1. Suppose you need to sort a relation of 40 gigabytes, with 4 kilobyte blocks, using a memory size of 40 megabytes. Assume that t_S = 5ms and t_T = 0.1ms. Find the cost of sorting the relation, in seconds, with b_b = 1, 9, 30, 300, and 900, where b_b is the number of buffer blocks allocated to each input run and output run.

The number of blocks in the main memory buffer available for sorting (M) is
$(40\times10^6) / (4\times10^3) = 10^4$.
The number of blocks containing records of the given relation (b_r) is
$(40\times10^9) / (4\times10^3) = 10^7$.
The initial number of runs is $\lceil b_r / M \rceil = \lceil 10^7/10^4 \rceil = 1{,}000$.
Here, $t_S = 5\times10^{-3}$ seconds and $t_T = 10^{-4}$ seconds.

For b_b = 1
The merge can be done in one pass.
The number of block transfers is $10^7\times3 = 30\times10^6$.
The number of disk seeks is $2\times1000 + 10^7 = 10{,}002{,}000$.
Therefore, the cost of sorting the relation is:
$10{,}002{,}000 \times (5\times10^{-3}) + (30\times10^6) \times 10^{-4} = 53{,}010$ seconds.

For b_b = 9
The merge can be done in one pass.
The number of block transfers is $10^7\times3 = 30\times10^6$.
The number of disk seeks is $2\times1000 + \lceil 10^7/9 \rceil = 1{,}113{,}112$.
Therefore, the cost of sorting the relation is:
$1{,}113{,}112 \times (5\times10^{-3}) + (30\times10^6) \times 10^{-4} = 8{,}566$ seconds.

For b_b = 30
In each pass, the number of runs decreases by a factor of $\lfloor M / b_b \rfloor - 1 = \lfloor (10^4/30) \rfloor - 1 = 332$.
The number of merge passes required is $\lceil \log_{332}1000 \rceil = 2$.
The number of block transfers is $10^7\times5 = 50\times10^6$
The number of disk seeks is $2\times1000 + \lceil 10^7/30 \rceil \times3 = 1{,}002{,}002$.
Therefore, the cost of sorting the relation is:
$1{,}002{,}002 \times (5\times10^{-3}) + (50\times10^6) \times 10^{-4} = 10{,}010$ seconds.

For b_b = 300
In each pass, the number of runs decreases by a factor of $\lfloor (10^4)/300 \rfloor - 1 = 32$.
The number of merge passes required is $\lceil \log_{32}1000 \rceil = 2$.
The number of block transfers is $10^7\times5 = 50\times10^6$
The number of disk seeks is $2\times1000 + \lceil 10^7/300 \rceil \times3 = 102{,}002$.
Therefore, the cost of sorting the relation is:
$102{,}002 \times (5\times10^{-3}) + (50\times10^6) \times 10^{-4} = 5{,}510$ seconds.

For b_b = 900
In each pass, the number of runs decreases by a factor of $\lfloor (10^4)/900 \rfloor - 1 = 10$.
The number of merge passes required is $\lceil \log_{10}1000 \rceil = 3$.
The number of block transfers is $10^7\times7 = 70\times10^6$
The number of disk seeks is $2\times1000 + \lceil 10^7/900 \rceil \times5 = 57{,}560$.
Therefore, the cost of sorting the relation is:
$57{,}560 \times (5\times10^{-3}) + (70\times10^6) \times 10^{-4} = 7{,}288$ seconds.

2. Apply merge-join to the following numerical example used in the lecture.

$n_{student}$ =5,000 $b_{student}$ =100

n_{takes} =10,000 b_{takes} =400

(a) What will be the cost if the relations are ***not*** sorted and the memory size is still ***3*** blocks?

(b) What will be the cost if the memory size is increased to ***25*** blocks and the relations are ***not*** sorted?

(c) In order to reduce the number of seeks (without increasing the number of block transfers), what is the number of buffer blocks (i.e., b_b) that should be allocated

(i) to each run and the output in the merge step of sorting and,

(ii) for buffering each relation and the output in the join step?

(d) What will be the cost of part (c)?

(a) The initial numbers of runs of *student* and *takes* are ⌈(100/3)⌉= 34 and ⌈400/3⌉=134, respectively.
The numbers of merge passes required for *student* and *takes* are ⌈$\log_2$34⌉=6 and ⌈$\log_2$134⌉=8, respectively.
The number of block transfers in sorting the two relations inclusive of the output
= 100(2*6+ 2) + 400(2*8 + 2)
= 1,400 + 7,200
= 8,600 block transfers.
The total number of block transfers
= 8,600 + 100 + 400
= 9,100 block transfers.
The number of seeks in sorting the two relations inclusive of the output
= 34*2+100*(2*6) + 134*2+400*(2*8)
= 7,936
The total number of seeks
= 7,936+100+400
= 8,436

(b) The initial numbers of runs of *student* and *takes* are ⌈100/25⌉= 4 and ⌈400/25⌉=16, respectively.
The numbers of merge passes required for *student* and *takes* are both 1.
The number of block transfers in sorting the two relations inclusive of the output
= 100(2 + 2) + 400(2 + 2)
= 400 + 1,600
= 2,000 block transfers.
The total number of block transfers
= 2,000 + 100 + 400
= 2,500 block transfers.
The number of seeks in sorting the two relations inclusive of the output
= 4*2+100*2+16*2+400*2
= 1,040
The total number of seeks
= 1,040+100+400
= 1,540

(c)

(i)

- To reduce the number of seeks in the merge step of sorting, 5 buffer blocks are allocated to each run and the output for merging the 4 runs of *student*.
- For *takes*, to keep one merge pass, the number of buffer blocks cannot be increased.

(ii) To reduce the number of seeks in the merge-join step, 8 blocks are allocated for buffering each relation and the output.

(d) The number of seeks in sorting the two relations inclusive of the output
= 4*2 + ⌈ (100/5) ⌉*2 + 16*2 + 400*2
= 880
The total number of seeks
= 880 + ⌈ (100/8) ⌉ + ⌈ (400/8) ⌉
= 943

3. Consider two relations $r_1(A, B, C)$ and $r_2(C, D, E)$. For r_1, it has 20,000 tuples and 25 tuples fit on one block. For r_2, it has 45,000 tuples and 30 tuples fit on one block. Which join algorithm gives the lowest worst-case cost estimate for $r_1 \bowtie r_2$? Consider only number of block transfers in this exercise. Specify the minimum amount of memory in number of blocks for the worst-case estimates.

Given: $b_{r1} = 20{,}000/25 = 800$, $b_{r2} = 45{,}000/30 = 1{,}500$

For nested-loop join:

- Using r_1 as the outer relation, we need $20000 \times 1500 + 800 = 30{,}000{,}800$ block transfers.
- If r_2 is the outer relation we need $45000 \times 800 + 1500 = 36{,}001{,}500$ block transfers.
- Memory: 1 block for each input and 1 block for output.

For block nested-loop join:

- If r_1 is the outer relation, we need $800 \times 1500 + 800 = 1{,}200{,}800$ block transfers.
- If r_2 is the outer relation we need $1500 \times 800 + 1500 = 1{,}201{,}500$ block transfers.
- Memory: 1 block for each input and 1 block for output.

For merge-join:

- Memory: 3 blocks
- Assuming that r_1 and r_2 are not initially sorted on the join key, the total sorting cost inclusive of the output is B_s
  $= 1500(2\lceil \log_2(1500/3) \rceil + 2) + 800(2\lceil \log_2(800/3) \rceil + 2)$
  $= 30{,}000 + 16{,}000$
  $= 46{,}000$ block transfers.
- The total cost is
  $B_s + 1500 + 800$
  $= 48{,}300$ block transfers.

For hash-join:

(a) We assume that memory is big enough for every partition of the build relation. Since r_1 is smaller, we use it as the build relation and r_2 as the probe relation.

(b) Memory:
$M > n$ (for partitioning) and $M > \lceil 800/n \rceil + 1$ (for build and probe),
$M = 30$ blocks.

(c) The cost is
$3(1500+800)$
$= 6{,}900$ block transfers.