Homework 1 for Design and Analysis of Algorithms (Spring 2021)

Due: March 12th, 2021

Problem 1. (Exercise 0.1 in [DPV08]) In each of the following situations, indicate whether $f = O(g)$, or $f = \Omega(g)$, or both (in which case $f = \Theta(g)$).

	$f(n)$	$g(n)$
(a)	$n - 100$	$n - 200$
(b)	$n^{1/2}$	$n^{2/3}$
(c)	$100n + \log n$	$n + (\log n)^2$
(d)	$n \log n$	$10n \log 10n$
(e)	$\log 2n$	$\log 3n$
(f)	$10 \log n$	$\log(n^2)$
(g)	$n^{1.01}$	$n \log^2 n$
(h)	$n^2/\log n$	$n(\log n)^2$
(i)	$n^{0.1}$	$(\log n)^{10}$
(j)	$(\log n)^{\log n}$	$n/\log n$
(k)	$\sqrt{n}$	$(\log n)^3$
(l)	$n^{1/2}$	$5^{\log_2 n}$
(m)	$n2^n$	3^n
(n)	2^n	2^{n+1}
(o)	$n!$	2^n
(p)	$(\log n)^{\log n}$	$2^{(\log_2 n)^2}$
(q)	$\sum_{i=1}^n i^k$	n^{k+1}.

Problem 2. (Exercise 0.2 in [DPV08]) Show that, if c is a positive real number, then $g(n) = 1 + c + c^2 + \cdots + c^n$ is:

(a) $\Theta(1)$ if $c < 1$.

(b) $\Theta(n)$ if $c = 1$.

(c) $\Theta(c^n)$ if $c > 1$.

Problem 3. Show that there exists a C++ program P who can generate all pairs of natural numbers (x, y) such that $P_x(y)$ terminates [1]. That is, the program P can keep printing pairs of numbers (x, y) such that $P_x(y)$ terminates and for every (x', y') such that $P_{x'}(y')$ terminates, the program P can print it at some time.

[1]Since the number of such pairs are infinite, the program P must run forever.

References

[DPV08] Sanjoy Dasgupta, Christos H Papadimitriou, and Umesh Virkumar Vazirani. *Algorithms.* McGraw-Hill Higher Education New York, 2008. 1

Algorithm Design and Analysis
Assignment 2

1. You are given two sorted lists of size m and n. Give an $O(\log m + \log n)$ time algorithm for computing the k-th smallest element in the union of the two lists.

2. A k-way merge operation. Suppose you have k sorted arrays, each with n elements, and you want to combine them into a single sorted array of kn elements. Design a efficient algorithm using divide-and-conquer.

3. Recall that we have learned how to find the k-th element in a list with a randomized algorithm (randomly choose a pivot), can we do it deterministically? In this exercise, we will develop one called the *Median of Medians* algorithm, invented by Blum, Floyd, Pratt, Rivest, and Tarjan.

   Why we need to pick the *pivot* x randomly in our randomized algorithm? This is to guarantee, at least in expectation, that the numbers less than x and the numbers greater than x are in close proportion. In fact, this task is quite similar to the task of "finding the k-th largest number" itself, and therefore we can bootstrap and solve it recursively!

   Assume we have an array A of n distinct numbers and would like to find its k-th largest number.

   (a) Consider that we line up elements in groups of three and find the median of each group. Let x be the median of these $n/3$ medians. Show that x is close to the median of A, in the sense that a constant fraction of numbers in a is less than x and a constant fraction of numbers is greater than x as well.

   (b) Design a recursive algorithm by the above idea and analyze the running time.

   (c) Can we improve the running time by increasing the number of elements (e.g. 4,5,6?) in each group? What is the best choice? Give the answer and the proof. Note that in this problem, we drop the big-O notation and discuss about the constants.

4. The Hadamard matrices $H_0,\ H_1,\ H_2, ...$ are defined as follows:

   - H_0 is the 1×1 matrix [1].
   - For $k > 0$, H_k is the $2^k \times 2^k$ matrix

   $$H_k = \begin{bmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{bmatrix}$$

   Show that if $\vec{v}$ is a column vector of length $n = 2^k$, then the matrix-vector product $H_k\vec{v}$ can be calculated using $O(n \log n)$ operations in word RAM.

5. How long does it take you to finish the assignment (include thinking and discussing)? Give a score (1,2,3,4,5) to the difficulty.

Algorithm Design and Analysis
Assignment 3

1. The following algorithm attempts to find the shortest path from node s to node t in a directed graph with some negative edges:

   - Add a large enough number to each edge weight so that all the weights become positive, then run Dijkstra's algorithm.

   Either prove this algorithm correct, or give a counter-example.

2. A bipartite graph is a graph $G = (V, E)$ whose vertices can be partitioned into two sets V_1 and V_2 (i.e., $V = V_1 \cup V_2$ and $V_1 \cap V_2 = \emptyset$) such that there are no edges between vertices in the same set. (For instance, if $u, v \in V_2$, then there is no edge between u and v.)

   (a) Show that an undirected graph is bipartite if and only if it contains no cycles of odd length.

   (b) Give a linear-time algorithm to determine whether an undirected graph is bipartite.

3. Let $G = (V, E)$ be an undirected connected graph. Let T be a depth-first search tree of G. Suppose that we orient the edges of G as follows: For each tree edge, the direction is from the parent to the child; for every non-tree (back) edge, the direction is from the descendent to the ancestor. Let G' denote the resulting directed graph.

   (a) Give an example to show that G' is not strongly connected.

   (b) Prove that if G' is strongly connected, then G satisfies the property that removing any single edge from G will still give a connected graph.

   (c) Prove that if G satisfies the property that removing any single edge from G will still give a connected graph, then G' must be strongly connected

   (d) Give an efficient algorithm to find all edges in a given undirected graph such that removing any one of them will make the graph no longer connected.

4. How long does it take you to finish the assignment (include thinking and discussing)? Give a score (1,2,3,4,5) to the difficulty.

Algorithm Design and Analysis
Assignment 4

1. You are given a set of n jobs, where each job j is associated with a size s_j (how much time it takes to process the job) and a weight w_j (how important the job is). Suppose you have only one machine that can process one unit of jobs per time slot. Assume all jobs are given at time $t_0 = 0$ and are to be processed one by one using this machine. Let $C_j > t_0$ be the time that job j is completed. The goal is to find a schedule (of all the jobs) that minimizes the weighted completion time, i.e.,$\sum_{j=1}^{n} w_j C_j$.

   Greedy algorithms first order the jobs according to some criteria, and then process them one by one. Only one of the following criteria gives an algorithm that minimizes the weighted completion time.

   - Highest Weight First: process jobs in in descending order of their weights.
   - Smallest Size First: process jobs in ascending order of their sizes.
   - Highest Density First: process jobs in descending order of their weight to size ratio, namely in descending order of w_j/s_j.

   (a) Find out which one is correct.

   (b) Reason about its correctness.

   (c) Give counter examples for the other criteria.

2. Let $G = (V, E)$ be a connected undirected graph with positive edge weights. Give an algorithm to find a subset of edges E' with the smallest total weight such that removing E' from G will leave a graph with no cycle. Note that E' must contain at least one edge on every cycle of G. You need to prove the correctness of your algorithm.

3. Alice wants to throw a party and is deciding whom to call. She has n people to choose from, and she has made up a list of which pairs of these people know each other. She wants to pick as many people as possible, subject to two constraints: at the party, each person should have at least five other people whom they know and five other people whom they don't know. Give an efficient algorithm that takes as input the list of n people and the list of pairs who know each other and outputs the best choice of party invitees. Give the running time in terms of n.

Algorithm Design and Analysis
Assignment 5

1. Consider the following 3-PARTITION problem. Given integers $a_1, ..., a_n$, we want to determine whether it is possible to partition of $\{1, ..., n\}$ into three disjoint subsets I, J, K such that
$$\sum_{i \in I} a_i = \sum_{j \in J} a_j = \sum_{k \in K} a_k = \frac{1}{3} \sum_{i=1}^{n} a_i.$$
For example, for input $(1, 2, 3, 4, 4, 5, 8)$ the answer is yes, because there is the partition $(1, 8)$, $(4, 5)$, $(2, 3, 4)$. On the other hand, for input $(2, 2, 3, 5)$ the answer is no. Devise and analyze a dynamic programming algorithm for 3-PARTITION that runs in time polynomial in n and in $\sum_i a_i$.

2. Let $X[1..n]$ be a reference DNA sequence. Let S be a set of m exon candidates of X, where each exon candidate is represented by a triple (i, j, w), which means that the strength or probability for fragment $X[i..j]$ being an exon is w. Notice that many triples in S are false exons, and true exons do not overlap. Show how to use dynamic programming to find a maximum-weight subset of S in which all exon candidates are non-overlapping. The time complexity should be linear in terms of n and m.

3. Consider the following game. A "dealer" produces a sequence $s_1, \ldots, s_n$ of "cards" facing up, where each card s_i has a value v_i. Then two players take turns picking a card from the sequence, but can only pick the first or the last card of the (remaining) sequence. The goal is to collect cards of largest total value. Assume n is even.

   (a) Show a sequence of cards such that it is not optimal for the first player to start by picking up the available card of larger value. That is, the natural *greedy* stratgy is suboptimal.

   (b) Give an $O(n^2)$ algorithm to compute an optimal strategy for the first player. Given the initial sequence, your algorithm should precompute in $O(n^2)$ time some information, and then the first player should be able to make each move optimally in $O(1)$ time by looking up the precomputed information.

4. Assume points $v_1, v_2, \ldots, v_n$ form a convex polygon in $\mathbb{R}^2$. Let $d(i,j)$ be the Euclidean distance between v_i and v_j if $i \le j$ and $d(i,j) = -\infty$ if $i > j$. For every $r \ge 0$, we use $d^{(r)}(i,j)$ to denote the length of the the *longest paths* from v_i to v_j using *at most* r edges. Therefore, $d(i,j) = d^{(1)}(i,j)$.

   (a) Let $s, t \ge 0$ be any two any integers satisfying $r = s + t$. For every $i \le j$, prove that $d^{(r)}(i,j) = \max_{i \le k \le j} \left\{ d^{(s)}(i,k) + d^{(t)}(k,j) \right\}$.

   (b) Prove that the distance $d(\cdot,\cdot)$ satisfies the *inverse Quadrangle Inequality* (iQI):

$$\forall i \le i' \le j \le j' : d(i,j) + d(i',j') \ge d(i',j) + d(i,j').$$

   (c) Prove that for any integer $r \ge 0$, $d^{(r)}(\cdot,\cdot)$ satisfies iQI as well.

   (d) If we let $K^{(r)}(i,j)$ denote $\max \left\{ k \mid i \le k \le j \text{ and } d^{(r)}(i,j) = d^{(s)}(i,k) + d^{(t)}(k,j) \right\}$, prove that

$$K^{(r)}(i,j) \le K^{(r)}(i,j+1) \le K^{(r)}(i+1,j+1), \quad \text{for} \quad i \le j.$$

   (e) Give an algorithm to compute $d^{(r)}(i,j)$ for all $1 \le i < j \le n$ in $O(\log r \cdot n^2)$ time [1].

5. How long does it take you to finish the assignment (include thinking and discussing)? Give a score (1,2,3,4,5) to the difficulty.

[1] That is, your algorithm needs to compute all the $\binom{n}{2}$ values within $O(\log r \cdot n^2)$ time.

Algorithm Design and Analysis
Assignment 6

1. You are given a set S of football teams, where each team $x \in S$ has already accumulated w_x wins so far. For every pair of teams $x, y \in S$ we know that there are g_{xy} games remaining between x and y (that is, if x wins a remaining games against y, then y must win $g_{xy} - a$ remaining games against x). Given a specific team $z \in S$, we would like to decide if z still has a chance to have the maximum number of wins. Alternatively, suppose we have the power to determine who wins each of the remaining games, is there a way such that z would get as many wins as anybody else by the end of the tournament? Give a polynomial-time algorithm for this problem. Hint: We can assume without loss of generality that z wins all remaining games. Is there a way to split the wins of the remaining games among the rest of the teams so that none of them get more wins than z? Consider a reduction to the network flow problem.

2. Let G be a flow network in which each edge e has a capacity $c(e)$ as well as a lower bound $d(e)$ on the flow it must carry. Note that $d(e) \leq c(e)$. A feasible flow assigns a value within the range of $[d(e), c(e)]$ to each edge e. Note that assigning a zero flow to every edge may not give a feasible flow of G (due to the lower bound requirement of the edges). Show how to make use a maximum flow algorithm to find a feasible flow of G.

3. How long does it take you to finish the assignment (include thinking and discussing)? Give a score (1,2,3,4,5) to the difficulty.

Algorithm Design and Analysis
Assignment 7

1. Show that the following problem is NP-Complete.

   Maximum common subgraph Input: Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$; a budget b. Output: Decide whether there exist two set of nodes $V_1' \subseteq V_1$ and $V_2' \subseteq V_2$ whose deletion leaves at least b nodes in each graph, and makes the two graphs identical.

2. SINGLE EXECUTION TIME SCHEDULING (SS). Given a set S of n jobs, a relation $\prec$ on S, a number of processors k and a time limit t. Does there exist a function f from S to $\{0, 1, ..., t-1\}$ such that

   1. $f^{-1}(i)$ has at most k members, and
   2. if $J \prec J'$, then $f(J) < f(J')$?

   For a verbal description of the problem, we need to assign those n jobs to those k processors. Each processor takes one unit of time to complete each job. $f(J)$ is the starting time for job J, and job J ends at time $f(J)+1$. The first requirement above says that at most k jobs can be executed simultaneously. For two jobs J and J', $J \prec J'$ indicates J must be finished before executing J'. This is captured by the second requirement. We are deciding if we can schedule the n jobs on those k processors such that all of them are finished before time t.

   Now consider a more complex version: Single execution time scheduling with variable number of processors (SSV). Given a set S of n jobs, a relation $\prec$ on S, a time limit t, and a sequence of integers $c_0, c_1 ...., c_{t-1}$, where $\sum_{i=0}^{t-1} c_i = n$, does there exist a function f from S to $0, 1, ..., t-1$ such that

   1. $f^{-1}(i)$ has exactly c_i members, and
   2. if $J \prec J'$, then $f(J) < f(J')$?

   Show that (SS) is NP-Complete. You may follows the three steps below:

   (a) Show (SS) is in NP.

   (b) Show that there is a Karp reduction from SSV to SS: SSV $\leq_k$ SS.

   (c) Show that there is a Karp reduction from 3SAT to SSV: 3SAT $\leq_k$ SSV.

   Hint for part (c):

   - For each variable x_i $(i = 1, \dots, n)$ in the 3SAT instance, create two gadgets corresponding to the two Boolean assignments to the variable as follows:
     - Gadget for $x_i =$ **true**: create $n+2$ jobs $x_{i0}, x_{i1}, \dots, x_{in}, y_i$ with $x_{i0} \prec x_{i1} \prec \dots \prec x_{in}$ and $x_{i(i-1)} \prec y_i$;
     - Gadget for $x_i =$ **false**: create $n+2$ jobs $\bar{x}_{i0}, \bar{x}_{i1}, \dots, \bar{x}_{in}, \bar{y}_i$ with $\bar{x}_{i0} \prec \bar{x}_{i1} \prec \dots \prec \bar{x}_{in}$ and $\bar{x}_{i(i-1)} \prec \bar{y}_i$.
     - Intuitive idea: we may imagine x_i (or $\bar{x}_i$) to be true if and only if x_{i0} (or $\bar{x}_{i0}$, respectively) is executed at time 0, all the other jobs are used to control the requirement of the 3SAT problem.

- For each clause C_j ($j = 1, \ldots, m$), create 7 jobs $c^j_{ttt}, c^j_{ttf}, c^j_{tft}, c^j_{ftt}, c^j_{tff}, c^j_{ftf}, c^j_{fft}$ corresponding to the seven possibilities of the values for the three literals in the clause (that makes the clause evaluated to **true**). For example, c^j_{ttf} corresponds to that the first and the second literals are **true** and the third literal is **false**. Then, link each of the seven jobs to the last job in the variable gadget accordingly. For example, if we have a clause $C_j = x_3 \vee \bar{x}_4 \vee x_7$ and consider the job c^j_{ttf} (which corresponds to $x_3 =$ **true**, $\bar{x}_4 =$ **true** and $x_7 =$ **false**), we have $x_{3n} \prec c^j_{ttf}$, $\bar{x}_{4n} \prec c^j_{ttf}$ and $\bar{x}_{7n} \prec c^j_{ttf}$.
- Consider $t = n + 2$. Setup the values for $c_0, c_1, \ldots, c_{t-1}$ appropriately such that, in order to finish all the tasks, we must have the followings:
  - For each $i = 1, \ldots, n$, we must have either $f(x_{i0}) = 0, f(\bar{x}_{i0}) = 1$ or $f(x_{i0}) = 1, f(\bar{x}_{i0}) = 0$. The former case will represent $x_i =$ **true** and the latter case will represent $x_i =$ **false**;
  - For the 7 jobs corresponding to each clause, exactly one job must be executed at time n, and the remaining 6 jobs must be executed at time $n + 1$. The job executed at time n will represent the values for the three literals in the clause in a satisfying Boolean assignment.

3. How long does it take you to finish the assignment (include thinking and discussing)? Give a score (1,2,3,4,5) to the difficulty.