ShanghaiTech University

CS240 Algorithm Design and Analysis

Spring 2024

Problem Set 5

Name: Zhou Shouchen

StudentID: 2021533042

Due: 11:59pm, June 14, 2024

1. Submit your solutions to the course Gradescope.
2. If you want to submit a handwritten version, scan it clearly.
3. You are required to follow ShanghaiTech's academic honesty policies. You are allowed to discuss problems with other students, but you must write up your solutions by yourselves. You are not allowed to copy materials from other students or from online or published resources. Violating academic honesty can result in serious penalties.

Problem 1:

Suppose you have $2n$ balls and 2 bins. For each ball, you throw it randomly into one of the bins. Let X_1 and X_2 denote the number of balls in the two bins after this process. Prove that for any $\varepsilon > 0$, there is a constant $c > 0$ such that the probability $\Pr[X_1 - X_2 > c\sqrt{n}] \leq \varepsilon$.

Solution:

Since we are given that each ball independently selects one of the two bins, both bins equally likely. So we can define the indicators $\mathbb{I}_i$ to represent the whether the i-th ball is in the first bin, i.e.

$$\mathbb{I}_i = \begin{cases} 1, \text{the } i\text{-th ball is in the first bin} \\ 0, \text{the } i\text{-th ball is in the second bin} \end{cases}$$

And from the information above, we can get that all indicators are independent. i.e. $\mathbb{I}_i \overset{i.i.d}{\sim} \text{Bern}\left(\frac{1}{2}\right)$. So $\mathbb{E}(\mathbb{I}_i) = \frac{1}{2}$, $\text{Var}(\mathbb{I}_i) = \frac{1}{4}$.

Then we can define the number of balls in the first bin as $X_1 = \sum_{i=1}^{2n} \mathbb{I}_i$, and the number of balls in the second bin as $X_2 = 2n - X_1$.
Since the indicators are independent, so the expectation and variance both has the linearity property, i.e.

$$\mathbb{E}(X_1) = \mathbb{E}\left(\sum_{i=1}^{2n} \mathbb{I}_i\right) = \sum_{i=1}^{2n} \mathbb{E}(\mathbb{I}_i) = \sum_{i=1}^{2n} \frac{1}{2} = n$$

$$\text{Var}(X_1) = \text{Var}\left(\sum_{i=1}^{2n} \mathbb{I}_i\right) = \sum_{i=1}^{2n} \text{Var}(\mathbb{I}_i) = \sum_{i=1}^{2n} \frac{1}{4} = \frac{n}{2}$$

From what we have learned, the Chebyshev's inequality is given by

$$\Pr[|X - \mathbb{E}(X)| \geq a] \leq \frac{\text{Var}(X)}{a^2}$$

So combine above, we have for any $\epsilon > 0$, there is a constant $c = \sqrt{\frac{2}{\epsilon}} > 0$, s.t.

$$\Pr\left(X_1 - X_2 \geq c\sqrt{n}\right) = \Pr\left(2X_1 - 2n \geq c\sqrt{n}\right) = \Pr\left(X_1 - n \geq \frac{c\sqrt{n}}{2}\right) = \Pr\left(X_1 - \mathbb{E}(X_1) \geq \frac{c\sqrt{n}}{2}\right)$$

$$\leq \Pr\left(|X_1 - \mathbb{E}(X_1)| \geq \frac{c\sqrt{n}}{2}\right) \leq \frac{\frac{n}{2}}{\left(\frac{c\sqrt{n}}{2}\right)^2} = \frac{2}{c^2} = \epsilon$$

So above all, we have proved that for any $\epsilon > 0$, there is a constant $c = \sqrt{\frac{2}{\epsilon}} > 0$ s.t. $\Pr[X_1 - X_2 > c\sqrt{n}] \leq \varepsilon$.

Problem 2:

Suppose that for a certain decision problem, we have an algorithm which computes the correct answer with a probability at least 2/3 on any instance. We wish to reduce the error probability by running the algorithm n times on the same input, using independent randomness between trials, and taking the most common result as the final answer. Using Chernoff bounds, provide an upper bound on the probability that this modified algorithm produces an incorrect result.

Solution:

Let X_i be the indicator of the i-th time's answer. i.e.

$$X_i = \begin{cases} 1, \text{the } i\text{-th time's answer is the correct answer} \\ 0, \text{otherwise} \end{cases}$$

If we consider the upper bound of the probability that prodeces an incorrect result, i.e.

$$P(X_i = 1) = \frac{2}{3}, P(X_i = 0) = \frac{1}{3}$$

Then we can get the expectation of X_i is $\mathbb{E}(X_i) = \frac{2}{3}$.

Then we define the total number of correct answers as

$$X = \sum_{i=1}^{n} X_i$$

From the linearity of expectation, we have

$$\mathbb{E}(X) = \mathbb{E}\left(\sum_{i=1}^{n} X_i\right) = \sum_{i=1}^{n} \mathbb{E}(X_i) = \frac{2n}{3}$$

According to Chernoff bounds:

$$\text{For } 0 \leq \delta \leq 1,\ \Pr(X \leq (1-\delta)\mathbb{E}(X)) \leq \exp\left(-\frac{\delta^2 \mathbb{E}(X)}{2}\right)$$

If we want to get the upper bound of the probability that this modified algorithm produces an incorrect result, we could know that the number of the correct answers is at most $\frac{n}{2}$, i.e.

$$\Pr\left(X \leq \frac{n}{2}\right) = \Pr\left(X \leq \frac{3}{4} \cdot \frac{2n}{3}\right) \leq \exp\left(-\frac{1}{2} \cdot \frac{2n}{3} \cdot \frac{1}{4} \cdot \frac{1}{4}\right) = \exp\left(-\frac{n}{48}\right)$$

Which uses the Chernoff bounds mentioned above, where $\delta = \frac{1}{4}$.
So above all, the upper bound of the probability that this modified algorithm produces an incorrect result is $\exp\left(-\frac{n}{48}\right)$.

Problem 3:

Suppose you have n coins, where the i'th coin has a size $c_i > 0$, and many piggy banks, each with a uniform capacity V, such that $V \geq \max(c_i)$. You want to place all the coins into the minimum number of piggy banks. To do this you use the following greedy strategy.

Start with one active piggy bank. Then, sequentially go through the coins, attempting to place each coin into any active piggy bank where it fits. If a coin does not fit into any active piggy bank, take a new piggy bank and make it active. The algorithm is shown below. Prove this algorithm is a 2-approximation, i.e. it uses at most two times the minimum number of piggy banks needed for all the coins.

Algorithm 1 Piggy Bank Coin Packing

Input: Sizes of coins $c_1, c_2, \ldots, c_n$; size of piggy bank V
Output: Number of piggy banks used
Initialize $b \leftarrow 1$ ▷ current number of active piggy banks
Initialize $P_1, P_2, \ldots \leftarrow 0$ ▷ space used in each piggy bank
for $i = 1$ to n **do**
    if $\exists j \leq b$ such that $P_j + c_i \leq V$ **then**
        Choose any j with $P_j + c_i \leq V$
        $P_j \leftarrow P_j + c_i$ ▷ put c_i in j-th active piggy bank
    else
        $b \leftarrow b + 1$
        $P_b \leftarrow c_i$ ▷ open a new piggy bank and put c_i in it
    end if
end for

Solution:

Suppose that the result generated by the Algorithm 1 mentioned above: The number of piggy banks used is k, and each bank has a size of $P_1, \cdots, P_k$. And suppose that the optimal solution totally uses k^* piggy banks.

And we could prove that: $\forall i, j \in \{1, \cdots, k\}, i \neq j$, then we have $P_i + P_j > V$. This could be proved by contradiction:
Suppose that there exist $i, j \in \{1, \cdots, k\}, i \neq j$, and $P_i + P_j \leq V$.
Then we could get that the total size of the coins in the i-th and j-th piggy banks is $P_i + P_j$, which is less than or equal to V.

So we can just merge the i-th and j-th piggy banks into one piggy bank, and the total number of piggy banks used would be $k-1$, which contradicts the algorithm.

So we could use this property to get that:

$$\sum_{i=1}^{n} c_i = \sum_{i=1}^{k} P_i \geq \frac{k}{2} V \tag{1}$$

And then consider the optimal solution, we could get that:

$$\sum_{i=1}^{n} c_i \leq k^* \cdot V \tag{2}$$

And combine equation 1 and equation 2, we could get that:

$$k^* \cdot V \geq \sum_{i=1}^{n} c_i \geq \frac{k}{2} V$$

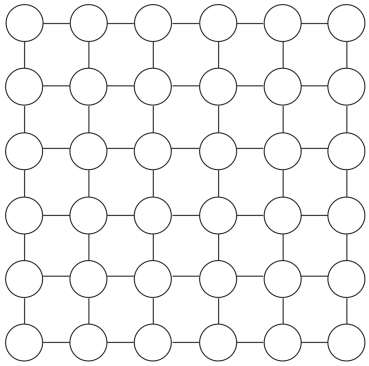
i.e.

$$\frac{k}{2} \leq k^*$$

So above all, we have proved that the algorithm is a 2-approximation.

Problem 4:

Consider an $n \times n$ grid graph G, as shown below.



Each node v in G has a weight $w(v) > 0$. You want to choose an independent set of nodes with maximum total weight. That is, you want to choose a set of nodes S with maximum total weight such that for any $v \in S$, none of v's neighbors are in S. To do this, consider the following greedy algorithm. Let V be the set of all nodes in G. Choose the node in V with the largest weight (breaking ties arbitrarily), add it to the independent set, then remove the node and all its neighbors from V. Repeat this process until V is empty. Let S be the output of this algorithm. Solve the following problems.

1. Let T be any independent set in G. Show that for each node $v \in T$, either $v \in S$, or there is a neighbor v' of v with $v' \in S$ and $w(v) \leq w(v')$.

2. Show that the greedy algorithm is a 4-approximation.

Solution:

1. <1> If $v \in S$, then according to the definition of the independent set, we could get that for any neighbor v' of v, $v' \notin S$.

<2> If $v \notin S$, then according to the independent set, there exist at least one neighbor v' of v such that $v' \in S$.

And according to the greedy algorithm, we could get if for all neighbors v' of v, $v' \in S$ has that $w(v') < w(v)$, the algorithm would choose v before v', which contradicts the greedy algorithm. So there must exist a neighbor v' of v such that $v' \in S$ and $w(v) \leq w(v')$.

So above all, we have proved that for each node $v \in T$, either $v \in S$, or there is a neighbor v' of v with $v' \in S$ and $w(v) \leq w(v')$.

2. Suppose that the greedy algorithm chooses the nodes consisting of S, and the optimal solution chooses the nodes consisting of S^*.

According to 1, we could get that for each node $v \in S$, either $v \in S^*$, or there is a neighbor v' of v with $v' \in S^*$ and $w(v) \leq w(v')$.

<1> If $v \in S$ and $v \in S^*$, then the weight of v contributes the same in both S and S^*.

<2> If $v \in S$ and $v \notin S^*$, then for the most extrame case, all the 4 neighbors of v are in S^*. According to the greedy's policy and analysis in 1, we could know that $w(v) > w(v')$, where v' is any of the neighbors of v. So the most extrame case is that contribue $w(v)$ in S, and at most $4w(v') < 4w(v)$ to S^*.

i.e. we could conclude that if S^* removes a node v from S, then for this modification, removing v and adding its 4 neighbors would increase the total weight of S^* by at most $4w(v)$.

So we could get that the total weight of $S : w(S)$ is at least $\frac{1}{4}w(S^*)$, which is the total weight of S^*.
i.e.

$$w(S) \geq \frac{1}{4}w(S^*)$$

So above all, we have proved that the greedy algorithm is a 4-approximation.