Boston University
CAS CS 562: Advanced Database Applications

Homework #2
Fall 2020

Due Date: April 2, 2020 at 11:59PM, please submit a PDF file using Gradescope.

Problem 1

Consider the following temporal evolution of a set of records. The first value is the time of the operation, the second value the object id, the third value is the value that is used for indexing and searching, and the last value is the operation. The operation can be either an insertion (i) or a deletion (d).

(time, oid, value, operation)

(1, 1, 34, i)	(11, 1, 34, d)	(21, 21, 43, i)	(31, 20, 122, d)
(2, 2, 23, i)	(12, 12, 27, i)	(22, 22, 32, i)	(32, 9, 29, d)
(3, 3, 4, i)	(13, 2, 23, d)	(23, 23, 44, i)	(33, 33, 81, i)
(4, 4, 56, i)	(14, 14, 55, i)	(24, 10, 6, d)	(34, 34, 21, i)
(5, 5, 7, i)	(15, 5, 7, d)	(25, 14, 55, d)	(35, 35, 57, i)
(6, 6, 24, i)	(16, 6, 24, d)	(26, 26, 72, i)	(36, 36, 16, i)
(7, 7, 3, i)	(17, 7, 3, d)	(27, 27, 13, i)	(37, 27, 13, d)
(8, 8, 49, i)	(18, 18, 19, i)	(28, 18, 19, d)	(38, 30, 50, d)
(9, 9, 29, i)	(19, 19, 8, i)	(29, 19, 8, d)	(39, 12, 27, d)
(10, 10, 6, i)	(20, 20, 122, i)	(30, 30, 50, i)	(40, 40, 38, i)

(a) Create a Snapshot Index on the evolution given above using pages with capacity b=4 entries (number of entries per page(block)), and utilization factor u = 0.5. Show the index after each 10 operations (after 10, 20,30, and 40 operations).

(b) Create a MVB-Tree on the **value attribute** of the above evolution using pages with capacity b=6 entries (number of entries per page(block)), d =2, and $\varepsilon = 0.5$. Show the index after each 10 operations (i.e., after the 10th, 20th, 30th, and 40th operations).

(c) Show how to answer to the following queries using the indices that you created in (a) and (b). Give the pages that you need to retrieve to answer these queries:

1) Timeslice query: Give all the alive records at time tq1 = 30.

2) Range timeslice query: Give all the alive records at time tq3 = 15 that have values between 25 and 35.

Problem 1

(a) <[t_1, t_2], oid, value>

Time instant 10:

Block A

SR record
<1, *, 1, 34>
<2, *, 2, 23>
<3, *, 3, 4>
<4, *, 4, 56>

Block B

SR record
<5, *, 5, 7>
<6, *, 6, 24>
<7, *, 7, 3>
<8, *, 8, 49>

Block C

SR record
<9, *, 9, 29>
<10, *, 10, 6>

Time instant 20:

Block A

SR record
<1, 11, 1, 34>
<2, 13, 2, 23>
<3, *, 3, 4>
<4, *, 4, 56>

Block B

SR record
<5, 15, 5, 7>
<6, 16, 6, 24>
<7, 17, 7, 3>
<8, 17, 8, 49>

Block C

SR record
<9, *, 9, 29>
<10, *, 10, 6>
<12, *, 12, 27>
<14, *, 14, 55>

Block D

SR record
<17, *, 8, 49>
<18, *, 18, 19>
<19, *, 19, 8>
<20, *, 20, 122>

Time instant 30:

Block A

SR record

<1, 11, 1, 34>
<2, 13, 2, 23>
<3, *, 3, 4>
<4, *, 4, 56>

Block B

SR record

<5, 15, 5, 7>
<6, 16, 6, 24>
<7, 17, 7, 3>
<8, 17, 8, 49>

Block C

SR record

<9, *, 9, 29>
<10, 24, 10, 6>
<12, *, 12, 27>
<14, 25, 14, 55>

Block D

SR record

<17, *, 8, 49>
<18, 28, 18, 19>
<19, 29, 19, 8>
<20, *, 20, 122>

Block E

SR record

<21, *, 21, 43>
<22, *, 22, 32>
<25, *, 23, 44>
<26, *, 26, 72>

Block F

SR record

<27, *, 27, 13>
<30, *, 30, 50>

Time instant 40:

Block A

SR record

<1, 11, 1, 34>
<2, 13, 2, 23>
<3, *, 3, 4>
<4, *, 4, 56>

Block B

SR record

<5, 15, 5, 7>
<6, 16, 6, 24>
<7, 17, 7, 3>
<8, 17, 8, 49>

Block C

SR record

<9, 32, 9, 29>
<10, 24, 10, 6>
<12, 32, 12, 27>
<14, 25, 14, 55>

Block D

SR record

<17, 31, 8, 49>

<18, 28, 18, 19>

<19, 29, 19, 8>

<20, 31, 20, 122>

Block E

SR record

<21, ∞, 21, 43>

<22, ∞, 22, 32>

<23, ∞, 23, 44>

<26, ∞, 26, 72>

Block F

SR record

<27, 37, 27, 13>

<30, 38, 30, 50>

<31, 39, 8, 49>

<32, 39, 12, 27>

Block G

SR record

<33, ∞, 33, 81>

<34, ∞, 34, 21>

<35, ∞, 35, 57>

<36, ∞, 36, 16>

Block H

SR record

<39, ∞, 8, 49>

<40, ∞, 40, 36>

Access forest:

time instant 10:

<S, [0, now]> ↔ <A, [1, now]> ↔ <B, [5, now]> ↔ <C, [9, now]>

AT

time	bid
0	S
1	A
5	B
9	C

time instant 20:

AT

time	bid
0	S
1	A
5	B
9	C
17	D

<S, [0, now]> ↔ <A, [1, now]> ↔ <C, [9, now]> ↔ <D, [17, now]>

↕

<B, [5, 17]>

time instant 30:

AT

time	bid	time	bid
0	S	17	D
1	A	21	E
5	B	27	F
9	C		

<S, [0, now]> ↔ <A, [1, now]> ↔ <C, [9, now]> ↔ <D, [17, now]> ↔ <E, [21, now]> ↔ <F, [27, now]>

↕

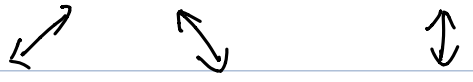
<B, [5, 17]>

time instant 40:

AT

time	bid	time	bid	time	bid
0	S	17	D	32	F
1	A	21	E	33	G
5	B	27	F	39	H
9	C	31	F		

<S, [0, now]> ⟷ <A, [1, now]> ⟷ <E, [21, now]> ⟷ <G, [33, now]> ⟷ <H, [39, now]>



<B, [5, 17]> ⟷ <C, [9, 32]> <F, [27, 39]>

<D, [17, 31]>

(c) Snapshot:

All the alive records at time 30:

4, 13, 27, 29, 32, 43, 44, 49, 50, 56, 72, 122

Blocks: F, E, A, C, D

All the alive records at time 15 that have values between 25 and 35:

27, 29

Blocks: C, A, B

Problem 1

(b) mvbt $b = 6$, $d = 2$, $\varepsilon = 0.5$

$b = kd \Rightarrow k = 3$

Weak version condition : $d = 2$

Strong version condition : $(1+\varepsilon)d \le N \le (k-\varepsilon)d$

where N is the number of current entries in a block.

Here, $3 \le N \le 5$

data : <value, in_version, del_version>

router : <key, in-version, del-version, reference>

Time instant 6:

A

<34,	1,	* >
<23,	2,	* >
<4,	3,	* >
<56,	4,	* >
<7,	5,	* >
<24,	6,	* >

Array

A[1, *]

Time instant 7:

R

<3 ,	7,	* ,	B>
<24 ,	7,	* ,	C>

Array

A[1, 7]

R[7, *]

A

<34,	1,	* >
<23,	2,	* >
<4,	3,	* >
<56,	4,	* >
<7,	5,	* >
<24,	6,	* >

B

<3,	7 ,	*>
<4,	3 ,	*>
<7,	5 ,	*>
<23,	2,	*>

C

<24,	6 ,	*>
<34,	1,	*>
<56,	4 ,	*>

Time instant 10:

R

Array A[1,7]

R[7,*]

R
<3, 7, *, B>
<24, 7, *, C>

A

A
<34, 1, *>
<23, 2, *>
<4, 3, *>
<56, 4, *>
<7, 5, *>
<24, 6, *>

B

B
<3, 7, *>
<4, 3, *>
<7, 5, *>
<23, 2, *>
<6, 10, *>

C

C
<24, 6, *>
<34, 1, *>
<56, 4, *>
<49, 8, *>
<29, 9, *>

Time instant 13

R

R
<3, 7, *, B>
<24, 7, *, C>

A

A
<34, 1, *>
<23, 2, *>
<4, 3, *>
<56, 4, *>
<7, 5, *>
<24, 6, *>

B

B
<3, 7, *>
<4, 3, *>
<7, 5, *>
<23, 2, 13>
<6, 10, *>

C

C
<24, 6, *>
<34, 1, 11>
<56, 4, *>
<49, 8, *>
<29, 9, *>
<27, 12, *>

Time instant 18

R

<3, 7, *, B>
<24, 7, 14, C>
<24, 14, *, D>
<49, 14, *, E>

A

<34, 1, *>
<23, 2, *>
<4, 3, *>
<56, 4, *>
<7, 5, *>
<24, 6, *>

B

<3, 7, 17>
<4, 3, *>
<7, 5, 15>
<23, 2, 13>
<6, 10, *>
<19, 18, *>

C

<24, 6, *>
<34, 1, 11>
<56, 4, *>
<49, 8, *>
<29, 9, *>
<27, 12, *>

D

<24, 6, 16>
<27, 12, *>
<29, 9, *>

E

<49, 8, *>
<55, 14, *>
<56, 4, *>

Time instant 20

R

<3, 7, 19, B>
<24, 7, 14, C>
<24, 14, *, D>
<49, 14, *, E>
<4, 19, *, F>

A

<34, 1, *>
<23, 2, *>
<4, 3, *>
<56, 4, *>
<7, 5, *>
<24, 6, *>

B

<3, 7, 17>
<4, 3, *>
<7, 5, 15>
<23, 2, 13>
<6, 10, *>
<19, 18, *>

C

<24, 6, *>
<34, 1, 11>
<56, 4, *>
<49, 8, *>
<29, 9, *>
<27, 12, *>

D

<24, 6, 16>
<27, 12, *>
<29, 9, *>

E

<49, 8, *>
<55, 14, *>
<56, 4, *>
<122, 20, *>

F

<4, 3, *>
<6, 10, *>
<8, 19, *>
<19, 18, *>

Time instant 30

R

<3, 7, 19, B>
<24, 7, 14, C>
<24, 14, *, D>
<49, 14, *, E>
<4, 19, *, F>

A

<34, 1, *>
<23, 2, *>
<4, 3, *>
<56, 4, *>
<7, 5, *>
<24, 6, *>

B

<3, 7, 17>
<4, 3, *>
<7, 5, 15>
<23, 2, 13>
<6, 10, *>
<19, 18, *>

C

<24, 6, *>
<34, 1, 11>
<56, 4, *>
<49, 8, *>
<29, 9, *>
<27, 12, *>

D

<24, 6, 16>
<27, 12, *>
<29, 9, *>
<43, 21, *>
<32, 22, *>
<44, 23, *>

E

<49, 8, *>
<55, 14, 25>
<56, 4, *>
<122, 20, *>
<72, 26, *>
<50, 30, *>

F

<4, 3, *>
<6, 10, 24>
<8, 19, 29>
<19, 18, 28>
<13, 27, *>

Time instant 40

R_1

<3, 7, 19, B>
<24, 7, 14, C>
<24, 14, * D>
<49, 14, 33, E>
<4, 19, 36, F>
<49, 33 *, G>

R_2

<24, 14, 40, D>
<49, 33, *, G>
<4, 36, *, H>
<32, 40, *, I>

A

<34, 1, *>
<23, 2, *>
<4, 3, *>
<56, 4, *>
<7, 5, *>
<24, 6, *>

B

<3, 7, 17>
<4, 3, *>
<7, 5, 15>
<23, 2, 13>
<6, 10, *>
<19, 18, *>

C

<24, 6, *>
<34, 1, 11>
<56, 4, *>
<49, 8, *>
<29, 9, *>
<27, 12, *>

E

<49, 8, *>
<55, 14, 25>
<56, 4, *>
<122, 20, 31>
<72, 26, *>
<50, 30, *>

D

<24, 6, 16>
<27, 12, 39>
<29, 9, 32>
<43, 21, *>
<32, 22, *>
<44, 23, *>

F

<4, 3, *>
<6, 10, 24>
<8, 19, 29>
<19, 18, 28>
<13, 27, *>
<21, 34, *>

G

<49, 8, *>
<50, 30, 38>
<56, 4, *>
<72, 26, *>
<81, 33, *>
<57, 35, *>

H

<4, 3, *>
<13, 27, 37>
<16, 36, *>
<21, 34, *>

I

<32, 22, *>
<38, 40, *>
<43, 21, *>
<44, 23, *>

(c) MVBT:

All the alive records at time 30:

4, 13, 27, 29, 32, 43, 44, 49, 50, 56, 72, 122

Blocks: D, E, F

All the alive records at time 15 that have values between 25 and 35:

27, 29

Blocks: B, D

Problem 2

Consider the following Time-Interval query: Given a time period T = [t_s, t_e], find all the records that have lifespan that intersects T. That is, find all the records that were alive at least one time instant during the period T. Provide an algorithm and explain what is the cost of this algorithm in terms of number of I/Os. Hint: Read the Snapshot Index paper.

Problem 3

Consider the following 2-dimensional time series:
A = [(2,3), (3,4), (4,5), (4,4), (5,5)] and B=[(2,2),(3,3),(15,2),(4,3),(4,5),(6,8)].

Compute the DTW and the LCSS distances between the two time series. For LCSS use $\varepsilon = 1.5$. To compute the distance between two 2-d points x=(x1, x2) and y=(y1, y2) you can use the L1 distance (L1(x, y) = |x1-y1| +|x2-y2|).

Problem 4

(a) Consider the GEMINI approach to index multimedia and time series data. What happens when the Lower bounding Lemma of the transformation to a lower dimensional space or feature extraction does not hold? Namely, when the $D_{feature}(F(x), F(y)) <= D(x, y)$ does not hold.

(b) Assume that you can prove that $D_{feature}(F(x), F(y)) <= 2\ D(x, y)$. How you will modify the GEMINI approach for RangeQuery and K_NNQuery algorithms in order to guarantee that the complete (correct) answer will be provided at the end?

Problem 2

```
resultSet = {}  // the blocks need to be accessed

main() {
    call timeIntervalQuery(ts, te);
    return resultSet;
}

timeIntervalQuery(ts, te) {
    Use array AT to find the time of the last object Y at time te.
    Starting from Y go up recursively and then we get block P.
    call searchHelperFunction(ts, P);
}

searchHelperFunction(ts, block) {

    if( block == null ) {
        return ;
    }

    if( block.te >= ts ) {
        // report all the records in this block that were alive during [ts, te]
        resultSet.add(block);
    }
    else {
        return ;
    }

    searchHelperFunction(ts, block->rightmost_child);
    searchHelperFunction(ts, block->left_sibling);
}
```

I/O complexity: $O(\log_b n + N/b)$, where n is the number of all records and N is the number of all real-world objects.

Problem 3

DTW

(5,5)	A_5	19	11	20	20	18	22
(4,4)	A_4	13	7	18	17	18	24
(4,5)	A_3	9	5	16	18	18	23
(3,4)	A_2	4	2	16	18	20	27
(2,3)	A_1	1	2	16	18	22	31

B_1	B_2	B_3	B_4	B_5	B_6
(2,2)	(3,3)	(15,2)	(4,3)	(4,5)	(6,8)

DTW = 22

LCSS $\varepsilon = 1.5$

(5,5)	A_5	0	1	2	2	3	4	4
(4,4)	A_4	0	1	2	2	3	3	3
(4,5)	A_3	0	1	2	2	2	3	3
(3,4)	A_2	0	1	2	2	2	2	2
(2,3)	A_1	0	1	1	1	1	1	1
	A_0	0	0	0	0	0	0	0
		B_0	B_1	B_2	B_3	B_4	B_5	B_6
			(2,2)	(3,3)	(15,2)	(4,3)	(4,5)	(6,8)

LCSS = 4

Problem 4

(a) GEMINI will not work if Lower bounding Lemma does not hold.

When we do a transformation / dimension reduction of the original feature space, the distance between two sequences should be shorter.

If $D_{feature}(F(x), F(y)) <= D(x, y)$ does not hold, that means the transformation makes no sense. Thus, Fourier / wavelet transformation were bad methods.

(b) $D_{feature}(F(x), F(y)) \leq 2D(x, y)$ means that after reduction, there are more features, which means the distance may become longer.

① In the original RangeQuery, we return the sequences that satisfy $D_{feature}(F(Q), F(S)) \leq \varepsilon$

It should be modified as $D_{feature}(F(Q), F(S)) \leq 2\varepsilon$

② In k_NNQuery, we issue a RangeQuery $(Q, 2\varepsilon_{max})$.