COMPSCI 2C03 – Week 3 Exercises

© 2023, Sam Scott, McMaster University

- UPDATE – fixed 4b solution on, question 1 solution on October 1
- You should try these on your own before you look at the solutions!
- Just because your solution doesn't match the one here, doesn't mean it's wrong.
  - Does your solution work?
  - Does the official solution work?

Lecture 1: Basic Sorting Algorithms

1. Compute the **worst case running time** function $T(n)$ for the algorithm below. Make sure to count all basic operations both inside and outside of the loop. The parameter **n** is the length of the array to search.

```
search(a: array, n: int, key):                            Worst case
      index ← 0                          1
      found ← false                      1
      while index < n and not found:     3     n times 3
            if a[index] == key:                        2
                  found ← true           1 (this can only happen once)
            index += 1                                 2
      return found                       1
```

   T(n)=7n+7

2. Compute the running time function $T(n)$ for the algorithm below. Make sure to count all basic operations both inside and outside of the loops. The **//** operator is **floor division**. It rounds down and returns the result as an integer.

```
sum ← 0                  1
k ← n//2                 2
while k > 0:             1  n/2 times 1
      for i←0 to k-1:                 1 n²/8+n/4 times* 2
            sum ← sum + 1                                 2
      k ← k - 1                       2

                         4    +      4n/2      +     4n²/8+4n/4
```

   $T(n)=4n^2/8+3n/4+4$

3. Write an insertion sort algorithm that works with a doubly linked list. When moving items within the list, you can assume it's ok just to move the contents of a node, you don't have to change the links to move the node itself.

```
insertion_sort(L, n):
```

```
if L.head == null:
    return

current ← L.head.next

while current != null:
  temp ← current.item
  j ← current.previous

  while j != null and temp<j.item
    j.next.item ← j.item
    j ← j.previous

  if j.previous == null:
    L.head.item ← temp
  Else:
    j.next.item ← temp

  current ← current.next
```

Lecture 2: Algorithm Analysis

4. Using the definition of $O(f(n))$ and $\Omega(f(n))$, prove the following statements:
    a. $(70n^3 - 300n + 2)/2 \in O(n^3)$

    b. $12n^5 + n^4 - 700 \in O(n^6)$

    Pre-proof analysis:

    a. Need $12n^5 + n^4 - 700 \leq cn^6$.
    b. Use the fact that $12n^5 < 12n^6$ for $n \geq 1$, $n^4 < n^6$ for $n \geq 1$ and $-700 < n^6$ for all n.
    c. $12n^5 + n^4 - 700 \leq 12n^6 + n^6 + n^6 = 14n^6 \leq cn^6$ for all $c \geq 14$, $n \geq 1$
    d. BUT we also need $0 \leq 12n^5 + n^4 - 700$. Use the fact that $700 \leq 12n^5$ for all $n \geq 4$.

    Proof: choose $c = 14$, $n_0 = 4$

    $12n^5 + n^4 - 700 \leq 12n^6 + n^6 + n^6 = 14n^6$ for all $n \geq 4$

    $700 \leq 12n^5 \leq 12n^5 + n^4$ for all $n \geq 4$

    Therefore $0 \leq 12n^5 + n^4 - 700 \leq cn^6$ for all $n \geq n_0$

    c. $20n^2 - 33n - 22 \in \Omega(n^2)$

    d. $11n^2 - 43 \in \Omega(n)$

5. Using the definition of $O(f(n))$ and $\Omega(f(n))$, prove the following statements:
    a. $(n^3 - 9n - 9)/2 \notin O(n^2)$
    b. $n^2 + n - 555 \notin O(n)$
    c. $11n^2 - 43 \notin \Omega(n^3)$

    Pre-proof analysis:

    We need to find an n to show that $11n^2 - 43 < cn^3$.
    Since $11n^2 - 43 < 11n^2$, we just have to show $11n^2 < cn^3$, so $(11/c)n^2 < n^3$, which is true when $n > 11/c$ and $n \geq 1$

    Proof: Using the def of Big Omega, rephrase the claim as: for every $c > 0$ and $n_0 >= 0$, there is an $n >= n_0$ such that $11n^2 - 43 < cn^3$.

    Let c and n_0 be any numbers such that $c > 0$ and $n_0 >= 0$. Then pick any n such that $n > 11/c$ and $n \geq n_0$ and $n \geq 1$.

    Since $n > 11/c$, therefore $(11/c)n^2 < n^3$ and $11n^2 < cn^3$. And since $11n^2 - 43 < 11n^2$, therefore $11n^2 - 43 < cn^3$ for some $n \geq n_0$.

    Since n_0 and c were arbitrarily chosen, it is proven.

Lecture 3: Working with Big O

6. Write an algorithm that, given two sorted arrays of n numeric values, prints all elements that appear in both arrays, in sorted order. The running time of the program should be $O(n)$.

   # note – assumes sorted order.

```
print_duplicates(a, b):
      ia = ib = 0
      while ia<len(a) and ib<len(b):
            if a[ia] < b[ib]:
                  ia ← ia + 1
            else if a[ia] > b[ib]:
                  ib ← ib + 1
            else:
                  print(a[ia])
                  ia ← ia + 1
                  ib ← ib + 1
```

   Each time through the loop, either ia or ib are incremented(or both). So the upper bound on the number of times through the loop is 2n. Therefore, the algorithm is O(n)

7. Use limits to prove the following:
   a. $12n^2 + 5$ is $\Theta(n^2)$
   b. $3n^3 - 2n^2 + 5$ is $\Omega(n^2)$ but not $O(n^2)$

   $$\lim_{n\to\infty} \frac{3n^3-2n^2+5}{n^2} = \lim_{n\to\infty} \frac{3n^3}{n^2} + \lim_{n\to\infty} \frac{2n^2}{n^2} + \lim_{n\to\infty} \frac{5}{n^2} = \lim_{n\to\infty} (3n + 2 + 0) = \infty$$

   Therefore it's $\Omega(n^2)$ but not $O(n^2)$

   c. $5n^6 + 3n^3$ is $O(n^8)$ but not $\Omega(n^8)$

8. Use the six Useful Big O facts from the lecture to show that:

$(n^2 + 5)(n - 4) + 15n \in O(1.5^n)$

n^2+5 is $O(n^2)$ by rule 5.
$n-4$ is $O(n)$ by rule 5.
$15n$ is $O(n)$ by rule 5.
By rule 3, $(n^2 + 5)(n - 4)$ is $O(n^2 n)=O(n^3)$.
By rule 2, $(n^2 + 5)(n - 4) + 15n$ is $O(n^3+n)$
By rule 5 $n^3+n = O(n^3)$, so $(n^2 + 5)(n - 4) + 15n \in O(n^3)$
By rule 6, n^3 is $O(1.5^n)$.
Therefore, $(n^2 + 5)(n - 4) + 15n \in O(1.5^n)$

9. Use the definition of Big O to prove Rule 1 from the "Big O Facts" slide:

If $d(n) \in O(f(n))$ then $k \cdot d(n) \in O(f(n))$ for $k > 0$.

If $d(n)$ is $O(f(n))$ then there exists c and n_0 such that $d(n) \leq cf(n)$ for all $n>n_0$

If $k>0$ then it's also the case that $kd(n) \leq kcf(n)$ for all $n>n_0$.

Using $c_2=kc$, we have $kd(n) \leq c_2 f(n)$ for all $n>n_0$, which satisfies the definition of Big O.

Therefore $kd(n)$ is in $O(f(n))$

10. Use the definition of Big O (not limits or other properties) to prove the following:
    a. $10 \log n \in O(\ln n)$

       Change of base formula: $\log n = \ln n / \ln 10$, so $\ln n = \ln 10 \log n$
       Consider $c = 10/(\ln 10)$, $n_0=1$
       $10 \log n = (10/\ln 10) \ln n$ (change of base formula)
       So $10 \log n \leq c \ln n$ for all $n \geq n_0$

    $n^2 + 2 \log n \in O(n^2)$

       Consider $c = 2$, $n_0=1$
       $\log n < n$ for all n, therefore $\log n^2 < n^2$, therefore $2 \log n < n^2$.
       Therefore $n^2+2\log n \leq 2n^2$.
       $N^2+2\log n \geq 0$ for all $n>1$.
       Therefore, $0 \leq n^2+2 \log n \leq cn^2$ for all $n \geq n_0$.

b. $n \in O(n \log n)$

Consider $c=1$, $n_0=10$.

For $n \geq 10$, $\log n \geq 1$. Therefore $n \leq n \log n$ for all $n \geq 10$.

Therefore $0 \leq n \leq c\, n \log n$ for all $n \geq n_0$