CS 344, Section 5-8, Sp2021
Homework 1.
100 points + 15 points EC

Note for this HW and all future HW: Unless otherwise specified, you may use any algorithm covered in class as a "black box" – for example you can simply write "sort the array in $O(n\log(n))$ time" or "solve sorted-2-sum(A,T) in $O(n)$ time" without having to describe how to do this.

1 Problem 1 (3 point per part) – 30 total

For each of the following functions, state whether $f(n) = O(g(n))$ or $f = \Omega(g(n))$, or if both are true, then write $f = \Theta(g(n))$. No proofs required for this problem.

1. $f(n) = n^2 - 7n$ and $g(n) = n^3 - 10n^2$
2. $f(n) = (\sqrt{n})^3$ and $g(n) = n^2 - (\sqrt{n})^3$
3. $f(n) = n^{log_3(4)}$ and $g(n) = n\log^3(n)$
4. $f(n) = 2^{\log_2(n)}$ and $g(n) = n$
5. $f(n) = log^5(n)$ and $g(n) = n/\log(n)$
6. $f(n) = 4^n$ and $g(n) = 5^n$
7. $f(n) = \log_4(n)$ and $g(n) = \log_5(n)$
8. $f(n) = n^3$ and $g(n) = 2^n$
9. $f(n) = \sqrt{n}$ and $g(n) = \log^3(n)$.
10. $f(n) = n\log(n)$ and $g(n) = n^2$

2 Problem 2 – 30 points total

- Part 1 (10 points): Prove by induction that $\sum_{i=0}^{k} i2^i = (k-1)2^{k+1} + 2$
- Part 2 (10 points): Prove that $\sum_{i=1}^{n} \frac{i^4}{10} = \Theta(n^5)$

  NOTE: for this problem, you may not use any outside formulas. In particular, you can look up online an exact expression for $\sum_{i=1}^{n} i^4$, but you are NOT allowed to use that. The whole point of this problem is that, as we saw in lecture, you can figure out that this sum is $\Theta(n^5)$ without needing any complicated formulas.
- Part 3 (10 points): What is $\sum_{i=0}^{\log_2(n)} 4^i$ equal to in Θ-notation? (No formal proof necessary, just a brief explanation.)

  HINT: use the formula for geometric sum: $\sum_{i=0}^{k} r^k = (r^{k+1}-1)/(r-1)$. This is generally a very useful formula.

3 Problem 3 (10 points total)

- Part 1 (2 point): Simplify $8^{log_4(n)}$
- Part 2 (4 points): Simplify $3^{log_2(n)}$ – in particular write is a n to the power of some number.
- Part 3 (4 points): Prove that for any constants c, c', $\log_c(n) = \theta(\log_{c'}(n))$.

4 Problem 4 (30 points total):

Write an algorithm in pseudocode for each of following two problems below. The algorithm should be simple in all three cases!

Part 1: Closest Pair (10 points)

- Input: An array A with n distinct (non-equal) elements
- Output: numbers x and y in A that minimize $|x-y|$, where $|x-y|$ denotes absolute-value(x-y)

The run-time should be significantly better than $O(n^2)$.

Part 2 (10 points): Given an array A of size n, give an algorithm FindMax(A) that computes the maximum value in A.

The run-time should be $O(n)$.

Part 3: (10 points) Given an array A of length n, find a triplet of indices i, j, k such that $A[i] + A[j] = A[k]$, or output "no such triplet exists". (If there are many such triplets, you only have to return one of them.)

In this case, the algorithm will be slower that in the above parts: your run-time should be $O(n^2)$.

WHAT TO RETURN: You can just return the numbers $A[i], A[j], A[k]$ for which $A[i] + A[j] = A[k]$. You could alternatively return the indices i, j, k themselves, but it's slightly easier to return the numbers $A[i], A[j], A[k]$, and that's totally fine for this problem.

NOTE: You may NOT use a hash map (i.e. dictionary) for this problem because we have not covered those and also hash-maps are randomized and so cannot give worst-case guarantees.

5 Problem 4 – EXTRA CREDIT – 15 points

Given an array A, for any pair of indices $j > i$, define the interval $A[i...j]$ to be the subarray consisting of element $A[i], A[i+1], ..., A[j-1], A[j]$. Now, define sum(i,j) to be the sum of all the values in $A[i...j]$; that is, sum(i,j) = $\sum_{k=i}^{j} A[k]$. Note that it is easy to compute sum(i,j) in time $\Theta(j - i)$ by doing a single pass over $A[i...j]$.

Now, consider the following problem:

Maximum Interval Value

- Input: an array A of length n with positive and negative numbers.
- Output: the maximum possible value sum(i,j)

For example, if A = 3,-5,4,-2,-1,4,-3,2, then the maximum interval value is 5, obtained via the interval [4,-2,-1,4]

Now, consider the following naive algorithm for this problem:
Algorithm:

1. For each index i

```
2.     For each index $j > i$
3.         Compute and store sum(i,j) in time $\Theta(j - i)$.
4. Return the maximum value sum(i,j) computed in step 3.
```

Question: Prove that the total running time of the algorithm is $\Theta(n^3)$. You need to argue that the number of steps is *at most* a constant times n^3 (easier part), and *at least* a constant times n^3 (harder part).