Rui Huang rhhq7

For homework one, at first, I just tried a very naïve method which read each line of lhs and each column of rhs. Below is the main code:

```
for(i = 0; i < lhs.size1();i++) {
    for(j = 0; j < rhs.size2();j++) {
        for(k = 0; k < rhs.size1();k++) {
        result(i,j) += lhs(i,k) * rhs(k,j);
      }
    }
  }
```

The running time was about 12 mins. Below were the reasons why it was slow and how I improved it:

1. In the for loop, I called the function rhs.size() i*j*k times and rhs.size2() i*j times and lhs.size1() I times. Calling function cost a lot of memory and time. it increased the running time very much. I solved it by defining three constant values out of the three loops and use these values in loops, The running time became better which was 11m0.831s. Main code is below:

```
for(i = 0; i < leftRow;i++) {
    for(j = 0; j < rightColumn;j++) {
        for(k = 0; k < rightRow;k++) {
            result(i,j) += lhs(i,k) * rhs(k,j);
        }
    }
}
```

2. In the third loop, the statement “result(i,j) += lhs(I,k) + rhs(k,j)” cost a lot of time because it read result(i,j) from the memory and write it back to memory. We need to pay attention that this statement was in the third loop. It will read and write memory i * j * k times without consideration of lhs(I,k) and rhs(k,j). In order to solve this problem, I defined a variable in the second loop and used this variable in the third loop. In this way, the variable can be kept in registor and cpu can just read and write back to register, which was much faster than memory. After this improvement, the running time is 8m56.871s and the main code is below:

```
for(i = 0; i < leftRow;i++) {
    for(j = 0; j < rightColumn;j++) {
            s = 0;
        for(k = 0; k < rightRow;k++) {
            s += lhs(i,k) * rhs(k,j);
        }
        result(i,j) = s;
    }
}
```

   Another improvement I did was that the statement “s += lhs(i,k) * rhs(k,j)” was very time costly because it called function of a matrix object every time which was much slower than just read from and write back to a 2-dimensions array. So, I converted lhs, rhs, result to 2-dimensions arrays and use them in the third loop. After that, I copied the value back to result. After that improvement, the running time was 5m28.003s and the main code is blow:

```
for(i = 0; i < leftRow;i++) {
        for(j = 0; j < rightColumn;j++) {
            s = 0;
            for(k = 0; k < rightRow;k++) {
                s += leftArray[i][k] * rightArray[k][j];
            }
            tempResult[i][j] = s;
        }
    }
for(i = 0; i < leftRow; i++) {
    for(j = 0; j < rightColumn; j++) {
        result(i,j) = tempResult[i][j];
    }
}
```

3. I noticed that the program read lhs line by line and rhs column by column. If the number of column of rhs is pretty large, each time an element of rhs was read, there was a cache miss. So, if we can read both lhs and rhs line by line, there would be some improvements.. So, instead of transposing rhs, I used the element lhs(i,j) to multiply all the elements in the jth line of rhs and then sumed them up. The benefit of this compared to transpose rhs is that lhs(i,j) can be kept at registor and doesn't need to be read from cahce or memory(if cache miss happen). Also, in the loop, I used index like leftArray[i][j] to visit the element. It contains several multiply operations which are time costly. In order to solve this issue, I used pointer and almost all the multiply operation because add operation. After these improvement, the running time is 5m21.314s and code is below:

```
float *left = *leftArray;
float *right = *rightArray;
float *temp = *tempResult;
float *pointer1 = temp - rightColumn;
 for(i = 0; i < leftRow; i++) {
      pointer1 += rightColumn;
          for(k = 0; k < leftColumn; k++) {
              register float s = *(left + i * leftColumn +k);
              register float *pointer2 = right + k * rightColumn;
                  for(j = 0; j < rightColumn; j++) {
                      *(pointer1+j) += s *    *(pointer2 + j);
              }
    }
}
```

4. 5 minutes was still pretty a lot. I use the compiler option O3 when compile the source file. After doing that, the running time became 48s.