Problem 5 (16 = 7 + 7 + 2 points)

Consider the following program used to compute $y = y + Ax$ where A is an $N \times N$ sparse matrix stored in CSR format (see Fig. 1 as an example for this format). The matrix A has K non-zero elements, and x and y are (of course) vectors of length N.

```c
void smvm(int n, const double* values, const int* col_idx,
      const int* row_start, double* x, double* y)
{
  int i,j;
  double d;

  /* loop over N rows */
  for (i = 0; i < n; i++) {
    d = y[i]; /* scalar replacement since reused */
    /* loop over non-zero elements in row i */
    for (j = row_start[i]; j < row_start[i+1]; j++)
      d += values[j] * x[col_idx[j]];
    y[i] = d;
  }
}
```

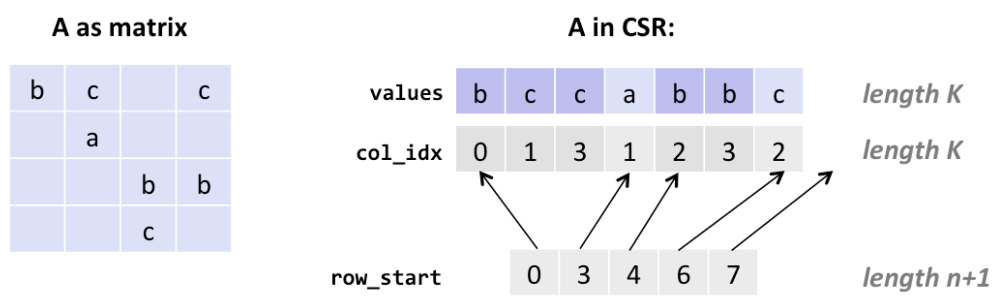

Figure 1: Compressed sparse row (CSR) format.

We assume that every row and every column of A has at least one non-zero element, and that the variables i and j are stored in registers. Further, we assume a cold (empty) cache with a cache block size of 8 bytes. Answer the following questions and provide enough detail so we see how you got to a solution.

1. Compute an upper bound for the operational intensity (unit: flops/byte) assuming only compulsory misses happen.

   Solution: Same as the master's solution.

2. Compute a lower bound for the operational intensity assuming that all array accesses lead to misses.

Let's assume that the elements of `row_start` and `row_end` do not stay in cache after consecutive accesses. In that case scenario, for the loads of double arrays, we have:

- N loads for `y` at line 9
- K loads for `values` at line 12
- K loads for `x` at line 12

And for the integer arrays, we have:

- N loads for `j = row_start[i]` at line 11 (loop initialization)
- N loads for `j < row_start[i+1]` at line 11 (since the check for the loop is performed upon each initialization).
- K loads for `col_idx[j]` at line 12
- K loads for `j < row_start[i+1]` at line 12 (since the check is performed for each iteration of the inner loop).

Therefore, the tighter lower bound for operational intensity, can be defined as:

$$I \geq \frac{2K}{(2K+N)\cdot 8 + (2K+2N)\cdot 8} \text{ flops/byte}$$

Assuming that the elements of `row_start` and `row_end` stay in cache after consecutive accesses, the operational intensity can be defined as:

$$I \geq \frac{2K}{(2K+N)\cdot 8 + \frac{K+N+1}{2}\cdot 8} \text{ flops/byte}$$