Tarea: MapReduce

Luis M. Román, Maria Fernanda Mora, Alfonso Kim, Andrea

13 de mayo de 2015

1. Máximo de un grupo

```
1 def calculate_max(numbers):
2     """ Calcula el maximo
3         :param numbers: La lista de enteros
4     """
5     return reduce((lambda x, y: x if x > y else y), numbers)
```

La línea *5* evalúa cada elemento de la lista y conserva en memoria el elemento más grande. Al final lo devuelve.

2. Promedio y desviación estandar

2.1. promedio

```
1 def average(numbers):
2     """ Calcula el promedio de la lista
3         :param numbers: La lista de enteros
4     """
5     values = map(lambda x: (1, x), numbers)
6     count, num_sum = reduce(lambda x, y: (x[0] + y[0], x[1] +
      ↪  y[1]), values)
7     return float(num_sum) / count
```

En la lina *5* simplemente se emite un 1 y el número evaluado. Durante el *reduce* se suman los unos y los números, de tal forma que queda una tupla con el conteo de valores y la suma de valores. El valor final es la división del segundo entre el primero.

2.2. Desviación Estándar

```
1  def standard_deviation(numbers):
2      """ Calcula la desviacion estandar de la lista
3          :param numbers: La lista de enteros
4      """
5      avg = average(numbers)
6      square_diffs = map(lambda x: (x - avg) ** 2, numbers)
7      return reduce(lambda x, y: x + y, square_diffs) ** 0.5
```

En la línea *5* se obtiene el promedio usando el método anterior, posteriormente se calcula la diferencia cuadrada del valor al promedio. En la línea *7* se suman las diferencias y se devuelve la raíz cuadrada de la suma.

3. Top ten de una cantidad

```
1  def top_n(numbers, n):
2      """ Calcula el top n de la lista
3          :param numbers: La lista de enteros
4          :param n: El numero de valores a encontrar
5      """
6      num_set = set(numbers)
7      top = []
8      # No me gusto que sea iterativo, si puedo lo arreglo
9      while len(top) < n and num_set:
10         max_n = calculate_max(num_set)
11         num_set.remove(max_n)
12         top.append(max_n)
13     return top
```

Para este problema simplemente se iteró N veces el método *calculate_max* de la sección 1.

4. Conteo por grupo

```
1  def world_count(numbers):
2      """ Cuenta las ocurrencias de un numero en la lista
3          :param numbers: La lista de enteros
4          :return: Un diccionario donde las llaves son los numeros
5                   y los valores son las veces que ocurrieron
6      """
7      worlds = {}
8      def sum_reduce(x):
9          """ El reductor: Se actualiza la lista de numeros con
               las veces que
10             ocurre cada numero
11         """
12         if x in worlds: worlds[x] += 1
13         else: worlds[x] = 1
14     map(sum_reduce, numbers)
          # Map no devuelve nada ya que el reductor no devuelve
15     return worlds
```

Se usó un reductor que va guardando los números procesados. Si se encuentra uno que se vió antes le suma uno al contador *línea 12*, si no crea un contador con 1 *línea 13*. La función *map* sólo itera sobre la lista de números usando el reductor definido.