LZW inspired De-Duplication algorithm for variable size blocks

Justin

1 Introduction

The variable size defined in the title is a multiple of the smallest block size. So if the block size is 1 KB then the variable size blocks can be of 1, 2, 3 $\cdots$ KBs. In the project I would be utilizing 4 KB blocks(debatable). I haven't yet decided whether to put an upper limit on the block size, but I am just limiting my self to 32 KB, which is 8 block long (highly unlikely to occur). So the block sizes will vary as a multiple of 4, ie. 4, 8, 12, 16, $\cdots$ 32 KB. An empty dictionary is used, as pre-calculating the hashes for all the blocks and adding to dictionary would be expensive on time and space. The dictionary is used only during de-duplication, and is not utilized while retrieving the original data from the de-duplicated data.

2 Algorithm - De-Duplication

1. Start
2. Initialize p (present) = false, size (number of matched blocks) = 0, i = 1
3. Retrieve next (first during the first retrieval) block. (B_i)
4. Compute hash of blocks $B_j, j \in [1, i]$ (H_i)
5. Check if H_i is present in the dictionary: if No goto step 7.
6. If yes, set p = true, size = size + 1, i = i+1. goto step 3.
7. If no, Add hash to dictionary.
8. Check if p = false. if No goto step 10.
9. If yes, retrieve next block, recompute hash, add to dictionary. Print block as it is, as its not a duplicate. Goto step 11.
10. If no, print (index of duplicate_block, size), Set p = false, size = 0, i = 1, move read head one block back.
11. Check if last block. If No goto step 3, else end.

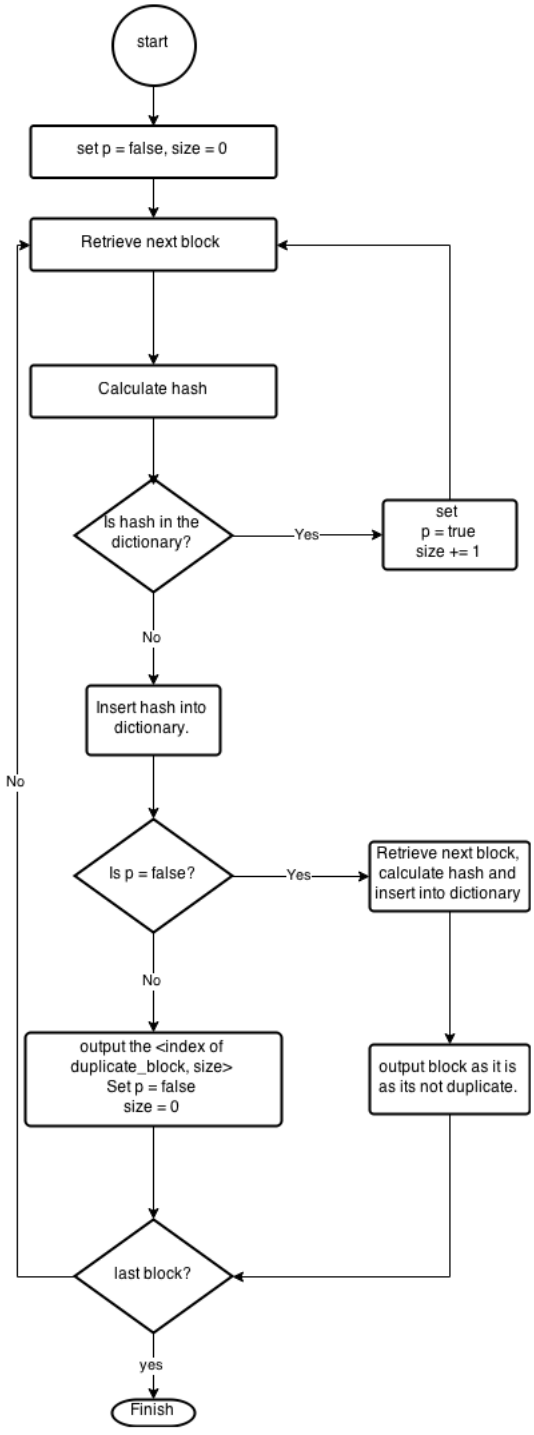

Figure 1: LZW inspired variable block de duplication

3 Re-duplication

1. Start
2. Read next/first entry of de-duplicated data.
3. If data block: output as it is.
4. Else if: index reference, then print blocks from the index till index+size.
5. Last entry? if yes end, else goto step 2.

4 Example

Consider the blocks be "ABABCABBBCABCCC" where each alphabet represents a block of 4KB and the alphabet values can be considered as the hash value.

Index	**Value**
0	A
1	B
2	A
3	B
4	C
...	...

Input = "ABABCABBBCABCCC"

Sl. No.	**Dictionary Value**	**Output Value**
1	A	-
2	AB	A
3	B	-
4	BA	B
5	ABC	(0,2)
6	C	-
7	CA	C
8	ABB	(0,2)
9	BB	(1,1)
10	BC	(1,1)
11	CAB	(4,2)
12	BCC	(3,2)
13	CC	(4,1)
14	-	(4,1)

4.1 Re Duplication

Input = "AB(0,2)C(0,2)(1,1)(1,1)(4,2)(3,2)(4,1)(4,1)"

Serial No.	Input Value	Output Value
1	A	A
2	B	B
3	(0,2)	AB (start at index 0, print 2 blocks)
4	C	C
5	(0,2)	AB
6	(1,1)	B
7	(1,1)	B
8	(4,2)	CA
9	(3,2)	BC
10	(4,1)	C
11	(4,1)	C