CS 240 Homework 4

吴天元

TOTAL POINTS

5 / 5

QUESTION 1

1 Problem 1 1 / 1

✓ **- 0 pts Correct**

- 0.4 pts Do not have/Incorrect probability that for a fixed min-cut C of G, there is no edge e \in C that is contracted by f(G, t).

- 0.3 pts Incorrect step of the Pr[cut2 (G) returns a min-cut in G]

- 0.3 pts Incorrect answer

QUESTION 2

2 Problem 2 1 / 1

✓ **- 0 pts Correct**

- 0.2 pts The answer of the first part is not correct, which should be "n choose 2"

- 0.2 pts The probability you computed is not correct/did not prove your upper bound/the process is not correct.

- 0.1 pts The iteration time is not enough

- 1 pts wrong

QUESTION 3

3 Problem 3 1 / 1

✓ **- 0 pts Correct**

- 0.1 pts E（T）= 7k/8

- 0.5 pts 从3个元素推广到n个元素并不直观，建议用概率计算代替‘this assignment can always’

- 1 pts Click here to replace this description.

- 0.8 pts obviously?

- 0.5 pts Case2中‘If there are less than 8 clauses, then’?当少于8个clauses，例如考虑三个元素的情况，这七个子句应该都可以满足（相当于忽视八个子句中那个不满足的clause）

- 0.2 pts 从3个元素推广到n个元素并不直观，建议用概率计算代替

- 0.7 pts ‘Moreover, argue that there always exists an assignment satisfying at least’?

QUESTION 4

4 Problem 4 1 / 1

✓ **- 0 pts Correct**

- 0.5 pts No calculation

- 0.5 pts Random algorithm interpretation incorrect

- 0.2 pts Random algorithm interpretation inaccurate

- 0.2 pts Wrong calculation

- 1 pts Wrong

QUESTION 5

5 Problem 5 1 / 1

✓ **- 0 pts Correct**

- 0.8 pts Wrong

- 1 pts Blank

- 0.5 pts Not complete

Problem 1

Solution

First, we calculate the probability that for a single call of $f(G, \lceil n/\sqrt{2}+1 \rceil)$, there is no edge $e \in C$ that is contracted by f.

$$\begin{aligned}
&P(\text{success of } f(G, \lceil n/\sqrt{2}+1 \rceil)) \\
&\geq (1-\frac{2}{n})(1-\frac{2}{n-1})\dots(1-\frac{2}{\lceil n/\sqrt{2}+2 \rceil}) \\
&= \prod_{i=\lceil n/\sqrt{2}+1 \rceil}^{n} (\frac{i-2}{i}) \\
&= \frac{\lceil n/\sqrt{2}+3 \rceil - 2}{n} \cdot \frac{\lceil n/\sqrt{2}+2 \rceil - 2}{n-1} \\
&= \frac{\lceil n/\sqrt{2}+1 \rceil}{n} \cdot \frac{\lceil n/\sqrt{2} \rceil}{n-1} \\
&> (\frac{\lceil n/\sqrt{2} \rceil}{n-1})^2 > (\frac{\lceil n/\sqrt{2} \rceil}{n})^2 \geq \frac{1}{2}
\end{aligned}$$

Hence, we can observe that for each call of $f(G, \lceil n/\sqrt{2}+1 \rceil)$, the probability of no edges contracted is $P > \frac{1}{2}$
Then, let $P(n)$ denotes the probability of $cut2$ gives the correct min cut, and we can observe this recursive equation:

$$\begin{aligned}
P(n) &= 1 - P(cut2(G1)\ fails) \& P(cut2(G2)\ fails) \\
&= 1 - (1 - P(cut2(G1)))(1 - P(cut2(G2))) \\
&\geq 1 - (1 - \frac{1}{2}P(\frac{n}{\sqrt{2}}))^2
\end{aligned}$$

So, we can get:

$$\begin{aligned}
P(n) &\geq 1 - (1 - \frac{1}{2}P(\frac{n}{\sqrt{2}}))^2 \\
&= P(\frac{n}{\sqrt{2}}) - \frac{1}{4}(P(\frac{n}{\sqrt{2}}))^2
\end{aligned}$$

Then we prove a lower bound of $P(n)$ is greater than $\frac{1}{log(n)}$ by induction:
The base case is, if $n = 2$, it's obviously that $P(2) = 1 \geq \frac{1}{log(2)}$, here, we use log as its base of 2
And for general cases:

$$\begin{aligned}
P(n) &\geq P(\frac{n}{\sqrt{2}}) - \frac{1}{4}(P(\frac{n}{\sqrt{2}}))^2 \\
&\geq \frac{1}{log(\frac{n}{\sqrt{2}})} - \frac{1}{4} \cdot (\frac{1}{log(\frac{n}{\sqrt{2}})})^2 \\
&= \frac{1}{log(n) - \frac{1}{2}} - \frac{1}{4(log(n) - \frac{1}{2})^2} \\
&= \frac{4log(n) - 3}{4log^2(n) - 4log(n) + 1} \\
&= \frac{1}{log(n)} + \frac{1 - \frac{1}{log(n)}}{4log^2(n) - 4log(n) + 1} \\
&\geq \frac{1}{log(n)}
\end{aligned}$$

Hence, we've proved a lower bound of $P(n)$ is greater than $\frac{1}{log(n)}$, which is the lower bound of $cut2()$ returns the correct min cut.

1 Problem 1 1 / 1

✓ **- 0 pts Correct**

- 0.4 pts Do not have/Incorrect probability that for a fixed min-cut C of G, there is no edge e \in C that is contracted by f(G, t).

- 0.3 pts Incorrect step of the Pr[cut2 (G) returns a min-cut in G]

- 0.3 pts Incorrect answer

Problem 2

Solution

1) **Upper bound**
The upper bound of the number of different min-cuts in a graph with n vertices is $\binom{n}{2} = \frac{(n-1)n}{2}$.
The proof is, by the Karger's algorithm covered in lecture, we know that the lower bound of the probability of getting correct min-cut by contracting edges (Karger's algorithm):

$$P(\text{success of Karger's algorithm}) \geq \frac{2}{n(n-1)}$$

Then we assume in a graph G, there are k different min-cuts $C_1, C_2, \ldots, C_k$, and let P_i denotes the probability of Karger's algorithm find C_i correctly. By following lemma, we know that for each min-cut C_i, Karger's algorithm has at least $\frac{2}{n(n-1)}$ probability to success.
Notice that these k min-cuts are disjoint from each other, so by the Union bound of disjoint events, we have:

$$P(\text{success of Karger's algorithm}) = \bigcup_{i=1}^{k}(P_i) \geq \frac{2k}{n(n-1)}$$

Notice that $P(\text{success of Karger's algorithm}) \leq 1$, so we can observe:

$$\frac{2k}{n(n-1)} \leq 1$$

Hence

$$k \leq \frac{n(n-1)}{2} = \binom{n}{2}$$

So, the upper bound of number of different min-cuts is $\binom{n}{2} = \frac{(n-1)n}{2}$

2) **Find all min-cuts**
To find all min-cuts, we can keep do the contraction algorithm $2n^2log(n)$ times, and keep track of all min-cuts we have found, that is:

Algorithm 1 Find all min-cuts

```
Initialize CutList = {}, Cardinality of mincut C = ∞
for i = 1, 2, 3, ..., 2n^2 log(n) do
    (curr_cut, cardinality of mincut) = Contraction(G)
    if cardinality of curr_cut < C then
        C = cardinality of mincut
        Assign CutList = {currcut}
    elseif cardinality of curr_cut == C
        Add curr_cut to CutList
    end if
end for
```

Analysis:
As shown in lecture, if we run the contraction algorithm $n^2log(n)$ times, then for each particular min cut C, the upper bound of probability of missing it is $\frac{1}{n^2}$, that is:

$$P(\text{miss C}) \leq \frac{1}{n^2}$$

Problem 2 continued on next page...

So, if we run the contraction algorithm $2n^2log(n)$ times, the upper bound of missing some particular cut is $\frac{1}{n^4}$.
And by the corollary of subproblem (1), the upper bound of the number of min-cuts is $\frac{n(n-1)}{2}$. Hence, the probability that we miss any min-cuts at all is:

$$P(\text{miss some min-cut}) \leq \frac{n(n-1)}{2} \cdot \frac{1}{n^4} = O(\frac{1}{n^2})$$

Thus, we have that, if we run the contraction algorithm $2n^2log(n)$ times, we have $1 - \frac{1}{n^2}$ probability of finding all min-cuts successfully, which is a very simple and elegant approch.

Problem 3

Solution

1) **Algorithm**
The input is a 3-SAT problem with n variables and k clauses.

Algorithm 2 Max-SAT Algorithm

```
repeat
    for i = 1, 2, 3, ..., n do
        Filp a coin
        if Coin lands with head then
            Assign variable x_i to True
        else
            Assign variable x_i to False
        end if
    end for
until number of clauses satisfied ≤ 7k/8
```

2) **Analysis**
As we need at least $\frac{7k}{8}$ clauses satisfied, by the expectation of first success distribution($FS(p)$), we know that the expected number of repeats until the first success is $\frac{1}{p}$, where p is the probability of success each time. Hence we have:

$$\begin{aligned}
\frac{7k}{8} = \mathbb{E}[\text{repeats}] &= \sum_{0 \leq i \leq k} i \cdot p_i \\
&= \sum_{0 \leq i \leq 7k/8} i \cdot p_i + \sum_{7k/8 < i \leq k} i \cdot p_i \\
&\leq (\frac{7k}{8} - \frac{1}{8}) \sum_{0 \leq i \leq 7k/8} p_i + k \sum_{7k/8 < i \leq k} p_i \\
&\leq (\frac{7k}{8} - \frac{1}{8}) + kp(\text{At least} \frac{7k}{8} \text{clauses satisfied})
\end{aligned}$$

Hence we can solve that $p(\text{At least} \frac{7k}{8} \text{clauses satisfied}) \geq \frac{1}{8k}$.
And the expected number of repeats is $\mathbb{E}[N] \leq 8k$.

Problem 3 continued on next page...

2 Problem 2 1 / 1

✓ **- 0 pts Correct**

- 0.2 pts The answer of the first part is not correct, which should be "n choose 2"

- 0.2 pts The probability you computed is not correct/did not prove your upper bound/the process is not correct.

- 0.1 pts The iteration time is not enough

- 1 pts wrong

So, if we run the contraction algorithm $2n^2log(n)$ times, the upper bound of missing some particular cut is $\frac{1}{n^4}$.
And by the corollary of subproblem (1), the upper bound of the number of min-cuts is $\frac{n(n-1)}{2}$. Hence, the probability that we miss any min-cuts at all is:

$$P(\text{miss some min-cut}) \leq \frac{n(n-1)}{2} \cdot \frac{1}{n^4} = O(\frac{1}{n^2})$$

Thus, we have that, if we run the contraction algorithm $2n^2log(n)$ times, we have $1 - \frac{1}{n^2}$ probability of finding all min-cuts successfully, which is a very simple and elegant approch.

Problem 3

Solution

1) **Algorithm**
The input is a 3-SAT problem with n variables and k clauses.

Algorithm 2 Max-SAT Algorithm

```
repeat
    for i = 1, 2, 3, ..., n do
        Filp a coin
        if Coin lands with head then
            Assign variable x_i to True
        else
            Assign variable x_i to False
        end if
    end for
until number of clauses satisfied ≤ 7k/8
```

2) **Analysis**
As we need at least $\frac{7k}{8}$ clauses satisfied, by the expectation of first success distribution($FS(p)$), we know that the expected number of repeats until the first success is $\frac{1}{p}$, where p is the probability of success each time. Hence we have:

$$\begin{aligned}
\frac{7k}{8} = \mathbb{E}[\text{repeats}] &= \sum_{0 \leq i \leq k} i \cdot p_i \\
&= \sum_{0 \leq i \leq 7k/8} i \cdot p_i + \sum_{7k/8 < i \leq k} i \cdot p_i \\
&\leq (\frac{7k}{8} - \frac{1}{8}) \sum_{0 \leq i \leq 7k/8} p_i + k \sum_{7k/8 < i \leq k} p_i \\
&\leq (\frac{7k}{8} - \frac{1}{8}) + kp(\text{At least} \frac{7k}{8} \text{clauses satisfied})
\end{aligned}$$

Hence we can solve that $p(\text{At least} \frac{7k}{8} \text{clauses satisfied}) \geq \frac{1}{8k}$.
And the expected number of repeats is $\mathbb{E}[N] \leq 8k$.

Problem 3 continued on next page...

3) **Proof of "exists an assignment that at least $\frac{7k}{8}$ clauses are satisfied"**
Define random variable R denotes the expectation of number of clauses satisfied, and R_j denotes the j^{th} clause is satisfied, that is:

$$R_j = \begin{cases} 0, & \text{not satisfied} \\ 1, & \text{satisfied} \end{cases}$$

We know that for a clause C_j, it's false if and only if 3 expressions in this clause is all False, for random assignment, the probability of 3 expressions are all False is $(\frac{1}{2})^3 = \frac{1}{8}$
Hence, the probability of C_j is satisfied is $1 - \frac{1}{8} = \frac{7}{8}$
So, the expectation number of satisfied clauses is:

$$R = \mathbb{E}[N] = \mathbb{E}[R_1 + R_2 + \ldots + R_k] = \sum_{i=1}^{k} \mathbb{E}[R_i] = \sum_{i=1}^{k} \frac{7}{8} = \frac{7k}{8}$$

And we know that for a random variable, its at least its expectation some of the time. (If for all times, the value of the random variable less than the expectation, then the expectation should be smaller, which is a contradiction.) So, there is at least one time that $R \geq \frac{7k}{8}$, hence there always exists an assignment that at least $\frac{7k}{8}$ clauses are satisfied

Problem 4

Algorithm
We can generate 3 coin-flips at the first time, which is a binary sequence 000, 001, 010, 011, ..., 111. If we the results are 000 to 101 (i.e. 0-5), we can just make the result of rolling the die to be $i+1$ (i.e. if the coins are 000, then the die is 1, and if the coins are 101, the die is 6). However, if the three coins are 110 or 111, we need to generate 2 more coin-flips(i.e. *second* = 00, 01, 10 and 11). Then, we let the new result be

$$new = \text{first mod } 6 + \text{second} << 1$$

That is, if the first 3 coin-flips gives 111, and the 2 more coin-flips gives 10, then *new* will be $7\%6 + (0b10) << 1 = 0b101 = 5$, then, if $0 \leq new \leq 5$, we make the result of the die to be $new + 1$, otherwise do this process recursively until $0 \leq new \leq 5$
Analysis
It's obviously that the binary sequence that we generated first time is uniformly distributed, so if we only need 3 coin-flips, the result is uniformly distributed. Moreover, the i^{th} recursion is independent of $(i+1)^{th}$ recursion, for they are on different bits of the sequence. Hence the die is uniformly distributed.
The expectation of the number of coin-flips used can be calculated as:

$$\begin{aligned} &\mathbb{E}(\#\text{ of coins used}) \\ &= 3 \cdot \frac{3}{4} + (3+2) \cdot \frac{1}{4} \cdot \frac{3}{4} + (3+2+2) \cdot (\frac{1}{4})^2 \cdot \frac{3}{4} + \ldots + (2n+1)(\frac{1}{4})^{n-1}\frac{3}{4} \ldots \\ &= \sum_{i=1}^{\infty} (2i+1)(\frac{1}{4})^{i-1}\frac{3}{4} \\ &= 3\sum_{i=1}^{\infty} (2i+1)(\frac{1}{4})^{i} \end{aligned}$$

Then, by basic properties of infinite series, we can observe:

$$\mathbb{E}(\#\text{ of coins used}) = 3\sum_{i=1}^{\infty} (2i+1)(\frac{1}{4})^{i} = \frac{11}{3}$$

Hence, our algorithm is correct which has expectation number of coin-flips $\frac{11}{3}$.

3 Problem 3 **1 / 1**

✓ **- 0 pts** **Correct**

- 0.1 pts E□T□= 7k/8

- 0.5 pts □3□□□□□□n□□□□□□□□□□□□□□□□□□□‘this assignment can always’

- 1 pts Click here to replace this description.

- 0.8 pts obviously?

- 0.5 pts Case2□‘If there are less than 8 clauses, then’?□□□8□clauses□□□clause□

- 0.2 pts □3□□□□□□n□□□□□□□□□□□□□□□□□□□

- 0.7 pts ‘Moreover, argue that there always exists an assignment satisfying at least’□

3) **Proof of "exists an assignment that at least $\frac{7k}{8}$ clauses are satisfied"**
Define random variable R denotes the expectation of number of clauses satisfied, and R_j denotes the j^{th} clause is satisfied, that is:

$$R_j = \begin{cases} 0, & \text{not satisfied} \\ 1, & \text{satisfied} \end{cases}$$

We know that for a clause C_j, it's false if and only if 3 expressions in this clause is all False, for random assignment, the probability of 3 expressions are all False is $(\frac{1}{2})^3 = \frac{1}{8}$
Hence, the probability of C_j is satisfied is $1 - \frac{1}{8} = \frac{7}{8}$
So, the expectation number of satisfied clauses is:

$$R = \mathbb{E}[N] = \mathbb{E}[R_1 + R_2 + \ldots + R_k] = \sum_{i=1}^{k} \mathbb{E}[R_i] = \sum_{i=1}^{k} \frac{7}{8} = \frac{7k}{8}$$

And we know that for a random variable, its at least its expectation some of the time. (If for all times, the value of the random variable less than the expectation, then the expectation should be smaller, which is a contradiction.) So, there is at least one time that $R \geq \frac{7k}{8}$, hence there always exists an assignment that at least $\frac{7k}{8}$ clauses are satisfied

Problem 4

Algorithm
We can generate 3 coin-flips at the first time, which is a binary sequence 000, 001, 010, 011, ..., 111. If we the results are 000 to 101 (i.e. 0-5), we can just make the result of rolling the die to be $i+1$ (i.e. if the coins are 000, then the die is 1, and if the coins are 101, the die is 6). However, if the three coins are 110 or 111, we need to generate 2 more coin-flips(i.e. *second* = 00, 01, 10 and 11). Then, we let the new result be

$$new = \text{first mod } 6 + \text{second} << 1$$

That is, if the first 3 coin-flips gives 111, and the 2 more coin-flips gives 10, then *new* will be $7\%6 + (0b10) << 1 = 0b101 = 5$, then, if $0 \leq new \leq 5$, we make the result of the die to be $new + 1$, otherwise do this process recursively until $0 \leq new \leq 5$
Analysis
It's obviously that the binary sequence that we generated first time is uniformly distributed, so if we only need 3 coin-flips, the result is uniformly distributed. Moreover, the i^{th} recursion is independent of $(i+1)^{th}$ recursion, for they are on different bits of the sequence. Hence the die is uniformly distributed.
The expectation of the number of coin-flips used can be calculated as:

$$\begin{aligned} &\mathbb{E}(\#\text{ of coins used}) \\ &= 3 \cdot \frac{3}{4} + (3+2) \cdot \frac{1}{4} \cdot \frac{3}{4} + (3+2+2) \cdot (\frac{1}{4})^2 \cdot \frac{3}{4} + \ldots + (2n+1)(\frac{1}{4})^{n-1}\frac{3}{4} \ldots \\ &= \sum_{i=1}^{\infty} (2i+1)(\frac{1}{4})^{i-1}\frac{3}{4} \\ &= 3\sum_{i=1}^{\infty} (2i+1)(\frac{1}{4})^{i} \end{aligned}$$

Then, by basic properties of infinite series, we can observe:

$$\mathbb{E}(\#\text{ of coins used}) = 3\sum_{i=1}^{\infty} (2i+1)(\frac{1}{4})^{i} = \frac{11}{3}$$

Hence, our algorithm is correct which has expectation number of coin-flips $\frac{11}{3}$.

4 Problem 4 1 / 1

✓ **- 0 pts Correct**

- 0.5 pts No calculation

- 0.5 pts Random algorithm interpretation incorrect

- 0.2 pts Random algorithm interpretation inaccurate

- 0.2 pts Wrong calculation

- 1 pts Wrong

Problem 5

Solution

Consider the recursion tree of the randomized quicksort algorithm, in each level of recursion, we define a "good split" to be: the smaller partition contains at least one quarter of all the elements.
It's obviously that the probability of we get a good split is:

$$Pr(\text{good split}) = \frac{1}{2}$$

For every recursion level t, let $X_{\alpha,t}$ be the indicator random variable for the event of choosing a bad pivot in α's subarray at level t, that is:

$$X_{\alpha,t} = \begin{cases} 1, \text{if at level t, quicksort chose a bad pivot} \\ 0, \text{otherwise.} \end{cases}$$

Then, we take the depth of recursion tree $T = klog(n)$, where c is a arbitrary constant. Also, we define that:

$$X_\alpha = \sum_{i=1}^{T} (X_{\alpha,i})$$

We know that expectation of X is:

$$\mathbb{E}[X_\alpha] = \sum_{i=1}^{T} Pr(X_{\alpha,i}) = \frac{T}{2}$$

Then we use Chernoff's bound, we wish to bound that:

$$Pr(X_\alpha \leq (1+\delta)\mathbb{E}[X_\alpha])$$

When choosing $\delta \leq 1$, and by Chernoff's bound:

$$\begin{aligned} & Pr(X_\alpha \geq (1+\delta)\mathbb{E}[X_\alpha]) \\ & \leq e^{-\delta^2 \mathbb{E}[X_\alpha]/3} \\ & = e^{-\delta^2 \sum_{i=1}^{T} \mathbb{E}[X_{\alpha,i}]/3} \\ & = e^{-\delta^2 T/6} \\ & = e^{-k\delta^2 log(n)/6} \\ & = \frac{1}{n^{k\delta^2/6}} = \frac{1}{n^c} \end{aligned}$$

And when choosing $\delta = 1$ and $k = 12$, we have:

$$Pr(X_\alpha \geq (1+\delta)\mathbb{E}[X_\alpha]) \leq \frac{1}{n^2}$$

Let $\mathbb{E}[\alpha]$ be the event that $\sum_{i=1}^{T} X_{\alpha,i} \geq 2\mathbb{E}[X_\alpha]$, - we have many bad pivots for α. By the union bound:

$$Pr[\text{many bad pivots for some } \alpha] = Pr[\bigcup_{\alpha} E_\alpha] \leq \sum_{\alpha} E[\alpha] = \frac{1}{n}$$

So, with high probability $(1 - \frac{1}{n^c})$, we have that $X < 2\mathbb{E}[X]$. Hence, for any $c > 0$, the probability randomized Quicksort finishes in $O(nlogn)$ time is $\geq 1 - \frac{1}{n^c}$

5 Problem 5 1 / 1

✓ **- 0 pts Correct**

- 0.8 pts Wrong

- 1 pts Blank

- 0.5 pts Not complete