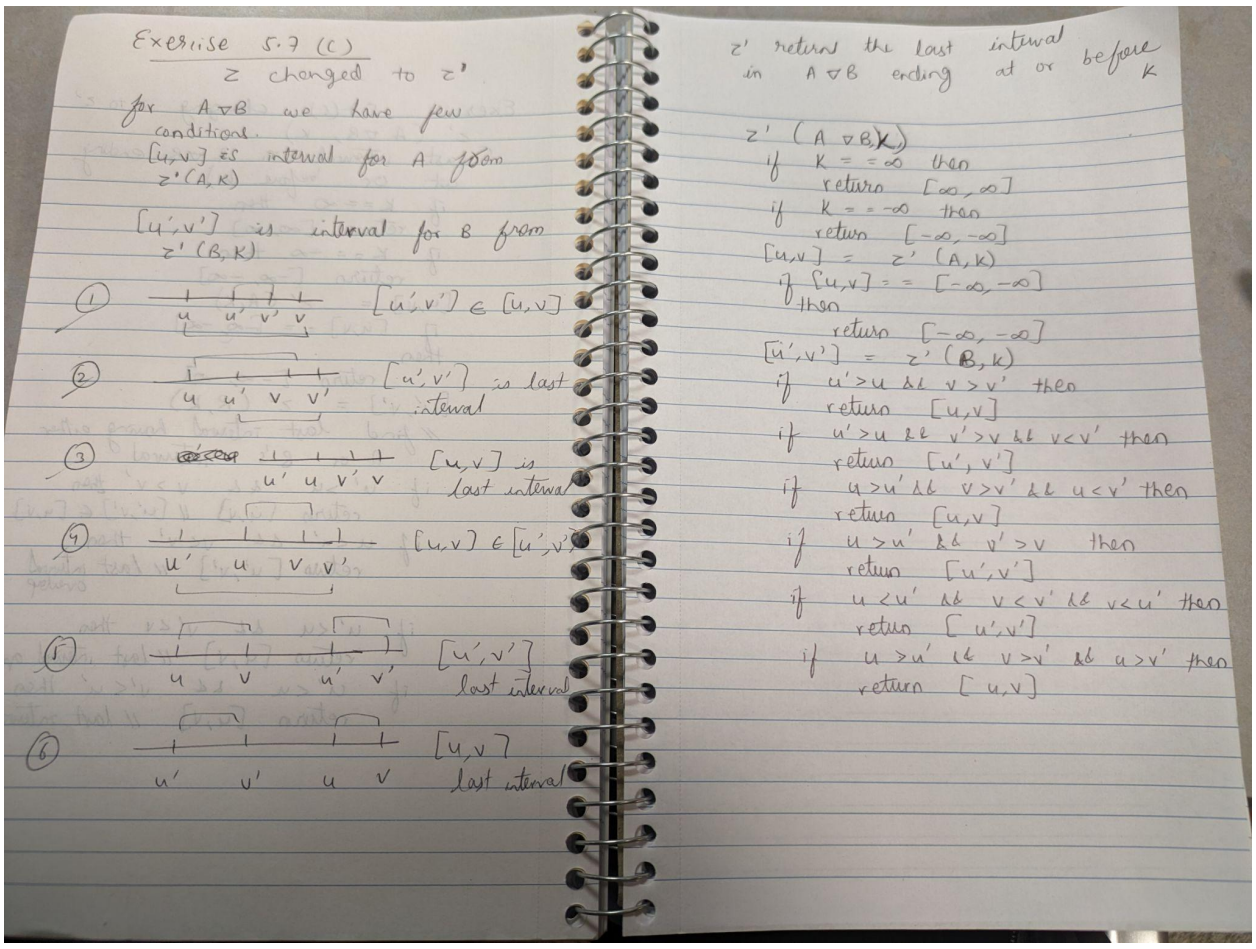
PART 1

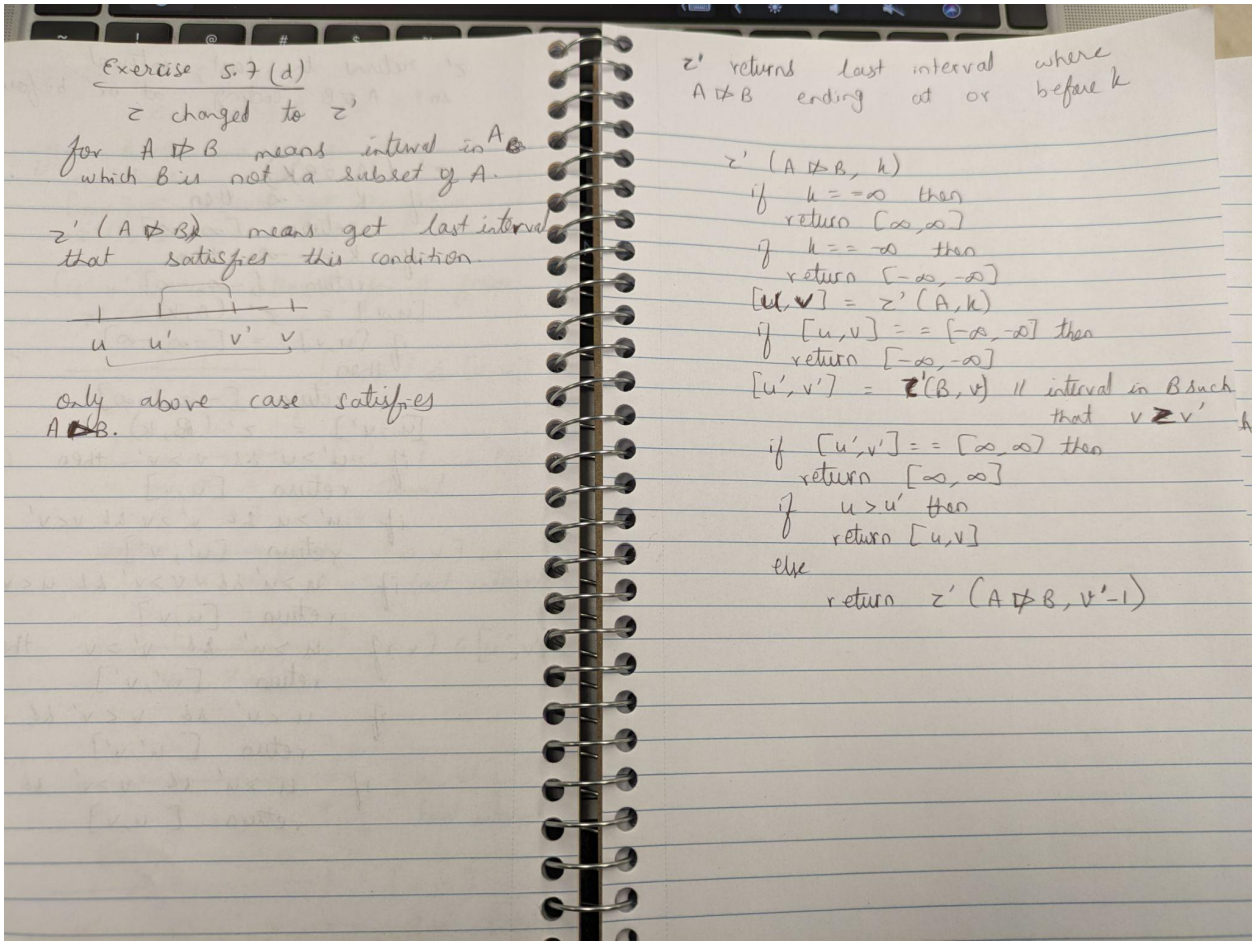
1. Exercise 5.7 (c) and (d) where you change τ to τ', (e) where you change ρ to ρ'

Solution:

5.7 c)



5.7(d)



Exercise 5.7 (d)

z changed to z'

for $A \not\supset B$ means interval in A which B is not a subset of A.

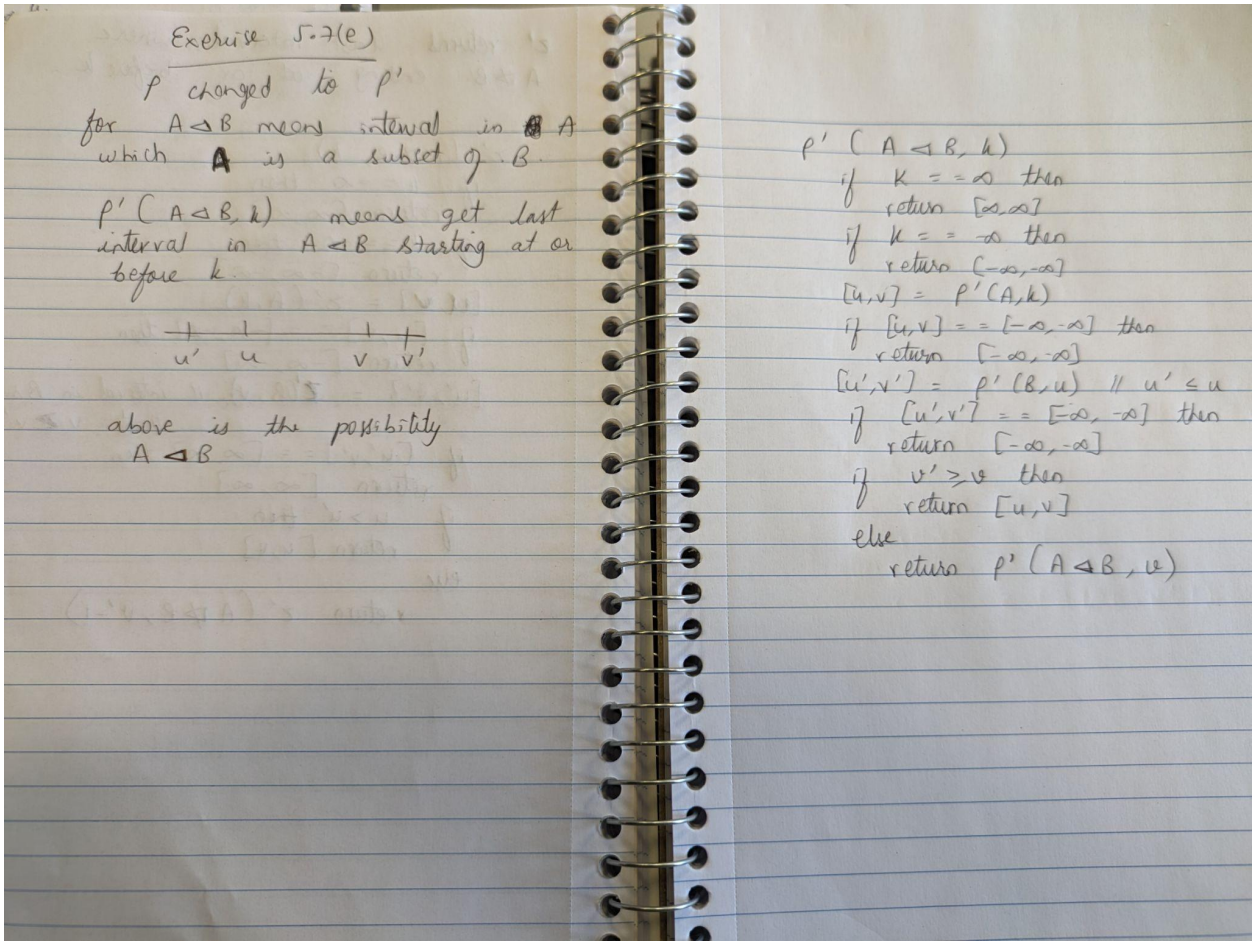
$z'(A \not\supset B)$ means get last interval that satisfies this condition

Only above case satisfies $A \not\supset B$.

z' returns last interval where $A \not\supset B$ ending at or before k

```
z'(A ⊅ B, k)
if k == ∞ then
    return [∞, ∞]
if k == -∞ then
    return [-∞, -∞]
[u, v] = z'(A, k)
if [u, v] == [-∞, -∞] then
    return [-∞, -∞]
[u', v'] = z'(B, v)  // interval in B such that v ≥ v'
if [u', v'] == [∞, ∞] then
    return [∞, ∞]
if u > u' then
    return [u, v]
else
    return z'(A ⊅ B, v'-1)
```

5.7(e)



Exercise 5.7(e)

ρ changed to ρ'

for $A \lhd B$ means interval in A which A is a subset of B.

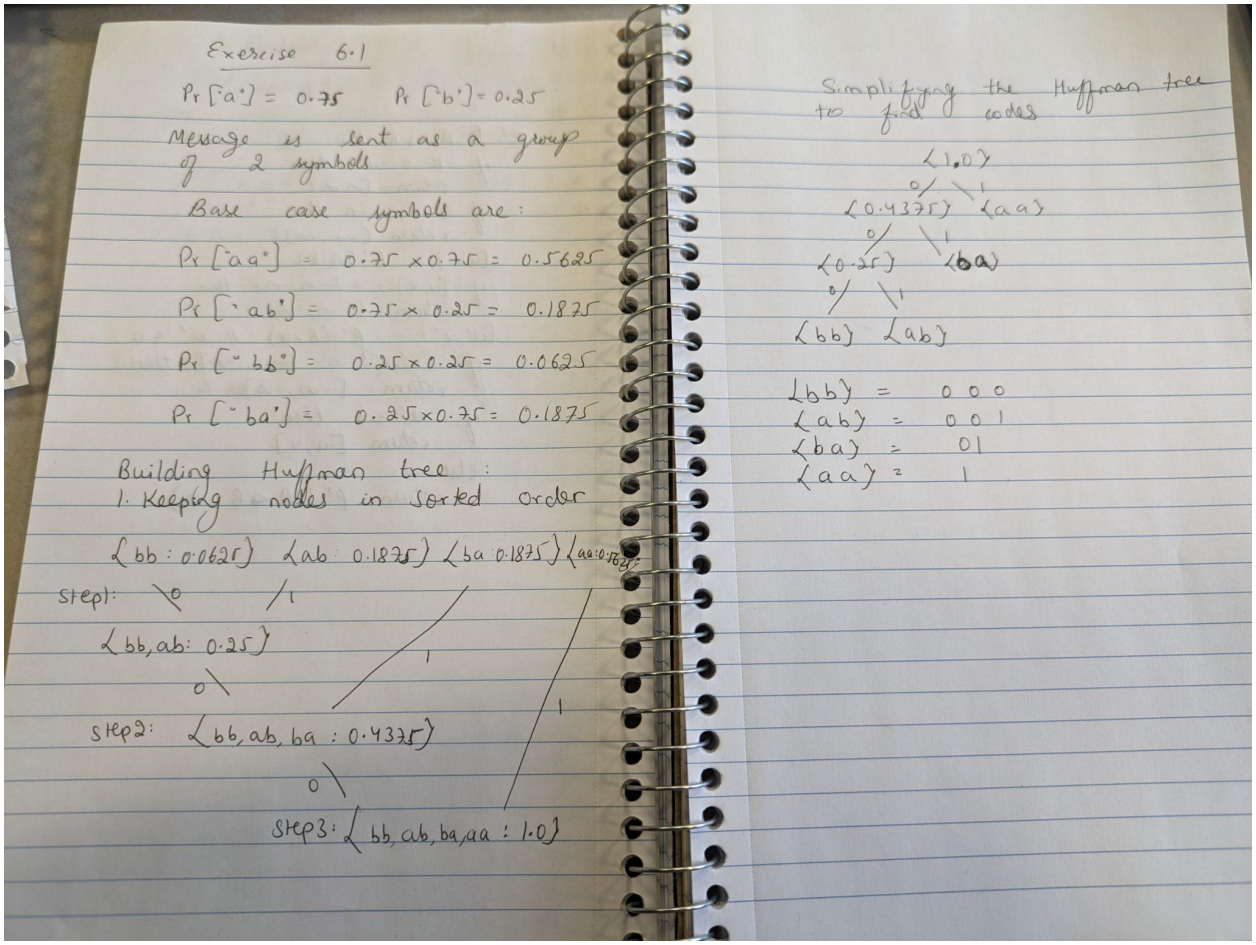
$\rho'(A \lhd B, k)$ means get last interval in $A \lhd B$ starting at or before k.

above is the possibility $A \lhd B$

$\rho'(A \lhd B, k)$
if $k == \infty$ then
return $[\infty, \infty]$
if $k == -\infty$ then
return $[-\infty, -\infty]$
$[u, v] = \rho'(A, k)$
if $[u, v] == [-\infty, -\infty]$ then
return $[-\infty, -\infty]$
$[u', v'] = \rho'(B, u)$ // $u' \leq u$
if $[u', v'] == [-\infty, -\infty]$ then
return $[-\infty, -\infty]$
if $v' \geq v$ then
return $[u, v]$
else
return $\rho'(A \lhd B, v)$

2. Exercise 6.1 where Pr["a"] = 0.75 and Pr["b"] = 0.25

Solution:



3. Show the ω code is prefix free (use induction)

Solution:

To show that the ω code is prefix-free using induction, we need to demonstrate that no codeword in the ω code is a prefix of another codeword. We can prove this property through mathematical induction.

Base Case (k = 1):
For k = 1, the ω code representation is simply "0," which is a single bit. Since there is only one codeword for k = 1, there are no other codewords to compare it to, and it vacuously satisfies the prefix-free property.

Inductive Hypothesis:

Assume that for some positive integer n, the ω code is prefix-free for all integers from 1 to n.
Inductive Step:
We want to show that the ω code remains prefix-free for n + 1. Let's consider the ω code representation for n + 1:
We start with "0."
We encode n + 1 as a binary number and prepend it to the existing code.
We then update n as ⌊log2(n + 1)⌋.
We know that by our inductive hypothesis, the ω code for integers from 1 to n is prefix-free. Let's assume there is a contradiction and that the ω code for n + 1 has a prefix that matches another codeword for some integer from 1 to n.
In the ω code, the binary representation of n + 1 is prepended to the existing code. If this binary representation matches another code, it implies that the binary representation of n + 1 must be a prefix of the binary representation of some integer from 1 to n.
However, this is not possible because n + 1 is larger than any integer from 1 to n, and the binary representation of n + 1 is not a prefix of any binary representation of a smaller integer.
Therefore, our assumption that the ω code for n + 1 has a prefix that matches another codeword is incorrect. This means that the ω code remains prefix-free for n + 1.
By the principle of mathematical induction, since it holds for the base case (k = 1) and the inductive step (n + 1), we can conclude that the ω code is indeed prefix-free for all positive integers.

4. Exercise 6.7

Solution:

Given N/Nt = 137 (p = 0.0073):

- M Calculation:*
  - M* is determined using Golomb coding: M* = -log(2) / log(1 - Nt/N)
  - Substituting the values: M* ≈ -1 / log(0.9927) ≈ -1 / (-0.0073) ≈ 136.99
- **M* (Golomb) can be either 136 or 137.**
- M (Rice) Calculation:*
  - Calculate the logarithm of M*: log(M*) ≈ 4.92
  - Floor and ceiling values for log(M*): floor(x) = 4, ceil(x) = 5
  - **M* (Rice) ranges from 2^4 to 2^5, which is 16 to 32.**
- M(opt) Calculation:*

        - M*(opt) represents the optimal value of M for Golomb coding:
        - Calculate M*(opt) = ceil(log(2 - Nt/N) / -log(1 - Nt/N))
        - Substituting the values: M*(opt) ≈ ceil(136.229305) ≈ 137
- Golomb = 137
- Bits Calculation for Golomb and Rice:
    - In Golomb coding:
        - The remainder part of the range [0, 2^ceil(log2(M)) - M - 1] requires 7 bits.
        - The part [2^ceil(log2(M)) - M, M-1] needs 8 bits.
    - For Rice coding with the range of 16 to 32, 5 bits are required.
- **Wasted Bits Comparison:**
    - The wasted bits in Rice coding compared to Golomb can be calculated as (2^ceil(log2(M)) - M) bits. These represent the fewer bits in the remainder part of Golomb for the first 2^ceil(log2(M)) - M codewords.

In this example with p = 0.0073, if Rice coding is used instead of Golomb, 119 bits would be wasted due to the difference in the coding method for the given M value of 137.

Arithmetic coding vs golomb

let's perform the calculations for the average number of bits wasted when using Golomb coding instead of arithmetic coding for the given dataset with probabilities. We'll use the example probabilities as previously provided:

- P(0) = 0.1
- P(1) = 0.2
- P(2) = 0.15
- P(3) = 0.15
- P(4) = 0.1
- P(5) = 0.1
- P(6) = 0.1
- P(7) = 0.1

We'll calculate the Golomb and arithmetic coding for each integer value and then find the difference in the expected code lengths, representing the average bits wasted.

- Expected code length for Golomb coding using M* (optimal Golomb parameter) ≈ 2.92:
- Golomb code length for 0 = (floor(0 / 2.92)) + 2 * (0 % 2.92) + 1 = 1

- Golomb code length for 1 = (floor(1 / 2.92)) + 2 * (1 % 2.92) + 1 = 3
- Golomb code length for 2 = (floor(2 / 2.92)) + 2 * (2 % 2.92) + 1 = 4
- Golomb code length for 3 = (floor(3 / 2.92)) + 2 * (3 % 2.92) + 1 = 4
- Golomb code length for 4 = (floor(4 / 2.92)) + 2 * (4 % 2.92) + 1 = 2
- Golomb code length for 5 = (floor(5 / 2.92)) + 2 * (5 % 2.92) + 1 = 2
- Golomb code length for 6 = (floor(6 / 2.92)) + 2 * (6 % 2.92) + 1 = 2
- Golomb code length for 7 = (floor(7 / 2.92)) + 2 * (7 % 2.92) + 1 = 2
- Expected code length for arithmetic coding using the given probabilities:
- Code length for 0 = -log2(0.1) ≈ 3.32 bits
- Code length for 1 = -log2(0.2) ≈ 2.32 bits
- Code length for 2 = -log2(0.15) ≈ 2.74 bits
- Code length for 3 = -log2(0.15) ≈ 2.74 bits
- Code length for 4 = -log2(0.1) ≈ 3.32 bits
- Code length for 5 = -log2(0.1) ≈ 3.32 bits
- Code length for 6 = -log2(0.1) ≈ 3.32 bits
- Code length for 7 = -log2(0.1) ≈ 3.32 bits
- Calculate the average bits wasted:
- Average bits wasted = (Expected code length for arithmetic coding) - (Expected code length for Golomb coding) = [3.32 + 2.32 + 2.74 + 2.74 + 3.32 + 3.32 + 3.32 + 3.32] - [1 + 3 + 4 + 4 + 2 + 2 + 2 + 2] = 35.62 bits - 20 bits = 15.62 bits

So, in this specific example, on average, about 15.62 bits are wasted per symbol when using Golomb coding with the optimal Golomb parameter M* instead of arithmetic coding. This represents the extra bits needed to encode the same dataset using Golomb coding compared to arithmetic coding.

5. Suppose we are using incremental indexing and we have currently generations 1, 2, 4. For the next three new partitions to be written to disk, what merging will occur? What will be the new generations after the incremental merging has been done? (Show after each new partition).

Solution:

1 **2** **4** we have gen1 , gen2 and gen4 paritions in disk.

3 **4** After 1st partition is written to disk, we have 2 gen1 partitions which merge to form 1 gen2 index. Now we have 2 gen2 in disk which merge to form gen3 index in disk. So we have in step 2 gen3 and gen4.

1 **3** **4** Next partition is written to disk , now we have gen1 , gen3 and gen 4

2 **3** **4** Next partition is written to disk, the disk would now contain 2 gen1 partitions which merge to gen2 index. Finally we have gen2, gen3 and gen4.

PART 2

Experiments-

1) Trec eval runs are in transcript.txt file. Here we have placed the screenshots of the 2 query rules and evaluation using trec eval.

We have kept few files for running the trec eval.

The **rel files** are : trec_rel_query_1 and trec_rel_query_2

(marked relevant for docs is they have atleast “halifax” in them for query 1 and “casualties” in them for query 2)

The **top files** are :

trec_eval_top_BM25_docAtATime_Query_1,
trec_eval_top_BM25_docAtATime_Query_2,
trec_eval_top_BM25_termAtATime_Query_1,
trec_eval_top_BM25_termAtATime_Query_2

The output of the files are in the required format of

1 0 {doc_id} {idx+1} {score} BM_TermAtATime

1 0 {doc_id} {idx+1} {score} BM_DocAtATime

Screenshots for the trec eval

a) Trec eval result for query 1 and docAtaTime rank

```
vigy@Vigneshs-MacBook-Pro files % trec_eval trec_rel_query_1.txt trec_eval_top_BM25_docAtATime_Query_1.txt
runid                   all     my-test
num_q                   all     1
num_ret                 all     10
num_rel                 all     10
num_rel_ret             all     10
map                     all     1.0000
gm_map                  all     1.0000
Rprec                   all     1.0000
bpref                   all     1.0000
recip_rank              all     1.0000
iprec_at_recall_0.00    all     1.0000
iprec_at_recall_0.10    all     1.0000
iprec_at_recall_0.20    all     1.0000
iprec_at_recall_0.30    all     1.0000
iprec_at_recall_0.40    all     1.0000
iprec_at_recall_0.50    all     1.0000
iprec_at_recall_0.60    all     1.0000
iprec_at_recall_0.70    all     1.0000
iprec_at_recall_0.80    all     1.0000
iprec_at_recall_0.90    all     1.0000
iprec_at_recall_1.00    all     1.0000
P_5                     all     1.0000
P_10                    all     1.0000
P_15                    all     0.6667
P_20                    all     0.5000
P_30                    all     0.3333
P_100                   all     0.1000
P_200                   all     0.0500
P_500                   all     0.0200
P_1000                  all     0.0100
```

b) Trec eval result for query 2 and docAtaTime rank

```
vigy@Vigneshs-MacBook-Pro files % trec_eval trec_rel_query_2.txt trec_eval_top_BM25_docAtATime_Query_2.txt
runid                   all     my-test
num_q                   all     1
num_ret                 all     6
num_rel                 all     6
num_rel_ret             all     6
map                     all     1.0000
gm_map                  all     1.0000
Rprec                   all     1.0000
bpref                   all     1.0000
recip_rank              all     1.0000
iprec_at_recall_0.00    all     1.0000
iprec_at_recall_0.10    all     1.0000
iprec_at_recall_0.20    all     1.0000
iprec_at_recall_0.30    all     1.0000
iprec_at_recall_0.40    all     1.0000
iprec_at_recall_0.50    all     1.0000
iprec_at_recall_0.60    all     1.0000
iprec_at_recall_0.70    all     1.0000
iprec_at_recall_0.80    all     1.0000
iprec_at_recall_0.90    all     1.0000
iprec_at_recall_1.00    all     1.0000
P_5                     all     1.0000
P_10                    all     0.6000
P_15                    all     0.4000
P_20                    all     0.3000
P_30                    all     0.2000
P_100                   all     0.0600
P_200                   all     0.0300
P_500                   all     0.0120
P_1000                  all     0.0060
```

c) Trec eval result for query 1 and termAtaTime rank

```
vigy@Vigneshs-MacBook-Pro files % trec_eval trec_rel_query_1.txt trec_eval_top_BM25_termAtATime_Query_1.txt
runid                   all     BM_TermAtATime
num_q                   all     1
num_ret                 all     9
num_rel                 all     10
num_rel_ret             all     7
map                     all     0.6268
gm_map                  all     0.6268
Rprec                   all     0.7000
bpref                   all     0.6000
recip_rank              all     1.0000
iprec_at_recall_0.00    all     1.0000
iprec_at_recall_0.10    all     1.0000
iprec_at_recall_0.20    all     1.0000
iprec_at_recall_0.30    all     1.0000
iprec_at_recall_0.40    all     0.8571
iprec_at_recall_0.50    all     0.8571
iprec_at_recall_0.60    all     0.8571
iprec_at_recall_0.70    all     0.7778
iprec_at_recall_0.80    all     0.0000
iprec_at_recall_0.90    all     0.0000
iprec_at_recall_1.00    all     0.0000
P_5                     all     0.8000
P_10                    all     0.7000
P_15                    all     0.4667
P_20                    all     0.3500
P_30                    all     0.2333
P_100                   all     0.0700
P_200                   all     0.0350
P_500                   all     0.0140
P_1000                  all     0.0070
```

d) Trec eval result for query 2 and termAtaTime rank

```
vigy@Vigneshs-MacBook-Pro files % trec_eval trec_rel_query_2.txt trec_eval_top_BM25_termAtATime_Query_2.txt
runid                   all     BM_TermAtATime
num_q                   all     1
num_ret                 all     10
num_rel                 all     6
num_rel_ret             all     6
map                     all     0.6347
gm_map                  all     0.6347
Rprec                   all     0.6667
bpref                   all     0.7000
recip_rank              all     0.5000
iprec_at_recall_0.00    all     0.7500
iprec_at_recall_0.10    all     0.7500
iprec_at_recall_0.20    all     0.7500
iprec_at_recall_0.30    all     0.7500
iprec_at_recall_0.40    all     0.7500
iprec_at_recall_0.50    all     0.7500
iprec_at_recall_0.60    all     0.6667
iprec_at_recall_0.70    all     0.6250
iprec_at_recall_0.80    all     0.6250
iprec_at_recall_0.90    all     0.6000
iprec_at_recall_1.00    all     0.6000
P_5                     all     0.6000
P_10                    all     0.6000
P_15                    all     0.4000
P_20                    all     0.3000
P_30                    all     0.2000
P_100                   all     0.0600
P_200                   all     0.0300
P_500                   all     0.0120
P_1000                  all     0.0060
```

FYI using top 10 documents and acc used in term at a time is 20

2) Accumulator Varying experiments

Query 1- "the halifax explosion”

Accumulator count	BM25_TermATATime result
20	1 0 6 1 0.2638357938638247 BM_TermAtATime 1 0 4 2 0.2479650991668339 BM_TermAtATime 1 0 5 3 0.23285265972369193 BM_TermAtATime 1 0 0 4 0.2317742992405818 BM_TermAtATime 1 0 2 5 0.22895055177605167 BM_TermAtATime 1 0 3 6 0.19496122882681294 BM_TermAtATime 1 0 1 7 0.19496122882681294 BM_TermAtATime 1 0 8 8 0.186668076425166 BM_TermAtATime 1 0 9 9 0.15672328500477978 BM_TermAtATime 1 0 9 10 0.15672328500477978 BM_TermAtATime
25	1 0 6 1 0.2638357938638247 BM_TermAtATime 1 0 4 2 0.2479650991668339 BM_TermAtATime 1 0 5 3 0.23285265972369193 BM_TermAtATime 1 0 0 4 0.2317742992405818 BM_TermAtATime 1 0 2 5 0.22895055177605167 BM_TermAtATime 1 0 3 6 0.19496122882681294 BM_TermAtATime 1 0 1 7 0.19496122882681294 BM_TermAtATime 1 0 8 8 0.186668076425166 BM_TermAtATime 1 0 9 9 0.15672328500477978 BM_TermAtATime 1 0 7 10 0.0 BM_TermAtATime
30	1 0 6 1 0.2638357938638247 BM_TermAtATime 1 0 4 2 0.2479650991668339 BM_TermAtATime 1 0 5 3 0.23285265972369193 BM_TermAtATime 1 0 0 4 0.2317742992405818 BM_TermAtATime 1 0 2 5 0.22895055177605167 BM_TermAtATime 1 0 3 6 0.19496122882681294 BM_TermAtATime 1 0 1 7 0.19496122882681294 BM_TermAtATime 1 0 8 8 0.186668076425166 BM_TermAtATime 1 0 9 9 0.15672328500477978 BM_TermAtATime 1 0 7 10 0.0 BM_TermAtATime
35	1 0 6 1 0.2638357938638247 BM_TermAtATime 1 0 4 2 0.2479650991668339 BM_TermAtATime 1 0 5 3 0.23285265972369193 BM_TermAtATime 1 0 0 4 0.2317742992405818 BM_TermAtATime 1 0 2 5 0.22895055177605167 BM_TermAtATime 1 0 3 6 0.19496122882681294 BM_TermAtATime 1 0 1 7 0.19496122882681294 BM_TermAtATime 1 0 8 8 0.186668076425166 BM_TermAtATime 1 0 9 9 0.15672328500477978 BM_TermAtATime 1 0 7 10 0.0 BM_TermAtATime
40	1 0 6 1 0.2638357938638247 BM_TermAtATime 1 0 4 2 0.2479650991668339 BM_TermAtATime 1 0 5 3 0.23285265972369193 BM_TermAtATime 1 0 0 4 0.2317742992405818 BM_TermAtATime 1 0 2 5 0.22895055177605167 BM_TermAtATime 1 0 3 6 0.19496122882681294 BM_TermAtATime 1 0 1 7 0.19496122882681294 BM_TermAtATime 1 0 8 8 0.186668076425166 BM_TermAtATime 1 0 9 9 0.15672328500477978 BM_TermAtATime 1 0 7 10 0.0 BM_TermAtATime

Query 2- "the huge casualties"

Accumulator count	BM25_TermATATime result
20	1 0 7 1 2.3705305568030535 BM_TermAtATime 1 0 4 2 2.2140665646002358 BM_TermAtATime 1 0 2 3 1.901723699254437 BM_TermAtATime 1 0 3 4 1.8636389916765572 BM_TermAtATime 1 0 0 5 0.6963144867362121 BM_TermAtATime 1 0 1 6 0.6670756358690695 BM_TermAtATime 1 0 9 7 0.0 BM_TermAtATime 1 0 8 8 0.0 BM_TermAtATime 1 0 6 9 0.0 BM_TermAtATime 1 0 5 10 0.0 BM_TermAtATime
25	1 0 7 1 2.3705305568030535 BM_TermAtATime 1 0 4 2 2.2140665646002358 BM_TermAtATime 1 0 2 3 1.901723699254437 BM_TermAtATime 1 0 3 4 1.8636389916765572 BM_TermAtATime 1 0 0 5 0.6963144867362121 BM_TermAtATime 1 0 1 6 0.6670756358690695 BM_TermAtATime 1 0 9 7 0.0 BM_TermAtATime 1 0 8 8 0.0 BM_TermAtATime 1 0 6 9 0.0 BM_TermAtATime 1 0 5 10 0.0 BM_TermAtATime
30	1 0 7 1 2.3705305568030535 BM_TermAtATime 1 0 4 2 2.2140665646002358 BM_TermAtATime 1 0 2 3 1.901723699254437 BM_TermAtATime 1 0 3 4 1.8636389916765572 BM_TermAtATime 1 0 0 5 0.6963144867362121 BM_TermAtATime 1 0 1 6 0.6670756358690695 BM_TermAtATime 1 0 9 7 0.0 BM_TermAtATime 1 0 8 8 0.0 BM_TermAtATime 1 0 6 9 0.0 BM_TermAtATime 1 0 5 10 0.0 BM_TermAtATime
35	1 0 7 1 2.3705305568030535 BM_TermAtATime 1 0 4 2 2.2140665646002358 BM_TermAtATime 1 0 2 3 1.901723699254437 BM_TermAtATime 1 0 3 4 1.8636389916765572 BM_TermAtATime 1 0 0 5 0.6963144867362121 BM_TermAtATime 1 0 1 6 0.6670756358690695 BM_TermAtATime 1 0 9 7 0.0 BM_TermAtATime 1 0 8 8 0.0 BM_TermAtATime 1 0 6 9 0.0 BM_TermAtATime 1 0 5 10 0.0 BM_TermAtATime
40	1 0 7 1 2.3705305568030535 BM_TermAtATime 1 0 4 2 2.2140665646002358 BM_TermAtATime 1 0 2 3 1.901723699254437 BM_TermAtATime 1 0 3 4 1.8636389916765572 BM_TermAtATime 1 0 0 5 0.6963144867362121 BM_TermAtATime 1 0 1 6 0.6670756358690695 BM_TermAtATime 1 0 9 7 0.0 BM_TermAtATime 1 0 8 8 0.0 BM_TermAtATime 1 0 6 9 0.0 BM_TermAtATime 1 0 5 10 0.0 BM_TermAtATime

Result analysis: In the first query as the accumulate size increases, the last document is scored 0, other tha that we dont see much of a difference in the result.

3) Relative speed experiment

Query, top 10	BM25_DocAtATimeHEAPMS (in secs)	BM25_TermAtATimeWithQUIT (in secs) acc=20
the halifax explosion	Time : 0.0023614390001966967	Total elapsed: 0.0001258850097656 25s
the huge casualties	Time : 0.0017484889995103003	Total elapsed: 0.0001559257507324 2188s

Result analysis:

As per above scenario , we have term at a time with QUIT faster that Doc at a time with heap max score.