CSC 402
Exam 2 Solutions
March 24, 2023

---

[20] 1.

a. For each pair of functions, circle *all* the relationships that apply. No justification is necessary.

i. $f(n) = n^2$
$g(n) = n^3$

Circle **all** that apply:

$f = \Omega(g)$ $\qquad$ $f = \Theta(g)$ $\qquad$ $\boxed{f = O(g)}$

ii. $f(n) = n^{1.001}$
$g(n) = n \log n$

Circle **all** that apply:

$\boxed{f = \Omega(g)}$ $\qquad$ $f = \Theta(g)$ $\qquad$ $f = O(g)$

iii. $f(n) = 2^n$
$g(n) = 2^{n+1}$

Circle **all** that apply:

$\boxed{f = \Omega(g)}$ $\qquad$ $\boxed{f = \Theta(g)}$ $\qquad$ $\boxed{f = O(g)}$

iv. $f(n) =$ solution to the runtime recurrence $T(n) = 4T(n/2) + n^2\sqrt{n}$ and $T(1) = 1$
$g(n) = n^{2.5}$

Circle **all** that apply:

$\boxed{f = \Omega(g)}$ $\qquad$ $\boxed{f = \Theta(g)}$ $\qquad$ $\boxed{f = O(g)}$

v. $f(n) =$ worst case running time of InsertionSort on an array of n integers.
$g(n) =$ solution to the recurrence $T(n) = 2T(n/2) + n$. Assume that n is an exact power of 2 and $T(1) = 1$.

Circle **all** that apply:

$\boxed{f = \Omega(g)}$ $\qquad f = \Theta(g) \qquad f = O(g)$

b. Use the expansion method to solve the following recurrence. Express your running time in Θ notation.

$$T(n) = T(n/3) + n$$

Assume that $T(n) = 1$ for all $n \leq 3$ and that n is an exact power of 3. A solution that does not use the method of expansion will receive no credit.

Solution. Expand the recurrence to see the pattern:

$$\begin{aligned} T(n) &= T(n/3) + n \\ &= T(n/9) + n/3 + n \\ &= T(n/27) + n/9 + n/3 + n \\ &\ldots \\ &\ldots \\ &= T(n/3^k) + n/3^{k-1} + \ldots + n/9 + n/3 + n \end{aligned}$$

The recursion bottoms out when $n/3^k = 3$, i.e., when $k = \log_3 n - 1$. So,

$$T(n) = 1 + n(\sum_{i=0}^{\log_3 n - 2} \frac{1}{3^i}) = \Theta(n)$$

(since the summation evaluates to less than 3).

---

[9] 2. Consider the following sorting algorithm that is a variation of merge sort: instead of splitting the list into two halves, we split it into three thirds. Then we recursively sort each third and merge them.

```
MergeSort3(A[0..n − 1])
    if n ≤ 1 then
        return A[0..n − 1]
    k ← ⌈n/3⌉
    m ← ⌈2n/3⌉
    return Merge3(MergeSort3(A[0..k − 1]),
                  MergeSort3(A[k..m − 1]),
                  MergeSort3(A[m..n − 1]))
```

```
Merge3(L0, L1, L2)
    return Merge(L0,Merge(L1, L2))
```

Assume that you have the procedure `Merge` from `MergeSort` that takes as input two sorted lists ℓ and ℓ' and returns a sorted merged list in $O(\ell + \ell')$ time. You may assume that n is a power of some constant that you like. **You don't need to justify your answers or show your work for the questions below.**

(a) What is the asymptotic running time of $\texttt{Merge3}(L_0, L_1, L_2)$, if L_0, L_1, and L_2 are three sorted lists, each of length $n/3$. Express your answer using $O(\cdot)$ notation.

Solution. $O(n)$.

(b) Let $T(n)$ denote the running time of `MergeSort3` on an array of size n. Write a recurrence relation for $T(n)$.

Solution. By assuming n as a power of 3 we get

$$\begin{aligned} T(n) &= 1, \quad\quad\quad\quad n = 1 \\ T(n) &= 3T(n/3) + O(n), \text{otherwise} \end{aligned}$$

(c) Solve the recurrence in part (b). Express your answer using $O(\cdot)$ notation.

Solution. $T(n) = O(n \log n)$.

(d) Is `MergeSort3` algorithm asymptotically faster than Insertion Sort?

Solution. Yes.

---

[16] 3. Recall the algorithm from lecture for fast integer multiplication that runs in time that is asymptotically faster than n^2. In particular, we can use the following simplification to multiply two n-digit integers x and y:

$$\begin{aligned} xy &= (x_1 \cdot 10^{n/2} + x_0)(y_1 \cdot 10^{n/2} + y_0) \\ &= x_1 y_1 \cdot 10^n + [(x_1 + x_0)(y_1 + y_0) - x_1 y_1 - x_0 y_0] \cdot 10^{n/2} + x_0 y_0 \end{aligned}$$

Answer the following questions:

i Suppose we use the algorithm to multiply $x = 12344321$ and $y = 12352452$. Consider the initial (top-level) call to the algorithm. What are the numerical values of x_0, x_1, y_0, y_1, n? No justification is necessary.

Solution. $x_1 = 1234, x_0 = 4321, y_1 = 1235, y_0 = 2452, n = 8$

ii Again, suppose we use the algorithm to multiply $x = 12344321$ and $y = 12352452$. The initial (top-level) call to the algorithm now makes 3 recursive calls. What 3 pairs of numbers will be passed in as inputs to the 3 recursive calls? Your answers should be numerical values (i.e., they should not be in terms of any variables). No justification is necessary.

Solution. $(x_1, y_1) = (1234, 1235)$, $(x_0, y_0) = (4321, 2452)$, $(x_1 + x_0, y_1 + y_0) = (5555, 3687)$

---

[20] 4. (a) What is the running time of the following code fragment? Express your answer using Θ notation. Justify your answer. When the function `duplicates` is first called, it is passed an array $A[1..n]$.

```
duplicates(A)
ℓ = length(A)
if ℓ = 1 then
   return False
else if duplicates(A[1..ℓ − 1]) = True then
   return True
for i = 1 to ℓ − 1 do
   if (A[ℓ] = A[i]) then
      return True
return False
```

Solution. For some constant c, the recurrence for the running time of the function `duplicates` on an array A of size n is

$$T(n) = \begin{cases} T(n-1) + cn & ,n \geq 2 \\ 1 & ,\text{otherwise} \end{cases}$$

We expand the recurrence as follows.

$$\begin{aligned} T(n) &= T(n-1) + cn \\ &= T(n-2) + c(n-1) + cn \\ &= T(n-3) + c(n-2) + c(n-1) + cn \\ &\cdots\cdots\cdots \\ &\cdots\cdots\cdots \\ &= T(n-k) + c\sum_{i=0}^{k-1}(n-i) \end{aligned}$$

The recursion bottoms out when $k = n - 1$. Thus we get

$$\begin{aligned} T(n) &= 1 + c\sum_{i=2}^{n} i \\ &= c\sum_{i=1}^{n} i \\ &= \Theta(n^2) \end{aligned}$$

(b) What is the running time of the following code fragment? Express your answer using Θ notation. Justify your answer. When the function `duplicates` is first called, it is passed an array $A[1..n]$.

```
duplicates(A)
  ℓ = length(A)
  if ℓ = 1 then
     return False
  else
     return (duplicates(A[1..ℓ − 1])) || (duplicates(A[2..ℓ])) || (A[1] = A[ℓ])
```

Solution. For some constant c, the recurrence for the running time of the function `duplicates` on an array A of size n is

$$T(n) = \begin{cases} 2T(n-1) + c & , n \geq 2 \\ 1 & , \text{otherwise} \end{cases}$$

This recurrence was solved in class for `powerof2` function and the answer is $T(n) = \Theta(2^n)$.

---

[13] 5. Given two arrays S_1 and S_2 of real numbers (ordered arbitrarily) and a real number z, give an algorithm that finds two numbers, one from S_1 and the other from S_2 whose sum is exactly z. If no such pair exists then your algorithm should output NIL. The algorithm should run in time $O(n \log n)$, where n is the number of elements in each array. Justify the running time of your algorithm. **No proof of correctness is required.**

Solution. Sort all elements in S_1. Then, for each element $y \in S_2$, we use Binary Search to check if $z - y \in S_1$. If the binary search returns `true` for any element $y \in S_2$ then we have found a pair of numbers from S_1 and S_2 whose sum is exactly z. Below is the algorithm in pseudocode form.

```
MergeSort(S_1) // S_1 is now sorted
for each element y in S_2 do
  x = z - y
  if (BinarySearch(S_1,x) == True) then
    return (x,y)
return Nil
```

The running time of Binary Search is $O(\log n)$ and hence the body of the for loop takes $O(\log n)$ time. The for loop runs for at most n times. Hence the total running time of the for loop is $O(n \log n)$. Combining this with $O(n \log n)$ time to sort S_1 gives us a total running time of $O(n \log n)$.

---

[12] 6. The following questions refer to matchings between people and pets with preferences as described for the Gale-Shapley algorithm. For each person, their highest preference pet is their *favorite* pet, and for each pet, their highest preference person is their *favorite* person.

i. If no two people have the same pet as their first preference and people propose, how many proposals occur in the Gale-Shapely algorithm before the algorithm terminates? Your answer must be exact (e.g., it cannot use asymptotic notation). No justification is necessary.

Solution. n.

ii. If all people have identical preference lists, how many proposals (people propose) occur before the Gale-Shapley algorithm terminates? Express your answer using Θ-notation. No justification is necessary.

Solution. $\Theta(n^2)$.

iii. **True/False.** Consider a matching between 3 people and 3 pets. If each person matches with their favorite pet, then the matching is stable. Justify your answer.

Solution. **True.** Since each person has their favorite pet, they have no incentive to elope and hence the matching is stable.

iv. **True/False.** For all integers $n > 0$, in any instance of the stable matching problem with n people and n pets, for each pair in a matching, if at least one of the following holds

- the person is matched with their favorite pet
- the pet is matched with their favorite person

then the matching is stable. Justify your answer.

Solution. **False.** Consider the following instance of people and pets with their preference lists: $p_1(t_1, t_2), p_2(t_1, t_2)$ and pets $t_1(p_2, p_1), t_2(p_2, p_1)$. Consider the matching containing pair (p_1, t_1) and (p_2, t_2). Note that in the matching, p_1 gets their favorite pet and t_2 get their favorite person, but (p_2, t_1) will elope.