

1. You are given an array of integers with total number of digits (over all integers) n . Show how to sort the array in time $O(n)$ in the worst case.
2. Consider the following pseudocode for a variant of Kruskal's algorithm for finding a minimum spanning tree of a weighted graph $G = (V, E)$ with weight function w .

MST-Kruskal(G, w)

```
1   $A \leftarrow \emptyset$ 
2  for each  $v \in V$  Make-Set( $v$ )
3  sort edges of  $E$  into increasing order by weight using merge-sort
4  for each edge  $(u, v) \in E$  taken in increasing order by weight
5      if Find-Set( $u$ )  $\neq$  Find-Set( $v$ )
6          then  $A \leftarrow A \cup \{(u, v)\}$ 
7              Union( $u, v$ )
8  return  $A$ 
```

Recall from our discussion of disjoint sets that m **Make-Set**, **Find-Set**, and **Union** operations, n of which are **Make-Set** operations, can be performed in time $O(m \lg^*(n))$.

- a) Analyze the running time of this variant of Kruskal's algorithm.
 - b) Suppose that the edge weights are independent and uniformly distributed between 0 and 1. That is, the probability that an edge has weight less than or equal to x is x , for $0 \leq x \leq 1$, independently of the other edges. Describe how to alter the previous algorithm so that it runs in time $O(|V| \lg^*(|V|))$ on average.
3. Design an efficient algorithm for finding a *longest* path from a vertex s to a vertex t in an acyclic weighted directed graph $G = (E, V)$. Specify the graph representation and any auxiliary data structures used. Also, analyze the time complexity of your algorithm.
 4. In the version of insertion sort that we discussed in class, at the $(j + 1)$ st step we search the sorted subarray $A[1 \dots j]$ to find where to insert $A[j + 1]$. In the worst case this takes j comparisons. A natural alternative is to use binary search to determine where to insert $A[j + 1]$.
 - a) Using the binary search alternative, analyze the number of comparisons required to sort an array of length n in the worst case.

b) Using the binary search alternative, analyze the number of data moves required to sort an array of length n in the worst case.

5. Design an algorithm that takes as input n distinct integers and returns the length of the longest increasing subsequence. For example, for 32, 17, 45, 22, 78, 19, 4, 88, the longest increasing subsequence is 4 (for example, 17, 45, 78, 88). Analyze the running time of your algorithm.

6. Suppose that a committee of n people are trying to decide on a chairman. Each person has a subset of other members (which includes themselves) who they support for chairman. We can represent the preferences with a digraph which has an edge from each vertex to itself, and an edge from vertex u to v if u supports v for chairman.

Before holding an election, the committee decides to compare their preferences and see if there is an individual who supports only themselves for chairman, and who everyone else also supports. Assume that the preference digraph is given as an adjacency matrix. Design a linear time algorithm for determining if there is such an individual.

7. Suppose that an array A contains n distinct integers A_1, A_2, \dots, A_n . Denote the order statistics by $A_{(1)} < A_{(2)} < \dots < A_{(n)}$. Suppose that we want to “approximately” sort A . Specifically, we want to rearrange A so that if $A_i = A_{(j)}$, then in the approximately sorted array $|i - j| \leq k$, for some fixed positive $k < n$. In other words, we want to rearrange A so that each element is no more than k locations away from where it should be in the sorted order.

Describe an efficient algorithm for approximately sorting A , and analyze its average running time if all orderings of A are equally likely.

Suppose that after approximately sorting A , you decide that you actually need it sorted. What would be an appropriate algorithm to use? Explain.

8. Suppose you were in charge of arranging hotel accommodations for a tour group of n tourists. The hotel has rooms that accommodate c_k people, costing r_k , for $k = 1, 2, \dots, m$ and $1 = c_1 < c_2 < \dots < c_m$. Assume that the cost per person decreases with room size. These are friendly tourists and don’t mind sharing a room with any other people in their group. Assume that there is no limit to the number of rooms of each type available. Give an efficient algorithm for determining the number of rooms of each type that results in the smallest total cost.

For example, if the rooms are singles for \$100, triples for \$180, and quadruples for \$200, then the best choice for $n = 5$ is one quadruple and one single, for a total cost of \$300.

A “greedy” approach to solving the problem is to keep choosing the biggest room you can still fill up with the remaining group of people.

(In the above example, if $n = 9$, then choose 2 quadruples and 1 single.) Does this approach always give an optimal solution?

Describe an efficient algorithm to solve this problem.

9. (CLRS) Arbitrage is the use of discrepancies in currency exchange rates to transform one unit of a currency into more than one unit of the same currency. For example, suppose that 1 U.S. dollar buys 46.4 Indian rupees, 1 Indian rupee buys 2.5 Japanese yen, and 1 Japanese yen buys 0.0091 U.S. dollars. Then, by converting currencies, a trader can start with 1 U.S. dollar and buy $46.4 \times 2.5 \times 0.0091 = 1.0556$ U.S. dollars, thus turning a profit of 5.56 percent.

Suppose that we are given n currencies c_1, c_2, \dots, c_n and an $n \times n$ table R of exchange rates, such that one unit of currency c_i buys $R[i, j]$ units of currency c_j .

Give an efficient algorithm to determine whether or not there exists an integer k and a sequence of currencies $(c_{i_1}, c_{i_2}, \dots, c_{i_k})$ such that

$$R[i_1, i_2] \cdot R[i_2, i_3] \cdots R[i_k, i_1] > 1.$$

Use algorithms from class or the book as subroutines. Analyze the running time of your algorithm.

10. You have three containers with capacities of 4, 7, and 10 liters, respectively. Initially the 4 and 7 liter containers are full of water and the 10 liter container is empty. You are only allowed the following operation: pour the contents of one container into another, stopping only when the source container is empty or the destination container is full. You want to know if there is a sequence of operations that leaves exactly 2 liters in each of the 4- and 7-liter containers.

a) Model this as a graph problem. What algorithm should be used to solve this problem?

b) Suppose you want to know the *fewest* operations that achieve the desired allocation ($= \infty$ if it is not possible). How would you solve this problem?

11. You are given a sorted array A of length m , and a sorted array B of length n . Give an efficient algorithm to find the k th smallest value of the union of the two arrays. Your running time should be $\mathcal{O}(\lg(m) + \lg(n))$.