Algorithms Tutorial 1
Problems

1. You are given an array S of n integers and another integer x.

   (a) Describe an $O(n \log n)$ algorithm (in the sense of the worst case performance) that determines whether or not there exist two elements in S whose sum is exactly x.

   (b) Describe an algorithm that accomplishes the same task, but runs in $O(n)$ **expected** (i.e., average) time.

2. Given two arrays of n integers, design an algorithm that finds out in $O(n \log n)$ steps if the two arrays have an element in common. (Microsoft interview question)

3. Given an array $A[1..100]$ which contains all natural numbers between 1 and 99, design an algorithm that runs in $O(n)$ and returns the duplicate value. (Microsoft interview question)

4. Assume you are given two arrays A and B, each containing n distinct positive numbers and the equation $x^8 - x^4y^4 = y^6 + x^2y^2 + 10$. Design an algorithm which runs in time $O(n \log n)$ which finds if A contains a value for x and B contains a value for y that satisfy the equation.

5. You're given an array of n integers, and must answer a series of n queries, each of the form: "how many elements of the array have value between L and R?", where L and R are integers. Design an $O(n \log n)$ algorithm that answers all of these queries.

6. Assume you have an array of $2n$ distinct integers. Find the largest and the smallest number using $3n - 2$ comparisons only.

7. Assume that you have an array of 2^n distinct integers. Find the largest and the second largest number using only $2^n + n - 2$ comparisons.

8. 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 your task can always be accomplished by asking no more than $3n - 3$ such questions, even in the worst case.

   (b) Show that your task can always be accomplished by asking no more than $3n - \lfloor \log_2 n \rfloor - 3$ such questions, even in the worst case.

9. Given n numbers $x_1, \ldots, x_n$ where each x_i is a real number in the interval $[0, 1]$, devise an algorithm that runs in linear time that outputs a permutation of the n numbers, say $y_1, \ldots, y_n$, such that $\sum_{i=2}^{n} |y_i - y_{i-1}| < 2$.

   Hint: this is easy to do in $O(n \log n)$ time: just sort the sequence in ascending order. In this case, $\sum_{i=2}^{n} |y_i - y_{i-1}| = \sum_{i=2}^{n} (y_i - y_{i-1}) = y_n - y_1 \leq 1 - 0 = 1$. Here $|y_i - y_{i-1}| = y_i - y_{i-1}$ because all the differences are non-negative, and all the terms in the sum except the first and the last one cancel out. To solve this problem, one might think about tweaking the BucketSort *algorithm, by carefully avoiding sorting numbers in the same bucket.*

10. Given n real numbers $x_1, ..., x_n$ where each x_i is a real number in the interval $[0, 1]$, devise an algorithm that runs in linear time and that will output a permutation of the n numbers, say $y_1, ..., y_n$, such that $\sum_{i=2}^{n} |y_i - y_{i-1}| < 1.01$.

11. Let M be an $n \times n$ matrix of distinct integers $M(i, j)$, $1 \leq i \leq n$, $0 \leq j \leq n$. Each row and each column of the matrix is sorted in the increasing order, so that for each row i, $1 \leq i \leq n$,

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

   and for each column j, $1 \leq j \leq n$,

$$M(1, j) < M(2, j) < \ldots < M(n, j)$$

   You need to determine whether M contains an integer x in $O(n)$ time.

12. 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.

    Hint: try "double" QUICKSORT*; one which uses a foot as a pivot for shoes and another one which uses a shoe as a pivot for feet.*

13. 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. We assume that students are not allowed to vote for themselves. However, the students have forgotten to specify the order of names on each piece of paper – for instance, "Alice Bob" could mean that Alice voted for Bob, or that 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 determine who these students voted for?

    (c) Suppose every student received at least one vote. What is the maximum possible number of votes received by any student? Justify your answer.

    (d) Using parts (b) and (c), or otherwise, design an 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. An $O(n^2)$ algorithm earns partial credit, but you should aim for an $O(n)$ algorithm.

    Hint: first, use part (c) to consider how you would solve it in the case where every student received at least one vote. Then, apply part (b).

14. 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) of the N teams. Not being the sporty type – but wanting to fit in nonetheless – you must decide for yourself some subset of teams (possibly all, or none) to support. You don't want to be branded a copycat, so your subset

must not be identical to anyone else's. The trouble is, you don't know which friends support which teams, so you can ask your friends some questions of the form "Does friend A support team B?" (you choose A and B before asking each question). Design an algorithm that determines a suitable subset of teams for you to support and asks as few questions as possible in doing so.

15. You are given an array A consisting of $2n-1$ integers. Design an algorithm which finds all of the n possible sums of n consecutive elements of A and **which runs in time $\mathbf{O(n)}$**. Thus, you have to find the values of all of the sums

$$\begin{aligned} S[1] &= A[1] + A[2] + \ldots + A[n-1] + A[n]; \\ S[2] &= A[2] + A[3] + \ldots + A[n] + A[n+1]; \\ &\vdots \qquad\quad \vdots \qquad\quad \vdots \\ S[n] &= A[n] + A[n+1] + \ldots + A[2n-2] + A[2n-1], \end{aligned}$$

and your algorithm **should run in time $\mathbf{O(n)}$**.

16. You are a fisherman, trying to catch fish with a net that is W meters wide. Using your advanced technology, you know that the positions of all N fish in the sea can be represented as integers on a number line. There may be more than one fish at the same location.
To catch the fish, you will cast your net at position x, and will catch all fish with positions between x and $x+W$, **inclusive**. Given N, W and an array $X[1..N]$ denoting the positions of fish in the sea, give an $\mathbf{O(N \log N)}$ algorithm to find the maximum number of fish you can catch by casting your net once.
For example, if $N = 7, W = 3$ and $X = [1, 11, 4, 10, 6, 7, 7]$, then the most fish you can catch is 4: by placing your net at $x = 4$, you will catch one fish at position 4, one fish at position 6 and two fish at position 7.

17. Your army consists of a line of N giants, each with a certain height. You must designate precisely $L \leq N$ 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 $H[1..N]$ 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 instance, suppose $N = 10, L = 3, K = 2$ 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.

18. You are given an array A of n distinct integers.

    (a) You have to determine if there exists a number (not necessarily in A) which can be written as a sum of squares of two distinct numbers from A in two different ways (note: $m^2 + n^2$ and $n^2 + m^2$ counts as a single way) and which runs in time $n^2 \log n$ in the **worst case** performance. Note that the brute force algorithm would examine all quadruples of elements in A and there are $\binom{n}{4} = O(n^4)$ such quadruples.

    (b) Solve the same problem but with an algorithm which runs in the **expected time** of $O(n^2)$.

19. Suppose that you bought a bag of n bolts with nuts screwed on them. Your 5 year old nephew unscrewed all the nuts from the bolts and put both the nuts and the bolts back into the bag. The bolts are all of similar quite large size but are actually of many different diameters, differing only by at most a few millimetres, so the only way to see if a nut fits a bolt is to try to screw it on and determine if the nut is too small, if it fits or if it is too large. Design an algorithm for matching each bolt with a nut of a fitting size which runs in the **expected** time $O(n \log n)$.

20. You are given 1024 apples, all of similar but different sizes and a small pan balance which can accommodate only one apple on each side. Your task is to find the heaviest and the second heaviest apple while making at most 1032 weighings.

21. You are in an orchard which has a quadratic shape of size $4n$ by $4n$ with equally spaced trees. You purchased apples from n^2 trees which also form a square, but the owner is allowing to choose such a square anywhere in the orchard. You

have a map with the number of apples on each tree. Your task is to chose such a square which contains the largest total number of apples and which runs in time $O(n^2)$. Note that the brute force algorithm would run in time $\Theta(n^4)$.

22. Read the review material from the class website on asymptotic notation and basic properties of logarithms, pages 38-44 and then determine Determine if $f(n) = O(g(n))$ or $g(n) = O(f(n)$ or both (i.e., $f(n) = \Theta(g(n))$) or neither of the two, for the following pairs of functions

    (a) $f(n) = (\log_2(n))^2$; $g(n) = \log_2(n^{\log_2 n})^2$;

    (b) $f(n) = n^{10}$; $g(n) = 2^{\sqrt[10]{n}}$;

    (c) $f(n) = n^{1+(-1)^n}$; $g(n) = n$.

23. Read the review material from the class website on asymptotic notation and basic properties of logarithms, pages 38-44 and then determine if $f(n) = \Omega(g(n))$, $f(n) = O(g(n))$ or $f(n) = \Theta(g(n))$ for the following pairs. Justify your answers.

$f(n)$	$g(n)$
$(\log_2 n)^2$	$\log_2(n^{\log_2 n}) + 2\log_2 n$
n^{100}	$2^{n/100}$
$\sqrt{n}$	$2^{\sqrt{\log_2 n}}$
$n^{1.001}$	$n \log_2 n$
$n^{(1+\sin(\pi n/2))/2}$	$\sqrt{n}$

You might find the following inequality useful: if $f(n), g(n), c > 0$ then $f(n) < c\, g(n)$ if and only if $\log f(n) < \log c + \log g(n)$. Also remember that $O(f(n))$ does not define a linear ordering; for some f, g neither $f = O(g)$ nor $g = O(f)$.

24. Determine the asymptotic growth rate of the solutions to the following recurrences. If possible, you can use the Master Theorem, if not, find another way of solving it.

    (a) $T(n) = 2T(n/2) + n(2 + \sin n)$

    (b) $T(n) = 2T(n/2) + \sqrt{n} + \log n$

    (c) $T(n) = 8T(n/2) + n^{\log n}$

    (d) $T(n) = T(n-1) + n$

25. Assume that you are given an array A containing $2n$ numbers. The only operation that you can perform is make a query if element $A[i]$ is equal to element $A[j]$, $1 \leq i, j \leq 2n$. Your task is to determine if there is a number which appears in A at least n times using an algorithm which runs in linear time.

    Warning and a Hint: a really tricky one. The reasoning resembles a little bit the reasoning used in the celebrity problem: try comparing them in pairs and first find one or at most two possible candidates and then count how many times they appear.