

## PART 1

1. Exercise 5.7 (c) and (d) where you change  $\tau$  to  $\tau'$ , (e) where you change  $\rho$  to  $\rho'$

Solution:

5.7 c)

Exercise 5.7 (c)  
 $z$  changed to  $z'$   
 for  $A \vee B$  we have few conditions.  
 $[u, v]$  is interval for  $A$  from  $z'(A, k)$   
 $[u', v']$  is interval for  $B$  from  $z'(B, k)$

①  $[u', v'] \in [u, v]$

②  $[u', v']$  is last interval

③  $[u, v]$  is last interval

④  $[u, v] \in [u', v']$

⑤  $[u', v']$  last interval

⑥  $[u, v]$  last interval

$z'$  returns the last interval in  $A \vee B$  ending at or before  $k$

$z'(A \vee B, k)$

if  $k = -\infty$  then  
 return  $[-\infty, -\infty]$

if  $k = \infty$  then  
 return  $[-\infty, \infty]$

$[u, v] = z'(A, k)$   
 if  $[u, v] = [-\infty, -\infty]$  then  
 return  $[-\infty, -\infty]$

$[u', v'] = z'(B, k)$   
 if  $u' > u$  &  $v > v'$  then  
 return  $[u, v]$

if  $u' > u$  &  $v' > v$  &  $v < v'$  then  
 return  $[u', v']$

if  $u > u'$  &  $v > v'$  &  $u < v'$  then  
 return  $[u, v]$

if  $u > u'$  &  $v' > v$  then  
 return  $[u', v']$

if  $u < u'$  &  $v < v'$  &  $v < u'$  then  
 return  $[u', v']$

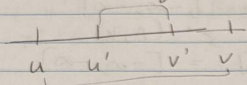
if  $u > u'$  &  $v > v'$  &  $u > v'$  then  
 return  $[u, v]$

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  $k$

$z'(A \not\supset B, k)$

if  $k = -\infty$  then

return  $[-\infty, -\infty]$

if  $k = \infty$  then

return  $[\infty, \infty]$

$[u, v] = z'(A, k)$

if  $[u, v] = [-\infty, -\infty]$  then

return  $[-\infty, -\infty]$

$[u', v'] = z'(B, v)$  // interval in  $B$  such  
that  $v \geq v'$

if  $[u', v'] = [\infty, \infty]$  then

return  $[\infty, \infty]$

if  $u > u'$  then

return  $[u, v]$

else

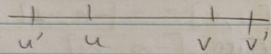
return  $z'(A \not\supset B, v'-1)$

5.7(e)

Exercise 5.7(e)

$P$  changed to  $P'$   
for  $A \triangleleft B$  means interval in  $A$   
which  $A$  is a subset of  $B$ .

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



above is the possibility  
 $A \triangleleft B$

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



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

Solution:

Exercise 6.1

$\Pr["a"] = 0.75$     $\Pr["b"] = 0.25$

Message is sent as a group of 2 symbols

Base case symbols are:

$\Pr["aa"] = 0.75 \times 0.75 = 0.5625$

$\Pr["ab"] = 0.75 \times 0.25 = 0.1875$

$\Pr["bb"] = 0.25 \times 0.25 = 0.0625$

$\Pr["ba"] = 0.25 \times 0.75 = 0.1875$

Building Huffman tree:

1. Keeping nodes in sorted order

$\{bb: 0.0625\}$   $\{ab: 0.1875\}$   $\{ba: 0.1875\}$   $\{aa: 0.5625\}$

Step 1:  $\{bb, ab: 0.25\}$

Step 2:  $\{bb, ab, ba: 0.4375\}$

Step 3:  $\{bb, ab, ba, aa: 1.0\}$

Simplifying the Huffman tree to find codes

$\{1.0\}$

$\{0.4375\}$   $\{aa\}$

$\{0.25\}$   $\{ba\}$

$\{bb\}$   $\{ab\}$

$\{bb\} = 000$

$\{ab\} = 001$

$\{ba\} = 01$

$\{aa\} = 1$

## 3. Show the $\omega$ code is prefix free (use induction)

Solution:

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

Base Case ( $k = 1$ ):

For  $k = 1$ , the  $\omega$  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  $\omega$  code is prefix-free for all integers from 1 to  $n$ .

Inductive Step:

We want to show that the  $\omega$  code remains prefix-free for  $n + 1$ . Let's consider the  $\omega$  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  $\lfloor \log_2(n + 1) \rfloor$ .

We know that by our inductive hypothesis, the  $\omega$  code for integers from 1 to  $n$  is prefix-free. Let's assume there is a contradiction and that the  $\omega$  code for  $n + 1$  has a prefix that matches another codeword for some integer from 1 to  $n$ .

In the  $\omega$  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  $\omega$  code for  $n + 1$  has a prefix that matches another codeword is incorrect. This means that the  $\omega$  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  $\omega$  code is indeed prefix-free for all positive integers.

#### 4. Exercise 6.7

Solution:

Given  $N/N_t = 137$  ( $p = 0.0073$ ):

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

- $M^*(opt)$  represents the optimal value of  $M$  for Golomb coding:
- Calculate  $M^*(opt) = \lceil \log(2 - Nt/N) / -\log(1 - Nt/N) \rceil$
- Substituting the values:  $M^*(opt) \approx \lceil 136.229305 \rceil \approx 137$
- Golomb = 137
- Bits Calculation for Golomb and Rice:
  - In Golomb coding:
    - The remainder part of the range  $[0, 2^{\lceil \log_2(M) \rceil} - M - 1]$  requires 7 bits.
    - The part  $[2^{\lceil \log_2(M) \rceil} - M, M-1]$  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  $(2^{\lceil \log_2(M) \rceil} - M)$  bits. These represent the fewer bits in the remainder part of Golomb for the first  $2^{\lceil \log_2(M) \rceil} - M$  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)  $\approx 2.92$ :
- Golomb code length for 0 =  $(\text{floor}(0 / 2.92)) + 2 * (0 \% 2.92) + 1 = 1$

- Golomb code length for 1 =  $(\text{floor}(1 / 2.92)) + 2 * (1 \% 2.92) + 1 = 3$
- Golomb code length for 2 =  $(\text{floor}(2 / 2.92)) + 2 * (2 \% 2.92) + 1 = 4$
- Golomb code length for 3 =  $(\text{floor}(3 / 2.92)) + 2 * (3 \% 2.92) + 1 = 4$
- Golomb code length for 4 =  $(\text{floor}(4 / 2.92)) + 2 * (4 \% 2.92) + 1 = 2$
- Golomb code length for 5 =  $(\text{floor}(5 / 2.92)) + 2 * (5 \% 2.92) + 1 = 2$
- Golomb code length for 6 =  $(\text{floor}(6 / 2.92)) + 2 * (6 \% 2.92) + 1 = 2$
- Golomb code length for 7 =  $(\text{floor}(7 / 2.92)) + 2 * (7 \% 2.92) + 1 = 2$
- Expected code length for arithmetic coding using the given probabilities:
- Code length for 0 =  $-\log_2(0.1) \approx 3.32$  bits
- Code length for 1 =  $-\log_2(0.2) \approx 2.32$  bits
- Code length for 2 =  $-\log_2(0.15) \approx 2.74$  bits
- Code length for 3 =  $-\log_2(0.15) \approx 2.74$  bits
- Code length for 4 =  $-\log_2(0.1) \approx 3.32$  bits
- Code length for 5 =  $-\log_2(0.1) \approx 3.32$  bits
- Code length for 6 =  $-\log_2(0.1) \approx 3.32$  bits
- Code length for 7 =  $-\log_2(0.1) \approx 3.32$  bits
- Calculate the average bits wasted:
- Average bits wasted = (Expected code length for arithmetic coding) - (Expected code length for Golomb coding) =  $[3.32 + 2.32 + 2.74 + 2.74 + 3.32 + 3.32 + 3.32 + 3.32] - [1 + 3 + 4 + 4 + 2 + 2 + 2 + 2] = 35.62 \text{ bits} - 20 \text{ bits} = 15.62 \text{ bits}$

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	<pre>1 0 6 1 0.2638357938638247 BM_TermAtATime 1 0 4 2 0.2479658991668339 BM_TermAtATime 1 0 5 3 0.23285265972369193 BM_TermAtATime 1 0 0 4 0.2317742992405818 BM_TermAtATime 1 0 2 5 0.22895855177605167 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 0.15672328580477978 BM_TermAtATime 1 0 9 10 0.15672328580477978 BM_TermAtATime</pre>
25	<pre>1 0 6 1 0.2638357938638247 BM_TermAtATime 1 0 4 2 0.2479658991668339 BM_TermAtATime 1 0 5 3 0.23285265972369193 BM_TermAtATime 1 0 0 4 0.2317742992405818 BM_TermAtATime 1 0 2 5 0.22895855177605167 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 0.15672328580477978 BM_TermAtATime 1 0 7 10 0.0 BM_TermAtATime</pre>
30	<pre>1 0 6 1 0.2638357938638247 BM_TermAtATime 1 0 4 2 0.2479658991668339 BM_TermAtATime 1 0 5 3 0.23285265972369193 BM_TermAtATime 1 0 0 4 0.2317742992405818 BM_TermAtATime 1 0 2 5 0.22895855177605167 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 0.15672328580477978 BM_TermAtATime 1 0 7 10 0.0 BM_TermAtATime</pre>
35	<pre>1 0 6 1 0.2638357938638247 BM_TermAtATime 1 0 4 2 0.2479658991668339 BM_TermAtATime 1 0 5 3 0.23285265972369193 BM_TermAtATime 1 0 0 4 0.2317742992405818 BM_TermAtATime 1 0 2 5 0.22895855177605167 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 0.15672328580477978 BM_TermAtATime 1 0 7 10 0.0 BM_TermAtATime</pre>
40	<pre>1 0 6 1 0.2638357938638247 BM_TermAtATime 1 0 4 2 0.2479658991668339 BM_TermAtATime 1 0 5 3 0.23285265972369193 BM_TermAtATime 1 0 0 4 0.2317742992405818 BM_TermAtATime 1 0 2 5 0.22895855177605167 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 0.15672328580477978 BM_TermAtATime 1 0 7 10 0.0 BM_TermAtATime</pre>

## Query 2- "the huge casualties"

Accumulator count	BM25_TermAtATime result
20	<pre> 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 </pre>
25	<pre> 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 </pre>
30	<pre> 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 </pre>
35	<pre> 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 </pre>
40	<pre> 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 </pre>

**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_DocAtATimeHEAP MS (in secs)	BM25_TermAtATimeWithQUIT (in secs) acc=20
the halifax explosion	Time : 0.0023614390001966967	Total elapsed: 0.000125885009765625s
the huge casualties	Time : 0.0017484889995103003	Total elapsed: 0.00015592575073242188s

*Result analysis:*

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