Algorithms M2–Final Exam 2020–2021

October 26, 2021

Instructions

1. When an exercise asks for an algorithm, you may give pseudocode or an informal description (as long as the description is clear enough).
2. You must give a correctness proof and explicitly state the complexity of all your algorithms.
3. You may give deterministic or randomized algorithms. For your randomized algorithms you must calculate the probability of error.
4. You may use all algorithms we have seen in class (for example, sorting algorithms), without proof.
5. In the last page you will find an appendix with some useful reminder.
6. This exam has 6 exercises.

1 Recurrence relations

Calculate the big-Oh complexity of the following functions. Give a justification for all your answers (using the Master Theorem or otherwise). (2 points)

1. $T_1(n) = n^2 + T_1(2n/3)$
2. $T_2(n) = 5T_2(n/3) + n$
3. $T_3(n) = 2T_3(n/4) + \sqrt{n}$
4. $T_4(n) = T_4(n-1) + n^2$

2 Roll the dice

We roll a standard (fair) 6-sided die. Let X be the result (which is a number between 1 and 6). We then roll the die X times. Let Y be the sum of these X rolls. In other words, the experiment we perform is that, we first roll a die, and the result of this first roll tells us how many rolls we must perform to calculate Y.

1. What is $E[X]$? (1 point)
2. What is $E[Y \mid X = 2]$? (1 point)
3. What is $E[Y]$? (1 point)
4. Use Markov's inequality to give an upper bound on $Pr[Y \geq 25]$. (1 point)

3 Multiple merge

Recall the `Merge` algorithm we have seen in class which, given two sorted arrays A, B, produces a sorted array that contains the union of the elements of A, B. In this exercise we consider a generalization of this problem.

We are given k *sorted* arrays $A_1, \ldots, A_n$, each containing n numbers (so, we have kn numbers in total). We would like to produce a merged array that contains all kn numbers in sorted order. Consider the following algorithm:

```
Res = A1
for i=2 to k
 Res = Merge(Res, Ai)
return Res
```

In other words, this algorithm will first merge A_1 with A_2, then merge the result with A_3, then merge the result with A_4, etc.

1. What is the complexity of this algorithm? (1 point)
2. Describe a divide&conquer algorithm for this problem which is more efficient than the previous algorithm, and also more efficient than placing all elements in an array and then calling `Mergesort`. (2 points)

4 A Tennis Tournament

We have n tennis players, numbered $1, \ldots, n$. Each player i has a "score" s_i which determines her ability, in the sense that, if $s_i > s_j$, this means that player i will definitely win a match against player j. The scores are unknown to us. The only way to gain information about

the players' abilities is to have them play against each other, but in this case we only learn who has the higher score.

Given the collection of n players, we want to organize as few matches as possible in a way that allows us to find out who are the best players.

1. Give an algorithm which in $O(n)$ matches finds out who is the best player. (1 point)
2. Give an algorithm which in $O(n)$ matches finds out who are the $\sqrt{n}$ best players and their ordering. (2 points)

5 Calculating Probabilities

We are given an array A of n positive real numbers $a_1, a_2, \ldots, a_n$ such that $\sum_{i=1}^{n} a_i = 1$. The idea is that this array represents the probabilities associated with a die with n sides that is biased. In other words, we are given a die with n sides, with the numbers $1, 2, \ldots, n$, and if we roll the die, the probability that we get i is given by a_i.

Suppose now that we roll the die twice and calculate the sum of the two rolls, which is a number between 2 and $2n$. We would like to calculate an array B such that $B[i]$ is equal to the probability that the sum of the two rolls is i.

Example: if $A = [\frac{1}{2}, \frac{1}{6}, \frac{1}{3}]$ that means we have a die with three sides, such that the probability of rolling $1, 2$ or 3 is respectively $\frac{1}{2}, \frac{1}{6}, \frac{1}{3}$. If we roll the die twice, the sum of the two rolls is between 2 and 6. The corresponding probabilities are given by the array $B = [\frac{1}{4}, \frac{1}{6}, \frac{13}{36}, \frac{1}{9}, \frac{1}{9}]$, which lists the probability that the sum is $2, 3, 4, 5$ or 6 respectively.

1. Give an algorithm for this problem that runs in $O(n^2)$, assuming that arithmetic operations (addition, subtraction, multiplication) take constant time. (1 point)
2. Give an algorithm (under the same assumptions) that runs in $O(n^{\log 3})$. (3 points)

6 Palindromic Substrings and Subsequences

Recall some definitions: given a string s, a **subsequence** of s is a string made up of letters that appear in s, in the same order, but not necessarily consecutively. A **substring** of s is a subsequence of s where the letters do appear consecutively.

For example: if s is the string `abcdabcdab`, then `aaab` is a subsequence of s but not a substring of s; `adcc` is neither a subsequence nor a substring of s; `cdab` is a substring of s.

A string is a palindrome if it reads the same from left to right and right to left. For example `ABBA`, `MADAM` are palindromes, while `ABAB` is not.

1. Describe an efficient algorithm which given a string s of length n finds the longest palindromic **substring** of s. What is the complexity of your algorithm? (2 points)
2. Describe an algorithm which given a string s of length n finds the longest palindromic **subsequence** of s. What is the complexity of your algorithm? (2 points)

Appendix

Probabilities

1. If X is a discrete random variable, $E[X] = \sum_i i \cdot Pr[X = i]$.
2. $Var[X] = E[X^2] - (E[X])^2$.
3. Union bound : $Pr[A \cup B] \leq Pr[A] + Pr[B]$
4. Conditional Probabilities : $Pr[A|B] = \frac{Pr[A \cap B]}{Pr[B]}$
5. Markov : if X is a non-negative random variable, then $\forall \alpha > 0, Pr[X \geq \alpha] \leq \frac{E[X]}{\alpha}$.
6. Chebyshev : $\forall \alpha > 0, Pr[|X - E[X]| \geq \alpha] \leq \frac{Var[X]}{\alpha^2}$.

Master Theorem

If $T(n) = aT(n/b) + O(n^d)$, then

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

Other facts

1. $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
2. $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$