MATH1058 coursework [40 marks]

Stefano Coniglio

May 13, 2019

1 Implementation

1.1 [5 marks] Pseudocode for the $O(n^3)$ version

Report the pseudocode of your implementation "`dijkstra1`".

Example (this code is entirely unrelated to the assignment!)

```
1 def dummy_function(myList):
2   n = len(myList)
3   for i in range(0,n):
4     myList[i] = myList[i] + 1
5     myList[i] = myList[i] % 2
6     for j in range(0,i):
7       myList[j] = myList[i] + i
8 return myList
```

1.2 [5 marks] Show that your implementation indeed has complexity $O(n^3)$

Proceed as we did in the lectures, analyzing the complexity of each line/block of lines.

Example adopting the previous code.

- Lines 2 and 5 take $O(1)$.
- Each time the block of lines 4-to-7 is repeated:
  - Lines 4-to-5 $O(1)$.
  - Each time it is repeated, line 7 takes O$O(1)$. Since, due to line 6, it is repeated $O(n)$ times, it overall takes $O(n)$.

  Overall, lines 4-to-7 take $O(n)$.

Overall, we have a complexity of:

$$O(1) + +O(n)\big(O(1) + O(n)\big) = O(n^2).$$

1.3 [5 marks] Pseudocode for the $O(n^2)$ version

Report the pseudocode of your implementation "`dijkstra2`".

Example (this code is entirely unrelated to the assignment!)

```
def dummy_function(myList):
  n = len(myList)
  for i in range(0,n):
    myList[i] = myList[i] + 1
    myList[i] = myList[i] % 2
    for j in range(0,i):
      myList[j] = myList[i] + i
return myList
```

1.4 [5 marks] Show that your implementation indeed has complexity $O(n^2)$

Proceed as we did in the lectures, analyzing the complexity of each line/block of lines.

Example adopting the previous code.

- Lines 2 and 5 take $O(1)$.
- Each time the block of lines 4-to-7 is repeated:
  - Lines 4-to-5 $O(1)$.
  - Each time it is repeated, line 7 takes O$O(1)$. Since, due to line 6, it is repeated $O(n)$ times, it overall takes $O(n)$.

  Overall, lines 4-to-7 take $O(n)$.

Overall, we have a complexity of:

$$O(1) + +O(n)\big(O(1) + O(n)\big) = O(n^2).$$

2 Experimental results and analysis

2.1 [5 marks] Empirical computing times

Report the empirical computing times you have computed in Excel for `dijkstra1` and `dijkstra2` (y_1 and y_2) in a table. Report in an Excel line chart the values of y_1 versus n and y_2 versus n.

Table 1: A table (not related to the coursework). Notice that captions should precede tables, whereas they should follow figures.

header	header2	header3
name1	12	12
$s = \|S\|$	5	42

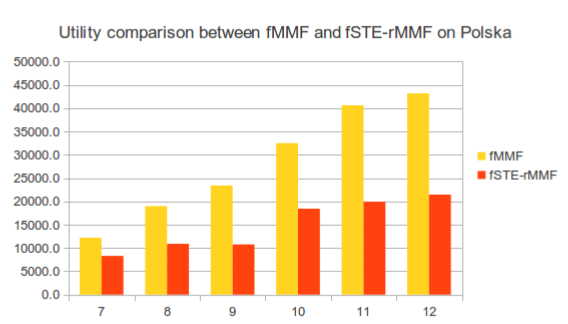

Figure 1: A dummy figure (also unrelated to the coursework).

2.2 [5 marks] Logarithms of the computing times

Report the logarithms of the empirical computing times you have computed in Excel for `dijkstra1` and `dijkstra2` ($\log(y_1)$ and $\log(y_2)$) in a table. Report in an Excel line chart the values of $\log(y_1)$ versus $\log(n)$ and $\log(y_2)$ versus $\log(n)$.

Table 2: Another dummy table.

header	header2	header3
name1	12	12
$s = \|S\|$	5	42

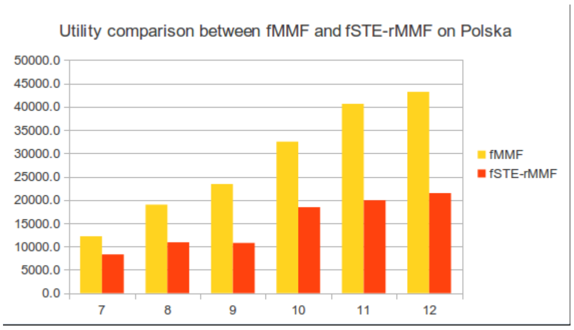

Figure 2: Another dummy figure.

2.3 [5 marks] Estimated complexities

Report the computational complexities of `dijkstra1` and `dijkstra2` (y_1' and y_2') you have estimated in Excel via linear/affine regression as two functions of n.

Example using dummy functions:

$$y_1' = \sin(n) \tag{2.1}$$

$$y_2' = \tan(n) \tag{2.2}$$

2.4 [5 marks] Speedup

Report the estimated speedup `dijkstra2` w.r.t. `dijkstra1` as the function (of the instance size n) $\frac{y_2'}{y_1'}$. Compare this empirical speedup to the theoretical one (the latter is equal to the ratio of the computational complexities of your implementations, expressed in big-O notation. Is the empirical one smaller or larger than the theoretical one? Motivate the answer.

Example using dummy functions:

$$\frac{y_2'}{y_1'} = \frac{\sin(x)}{\tan(n)} \tag{2.3}$$

Theoretical speedup:

$$\frac{O(n^8)}{O(n^{11})} = O(n^{-2}). \tag{2.4}$$

Some explanation should be written here.