Randomised Algorithms

Winter term 2022/2023, Exercise Sheet No. 4

Hand-out: Mon, 7. Nov.
Hand-in: Sun, 13. Nov.

UNIVERSITY OF PASSAU
Faculty of Computer Science and Mathematics

Dr. Duc-Cuong Dang
Prof. Dr. Dirk Sudholt

Exercise 1 [8 points]

Assume you own a combination lock to which you have forgotten the combination to unlock it. The only way to open the lock is to try multiple combinations until you find the right one that unlocks the lock. Let us denote the set of all possible combinations by S and the unknown combination that opens the lock as $x^* \in S$.

(a) Show that, for every deterministic algorithm that works correctly on all inputs, there is an input x^* that makes the algorithm try all $|S|$ combinations.

(b) Consider a simple algorithm that tries all possible combinations in a uniform random order, and let T be the number of combination tried before x^* is found. Show by induction that $\text{Prob}(T = k) = \frac{1}{|S|}$ for all $1 \leq k \leq |S|$ and use this to compute $\text{E}(T)$.

(c) Apply Yao's minimax principle to find a lower bound on the expected number of combinations that *every* randomised algorithms must try to find any given input x^* and use this to deduce whether the simple algorithm in (b) is optimal or not.

Hint: We shall view x^* as an *input* to the problem (without knowing x^*). All algorithms can be identified by a sequence in which combinations are tested, as long as the optimum is not found. You may use without proof that in the application of Yao's minimax principle, it suffices to consider all correct algorithms *that make no redundant tests*. The set of such algorithms is finite as required for Yao's minimax principle.

Exercise 2 [4 points]

Recall that a cut of an undirected graph $G = (V, E)$ is a partition of its vertices V into two disjoint sets (S_1, S_2), and we say an edge $(u, v) \in E$ is a cut edge of (S_1,S_2) if the end points u, v lie in different sets.

Use an approach similar to Exercise 2.3 (Exercise 3 from Sheet 02), ie. by constructing a very simple randomised algorithm, to prove that there exists a cut of at least $|E|/2$ cut edges for any given graph $G = (V, E)$.

Exercise 3 [8 points]

Consider the following algorithm for the MinCut problem, which contracts the graph down to t nodes and then uses a deterministic algorithm. The value $t \in \{2, \ldots, n\}$ is a parameter whose influence will be analysed below.

ModeratelyFastCut(G, t)

1: $H := G$.
2: **while** H has more than t nodes **do**
3: Choose an edge e from H uniformly at random.
4: $H :=$ contraction(H, e).
5: Compute a minimum cut on H using a deterministic algorithm.

(a) Give a lower bound on the probability that ModeratelyFastCut outputs a given minimum cut, as a function of t and n. (You may cite results from the lecture.)

(b) Assume that the deterministic algorithm has cubic running time. Analyse the running time of ModeratelyFastCut as a function of t and n.

(c) With probability amplification, we can decrease the (upper bound for the) error probability of ModeratelyFastCut to a constant smaller than 1. Give the running time of the resulting algorithm, as a function of t and n. What value of t would you choose to get an efficient algorithm? State your choice of t and the resulting running time.

(d) Compare the running time of ModeratelyFastCut for your choice of t with the running times of Randomised Contraction and FastCut (each with probability amplification to guarantee a constant error smaller than 1).