CSE 6512 Randomization in Computing. Spring 2016

Exam #1 Solutions

---

1. Consider the following algorithm:

```
for $i := 1$ to $\alpha \log n$ do
    Pick a random $j \in [1, n]$. If $a[j] = 1$ then output "Type II" and quit;
Output: "Type I";
```

Analysis: Note that if the array is of type I, the above algorithm will never give an incorrect answer. Thus assume that the array is of type II. We'll calculate the probability of an incorrect answer as follows.

The output will be incorrect if all of the random elements picked are zeros. The probability of this happening is $\leq \left(\frac{1}{2}\right)^{\alpha \log n} = n^{-\alpha}$.

Thus the output of this algorithm is correct with high probability.

2. Pick a random sample of $c(\alpha+1)\log n$ elements from the array. Identify the element that occurs the most number of times in the sample and output this element. This can be done by sorting the sample elements in $O(\log n \log\log n)$ time.

Let n_1 and n_2 be the number of copies of x and y occurring in the sample, respectively. Applying Chernoff bounds (with $\epsilon = \frac{1}{5}$), we see that

$$Prob.\left[n_1 \leq \frac{2}{5}c(\alpha+1)\log n\right] \leq \exp\left(-\frac{c(\alpha+1)\log n}{100}\right).$$

The above probability will be $\leq n^{-(\alpha+1)}$ if $c \geq 100$. Also,

$$Prob.\left[n_2 \geq \frac{3}{10}c(\alpha+1)\log n\right] \leq \exp\left(-\frac{c(\alpha+1)\log n}{300}\right).$$

The above probability will be $\leq n^{-(\alpha+1)}$ if $c \geq 300$.

Thus if we pick a random sample with $300(\alpha+1)\log n$ elements then n_1 will be larger than n_2 (and n_1 will also be larger than the number of occurrences of any other element) with a probability of $\geq (1 - n^{-\alpha})$.

3. The approach is similar to what we did in class to check if $AB = C$. Here is an algorithm:

```
for $i = 1$ to $\alpha \log n$ do
    Pick a random binary vector $r$ and check if $A^k B^k r = C^k r$;
    if no then output NO and quit;
Output YES
```

Analysis: If $A^k B^k = C^k$, then the above algorithm will never output an incorrect answer. Thus consider the case $A^k B^k \neq C^k$. For a randomly chosen vector r, probability that $A^kB^kr = C^kr$ is no more than $\frac{1}{2}$ as was shown in class. Thus, the probability that $A^kB^kr = C^kr$ on all the $\alpha \log n$ vectors chosen is $\leq \left(\frac{1}{2}\right)^{\alpha \log n} = n^{-\alpha}$.

The computation of A^kB^kr (for any vector r) takes $O(n^2k)$ time since it involves $2k$ matrix-vector products. Similarly, the computation of C^kr takes $O(n^2k)$ time (for any given r). Thus the run time of the algorithm is $O(n^2k \log n)$.

4. The degree of both $[f(x)]^m$ and $[g(x)]^k$ is $N = mn$. We can use an algorithm similar to the one discussed in class. The idea is to pick a random element r from a subset $\mathcal{S}$ of the field and check if $[f(r)]^m = [g(r)]^k$. If yes we output YES else we output NO.

   Analysis: If $[f(x)]^m = [g(x)]^k$, then $[f(r)]^m = [g(r)]^k$ for any r and hence the algorithm will never output an incorrect answer. If $[f(x)]^m \neq [g(x)]^k$, then the probability that $[f(r)]^m = [g(r)]^k$ is $\leq \frac{mn}{|\mathcal{S}|}$ as was shown in class. This probability will be $\leq N^{-\alpha}$ if $|\mathcal{S}| \geq mnN^{\alpha}$.

   The time needed to compute $[f(r)]^m$ is $O(n \log m)$. Likewise the time needed to compute $[g(r)]^k$ is $O(d \log k)$. Thus the time needed to compute both is $O(d \log m)$.

   A deterministic algorithm: Here we use the fact that we can multiply two degree N polynomials in $O(N \log N)$ time. Compute both $[f(x)]^m$ and $[g(x)]^k$ using the repeated squaring technique and check for equality. For simplicity assume that $m = 2^q$ for some integer q. In this case compute the following: $[f(x)]^2, [f(x)]^{2^2}, \ldots, [f(x)]^{2^q}$ by repeated squaring. The total time is $O((n + 2n + \ldots + 2^q n) \log(mn)) = O(mn \log(mn))$.