CSE 632: Analysis of Algorithms II: Randomized Algorithms **Spring 2024**

Lecture 1: Examples of Randomized Algorithms

Lecturer: Zongchen Chen

1 Maximum Cut

Let $G = (V, E)$ be a graph. For a subset $S \subseteq V$ of vertices, the cut set is defined as

$$E(S, V \setminus S) = \{uv \in E : u \in S, v \in V \setminus S\}.$$

The max cut of G is defined as

$$\textsf{max-cut}(G) = \max_{S \subseteq V} |E(S, V \setminus S)|.$$

Finding the max cut is $\textsf{NP}$ hard. However, one can show that the max cut always contains at least half of the edges, using a randomized algorithm.

Lemma 1. *For every graph $G = (V, E)$, it holds*

$$\textsf{max-cut}(G) \geq \frac{|E|}{2}. \tag{1}$$

Proof. We prove the lemma by finding a subset $S \subseteq V$ such that $|E(S, V \setminus S)| \geq |E|/2$. Instead of constructing S explicitly or deterministically, we consider a randomized algorithm which just outputs a subset $S \subseteq V$ uniformly at random. More precisely, $S \subseteq V$ is constructed randomly as follows.

Algorithm 1 Generating $S \subseteq V$ uniformly at random

$S \leftarrow \emptyset$;
for all $v \in V$ independently **do**
    Flip a *fair* coin;
    if Head **then**
        $S \leftarrow S \cup \{v\}$;
    else
        $S \leftarrow S$;
    end if
end for
    return S

We show that

$$\mathbb{E}[|E(S, V \setminus S)|] = \frac{|E|}{2}. \tag{2}$$

Observe that, Eq. (2) implies the existence of a subset $S \subseteq V$ such that $|E(S, V \setminus S)| \geq |E|/2$, since the "maximum value" must be no less than the "average value":

$$\textsf{max-cut}(G) = \max_{S \subseteq V} |E(S, V \setminus S)| \geq \mathbb{E}[|E(S, V \setminus S)|] = \frac{|E|}{2}.$$

We now prove Eq. (2). For each edge $e \in E$, define an indicator random variable by

$$X_e = \begin{cases} 1, & \text{if } e \in E(S, V \setminus S); \\ 0, & \text{otherwise.} \end{cases}$$

For each $e = uv \in E$, observe that X_e is a Bernoulli random variable with expectation

$$\begin{aligned}\mathbb{E}[X_e] &= \Pr(X_e = 1) = \Pr\left(e \in E(S, V \setminus S)\right) \\ &= \Pr\left(u \in S, v \notin S\right) + \Pr\left(u \notin S, v \in S\right) = \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2}.\end{aligned}$$

It follows that

$$\mathbb{E}[|E(S, V \setminus S)|] = \mathbb{E}\left[\sum_{e \in E} X_e\right] = \sum_{e \in E} \mathbb{E}[X_e] = \sum_{e \in E} \frac{1}{2} = \frac{|E|}{2},$$

where we use the linearity of expectation: for two random variables X and Y (they can be dependent) it holds $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$. □

2 Coupling

Consider the process of generating a uniformly random subset $S \subseteq V = \{1, 2, \ldots, 10\}$ as described in Algorithm 1. Let S be the (random) output of Algorithm 1. Now consider the same process but with a biased coin instead of a fair one; more precisely, suppose $\Pr(\text{Head}) = 0.51$ so that each element is included in S independently with a larger probability 0.51. Denote the (random) output using such a biased coin by T.

Lemma 2. *Show that*

$$\Pr\left(\sum_{i \in S} i \geq 30\right) < \Pr\left(\sum_{i \in T} i \geq 30\right).$$

Proof. While it is true that each i is strictly more likely to be added to T than S, and that $\mathbb{E}\left[\sum_{i \in S} i \geq 30\right] < \mathbb{E}\left[\sum_{i \in T} i \geq 30\right]$, these do not immediately imply the lemma. To prove a lemma, we define a random process that generates both S and T *simultaneously*; in particular, the outputs S and T are not independent and always satisfy $S \subseteq T$.

Algorithm 2 Generating $S, T \subseteq V$ simultaneously

$S, T \leftarrow \emptyset$;
for all $i \in [10]$ independently **do**
  With probability 0.5: $S \leftarrow S \cup \{i\}$ and $T \leftarrow T \cup \{i\}$;
  With probability 0.01: $S \leftarrow S$ and $T \leftarrow T \cup \{i\}$;
  With probability 0.49: $S \leftarrow S$ and $T \leftarrow T$.
end for
  return S, T

Observe that, both S and T are distributed as required. However, they are not independent and we always have $S \subseteq T$, i.e., $\Pr(S \subseteq T) = 1$. Furthermore, if $\sum_{i \in S} i \geq 30$ then $\sum_{i \in T} i \geq 30$ since $S \subseteq T$. Therefore,

$$\Pr\left(\sum_{i \in S} i \geq 30\right) < \Pr\left(\sum_{i \in T} i \geq 30\right).$$

Note that the inequality is strict because it is possible (happens with positive probability) to output $S = \emptyset$ and $T = [10]$ at the same time. □

3 Matrix Multiplication Verification

The matrix multiplication verification problem is as follows.

- Input: $A, B, C \in \mathbb{Z}^{n \times n}$.
- Decide $AB = C$ vs $AB \neq C$.

Algorithm 3 Straightforward algorithm

Input: $A, B, C \in \mathbb{Z}^{n \times n}$

Compute $C' = AB$;

if $C = C'$ **then**

  return Yes;

else (namely $C \neq C'$)

  return No;

end if

Straightforward algorithm. The running time of Algorithm 3 depends on how fast we can multiply two $n \times n$ matrices. Doing it directly by definition requires $O(n^3)$ time. The current fastest algorithm for matrix multiplication runs in $O(n^{2.37\cdots})$ time.

Algorithm 4 Freivalds' algorithm

Input: $A, B, C \in \mathbb{Z}^{n \times n}$

Sample $x \in \{0,1\}^n$ uniformly at random;

Compute $y = ABx$ and $y' = Cx$;

if $y = y'$ **then**

  return Yes;

else (namely $y \neq y'$)

  return No;

end if

Randomized algorithm. Freivalds' algorithm (Algorithm 4) is a randomized algorithm for matrix multiplication verification. The running time of Algorithm 4 is $O(n^2)$; this is because we can avoid matrix multiplication by the associative property

$$(AB)x = A(Bx)$$

and compute $y = ABx$ with only matrix-vector multiplication.

If $AB = C$, then no matter what $x \in \{0,1\}^n$ is chosen it holds $ABx = Cx$ and hence Algorithm 4 outputs "Yes" always. If $AB \neq C$, we show in the next lemma that $ABx \neq Cx$ for at least half of the vectors $x \in \{0,1\}^n$ and hence Algorithm 4 outputs "No" with probability at least $1/2$.

Lemma 3. *If $AB \neq C$, then* $\Pr(ABx \neq Cx) \geq 1/2$.

Proof. Let $D = AB - C \neq 0$. We need to show $\Pr(Dx \neq 0) \geq 1/2$. Since $D \neq 0$, there exist $i, j \in [n]$ such that $D_{ij} \neq 0$. Consider the i'th entry of Dx, which is given by

$$(Dx)_i = \sum_{k=1}^{n} D_{ik} x_k = \left(\sum_{k \neq j} D_{ik} x_k \right) + D_{ij} x_j.$$

Observe that, flipping the value of x_j changes the value of $(Dx)_i$ since $D_{ij} \neq 0$. Thus, we have $Dx \neq Dx'$ where x' is obtained from x by flipping the j'th entry, and in particular we have either $Dx \neq 0$ or $Dx' \neq 0$. We can then pair up all vectors in $\{0,1\}^n$ such that in each pair the two vectors differ exactly at the j'th entry, and hence at least one of them satisfies $Dx \neq 0$. This implies that $\Pr(Dx \neq 0) \geq 1/2$ since x is chosen uniformly at random. □

The success probability of Algorithm 4 is summarized in the table below.

	Pr(**return** Yes)	Pr(**return** No)
$AB = C$	1	0
$AB \neq C$	$\leq \frac{1}{2}$	$\geq \frac{1}{2}$

Note that in the $AB \neq C$ case, we can only guarantee a success probability of at least $1/2$. However, we can boost up the success probability by running multiple trials. More precisely, we run Algorithm 4 for k times independently, and return "Yes" if Algorithm 4 outputs "Yes" in all these k trials, and return "No" if Algorithm 4 outputs "No" in at least one of the k trials. Then, in the $AB \neq C$ case, we get "Yes" with probability at most $1/2^k$, which can be arbitrarily small by choosing a large enough k. The success probability is summarized below.

	Pr(**return** Yes in all k trials)	Pr(**return** No in one of k trials)
$AB = C$	1	0
$AB \neq C$	$\leq \frac{1}{2^k}$	$\geq 1 - \frac{1}{2^k}$

If $k = 10$, then the success probability is already at least $1 - 1/2^{10} > 0.999$.