Basic Algorithms (CS 301, Section 1), Professor Yap – Fall 2023

# HOMEWORK

December 7, 2023

## Homework 7 Due: December 13, 2023

**INSTRUCTIONS**:

- Carefully read the standard Homework Instructions (Integrity Policy, justifying answers, etc) in hw1 or hw2.
- This is the last homework.
- ABSOLUTELY NO LATE SUBMISSIONS FOR THIS HOMEWORK. We will publish the solution immediately after due date.

---

**QUESTIONS on Probability, Divide-and-Conquer, FFT, Hashing**

| |
|---|
| These topics are in **Lecture Slides 16, 17, 18, 19, 20, 21**.<br>You may, if you like, look at corresponding chapters in my notes **Lecture VIII, II, XI** (approx)<br>in ClassWiki→Schedule Page. |

---

Q 1. (5+5+10+10 Points) Probability (Simulating Dice with Coins)

Professor X likes to play a dice game in his probability class, but has no dice. To simulate a dice roll, he asks three students to each toss a fair coin, yielding a binary number between 0 and 7. If 0 or 7 are tossed, the three coins are tossed again. The process is repeated until a number between 1 and 6 is tossed.

(a) The Craps Principle says: if $A, B$ are disjoint events, then

$$\Pr(A|\ A \cup B) = \frac{\Pr(A)}{\Pr(A \cup B)}.$$

Note that we do not assume that $A \cup B$ are exhaustive, i.e., $\Pr(A \cup B) < 1$ is OK. This principle can be used to calculate the odds of a casino game called Craps.

(b) Suppose you and I are playing a game in which there is one winner. Let $A$ mean "I win", and $B$ mean "you win". But we could also draw, or perhaps somebody else also playing the game could win. Suppose $\Pr(A) = 0.2$ and $\Pr(B) = 0.3$. Then if we are told that either you or I have won. What is the probability that I had won?

(c) Prove that Professor X's dice-from-coins simulation does produce a fair dice.

(d) What is the expected number of individual coin tosses needed to get a dice roll? Note that the number of coin tosses is always a multiple of 3.
HINT: Let $C$ be the expected number of coin tosses. Write a recurrence for $C$ and solve.

Q 2. (12+4+6+8 Points) Probability (Decision Process)

Consider the following "Decision Process" which consists of a sequence of steps. At each step, we roll a dice that has the usual possible outcomes: $1, 2, 3, 4, 5, 6$. In the $i$-th step, if the outcome is $\leqslant 2i$, we stop. Otherwise, we go to the next step.

For the first step, $i = 1$, we stop iff outcome is 1 or 2. Note that we will surely stop after the 3rd step. Let $X$ be the random variable corresponding to the number of steps.

(a) What is the sample space $\Omega$ and the probability function Pr for this problem?
HINT: draw a DAG (directed acyclic graph) with **root node**, with 6 children $(a_1, a_2, \ldots, a_6)$ corresponding to the possibilities after the first step. The nodes $a_1, \ldots, a_6$ constitute **level 1**. In general, each level has 6 nodes, and edges goes from level $i$ to level $i + 1$. A node either has no children (terminal) or has 6 children of the next level. All the nodes at **level 3** are terminal. Each $\omega \in \Omega$ may be identified with the terminal nodes. We can assign a probability to each node in the DAG. This may be called a **decision DAG**.

(b) Describe the function $X : \Omega \to \mathbb{N}$.

(c) Compute the expected value of $X$.

(d) Compute the variance of $X$.

Q 3. (8+12+10+15+15 Points) Divide-and-Conquer (Simulations)

Figure 2 illustrates how to simulate a divide-and-conquer algorithm like MergeSort: there is a tree representing the "divide" stages, and an inverted tree representing the "conquer" stages. These 2 trees are joined together at their leaves.

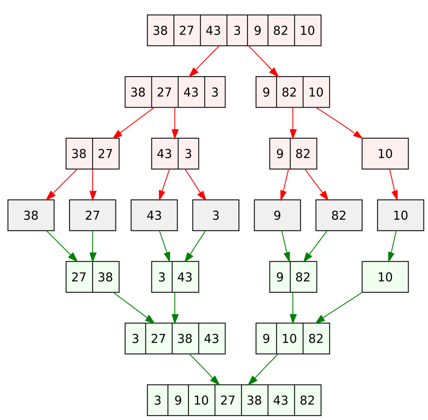


Figure 1: Simulating MergeSort algorithm on $(38, 27, 43, 3; 9, 82, 10)$

(a) (MergeSort) Please simulate MergeSort on the input

$$(5, 7, 3, 11; 35, 27, 46, 12; 1, 4, 8)$$

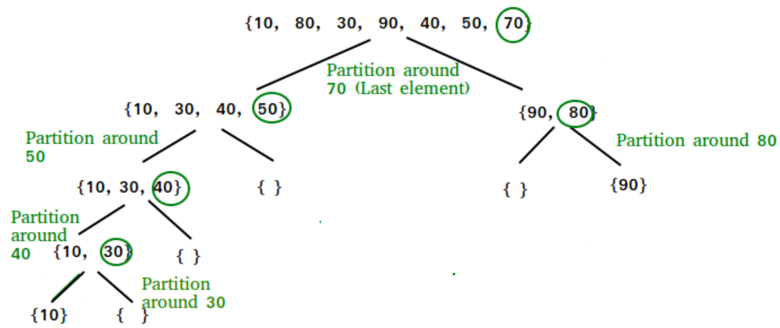


Figure 2

You must use the following rule to "divide" a problem of size $n$: the first recursive call has size $\lceil n/2 \rceil$, the second has size $\lfloor n/2 \rfloor$.

(b) (QuickSort) The QuickSort algorithm works by first calling a subroutine called **partition**. It first picks an arbitrary element $p$ from the input; call this the **pivot element**. Then it splits the input into two lists, $L$ and $R$ where $L$ contains those elements smaller than $p$, and $R$ contains those larger than $p$. For simplicity, assume the elements are distinct. It recursively sorts $L$ and $R$. See Figure 3 for simulation of QuickSort in which we always pick the last element of the array as the pivot.

Notice that we do not have a "conquer" stage in Figure 3. That is because we assume that the input is an array $A[0..n) = A[0..n-1]$, and the **partition** subroutine re-shuffles array $A$ so that $L$ is in $A[0..i)$ and $R$ is in $A[i+1..n)$ and $A[i] = p$. After the recursive sorting of $A[0..i)$ and $A[i+1..n)$, the entire array is already sorted! We call this an **in place** algorithm because we do not need extra array, as the elements always remain in the original. are Also assume that this partition subroutine is **stable** (that means, the relative ordering of elements in $A[0..i)$ should be the same as their original ordering, and similarly for $A[i+1..n)$. All the recursive calls of QuickSort are on subarrays of the form $A[i..j)$ where $0 \leqslant i \leqslant j \leqslant n$.

Please simulate QuickSort on the input

$$(5, 7, 3, 11; 35, 27, 46, 12; 1, 4, 8) \tag{1}$$

(c) The real Quicksort algorithm picks the pivot $p$ randomly. Then we can prove that its expected running time is $O(n \log n)$. *But for our simulation purposes*, we always *pick the last element of* $A[i..j)$ *as pivot*. Please re-order the elements of (1) so that our Quicksort simulation has the worst possible performance. Assume that partition on $A[i..j)$ takes time $O(j - i + 1)$.

(d) Same as previous question, but re-order the elements so that the QuickSort performance is the best possible. Please explain how you created this answer.

(e) We had explained how to simulate Karatsuba's multiplication of Integers in class. This is illustrated in Figure 4.

Please simulate Karatsuba's algorithm for multiplying $X = 123456$ and $Y = 78965$.

Q 4. (4x5+10 Points) Master Theorem and Solving Recurrences

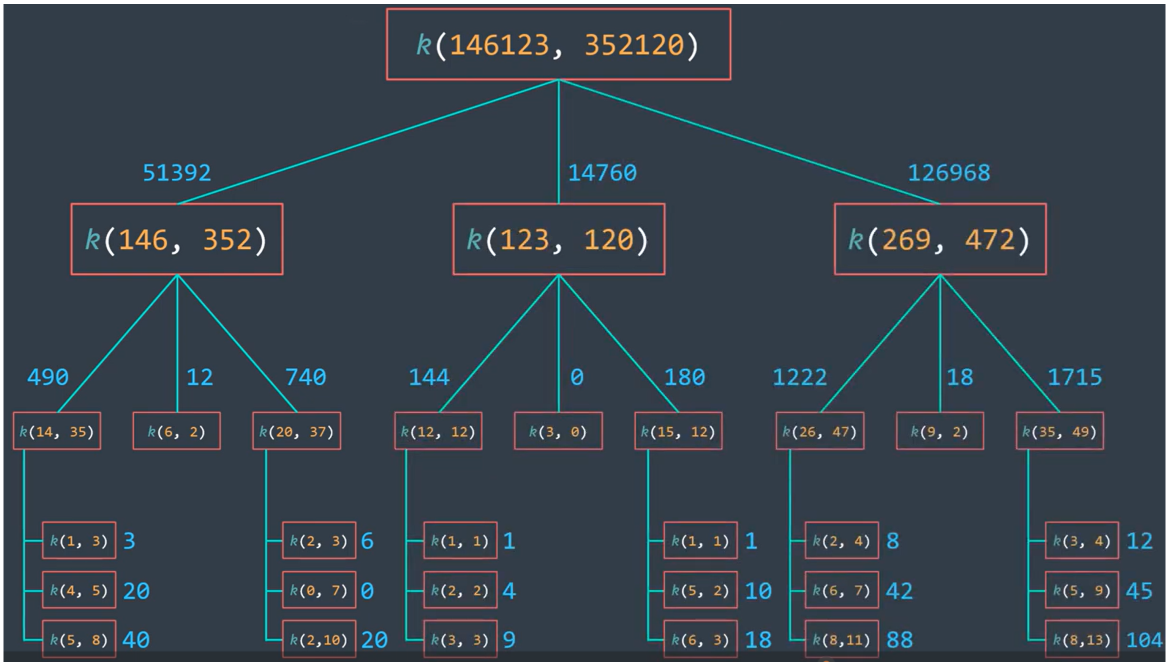


Figure 3

Please solve the following recurrences. Use the Master Theorem whenever possible.

(a) $T(n) = 2T(\frac{n}{4}) + 1$.

(b) $T(n) = 2T(\frac{n}{4}) + \sqrt{n}$.

(c) $T(n) = 2T(\frac{n}{4}) + n$.

(d) $T(n) = 2T(\frac{n}{4}) + n^2$.

(e) $T(n) = 2T(n-4) + n$.

PLEASE NOTE: To get full credit, when you use the Master Theorem, you MUST explicitly state the values of the constants $a, b, e, f$ of the Master Theorem.

Q 5. (8+6+4+10+12+15 Points) FFT over Finite Fields

We want to multiply two polynomials $f(x), g(x) \in F[x]$ where $F$ is a field. Assume $f(x), g(n)$ have degrees $< n$ where $n$ is a power of 2. To use the FFT algorithm, we need an $\omega \in F$ that is a primitive $n$th root of unity. In this question, $F = \mathbb{Z}_{17}$ and we let $\omega = 3$ be a primitive 16th root of unity (see page 9, `18-polynom-fft.pdf`).

(a) To do the inverse FFT, we need the inverse $\omega^{-1}$. Determine $\omega^{-1}$ from $\omega$ using the extended Euclidean algorithm. Show your steps.

(b) Please determine a primitive 8th root of unity. Determine a primitive 4th root of unity in $\mathbb{Z}_p$.

(c) Let $f(x) = 1 + x + x^2 + x^3$ and $g(x) = 2x + 3x^4 + 5x^6$. Please comupte $h(x) = f(x)g(x)$ (using High School Algorithm).

(d) Please evaluate $f(x)$ at the points $\omega^0, \omega^2, \omega^4, \omega^6, \dots, \omega^7$ using the FFT algorithm. Please show your simulatation.

(e) Suppose $h(x) = c_0 + c_1x + c_2x^2 + c_3x^3$ is unknown. But you are told that the evaluation of $h(x)$ at each of the points $(\omega^0, \omega^4, \omega^8, \omega^{12})$ is $(1, 1, 1, 1)$. What is $h(x)$? NOTE: we are asking to solve the polynomial interpolation problem. You need the inverse FFT.

(f) Simulate the FFT multiplication the polynomials $f(x), g(x)$ from part(c). Note that this should give the same answer as part(c).

Q 6. (20+25+5 Points) Multiplying Polynomials (FFT implementation)

Please study the Java program `PolyMul.java` that we provide for multiplying two polynomials using the FFT algorithm. This is a fully working program, except that we have body of the `FFT(...)` method for you to fill in.

(a) `PolyMul.java` is able to multiply two integer polynomials of very high degrees (but there is a limit). Please explain how we do it.

(b) Please do the programming part of this assignment: basically, you must fill in the body of the method called `FFT(int[] a, int w, int p)`. This method evaluates the input polynomial (represented by array `a`) at the powers of the primitive root `w` (mod `p`). If you do it correctly, then running the program in modes `mm=1,2,3` will reproduce an output that looks like the provided files `TESTOUTPUT-mm`.

(c) Given $f, g \in \mathbb{Z}[x]$, let $h = f \cdot g \in \mathbb{Z}[x]$ denote their product as integer polynomials. Does our program really compute $h$? Explain.

Q 7. (20+10 Points) Perfect Hashing

Assume a hash family $\mathcal{H} = \{h_\lambda : \lambda \in \Lambda\}$ that is **universal**, and $h_\lambda : \mathcal{U} \to \mathbb{Z}_m$ where $m \geqslant n(n-1)$. Given a set $K \subseteq \mathcal{U}$, we want to find a **perfect hash function** $h \in \mathcal{H}$ for $K$. "Perfect" means $h$ has no collisions on $K$.

Please refer to slides `21-hash3.pdf`. On page 2, we stated a result called the **Union Bound**:

$$\Pr[collision] \leqslant \frac{n(n-1)}{2m}.$$

The slides suggest this "Strategy"

> Repeat
>     Choose a random hash function $h$
>     If $h$ is perfect for $K$, return $h$

to find a perfect hash function for

$$K = \{a_1, \ldots, a_n\}.$$

(a) Use pseudo-code to flesh out the above "Strategy" a little more, writing a method called

$$\texttt{perfect}(K, H, \Lambda)$$

where $|K| = n$, $H$ is a 2-argument function $H(\lambda, a)$ such that for all $\lambda \in \Lambda$ and $a \in \mathcal{U}$, we have $H(\lambda, a) := h_\lambda(a)$. The method `perfect`$(K, H, \Lambda)$ return a $\lambda \in \Lambda$ such that $h_\lambda$ is perfect for $K$.

(b) Let $T$ be the number of iterations inside the Repeat-loop. Note that $T$ is a random variable. What is the expected value of $T$? **HINT**: set up a recurrence for $T$.