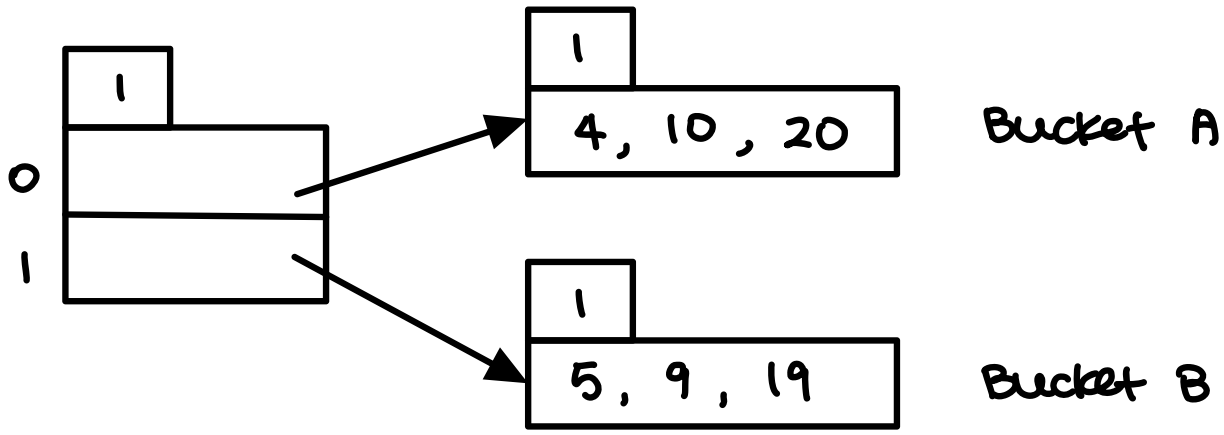
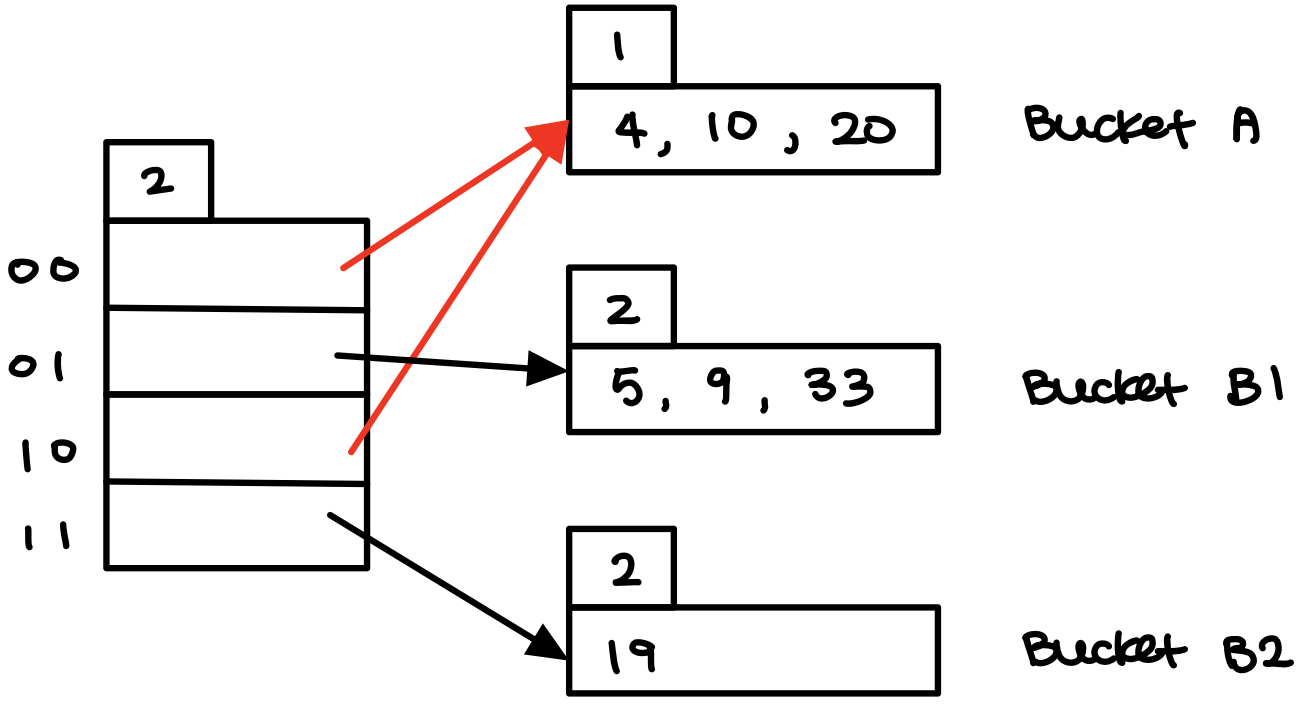
Problem 1

Megan Sin

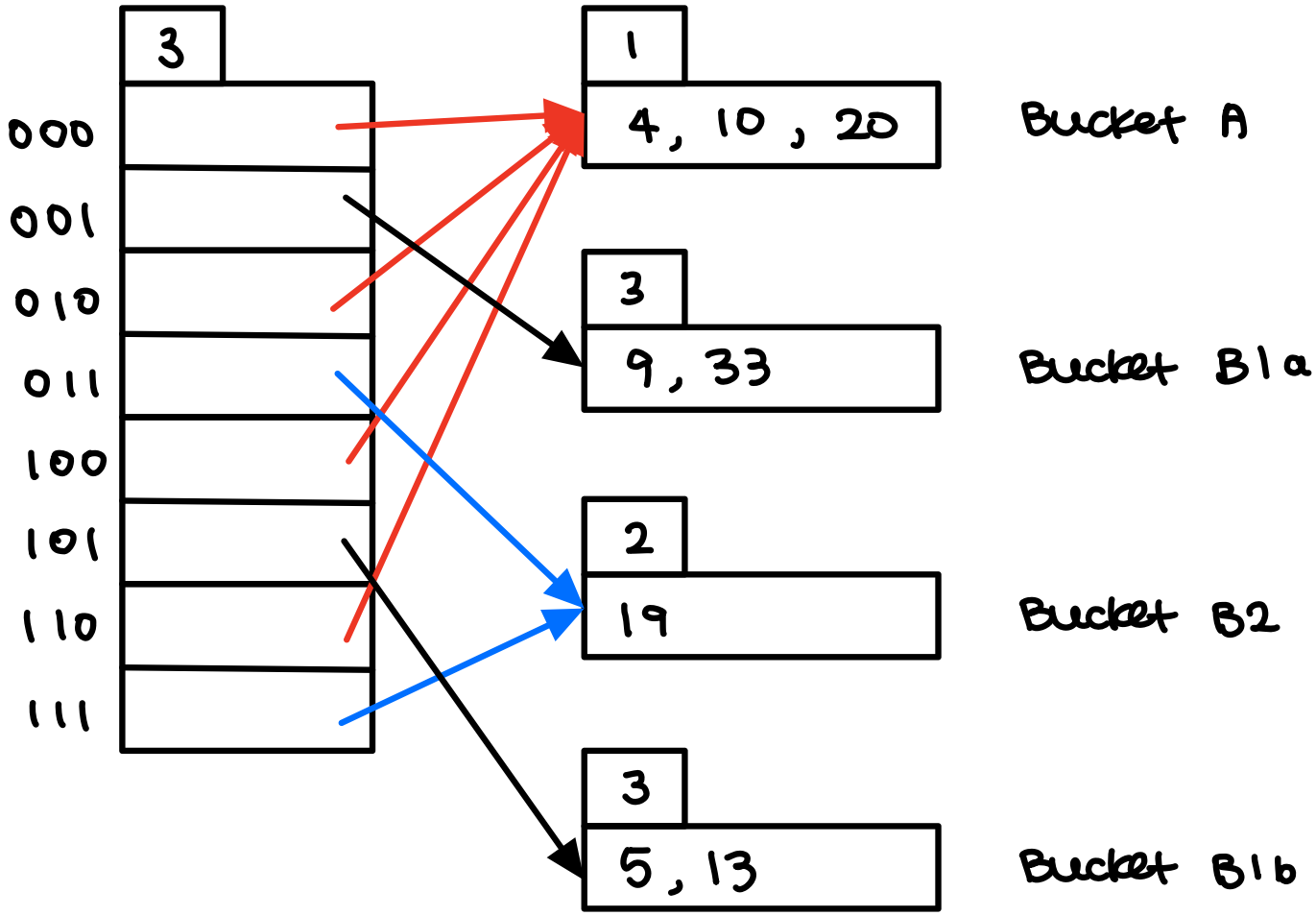
Insert 20:



Insert 33: Bucket B is full → split

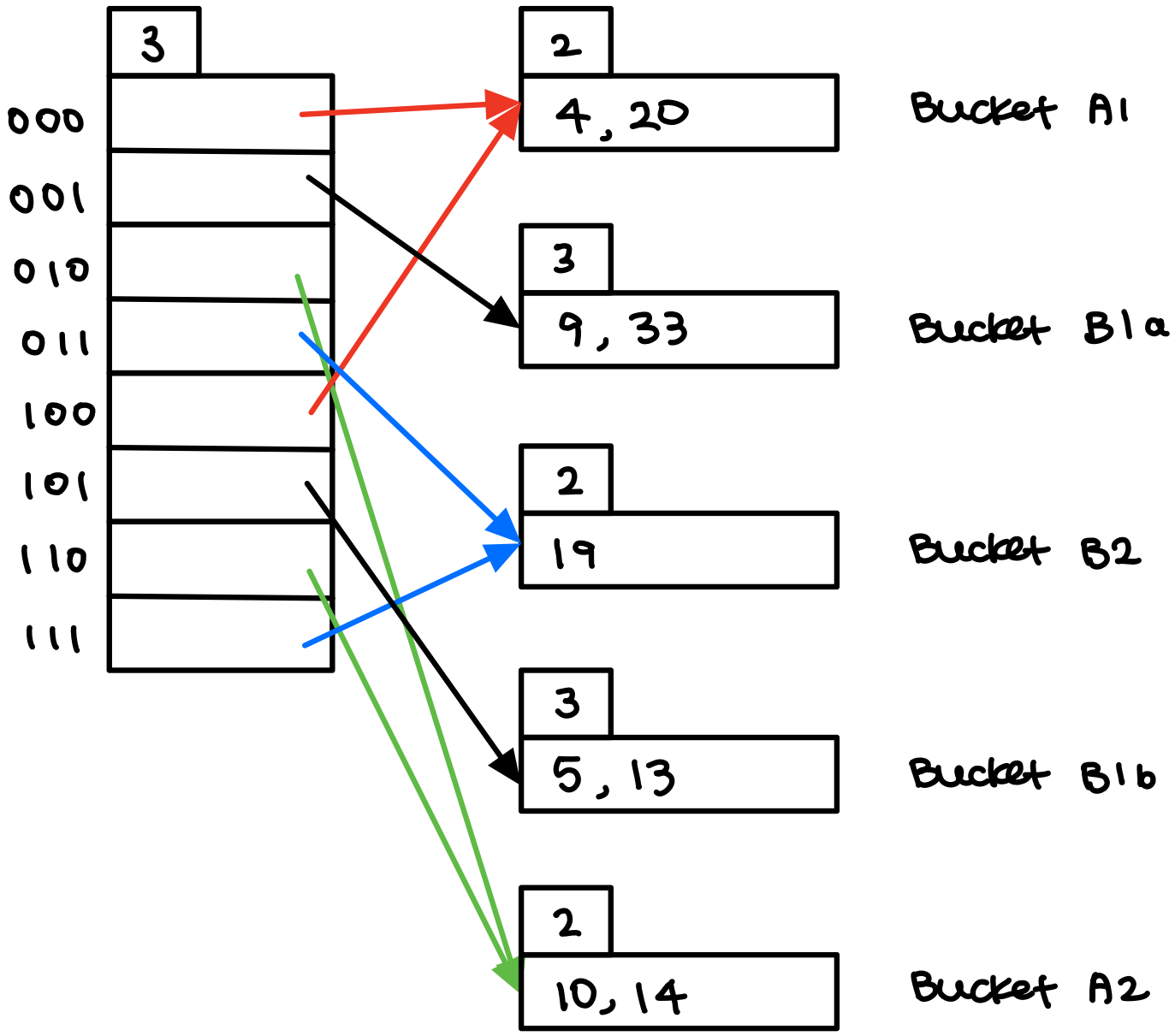


Insert 13 : Bucket B1 is full → split



Insert 14 : Bucket A is full → split

↳ no need to double directory as local depth (1) < global depth (3)



Problem 2

1a) Non-blocking
- ↳ first occurance of groups of identical tuples can be reported as soon as it's found

1b) Non-blocking
- ↳ since R is sorted on column X, as value of X changes → tuples within each group can be detected → previous group can be reported

1c) Blocking
- ↳ no group can be reported until all input tuples processed

1d) Blocking
- ↳ no tuples can be reported until all input tuples processed

1e) Non-blocking
- ↳ since we can use a B-tree index that exists on R.X to read tuples, can follow the order of the B-tree index and start reporting tuples as they appear in the B-tree's leaves

1f) Non-blocking
- ↳ can start producing joined tuples before all input from R and S processed

1g) Non-blocking
↳ bag union produces total tuples from both R and S → tuples can be streamed and reported from R and then S

2a) · can be done in one pass
· size constraint: 199 blocks

2b) · can be done in one pass
· size constraint: none

2c) · can be done in one pass
· size constraint: 199 blocks

2d) · cannot be done in one pass
· need two pass sorting algorithm
· I/O cost: $3B(R) = 3(1000) = 3000$ I/Os

2e) · can be done in one pass
↳ by following entries in leaf nodes in the index
↳ size constraint n/a

2f) · can be done in one pass (S fits in memory)
↳ since S fits in memory → no size constraint
· 151 blocks used for input

2g) · can be done in one pass
· 1 buffer in memory used to process R and S one block at a time