SUPSI University of Applied Sciences and Arts of Southern Switzerland

Problem Set 4 - Algorithm and complexity

Hello students,
during last lecture you learned about algorithm and complexity. Today, we will deepen your understanding by doing some practical exercise.

Write the results of the complexity analysis by using a "Word-like" document editor, in particular the "equation" functionality, and by exporting them to PDF format.

Assignment 1 - Algorithm analysis

Assignment 1.1 - sorting algorithms

Arrange the following expressions from slowest to fastest growth rate.

$n \log_2 n \qquad 4^n \qquad \mathrm{k} \log_2 n \qquad 5n^2 \qquad 40 \log_2 n \qquad \log_4 n \qquad 12n^6$

Assignment 1.2 - Determining O(·)

Determine the O(·) for each of the following functions, which represent the number of steps required for some algorithm.

(a) $T(n) = n^2 + 400n + 5$
(b) $T(n) = 67n + 3n$
(c) $T(n) = 2n + 5n \log n + 100$
(d) $T(n) = \log n + 2n^2 + 55$
(e) $T(n) = 3\left(2^n\right) + n^8 + 1024$
(f) $T(n, k) = kn + \log k$
(g) $T(n, k) = 9n + k \log n + 1000$

Assignment 1.3 - Analyse the code written

1. What is the time-complexity of the `printCalendar()` function implemented in Problem Set 1 - Assignment 3?

2. What is the time-complexity of the *proper_subset* test operation implemented in Problem Set 3 - Assignment 1.3?

3. Prove or show why the worst case time-complexity for the `insert`() and `remove`() list operations is O(n).

Assignment 1.4 - Code evaluation

Evaluate each of the following code segments and determine the O(·) for the **best** and **worst case**. Assume an input size of n.

(a)
```
sum = 0
  for i in range(n):
    sum += i
```

(b)
```
sum = 0
while i > 1:
  sum += i
  i = i/2
```

(c)
```
for i in range(n):
  if i % 3 == 0 :
    for j in range( n / 2 ) :
      sum += j
  elif i % 2 == 0 :
    for j in range( 5 ) :
      sum += j
  else :
     for j in range( n ) :
       sum += j
```

Assignment 1.5 - slice the list

The slice operation is used to create a new list that contains a subset of items from a source list. Implement the *my_slice*() function:

```
def my_slice( input_list, first, last )
```

which accepts a list and creates a sub-list of the values in *input_list*.

What is the worst case time for your implementation and what is the best case time?

Note: *first* is included, *last* is not included.

Example:

```
test_list = [i for i in range(10)]

print(my_slice(test_list, 1, 2))  # [1]
print(my_slice(test_list, 0, 10))  # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(my_slice(test_list, 2, 5))  # [2, 3, 4]
```

Assignment 2 - complete the *SparseMatrix* class

Assignment 2.0 - Transcribe the example

In the reference-book, is proposed a basic partial implementation of the **SparseMatrix ADS**. This is shown in the listing **4.2** (**pages 116 - 117**).

Read through the code, and copy it to a file called “sparse_matrix.py”.

ATTENTION : there may be a few mistakes in the code of the book, fix them!

Assignment 2.1 - add the missing methods

Complete the Matrix examples by adding the following functionality:

1. **`__getitem__`** - returns the element located at such coordinates.
1. **`subtract`** - The same as the add() operation but subtracts the two matrices.
2. **`multiply`** - Creates and returns a new matrix that is the result of multiplying this matrix to the given rhsMatrix. The two matrices must be of appropriate sizes as defined for matrix multiplication.
3. **`transpose`** - Returns a new matrix that is the transpose of this matrix.

Assignment 2.2 - magic operators

Add Python’s magic operator methods to the Set class that can be used to perform similar operations to those already defined by named methods:

Magic Method	Current Method
`__add__(rhsMatrix )`	`union(rhsMatrix)`
`__mul__(rhsMatrix)`	`intersect(rhsMatrix)`
`__sub__(rhsMatrix)`	`difference(rhsMatrix)`

Assignment 2.3 - analyse the methods

Determine the worst case time-complexities for the SparseMatrix methods implemented in the previous question.