CSE421: Design and Analysis of Algorithms March 12, 2020

Homework 9: Practice Problems

Anup Rao Due: Never

Asymptotics

In each of the following situations, indicate whether $f = O(g)$, $f = \Omega(g)$, or both:

1. $f(n) = n - 100, g(n) = n - 200$.
2. $f(n) = \sqrt{n}, g(n) = \sqrt[3]{n}$.
3. $f(n) = 100n + \log n, g(n) = n + \log^2 n$.
4. $f(n) = n \log n, g(n) = 10n \log(10n)$.
5. $f(n) = \log 2n, g(n) = \log 3n$.
6. $f(n) = 10 \log n, g(n) = \log n^2$.
7. $f(n) = n^{1.01}, g(n) = n \log^2 n$.
8. $f(n) = n^2/\log n, g(n) = n \log^2 n$.
9. $f(n) = (\log n)^{\log n}, g(n) = n/\log n$.
10. $f(n) = \sqrt{n}, g(n) = \log^3 n$.
11. $f(n) = \sqrt{n}, g(n) = 5^{\log n}$.
12. $f(n) = n2^n, g(n) = 3^n$.
13. $f(n) = 2^n, g(n) = 2^{n+1}$.
14. $f(n) = n!, g(n) = 2^n$.
15. $f(n) = (\log n)^{\log n}, g(n) = 2^{\log^2 n}$.

Graphs

1. You are given a weighted directed graph, where every edge has a non-negative weight that represents the distance between the nodes. Each node in the graph represents either a residence or a hospital in the city, with the edges representing cities. There are n nodes and m edges. Give an $O((m + n) \log n)$ time algorithm to compute the distance of every residence to the closest hospital.

2. Let $deg(u)$ denote the degree of a vertex u.

   (a) Show that the number of vertices of odd degree is even.
   (b) Does a similar statement hold for the vertices of odd outdegree in a directed graph?

3. We saw that a graph is bipartite exactly when it can be colored with 2 colors. How many colors are needed to color a graph that has exactly one odd cycle?

4. Design a linear time algorithm which, given an undirected graph G and a particular edge e in it, determines whether G has a cycle containing e.

5. Give a linear time algorithm that takes a directed acyclic graph as input and outputs whether or not there is a directed path that touches every vertex exactly once.

6. Show that in any undirected graph, it is possible to pair up the vertices of odd degree and find paths between each such pair so that all these paths are edge-disjoint.

Greedy

1. Give a linear time algorithm that takes two strings x, y over some constant size alphabet and decides if x is a subsequence of y. For example "cat" is a subsequence of "scratch".

2. Design a linear time algorithm that takes an undirected graph as input, and finds an edge that can be removed while leaving the graph connected, if such an edge exists.

3. Let T be an MST of G, which has unique edge weights. Given a connected subgraph H of G, show that $T \cap H$ is contained in some MST of H.

4. Give an $O(m \log n)$ time algorithm that takes a weighted undirected graph G, and a subset U of the vertices as input, and finds the minimum spanning tree that has the property that the vertices of U are leaves in the tree. HINT: Consider what happens when the vertices of U are removed from the optimal solution.

5. Give an algorithm that takes a list of numbers $d_1, \ldots, d_n$ and outputs whether or not there is an undirected graph where the vertices have degrees $d_1, \ldots, d_n$.

6. Alice wants to throw a party where every person knows at least 5 people, and doesn't know at least 5 people. Give an efficient algorithm that given the undirected graph representing the friendship status of her n friends, finds the largest number of people she can invite subject to those constraints.

7. You are given an undirected graph where every edge has weight either $1, 2, 3$. Give a linear time algorithm to find a spanning tree in the graph.

8. A server has n customers waiting to be served. Each customer i will take t_i time to be served. The goal is to find the optimal order to serve the customers so that the total waiting time is minimized for all customers. For example, if the customers are served in order $1, 2, \ldots, n$ then the waiting time for i is $\sum_{j=1}^{i-1} t_j$. Give an algorithm that finds the optimal order for serving the customers.

Divide and Conquer

1. Solve the following recurrences:
   (a) $T(n) = 2T(n/3) + 1$.
   (b) $T(n) = 5T(n/4) + n$.
   (c) $T(n) = 7T(n/7) + n$.
   (d) $T(n) = 9T(n/3) + n^2$.
   (e) $T(n) = 13T(n/3) + n^3$.

2. You are given an array of numbers and some of them are duplicates. Give an $O(n \log n)$ time algorithm to eliminate all duplicates.

3. You are given an array of n distinct numbers $A[1...n]$, such that $A[2] < A[1] < A[n]$. $1 < i < n$ is a local minimum if $A[i] < A[i+1]$ and $A[i] < A[i-1]$. Give an $O(\log n)$ time algorithm to find a local minimum in the numbers.

4. In homework you designed an algorithm to find a majority element among n elements, should one exist, in $O(n \log n)$ time. Can you improve the algorithm to run in $O(n)$ time? HINT: Pair up the elements, and discard all pairs that have two distinct values. Repeat.

5. You are given a sorted array $A[]$ (increasing order), but you don't know how many elements are in the array. If you try to access location $A[i]$, but i is bigger than the length of the array, you get the value ∞. Give an algorithm to find the location of element x, in time $O(\log n)$, where n is the length of the array.

6. Here we will develop a recursive algorithm for computing the greatest common divisor (gcd) of two numbers.
   (a) Show that the following holds:
   $$\gcd(x, y) = \begin{cases} 2\gcd(a/2, b/2) & \text{if both } a, b \text{ are even,} \\ \gcd(a/2, b) & \text{if } a \text{ is even and } b \text{ is odd,} \\ \gcd(|a-b|/2, b) & \text{if } a, b \text{ are both odd.} \end{cases}$$
   (b) Use the above equation to give an algorithm for computing $\gcd(a, b)$ and analyze its running time.

Dynamic Programming

1. You are given a list of n positive integers $a_1, \ldots, a_n$, and another positive integer t. Your goal is figure out if there is some subset of the a_i's that add up to t. Give an $O(nt)$ time algorithm for this task.

2. At a lumber mill, the time it takes to saw a log into two pieces is proportional to the length of the log. You are given a log of length n and the (integer) location of m cuts, finds the best sequence of cuts to ensure that the total time it takes to implement the cuts is minimized. For example, if you want to cut a log of length 10 at positions 2 and 4, if the first cut is made at 2 the cost will be 10+8=18, but if the first cut is made at 4, the cost will be 10+4 = 14.

3. You are given a convex polygon on n vertices in the plane, specified by their vertices. A triangulation of the polygon is specified by $n-3$ diagonals of the polygon such that no two diagonals intersect. This splits the polygon into $n-2$ disjoint triangles. We want to find the triangulation where the total length of all diagonals is minimized. Give an efficient algorithm to find this. HINT: Label the vertices in clockwise order. Let $A(i,j)$ denote the minimum cost triangulation of the polygon spanned by the vertices $i, i+1, ..., j$.

4. Given n positive integers $a_1, \ldots, a_n$, we want to know whether the integers can be partitioned into 3 sets, such that each set has the same sum. Give algorithm that runs in time polynomial in n and $\sum_i a_i$ for doing this.

5. You are given the price of a particular stock over a period of n days. You want to figure out when the best time to buy the stock, and sell the stock would have been, in retrospect. So for example if the sequence of prices is $3, 4, 2, 1, 6, 3, 4, 9$, then the best thing would be to buy when the price was 1 and sell when it was 9. Since there are $\binom{n}{2}$ choices for when to buy and sell, there is an obvious $O(n^2)$ time algorithm. Give an algorithm that finds the best time to buy and the best time to sell in $O(n)$ time.

Flows and Cuts

1. You are given a flow network where every edge has capacity 1. Your goal is to find a set of k edges such that deleting those edges reduces the value of the maximum flow by the most. Give a polynomial time algorithm for this.

2. You are given an $n \times n$ matrix that has 0/1 entries. You are allowed to modify the matrix by swapping two rows, or swapping two columns. You want to know whether it is possible to modify the matrix in this way so that you eventually end up with a matrix that has only 1's on the diagonal. Give a polynomial time algorithm for this.

3. Suppose you have two jigsaw puzzles that each make a pictures of size n by n, and further, all of the jigsaw pieces have area 1. Show that if both puzzles are assembled, and laid on top of each other, then there is a way to insert n^2 pins through both puzzles simultaneously, so that every jigsaw piece gets exactly one pin going through it. Give an efficient algorithm that can find the locations of the pins, given a description of the jigsaws.

4. You have n projects that you can do, with the i'th project costing c_i dollars. You find m investors, and the i'th investor is willing to give you a_i dollars, as long as the projects from the set $S_i \subseteq \{1, 2, \ldots, n\}$ are carried out. Give an efficient algorithm that computes the optimal set of projects and investors to use, to maximize your profit.

5. You are given a directed graph with a source node s and a disjoint collection of terminal nodes T. Your goal is to separate as many of terminal nodes from the source, by deleting as few edges as possible. For each set of edges F, let $q(F)$ denote the number of terminal nodes that are unreachable from s once F is deleted. You want to find the F that maximizes $q(F) - |F|$. Give a polynomial time algorithm for this.

P vs NP

1. You are given a collection of sets $S_1, S_2, \ldots, S_m \subseteq \{1, 2, \ldots, n\}$. For a parameter k, your goal is find k elements in the universe such that every set contains at least one of the elements. Show that deciding whether such a set of elements exists is an NP-complete problem.

2. Given an undirected graph, a subset of the vertices is called an *independent set* if no edges are contained in the set. Show that the decision version of finding the largest independent set is NP-complete.

3. (hard) In the Steiner tree problem, you are given an undirected graph and a subset S of the vertices, and want to find smallest subset of edges that connects all the vertices of S. Show that the decision version of this problem is NP-complete. HINT: Reduce from the vertex cover problem. Given an input graph G to vertex cover, construct a new graph G' as follows. Every vertex a of G corresponds to a vertex a of G'. Every edge (a, b) of G corresponds to a vertex $[a, b]$ of G'. In G', connect every vertex a to every vertex b, and connect every vertex $[a, b]$ to the vertices a and b. The set S corresponds to all vertices of G' of the form $[a, b]$. Argue that the smallest steiner tree in G' can be used to recover the smallest vertex cover in G.