Algorithms: Assignment sheet 3

Due date: December 11, 2020

---

1. Let T be a minimum spanning tree of a graph G, and let L be the sorted list of the edge weights of T. Show that for any other minimum spanning tree T' of G, the list L is also the sorted list of edge weights of T'.

2. *Boruvka's algorithm.* Assume all edge weights are distinct in an undirected connected graph $G = (V, E)$ on m edges and n vertices. Consider the following step: we simultaneously contract the minimum weight edge incident on every vertex.

   (a) Show that this step can be performed in $O(m)$ time.

   (b) Show that the resulting graph has at most $n/2$ vertices.

   (c) Show that the MST of G is the union of the edges marked for contraction and the edges of the MST of the resulting graph.

3. Euclid's gcd algorithm takes two numbers a and b such that $a > b > 0$, and determines their gcd by computing the following sequence starting with $r_0 = a$ and $r_1 = b$.

$$\begin{array}{lll} r_2 = r_0 \bmod r_1 & q_2 = r_0 \text{ div } r_1 & (0 < r_2 < r_1) \\ r_3 = r_1 \bmod r_2 & q_3 = r_1 \text{ div } r_2 & (0 < r_3 < r_2) \\ & \cdots & \\ r_k = r_{k-2} \bmod r_{k-1} & q_k = r_{k-2} \text{ div } r_{k-1} & (0 < r_k < r_{k-1}) \end{array}$$

   The sequence r_i is strictly decreasing, implying that the algorithm terminates in a finite number of stages. Termination occurs when $r_{k-1} \bmod r_k = 0$ (that is, r_k divides r_{k-1}).

   (i) Show that $r_k = \gcd(a, b)$.

   (ii) Show that this is a polynomial time algorithm.

   (iii) Let F_n be the nth Fibonacci number. Show that the worst case for Euclid's algorithm is when a and b are consecutive Fibonacci numbers. If $a = F_{n+1}$ and $b = F_n$, then the number of stages k equals n. Noting that $F_n \approx \phi^n/\sqrt{5}$, where ϕ is the golden ratio 1.618..., prove that the running time of this algorithm is polynomial in the lengths of a and b.

   (iv) Show that for all $a > b > 0$, there exist integers x and y such that

$$\gcd(a, b) = ax + by.$$

   Moreover, x and y can be computed in polynomial time.

4. This problem deals with an efficient technique for *verifying* matrix multiplication. The fastest known algorithm for multiplying two $n \times n$ matrices runs in $O(n^\omega)$ time, where $\omega \approx 2.37$. This is significantly faster than the obvious $O(n^3)$ algorithm but this $O(n^\omega)$ algorithm has the disadvantage of being extremely complicated. Suppose we are given an implementation of this algorithm and would like to verify its correctness. Since program verification is a difficult task, a reasonable goal might be to verify the correctness of the output produced on specific executions of the algorithm. In other words, given $n \times n$ matrices A, B, and C with entries from rational numbers, we would like to verify that $AB = C$. Note that here we want to use the fact that we do not have to compute C; rather, our task is to verify that the product is indeed C. Give an $O(n^2)$ time randomized algorithm for this problem with error probability at most $1/2$.

   [Hint: Choose a random vector $r \in \{0,1\}^n$ (each entry in r is chosen independently and uniformly at random from $\{0,1\}$) and proceed.]

5. (i) We choose a number $a \in \{1, \ldots, n-1\}$ uniformly at random in Miller-Rabin algorithm. How do we perform this step in $O(\mathsf{poly}(\log n))$ time?

   (ii) For any integer n, show that the set $\mathbb{Z}_n^* = \{k : 1 \leq k \leq n \text{ and } \gcd(k,n) = 1\}$ forms a group under the operation $*_n$, i.e., multiplication modulo n.

6. Let n be a *non-Carmichael* odd composite number. Consider the Miller-Rabin algorithm on input n. Show that with probability $\geq 3/4$, (one invocation of) this algorithm returns the answer "composite".

7. We have seen an efficient Monte Carlo algorithm for testing if a given number is prime. In several applications (for example, the RSA crypto scheme), it is necessary to pick large prime numbers at random. Suggest an efficient Monte Carlo algorithm for generating a random $\Theta(\log n)$ bit length prime.

   [Hint: For any m, the probability that a random integer in $\{1, \ldots, m\}$ is prime $\approx 1/\ln m$.]

8. Design a randomized polynomial time algorithm for determining a non-trivial factor of n, given a composite number n and $\phi(n)$. That is, the running time of your algorithm should be $\mathsf{poly}(\log n)$ and with probability $\geq 3/4$, it should return a factor of n.

   Hint: Pick an $a \in \{1, \ldots, n-1\}$ uniformly at random. The non-trivial case is when $a \in \mathbb{Z}_n^*$. Use the idea for Carmichael numbers in Miller-Rabin algorithm.

9. Consider the *online edge colouring* problem where the vertex set V of the graph is fixed and the edges in the graph are presented to you in an online manner, one after another. As each edge e is specified, your algorithm must assign this edge e a colour and this colour of e cannot be changed henceforth. While the offline edge colouring of G knows all the edges of E while deciding on their colours, the online edge colouring algorithm has to decide on the colour of each edge e without any knowledge of the future edges. Design a 2-competitive algorithm for the online edge colouring problem.

10. Assume that renting skis costs Rs.100 per day and buying skis costs Rs.1000. Everyday you have to decide whether you will continue to rent skis for one more day, or buy a pair of skis. The online adversary, during the course of some unknown future day D, will change the weather and stop the skiing season. You would like to minimize the cost of skiing.

    If you knew the day D in advance, then you would buy skis at the start if $1000 \leq D \cdot 100$, else you would rent skis everyday for D days. This is what the optimal offline algorithm will do. However you do not know the day D in advance; design a 2-competitive online algorithm for this problem. Show that every deterministic online algorithm for the ski-rental problem is 2-competitive.