CIS 677

Homework 2

Daniel Sparber

Collaborators: Julian Schnitzler & Seyoung Kim

Problem 1

Algorithm

Let A be the integer representation of Alice's bitstring, let B be the integer representation of Bob's bitstring.

Let $k = \sqrt{2n}$

1. Alice picks $k \cdot \log n$ distinct primes $p_i \in [2..2n \cdot \log n]$, $i \in [1..k]$
2. Bob picks k primes $q_i \in [2..2n \cdot \log n]$, $i \in [1..k]$
3. Alice computes and send all pairs $(p_i, A \bmod p_i)$ and Bob all pairs $(q_i, B \bmod q_i)$ for $i \in [1..k]$ to Charlie.
4. Charlie looks for one pair where $p_i = q_j$ for some $i, j \in [1..k]$. If such a pair exists, Charlie returns whether the second element of the respective pair is equal. Otherwise Charlie returns $A \neq B$.

Communication

Since Alice sends $k \log n \in \mathcal{O}(\sqrt{n} \log n)$ and each pair has $\mathcal{O}(\log n)$ size (see lecture), the total number of bits for the communication is in $\mathcal{O}(\sqrt{n} \cdot \log^2 n)$.

Correctness

Let A and B be arbitrary.

- Case $A = B$

  If Alice and Bob have a prime in common, the algorithm correctly outputs $A = B$ since $A \bmod p = B \bmod p$.

  Otherwise, the algorithm makes a mistake. The probability for having no prime in common can be bound by using the birthday paradox.

  $\Pr[\textit{No primes in common}] = \Pr[\textit{Bob draws non of Alice's primes}] \leq$

$\leq \left(1 - \frac{k \cdot \log n}{2n \cdot \log n}\right)^k = \left(1 - \frac{1}{\sqrt{2n}}\right)^{\sqrt{2n}} \leq \frac{1}{e}$

By repeating $\ln n$ times, the error probability can be reduced to $1 - n$. The communication increases to $\mathcal{O}(\sqrt{n} \cdot \log^3 n)$.

The success probability is therefore $1 - \frac{1}{n}$

- Case $A \neq B$

  The algorithm only fails, if there are some i, j such that $p_i = q_j$ and $A \bmod p = B \bmod p$.

  The probability of having such i, j is less than 1.

  By revising the algorithm from the lecture, the probability of $A \bmod p = B \bmod p$ given $A \neq B$ can be bounded by $\frac{1}{2}$.

  $\Pr[A \bmod p = B \bmod p \mid A \neq P] = \frac{\textit{number of distinct prime divisors of } |A-B|}{\textit{number of distinct primes in } [2..n \log n]}$

  $\leq \frac{n}{\frac{2n \log n}{log(2n \log n)}} \leq \frac{1}{2}$

  Thus, the algorithm fails with probability less equal than $1 \cdot \frac{1}{2}$. By repeating $\log_2 n$ times, the error probability can be reduced to $1 - n$. The communication increases to $\mathcal{O}(\sqrt{n} \cdot \log^3 n)$.

  The success probability is therefore $1 - \frac{1}{n}$

Thus, the algorithm is outputs the correct answer with probability at least $1 - \frac{1}{\textit{poly(n)}}$.

Problem 2

First consider the case, where S and T only differ at one element, i.e. $|T| = |S| - 1$.

Consider the following algorithm:

1. Alice calculates the sum of all her elements S_A and sends it to Bob
2. Bob calculates the sum over all his elements S_B
3. Bob outputs $S_A - S_B$

S_A is at most $\frac{n\cdot(n+1)}{2} \leq n^2$ and can therefore be decoded with $2 \cdot \log n$ bits.

Since there is only one element $x \in S$ that is not in T, the difference of the sums must equal x. Therefore the algorithm outputs an element in $S \setminus T$.

Now, let T and S be arbitrary subsets of $[1..n]$ with $T \subset S$.

Let $d = |S| - |T|$, let n^* be the smallest power of 2 such that $n^* \geq n$.

Since Alice does not know d, she executes the following algorithm:

For $p \in \left\{1, \frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{n^*}\right\}$:

1. Bob and Alice choose $V \subset [1..n]$, $\forall i \in [1..n] : i \in V$ with probability p, from their shared source of randomness
2. Alice computes $S_A^* = \text{sum}(S \cap V)$
3. Alice computes and sends $(S_A^*, |S \cap V|)$
4. Bob: computes $S_B^* = \text{sum}(S \cap T)$
5. If $|S \cap V| = |T \cap V| + 1$: Bob returns $S_A^* - S_B^*$

Note: Instead of sending individual messages, all messages can be combined to one. Bob does his calculations after receiving this one message.

For one iteration the communication is in $\mathcal{O}(logn)$ bits, since both S_A^* and $|A \cap S|$ require less than $2 \cdot \log n$ bits. In total, there are $\log n^*$ repetitions, thus the total communication is in $\mathcal{O}(\log^2 n)$.

Since $(T \cap V) \subset (S \cap V)$, the results from the previous algorithm can be used. With this, a correct result is always returned if $|S \cap V| = |T \cap V| + 1$.

Thus we only need to show, that the probability of $|S \cap V| = |T \cap V| + 1$ is high enough for at least one iteration.

If $|S| - |T| = 1$, the algorithm succeeds with probability 1 in the first iteration. Otherwise the following analysis can be made:

There is one iteration where $p = \frac{1}{2^k}$ for each $k \in [2.. \log n^*]$. Let k' be such that $2^{k'-1} < d \leq 2^{k'}$. Let

$c = 2^{k'}$. Then $\frac{c}{2} < d \leq c \Leftrightarrow \frac{c}{2} + 1 \leq d \leq c$.

$\Pr[\textit{Picking exactly 1 of the } d \textit{ elements when } p = 1/c]$

$\geq \Pr[\textit{Picking exactly 1 of } c/2+1 \textit{ elements when } p = 1/c] =$ *(by binomial distribution)*

$= \binom{c/2+1}{1} \cdot \frac{1}{c} \cdot \left(1 - \frac{1}{c}\right)^{c/2} = (c/2+1) \cdot \frac{1}{c} \cdot \sqrt{\left(1 - \frac{1}{c}\right)^{c}} \geq \frac{1}{2} \cdot \sqrt{\left(1 - \frac{1}{c}\right)^{c}} \geq$ *(by using calculus and $c \geq 2$)*

$\geq \frac{1}{2}\sqrt{\frac{1}{4}} = \frac{1}{4}$

Therefore, the probability of failure is at most $1 - \frac{1}{4} = \frac{3}{4}$.

Instead of executing the algorithm once, we execute it $9 \cdot \log(1/\delta)$ times. The total communication is now in $\mathcal{O}(\log^2 n \log(1/\delta))$.

$\Pr[\textit{failure}] \leq \left(\frac{3}{4}\right)^{9 \cdot \log(1/\delta)} = e^{9 \cdot \log(1/\delta) \cdot \log(3/4)} \leq e^{-\log(1/\delta)} = e^{\log \delta} = \delta$

Thus, the success probability is at least $1 - \delta$.

Problem 3

First consider the case, where all colors are available for a given node v. Since Δ is the maximum degree in G, v has at most Δ neighbors. In the worst case, those Δ neighbors all need to be colored with different colors. Even so, there are still $2\Delta - \Delta = \Delta$ colors remaining to color v. Thus, there are at least Δ good colors for v.

For every node $2\log n$ colors are picked uniformly at random. The following steps calculate a lower bound for the probability of picking at least one good color for every node.

Let $v \in V$ be arbitrary.

$\Pr[\textit{Picking a good color for v with one trial}] \geq \frac{\Delta}{2\Delta} = \frac{1}{2}$

$\Rightarrow \Pr[\textit{Picking a bad color for v with one trial}] \leq \frac{1}{2}$

$\Rightarrow \Pr[\textit{Picking only bad colors for v (with } 2\log n \textit{ trials)}] \leq \left(\frac{1}{2}\right)^{2\log n} = 2^{-2\log n} = n^{-2} = \frac{1}{n^2}$

$\Rightarrow \Pr[\textit{Some node picks only bad colors}] \leq n \cdot \frac{1}{n^2} = \frac{1}{n}$

$\Rightarrow \Pr[\textit{All nodes pick at least one good color}] = 1 - \Pr[\textit{Some node picks only bad colors}] \geq 1 - \frac{1}{n}$

Therefore, the probability that a (2Δ)-coloring of the graph, such that every vertex is assigned only one of the colors it has sampled, exists with propability at least $1 - 1/n$.

Problem 4

Simple randomized algorithm

Algorithm

1. Assign every $v \in V$ with $p = \frac{1}{2}$ to U
2. Return U

Analysis

By definition, there is some maximum directed cut in G (not necessarily unique). Let E_{max} be the set of all edges in that cut.

The probability of any edge $(i, j) \in E_{max}$ being in the cut that the algorithm generates, can be calculated as following:

$\Pr[\textit{(i, j) part of cut}] = \Pr[i \in U] \cdot \Pr[j \in W] = \Pr[i \in U] \cdot \Pr[j \notin U] = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$

Thus, this algorithm yields a 4-approximation.

Integer linear program

z_{ij} is 1 if the edge $(i, j) \in E$ is part of the cut, otherwise 0.

x_i is 1 if $i \in U$, otherwise 0.

Maximizing $\sum\limits_{(i,j) \in E} z_{ij}$ will therefore maximize the number of edges going from U to W.

Every $i \in V$ is either part of U or not, there is no in between. This is enforced by $\forall i \in V, x_i \in \{0, 1\}$.

z_{ij} can be 1 only if $i \in U$ and $j \in W$, otherwise it needs to be 0. This is strictly enforced by the three constraints $\forall (i, j) \in E, z_{ij} \leq x_i$, $\forall (i, j) \in E, z_{ij} \leq 1 - x_j$ and $\forall (i, j) \in E, 0 \leq z_{ij} \leq x_i$ combined with the fact, that $x_i \in \{0, 1\}$

Therefore the ILP correctly models the maximum directed cut problem

LP relaxation

The constraint $\forall i \in V, x_i \in \{0, 1\}$ is changed to $\forall i \in V, x_i \in [0, 1]$.

Claim. The integrality gap for the complete graph graph converges to 2, as $|V| \to \infty$.

Proof. For the complete graph, assigning $x_i = 0.5, i \in V$ and assigning $z_{i,j} = 0.5, (i, j) \in E$ for the LP relaxed problem yields $\frac{|E|}{2}$.

However, the best ILP solution is assigning half of the nodes to U, which converges to $\frac{|E|}{4}$ for the complete graph, as $|V| \to \infty$.

Thus the integrality gap equals 2.

Rounding scheme

The proposed rounding scheme returns a valid directed cut by construction, since every vertex is either assigned to U or not U (i.e. W).

Let $(i,j) \in E$ be arbitrary.

$\Pr[(i,j) \ in \ cut] = \Pr[i \in U, j \notin U]$

$= \left(\frac{1}{4} + \frac{x_i}{2}\right) \cdot \left[1 - \left(\frac{1}{4} + \frac{x_j}{2}\right)\right] = \left(\frac{1}{4} + \frac{x_i}{2}\right) \cdot \left(\frac{3}{4} - \frac{x_j}{2}\right) = \left(\frac{1}{4} + \frac{x_i}{2}\right) \cdot \left(\frac{1}{4} + \frac{1-x_j}{2}\right)$

$\geq \left(\frac{1}{4} + \frac{z_{ij}}{2}\right) \cdot \left(\frac{1}{4} + \frac{z_{ij}}{2}\right) = \frac{1}{16} + \frac{z_{ij}}{4} + \frac{z_{ij}^2}{4} = \frac{1}{16} - \frac{z_{ij}}{4} + \frac{z_{ij}^2}{4} + \frac{z_{ij}}{2} = \left(\frac{1}{4} - \frac{z_{ij}}{2}\right)^2 + \frac{z_{ij}}{2}$

$\geq \frac{z_{ij}}{2}$

With this probability the expected value of number of edges going from U to W can be calculated.

$\mathbb{E}[\textit{Total number of edges going from U to W}] = \sum\limits_{(i,j)\in E} 1 \cdot Pr[(i,j) \ in \ cut]$

$\geq \sum\limits_{(i,j)\in E} \frac{z_{ij}}{2} = \frac{1}{2} \cdot \underbrace{\sum_{(i,j)\in E} z_{ij}}_{OPT_{LP}} \geq \frac{1}{2} \cdot OPT$

Thus, the proposed scheme yields a 2-approximation. Therefore, the integrality gap is at most 2.

Problem 5

We use the same LP relaxation as described in the lecture.

Let $j \in [1..n]$ be an arbitrary element.

From the lecture, we know:

$\Pr[j \textit{ is not covered}] \leq \frac{1}{e}$

Instead of repeating $\log n$ times to get a valid solution, we now use a different approach:

Repeat $\log \Delta$ times to get a better probability:

$\Pr[j \textit{ is not covered after } \Delta \textit{ tries}] \leq \left(\frac{1}{e}\right)^{\log \Delta} = \frac{1}{\Delta}$

After that, for every uncovered element, pick the set with the smallest weight (greedy).

Suppose $OPT = \{S_1, \ldots, S_k\}$ is an optimal solution. Every S_i of the solution covers at most Δ elements. Thus, the cost of picking sets by the greedy step is at most Δ times worse than the optimal solution.

Therefore, in total the expected cost increases by $\Delta \cdot \frac{1}{\Delta} = 1$ when doing the greedy step. We get a $(\log \Delta)$-approximation.