CSE 201
Analysis of Algorithms
Winter 2020
Midterm Exam 2 **Solutions**

1. (25 Points) Let k be an integer in the range $9 < k < 27$, and assume there exists a method for multiplying two 3×3 matrices by performing sums and products of the matrix elements, and in which only k of the operations are products (and which product is 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 3. (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 3) as a 3×3 matrix, each of whose 9 elements is a square submatrix of size $\frac{n}{3} \times \frac{n}{3}$. To multiply two $n \times n$ square matrices, we multiply two 3×3 matrices of matrices. We suppose this multiplication can be done by performing only k multiplications of the underlying $\frac{n}{3} \times \frac{n}{3}$ matrices (which are non-commutative operations). Since n is a power of 3, 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 9 matrices whose size is 1/3rd 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 $T(n) = kT(n/3) + \Theta(1)$ or $T(n) = kT(n/3) + \Theta(n^2)$. 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:
   $T(n) = kT(n/3) + \Theta(1)$:
   Compare $1 = n^0$ to $n^{\log_3(k)}$. Let $\epsilon = \log_3(k) - 0$. Then $\epsilon > 0$, and $1 = O\left(n^{\log_3(k)-\epsilon}\right)$, so by case 1 we have $T(n) = \Theta\left(n^{\log_3(k)}\right)$.

   $T(n) = kT(n/3) + \Theta(n^2)$:
   Compare n^2 to $n^{\log_3(k)}$. Let $\epsilon = \log_3(k) - 2$. Then $\epsilon > 0$ since $k > 9$, and so $n^2 = O\left(n^{\log_3(k)-\epsilon}\right)$. Again by case 1 we have $T(n) = \Theta\left(n^{\log_3(k)}\right)$. ■

   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 $n^{\log_3(k)} = o\left(n^{\log_2(7)}\right)$, or equivalently $\log_3(k) < \log_2(7)$. Therefore $k < 3^{\log_2(7)}$, and hence $k = \left\lfloor 3^{\log_2(7)} \right\rfloor = 21$. ■

2. (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 $0 \leq i < j \leq k$, 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

$$\begin{aligned} &p_1\colon \ x = v_0, \dots, v_i = r \\ &p_2\colon \ r = v_i, \dots, v_j = s \\ &p_3\colon \ s = v_j, \dots v_k \end{aligned}$$

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. ■

3. (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 $1 \le i \le 5$ and $0 \le j \le 14$.

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:

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

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

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

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

4. (25 Points) A thief wishes to steal objects $\{1, 2, 3, 4, 5, 6\}$, having values $v[1 \cdots 6] = (5, 5, 9, 4, 4, 12)$ and weights $w[1 \cdots 6] = (1, 4, 3, 4, 1, 6)$, where it is permissible to steal a fraction of an object. His goal is to maximize the total value of the goods stolen $\sum_{i=1}^{6} x_i v_i$, where x_i denotes the fraction of object i to be stolen ($0 \leq x_i \leq 1$ for $1 \leq i \leq 6$). The total weight of the stolen goods $\sum_{i=1}^{6} x_i w_i$ must not exceed the capacity of his knapsack: $W = 9$. Determine an optimal solution to this problem using a greedy strategy, with selection function $f(i) = v_i/w_i$, i.e. order the objects by decreasing value-to-weight ratios, then steel as much of each object as is possible, in that order, never exceeding the capacity of the knapsack. Express your solution as the vector $x = (x_1, x_2, x_3, x_4, x_5, x_6)$, and give the value of this optimal solution.

Solution:
The value to weight ratios are: $(5, 1.25, 3, 1, 4, 2)$. Thus the thief should steal, in order

- All of object 1 (value 5 and weight 1)
- All of object 5 (value 4 and weight 1)
- All of object 3 (value 9 and weight 3)
- $2/3$ of object 6 (value 8 and weight 4)

The solution vector is therefore $x = (1, 0, 1, 0, 1, 2/3)$, with total weight 9 and total value 26. ■