0368-4159: First Course in Derandomization 05/11/2018 – Lecture 4

Derandomizing MIS and Primality Testing

Lecturer: Amnon Ta-Shma

Scribe: Or Karni

1 Maximal Independent Set - MIS

We begin with a short remainder. The problem of *MIS* is the following: We get as input a graph $G = (V, E)$. The output should be an independent set $A \subseteq V$ that cannot be further expanded, meaning that for all $v \notin A$, $A \cup \{v\}$ is not independent. In the last lecture we have seen a probabilistic algorithm for the problem. The probabilistic algorithm works in rounds. In each round we compute a total ordering on the vertices, such that smaller degree vertices are smaller (the degree is with the respect to the ramining graph at that stage). In each round, each vertex v suggests itself as a candidate with probability $\frac{1}{2d(v)}$. Then, we solve *conflicts*: if two neighboring vertices both suggested themselves as candidates, then the smaller vertex (with the lower degree) is disqualified. At the end of the round, we add all the surviving candidates to our result I and remove them and their neighbors from the graph.

We defined the set $\mathit{Good} \subseteq V$ to contain all vertices that have at least $\frac{1}{3}$ smaller neighbors. Our two main Lemmas were:

Lemma 1.

$$\Pr_{(v,w)\in E}[v \in \mathit{Good} \vee w \in \mathit{Good}] \quad \geq \quad \frac{1}{2}.$$

Lemma 2. *There exists a constant $\alpha > 0$ such that for every $v \in \mathit{Good}$,*

$$\Pr[v \in (I \cup \Gamma(I))] \quad \geq \alpha.$$

where the probability is over the randomness of the probabilistic algorithm.

Lemma 1 was proved last lecture.

1.1 Proving lemma 2 and completing the proof

We begin with a simple claim:

Claim 3. *For every $v \in V$, $\Pr[v \text{ is disqualified} \mid v \text{ is a candidate}] \leq \frac{1}{2}$.*

Proof. v is disqualified when it has a larger neighbor candidate. Thus,

$$\begin{aligned} \Pr[v \text{ is disqualified} \mid v \text{ is a candidate}] &= \Pr\left[\bigcup_{w\in\Gamma(v),v<w} \text{w is a candidate}\right] \\ &\leq \sum_{w\in\Gamma(v),v<w} \Pr[\text{w is a candidate}] = \sum_{w\in\Gamma(v),v<w} \frac{1}{2d(w)} \\ &\leq d(v)\frac{1}{2d(v)} = \frac{1}{2}. \end{aligned}$$

□

With that we prove Lemma 2.

Proof of Lemma 2. Let C_v be a boolean random variable taking value 1 when v is chosen to be a candidate. Let v be a good vertex. We want a lower bound on the probability that v or one of its neighbors is a candidate:

$$\begin{aligned} \Pr[\bigcup_{w\in\{v\}\cup\Gamma(v)} C_w] &\geq \Pr[\bigcup_{w\in\Gamma(v):w<v} C_w] \\ &= 1-\Pr[\bigcap_{w\in\Gamma(v):w<v} \neg C_w] = 1-\prod_{w\in\Gamma(v):w<v}\left(1-\frac{1}{2d(w)}\right) \\ &\geq 1-\prod_{w\in\Gamma(v),w<v}\left(1-\frac{1}{2d(v)}\right) \geq 1-\left(1-\frac{1}{2d(v)}\right)^{\frac{d(v)}{3}} \geq 1-e^{-1/3}. \end{aligned}$$

The last step is correct because $\lim_{x\to\infty}\left(1-\frac{1}{x}\right)^x = e^{-1}$, and specifically $\left(1-\frac{1}{x}\right)^x \leq e^{-1}$. □

This completes the analysis of the probabilistic algorithm.

1.1.1 A Derandomized Algorithm

We now derandomize the probabilistic algorithm. In the randomized algorithm every node decides whether it is a candidate or not, by tossing an independent random string. The main idea behind the derandomization is that we can replace the independent random coins with correlated ones, and in fact we can use a distribution with such a small support such that we can turn it into a deterministic algorithm by trying (in parallel) all the possibile elemtns in its support.

Specifically, suppose the graph has n vertices and the degree of vertex i is d_i. We will use a distribution over $[d_1] \times \ldots [d_n]$ that has the following properties: Let X_i dentoe the marginal distribution on the i'th component, and (X_i, X_j) the marginal distribution on the u'th and j'th components together. We say that $X = (X_1, \ldots, X_n)$ is *pair-wise* independent if

- Each X_i is distributed uniformly over $[2d_i]$.
- $\forall i \neq j$, (X_i, X_j) is distributed uniformly over $[2d_i] \times [2d_j]$.

We remark, that in truth, we will not be able to build such a distribution X, and instead we will only approximate such a distribution. For the rest of the lecture, we assume we can construct such an exact distribution, and in the HW you are asked to approximate such a distribution and show that the approximation suffices for the derandomization.

Our goal is to find such a pair-wise independent distribution with a small support. Having such a distribution we do the following:

Algorithm 1 Derandomized MIS algorithm

```
In each step, for each element $x = (x_1, \dots, x_n)$ in the support of $X = (X_1, \dots, X_n)$, do in parallel:
    1. Run the original algorithm using $x_i$ as the random string tossed by vertex $i$, i.e., vertex $v$ is a candidate iff $x_v = 0$ (which happens with probability $\frac{1}{d_v}$).
    2. A run is considered good if at least $\frac{|E|}{24}$ edges were removed. Search, in parallel, for a good run and if there is one use the lexicographically first such run.
Repeat the process until the graph is empty.
```

The reader is encouraged to verify that each such step can be implemented in parallel logarithmic time.

We now turn to correctness. Lemma 1 has nothing to do with the randomness and so still holds. The proof of Claim 3 requires only pairwise independence (check!) and so still holds. The proof of Lemma 2 requires full independence, so we need to give it an alternative proof (with a worse constant) that uses only pair-wise independence. For that we prove the following claim:

Lemma 4. *Let* $Y_1, \dots, Y_m$ *be pair-wise independent boolean random variables, and let* $p_i = \Pr[Y_i = 1]$*. Then* $\Pr\left[\bigcup_{i=1}^m Y_i = 1\right] \ge \frac{1}{2}\min\{1, \sum_i p_i\}$.

Of Lemma 2. W.l.o.g. assume that $p_1 \ge p_2 \ge \cdots \ge p_m$. Let k be the first index for which $\sum_{i=1}^k P_i \ge 1$. If there's no such k let $k = m$. Denote $\alpha = \sum_{i=1}^k p_i$. Then

$$\Pr\left[\bigcup_{i=1}^m Y_i = 1\right] \ \ge \ \Pr\left[\bigcup_{i=1}^k Y_i = 1\right] \ge \sum_{i=1}^k \Pr[Y_i = 1] - \sum_{1 \le i < j \le k} \Pr[Y_i = Y_j = 1] \ge \sum_{i=1}^k p_i - \sum_{1 \le i < j \le k} p_i p_j$$

Now the minimum of $\sum_{1 \le i < j \le k} p_i p_j$ under the constraint that $\alpha = \sum_{i=1}^k p_i$ is achieved when $p_1 = \cdots = p_k = \frac{\alpha}{k}$ and so $\Pr\left[\bigcup_{i=1}^m Y_i = 1\right] \ge \alpha - \binom{k}{2}\left(\frac{\alpha}{k}\right)^2 = \alpha\left(1 - \frac{\alpha(k-1)}{2k}\right)$.

We now divide to cases. If $\alpha \ge 1$ then $\alpha = \sum_{i=1}^k p_i = \sum_{i=1}^{k-1} p_i + p_k$ and $\sum_{i=1}^{k-1} p_i < 1$. Thus, $\alpha \le 1 + p_k$. Also, $p_k \le \frac{1}{k-1}$ (because $p_1 \ge \ldots \ge p_{k-1}$ and their sum is at most one), hence,

$$\alpha\left(1 - \frac{\alpha(k-1)}{2k}\right) \ge \left(1 - (1 + \frac{1}{k-1}) \cdot \frac{(k-1)}{2k}\right) = \left(1 - \frac{k-1}{2k} - \frac{1}{2k}\right) = \left(1 - \frac{k}{2k}\right) = \frac{1}{2}$$

On the other hand, if $\alpha \leq 1$,

$$\alpha\Big(1 - \frac{\alpha(k-1)}{2k}\Big) > \alpha\Big(1 - \frac{1}{2}\cdot\frac{(k-1)}{k}\Big) > \frac{\alpha}{2} = \frac{\sum_{i=1}^{k} p_i}{2}.$$

□

Thus, if we let $Y_i = 1$ iff the i'th element of $v \cup \Gamma(v)$ is a candidate, then we see that for a good vertex v the probability that v or one of its neighbors is a candidate is $\Pr\left[\bigcup_{i=1}^{m} Y_i = 1\right] \geq \frac{1}{2}\min\{1, \sum_i p_i\} \geq \frac{1}{2}\min\left\{1, \sum_{w\in\Gamma(v), w<v} \frac{1}{2d_w}\right\} \geq \min\left\{1, \frac{d_v}{3}\frac{1}{2d_v}\right\}$ and is a positive constant. Thus, a variant of Lemma 2 is true even when the random variables X_i are only pair-wise independent.

Having that, we see that a constant fraction of the elements in the support of X eliminate a constant fraction of the edges. In particular, one such element succeeds. Hence, we will proceed to the next stage.

1.2 Constructing a pair-wise independent distribution

We now construct a pairwise independent distribution over n variables $(X_1, \ldots, X_n)$. For simplicity we assume all the variables are distributed over a domain $[n]$ of the same size, and in the HW you are asked to adjust the construction to what is actually needed.

Let $\mathbb{F}$ be a field with n elements and assume n is a power of 2. The elements in the support of the distribution are indexed by $(a,b) \in \mathbb{F}^2$. For every $a, b \in \mathbb{F}$ define $X_i = a + i \cdot b$. The support size of X is thus just $|\mathbb{F}|^2 = n^2$. For every $i \neq j$ and $\alpha, \beta \in \mathbb{F}$ we have:

$$\begin{aligned}\Pr[X_i = \alpha \wedge X_j = \beta] &= \Pr_{a,b\in\mathbb{F}}[a + ib = \alpha \wedge a + jb = \beta] = \Pr_{a,b\in\mathbb{F}}\left[\begin{pmatrix}1 & i\\ 1 & j\end{pmatrix}\begin{pmatrix}a\\ b\end{pmatrix} = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}\right]\\ &= \Pr_{a,b\in\mathbb{F}}\left[\begin{pmatrix}1 & i\\ 1 & j\end{pmatrix}^{-1}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}a\\ b\end{pmatrix}\right] = \frac{1}{|\mathbb{F}|^2} = \frac{1}{n^2},\end{aligned}$$

where we have used the fact that $\begin{pmatrix}1 & i\\ 1 & j\end{pmatrix}$ has a non-zero determinant (because $i \neq j$) and is invertible, and that $\begin{pmatrix}1 & i\\ 1 & j\end{pmatrix}^{-1}\begin{pmatrix}\alpha\\ \beta\end{pmatrix}$ is a constant vector, and therefore the probability (over $a, b \in \mathbb{F}$) that $\begin{pmatrix}a\\ b\end{pmatrix}$ equals that vector is exactly $\frac{1}{|F|}^2$. Thus the distribution is pairwise independent.

Note that pairwise independence is the best we can hope from this distribution. If it was 4-wise independent, the first 4 variables were independent. Each one has n possible values, so the support of any 4 random variables alone would be n^4. But the support of the entire distribution is n^2!

1.3 2-Universal Family of Hash Functions

Definition 5. *We say a family $H \subseteq \{h : A \to B\}$ is 2UFOHF (read: 2 universal family of hash functions) if for every $a_1 \neq a_2 \in A$, $b_1, b_2 \in B$:*

$$\Pr_{h \in H}[h(a_1) = b1 \wedge h(a_2) = b_2] = \frac{1}{|B|^2}].$$

This concept is almost identical to pairwise independence, where we sample $h \in H$ and set $X_i = h(a_i)$ for each $a_i \in A$. The random variables $X_1, \ldots, X_|A|$ take values in B and are pairwise independent.

2 Primality Testing

The ancient problem of determining whether a number is prime is given as input a number n decide if it is prime. Notice that the input length is the number of bits needed to represent n, i.e., $\log n$.

Clearly the problem is in *co*NP, but it was also shown to be in NP [1]. There are several algorithms putting it in *BPP*. We are now going to see a deterministic polynomial time algorithm for it, putting it in *P* [2].

2.1 Random Algorithms

There are several probabilistic algorithms for the problem that rely on the following property of primes numbers : n is prime implies that for any a, $a^n = a \mod n$. We remark, however, that this is not a characterization of prime numbers, and there are non-prime numbers that have this property.

A naive attempt is the following randomized algorithm for Primality Testing:

Algorithm 2 Algorithm for PT

```
1. Sample $a \in \mathbb{F}_n^*$ uniformly.
2. If $gcd(a, n) \neq 1$, return NO.
3. If $a^n = a \mod n$ return YES, else return NO.
```

If n is prime, we'll always return YES.However, there are non-prime integers n that will always pass the test. The probability across all possible n is small, but we want a *worst-case* algorithm, and therefore that algorithm has to be good for all possible inputs n (and for every input n, the algorithm should have a small error probability, over the random coins of the algorithm).

2.2 Characterising primes with a polynomial identity

The identity $a^n = a \mod n$ over $\mathbb{Z}$ is true for primes n, but does not characterize primes. Now, instead of working over $\mathbb{Z}$, we will work over the polynomial ring $\mathbb{Z}_n[x]$, the ring of all polynomials with degreeat most n over $\mathbb{Z}$, and in return we will get a characterization. Working with polynomials might look more complicated than working over the integers, but this is often not the case. In fact, quite the contrary is true. For example, polynomials (as integers) have unique factorization to irreducible polynomials. The problem of polynomial factorization can be done in polynomial time (sometimes, probabilistic polynomial time and sometimes deterministic polynomial time, depending on the field over which we work). In contrast, integer factorization is believed to be hard, and many cryptographic algorithms rely on the assumption that (at the very least) factorization is not in BPP. We claim:

Lemma 6. *Suppose $a, n \in \mathbb{Z}$ and $gcd(a,n) = 1$. Then, n is prime iff $(x+a)^n = x^n + a$, where the equality is over $\mathbb{Z}_n[x]$.*

Proof.

- Suppose n is prime. Then, $(x+a)^n = \sum_{i=0}^{n} \binom{n}{i} x^i a^{n-i}$. Clearly $\binom{n}{0} = \binom{n}{n} = 1$. Furthermore, for all other i,

  $$\binom{n}{i} = \frac{n(n-1)\dots(n-i+1)}{1 \cdot 2 \cdot \dots \cdot i}$$

  The prime n divides the nominator once, and does not divide the denominator, hence $\binom{n}{i} \mod n = 0$ and we are done (and for this part a can be any integer, not necessarily co=prime with n).

- Suppose n is not prime and $a \in \mathbb{Z}$ s.t. $gcd(a,n) = 1$. Then n has a prime factor p and an integer k such that p^k divides n but p^{k+1} does not. Consider the monomial with coefficient

  $$\binom{n}{p} = \frac{n(n-1)\dots(n-p+1)}{1 \cdot 2 \cdot \dots \cdot p}.$$

  p^k divides n but p doesn't divide $(n-1), \dots, (n-p+1)$, meaning $p^k | n(n-1)\dots(n-p+1)$ but p^{k+1} does not. Also, p divides p but not $1, \dots, p-1$, meaning $p | 1 \cdot 2 \cdot \dots \cdot p$ but p^2 does not. Therefore, $p^{k-1} | \binom{n}{p}$, but $p^k \nmid \binom{n}{p}$. This means $n \nmid \binom{n}{p}$, so the monomial does not zero out.

□

We could have turned the above characterization to a deterministic test: choose $a \in \mathbb{Z}$, say $a = 2$. If $gcd(a,n) > 1$, n is not prime. Otherwise check whether the polynomial $(x+a)^n$ is $x^n + a$ over $\mathbb{Z}_n[x]$. If it does, n is prime. Otherwise it does not. We can compute $(x+a)^n$ with fast exponentiation, and this takes $O(\log n)$ operations. However, the objects we deal with are elements in $\mathbb{Z}_n[x]$ of degree up to n, and so the *representation size* of such an element is $O(n)$ (to keep the n possible coefficients) and so this naive algorithm also runs in exponential time. In order to keep the objects we work with small enough, we choose a random irreducible univariate polynomial φ of degree $O(\log n)$ and check whether $(x+1)^n = x^n + 1 \mod \varphi$.

2.3 The AKS algorithm

Set $\varphi(x) = x^r - 1$, where r is an integer such that $ord_r(n) > \log^2 n$ ($ord_r(n)$ - the multiplicative order of n modulo r, is the first integer k such that $n^k = 1 \mod r$). We will see that always $r \leq \log^{10} n$.

Algorithm 3 AKS

```
1. Check whether $n$ is a perfect power. If so, return NO.
2. Find $r \leq \log^{10} n$ s.t. $ord_r(n) \geq \log^2 n$, by simply trying all small $r$.
3. For all $a = 1, \ldots, \log^{10} n$, check whether $(x+a)^n = x^n + a$ over $\mathbb{Z}_n[x]$ (mod $x^r - 1$).
4. If one of the tests failed - return NO. Else, return YES.
```

Step 1 of the algorithm might seem difficult, bet if $n = b^k$, then $k \leq \log n$ (because $n = 2^{\log n}$). So we can just iterate over all possible k's. Make sure you see the algorithm runs in deterministic polynomial time. We will prove correctness in the next lecture.

References

[1] Pratt, V. Every Prime Has a Succinct Certificate *SIAM Journal on Computing*, pages 214-220, 1975.

[2] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of mathematics*, pages 781–793, 2004.