CSE 201
Analysis of Algorithms
Winter 2021
Midterm Exam 2 Solutions

1. (25 Points) The following recursive algorithm determines whether a sub-array $A[p \cdots r]$ is sorted. The variables B_1, B_2 and B_3 are Boolean, and $\wedge$ represents the Logical And operator.

```
Sorted(A, p, r)   precondition: r ≥ p
-----------------------------------------------
1. if r = p
2.     return True
3. else
4.     q = ⌊(p + r)/2⌋
5.     B1 = Sorted(A, p, q)
6.     B2 = Sorted(A, q + 1, r)
7.     B3 = (A[q] ≤ A[q + 1])
8.     return (B1 ∧ B2 ∧ B3)
```

a. (15 Points) Use induction on $m = \text{length}(A[p \cdots r])$ to prove correctness of the above algorithm, i.e. prove that $\text{Sorted}(A, p, r)$ returns True if and only if $A[p \cdots r]$ is sorted in increasing order

Proof:

I. Let $m = 1$. Then $\text{length}(A[p \cdots r]) = r - p + 1 = 1 \Rightarrow r = p$, and True is returned on line 2 of the algorithm. Indeed, an array of length 1 is always sorted, so the algorithm returns the correct value, establishing the base case.

II. Let $m > 1$ and assume Sorted() returns a correct value (True or False) on all sub-arrays of length less than m. We must show that Sorted() returns a correct value when run on any sub-array of length m. Since $1 < m = r - p + 1 \Rightarrow r > p$, line 2 is skipped and lines 4-8 are executed.

Also $p < \frac{p+r}{2} < r \implies p \leq q < r$, and therefore

$$\text{length}(A[p \cdots q]) = q - p + 1 < r - p + 1 = m$$

and

$$\text{length}(A[q + 1 \cdots r]) = r - (q + 1) - 1 = r - q \leq r - p < r - p + 1 = m$$

The induction hypothesis guarantees that lines (5) and (6) return correct values for sub-arrays $A[p \cdots q]$ and $A[q + 1 \cdots r]$, respectively. Observe $A[p \cdots r]$ is sorted in increasing order if and only if: $A[p \cdots q]$ is sorted, $A[q + 1 \cdots r]$ is sorted and $A[q] \leq A[q + 1]$. Thus $A[p \cdots r]$ is sorted if and only if the value of the Boolean expression $B_1 \wedge B_2 \wedge B_3$ returned on line (8) is True. Therefore, Sorted(A, p, r) returns True if and only if $A[p \cdots r]$ is sorted in increasing order, as required. ■

b. (10 Points) Let $T(n)$ denote the number of array comparisons performed by Sorted() on an array of length n. Write a recurrence relation for $T(n)$. Determine a tight asymptotic bound for $T(n)$.

Solution:
If $p = 1$, $r = n$, and $q = \lfloor (n+1)/2 \rfloor$ then $\text{length}(A[1 \cdots q]) = q = \lfloor (n+1)/2 \rfloor = \lceil n/2 \rceil$, and $\text{length}(A[q+1 \cdots n]) = n - q = n - \lceil n/2 \rceil = \lfloor n/2 \rfloor$. Therefore $T(n)$ must satisfy the recurrence

$$T(n) = \begin{cases} 0 & n = 1 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1 & n \geq 2 \end{cases}$$

We first simplify the recurrence to $T(n) = 2T(n/2) + 1$, and compare $1 = n^0$ to $n^{\log_2(2)} = n^1$. Let $\epsilon = 1 - 0 = 1$. Then $\epsilon > 0$ and $1 = O(n^0) = O\left(n^{\log_2(2) - \epsilon}\right)$, and by case (1) of the Master Theorem, we have $T(n) = \Theta(n)$. ■

Alternative Solution:
One can show directly that $T(n) = n - 1$ is an exact solution to this recurrence. First note that when $n = 1, T(1) = 0$. If $n \geq 1$ then

$$\begin{aligned} \text{RHS} &= T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1 \\ &= (\lfloor n/2 \rfloor - 1) + (\lceil n/2 \rceil - 1) + 1 \\ &= (\lfloor n/2 \rfloor + \lceil n/2 \rceil) - 1 \\ &= n - 1 \\ &= T(n) \\ &= \text{LHS} \end{aligned}$$

so $T(n) = n - 1$ solves the recurrence, and $T(n) = \Theta(n)$. ■

2. (25 Points) Let k be an integer in the range $16 < k < 64$, and assume there exists a method for multiplying two 4×4 matrices by performing sums and products of the matrix elements, and in which only k of the operations are products (not assumed to be commutative.)

   a. (10 Points) Explain how this method can be used to recursively multiply two $n \times n$ real matrices, where n is an exact power of 4. (You need not write pseudo-code, a verbal description will suffice.)

      Solution:
      Regard an $n \times n$ square matrix (where n is a power of 4) as a 4×4 matrix, each of whose 16 elements is a square submatrix of size $\frac{n}{4} \times \frac{n}{4}$. To multiply two $n \times n$ square matrices, we multiply two 4×4 matrices of matrices. We suppose this multiplication can be done by performing only k multiplications of the underlying $\frac{n}{4} \times \frac{n}{4}$ matrices (which are non-commutative operations). Since n is a power of 4, we can recur on this process down to matrices of size 1×1, where the recursion halts. At each level, there are k recursive multiplications of 16 matrices whose size is $1/4^{\text{th}}$ that of the current level. ■

   b. (5 Points) Write a recurrence relation for the running time $T(n)$ of the algorithm you described in (a). (Note that this recurrence will contain k as a parameter.)

      Solution:
      We can write this as either $T(n) = kT(n/4) + \Theta(1)$ or $T(n) = kT(n/4) + \Theta(n^2)$. The first term is the cost of the k recursive calls. In the first recurrence, $\Theta(1)$ is the overhead cost of the current recursive invocation. In the second recurrence, $\Theta(n^2)$ is the cost of the real number additions needed to compute the product. (Note to grader: consider either recurrence to be correct.) ■

   c. (5 Points) Use the Master Theorem to find an asymptotic solution to the recurrence you found in (b). (Note your answer will again depend on k.)

      Solution:
      $T(n) = kT(n/4) + \Theta(1)$:
      Compare $1 = n^0$ to $n^{\log_4(k)}$. Let $\epsilon = \log_4(k) - 0$. Then $\epsilon > 0$, and $1 = O\left(n^{\log_4(k)-\epsilon}\right)$, so by case 1 of the Master Theorem, we have $T(n) = \Theta\left(n^{\log_4(k)}\right)$.

      $T(n) = kT(n/4) + \Theta(n^2)$:
      Compare n^2 to $n^{\log_4(k)}$. Let $\epsilon = \log_4(k) - 2$. Then $\epsilon > 0$ since $k > 16$. Therefore, we have that $n^2 = O\left(n^{\log_4(k)-\epsilon}\right)$. We obtain again by case 1, $T(n) = \Theta\left(n^{\log_4(k)}\right)$. ■

   d. (5 Points) Determine the largest integer k for which $T(n) = o\left(n^{\log_2(7)}\right)$, making your algoithm in (a) better than Strassen's.

      Solution:
      We seek the largest integer k such that $n^{\log_4(k)} = o\left(n^{\log_2(7)}\right)$. Equivalently $\log_4(k) < \log_2(7)$, and hence $k < 4^{\log_2(7)} = 7^{\log_2(4)} = 7^2 = 49$. It follows that $k = 48$. ■

3. (25 Points) Let G be a graph, let x and y be vertices in G, and let

$$p\colon\ x = v_0, v_1, v_2, \dots v_k = y$$

be a shortest x-y path in G. Show that any subsequence of p is also a shortest path joining its two ends. In other words, if $r = v_i$ and $s = v_j$ are any two intermediate vertices with $0 \le i < j \le k$, then the subsequence $r = v_i, \dots, v_j = s$ is a shortest r-s path in G.

Proof:
Let the subsequences p_1, p_2 and p_3 of p be defined by

$$p_1\colon\ x = v_0, \dots, v_i = r$$
$$p_2\colon\ r = v_i, \dots, v_j = s$$
$$p_3\colon\ s = v_j, \dots v_k$$

We must show that p_2 is a shortest r-s path. Assume, to get a contradiction, that G contains an r-s path shorter than p_2, call it p'. Then $\text{length}(p') < \text{length}(p_2) = j - i$, and hence the walk obtained by concatenating p_1, p' and p_3 has length

$$\text{length}(p_1)+\text{length}(p')+\text{length}(p_3) < i + (j - i) + (k - j) = k = \text{length}(p),$$

contradicting that p is a shortest x-y path. This contradiction shows that p_2 is a shortest r-s path in G, as required. ■

4. (25 Points) Suppose we are given an unlimited number of coins in each of the denominations $d = (1, 2, 5, 7, 9)$. We wish to pay $N = 14$ monetary units using the least number of coins. Let $C[i, j]$ denote the minimum number of coins needed to pay j units using only coins in the denominations $(d_1, \dots, d_i)$, where $1 \leq i \leq 5$ and $0 \leq j \leq 14$.

   a. (10 Points) Write a recursive formula for $C[i, j]$. Carefully define boundary values and out-of-bounds values in such a way that $C[i, j]$ is defined for all i and j.

   Solution:

$$C[i,j] = \begin{cases} 0 & i \geq 1 \text{ and } j = 0 \\ \min(C[i-1,j], 1 + C[i, j-d_i]) & i \geq 1 \text{ and } j > 0 \\ \infty & i \leq 0 \text{ or } j < 0 \end{cases}$$

   b. (10 Points) Fill in the following table containing the values of $C[i, j]$.

		j														
i	d	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
2	2	**0**	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**	**5**	**5**	**6**	**6**	**7**	**7**
3	5	**0**	**1**	**1**	**2**	**2**	**1**	**2**	**2**	**3**	**3**	**2**	**3**	**3**	**4**	**4**
4	7	**0**	**1**	**1**	**2**	**2**	**1**	**2**	**1**	**2**	**2**	**2**	**3**	**2**	**3**	**2**
5	9	**0**	**1**	**1**	**2**	**2**	**1**	**2**	**1**	**2**	**1**	**2**	**2**	**2**	**3**	**2**

   c. (5 Points) Use this table to determine *two* optimal solutions to this problem, i.e. two different ways to pay 14 monetary units using the least number of possible coins. Express your solutions by giving a vector $x = (x_1, x_2, x_3, x_4, x_5)$ for which $\sum_{i=1}^{5} x_i d_i = 14$.

   Optimal Solution 1: $\boldsymbol{x = (0, 0, 1, 0, 1)}$**, one 5 unit coin, and one 9 unit coin.**

   Optimal Solution 2: $\boldsymbol{x = (0, 0, 0, 2, 0)}$**, two 7 unit coins.**