CSE103 Introduction to Algorithms

Midterm Exam

May 6th, 2021

You have 1 hour and 30 minutes.
No computer, laptop, phone or any Internet-connected device authorized.
You may use the supplementary material on recurrences.
Each question/subquestion is independent of the others and may be answered in any order.
The maximum score for the exam is 50 points. The number of points awarded by fully answering each question is written next to the question/subquestion itself.

Exercise 1: asymptotic notation (12 points)

True or false? Please give a brief justification of your answer.

1. **(2 points)** $n^2 + 10n + 6 = O(n^3)$
2. **(2 points)** $2^n + 42n^{1000} = \Theta(2^n)$
3. **(2 points)** $4\sqrt[4]{n} + \log n = O(\log n)$
4. **(2 points)** $37n^4 + n + 15 = \Omega(n^2)$
5. **(2 points)** if $f(n) = n2^{O(n)}$, then $\log f(n) = O(n)$
6. **(2 points)** $2^{\frac{n \log n}{2}} = O(2^n)$

Exercise 2: recurrences (12 points)

Give the best asymptotic upper bound you can to the following recurrences (where, in all cases, $T(0) = 1$). Please justify your answer.

1. **(3 points)** $T(n) = 2T\left(\frac{n}{3}\right) + 5n + \log n$
2. **(3 points)** $T(n) = 2T\left(\frac{n}{2}\right) + 2^n$
3. **(3 points)** $T(n) = 4T\left(\frac{n}{2}\right) + \frac{n^2}{\sin n + 2}$
4. **(3 points)** $T(n) = T\left(\frac{2n}{3}\right) + 1$

Exercise 3: a weird sorting algorithm (13 points)

Believe it or not, the following function sorts a list `l`:

```
def weirdSort(l):
    n = len(l)
    r = l
    if n == 2 and l[0] >= l[1]:
        r = [l[1],l[0]]
    elif n > 2:
        a = n // 3
        b = 2*a + (n % 3)
        r1 = weirdSort(l[0:b]) + l[b:n]
        r2 = r1[0:a] + weirdSort(r1[a:n])
        r = weirdSort(r2[0:b]) + r2[b:n]
    return r
```

Please answer the following questions, justifying your answer in each case:

1. **(8 points)** Is this algorithm more or less efficient than insertion sort? Justify your answer by a complexity analysis using one of the techniques you have learned.

2. **(5 points)** Suppose that the above function is modified by replacing the `if` condition at the third line with

   ```
   n == 2 and l[0][0] >= l[1][0]
   ```

   In this way, the function operates on lists of lists, and sorts them according to the value of the first element of the inner lists. For example, `weirdSort([[1,4],[2,3],[0,2]])` will return `[[0,2],[1,4],[2,3]]`.
   Recall that a sorting algorithm is *stable* if it preserves the relative position of elements with identical sorting key (in this case, the sorting key is the value of the first element of the inner lists). For example, when sorting `[[0,4],[2,3],[0,2]]`, a stable sorting algorithm must return `[[0,4],[0,2],[2,3]]`, because `[0,4]` comes before `[0,2]` in the original list, whereas an unstable sorting algorithm may return `[[0,2],[0,4],[2,3]]`, which is equally well sorted, but the position of the elements `[0,4]` and `[0,2]` has been swapped.
   Show, by means of an example, that `weirdSort` as modified above is *not* stable. Is it possible to modify it so that it becomes stable? How?

Exercise 4: Russian multiplication (13 points)

The following is an algorithm for multiplying two *strictly positive* integer numbers `x` and `y`:

```
def mul(x, y, s=0):
    if x == 1:
        return s + y
    if x % 2 != 0:
        s += y
    return mul(x//2, y*2, s)
```

Known by some as "Russian multiplication", it is in fact a variant of a very old multiplication algorithm used by the ancient Egyptians. Notice that it only requires knowing how to add, multiply by 2, divide by 2, and recognizing whether a number is even or odd.

We will consider the following cost model, which is realistic when representing arbitrarily big numbers in base 2:

- addition of two n-digit numbers costs $\Theta(n)$;
- multiplying or dividing an arbitrary number by 2 costs $\Theta(1)$;
- verifying whether an arbitrary number is even or odd costs $\Theta(1)$.

Please answer the following questions, justifying your answer in each case:

1. **(5 points)** What is the worst-case asymptotic (big O) complexity of multiplying two n-digit numbers using Russian multiplication?
2. **(3 points)** What is the worst-case asymptotic (big O) complexity of executing `mul(1,y)` with respect to the number n of digits of `y`?
3. **(3 points)** What is the worst-case asymptotic (big O) complexity of executing `mul(x,1)` with respect to the number n of digits of `x`?
4. **(2 points)** The ancient Egyptians did not consider zero to be a number. What happens if the above algorithm is applied to `x = 0`? What about `y = 0`?