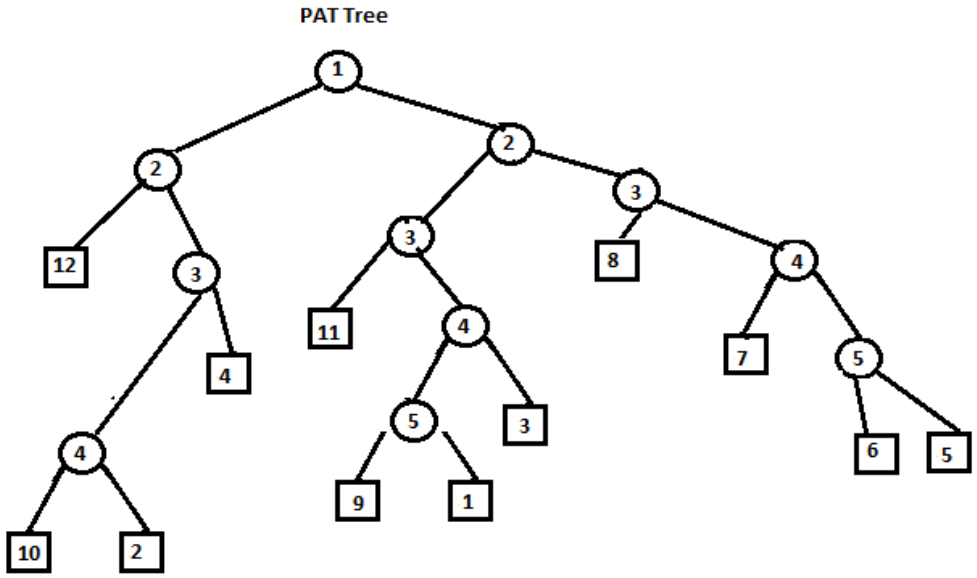
Arif Usta

Assignment 5

Q1) a)

Sistrings:

1) :10101111101000111
2) :0101111101000111
3) :101111101000111
4) :01111101000111
5) :1111101000111
6) :111101000111
7) :11101000111
8) :1101000111
9) :101000111
10) :01000111
11) :1000111
12) :000111



PAT Array : [12, 10, 2, 4, 11, 9, 1, 3, 8, 7, 6, 5]

b) In order to catch such kind of a query, proximity searching algorithm in Pat tree can be used. To do that, we first search for A and B. Assume that answer set sizes are L1 and L2 respectively with a condition L1<= L2. Sort the answer set of A. Check every sorted answer in A to see whether it satisfies the distance condition in the answer set of B.

Q2) For the reciprocal rank

r(a) = 1/ (1/3 + 1/3 + 1/2 + 1) = 0,46
r(b) = 1/ (1/2 + 1 + 1/3 + 1/2) = 0.42
r(c) = 1/ (1 + 1/2 + 1 + 1/2) = 0.33
r(d) = 1/ (1/4) = 4
r(e) = 1 / (1/4) = 4
r(f) = 1 / (1/4) = 4

The final ranked list of documents is c>b>a>d=e=f

Borda Count

r(a) = 4 + 4 + 5 + 6 = 19
r(b) = 5 + 6 + 4 + 5 = 20
r(c) = 6 + 5 + 6 + 5 = 22
r(d) = 3

r(e) 3
r(f) = 3

The final ranked list of documents is c>b>a>d=e=f

Condorcet's Method

First stage
Comparison matrix is as follows:

	a	b	c	d	e	f
a	-	2,2,0	1,3,0	4,0,0	4,0,0	4,0,0
b	2,2,0	-	1,2,1	4,0,0	4,0,0	4,0,0
c	3,1,0	2,1,1	-	4,0,0	4,0,0	4,0,0
d	0,4,0	0,4,0	0,4,0	-	1,1,2	1,1,2
e	0,4,0	0,4,0	0,4,0	1,1,2	-	1,1,2
f	0,4,0	0,4,0	0,4,0	1,1,2	1,1,2	-

Second Stage; win,lose and tie values for each document

	Win	Lose	Tie
a	3	1	1
b	3	1	1
c	5	0	0
d	0	3	2
e	0	3	2
f	0	3	2

The final ranked list of documents is c>a=b > d=e=f
Q3) Signatures are as follows with k =2;

S1 = 1100 1100
S2 = 1100 0011
S3 = 0011 1100
S4= 0000 1111
S5 = 1011 0100
S6 = 0100 1011

a) Using fixed prefix method to partition, partitions are as follows:

k=2 => we have 4 partitions.

in 00 we have S3 (0011 1100) and S4 (0000 1111)
in 01 we have only S6 (0100 1011)
in 10 we have only S5 (1011 0100)
in 11 we have S1 (1100 1100) and S2 (1100 0011)

b) Queries are as follows:

Q1 = 1110 0001
Q2 = 0110 0011
Q3 = 1100 1100
Q4 = 0011 1100

In order to find partitions, we basically AND the first 2 bits of the query with partition representative bits. If we end up having the bits same as query's one, then we need to look at that partition.

for query Q1 , we need to look at partition 11
for query Q2, we need to look at partitions 01, 11
for query Q3, we need to look at partition 11
for query Q4 , we need to look at partitions 00,01,10,11

For the sequential environment

Query 1 = 0-1
Query 2 = 1-3
Query3 = 3-4
Query4 = 4-8 TU for process,

Thus, in total we have 1 + 3 + 4 + 8 = 16 TU => Avg. TU = 16/4 = 4

For parallel environment
We can simply be able to look at partitions that we need to look directly, thus TU for each query to process is as follows:

Query1 = 1
Query2 = 2
Query3 = 3
Query4 =4 TU

In total, we have 1 + 2 + 3 + 4 = 10 TU => Avg. TU = 10/4 = 2.5
Therefore, using parallel environment against sequential environment the speed up is 4/2.5 = 1.6

Q 4) Signatures are as follows:

S1 = 1100 1100
S2 = 1100 0011
S3 = 0011 1100
S4 = 0000 1111
S5 = 1011 0100
S6 = 0100 1011

a) Extended Prefix Partitioning Method (EPP) with z = 2

z=2 => we are to find the smallest prefix having exactly 2 zeros.

from S1, key is **1100**
from S2, key is **1100**
from S3, key is **00**
from S4, key is **00**
from S5, key is **10110**
from S6, key is **0100**

Then, we have 4 unique keys , indicating that we have 4 partitions.
Partition 1 with key **1100** = S1, S2
Partition 2 with key **00** = S3, S4
Partition 3 with key **10110** = S5
Partition 4 with key **0100** = S6

b) Floating Key Partitioning Method (FKP) with k = 2

k=2 => we are to examine the 2-substrings in each signature to find the leftmost one having the least amount of 1s.

from S1, key is 11 **00** 1100
from S2, key is 11 **00** 0011
from S3, key is **00** 11 1100
from S4, key is **00** 00 1111
from S5, key is 1011 01 **00**
from S6, key is 01 **00** 1011

Then, we have 3 unique keys , indicating that we have 4 partitions.
Partition 1 with key XX **00** XXXX= S1, S2,S6
Partition 2 with key **00** XX XXXX= S3, S4
Partition 3 with key XXXX XX **00**= S5

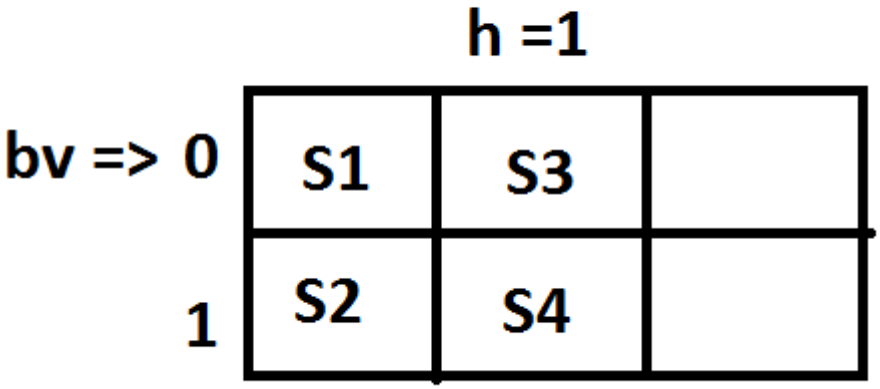
c) EPP

Query1 = 1110 0001 => No page, because there is no match for key 11100
Query2 = 0110 0011 => No page, because there is no match for key 0110
Query3 = 1100 1100 => Partition 1 with key 1100
Query4 = 0011 1100 => Partition 2 with key 00

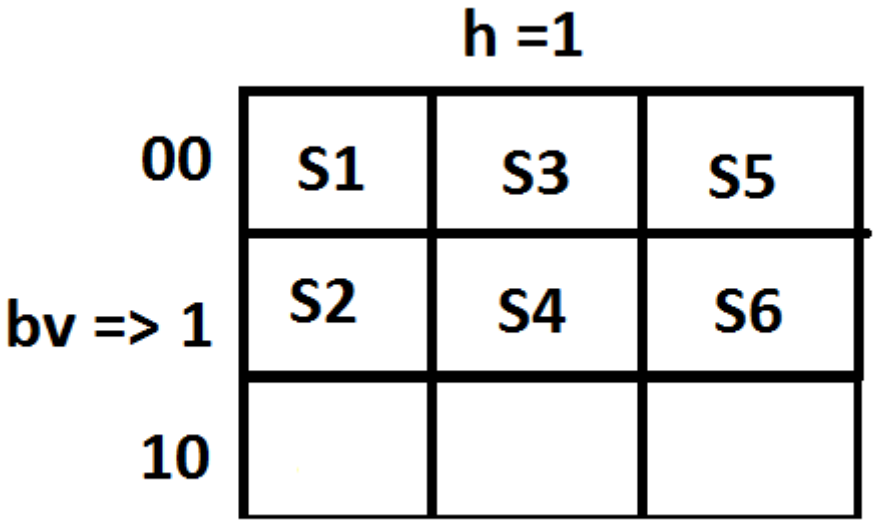
FKP

Query1 = 1110 0001 => No page, there is no match for key XXXX **00** XX
Query2 = 0110 0011 => No page, there is no match for key XXXX **00** XX
Query3 = 1100 1100 => Partition 1 with key XX **00** XXXX and Partition 3 with key XXXX XX **00**
Query4 = 0011 1100 => Partition 2 with key **00** XX XXXX and Partition 3 with key XXXX XX **00**

Q5) Since desired load factor is 2/3 and Bfr, number of signatures in a page is 3, then there will be no problem adding first 4 signatures to the structure , since 4/6 = 2/3. Linear hashing structure is as follows:



Now, we need to update the structure. To do that, first we add LF*Bfr number of signatures and update the bv value. 2/3 * 3 = 2, thus we add remaning S5 and S6, then update the bv value. The structure after update is as follows:



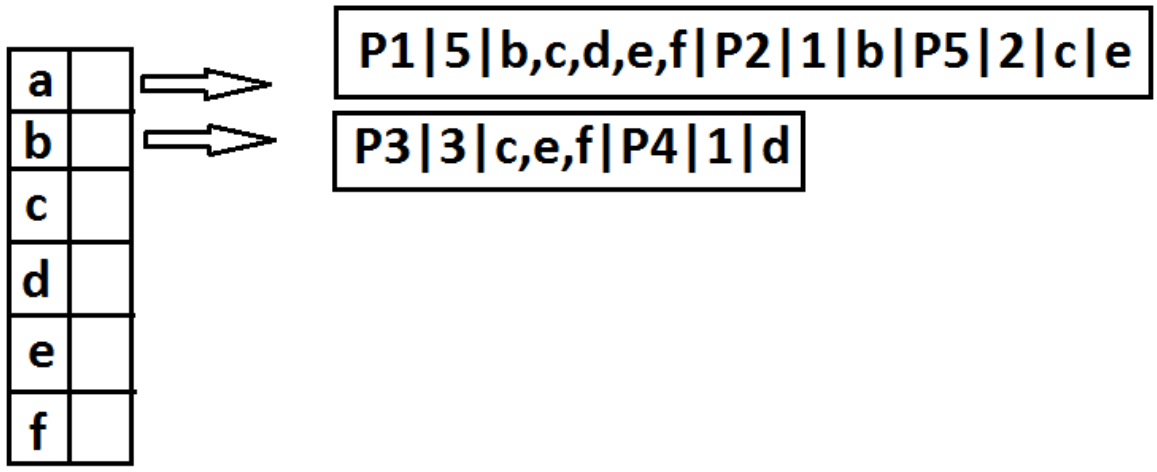
Q6) n =12

h = floor (log n) = floor (log12) = floor (3,..) = 3
bv = n- 2^h = 12 – 2^3 = 12-8 = 4
The last page at level h = 2^3 -1 = 7
#pages at level h+1 = 2*bv = 4*2 = 8
#pages at level h = 12-8 = 4

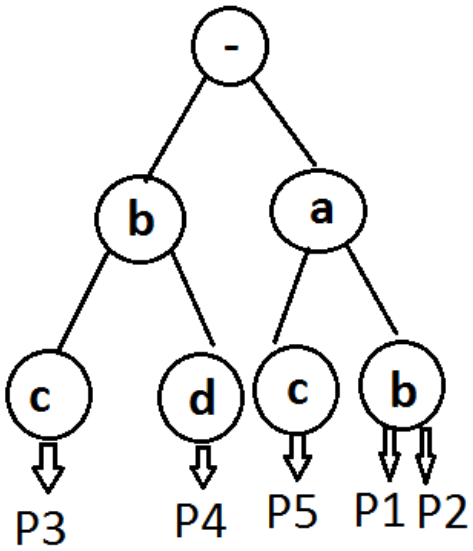
n = 120
h = floor (log n) = floor (log120) = floor (6,..) = 6
bv = n- 2^h = 120 – 2^6 = 120-64 = 56
The last page at level h = 2^6 -1 = 63
#pages at level h+1 = 2*bv = 56*2 = 112
#pages at level h = 120-112 = 8

Q7)

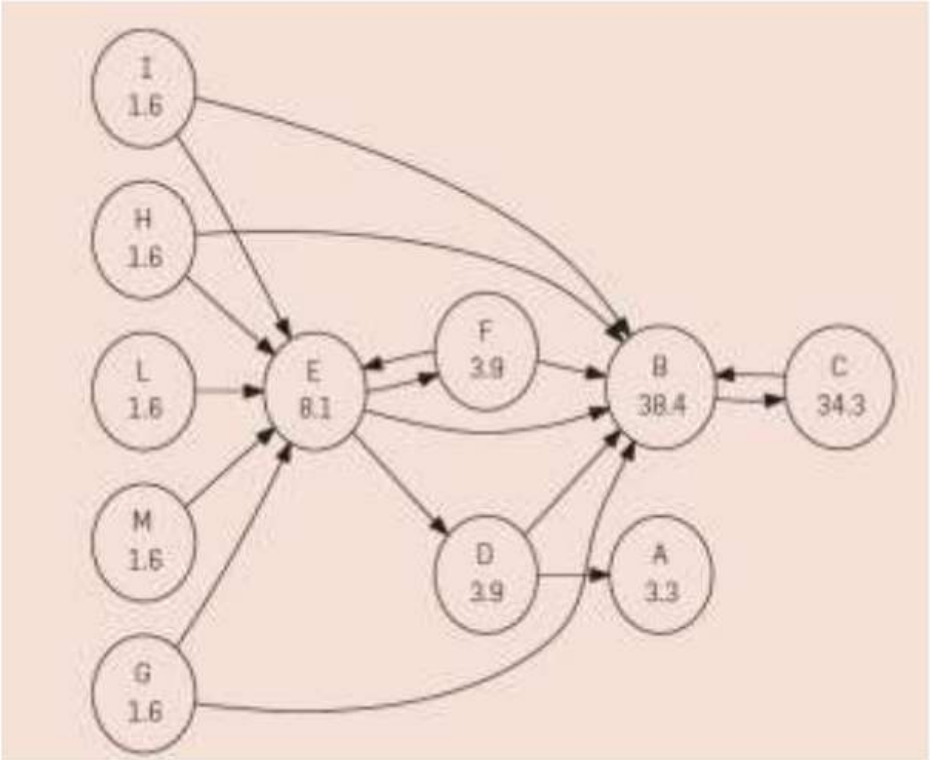
The directory and the posting lists for ranked key method are as follows:



The tree structure of the tree method is as follows:



Q8) The graph given in the assignment is as follows:



Page Rank algorithm is a recursive algortihm, that tries to find the page rank of each page by approximating in each iteration. I assume the above graph is an intermadiate page rank scores of pages. In order to find the page rank of F and D, we first need to find the page rank values of pages pointing to them. The only page pointing F and D is E. So, first we need to find the page rank of E. (This solution needs some corrections.)

PR(E) = (1-a) + a(PR(F)/2 + PR(I)/2 + PR(H)/2 + PR(L)/1 + PR(M)/1 + PR(G)/2)

= 0.15 + 0.85(3.9*0.5 + 1.6*0.5 + 1.6*0.5 + 1.6 + 1.6 + 1.6*0.5)

= 0.15 + 0.85(1.95 + 0.8 + 0.8 + 1.6 + 1.6 + 0.8)

= 0.15 + 0.85(7.55)

= 0.15 + 6.41 = 6.56

PR(F) = (1-a) + a(PR(E)/3)

= 0.15 + 0.85(6.56/3)

= 0.15 + 0.85*2.18

=0.15 + 1.86 = 2.008

PR(D) = PR(F) , since each node has only incoming link from E, thus calculations will be same.