Homework 10

November 16, 2018

1 Sorting

This part of the homework is similar to homework 9 except that the methods are curried with the first parameter being the comparison function.

1. Write a method `merge_sort: (f: (Int, Int) => Boolean) (lst: List[Int]) List[Int].` that takes a function, a list, and return a sorted list.
2. Write a method `selection_sort: (f: (Int, Int) => Boolean) (lst: List[Int]) List[Int]` that takes a function, a list and return a sorted list.
3. Write a method `insertion_sort: (f: (Int, Int) => Boolean) (lst: List[Int]) List[Int]` that takes a function, a list and return a sorted list.

2 Vector and matrix operations

This part of the homework is similar to homework 6 except that you will use Scala. Scala has a builtin `List` class and it has methods `map`, `zip`, and `reduce` that you can use to define your solution.

The following questions are about vectors and matrix. We represent vectors using lists. For example, `List(2,3,5,4)` represents a vector of four integers. We represent a matrix using a list of lists. For example, the matrix

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

is written as `List( List(1,2,3), List(4, 5, 6) )`.

1. Write a function `vectorAdd: (List[Int], List[Int]) => List[Int]` that add two integer vectors of the same size.

   For example, `vectorAdd (List(1,2,3), List(4,5,6))` should return `List(5, 7, 9)`.

2. Write a function `svProduct: (Int, List[Int]) => List[Int]` that multiple an integer with an integer list.

   For example, `svProduct(2, List(1,2,3))` should return `List(2,4,6)`.

3. Write a function `vmProduct: (List[Int], List[List[Int]]) => List[Int]` that multiple a row vector of size n with a matrix with n rows and m columns to produce a vector of size m.

   For example, `vmProduct(List(1,2,3), List(List(1,1), List(2,1), List(3,1)))` should return `List(14, 6)`. Or,

$$\begin{aligned}[1\ \ 2\ \ 3] \times \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{bmatrix} &= 1 \times [1\ \ 1] + 2 \times [2\ \ 1] + 3 \times [3\ \ 1] \\ &= [1\ \ 1] + [4\ \ 2] + [9\ \ 3] \\ &= [14\ \ 6]\end{aligned}$$

   This function uses the functions `svProduct` and `vectorAdd` defined earlier.

4. Write a function `matrixProduct: (List[List[Int]], List[List[Int]]) => List[List[Int]]` that multiple a $m \times n$ matrix with a $n \times k$ matrix to obtain a $m \times k$ matrix.
   For example

$$\begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 14 & 6 \\ 6 & 3 \end{bmatrix}$$

   where

$$v_1 = [1\ \ 2\ \ 3] \times \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} 14 & 6 \end{bmatrix}$$

   and

$$v_2 = [1\ \ 1\ \ 1] \times \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} 6 & 3 \end{bmatrix}$$

   That is,

```
matrixProduct(List(List(1,2,3), List(1,1,1)),
              List(List(1,1), List(2,1), List(3,1)))
  = List( List(14, 6), List(6, 3) )
```

   This problem will use the function `vmProduct` defined previously.

Submission

Write your solution in a file by the name of `Hwk10.scala`.

Testing

Use the following template to test your program.

```
object Hwk10{
  // your methods go here

  def main(args: Array[String]) {
    val lst = List(5,4,11,2,3,1,0,9)
    println(merge_sort(_>_)(lst))
    println(selection_sort(_>_)(lst))
    println(insertion_sort(_>_)(lst))
    println(merge_sort(_<_)(lst))
    println(selection_sort(_<_)(lst))
    println(insertion_sort(_<_)(lst))

    val v1 = List(1,2,3)
    val v2 = List(4,5,6)
    println(vectorAdd(v1, v2))
    println(svProduct(2, v1))
    val m1 = List(List(1,1), List(2,1), List(3,1))
    println(vmProduct(v1, m1))
    val m2 = List(List(1,2,3), List(1,1,1))
    println(matrixProduct(m2, m1))
  }
}
```

You should expect the following output.

```
List(11, 9, 5, 4, 3, 2, 1, 0)
List(11, 9, 5, 4, 3, 2, 1, 0)
List(11, 9, 5, 4, 3, 2, 1, 0)
List(0, 1, 2, 3, 4, 5, 9, 11)
List(0, 1, 2, 3, 4, 5, 9, 11)
List(0, 1, 2, 3, 4, 5, 9, 11)
List(5, 7, 9)
List(2, 4, 6)
List(14, 6)
List(List(14, 6), List(6, 3))
```