Algorithms and Data Structures: Assignment 4, due Nov. 15

Please read the assignment policies on the course homepage before starting the assignment.

---

1. Reading: (i) Introduction of Chapter 34, and Sections 34.1, 34.2. (ii) Sections C.1, C.2, C.3 in the Appendix.

2. (20 marks) Given a directed graph with positive edge weights, give an algorithm to find the *second* min-weight path between vertices s and t. Note that multiple s-t paths may have the same weight, so the path returned must have weight strictly greater than the minimum weight path.

3. (15 marks) Let G be a graph with $2n$ vertices. A *bisection* of G is a cut (S, T) with $|S| = |T|$. Since G has an even number of vertices, it clearly has a bisection. Find a probabilistic polynomial-time algorithm that produces a bisection with an expected cut (i.e., number of edges across cut) at least half that of the maximum. Then convert your algorithm to a deterministic polynomial-time algorithm.

4. (10 marks) Consider a random walk on a path with vertices numbered $1, 2, \ldots, n$ from left to right. At each step, we flip an unbiased coin to decide which direction to walk, moving one step left or one step right with equal probability. The random walk ends when we fall off one end of the path either by moving left from vertex 1 or right from vertex n. If we start from vertex 1, what is the probability that the walk ends by falling off the right end of the path?

5. (20 marks) 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 $1/2$.

6. (20 marks) Show how to implement the push-relabel algorithm using $O(n)$ time per relabel operation, $O(1)$ time per push, and $O(1)$ time to select an applicable operation, for a total running time of $O(mn^2)$.

   These questions are added on Nov 9th

7. (20 marks) Given an undirected graph G, an edge colouring is an assignment of colours to the edges of G such that no two edges incident to the same vertex get the same colour. The edge colouring problem asks for an edge colouring using the minimum number of colours. This is a hard problem to solve. Here we are interested in the *online edge colouring* problem where the vertex set V of the graph G is fixed and the edges in the graph are presented to us in an online manner, one after another.

   As each edge e is specified, our 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.

8. (20 marks) 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. Give an efficient Monte Carlo algorithm for generating a random $\Theta(\log n)$ bit length prime.

9. (20 marks) Show that the number of distinct minimum cuts in an undirected graph (assume all edge weights are one) is at most $\binom{n}{2}$. That is, show that the number of distinct cuts whose value is equal to the value of the min-cut in the graph is at most $\frac{n(n-1)}{2}$.

10. (20 marks) We are given an $n \times n$ 0-1 matrix M whose determinant value is known to be between -2^n and 2^n. There is a simple algorithm for computing the determinant of an $n \times n$ matrix using $O(n^3)$ arithmetic operations and it works over any field.

    Arithmetic on $O(\log n)$ bit numbers takes unit time – so when all arithmetic operations in the above algorithm are on $O(\log n)$ bit numbers, then the running time of this algorithm is $O(n^3)$. However when numbers are large, then arithmetic operations cannot be assumed to take unit time; for example, doing arithmetic on n-bit numbers takes $\Theta(n)$ time.

    Show a randomized algorithm with expected running time $O(n^3 \cdot \mathsf{poly}(\log n))$ and success probability $\geq 1 - 1/n$ for determining whether M is singular or not.

    [Hint: You can take the following fact for granted. For any m, the probability that an integer chosen uniformly at random in $\{1, \ldots, m\}$ is prime $\approx c/\ln m$ for some constant c.]