CS 537 Homework #6 Problem #1 Peilun Dai
Collaborators: None

10/10

1. (Improving guarantees of randomized algorithms)

(a) Solution: Let X denote the number of times needed to get the first non-fail outputs for an input of size n and let $T_n = O(n^2)$ be the time that a single run of the algorithm takes to get an output (either "fail" or correct output). Then the total time of getting a correct answer $T(n) = XT_n$. Let $p \geq 1 - 0.99 = 0.01$ be the probability that the algorithm will output a correct answer in a single run. Then we have $X \sim Geom(p)$. Thus

$$\mathbf{E}[T(n)] = \mathbf{E}[X]O(n^2) \tag{1}$$
$$= \frac{1}{p}O(n^2) \tag{2}$$
$$\leq \frac{1}{0.01}O(n^2) \tag{3}$$
$$= O(n^2) \tag{4}$$

+2

Note that we only stop our runs of the algorithm when we have a correct output, thus this new algorithm always computes a correct answer and runs in expected time $O(n^2)$.

(b) Solution: Let X be the random variable of the running time of algorithm $\mathcal{B}$ on an input of size n. The using Markov inequality, for any positive number b,

$$Pr[X \geq b] \leq \frac{\mathbf{E}[X]}{b} \tag{5}$$
$$= \frac{T(n)}{b} \tag{6}$$
$$\leq 1 - 0.95 = 0.05 \tag{7}$$

+4

We can solve for $b \geq 20T(n)$. Thus we can choose $a = 20$ and run the algorithm $\mathcal{B}$ for $20 \cdot T(n)$ time, and the failure probability is less than 0.05 (i.e. with probability at least 0.95, the new algorithm can solve the problem).

If we know the variable of the algorithm to be at most $\sqrt{n}$, then we can use Chebyshev inequality instead of Markov inequality, for any $b > 0$,

$$Pr[|X - \mathbf{E}[X]| \geq b] \leq \frac{\mathbf{Var}[X]}{b^2} \tag{8}$$
$$Pr[X - \mathbf{E}[X] \geq b] \leq \frac{\mathbf{Var}[X]}{b^2} \tag{9}$$
$$Pr[X \geq T(n) + b] \leq \frac{\mathbf{Var}[X]}{b^2} \tag{10}$$
$$\leq \frac{\sqrt{n}}{b^2} \leq 0.05 \tag{11}$$
$$\tag{12}$$

We can solve for $b \geq 2\sqrt{5}n^{1/4}$. Thus, we can let the $\mathcal{B}$ algorithm run $T(n) + 2\sqrt{5}n^{1/4}$ time, and the success probability will be at least 0.95.

(c) Solution: For the algorithm to succeed if we run C algorithm k times, since we don't know if the incorrect solutions are the same, for the worst case, we need the number of correct solutions to be at least $0.5k$. Then the possible failure case is when the number of correct solution to be less than $0.5k$. Let X_i to be an indicator variable for correct solution returned at run i, and let $X = \sum_i^k X_i$ be the random variable of the number of correct solutions in k runs. $\mathbf{E}[X] = 0.7k$. Using Chernoff bounds, for $\delta = 2/7$,

$$Pr[X \leq (1-\delta)\mathbf{E}[X]] \leq e^{-\mathbf{E}[X]\delta^2/2} \tag{13}$$

$$Pr[X \leq (1-2/7)0.7k] \leq e^{-0.7k(2/7)^2/2} \tag{14}$$

$$Pr[X \leq 0.5k] \leq e^{-k/35} \leq e^{-t} \tag{15}$$

$$-k/35 \leq -t \tag{16}$$

$$k \geq 35t \tag{17}$$

Thus, if we want the algorithm to make a mistake in computing $f(x)$ with probability at most 2^{-t}, then we need to set $k \geq 35t$.

CS 537 Homework #6 Problem #2
Collaborators: None

10/10

Peilun Dai

2. (Exercise 4.10 from MU)

(a) Solution: Let R.V. X_i be the amount of money the casino lost in game i. $\mathbf{E}(X_i) = (3-1)\frac{4}{25} + (100-1)\frac{1}{200} + (1)(1-\frac{4}{25}-\frac{1}{200}) = -0.02$. In expectation, in each game, the casino will make 0.02 dollars. Let X denote the total amount of loss the casino has in the first 1 million games, i.e. $X = \sum_{i=1}^{1,000,000} X_i$ with $\mathbf{E}[X] = 1,000,000 \cdot (-0.02) = -20,000$. Using Theorem 4.12, $a = -1, b = 99$, we can set $\epsilon = 0.03$,

$$Pr\Big[\frac{1}{1,000,000}\sum_{i=1}^{1,000,000} X_i - \mu \geq \epsilon\Big] \leq e^{-2n\epsilon^2/(b-a)} \tag{1}$$

$$Pr\Big[\frac{1}{1,000,000}X - (-0.02) \geq 0.03\Big] \leq e^{-0.18} \tag{2}$$

$$Pr\Big[X \geq 10,000\Big] \leq e^{-0.18} = 0.84 \tag{3}$$

$$\tag{4}$$

(b) Solution: Since X_i are mutually independent,

$$\mathbf{E}[e^{tX}] = \mathbf{E}\Big[\exp\{t\sum_{i=1}^{1,000,000} X_i\}\Big] \tag{5}$$

$$= \mathbf{E}\Big[\prod_{i=1}^{1,000,000} \exp\{tX_i\}\Big] \tag{6}$$

$$= \prod_{i=1}^{1,000,000} \mathbf{E}[e^{tX_i}] \tag{7}$$

$$= \prod_{i}^{1,000,000} \sum_{X_i \in \{-1,2,99\}} Pr[X_i]e^{tX_i} \tag{8}$$

$$= \prod_{i}^{1,000,000} \Big(\frac{4}{25}e^{2t} + \frac{1}{200}e^{99t} + \frac{167}{200}e^{-t}\Big) \tag{9}$$

(c) Solution:

$$Pr[X \geq 10,000] = Pr[e^{tX} \geq e^{10,000t}] \tag{10}$$

$$\leq \frac{\mathbf{E}(e^{tX})}{e^{10,000t}} \tag{11}$$

$$= \frac{\prod_i^{1,000,000}\Big(\frac{4}{25}e^{2t} + \frac{1}{200}e^{99t} + \frac{167}{200}e^{-t}\Big)}{e^{10,000t}} \tag{12}$$

$$= \frac{\prod_i^{1,000,000}\Big(\frac{4}{25}e^{2\cdot 0.0006} + \frac{1}{200}e^{99\cdot 0.0006} + \frac{167}{200}e^{-0.0006}\Big)}{e^{10,000\cdot 0.0006}} \tag{13}$$

$$\approx 0.000160646 \tag{14}$$

where we have used $t = 0.0006$ and also results from (9). Notice this is a march tighter bound compared to the bound in (a).

CS 537 Homework #6 Problem #3 10/10 Peilun Dai
Collaborators: None

3. (Concentration for the running time of Randomized Quicksort)

(a) Solution: Let's consider a path that could achieve the maximum number of good nodes. Based on the information provided, we know the following: 1) Since a bad node along the path will always decrease the size of the array without adding to the number of good nodes, thus this path could only contain good nodes.2) In addition, at each node, the path should routing in the larger size of the partition, otherwise, the larger partition path could contain a path with larger number of good nodes. 3) In order to get the largest depth of the path, at each node, the partition should give a maximum size to the larger subset, i.e. $2/3s$, where s is the size of the set at that node. 4) From point 1) to 3), we conclude that for the largest good node path possible, each partition along the path will divide the set to be of size $2/3s$ and $1/3s$, and the path will always follow the larger subset.

Now, let's compute the length of this path(i.e. the largest number of good nodes for a path). Let k be the number of good nodes along this path. Thus,

$$(\frac{2}{3})^{k+1} n \geq 1 \tag{1}$$
$$k+1 \leq \log_{2/3} 1/n \tag{2}$$
$$\leq 1 - \log_{2/3} n \tag{3}$$
$$\leq 1 - \frac{1}{\log_2 2/3} \log_2 n \tag{4}$$
$$k \leq -\frac{1}{\log_2 2/3} \log_2 n \approx 1.71 \log_2 n \tag{5}$$

We have shown that the maximum number of good nodes is no more than $1.71 \log_2 n$, i.e. $c = 1.71$.

(b) Solution: We get more than t nodes only if in the first t trials we get fewer than $c \log_2 n$ good nodes where c value is from (a). Then we can consider the first t nodes, and model the number of good nodes X as a binomial distribution, i.e. $X \sim Bin(t, 1/3)$. Note in order for a node to be a good node, it must choose the middle $1/3$ of the elements as pivot, and thus, $\mathbf{E}(X) = t/3$. We can set $t = c' \log_2 n$, and apply Chernoff bounds to bound the tail probability to be less than $1/n$.

First, we bound the right hand side of the Chernoff bounds,

$$e^{-\mathbf{E}(X)\delta^2/2} \leq 1/n^2 \tag{6}$$
$$\delta \geq 4/c' \tag{7}$$

Then we plug in the value of δ to the left hand side of Chernoff bounds,

$$Pr[X \leq c \log_2 n] = Pr[X \leq (1-\delta)\mathbf{E}[X]] \tag{8}$$
$$= Pr[X \leq (1-\delta)c' \log_2 n/3] \tag{9}$$
$$\leq Pr[X \leq (1-4/c')c' \log_2 n/3] \tag{10}$$

Now by comparing terms, the following inequality needs to hold

$$c \log_2 n \leq (1-4/c')c' \log_2 n/3 \tag{11}$$
$$c' \geq 3c + 4 \tag{12}$$
$$\geq 3 \cdot 1.71 + 4 \approx 9.13 \tag{13}$$

(c)

Proof. Let's use the results from part (b): with at most probability $1/n^2$, the number of nodes in a given root to leaf path of the tree is greater than $c' \log_2 n$, where c' is a constant we derived from part (b). Let X_i be the root-leaf distance of any path i and let N_p be the total number of path, and et $X_l = \max_{i \in [N_p]} X_i$ be the root-leaf distance of the longest path, and $N_p \leq n$, Using union bound,

$$Pr[X_l > c' \log_2 n] \leq \sum_{i}^{N_p} Pr[X_i > c' \log_2 n] \tag{14}$$

$$\leq \sum_{i}^{n} Pr[X_i > c' \log_2 n] \tag{15}$$

$$\leq \sum_{i}^{n} 1/n^2 = 1/n \tag{16}$$

(d) Solution: Using solutions from (c): with high probability (greater than $1-1/n$), the number of nodes in the longest root to leaf path is not greater than $c' \log_2 n$, at each node along this path, the total comparisons of all nodes with the same depth is less than the total number of elements, n. Thus, the total running time $T(n) \leq n \cdot c' \log_2 n = c' n \log_2 n = O(n \log_2 n)$ with probability at least $1 - 1/n$.