

Instructions: Please submit your solutions to my office by Noon on Monday, May 5, 2008. You should imagine that this as an **in class, open book exam**: in other words you can use books and notes, but you are expected to work completely independently, *not even to discuss the questions* with other persons. A (*) next to a question indicates that it may be challenging.

- Please write “**these solutions are entirely my own work**” on the first page of your solutions and **then sign your name**.
- Use sufficient detail to make answers and explanations convincing. When giving an algorithm for some task, always give the most efficient one you can find. In any case make sure you give SOME algorithm that can do the task. Always give the complexity of your alg.
- Be explicit if you employ algorithms we have already studied, and state which of their properties you are using (e.g. “... using the Floyd-Rivest selection algorithm with $1.5n + o(n)$ comparisons, *expected*”).
- (**NOTE:**) If you explicitly use material from a book or a paper, or from the web, cite the sources in detail.
- Please try to avoid pseudocode unless it really simplifies the description of an algorithm.
- I gave VERY ROUGH estimates for relative value of the questions, and this is likely to change a little.

1. (35 pts) You are given a graph $G = (V, E)$ with n vertices and m edges. A *3-coloring* of G is an assignment of a color (red, green or blue) to each vertex in such a way that for each edge $e = \overline{xy} \in E$, the endpoints x and y are assigned different colors. In this case we say e is *well-colored*. It may be impossible to color the vertices of G with (only) 3 colors so every edge is well-colored, but we can always compute $OPT(G) \equiv \max |\{e \in E : e \text{ is well-colored}\}|$, the max taken over ALL possible 3-colorings (for example $OPT(K_4) = 5$). If $OPT = m$, G is *3-colorable* (K_4 is *not*).

The computation of OPT is an NP -hard problem called MAX3COLORING. Here is a Monte-Carlo algorithm that is fairly efficient and approximates OPT fairly well:

COLOR(G)

- For each vertex $v \in V$ toss a fair die: if its 1 or 2 color v RED; if its 3 or 4 color v Blue; otherwise color v Green.
- Compute X the number of well-colored edges in the (random) coloring.

END

- (a) What is the running time of this algorithmmm?

- (b) For each $e \in E$ let X_e be the indicator of the event that e is well-colored. Argue that $X = \sum_{e \in E} X_e$. Then compute $\mu = E(X)$ and $Var(X)$, the mean and variance. Show that $\mu \geq \frac{2}{3}OPT$. Explain all your work.
- (c) Let $\varepsilon > 0$ and $\delta > 0$ be given small constants. Carefully describe a Monte-Carlo algorithm using COLOR that finds a coloring of G where $X \geq (1 - \varepsilon)\mu$ with probability at least $1 - \delta$. What is the running time of your algorithm? [can it be done in linear time? why or why not?]
- (d) Now give an efficient Las-Vegas algorithm to produce a coloring with $X \geq (1 - \varepsilon)\mu$. What is its running time? Is this optimal?
- (e) (*) Find efficient algorithms analogous to those in c) and d) above, but which find a coloring with $X \geq \mu$? Carefully describe your work.
2. We are given $A = \{a_1, \dots, a_n\}$ and $B = \{b_1, \dots, b_n\}$, two sets of size n , each of whose elements are (not necessarily distinct) *integers* in $[0, m]$. These sets are inputs to the problem SET EQUALITY in which we must decide if $A = B$, or not. If both sets were given in sorted order, then the answer is YES if and only if $a_i = b_i$ for each $i = 1, \dots, n$. This definition implies an obvious $O(n \log n)$ algorithm. In fact there is a matching $\Omega(n \log n)$ lower bound. We will try for a faster algorithm using probability.
- (a) (12 pts) For A define the polynomial $Q_A(x) = (x - a_1)(x - a_2) \cdots (x - a_n)$, and the analogous one for B . Show $A = B$ if and only if $Q_A - Q_B \equiv 0$ (the zero polynomial). Now, carefully describe a fingerprinting algorithm to test the set equality of A and B . Carefully detail the computations and random choices you make. (If $A = B$, your algorithm should always discover this and say YES. Otherwise, with probability at least $1/2$, the algorithm should discover $A \neq B$ and report NO.)
- (b) (3 pts) What is the running time of your algorithm. Carefully explain.
- (c) (10 pts) If the integers in this problem remain small, it makes sense to treat the operations to evaluate Q_A and Q_B as using *constant time* each. Now show how you could modify the algorithm in a) to cope with many, possibly large inputs. Strive for a fingerprint using $O(\log n + \log \log m)$ bits. Now what is the running time?

3. Let $G = (V, E)$ be an undirected graph with n vertices and m edges. An *independent set* is a set of vertices $S \subseteq V$ so that *NO TWO* of them are connected by an edge of E . We want to construct a large independent set $S \subseteq V$ (i.e., vertices u and v can be in S *only if* the edge $\overline{uv} \notin E$). Recall that $deg(v)$ - the *degree* of vertex $v \in V$ - is the number of edges in E having v as an endpoint. In parts (a) - (d) we treat the case where $d \equiv \max_{v \in V} deg(v)$ is a fixed constant (think of it as 20) independent of n . In later parts we will replace this by a different assumption. Here is an algorithm for the small max-degree case:

RAND

- S is empty
- Generate $\underline{\pi} = (\pi_1, \dots, \pi_n)$, a permutation of $1, \dots, n$, uniformly at random.
- For each $i = 1, \dots, n$, if $\pi_i < \pi_j$ for each j for which $\overline{v_i v_j} \in E$, add v_i to S (i.e., v_i has the minimum π among v_i and all of its at most d neighbors).

END

- (a) (4 pts) What is the running time of RAND? You may have to mention the representation of G that you would use. Now show that the set S it computes is an independent set.
- (b) (2 pts) Let X_v denote the indicator of the event that at the end of the algorithm, $v \in S$. Compute the expectation $E(X_v)$ in terms of $\deg(v)$.
- (c) (4 pts) Let $X = |S|$ denote the size of the independent set returned by RAND. Argue that $X = \sum_{v \in V} X_v$ and that G must contain an independent set of size at least $n/(d+1)$ (d is the max degree).
- (d) (*) (15 pts) Try to derandomize this algorithm using the method of conditional expectations.
- (e) **In the remaining parts of this question we drop the max-degree assumption and instead assume that the graph G has $m = 2n$ edges.** Here is another algorithm to construct S .

First we construct the subgraph $G' = (V', E')$ by random pruning, as follows:

Phase I

- Initially, set $V' = V$.
- For each $v_i \in V$ we toss two fair coins. If *both* are HEAD, v_i remains in V' ; otherwise it is deleted.
- After all n vertices have been subject to this random pruning we take $E' \subseteq E$ as the set of edges in E , *both* of whose endpoints are in V' .

END

Next we get S by more pruning.

PHASE II

- Initially set $S = V'$.
- For each $e \in E'$, if both endpoints are in S , delete one, chosen at random.

END

(5 pts) Find the expected sizes of V' and E' from Phase I and explain your answers. Find the expected size of S . Show that G has an independent set of size at least $n/8$ vertices.

- (f) (5 pts) Why did we use *two* coins in Phase I? Discuss the possibility of doing the pruning in Phase I using only ONE fair coin toss for each vertex. What happens in Phase II?
- (g) (15 pts) Carefully describe an efficient algorithm that will give an independent set of size at least $n/16$ with probability at least $1/15$. What is its running time? Then explain how to convert your algorithm into one that ALWAYS gives an independent set of size at least $n/16$. What can you say about the running time of *this* algorithm?
4. (25 pts) This question is about the random walk on 6 points lying on the the circumference of the unit circle ($C = \{(x, y) : x^2 + y^2 = 1\}$). For $i = 0, \dots, 5$ P_i denotes the point on C defined by

$$P_i = \left(\cos \frac{\pi i}{3}, \sin \frac{\pi i}{3} \right).$$

The process can start at P_i , for any $i \in \{1, 2, 3, 4, 5\}$. It is finished when it reaches P_0 . If it is presently at P_j ($j \neq 0$) it moves to P_{j+1} or to P_{j-1} with probability $1/2$ each (P_6 is P_0).

- (a) Let N_k denote the number of random walk steps starting from P_k to first reach P_0 , $k = 1, 2, 3, 4, 5$. Find the expectations $E(N_k)$, $k = 1, 2, 3, 4, 5$.
- (b) Repeat where now there are 6 points given by $P_i = (\cos \frac{2\pi i}{7}, \sin \frac{2\pi i}{7})$, $i = 0, \dots, 6$.
- (c) The above is a warmup for the analysis of the Papadimitriou 2-SAT algorithm. We have $\phi_1(z_1, \dots, z_n)$ and $\phi_2(z_1, \dots, z_n)$. Both are 2-SAT formulas in n Boolean variables and we are guaranteed that
- if $t^* = (t_1, \dots, t_n) \in [0, 1]^n$ is a satisfying assignment for ϕ_1 then $\overline{t^*}$ satisfies ϕ_2 ($\overline{t^*}$ is t^* with all bits reversed)

We want to satisfy the formula $\phi = \phi_1.or.\phi_2$ and will use Papadimitriou's algorithm. Show that if $n = 2j$ is even, and if ϕ_1 is satisfiable, then the expected number of steps is $O(j^2)$.

- (d) (*) Suppose t^* satisfies ϕ_1 . Let N_k denote the number of steps for termination of Papadimitriou's algorithm applied to ϕ , assuming we start with an assignment $b = (b_1, \dots, b_n) \in [0, 1]^n$ that differs from t^* in k bits. Show that $E(N_j) = j^2$. Find $E(N_k)$ for all $k \in (0, n)$.