COMP3121/9101
Algorithm Design

PRACTICE PROBLEM SET 1 – DATA STRUCTURES REVISION

[**K**] – key questions [**H**] – harder questions [**E**] – extended questions [**X**] – beyond the scope of this course

Contents

§ SECTION ONE: DATA STRUCTURES AND ALGORITHMS

[K] Exercise 1. Let A be an array with $n-1$ elements, containing all integers from 1 to n except for one. Design an $O(n)$ algorithm that finds the missing integer.

[K] Exercise 2. Let A be an array with n elements. We say that A is *palindromic* if it can be read the same forwards and backwards. For example, the array $A = [1, 2, 3, 2, 1]$ is palindromic while $B = [1, 2, 3, 4, 2, 1]$ is not. Design an $O(n)$ algorithm that determines whether A is a palindromic array.

[K] Exercise 3. Let A be an array with n distinct integers. You have to determine if there exist an integer (not necessarily in A) which can be written as a sum of squares of two distinct integers from A in two different ways. Note that $A[i]^2 + A[j]^2$ is treated the same as $A[j]^2 + A[i]^2$.

For example, if $A = [1, 8, 9, 12]$, then the integer 145 can be written as $1^2 + 12^2$ *and* also $8^2 + 9^2$.

(a) Design an $O(n^2 \log n)$ algorithm that determines if such an integer exists in the *worst case*.

(b) Design an algorithm that solves the same problem and runs in $O(n^2)$ in the *expected case*.

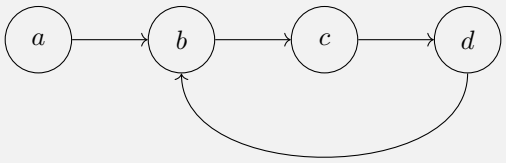
[K] Exercise 4. Let A be an array with n integers, and let k be a positive integer. You have to determine if there exist two integers in A whose absolute difference is exactly k. In other words, you want to determine if there exist distinct indices i, j such that $|A[i] - A[j]| = k$.

(a) Design an $O(n \log n)$ algorithm that determines if two such integers in A exist.

(b) Design an algorithm that solves the same problem and runs in $O(n)$ in the *expected case*.

[K] Exercise 5. Let L be a linked list with n nodes. We say that L is *cyclic* if there is a node that can be reached again by continuously following the next node.

For example, the linked list visualised as



is cyclic because node b can be reached again from d. Design an $O(n)$ algorithm that determines whether a linked list is cyclic.

[H] Exercise 6. You are at a party attended by n people (not including yourself), and you suspect that there might be a celebrity present. A *celebrity* is someone known by everyone, but who does not know anyone else present. Your task is to work out if there is a celebrity present, and if so, which of the n people present is a celebrity. To do so, you can ask a person X if they know another person Y (where you choose X and Y when asking the question).

(a) Show that there can be at most one celebrity. In other words, *if* a celebrity exists, then the celebrity is unique.

(b) Use the previous part to show that your task can always be accomplished by asking no more than $3n - 3$ such questions.

(c) Show that your task can always be accomplished by asking no more than $3n - \lfloor \log_2 n \rfloor - 3$ such questions.

[H] Exercise 7. You are in a square orchard of $4n$ by $4n$ equally spaced trees. You want to purchase apples from precisely n^2 many of those trees, which also form a square. Fortunately, the owner is allowing you to choose such a square anywhere in the orchard and you have a map with the number of apples on each tree. Your task is to choose a square that contains the largest amount of apples and which runs in time $O(n^2)$.

[E] Exercise 8. There are n teams in the local cricket competition and you happen to have n friends that keenly follow it. Each friend supports some subset (possibly all or none too) of the n teams. Not being the sporty type, but wanting to fit in nonetheless, you must decide for yourself which subset of teams (again, possibly all or none too) to support.

You don't want to be branded as a copy cat so your subset must not be identical to anyone else's. The trouble is, you don't know which friends support which teams but you can ask your friends some question of the form: "*does friend A support team B?*" (you choose A and B in advance).

Design a strategy that determines a suitable subset of teams for you to support and asks the fewest number of questions as possible.

> Given your n friends, how many questions do you have to at least ask each friend? Can we generate a strategy just asking these questions?

[E] Exercise 9. You are conducting an election among a class of n students. Each student casts precisely one vote by writing their name and that of their chosen classmate on a single piece of paper. However, the students have forgotten to specify the order of names on each piece of paper; for example, "Alice Bob" could mean that *Alice voted for Bob* or *Bob voted for Alice*.

(a) Show how you can still uniquely determine how many votes each student received.

(b) Hence, explain how you can determine which students did not receive any votes. Can you also determine who these people voted for?

(c) Suppose now that every student received at least one vote. Show that each student received *exactly* one vote.

(d) Using the previous two parts, design an $O(n)$ algorithm that constructs a list of votes of the form "X voted for Y" consistent with the pieces of paper. Specifically, each piece of paper should match up with precisely one of these votes. If multiple such lists exist, produce any.

> First, use **(c)** to consider how you would solve it in the case where every student received at least one vote. Then apply **(b)**.

§ SECTION TWO: SEARCHING AND SORTING ALGORITHMS

[K] Exercise 10.

(a) Assume you have an array of $2n$ distinct integers. Find the largest and the smallest number using at most $3n-2$ comparisons.

(b) Assume you have an array of 2^n distinct integers. Find the largest and second largest number using at most $2^n + n - 2$ comparisons.

[K] Exercise 11. You are given an array A of n integers and an integer x.

(a) Design an $O(n \log n)$ algorithm that determines whether or not there exist two integers in A that sum to x.

(b) Design an algorithm that solves the same problem and runs in $O(n)$ **expected** time.

[K] Exercise 12. Let $f : \mathbb{N} \to \mathbb{Z}$ be a monotonically increasing function. That is, for all $i \in \mathbb{N}$, we have that $f(i) < f(i+1)$. Our goal is to find the smallest value of $i \in \mathbb{N}$ so that $f(i) \geq 0$. Design an $O(\log n)$ algorithm to find the value of i so that $f(i) \geq 0$ for the first time.

> How could you use binary search here?

[K] Exercise 13. Let M be an $n \times n$ matrix of distinct integers $M(i,j)$ where $1 \leq i, j \leq n$. Each row and each column of the matrix is sorted in increasing order, so that for each row i,

$$M(i,1) < M(i,2) < \cdots < M(i,n)$$

and for each column j,

$$M(1,j) < M(2,j) < \cdots < M(n,j).$$

Design an $O(n)$ algorithm that, given an integer x, determines whether M contains the integer x.

[H] Exercise 14. You are given two arrays, A and B, each containing n distinct positive integers each. Let $f(x,y) = y^6 + x^4y^4 + x^2y^2 - x^8 + 10$.

(a) For any fixed value x, show that $f(x,y)$ is an increasing function in terms of y.

(b) Hence, design an $O(n \log n)$ algorithm that determines if A contains a value for x and B contains a value for y such that $f(x,y) = 0$.

[H] Exercise 15. Let A be an array with n integers. You need to answer a series of n queries, each of which is of the form "*how many elements a of the array A satisfy $L_k \leq a \leq R_k$?*", where L_k, R_k (for some $1 \leq k \leq n$) are integers such that $L_k \leq R_k$. Design an $O(n \log n)$ algorithm that answers each of these n queries.

[H] Exercise 16. Suppose that you are taking care of n kids, who took their shoes off. You have to take the kids out and it is your task to make sure that each kid is wearing a pair of shoes of the right size (not necessarily their own, but one of the same size). All you can do is to try to put a pair of shoes on a kid, and see if they fit, or are too large or too small; you are NOT allowed to compare a shoe with another shoe or a foot with another foot. Describe an algorithm

whose expected number of shoe trials is $O(n \log n)$ which properly fits shoes on every kid.

[E] Exercise 17. Your army consists of a line of n giants, each with a certain height. You must designate precisely $\ell \leq n$ of them to be leaders. Leaders must be spaced out across the line; specifically, every pair of leaders must have at least $k \geq 0$ giants standing in between them. Given n, ℓ, k and the heights, $H[1], H[2], \ldots, H[n]$, of the giants in the order that they stand in the line as input, find the *maximum* height of the *shortest* leader among all valid choices of ℓ leaders. We call this the *optimisation* version of the problem.

For example, consider the following inputs: $n = 10$, $\ell = 3$, $K = 2$, and $H = [1, 10, 4, 2, 3, 7, 12, 8, 7, 2]$. Then, among the 10 giants, you must choose 3 leaders so that each pair of leaders has at least 2 giants standing in between them. The best choice of leaders has heights 10, 7 and 7, with the shortest leader having height 7. This is the best possible for this case.

(a) In the *decision* version of this problem, we are given an additional integer T as input. Our task is to decide if there exists some valid choice of leaders satisfying the constraints whose shortest leader has height no less than T.

Give an algorithm that solves the decision version of this problem in $O(n)$ time.

(b) Hence, show that you can solve the optimisation version of this problem in $O(n \log n)$ time.

[E] Exercise 18. You are given n numbers $x_1, \ldots, x_n$, where each x_i is a real number in the interval $[0, 1]$.

(a) Describe an $O(n \log n)$ algorithm that outputs a permutation $y_1, \ldots, y_n$ of the n numbers such that

$$\sum_{i=1}^{n} |y_i - y_{i-1}| < 2.$$

In other words, the sum of the absolute difference between adjacent elements is strictly smaller than 2.

(b) Describe an $O(n)$ algorithm that solves the same problem as the previous question.

Tweak the BucketSort algorithm. What size buckets should we use?

§ SECTION THREE: TIME COMPLEXITY ANALYSIS

[K] Exercise 19. Show the following asymptotic relations by providing suitable constants for c and N.

(a) $n^2 = O(n^3)$.

(b) $4n^4 + 8n^2 + 1 = \Omega(n^3)$

(c) $n^2 + \sin(n) = O(n^2)$.

(d) $\dfrac{2n^2 + 3n + 1}{3n + 1} = \Theta(n)$.

(e) $\cos(n) + \sin(n) = \Theta(1)$.

[K] Exercise 20. Determine if $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$, both (i.e. $f(n) = \Theta(g(n))$) or neither for the following pairs of functions. Justify your answer in each.

(a) $f(n) = (\log_2 n)^2$, $g(n) = \log_2\left(n^{\log_2 n}\right) + 2\log_2 n$.

(b) $f(n) = n^{100}$, $g(n) = 2^{n/100}$.

(c) $f(n) = \sqrt{n}$, $g(n) = 2^{\sqrt{\log_2 n}}$.

(d) $f(n) = n^{1.001}$, $g(n) = n \log_2 n$.

(e) $f(n) = n^{(1+\sin(\pi n/2))/2}$, $g(n) = \sqrt{n}$.

[K] Exercise 21. Let $f(n)$ and $g(n)$ be two positive functions on $\mathbb{N}$. Prove that $f(n) = O(g(n))$ if and only if $g(n) = \Omega(f(n))$.

First show that, if $f(n) = O(g(n))$, then $g(n) = \Omega(f(n))$. Then show that, if $g(n) = \Omega(f(n))$, then $f(n) = O(g(n))$.

[K] Exercise 22. Let $f(x) = x^2 - 12$, $g(x) = x^3$, and $h(x) = 7x - 3$. Show the following asymptotic relations.

(a) $f(x) = O(g(x))$.

(b) $f(x) \cdot h(x) = \Theta(g(x))$.

(c) $\dfrac{1}{f(x)} = O(1)$.

(d) $\dfrac{1}{g(x)} = \Omega(e^{-x})$.

[K] Exercise 23. Let $f(n)$ and $g(n)$ be two positive functions for all $n \in \mathbb{N}$.

(a) Show that, if $\lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} = 0$, then $f(n) = O(g(n))$.

(b) Show that, if $\lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} = \infty$, then $f(n) = \Omega(g(n))$.

[K] Exercise 24. Prove the following Big-Oh properties.

(a) If $c > 0$ is a constant, then $c \cdot f(n) = O(f(n))$. In other words, constants do not affect the growth rate.

(b) If $f(n) = O(g(n))$ and $g(n) = O(h(n))$, then $f(n) = O(h(n))$. In other words, Big-Oh is transitive.

(c) If $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$, then $f_1(n) + f_2(n) = O(g_1(n) + g_2(n))$.

(d) $O(f(n) + g(n)) = O\left(\max\{f(n), g(n)\}\right)$.

[H] Exercise 25. This exercise shows that Big-Oh is not preserved under monotonic functions. Suppose that $f(n) = O(g(n))$, and let $h(n)$ be some monotonic function in n.

(a) Let $h(n) = 2^n$. By considering the derivative of $h(n)$, or otherwise, show that $h(n)$ is monotonic in n.

(b) Consider $f(n) = 2^{n+1}$ and $g(n) = 2^n$. Show that $f(n) = O(g(n))$.

(c) Show that $h(f(n)) \neq O(h(g(n)))$. This shows that Big-Oh is not necessarily preserved under monotonic functions.

[E] Exercise 26. Classify the following pairs of functions by their asymptotic relation; that is, determine if $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$, both (i.e. $f(n) = \Theta(g(n))$) or neither.

(a) $f(n) = n^{\log n}$, $g(n) = (\log n)^n$.

(b) $f(n) = (-1)^n$, $g(n) = \tan(n)$.

(c) $f(n) = n^{5/2}$, $g(n) = \left\lfloor \log\left(\sum_{k=0}^{\infty} \frac{4^{k+n} n^k}{k!}\right)\right\rfloor^2$.

[E] Exercise 27. Let $f : \mathbb{N} \to \mathbb{R}$ be a positive and strictly increasing function; that is, for all $n \in \mathbb{N}$, $f(n) > 0$. Show that $1/f(n) = O(f(n))$.

> Since f is strictly increasing, how large can $f(n)$ grow to and what happens to $1/f(n)$ as $n \to \infty$?

[X] Exercise 28. Define $f : \mathbb{N} \to \mathbb{R}$ by

$$f(n) = \begin{cases} 1 & \text{if } n \leq 2, \\ f\left(\left\lceil \frac{n}{\log_2 n} \right\rceil\right) + n & \text{if } n \geq 3. \end{cases}$$

Show that $f(n) = \Theta(n)$.

> - Show that $f(n) \leq 2n$ for $n \geq 512$.
> - Show that $f(n) \geq n$ for $n > 2$.