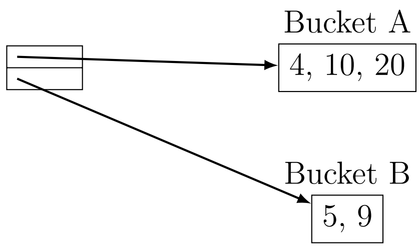
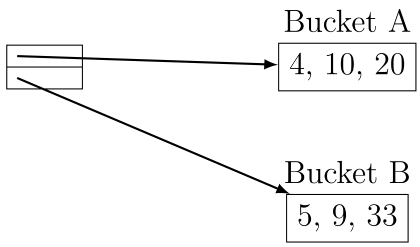
Problem 1

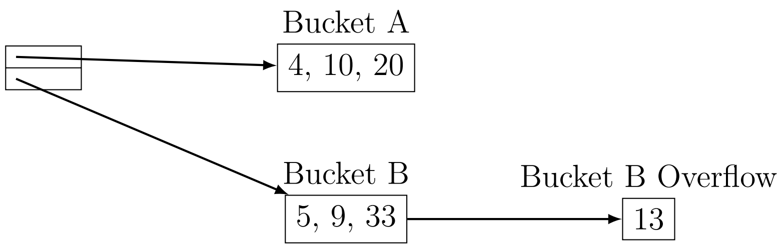
After inserting 20:



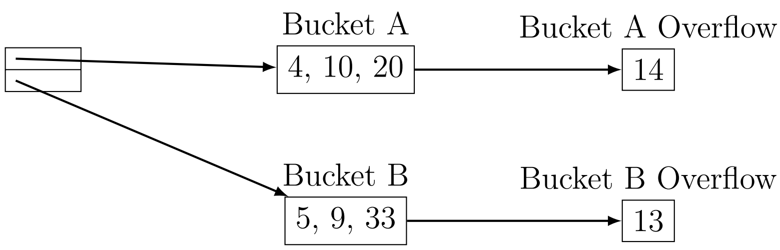
After inserting 33:



After inserting 13:



After inserting 14:



Problem 2

1

(a) Non-blocking, because it can output tuples as it processes inputs, and just won't output them again if it has encountered the same tuple before.

(b) Non-blocking. If column X is sorted, once it finds a different value of X it can output all tuples of the previous value because they make up an entire group.

(c) Blocking, because it needs to find all elements in a specific group before it can start outputing them.

(d) Blocking, because it needs to process all tuples in R before it can output the sorted list of tuples.

(e) Non-blocking. Since the leaves of the B-tree are already sorted it can just output them in order as it reads them in.

(f) Blocking, because it must first sort R and S, and then merge join them.

(g) Non-blocking, it can output the resulting tuples as it reads in the input.

2

(a) Can be done in one pass assuming the distinct tuples of R fit in 199 buffers.

(b) Can be done in one pass as long as the biggest group can fit in 199 buffers.

(c) Can be done in one pass as long as R can fit in 199 buffers.

(d) Cannot be done in one pass. A two-pass external sort reads M blocks at a time, sorts them, and writes them to the disk as the first run, then merges the runs to produce a sorted output. The I/O cost will be $2 \times B(R)$ for the first pass and $B(R)$ for the second pass, for a total of $3B(R) = 3 \times 1000 = 3000$ I/Os.

(e) Can be done in one pass, with 70 I/Os to read the index, plus 2000 I/Os to read and write the sorted relation R, for a total of 2070 I/Os.

(f) Cannot be done in one pass. Phase one can sort R and S, while phase two merges and joins the sorted relations R and S. The I/O cost is $2 \times B(R) + 2 \times B(S)$ for phase one sorting, plus $B(R) + B(S)$ for the merge and join phase two, for a total cost of $3(B(R) + B(S)) = 3(1000 + 150) = 3450$ I/Os.

(g) Can be done in one pass because S can fit in 199 buffers.

Problem 3

1. $Q = T(R1)/V(R1_a) = 400/50 = 8$
2. $Q = \left(\frac{T(R1)}{V(R1_a)}\right) \times 41 = 328$
3. $Q = 328 \times (1/50) = 7$
4. $Q = T(R1) \times T(R2) \div \max(V(R1_b), V(R2_b)) = 400 \times 500 \div \max(50, 40) = 4000$
5. $Q = 4000 \times T(R3) \div \max(V(R2_c), V(R3_c)) = 4000 \times 1000 \div 100 = 40000$
6. $Q = (328 \bowtie R2) \bowtie R3 = (328 \times 500 \div 50) \bowtie R3 = 3280 \bowtie R3 = 3280 \times 1000 \div 100 = 32800$

Problem 4

1. a. R2, because it's smaller

   b. $B(R2) + \frac{B(R2)}{M-1} \times B(R1) = 200 + \frac{200}{51} \times 1000 = 4122$ IOs

   c. $B(R1) + \frac{B(R1)}{M-1} \times B(R2) = 1000 + \frac{1000}{51} \times 200 = 4922$ IOs

2. a. $3(B(R1) + B(R2)) = 3(1000 + 200) = 3600$

   b. $B(R1) + B(R2) \leq M^2$
   $1000 + 200 \leq M^2$
   $M >= 34$
   Minimum number of buffers necessary is 34

3. a. $3(B(R1) + B(R2)) = 3(1000 + 200) = 3600$

   b. $B(R2) \leq M^2$
   $200 \leq M^2$
   $M >= 14$
   Minimum number of buffers is 14

4. a. Assuming R2 is clusteded, R1.a index is in memory, and the index is clustered.
   $B(R2) + T(R2) \times (B(R1)/V(R1, a)) = 200 + 2000 \times (1000/500) = 4200$