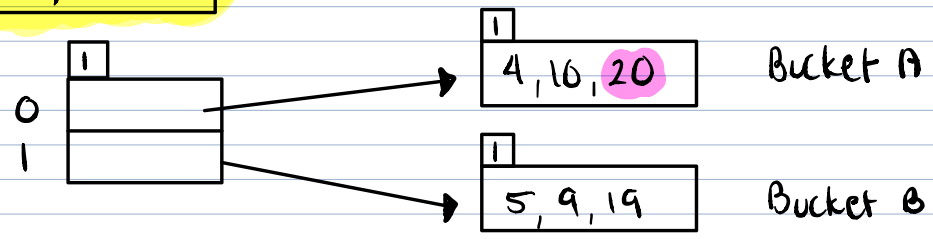
Jerry Perez

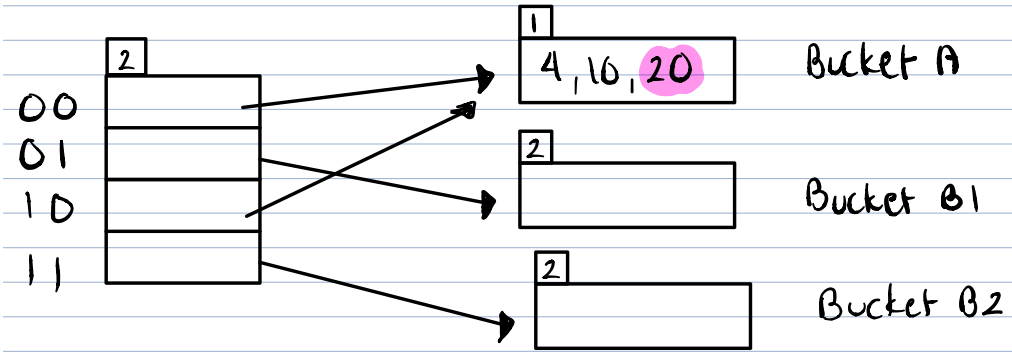
Problem 1



Binary conversions

	32	16	8	4	2	1	
insert 20	0	1	0	1	0	0	, Bucket A
insert 33	1	0	0	0	0	1	, Bucket B - split

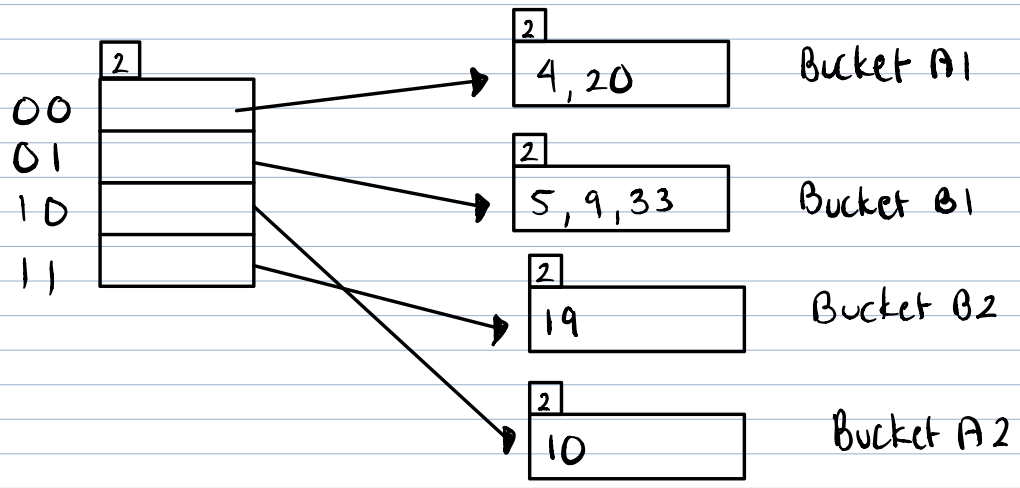
5, 9, 19



Binary conversions

	32	16	8	4	2	1	
check 5	0	0	0	1	0	0	, Bucket A - split

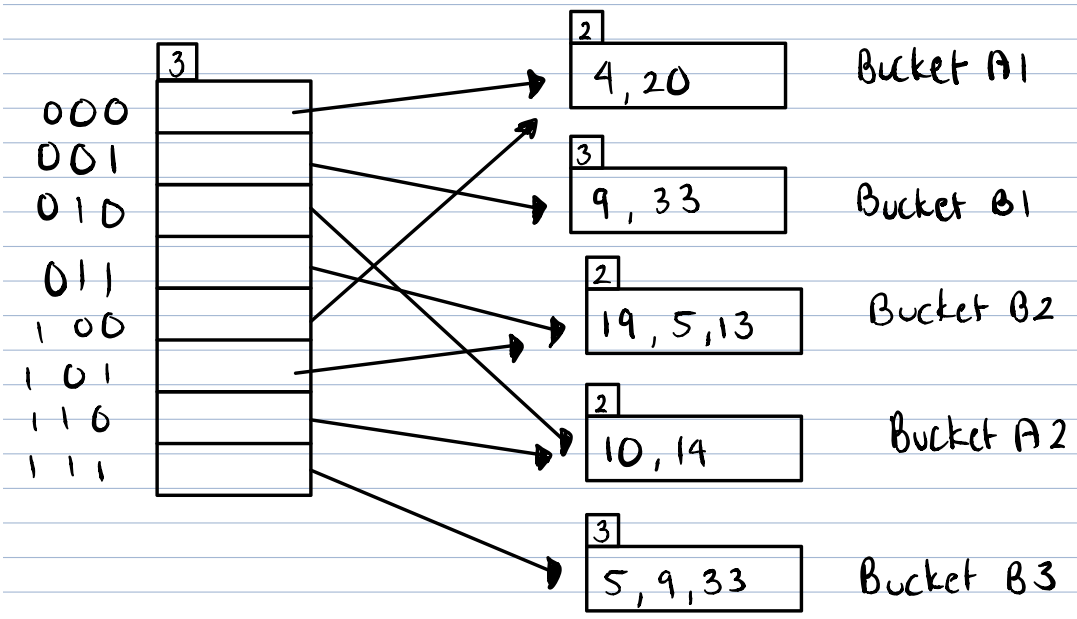
4, 10, 20



Binary conversions

	32	16	8	4	2	1	
check 4	0	0	0	1	0	0	, Bucket A1
Check 10	0	0	1	0	1	0	, Bucket A2
check 20	0	1	0	1	0	0	, Bucket A1
check 5	0	0	0	1	0	1	, Bucket B1
check 9	0	0	1	0	0	1	, Bucket B1
check 19	0	1	0	0	1	1	, Bucket B2
insert 33	1	0	0	0	0	1	, Bucket B1
insert 13	0	0	1	1	0	1	, Bucket B1 - split

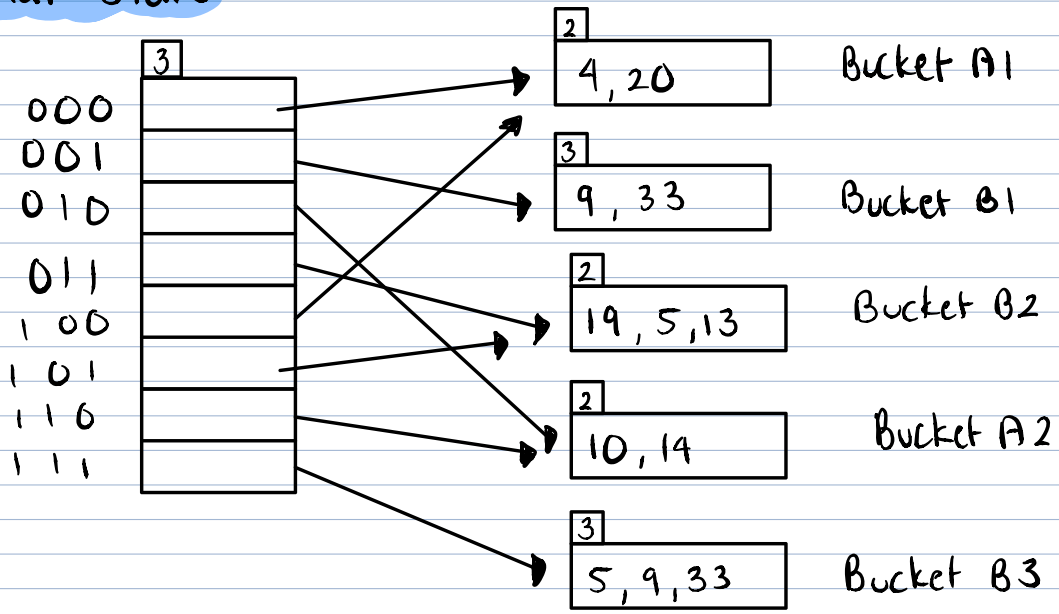
5, 9, 33



Binary conversions

	32	16	8	4	2	1	
check 5	0	0	0	1	0	1	, Bucket B2
check 9	0	0	1	0	0	1	, Bucket B1
check 33	1	0	0	0	0	1	, Bucket B1
insert 13	0	0	1	1	0	1	, Bucket B2
insert 14	0	0	1	1	1	0	, Bucket A2

Final State



Problem 2

1. a. Non-blocking since it can output tuples as it processes the input.

   b. Non-blocking since column x is sorted... once we find a varying value of x, it can output all tuples of the previous values since they make up a group.

   c. Blocking since it needs to locate elements in a group before outputting.

   d. Blocking since it needs to process all tuples before outputting the sorted list.

   e. Non-blocking since a B-tree's leaves are sorted which would mean the tuples can just be outputted as they are read.

   f. Blocking since sorting must be done, then the join.

   e. Non-blocking since we could just output the tuples as we see them.

2.

a. Can be done in one pass with the assumption that R's distinct tuples fit in M-1 buffers, or 199 buffers.

b. can be done in one pass following the constraint of the largest group fitting in 199 buffers.

c. can be done in one pass as long as relation R fits in 199 buffers.

d. cannot be done in one pass and the I/O cost would be 2 * B(R) for the first phase and B(R) for the second phase. This would be a total of 3 B(R) → 3*1000 = 3000 I/Os. Two pass external sort reads M blocks at once, then proceeds to sort, writes to disk as the first run (or phase) and finally merges the runs generating a sorted output.

e. can be done in one pass with a total of 2,070 I/O's. It would take 2000 I/O's to read and write the sorted relation R, and 70 additional I/O's to read the index.

f. cannot be done in one pass and in fact uses an efficient two pass algorithm with 3(B(R) + B(S)) as an I/O cost. It would result in a total of: 3(1,000 + 150) = 3,450 I/O's. The first phase sorts relations R and S, and the second phase merges and joins those sorted relations.

g. can be done in one pass since relation S fits in M-1, or 199 buffers.

Problem 3

1. $Q = T(R1) / V(R1, a)$

$= 400 / 50$

$= \boxed{8}$

2. $Q = T(R1) / V(R1, a) * 41$

$= 8 * 41$

$= \boxed{328}$

3. $Q = [T(R1) / V(R1, a) * 41] * (1/50)$

$= 328 * 1/50$

$= \boxed{7}$

4. $Q = T(R1) * T(R2) / \max(V(R1,b), V(R2,b))$

$= 100 * 500 / 50$

$= \boxed{1,000}$

5. $Q = [T(R1) * T(R2) / \max(V(R1,b), V(R2,b))]$
$* T(R3) / \max(V(R2,c), V(R3,c))$

$= 1,000 * T(R3) / \max(V(R2,c), V(R3,c))$

$= 1,000 * 1,000 / 100$

$= \boxed{10,000}$

6. $Q = [T(R1) / V(R1,a) * 1] * [T(R2) / \max(V(R1,b), (R2,b))]$
$* T(R3) / \max(V(R2,c), (R3,c))$

$= 328 * 500 / 50 \ldots$

$= 328 * 10$

$= 3,280 * 1,000/100$

$= 3,280 * 10$

$= \boxed{32,800}$