Advanced Algorithms X (Fall 2020)

Instructor: Chihao Zhang
Scribed by: Haobo Ma, You Lv

Last modified on Nov 12, 2020

In this lecture, we first introduce Wald's equation which is a useful tool for analyzing cost of algorithms while the number of iterations is itself random. We prove it using OST. Then we talk about the probabilistic method in the algorithm design. Finally, we introduce the Lovász Local Lemma.

1 Wald's Equation

While analyzing randomized algorithms, it is common to meet following procedures:

Algorithm 1

```
1: while Cond do
2:     Compute();
3: end while
```

Assume each calling of `Compute()` costs X time and the procedure terminates in T iterations where both X and T are random variables. The total time cost is therefore $N \triangleq \sum_{i=1}^{T} X$. The question is how to compute the $\mathbf{E}[N]$.

The Wald's equation states a sufficient condition for $\mathbf{E}[N] = \mathbf{E}[T] \cdot \mathbf{E}[X]$ to hold.

Theorem 1 (Wald's Equation). *If we have*

- $X_1, X_2, \dots$ *are non-negative, independent, identically distributed random variables with distribution same as* X*;*
- T *is a stopping time for* $\{X_t\}_{t \geq 1}$*;*
- $\mathbf{E}[T], \mathbf{E}[X] < \infty$.

Then $\mathbf{E}\left[\sum_{i=1}^{T} X_i\right] = \mathbf{E}[T] \cdot \mathbf{E}[X]$.

The proof relies on the optional stopping theorem we met before:

Theorem 2. *Let* $\{X_t\}_{t \geq 0}$ *be a martingale and* τ *be a stopping time with respect to* $\{\mathcal{F}_t\}_{t \geq 0}$. *Then* $\mathbf{E}[X_\tau] = \mathbf{E}[X_0]$ *if* at least one of *the three following conditions holds:*

(1) τ *is bounded, or*

(2) $\mathbf{Pr}[\tau < \infty] = 1$, *and there is a finite* M *such that* $|X_t| \leq M$ *for all* $t < \tau$, *or*

(3) *$\mathbf{E}[\tau] < \infty$, and there is a constant c such that $\mathbf{E}[|X_{t+1} - X_t| \mid \mathcal{F}_t] \le c$ for all $t < \tau$.*

Proof of Theorem 1. For $i \ge 1$, let $Z_i := \sum_{j=1}^{i}(X_j - \mathbf{E}[X])$. Then it is clear that $\{Z_i\}_{i\ge 1}$ is a martingale with respect to $\{\mathcal{F}_i\}_{i\ge 1}$ where $\mathcal{F}_i = \sigma(X_1, \ldots, X_i)$ is the sigma algebra generated by $X_1, \ldots, X_i$. We have

$$\mathbf{E}[|Z_{i+1} - Z_i| \, |\mathcal{F}_i] = \mathbf{E}[|X_{i+1} - \mathbf{E}[X]| \, |\mathcal{F}_i] \le \mathbf{E}[X_{i+1} + \mathbf{E}[X] \ |\mathcal{F}_i] \le 2\mathbf{E}[X] .$$

Since we have that $\mathbf{E}[T], \mathbf{E}[X] < \infty$, the condition of Theorem 2 meets. This implies that $\mathbf{E}[Z_T] = \mathbf{E}[Z_1] = 0$. On the otherhand, we have

$$0 = \mathbf{E}[Z_T] = \mathbf{E}\left[\sum_{j=1}^{T}\left(X_j - \mathbf{E}[X]\right)\right] = \mathbf{E}\left[\sum_{i=1}^{T} X_i - T \cdot \mathbf{E}[X]\right] = \mathbf{E}\left[\sum_{i=1}^{T} X_i\right] - \mathbf{E}[T] \cdot \mathbf{E}[X] .$$

□

1.1 Applications of Wald's Equation

Rolling dices

Let's first consider the following process of tossing dices.

- Roll a dice to get X.
- Roll X independent dices and calculate the sum N.

It follows from Wald's equation that

$$\mathbf{E}[N] = \mathbf{E}[X] \cdot \mathbf{E}[X] = \frac{7}{2} \cdot \frac{7}{2} = \frac{49}{4}.$$

Routing

As shown in Figure 1, there are n senders and one receiver. Senders need to send packets to the receiver via a single channel. Each sender will propose to send a packet to the receiver with probability $\frac{1}{n}$ per second. However, if there are multiple packets proposed to send at the same time, all of them will fail. The problem is to determine the expected time to wait until each sender successfully sends at least one packet.

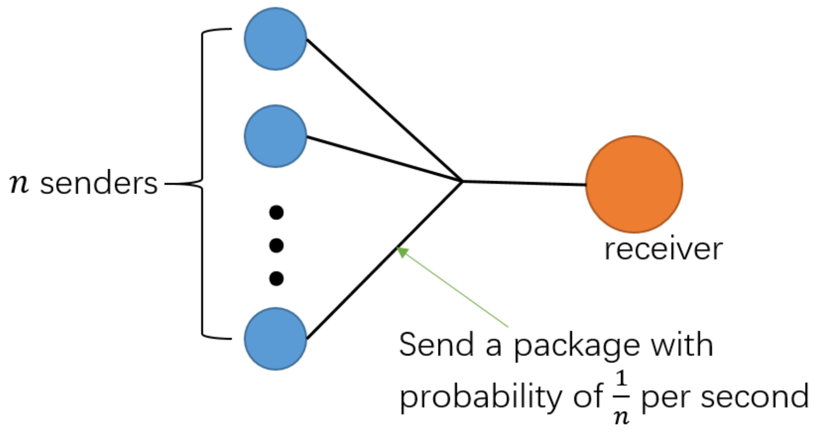

Figure 1: n senders and one receiver

To solve this problem, we first let X_i be how long the receiver needs to get another packet after it has received $i-1$ packets (packets from the same sender also counts). And let T be the number of all packets received when each sender has successfully sent one packet. Now we define:

$$N \triangleq \sum_{i=1}^{T} X_i$$

It is clear that all X_i's are identically distributed. By Wald's equation, $\mathbf{E}[N] = \mathbf{E}[T] \cdot \mathbf{E}[X_1]$. The value of $\mathbf{E}[T]$ is exactly what we need to calculate.

Each time the receiver receives a new packet, its sender is uniform in $[n]$ as all senders are symmetric. Therefore, T is a random variable we met in the coupon collector problem and its expectation $\mathbf{E}[T] = nH_n \approx n \log n$. On the other hand, we have

$$\mathbf{Pr}[X_1 = 1] = n \cdot \frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1} \approx e^{-1}.$$

X_1 is a geometric variable, therefore $\mathbf{E}[X_1] \approx e$ and $\mathbf{E}[N] \approx en \log n$.

2 The Probabilistic Method

In the class of Combinatorics, we have already learnt the probability method. To prove the existence of some combinatorial structures, we can design a probability space and show that $\mathbf{Pr}[\text{the object exists}] > 0$. In computer science, an existence proof is not enough. We always need to find the object *efficiently*.

2.1 MaxCut

Given an undirected graph $G = (V, E)$, the max cut of G is the partition $V = S \cup \overline{S}$ such that $|E(S, \overline{S})|$ is maximized.

Fact 1. *Each graph $G = (V, E)$ contains a cut of size at least $\frac{|E|}{2}$*

Proof. We can find a partition $(S, \overline{S})$ by tossing a fair coin at each vertex v. If the coin gives HEAD, we put v in S, otherwise, put v in $\overline{S}$. We let $X = \left|E(S, \overline{S})\right|$ be the number edges between S and $\overline{S}$. Then $\mathbf{E}[X] = \sum_{e \in E} \mathbf{Pr}[e \text{ is in the cut}] = \frac{|E|}{2}$. So $\mathbf{Pr}\left[X \geq \frac{|E|}{2}\right] > 0$ □

Can we turn the existence proof into an algorithm to find such a cut? It is straightforward to turn the argument into a Las-Vegas algorithm: Repeat generating random partition $(S, \overline{S})$ until we meet one with $E(S, \overline{S}) \geq \frac{|E|}{2}$. Let p be the probability that our algorithm terminates in one round, i.e., $p = \mathbf{Pr}\left[\left|E(S, \overline{S})\right| \geq \frac{m}{2}\right]$, where $m = |E|$. Then

$$\frac{m}{2} = \mathbf{E}[X] = \sum_{i=0}^{m} i \cdot \mathbf{Pr}[X = i] \leq \left(\frac{m}{2} - 1\right)(1 - p) + pm.$$

So $p \geq \frac{2}{m+2}$, and the expect running time of our algorithm is $\frac{1}{p} \leq \frac{m+2}{2}$. Define the approximation ratio of a maxcut algorithm $\mathcal{A}$ by $\alpha(\mathcal{A}) = \min_G \frac{\mathcal{A}(G)}{\mathrm{OPT}(G)}$, then we obtained a polynomial-time randomized approximation algorithm with approximation ratio 0.5.

2.2 Derandomization

We can use the method of conditional expectation to get a deterministic algorithm. We use a random variable $X_i \in \{-1, 1\}$ to represent that node i is in S or $\overline{S}$. Then

$$\frac{m}{2} = \mathbf{E}[X] = \mathbf{E}[\mathbf{E}[X \mid X_1, X_2, ..., X_n]] \overset{(\heartsuit)}{=} \frac{1}{2}\mathbf{E}[\mathbf{E}[X \mid X_1 = 1, X_2, ..., X_n]] + \frac{1}{2}\mathbf{E}[\mathbf{E}[X \mid X_1 = -1, X_2, ..., X_n]],$$

where in $(\heartsuit)$ we used the fact that all X_i are independent.

Then we know that either $\mathbf{E}[\mathbf{E}[X \mid x_1 = 1, x_2, ..., x_n]] \geq \frac{m}{2}$ or $\mathbf{E}[\mathbf{E}[X \mid x_1 = -1, x_2, ..., x_n]] \geq \frac{m}{2}$. Since we can calculate both conditional expectations efficiently in the same manner as we calculated $\mathbf{E}[X]$, we can fix the value of X_1 by choosing the one with greater conditional expectation. We can do the same to fix the value of $X_2, X_3, ..., X_n$.

3 Lovász Local Lemma

Lovász Local Lemma is a powerful and widely used tool in the probabilistic method. In many problems, a certain object exists if none of a family of bad events happens. We usually use $B_1, ..., B_n$ to denote the bad events and need to prove $\mathbf{Pr}\left[\overline{B_1} \cap \overline{B_2} \cap ... \cap \overline{B_n}\right] > 0$. Assume that $\mathbf{Pr}[B_i] = p < 1$. Two extreme cases are

- $\forall i \neq j, B_i \cap B_j = \emptyset$. Then we have $\mathbf{Pr}\left[\overline{B_1} \cap \overline{B_2} \cap ... \cap \overline{B_n}\right] = 1 - np$. Therefore, we need $p < \frac{1}{n}$ to guarantee $\mathbf{Pr}\left[\overline{B_1} \cap \overline{B_2} \cap ... \cap \overline{B_n}\right] > 0$.
- Events in $\{B_i\}_{1 \leq i \leq n}$ are mutually independent. Then we have $\mathbf{Pr}\left[\overline{B_1} \cap \overline{B_2} \cap ... \cap \overline{B_n}\right] = (1 - p)^n$ which is positive for any $p < 1$.

The dependency between bad events is crucial to the probability we are interested in. The Lovász Local Lemma introduces the notion of dependency graphs which characterizes the property.

Definition 3. *A dependency graph for a set of events $B_1, \ldots, B_n$ is a graph $G = (V, E)$ such that $V = \{1, \ldots, n\}$ and for $i = 1, \ldots, n$, event B_i is mutually independent of the events $\{B_j | (i, j) \notin E\}$*

Theorem 4 (Lovász Local Lemma: symmetric version). *Let $B_1, \ldots, B_n$ be a set of events, and assume that the following hold:*

1. *$\forall i, \mathbf{Pr}[i] \leq p$;*
2. *the degree of the dependency graph given by $B_1, \ldots, B_n$ is bounded by d;*
3. *$4pd \leq 1$.*

Then $\mathbf{Pr}\left[\bigcap_{i=1}^{n} \overline{B_i}\right] > 0$.

We will prove a more general version in the next lecture.

3.1 Application

Definition 5. *A k-CNF formula ϕ is of the form $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$, where $C_i = x_{i1} \vee \ldots \vee x_{ik}$ and x_{ij} is a variable appeared either positively or negatively. The k-SAT problem is to determine whether a given k-CNF formula ϕ is satisfiable.*

Definition 6. *The degree of a variable x is the number of clauses that x or $\neg x$ belongs to.*

Theorem 7. *Let d be the maximum degree of variables in a k-CNF ϕ. if $4k(d-1) \leq 2^k$, then ϕ is satisfiable.*

Proof. Let V be the set of variables of ϕ. Assume ϕ has n variables and m clauses. The probability space is the uniform distribution over $\{\texttt{true}, \texttt{false}\}^V$, or equivalently a uniform assignment of V. Each clause C_i defines a bad event $B_i \triangleq$ "C_i is not satisfied". So ϕ is satisfiable iff $\mathbf{Pr}\left[\bigcap_{i=1}^{m} \overline{B_i}\right] > 0$. We know that each clause C_i satisfies $\mathbf{Pr}\left[\overline{B_i}\right] = 2^{-k}$. Note that two clauses are dependent only if they share some variables. For a clause C_i, it has at most k variables and each variable shows in at most $d-1$ other clauses. So the maximum degree of the dependency graph is at most $k(d-1)$. Then the theorem follows from the LLL. $\square$

References