Information Retrieval (29.10.2018) Aleksandra Maslennikova

Ex 1

S = {bio, bionic, bit, bitly, buzz, car, caso, cast, z00}

The main idea of algorithm: divide alphabetically ordered strings in groups of size which can be stored inside the memory, then take the first strings of each group-block and and create a ~~tree~~ prefix-tree. So when we will need to find our query we would be able to find the nessecary block using the tree.

1) We divides our set in 3 blocks. We can compess the data inside the blocks so they will take less space on the disk:

   $block_1$ = {bio, bionic, bit} = {bio, 3nic, 2t} (so 3nic is equal to take 3 first letters of last word and add nic; this compession will give good results because words are alphabeticaly ordered)

   $block_2$ = {bitly, buzz, car} = {bitly, 2uzz, 0 car}

   $block_3$ = {caso, cast, z00} = {caso, 3t, 0z00}

2) Now we build a tree. Lets assume that the number of blocks isn't big so we can store in memory the tree based on first element of each block

Excution of search:

With our assuption we can expect that we will get result with going to disk only once (to get the block where we might find our string).

Search of string P = bis:

1) The track of searching in the tree is shown with the red pen. So we will first go to branch bi and then our string must be somewhere between two branches because o<s<t, so we must search the P inside the $block_1$

2) From disk we get $block_1$ = {bio, 3nic, 2t}. Scanning through the block we decompress it and check if any string is equal to our searched P. In this case we won't find one. This means there is no string P in set S

Ex 2.

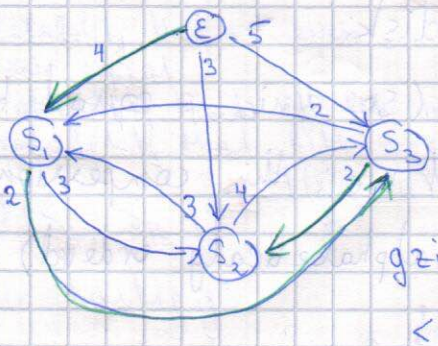
S_1 = "abaco" S_2 = "dada" S_3 = "coaba"

1) gzip($S_1 S_2 S_3$) = gzip(abacodadacoaba) = <0,0,a>;
w=3

abacodadacoaba <0,0,b>;
abacodadacoaba <2,1,c>;
abacodadacoaba <0,0,d>;
abacodadacoaba <0,0,a>;
abacodadacoaba <2,2,c>;
abacodadacoaba <0,0,d>;
abacodadacoaba <3,1,b>;
abacodadacoaba <2,1,$>;

↑ simbol of end of string because we don't need to add anything else.

2)



gzip(S_1) = gzip(abaco) = <0,0,a>; <0,0,b>; <2,1,c>; <0,0,0>

gzip(S_2) = gzip(dada) = <0,0,d>; <0,0,a>; <2,2,$>

gzip(S_3) = gzip(coaba) = <0,0,c>; <0,0,o>; <0,0,a>; <0,0,b>; <2,1,$>

gzip($S_1|S_2$) = gzip(abaco|dada) = <0,0,d>; <4,1,d>; <2,1,$>

gzip($S_1|S_3$) = gzip(abaco|coaba) = <2,2,a>; <7,2,$>

gzip($S_2|S_1$) = gzip(dada|abaco) = <1,1,b>; <2,1,c>; <0,0,0>

gzip($S_2|S_3$) = gzip(dada|coaba) = <0,0,c>; <0,0,o>; <3,1,b>; <2,1,$>

gzip($S_3|S_1$) = gzip(coaba|abaco) = <3,3,c>; <8,1,$>

gzip($S_3|S_2$) = gzip(coaba|dada) = <0,0,d>; <2,3,$>

So now we should find the "trace" in graph which will "visit" each node and will weight the less. In this case we have two routes with equal minimum weight 8, I will take randomly one: S_1, S_3, S_2.

So zdelta compression will be like this: gzip(S_1), gzip($S_1|S_3$), gzip($S_3|S_2$)