Quiz #1
Algorithms I
600.463

Thursday, March 16th, 12-1.15pm

Name: Alan **Class:** 600.463.01/600.463.02 (circle one) JHU 2017 S

Ethics Statement

I agree to complete this exam without unauthorized assistance from any person, materials, or device.

Signature Date

Brief solns

1 Problem 1 (10 points: each subproblem is 2 points)

For each statement below state if it is true or false.

1. If $f(n) = \Theta(g(n))$, then $g(n) = \omega(f(n))$.

   true false

2. If $f(n) = O(g(n))$ and $g(n) = O(f(n))$, then $f(n) = g(n)$.

   true false

3. $n^{\log n} = O(n^4)$

   true false

4. $n \ln(n^{100}) = o(n^2)$

   true false

5. The running time of the insertion sort is in $\Omega(n)$.

   true false

2 Problem 2 (20 points; each subproblem is 10 points)

Give asymptotic upper bounds for the following recurrences. You can use Master theorem, when it is applicable. Assume that $T(0) = T(1) = 1$.

1. $T(n) = 8T(n/2) + 100n^2 + n$

$a = 8$, $b = 2$, $f(n) = O(n^2)$

$\because \log_b a = \log_2 8 = 3$

$\therefore n^3 > n^2 \Rightarrow$ master theorem case 1

$\Rightarrow T(n) = O(n^3)$

2. $T(n) = 2T(\sqrt{n}) + \log^2 n$

by substitution

let $n = 2^m$

$\Rightarrow T(2^m) = 2T(2^{\frac{m}{2}}) + m^2$

let $S(m) = 2S(\frac{m}{2}) + m^2$

from master theorem $a = 2$, $b = 2$

$S(m) = O(m^2)$

$\Rightarrow T(n) = O(\log^2 n)$

3 Problem 3 (60 points)

Let A and B be two sorted arrays of n elements each. We can easily find the median element in A — it is just the element in the middle —and similarly we can easily find the median element in B. (For any k, define the median of $2k$ elements as the element that is greater than $k-1$ elements and less than k elements.) However, suppose we want to find the median element overall— i.e., the n-th smallest in the union of A and B. Devise an efficient algorithm to find the median in the union of A and B. You may assume that all elements in A and B are distinct.

25 points will be given if (1) your algorithm works correctly, (2) your algorithm solves the problem by using the time worse than $O(\log n)$, (3) your analysis is correct and (4) your explanations and proofs are clear and with enough details. If you cannot prove your claims formally, give your best intuition.

The full credit will be given if (1) your algorithm works correctly, (2) your algorithm solves the problem in $O(\log n)$ time, (3) your analysis is correct and (4) your explanations and proofs are clear and with enough details. If you cannot prove your claims formally, give your best intuition.

Algorithm: Since A and B are sorted, we can access their medians directly ($O(1)$ time)

let m_A be A's median, m_B be B's median

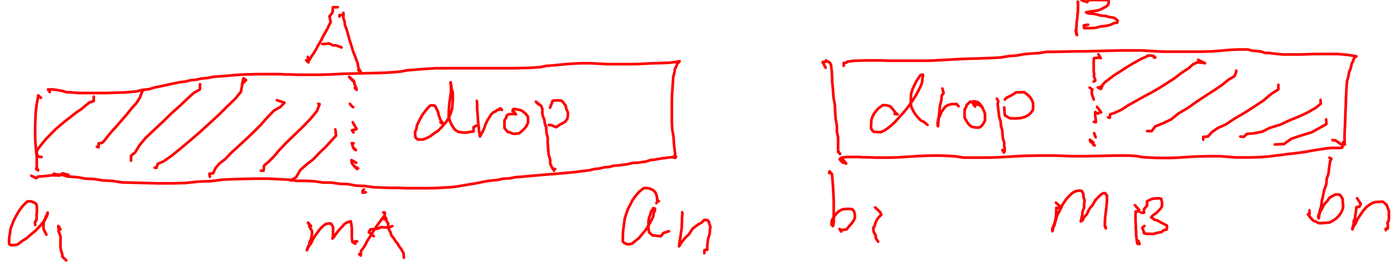
consider three cases:

① If $m_A = m_B$, return m_A/m_B as result.

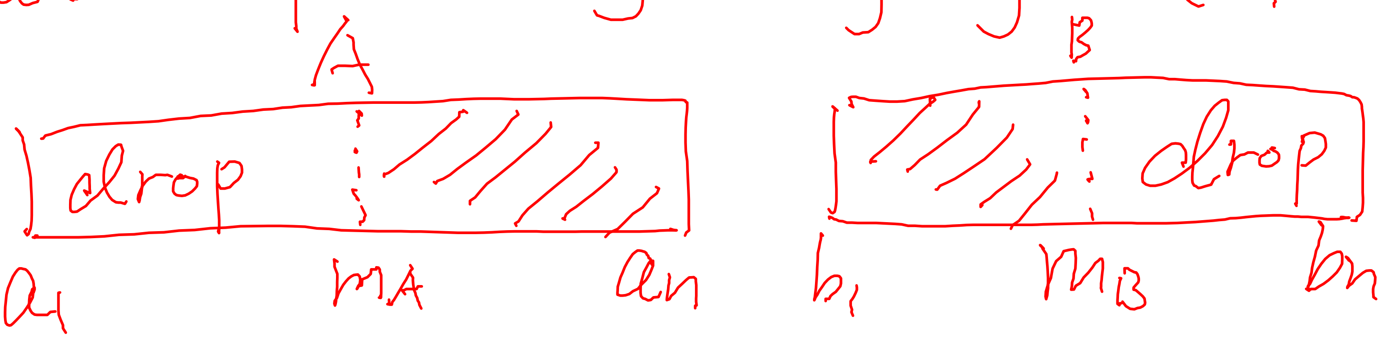
② If $m_A > m_B$, the real median is in the range from m_B to m_A

Let's keep the right half of B ($m_B \cdots b_n$)

and keep the left half of A ($a_1 \cdots m_A$)



③ If $m_A < m_B$ the real median is in the range from m_A to m_B. let's keep the right half of $A(m_A \cdots b_n)$ and keep the left half of $B(b_1 \cdots m_B)$



The algo keeps finding the medians in the subarrays of A and B, and compare A, B in the above 3 cases, until finding the real median.
Each time accessing the medians need O(1) time. and in total will need $\log_2 n$ times atmost. So running time is $O(\log n)$

4 Problem 4 (60 points)

You are given t types of coin denominations of values $C_1 < C_2 < \cdots < C_t$, where $C_1 = 1$ and all values are positive integers. Give an efficient dynamic programming algorithm to make change for an amount of money A with as few coins as possible, assuming that you have infinite supply of each type of the coins.

For example, given $A = 20$ and the coin set is $\{1, 5, 10, 15\}$, a minimum of 2 coins are required.

25 points will be given if (1) your DP algorithm works correctly, (2) your algorithm solves the problem by using the time worse than $O(At)$, (3) your analysis is correct and (4) your explanations and proofs are clear and with enough details. If you cannot prove your claims formally, give your best intuition.

The full credit will be given if (1) your DP algorithm works correctly, (2) your algorithm solves the problem in $O(At)$ time, (3) your analysis is correct and (4) your explanations and proofs are clear and with enough details. If you cannot prove your claims formally, give your best intuition.

A DP sol with 1-D array.

Recursive formula: If $A=0$, minCoin $=0$

If $A>0$, $\text{minCoin}(\{C_1 \cdots C_t\}, A)$

$$= \min_i \{\text{minCoin}(A - C_i) + 1\}$$

for any i in $1 \cdots t$ and $C_i \leq A$

Thus, let's define an array T to setup our Dp algo.

```
Int minCoin(C[1....t], A)
{   t ← C[].size()
    T[0 .... A] // init an array
    T[0] ← 0 // base case
    T[1...A] ← Max_INT
    for i = 1 to A
        for j = 1 to t
        { if ( C[j] <= i )
              { tmp ← T[i - C[j]]
                if (tmp != Max_INT
                and tmp+1 < T[i])
                    T[i] ← tmp + 1
              }
        }
    return T[A]
```

proof: by Induction (eliminated here)

time: $O(At)$ from two for loops,