CSE548, AMS542: Analysis of Algorithms, Spring 2019 **Date:** March 13, 2019

Midterm Exam

(2:30 PM – 3:45 PM : 75 Minutes)

- This exam will account for either 15% or 30% of your overall grade depending on your relative performance in the midterm and the final. The higher of the two scores (midterm and final) will be worth 30% of your grade, and the lower one 15%.
- There are three (3) questions, worth 75 points in total. Please answer all of them in the spaces provided.
- There are 16 pages including four (4) blank pages and two (2) pages of appendices. Please use the blank pages if you need additional space for your answers.
- The exam is *open slides* and *open notes*. But *no books* and *no computers*.

Good Luck!

Question	Pages	Score	Maximum
1. A Broken ATM	2–4		25
2. Hops	6–9		25
3. Recurrences with Triangular Numbers	11–12		25
Total			75

Name: ______________________________

Question 1. [**25 Points**] **A Broken ATM.** This question is about an ATM (Automated Teller Machine) that can store dollar bills of exactly n different integral values, but when a customer tries to withdraw cash the machine fails unless it can output the amount using exactly k bills, where both n and k are positive integers. We assume that the value of the largest bill the machine stores is not more than cn for some constant $c \geq 1$. We also assume that before each transaction the machine will have at least k bills of each of the n different dollar values it stores (i.e., it will be refilled as soon as the number of bills of any value drops below k).

Now the question is: with any given n and k as above, how many distinct cash amount the ATM can successfully deliver?

1(a) [**5 Points**] Show that for any given k you can output all distinct withdrawal amounts the ATM can successfully deliver in $\mathcal{O}\left(n^2k^2\right)$ time. For example, if the ATM stores only \$5, \$10, \$20 and \$50 bills and $k = 2$, then it can fulfill the following 10 distinct withdrawal amounts:

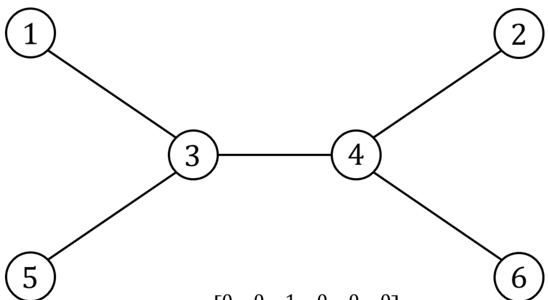
1. \$10 (= \$5 + \$5)	**2.** \$15 (= \$5 + \$10)	**3.** \$20 (= \$10 + \$10)	**4.** \$25 (= \$5 + \$20)	**5.** \$30 (= \$10 + \$20)
6. \$40 (= \$20 + \$20)	**7.** \$55 (= \$5 + \$50)	**8.** \$60 (= \$10 + \$50)	**9.** \$70 (= \$20 + \$50)	**10.** \$100 (= \$50 + \$50)

1(b) [**10 Points**] Explain how you will output all distinct withdrawal amounts in $\mathcal{O}\left(n^{1+\epsilon}\right)$ time when $k = 2$, where ϵ is any given positive constant which can be arbitrarily close to zero.

1(c) [**10 Points**] Explain how you will extend your algorithm from part 1(b) to output all distinct withdrawal amounts in $\mathcal{O}\left(nk(n^{\epsilon}+k^{\epsilon})\right)$ time for any given k, where ϵ is a given constant as in part 1(b).

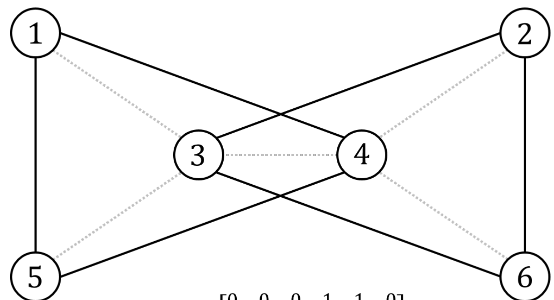
Use this page if you need additional space for your answers.

Question 2. [**25 Points**] **Hops.** Suppose G is an undirected graph that has n vertices. Each vertex of G is identified by a unique integer in $[1, n]$. We say that two vertices u and v of G are adjacent provided they are connected by an edge. All edges of G are recorded in an $n \times n$ adjacency matrix A, where $A[u][v]$ is set to 1 provided vertices u and v are connected by an edge (i.e., provided edge (u, v) exists in G), otherwise $A[u][v]$ is set to 0. Since G is undirected $A[u][v] = A[v][u]$ always holds. We say that vertices u and v are connected by an h-hop path provided v can be reached from u following a path containing exactly h edges and vice versa. An $n \times n$ matrix $D^{(h)}$ which we call an h-hop matrix, records each pair of vertices that are connected by h-hop paths. Entry $D^{(h)}[u][v]$ is set to 1 provided u and v are connected by an h-hop path, and 0 otherwise. Again $D^{(h)}[u][v] = D^{(h)}[v][u]$ for all $u, v \in [1, n]$. Clearly, $D^{(1)} = A$.



$$D^{(1)} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Figure 1: An undirected graph whose edges (i.e., 1-hop paths) are captured by the matrix $D^{(1)}$ which is also the adjacency matrix of this graph.



$$D^{(2)} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Figure 2: The solid edges show the vertices connected by 2-hop paths in the graph on the left. Matrix $D^{(2)}$ marks every pair of vertices connected by 2-hop paths in that graph.

Figure 1 shows an example undirected graph containing 6 vertices and its $D^{(1)}$ matrix which is the same as its adjacency matrix. Figure 2 shows the $D^{(2)}$ matrix for the graph in Figure 1.

```
Iter-Reach ( Z, X, Y )        { X, Y, Z are n × n matrices,
                                where n is a positive integer }
for i ← 1 to n do
    for j ← 1 to n do
        Z[ i ][ j ] ← 0
        for k ← 1 to n do
            Z[ i ][ j ] ← Z[ i ][ j ] ⊕ X[ i ][ k ] ⊗ Y[ k ][ j ]
```

Figure 3: Combining an h_1-hop matrix $X = D^{(h_1)}$ and an h_2-hop matrix $Y = D^{(h_2)}$ to obtain an $(h_1 + h_2)$-hop matrix $Z = D^{(h_1+h_2)}$.

```
Iter-MM ( Z, X, Y )        { X, Y, Z are n × n matrices,
                             where n is a positive integer }
for i ← 1 to n do
    for j ← 1 to n do
        Z[ i ][ j ] ← 0
        for k ← 1 to n do
            Z[ i ][ j ] ← Z[ i ][ j ] + X[ i ][ k ] · Y[ k ][ j ]
```

Figure 4: Multiplying two $n \times n$ matrices X and Y and putting the result in another $n \times n$ matrix Z.

Figure 3 shows an iterative algorithm Iter-Reach that uses bitwise OR ($\oplus$) and bitwise AND

($\otimes$) operators to obtain a new $(h_1 + h_2)$-hop matrix $Z = D^{(h_1+h_2)}$ by combining an h_1-hop matrix $X = D^{(h_1)}$ and an h_2-hop matrix $Y = D^{(h_2)}$.

Observe that Iter-Reach can be obtained from the standard iterative matrix multiplication algorithm Iter-MM shown in Figure 4 simply by replacing the standard addition (+) and multiplication ($\times$) operators with the bitwise OR ($\oplus$) and bitwise AND ($\otimes$) operators, respectively. Both algorithms run in $\Theta\left(n^3\right)$ time.

Now answer the following questions.

2(a) [**8 Points**] Argue that you cannot obtain a $\Theta\left(n^{\log_2 7}\right)$ time algorithm for computing $D^{(h_1+h_2)}$ from $D^{(h_1)}$ and $D^{(h_2)}$ by simply replacing the + and $\times$ operators with $\oplus$ and $\otimes$ operators, respectively, in Strassen's matrix multiplication algorithm given in the Appendix.

$2(b)$ [**10 Points**] Give an $\Theta\left(n^{\log_2 7}\right)$ time algorithm for correctly computing $D^{(h_1+h_2)}$ from $D^{(h_1)}$ and $D^{(h_2)}$ based on Strassen's matrix multiplication algorithm.

2(c) [**7 Points**] For any positive integer n, explain how you will compute $D^{(n)}$ in $\Theta\left(n^{\log_2 7}\log n\right)$ time.

Use this page if you need additional space for your answers.

Question 3. [**25 Points**] **Recurrences with Triangular Numbers.** The k-th *triangular number* $\triangle k$ is defined as follows: $\triangle k = 1 + 2 + \ldots + k$, where k is a natural number. The first few triangular numbers ($\triangle 1$, $\triangle 2$, $\triangle 3$, $\triangle 4$, $\triangle 5$ and $\triangle 6$) are shown in Figure 5 below.

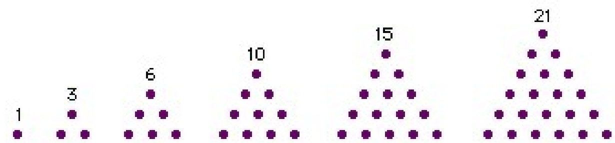


Figure 5: The first 6 triangular numbers.

3(a) [**10 Points**] The time $T(n)$ needed to query a widely used data structure of size n can be described by the following recurrence relation involving triangular numbers:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 6, \\ \sum_{k=2}^{5} \frac{1}{\triangle k} T\left(\frac{kn}{k+1}\right) + \frac{1}{3}T(n) + \Theta(1) & \text{otherwise.} \end{cases}$$

Solve the recurrence for finding an asymptotic tight bound for $T(n)$.

3(b) [**15 Points**] The expected running time $T(n)$ of a randomized algorithm on an input of size n can be described by the following recurrence relation involving triangular numbers $\triangle 2 = 3$, $\triangle 3 = 6$ and $\triangle 4 = 10$:

$$T(n) = \begin{cases} \Theta(n) & \text{if } n \leq 1024, \\ \frac{1}{3} n^{\frac{2}{3}} T\left(n^{\frac{1}{3}}\right) + \frac{1}{6} n^{\frac{5}{6}} T\left(n^{\frac{1}{6}}\right) + \frac{1}{10} n^{\frac{9}{10}} T\left(n^{\frac{1}{10}}\right) + \frac{2}{5} T(n) + \Theta(n \log \log n) & \text{otherwise.} \end{cases}$$

Solve the recurrence for finding an asymptotic tight bound for $T(n)$.

Use this page if you need additional space for your answers.

Use this page if you need additional space for your answers.

Appendix: Recurrences

Master Theorem. Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise,} \end{cases}$$

where, $\frac{n}{b}$ is interpreted to mean either $\left\lfloor \frac{n}{b} \right\rfloor$ or $\left\lceil \frac{n}{b} \right\rceil$. Then $T(n)$ has the following bounds:

Case 1: If $f(n) = \mathcal{O}\left(n^{\log_b a - \epsilon}\right)$ for some constant $\epsilon > 0$, then $T(n) = \Theta\left(n^{\log_b a}\right)$.

Case 2: If $f(n) = \Theta\left(n^{\log_b a} \log^k n\right)$ for some constant $k \geq 0$, then $T(n) = \Theta\left(n^{\log_b a} \log^{k+1} n\right)$.

Case 3: If $f(n) = \Omega\left(n^{\log_b a + \epsilon}\right)$ for some constant $\epsilon > 0$, and $af\left(\frac{n}{b}\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.

Akra-Bazzi Recurrences. Consider the following recurrence:

$$T(x) = \begin{cases} \Theta(1), & \text{if } 1 \leq x \leq x_0, \\ \sum_{i=1}^{k} a_i T(b_i x) + g(x), & \text{otherwise,} \end{cases}$$

where,

1. $k \geq 1$ is an integer constant,
2. $a_i > 0$ is a constant for $1 \leq i \leq k$,
3. $b_i \in (0, 1)$ is a constant for $1 \leq i \leq k$,
4. $x \geq 1$ is a real number,
5. x_0 is a constant and $\geq \max\left\{\frac{1}{b_i}, \frac{1}{1-b_i}\right\}$ for $1 \leq i \leq k$, and
6. $g(x)$ is a nonnegative function that satisfies a polynomial growth condition (e.g., $g(x) = x^\alpha \log^\beta x$ satisfies the polynomial growth condition for any constants $\alpha, \beta \in \Re$).

Let p be the unique real number for which $\sum_{i=1}^{k} a_i b_i^p = 1$. Then

$$T(x) = \Theta\left(x^p\left(1 + \int_1^x \frac{g(u)}{u^{p+1}} du\right)\right).$$

Appendix: Computing Products

Integer Multiplication. Karatsuba's algorithm can multiply two n-bit integers in $\Theta\left(n^{\log_2 3}\right) = \mathcal{O}\left(n^{1.6}\right)$ time (improving over the standard $\Theta\left(n^2\right)$ time algorithm).

Matrix Multiplication. Strassen's algorithm can multiply two $n \times n$ matrices in $\Theta\left(n^{\log_2 7}\right) = \mathcal{O}\left(n^{2.81}\right)$ time (improving over the standard $\Theta\left(n^3\right)$ time algorithm).

Polynomial Multiplication. One can multiply two n-degree polynomials in $\Theta\left(n \log n\right)$ time using the FFT (Fast Fourier Transform) algorithm (improving over the standard $\Theta\left(n^2\right)$ time algorithm).

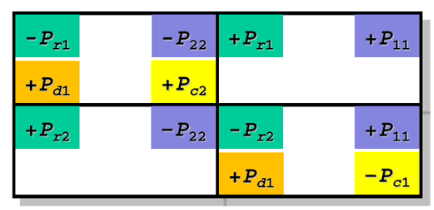
Appendix: Strassen's Matrix Multiplication Algorithm

$$\begin{array}{|c|c|}\hline Z_{11} & Z_{12} \\ \hline Z_{21} & Z_{22} \\ \hline \end{array} = \begin{array}{|c|c|}\hline X_{11} & X_{12} \\ \hline X_{21} & X_{22} \\ \hline \end{array} \times \begin{array}{|c|c|}\hline Y_{11} & Y_{12} \\ \hline Y_{21} & Y_{22} \\ \hline \end{array} = \begin{array}{|c|c|}\hline X_{11}Y_{11} + X_{12}Y_{21} & X_{11}Y_{12} + X_{12}Y_{22} \\ \hline X_{21}Y_{11} + X_{22}Y_{21} & X_{21}Y_{12} + X_{22}Y_{22} \\ \hline \end{array}$$

Sums:

$$\begin{aligned} X_{r1} &= X_{11} + X_{12} & Y_{r1} &= Y_{11} + Y_{12} \\ X_{r2} &= X_{21} + X_{22} & Y_{r2} &= Y_{21} + Y_{22} \\ X_{c1} &= X_{11} - X_{21} & Y_{c1} &= Y_{11} - Y_{21} \\ X_{c2} &= X_{12} - X_{22} & Y_{c2} &= Y_{12} - Y_{22} \\ X_{d1} &= X_{11} + X_{22} & Y_{d1} &= Y_{11} + Y_{22} \end{aligned}$$

$=$



Products:

$$\begin{aligned} P_{11} &= X_{11} \cdot Y_{c2} & P_{c1} &= X_{c1} \cdot Y_{r1} \\ P_{22} &= X_{22} \cdot Y_{c1} & P_{c2} &= X_{c2} \cdot Y_{r2} \\ P_{r1} &= X_{r1} \cdot Y_{22} & P_{d1} &= X_{d1} \cdot Y_{d1} \\ P_{r2} &= X_{r2} \cdot Y_{11} & & \end{aligned}$$

Sums:

$$\begin{aligned} Z_{11} &= -P_{r1} - P_{22} + P_{d1} + P_{c2} \\ Z_{12} &= +P_{r1} + P_{11} \\ Z_{21} &= +P_{r2} - P_{22} \\ Z_{22} &= -P_{r2} + P_{11} + P_{d1} - P_{c1} \end{aligned}$$

Running Time:

$$T(n) = \begin{cases} \Theta(1), & \text{if } n = 1, \\ 7T\left(\frac{n}{2}\right) + \Theta\left(n^2\right), & \text{otherwise.} \end{cases}$$

$$= \Theta\left(n^{\log_2 7}\right)$$

$$= \mathrm{O}\left(n^{2.81}\right)$$