Abstract. The problem of polynomial interpolation is to reconstruct a polynomial based on its evaluations on a set of inputs $I$. We consider the problem over $\mathbb{Z}_m$ when $m$ is composite. We ask the question: Given $I \subseteq \mathbb{Z}_m$, how many evaluations of a polynomial at points in $I$ are required to compute its value at every point in $I$? Surprisingly for composite $m$, this number can vary exponentially between $\log |I|$ and $|I|$ in contrast to the prime case where $|I|$ evaluations are necessary. While we show this minimization problem to be NP-hard, we give an efficient algorithm of query complexity within a factor $t$ of the optimum where $t$ is the number of prime factors of $m$. We use our interpolation algorithm to design algorithms for zero-testing and distributional learning of polynomials over $\mathbb{Z}_m$. In some cases, we get an exponential improvement over known algorithms in query complexity and running time.

Our main technical contribution is the notion of an interpolating set for $I$ which is a subset $S$ of $I$ such that a polynomial which is 0 over $S$ must be 0 at every point in $I$. Any interpolation algorithm needs to query an interpolating set for $I$. Our query-efficient algorithms are obtained by constructing interpolating sets whose size is close to optimal.

Key words. Polynomials, Composites, Interpolation.

AMS subject classifications. 68Q25, 12Y05

1. Introduction. The problem of polynomial interpolation is to reconstruct a polynomial from its evaluations. This is a fundamental algorithmic question in algebra with numerous applications. The problem is especially well studied when the polynomial is over a field such as $\mathbb{R}$ or $\mathbb{Z}_p$ dating back to Newton and Lagrange. Relatively less is known about interpolation over rings which contain zero divisors, in particular over $\mathbb{Z}_m$ with $m$ composite. The zero-testing problem is a special case of the interpolation problem where we want to know if the polynomial is 0 everywhere. In this paper we study the problem of learning a univariate polynomial in $\mathbb{Z}_m[X]$ based on its evaluations at a set $I \subseteq \mathbb{Z}_m$. We ask the question: Given $I \subseteq \mathbb{Z}_m$, how many evaluations of a polynomial at points in $I$ are required to compute its value at every point in $I$? Throughout, we will consider a polynomial as a function rather than a formal sum and our aim will be to correctly predict its values at every point in $I$.

Polynomials and specifically the problem of interpolation over $\mathbb{Z}_m$ are well studied in mathematics [18, 31]. Dueball [31] shows that when $I = \mathbb{Z}_m$, there is a subset $S$ whose size can lie between $\log m$ and $m$ such that the evaluations at $S$ are sufficient for interpolation. However, this result leaves open the question of whether a similar statement holds for subsets of $\mathbb{Z}_m$. Polynomials over $\mathbb{Z}_m$ have many applications in computer science. They are used in algorithms for primality testing [1, 2], in the construction of explicit Ramsey graphs and extremal set systems [21, 24] and in circuit lower bounds [7, 36]. As we will elaborate shortly, many of these applications implicitly address questions related to polynomial interpolation. We hope that a better understanding of the interpolation problem will throw new light on these problems.

The polynomial interpolation problem over $\mathbb{Z}_m$ is very different from $\mathbb{Z}_p$ since it is no longer true that a degree $d$ polynomial has at most $d$ zeroes. For instance $X^k \equiv
0 mod $2^k$ has $2^{k-1}$ roots. This implies that even two polynomials of small degree can agree on a large fraction of points in $\mathbb{Z}_m$. Hence, unlike over $\mathbb{Z}_p$, one cannot interpolate even low degree polynomials from their evaluations at a few arbitrarily chosen points. On the other hand, not every function $f : \mathbb{Z}_m \rightarrow \mathbb{Z}_m$ is a polynomial, since functions defined by polynomials need to satisfy certain congruences. For instance let $m = pq$ and $x, y \in \mathbb{Z}_m$ such that $x \equiv y \pmod{p}$. Then $P(x) \equiv P(y) \pmod{p}$ for any polynomial $P(X) \in \mathbb{Z}_m[X]$. Thus the values of a polynomial at a point give some information about its values at other points. This raises the possibility of learning a polynomial by looking at its evaluations at only a few carefully chosen points.

We give an query-efficient algorithm to solve the following problem:

**Problem 1. Generalized Polynomial Interpolation:** Given $m$, a set $I \subseteq \mathbb{Z}_m$ and black-box access to the values of a polynomial $P(X) \in \mathbb{Z}_m[X]$ at points in $I$. Compute $P(X)$ and minimize the number of black-box queries.

The query complexity of our algorithm is within a factor $t$ of the optimum where $t$ is the number of prime divisors of $m$. One has to settle for approximation since we prove that the problem of minimizing the number of queries is NP-hard. Our main technical contribution is the notion of interpolating sets for $I$ which are subsets of $I$ such that a polynomial which is 0 over that subset must in fact be 0 at every point in $I$. We show that any interpolation algorithm needs to query an interpolating set for $I$. Our query-efficient algorithms are obtained by constructing interpolating sets whose size is close to optimal. We use our interpolation algorithm to design algorithms for zero-testing and distributional learning of polynomials over $\mathbb{Z}_m$. In some cases, we get an exponential improvement over known algorithms in query complexity and running time. We show some new results about the structure of polynomials over $\mathbb{Z}_m$, which may be useful in other applications.

### 1.1. History and Motivation

Given a commutative ring $R$, a function $f : R \rightarrow R$ which can be computed by a polynomial in $R[X]$ is called a polynomial function. Polynomial functions over various commutative rings are well studied in algebra [15, 31, 18]. The problem of characterizing polynomial functions over $\mathbb{Z}_m$ was first studied by Carlitz and Spira [35] (see also the book by Narkiewicz and references therein [31]). Kempner gave a canonical polynomial for every polynomial function over $\mathbb{Z}_m$ [30]. Dueball studied the problem of interpolation over $\mathbb{Z}_m$ [31]. He proved that one can solve the interpolation problem over $\mathbb{Z}_m$ with as few as $O(\log m)$ queries for some composites $m$. More precisely, he showed the following result:

Let $k(m)$ be the smallest integer such that $k(m)! \equiv 0 \pmod{m}$. Every polynomial function over $\mathbb{Z}_m$ can be learnt from its values at $\{0, \cdots, k(m) - 1\}$.

In the zero-testing problem, we are given an implicit representation of a polynomial $P$, either as a circuit or a black-box which returns the value $P(x)$ on query $x$. We wish to determine if $P$ is the 0 polynomial. The problem of zero-testing for polynomials over $\mathbb{Z}_m$ was studied by Agrawal and Biswas [1], motivated by primality testing. They gave a randomized algorithm for this problem. However, they view polynomials as formal sums rather than as functions and this is important for their application. Karpinski, van der Poorten and Shparlinski [28] give a black-box algorithm for zero-testing over $\mathbb{Z}_m$. However they require that all non-zero coefficients of the polynomial are relatively prime to $m$. Bshouty, Tamon and Wilson give a randomized algorithm for interpolation over $\mathbb{Z}_m$ [14]. However if the smallest prime dividing $m$ is $p$, they require the degree to be at most $\frac{\log m}{\log p}$. The results of [14, 28] hold for multivariate polynomials, but in the univariate case Dueball’s result is stronger.

Interpolation and zero-testing for polynomials over $\mathbb{Z}_p$ have been studied exten-
sively in computer science, motivated by applications in coding theory, proof checking
and several other areas (see for instance [20]). Most of these applications crucially use
the fact that a degree $d$ polynomial over $\mathbb{Z}_p$ has at most $d$ roots, which does not hold
for polynomials over $\mathbb{Z}_m$. Nevertheless, polynomials over $\mathbb{Z}_m$ have found surprising
applications in algorithms, combinatorics and complexity, which rely on the fact that
they behave differently from polynomials over $\mathbb{Z}_p$.

- **Primality and Factoring:** The primality testing algorithms of Agrawal
  and Biswas and the AKS algorithm reduce testing primality to testing a
  polynomial identity over $\mathbb{Z}_m[X]$ [1, 2]. They then devise algorithms to solve
  the problem of zero-testing over $\mathbb{Z}_m[X]$. Shamir [33] shows that the problem
  of factoring polynomials over $\mathbb{Z}_m[X]$ is as hard as integer factoring.

- **Boolean Function Complexity:** A frontier open problem in complexity
  theory is to show lower bounds for circuits with Mod-$m$ gates. Strong lower
  bounds are known when the circuit contains only Mod-$p$ gates for a single
  prime $p$[32, 34]. In contrast much less is known if Mod-$m$ gates are allowed
  with $m$ composite, or Mod-$p$ and Mod-$q$ gates for distinct primes [26, 16].
  Motivated by this problem, Barrington, Beigel and Rudich [7] studied repre-
  sentations of Boolean functions by polynomials over $\mathbb{Z}_m$. They proved that
  functions like OR can have low degree representations when $m$ is com-
  posite, unlike in the prime case. The problem of showing tight bounds for such
  representations over $\mathbb{Z}_m$ is wide open [4, 13, 36].

- **Extremal Set Theory:** A set system on $[n]$ is said to have restricted inter-
  sections modulo $m$ if there exists $L \subseteq \mathbb{Z}_m$ such that the pairwise intersections
  mod $m$ lie in $L$ but the set sizes lie outside it [5]. Polynomials over $\mathbb{Z}_m$ have
  been used to explicitly construct large set systems and also to prove upper
  bounds on their size. A surprising insight from this area is that when $m$ is
  a prime or a prime-power, the size of such sets systems is polynomial in the
  number of elements in the universe [5, 6, 21]. In contrast, when $m$ has two
  or more prime divisors, the size can be super-polynomial [24]. These results
  about set systems have important combinatorial applications [5].

- **Explicit Ramsey Constructions:** A Ramsey graph is a graph with no large
  cliques and independent sets. The problem of explicitly constructing good
  Ramsey graphs is an important open problem in combinatorics. Recently, the
  author [21] showed that the algebraic Ramsey graph constructions of Alon [3],
  Frankl-Wilson [17] and Grolmusz [24, 25] can be derived in a unified manner
  from low degree polynomials over $\mathbb{Z}_m$. Further, facts about interpolating
  sets over $\mathbb{Z}_p$ from this work are used in [21] in order to show that certain
  approaches (based on symmetric polynomials) cannot yield better Ramsey
  graphs.

The last three applications deal with whether certain functions can be computed
by low degree polynomials over $\mathbb{Z}_m$, hence they all implicitly address questions related
to polynomial interpolation.

1.2. Our Results.

**The Generalized Interpolation Problem.** Our main result is an efficient
algorithm to solve the generalized interpolation problem. We prove that minimizing
the number of queries is NP-hard, hence one can only hope to approximately minimize
the query complexity. Our algorithm has query complexity close to optimal.

**Theorem 1.** Let $t$ be the number of distinct prime factors of $m$. There is an
algorithm to solve the general interpolation problem over $\mathbb{Z}_m$, with query complexity
within a factor $t$ of the optimum.

In fact the guarantee is slightly stronger. When the algorithm terminates, it produces a factorization of $m$ into $t' \leq t$ relatively prime factors. The approximation factor is in fact bounded by $t'$. Thus input sets $I$ which force the algorithm to make several queries must also reveal the factorization of $m$. The algorithm first computes a set $S$ of queries to ask based on the input set $I$. Thus the set of queries is chosen non-adaptively. The size of $S$ is within a factor $t'$ of the optimal query complexity. This step takes time proportional to $|S| \cdot |I|$. Once the set $S$ is found, the polynomial can be computed with $|S|$ queries in time poly(log $m$, $|S|$).

While Dueball’s result gives an efficient algorithm for the case when $I = \mathbb{Z}_m$, it does not imply anything for the general interpolation problem. The naive approach for this problem would be to write a linear equation for each point in $I$. We can replace each equation $\sum_j a_{ij} x_{ij} = b_i \mod m$ with $\sum_j a_{ij} x_{ij} = b_i + y_i m$ where the $y_i$s are integer variables, and find integer solutions to the resulting system of equations. This has query complexity $|I|$, which can be exponentially larger than the complexity of our algorithm.

The generalized zero-testing problem is a special case of the interpolation problem, where we wish to know if some identity holds for every point in $I$. Theorem 1 implies a query-efficient algorithm for this problem. It improves on the algorithms of [14, 28] since there are no restrictions on the degree or coefficients of the polynomial. Our results are incomparable with those of [1], who view polynomials as formal sums.

**Learning under a Distribution.** We give the first efficient algorithms for learning polynomials over $\mathbb{Z}_m$ under a distribution. Here we are given evaluations of the polynomial at points which are drawn from some distribution and we are asked to learn the polynomial. See §5 for precise problem definitions.

**Theorem 2.** Polynomials in $\mathbb{Z}_m[X]$ are exactly learnable under the uniform distribution and PAC-learnable under an arbitrary distribution in polynomial time. The algorithm for the uniform distribution learns the polynomial exactly, but its running time is a random variable. These algorithms use the algorithm for the general interpolation problem as a subroutine. For distributional learning it is essential that our algorithm solves the general interpolation problem, where inputs come from some subset $I \subseteq \mathbb{Z}_m$ rather than all of $\mathbb{Z}_m$.

**Interpolating Sets.** The crux of our algorithm is the notion of an interpolating set which we introduce and study here. A set $S \subseteq I$ is an interpolating set for $I$ if knowing the values of any polynomial at $S$ fixes its value at every point in $I$. We show that the set of queries of an interpolation algorithm must correspond to an interpolating set for $I$, thus the problem of designing query-efficient algorithms reduces to finding small interpolating sets.

Let $k(I)$ denote the size of the smallest interpolating set for $I$. In general $k(I)$ can lie between log $|I|$ and $|I|$. However we prove that the problem of computing a minimum interpolating set for $I$ is NP-hard. We define a related quantity $\bar{k}(I)$, which is the smallest integer such that there is a degree $\bar{k}(I)$ monic polynomial $M(X) \in \mathbb{Z}_m[X]$ which is 0 over $I$. This quantity can be computed in polynomial time by solving a system of linear equations. We show that for $I \subseteq \mathbb{Z}_m$ where $m$ has $t$ prime divisors, the following relation holds:

$$\bar{k}(I) \leq k(I) \leq t \cdot \bar{k}(I)$$

Thus $\bar{k}(I)$ is a factor $t$ approximation to $k(I)$ where $t$ is the number of prime divisors.
of \( m \). This is where the approximation factor of \( t \) in the query complexity of our algorithm comes from.

We sketch the idea behind the algorithm for computing an interpolating set. For the prime-power case, we use a greedy algorithm. There is a natural metric on the points \( I \subseteq \mathbb{Z}_{p^a} \), namely \( p \)-adic distance. Our algorithm finds a set of points so that the sum of pairwise distances is maximized, this is done by picking a new point in a natural greedy manner. We show that this in fact gives an interpolating set. For the composite case, we essentially try and repeat this greedy approach. However, this approach might fail: firstly, we do not know the factorization of \( m \), and secondly distinct prime divisors \( p \) and \( q \) give different metrics on the set \( I \). However, we show that when it fails, one can get a factorization \( m = m_1 \cdot m_2 \) where \( (m_1, m_2) = 1 \). This allows us to use divide and conquer: we find interpolating sets modulo \( m_1 \) and \( m_2 \) independently and combine the result using the Chinese Remainder Theorem.

Interpolating sets over \( \mathbb{Z}_{p^a} \) have rich algebraic and combinatorial structure which we study in detail, these properties are also useful in analyzing our algorithm. In proving these properties, we make crucial use of the fact that the underlying space is in fact an ultrametric space (metrics where the following strengthening of the triangle inequality holds: \( d(x, y) \leq \max(d(x, z), d(y, z)) \)). Ultrametric spaces are well-studied in computer science [8, 9, 10]. We show that many algebraic properties of polynomials can be reinterpreted as geometric properties of ultrametric spaces. Further, the proof of these properties for general ultrametric spaces follows directly from the proof for polynomials over \( \mathbb{Z}_{p^a} \). We note that our notion of interpolating sets over \( \mathbb{Z}_{p^a} \) is closely related to Very Well Distributed and Well Ordered sequences that have been studied in mathematics [15].

1.3. Organization of this Paper. In the next section, we give some basic definitions and results about interpolation over \( \mathbb{Z}_m \). We study interpolating sets in §3. We present our algorithms for interpolation in §4 and learning algorithms in §5. We present other algebraic and combinatorial characterizations of interpolating sets in §6. In §7, we use these characterizations to establish some geometric properties of ultrametric spaces. An extended abstract of this paper appeared in SODA’06 [22].

2. Preliminaries. We will use \( X \) to denote a variable, and \( x \) for a constant. Let \((a, b)\) denote the greatest common divisor of \( a \) and \( b \).

Given \( x \in \mathbb{Z}_{p^a}, \ x \neq 0 \), let its \( p \)-adic valuation \( \text{val}_p(x) \) be the highest power of \( p \) which divides \( x \). Set \( \text{val}_p(0) = \infty \). We have the so-called ultrametric inequality which states:

\[
\text{val}_p(x + y) \geq \min\left(\text{val}_p(x), \text{val}_p(y)\right)
\]

The \( p \)-adic norm of \( x \) is defined as

\[
|x|_p = p^{-\text{val}_p(x)}.
\]

The \( p \)-adic norm satisfies the condition:

\[
|x + y|_p \leq \max(|x|_p, |y|_p).
\]

This induces a metric (the \( p \)-adic metric) on \( \mathbb{Z}_{p^a} \) given by \( d(x, y) = |x - y|_p \). This metric satisfies the following strong form of the triangle inequality:

\[
d(x, z) \leq \max(d(x, y), d(y, z)). \tag{2.1}
\]
Metrics which satisfy Equation 2.1 are known as ultrametrics.

We will also define valuations over \( \mathbb{Z}_m \) where \( m \) is not a prime power. Assume that \( p \) divides \( m \) and let \( p^\ell \) be the highest power of \( p \) dividing \( m \). For \( x \in \mathbb{Z}_m \), we define

\[
\text{val}_p(x) = \text{val}_p(x \mod p^\ell).
\]

All our results can be stated either in terms of \( p \)-adic valuations or norms. For our algorithmic results, it is more natural to work with valuations. For our combinatorial results, we will use \( p \)-adic norms, since it is easier to translate these results to general ultrametric spaces.

We start with some basic algebraic facts that will be useful to us.

**Proposition 3.** [27] Let \( a, b \in \mathbb{Z}_m \) and \( a \neq 0 \). The equation

\[
aX \equiv b \mod m
\]

has a solution in \( \mathbb{Z}_m \) iff \( (a, m) \mid b \). If this condition holds, there is a unique solution in the interval \([0, \ldots, m/(a,m) - 1]\).

**Proposition 4.** Let \( M(X) \) be a monic polynomial in \( \mathbb{Z}_m[X] \) of degree \( k \). Given \( P(X) \in \mathbb{Z}_m[X] \), we can divide it by \( M(X) \) and get a remainder of degree at most \( k - 1 \).

**Proof.** This is just Euclidean division. Let \( P(X) = \sum_{i \leq d} c_i X^i \) where \( d \geq k \). Since \( M(X) \) is monic, \( P(X) - c_d X^{d-k} M(X) \) has degree \( d - 1 \). Now repeat the same procedure till we are left with a polynomial of degree \( \leq k - 1 \).

**Proposition 5.** Let \( N_0(X), \ldots, N_k(X) \) be polynomials in \( \mathbb{Z}_m[X] \) where \( N_i(X) \) is a monic polynomial of degree \( i \). Every polynomial \( P(X) \) of degree at most \( k \) can be written as

\[
P(X) = \sum_{i=0}^{k} c_i N_i(X).
\]

Further if \( P(X) \) is a monic polynomial of degree \( k \), then \( c_k = 1 \).

**Proof.** The proof is by induction on \( k \). When \( k = 0 \), \( N_0(X) = 1 \) so there is nothing to prove. Assume the claim holds for \( k - 1 \). Let \( P(X) = \sum_{i \leq k} a_i X^i \). Since \( N_k(X) \) is monic, \( P(X) - a_k N_k(X) \) has degree \( k - 1 \), so we can apply induction to it. Note that the leading coefficient in the monomial basis and the \( \{N_i(X)\} \) basis is the same. This shows the second part of the claim.

We can use this to give a canonical form for polynomial functions over \( \mathbb{Z}_m \) due to Kempner [30]. Let \( N_0(X) = 1 \) and for \( j \geq 1 \), let

\[
N_j(X) = \prod_{i=0}^{j-1} (X - i).
\]

Let the elements \( \{0, \ldots, m - 1\} \) of \( \mathbb{Z}_m \) be endowed with the ordering \( 0 < 1 < \cdots < m - 1 \). Let \( k(m) \) be the smallest integer such that \( k(m)! \equiv 0 \mod m \).

**Lemma 6.** [30] Every polynomial function over \( \mathbb{Z}_m \) is computed by a unique polynomial of the form

\[
P(X) = \sum_{j=0}^{k(m)-1} c_j N_j(X) \quad 0 \leq c_j < \frac{m}{(m,j)!}.
\]
Proof. For any \( x \in \mathbb{Z} \),

\[
N_j(x) = \prod_{i=0}^{j-1} (x - i) = \binom{x}{j} j!
\]

Hence \( N_j(x) \) is divisible by \( j! \). So the polynomial \( \frac{m}{(m,j)!} N_j(X) \) is zero over \( \mathbb{Z}_m \). In particular, \( N_{k(m)}(X) \) is a degree \( k(m) \) monic polynomial which is 0 over \( \mathbb{Z}_m \).

Given an arbitrary polynomial \( Q(X) \), we first divide by \( N_{k(m)}(X) \) to get a polynomial \( Q'(X) \) of degree \( k(m) - 1 \). Since the polynomials \( N_j(X) \) is monic and of degree \( j \) for \( j \in \{0, \cdots, k(m) - 1\} \), we can write \( Q(X) \) as

\[
Q'(X) = \sum_{j=0}^{k(m)-1} c_j N_j(X)
\]

We can reduce this to the form of Equation 2.2 by subtracting an appropriate multiple of \( \frac{m}{(m,j)!} N_j(X) \) for \( j \leq k(m) \). Since we are only subtracting polynomials that are 0 over \( \mathbb{Z}_m \), the resulting polynomial computes the same function as the polynomial \( Q(X) \) that we started with.

To show that this representation is unique, take two polynomials

\[
P(X) = \sum_{j=0}^{k(m)-1} c_j N_j(X) \quad 0 \leq c_j < \frac{m}{(m,j)!}
\]

\[
Q(X) = \sum_{j=0}^{k(m)-1} d_j N_j(X) \quad 0 \leq d_j < \frac{m}{(m,j)!}
\]

Pick the smallest index \( j \) so that \( c_j \neq d_j \), and assume that \( c_j > d_j \). We claim that \( P(j) \neq Q(j) \mod m \). Since \( N_i(j) = 0 \) for \( i > j \) and \( c_i = d_i \) for \( i < j \), we have

\[
P(j) - Q(j) = (c_j - d_j)N_j(j) = (c_j - d_j)j! \equiv 0 \mod m
\]

since

\[
0 < c_j - d_j < \frac{m}{(m,j)!}.
\]

\[\square\]

An easy consequence is the following result of Dueball [31].

Corollary 7. [31] Every polynomial function over \( \mathbb{Z}_m \) can be interpolated from its evaluations at the points \( \{0, \cdots, k(m) - 1\} \).

Proof. Let

\[
P(X) = \sum_{j} c_j N_j(X) \quad 0 \leq c_j < \frac{m}{(m,j)!}.
\]

We let \( c_0 = P(0) \). Assuming we know \( c_0, \cdots, c_{j-1} \), we solve for \( c_j \) from the equation

\[
c_j j! \equiv P(j) - \sum_{i<j} c_i N_i(j) \mod m
\]
The coefficients of \( P(X) \) in the canonical form are a solution to this equation. Further, the solution must be unique, since the canonical form is unique. \( \Box \)

The following estimates for \( k(m) \) show that the number of queries can be significantly smaller than \( m \) if \( m \) is smooth.

**Lemma 8.** For prime powers,

\[
p(a - 1) + 1 \leq k(p^a) \leq pa.
\]

If \( m = \prod_j p_j^{a_j} \), then

\[
k(m) = \max_j k(p_j^{a_j}).
\]

**Proof.** Since \((pa)! \equiv 0 \mod p^a\), \( k(p^a) \leq pa \). Let \( k = \sum_i k_ip^i \) be the base-\( p \) expansion of \( k \). Using a formula due to Legendre [23],

\[
\text{val}_p(k!) = \sum_i \left\lfloor \frac{k}{p^i} \right\rfloor = \frac{k - \sum_i k_i}{p - 1}
\]

(2.3)

Hence if \( k! \equiv 0 \mod p^a \), then

\[
\frac{k - \sum_i k_i}{p - 1} \geq a \Rightarrow k \geq (p - 1)a + \sum_i k_i \geq (p - 1)a + 1
\]

For \( m = \prod_j p_j^{a_j} \), by Chinese Remaindering, \( k! \equiv 0 \mod m \) is equivalent to \( k! \equiv 0 \mod p_j^{a_j} \) for all \( j \). So \( k(m) = \max_j k(p_j^{a_j}) \). \( \Box \)

Next we show that the problem of computing \( k(m) \) from \( m \) is as hard as factoring \( m \).

**Lemma 9.** The problem of computing \( k(m) \) given \( m \) as input is equivalent to factoring \( m \).

**Proof.** One can check in polynomial time if \( m \) is a prime power [11], so assume it is not. We will show that \((k(m), m)\) gives a non-trivial factor of \( m \). Note \( k(m) = \max_i k(p_i^{a_i}) \). Assume this maximum is attained for the prime \( p_i \). Note that \( k(p_i^{a_i}) \equiv 0 \mod p_i \), else

\[
\text{val}_p(k(p_i^{a_i})!) = \text{val}_p((k(p_i^{a_i}) - 1)!)\]

which contradicts the definition of \( k(p_i^{a_i}) \). Then \( k(m) = k(p_i^{a_i}) \) is divisible by \( p_i \). Further

\[
k(p_i^{a_i}) \leq p_ia_i \leq p_i^{a_i} < m
\]

since \( m \) is not a prime power. Hence

\[
p_i \leq (k(m), m) < m.
\]

Thus we get a non-trivial factor of \( m \). If \((k(m), m)\) is not a prime power, we can repeat this procedure till we get a prime power divisor of \( m \). \( \Box \)
3. Interpolating Sets. We say that a polynomial \( P(X) \) is 0 over set \( S \) if it evaluates to 0 at every point in \( S \).

**Definition 1.** Given \( I \subseteq \mathbb{Z}_m \), \( S \subseteq I \) is an interpolating set for \( I \) if every polynomial which is 0 over \( S \) is 0 over \( I \). Let \( k(I) \) denote the size of the smallest interpolating set for \( I \).

Note that \( I \) itself is trivially an interpolating set. However in general there can be interpolating sets which are significantly smaller than \( I \). Note that two polynomials \( P(X) \) and \( Q(X) \) that agree at \( S \) must in fact agree at every point in \( I \), by considering \( P(X) - Q(X) \). Thus the values of a polynomial over \( I \) are uniquely determined by the values at points in an interpolating set. The next lemma shows that the minimum number of queries to interpolate a polynomial over \( I \) is \( k(I) \).

**Lemma 10.** The set of black-box queries of any interpolation algorithm is an interpolating set for \( I \).

**Proof.** Assume that the set \( S \) of queries to the black-box is not an interpolating set. Then there exists polynomial \( Q(X) \in \mathbb{Z}_m[X] \) such that \( Q(x) \) is 0 at all \( x \in S \) but non-zero at some point \( y \in I \). Hence the algorithm cannot distinguish between polynomials \( P(X) \) and \( P(X) + Q(X) \) which agree on \( S \) but are different at \( y \in I \).

Note that this bound holds even if the algorithm chooses its queries adaptively.

**Definition 2.** Let \( k(I) \) be the smallest integer such that there is a degree \( k(I) \) monic polynomial \( M(X) \in \mathbb{Z}_m[X] \) which is 0 over \( I \).

If \( S \) is an interpolating set of size \( k(I) \), then the polynomial \( \prod_{\alpha \in S} (X - \alpha) \) is a monic polynomial of degree \( k(I) \). It is zero over \( S \) and hence over \( I \). Hence

\[
\hat{k}(I) \leq k(I).
\] (3.1)

This lets us prove lower bounds on \( k(I) \) by showing that any polynomial that is 0 over \( I \) must have certain degree.

**Example 1.** For \( I \subseteq \mathbb{Z}_{p_m} \), \( \hat{k}(I) = k(I) = |I| \). Since \( \mathbb{Z}_{p_m} \) is a field, the smallest degree monic polynomial which is 0 over \( I \) is \( M(X) = \prod_{\alpha \in I} (X - \alpha) \), hence \( k(I) = |I| \).

**Example 2.** For \( I = \mathbb{Z}_m \), \( k(I) = \hat{k}(I) = k(m) \). By Corollary 7, the set \( S = \{0, \cdots, k(m) - 1 \} \) is an interpolating set so \( k(I) \leq k(m) \). To show \( k(I) \geq k(m) \), assume that \( M(X) \) is a monic polynomial of degree \( d < k(m) \). Writing it in the canonical form, we get \( M(X) = \sum_{i \leq d} c_i N_i(X) \) where \( c_d = 1 \). So \( M(X) \) cannot be zero over \( \mathbb{Z}_m \) by Lemma 6.

One can use the Chinese Remainder Theorem to relate the problem of computing \( k(I) \) and \( \hat{k}(I) \) for \( I \subseteq \mathbb{Z}_m \) for composite \( m \) to the prime power case. First we need to introduce some notation. Let \( m = \prod_{j=1}^t p_j^{n_j} \). Given a set \( L \subseteq \mathbb{Z}_m \) we define the projection \( L_j \) of \( L \) mod \( p_j^{n_j} \) as

\[
L_j = \{ y \in \mathbb{Z}_{p_j^{n_j}} | \exists x \in L, x \equiv y \mod p_j^{n_j} \}.
\]

For a polynomial \( P(X) \in \mathbb{Z}_m[X] \), we define \( P_j(X) \in \mathbb{Z}_{p_j^{n_j}}[X] \) to be its projection modulo \( p_j^{n_j} \) obtained by taking each coefficient of \( P(X) \) modulo \( p_j^{n_j} \). Conversely, given polynomials \( P_j(X) \in \mathbb{Z}_{p_j^{n_j}}[X] \) we can combine the coefficients using the Chinese Remainder Theorem to get a unique polynomial \( P(X) \in \mathbb{Z}_m[X] \) whose projections are the polynomials \( P_j(X) \). We call \( P(X) \) the lift of the \( P_j(X) \)s.

**Lemma 11.** Let \( I \subseteq \mathbb{Z}_m \). Then

\[
\hat{k}(I) = \max_j \hat{k}(I_j).
\] (3.2)
Proof. It is easy to show using the Chinese Remainder Theorem that a polynomial $P(X)$ is zero over $I$ iff $P_j(X)$ is zero over $I_j$ for all $j$. Let $M(X)$ be a monic polynomial which is zero over $I \subseteq \mathbb{Z}_m$. Then by the Chinese Remainder Theorem, $M_j(X)$ is a monic polynomial which is zero over $I_j \subseteq \mathbb{Z}_{p_j^s}$. Hence $k(I) \geq k(I_j)$ for every $j$.

Conversely let $\max_j k(I_j) = d$. For each $j$, there is a monic polynomial $M_j(X)$ of degree $d_j \leq d$ which is zero over $I_j$. The polynomial

$$M'_j(X) = X^{d-d_j}M_j(X)$$

is a degree $d$ monic polynomial which is zero over $I_j$. Let $M'(X) \in \mathbb{Z}_m[X]$ be the lift of the $M'_j(X)$s. It follows that $M'(X)$ is a monic polynomial of degree $d$ and it is zero over $I$.

Lemma 12. Let $I \subseteq \mathbb{Z}_m$. The set $S$ is an interpolating set for $I$ iff $S_j$ is an interpolating set for $I_j$ for every $j$.

Proof. Assume that $S_j$ is an interpolating set for $I_j$ for every $j$, but $S$ is not an interpolating set for $I$. Then there is a polynomial $Q(X) \in \mathbb{Z}_m[X]$ such that at every point $x \in S, Q(x) \equiv 0 \mod m$, but for some $y \in I$, $Q(y) \not\equiv 0 \mod m$. But then $Q(y) \not\equiv 0 \mod p_j^{s_j}$ for some $j$. Consider the polynomial $Q_j(X)$. Since $Q(X)$ is zero over $S$, $Q_j(X)$ is zero over $S_j$. However there exists $y' \equiv y \mod p_j^{s_j}$ in $I_j$ such that $Q_j(y') \not\equiv 0 \mod p_j^{s_j}$. This contradicts the assumption that $S_j$ is an interpolating set for $I_j$.

In the other direction, assume that $S$ is an interpolating set for $I$ but $S_j$ is not an interpolating set for $I_j$. Then there is a polynomial $Q_j(X) \in \mathbb{Z}_{p_j^{s_j}}[X]$ such that for every $x \in S_j, Q(x) \equiv 0 \mod p_j^{s_j}$, but there exists $y \in I_j$ such that $Q(y) \not\equiv 0 \mod p_j^{s_j}$. Take $Q_i(X) = 0$ for $i \neq j$ and set $Q(X) \in \mathbb{Z}_m[X]$ to be the lift of the $Q_i(X)$s. Then $Q(X)$ is zero over $S$ since it is zero over every $S_j$. But it is not zero at some point in $I$ since $Q_j$ is not zero over $I_j$. This contradicts the assumption that $S$ is an interpolating set.

Corollary 13. Let $I \subseteq \mathbb{Z}_m$. Then

$$\max_j k(I_j) \leq k(I) \leq \sum_{j=1}^t k(I_j). \quad (3.3)$$

Proof. The bound $k(I) \geq k(I_j)$ follows trivially since $|S| \geq |S_j| \geq k(I_j)$. To prove the other direction, let $S_j$ be a minimum interpolating set for $I_j$. For $y \in S_j$, there exists a preimage $x \in I$ such that $x \equiv y \mod p_j^{s_j}$. Define $S'_j \subseteq I$ by choosing one preimage for each $y$. Set $T = \cup_j S'_j$. $T$ is an interpolating set for $I$ since $S_j \subseteq T_j$ is an interpolating set for $I_j$. Also $|T| \leq \sum_{j=1}^t k(I_j)$. 

Example 3. We give an example where $k(I) < k(I)$ and the upper bound in Equation $3.3$ is (near) tight. Let $m = p_1p_2$, and let

$$I = \{a p_1 | 1 \leq a \leq p_2 - 1\} \cup \{b p_2 | 1 \leq b \leq p_1 - 1\}$$

It is easy to see that $k(I_1) = k(I_2) = p_1$, $k(I_2) = k(I_2) = p_2$ hence $k(I) = \max(p_1, p_2)$. On the other hand, the only interpolating set for $I$ is $I$ itself. Each point of the form $ap_1$ must be included since it is the only point in its congruence class modulo $p_2$. Similarly, every point $bp_2$ must be included. Thus $k(I) = p_1 + p_2 - 2$.

The following extension of the above lemmas can be proved similarly using the Chinese Remainder Theorem.
Corollary 14. Let \( m = \prod_{j=1}^{t'} m_j \) and \((m_i, m_j) = 1\). Let \( I_j \) denote the projection of \( I \) modulo \( m_j \).

\[
\tilde{k}(I) = \max_j \tilde{k}(I_j) \\
\max_j k(I_j) \leq k(I) \leq \sum_j k(I_j).
\]

Theorem 15. The problem of computing \( k(I) \) given \( I \) and \( m \) as input is NP-hard.

Proof. Consider the following decision problem:

Problem 2. \textsc{Min-Interpolating-Set}: Given \( m \) and \( I \subseteq \mathbb{Z}_m \), is \( k(I) \leq n \)?

We prove this problem is NP-hard by reduction from 3D-matching [19].

Problem 3. 3-Dimensional Matching: Given sets \( U, V, W \) of size \( n \) and a set of edges \( E \subseteq U \times V \times W \), is there a subset of \( E \) of size \( n \) that covers all the vertices?

Take \( p_1, p_2, p_3 \) to be 3 distinct primes greater than \( n \). Let \( m = p_1 p_2 p_3 \). For each triple \((u_i, v_j, w_k) \in E\) where \( x \equiv i \mod p_1, x \equiv j \mod p_2, x \equiv k \mod p_3 \). We claim that there is a matching of size \( n \) iff the set \( I \) has an interpolating set of size \( n \). We may assume that every vertex occurs in some edge, hence \( |I_1| = |I_2| = |I_3| = n \). Thus \( S \) is an interpolating set for \( I \) iff \( S_j = I_j \) for \( 1 \leq j \leq 3 \). Thus an interpolating set corresponds to a set of edges that cover every vertex. If there is an interpolating set of size \( n \), then there is a cover of size \( n \) and vice versa.

In fact, it is possible to show that the \textsc{Min-Interpolating-Set} problem is NP-complete, we skip the proof. \( \square \)

In Theorem 17, we will show that for \( I \subseteq \mathbb{Z}_{p^a} \), \( \tilde{k}(I) = k(I) \). Combining this with Equations 3.2 and 3.3

\[
\tilde{k}(I) \leq k(I) \leq tk(I).
\] (3.4)

Thus \( \tilde{k}(I) \) is a factor \( t \) approximation to \( k(I) \) where \( t \) is the number of prime divisors of \( m \).


4.1. The Prime Power Case. We give an algorithm to solve the polynomial interpolation problem over \( \mathbb{Z}_{p^a} \) using exactly \( k(I) \) queries. We first give a (greedy) algorithm to find a minimum interpolating set.

We start by picking an arbitrary element in \( I \). Suppose that we have chosen \( \{\alpha_0, \cdots, \alpha_{i-1}\} \) so far. If the polynomial \( N_S^I(X) = \prod_{j \in S}(X - \alpha_j) \) is 0 over \( I \) we stop. Else we choose the next element \( \alpha_i \in I \) so that \( \text{val}_p(N_S^I(\alpha_i)) \) is minimized.
Algorithm 1. IntSet(I, p^a)

Input: Set I ⊆ \mathbb{Z}_p^a.

Output: Interpolating set S for I.

Pick \( \alpha_0 \in I \) arbitrarily. Set \( S = \{ \alpha_0 \}, i = 1 \).

Repeat

Let \( N^S_i(X) = \prod_{j<i} (X - \alpha_j) \).

If \( N^S_i(x) \) is zero for all \( x \in I \),

Output \( S = \{ \alpha_0, \ldots, \alpha_{i-1} \} \). Stop.

Else

Find \( x \in I \) that minimizes \( \text{val}_p(N^S_i(x)) \).

Set \( \alpha_i = x, \ i = i + 1 \).

Assume that the algorithm outputs a set \( S = \{ \alpha_0, \ldots, \alpha_{k-1} \} \) of size \( k \) and let \( e_i = \text{val}_p(N^S_i(\alpha_i)) \).

Lemma 16. Every polynomial function over \( I \) is computed by a unique polynomial of the form

\[
P(X) = \sum_{j=0}^{k-1} c_j N^S_j(X) \quad 0 \leq c_j < p^{a-e_j}
\]  

(4.1)

Proof. The Proof is similar to that of Lemma 6. Given any polynomial \( Q(X) \), we give an algorithmic procedure to construct \( P(X) \) with the above form that agrees with \( Q(X) \) on \( I \). By the termination condition, the polynomial \( N^S_k(X) = \prod_{j<k} (X - \alpha_j) \) is identically zero over \( I \). Dividing \( Q(X) \) by \( N^S_k(X) \) and taking the remainder, we get \( Q'(X) \) of degree \( k-1 \) that computes the same function on \( I \). Let us set \( N^S_0(X) = 1 \).

Since the polynomials \( N^S_j(X) \) is monic and of degree \( j \) for \( j \in \{0, \ldots, k-1\} \), we can write any polynomial of degree at most \( k-1 \) as a linear combination of these polynomials. Hence we have

\[
Q'(X) = \sum_{j<k} c_j N^S_j(X)
\]

Note that by our choice of \( \alpha_j \),

\[
e_j = \text{val}_p(N^S_j(\alpha_j)) \leq \text{val}_p(N^S_j(x)) \quad \text{for } x \in I.
\]

So the polynomials \( p^{a-e_j} N^S_j(X) \) are 0 over \( I \). So by subtracting appropriate multiples of these polynomials from \( Q'(X) \) we can get a polynomial \( P(X) \) where \( 0 \leq c_j < p^{a-e_j} \) that computes the same function as \( Q(X) \).

To show uniqueness of this representation, consider two polynomials

\[
P(X) = \sum_{j=0}^{k-1} c_j N^S_j(X) \quad 0 \leq c_j < p^{a-e_j}
\]

\[
Q(X) = \sum_{j=0}^{k-1} d_j N^S_j(X) \quad 0 \leq d_j < p^{a-e_j}
\]

with different canonical forms. Pick the smallest \( j \) such that \( c_j \neq d_j \). We claim that \( P(\alpha_j) \neq Q(\alpha_j) \). Note that

\[
P(\alpha_j) - Q(\alpha_j) = \sum_i (c_i - d_i) N^S_i(\alpha_j).
\]
Since $N_i^S(\alpha_j) = 0$ for $i > j$ and $c_i = d_i$ for $i < j$, we have
\[ P(\alpha_j) - Q(\alpha_j) = (c_j - d_j)N_j^S(\alpha_j). \]
Since $\text{val}_p(N_j^S(\alpha_j)) = c_j$ and $\text{val}(c_j - d_j) < a - e_j$ hence
\[ P(\alpha_j) - Q(\alpha_j) \not\equiv 0 \mod p^a. \]

**Theorem 17.** The set $S$ is a minimum interpolating set. In fact $k(I) = k(I) = |S|$.

**Proof.** Since $S$ is an interpolating set of size $k$, $k(I) \leq k$. It suffices to show that $k(I) \geq k$. Let $M(X)$ be a monic polynomial of degree $d \leq k - 1$. We can put $M(X)$ in the canonical form using the procedure above to get
\[ M(X) = \sum_{j \leq d} c_j N_j^S(X) \quad 0 \leq c_j < p^a - e_j. \]
Since $M(X)$ is monic, it follows that $c_d = 1$. Thus $M(X)$ does not compute the 0 function by Lemma 16. So $k(I) \geq k$. \qed

Note that Algorithm 1 for picking a minimum interpolating set is essentially a greedy algorithm: at each stage it picks a new element $x$ that minimizes $\sum_{j<i} \text{val}_p(x - \alpha_j)$. One can ask what objective function is being optimized by this greedy algorithm. In Theorem 30 (proved in Section 6), we prove that this algorithm minimizes the power of $p$ that divides the Vandermonde determinant of $\prod_{i<j}(\alpha_i - \alpha_j)$. In other words, the minimum interpolating sets of $I$ are all subsets $S = \{\beta_i\}$ of size $k(I)$ that minimize
\[ \sum_{i < j \leq k(I)} \text{val}_p(\beta_i - \beta_j). \]
This gives a simple algorithm to check if a set $T = \{\beta_i\}$ is a minimum interpolating set for $I$. We first compute an interpolating set $S = \{\alpha_i\}$ using Algorithm 1 and then check that $|T| = |S|$ and that $\sum \text{val}_p(\beta_i - \beta_j) = \sum \text{val}_p(\alpha_i - \alpha_j)$.

We now give an algorithm for Polynomial Interpolation over $\mathbb{Z}_{p^a}$ whose query complexity is optimal. One can show that this Algorithm computes the canonical form of $P(X)$ using an argument similar to Corollary 7.

**Algorithm 2.** Interpolate($I$, $\mathbb{Z}_{p^a}$)

**Input:** Set $I \subseteq \mathbb{Z}_{p^a}$, an black-box for $P(X)$ evaluated at $I$.

**Output:** The polynomial $P(X)$.

- Compute $S = \{\alpha_0, \ldots, \alpha_{k-1}\}$ using IntSet($I$, $p^a$).
- For $i = 0, \ldots, k - 1$,
  - Query $P(\alpha_i)$.
  - Compute $c_i$ so that $0 \leq c_i < p^a - e_i$ and
    \[ P(\alpha_i) \equiv \sum_{j \leq i} c_j N_j^S(\alpha_i) \mod p^a \]
- Output $P(X) = \sum_{i<k} c_i N_i^S(X)$. 
4.2. The General Interpolation Problem. We first give an algorithm to find interpolating sets over \( \mathbb{Z}_m \). The algorithm is given \( I \) as input, it does not have the factorization of \( m \). It computes an interpolating set for \( I \). We sketch the idea of the algorithm for \( m = pq \). The algorithm tries to add elements to \( S \) greedily like in the prime power case. Assume we have picked \( \{\alpha_0, \cdots, \alpha_{i-1} \} \) and let \( N^S_i(X) = \prod_{j<i}(X - \alpha_j) \). We compute \( g(x) = (N^S_i(x), m) \) for every \( x \in I \). This quantity plays the role of \( \text{val}_p(N^S_i(X)) \) in Algorithm 1.

1. If there is an \( x \) such that \( \text{val}_p(N^S_i(X)) \) and \( \text{val}_q(N^S_i(X)) \) are both minimized at \( x \), then \( g(x) = g(y) \) for all \( y \in I \). We add \( x \) to \( S \) and proceed.
2. If \( \text{val}_p(N^S_i(X)) \) and \( \text{val}_p(N^S_i(X)) \) are minimized at distinct points \( x \) and \( y \), then \( g(x) \neq g(y) \) and vice versa. Here the greedy approach fails. But in this case we can efficiently factor \( m = pq \) using \( g(x) \) and \( g(y) \). We then use divide and conquer.

For general \( m \), in case 2 we compute a factorization \( m = m_1m_2 \) where \( (m_1, m_2) = 1 \) using the subroutine Factor and then use divide and conquer.

**Algorithm 3. IntSet(I, m)**

**Input:** Set \( I \subseteq \mathbb{Z}_m \).

**Output:** An factorization \( m = \prod_{j=1}^{t'} m_j' \) where \( (m', m_j') = 1 \) and a minimum interpolating set \( S_j \) for \( I_j = I \mod m_j' \).

Pick \( \alpha_0 \in I \) arbitrarily. Set \( S = \{\alpha_0\}, i = 1 \).

Repeat

Let \( N^S_i(X) = \prod_{j<i}(X - \alpha_j) \).

If \( N^S_i(x) \) is zero for all \( x \in I \),

Output \( S = \{\alpha_0, \cdots, \alpha_{i-1}\}, m \) Stop.

Else

For each \( x \in I \), set \( g(x) = (N^S_i(x), m) \)

If some \( g(x) \) divides \( g(y) \) for all \( y \in I \),

Set \( \alpha_i = x \), \( i = i + 1 \).

Else

Find \( g(x), g(y) \) that do not divide each other.

Factor \( (m, g(x), g(y)) = m_1 \cdot m_2 \)

Return \( \text{IntSet}(I_1, m_1), \text{IntSet}(I_2, m_2) \). Stop.

We first analyze the Algorithm when Factor is not called. We then present the factoring subroutine. In particular, Lemmas 18, 19, 20 all assume that Factor was not called. In this case, the behavior of the algorithm is similar to the prime power case.

Let \( m = \prod_{j=1}^{t} p_j^{\alpha_j} \). Let \( S = \{\alpha_i\} \) be the set output. Let \( I_j \) and \( S_j \) be the projections of \( I \) and \( S \) modulo \( p_j^{\alpha_j} \). We show that if Factor is not called, then the Algorithm finds a minimum interpolating set by showing \( k(I) = k(I) = |S| \). This is done by simulating Algorithm 1 on \( I_j \) and showing that it would produce the same outcome.

Fix a prime \( p_j \). Let \( \alpha_i' \equiv \alpha_i \mod p_j^{\alpha_j} \). We take \( T \) to be the projection of the first \( k(I_j) \) elements of \( I \). In other words, let \( T = \{\alpha_1', \cdots, \alpha_{k(I_j)}'\} \). Note that \( T \subseteq S_j \).

**Lemma 18.** The set \( T \) is a minimum interpolating set for \( I_j \).

**Proof.** We will show that \( \text{val}_p(N^T_i(x)) \) is minimized over \( I_j \) at \( \alpha_i' \). Hence the set \( T \) is a possible output when we run Algorithm 1 on the set \( I_j \).
Assume that there is a \( y' \in I_j \) such that
\[
\text{val}_{p_j}(N^T_i(y')) < \text{val}_{p_j}(N^T_i(\alpha'_i)).
\]
Choose \( y \in I \) so that \( y \equiv y' \mod p_j^{\alpha'_i} \). Note that
\[
\text{val}_{p_j}(N^T(y')) = \text{val}_{p_j}(g(y)), \quad \text{val}_{p_j}(N^T(\alpha'_i)) = \text{val}_{p_j}(g(\alpha'_i))
\]

Hence \( \text{val}_{p_j}(g(y')) < \text{val}_{p_j}(g(\alpha'_i)) \).

So \( g(\alpha'_i) \) cannot divide \( g(y) \). But since Factor is not used, \( \alpha'_i \) satisfies \( g(\alpha'_i)|g(y) \) for all \( y \in I \), which is a contradiction. \( \square \)

**Lemma 19.** The set \( S \) is a minimum interpolating set for \( I \). In fact, \( k(I) = k^*(I) = |S| \).

**Proof.** By Lemma 18, the set \( S_j \) is an interpolating set for \( I_j \) so \( S \) is an interpolating set for \( I \). We will show that \( \bar{k}(I) = k(I) = |S| \).

For each \( j \), the polynomial \( \prod_{i<k(I_j)}(X - \alpha_i) \) is 0 mod \( p_j^{\alpha'_i} \) over \( I_j \) because the first \( k(I_j) \) elements are an interpolation set for \( I_j \). Take \( k = \max_j k(I_j) \). The polynomial \( \prod_{i<k}(X - \alpha_i) \) is 0 mod \( m \) over \( I \). Since this is the termination condition for Algorithm 3, it will stop after \( k \) steps and output \( S \) of size \( k = \max_j k(I_j) \). By Equation 3.3, we have
\[
\max_j k(I_j) \leq k(I).
\]

Hence the set \( S \) is a minimum interpolating set. Further by Equation 3.2,
\[
\bar{k}(I) = \max_j \bar{k}(I_j)
\]

But for prime powers, \( \bar{k}(I_j) = k(I_j) \). So we conclude that
\[
\bar{k}(I) = \max_j k(I_j) = k(I).
\]

\( \square \)

\( P(X) \) can be computed by a procedure similar to Algorithm 2.

**Lemma 20.** The polynomial \( P(X) \) can be computed from the values at points in \( S \).

**Proof.** The proof of correctness is similar to Corollary 7. For \( i \leq k \), the polynomials \( m \frac{m \prod_{i,N^S_i(\alpha_i)} N^S_i(X)} \) are 0 over \( I \). This is because
\[
\frac{m}{(m, N^S_i(\alpha_i))} N^S_i(\alpha_i) \equiv 0 \mod m
\]

and for every \( x \in I \),
\[
N^S_i(x) = y \cdot N^S_i(\alpha_i) \mod m \quad \text{for some } y \in \mathbb{Z}_m.
\]

Hence every polynomial function over \( I \) can be canonically represented as
\[
P(X) = \sum_{i=0}^{k-1} c_i N^S_i(X) \quad 0 \leq c_i < \frac{m}{(m, N^S_i(\alpha_i))}
\]
To compute the canonical form of $P(X)$, we query the value of $P(X)$ at every point in $S$. For $i \leq k - 1$ we solve the equation

$$c_i N_i^S(\alpha_i) \equiv P(\alpha_i) - \sum_{j<i} c_j N_j^S(\alpha_i) \mod m \quad 0 \leq c_i < \frac{m}{(m, N_i^S(\alpha_i))}$$

The unique solution to this system is the canonical from of $P(X)$. \(\square\)

We now turn to the subroutine for factoring. The idea is to use $g(x)$ and $g(y)$ to get $m_1, m_2$ which divide $m$ and are relatively prime. Their product $m_1 m_2$ might be less than $m$. At each step, we take a non-trivial divisor of $\frac{m}{m_1 m_2}$ and multiply either $m_1$ or $m_2$ by it. We do this in such a way that they stay relatively prime.

**Algorithm 4. Factor(m,g(x),g(y))**

**Input:** A number $m$ and $g(x), g(y)$ that divide $m$ but do not divide each other.

**Output:** $m_1 \cdot m_2 = m$ and $(m_1, m_2) = 1$.

Let $g = (g(x), g(y))$. Let $m_1 = \frac{g(x)}{g}, m_2 = \frac{g(y)}{g}$.

Repeat

1. Set $c = \frac{m}{m_1 \cdot m_2}$.
2. If $(c, m_1) = 1$, Set $m_2 = m_2 \cdot c$.
3. Else, Set $m_1 = m_1 \cdot (c, m_1)$.
4. If $m_1 \cdot m_2 = m$, Output $m_1, m_2$. Stop.

**Lemma 21.** Factor($m, g(x), g(y)$) returns $m_1, m_2$ such that $m_1 \cdot m_2 = m$ and $(m_1, m_2) = 1$.

**Proof.** At the start of the algorithm,

$$m_1 = \frac{g(x)}{(g(x), g(y))}, \quad m_2 = \frac{g(y)}{(g(x), g(y))}$$

so $(m_1, m_2) = 1$. Also $m_1, m_2$ are non-trivial divisors of $m$ since $g(x)$ and $g(y)$ do not divide each other.

Let

$$c = \frac{m}{m_1 m_2}.$$

If $(m_1, c) = 1$, since $(m_1, m_2) = 1$, we have $(m_1, cm_2) = 1$. So we set $m_2 = cm_2$ and we are done. If $(c, m_1) = d > 1$, then since $d$ divides $m_1$, we have $(d, m_2) = 1$. So we set $m_1 = dm_1$. In either case, the product $m_1 m_2$ increases by a factor of 2, hence the algorithm terminates in $O(\log m)$ iterations. \(\square\)

The subroutine above is somewhat inefficient. The running time can be considerably improved by running the factor refinement algorithm of Bernstein [12] on $g(x), g(y)$ and $m$. This algorithm gives a factorization into coprimes in near linear time.

If we find factors $m_1, m_2$ which are relatively prime, then we run IntSet($I_1, m_1$) and IntSet($I_2, m_2$). In doing so we could find further factors of $m_1$ and $m_2$, but these will be relatively prime, since $m_1$ and $m_2$ are relatively prime. So finally, the algorithm returns a factorization $m = \prod_{1 \leq i \leq t'} m_i'$ where the $m_i'$s are relatively prime. If $m$ has $t$ distinct prime factors then clearly $t' \leq t$. We now solve the interpolation problem
modulo \(m'_j\) using Lemma 20 and combine the results using the Chinese Remainder Theorem.

**Algorithm 5.** \textbf{Interpolate}(\(I, \mathbb{Z}_m\))
\begin{itemize}
  \item **Input:** Set \(I \subseteq \mathbb{Z}_m\), an black-box for \(P(X)\) evaluated at \(I\).
  \item **Output:** The polynomial \(P(X)\).
\end{itemize}

Using \textsc{IntSet}(\(I, m\)), compute \(m = \prod_{j=1}^{t'} m'_j\) and interpolating sets \(S_j\) for \(I_j\).

For each \(j \in 1, \ldots, t'\)
\begin{itemize}
  \item For each \(y \in S_j\),
    \begin{itemize}
      \item Query \(P(X)\) at \(x \in I\) s.t. \(x \equiv y \mod m'_j\).
    \end{itemize}
  \end{itemize}

Use these to compute \(P_j(X) \mod m'_j\).

Lift the polynomials \(P_j(X)\) to a polynomial \(P(X) \in \mathbb{Z}_m[X]\) using Chinese Remaindering.

**Lemma 22.** Algorithm 5 solves the interpolation problem over \(\mathbb{Z}_m[X]\). The number of queries is within a factor \(t'\) of the optimal.

**Proof.** The proof that the polynomial \(P(X)\) is correct follows by the Chinese Remainder Theorem. The number of queries is at most \(\sum_{j \leq t'} |S_j|\). By Lemma 19 since each \(m_j\) is not factored further, the set \(S_j\) is a minimum interpolating set for \(I_j\). Hence \(|S_j| = k(I_j)|\). Hence by Corollary 14,
\[
\max_j |S_j| \leq k(I) \leq \sum_{j \leq t'} |S_j| \leq t'k(I).
\]

Also by Corollary 14,
\[
\bar{k}(I) = \max_j k(I_j) = \max_j |S_j|.
\]

Hence Algorithm 3 can be used to compute \(\bar{k}(I)\) exactly. (Note that this can also be done by solving a system of linear equations).

5. Learning Algorithms. One can use the algorithms in the previous section to design efficient algorithms for interpolation over \(\mathbb{Z}_m\) in various learning theoretic settings. We consider the problem of learning under the uniform distribution and PAC-learning under an arbitrary distribution. In the uniform distribution problem, we are given evaluations of a polynomial \(P(X)\) at points \(x\) chosen at random from \(\mathbb{Z}_m\). In the PAC-learning problem, the samples are drawn from an unknown distribution \(D\) over \(\mathbb{Z}_m\). We are required to output a polynomial that computes \(P(X)\) correctly with good probability on points chosen from the same distribution. In this setting, it is necessary to allow some error probability. Consider a distribution \(D\) which is concentrated on a set \(I\) which does not contain an interpolating set for \(\mathbb{Z}_m\). A polynomial time algorithm cannot distinguish between the 0 function and a function which is 0 on \(I\) but non-zero elsewhere.

For learning algorithms, the notion of polynomial running time needs to be defined carefully. Let \(F(m)\) denote the number of polynomial functions over \(\mathbb{Z}_m\). The algorithm is required to output some polynomial function which requires at least \(\log F(m)\) bits to represent. Hence we say the algorithm runs in polynomial time if the running time is \(\text{poly}(\log F(m))\).
From Theorem 6, we get

\[ F(m) = \prod_{0 \leq j < k(m)} \frac{m}{(m, j^m)} \quad (5.1) \]

Note that \( \log F(m) \) can vary between \( \log m \) and \( m \) depending on the prime factorization of \( m \). We compute a rough lower bound on \( \log F(m) \) in terms of its factorization. Note that if \( m = p^a \), it follows from Theorem 6 that

\[ F(p^a) \geq (p^a)^p = p^{pa}. \]

Hence if \( m = \prod_{j=1}^{t} p_j^{a_j} \), then

\[ F(m) \geq \prod_j p_j^{a_j p_j} \Rightarrow \log F(m) \geq \sum_{j \leq t} a_j p_j \log p_j. \]

5.1. Learning under the Uniform Distribution. We first consider the problem of learning polynomials from their evaluations at random points in \( \mathbb{Z}_m \).

**Problem 4. Learning under the Uniform Distribution:** Given samples \((x, f(x))\), where \( x \) is drawn uniformly from \( \mathbb{Z}_m \) and \( f \) is a polynomial function, find a polynomial \( P(X) \) that computes \( f \).

**Algorithm 6. Interpolation under the Uniform Distribution**

**Input:** Black-box for evaluations of \( P(X) \) under the uniform distribution.

**Output:** The polynomial \( P(X) \).

- Compute the factorization \( m = \prod_{j=1}^{t} p_j^{a_j} \).
- Draw samples till we have an interpolating set \( S_j \) for \( \mathbb{Z}_{p_j^{a_j}} \).
- Compute \( P_j(X) \) for each \( j \).
- Let \( P(X) \) be the lift of the \( P_j(X) \)'s.

We compute the factorization using brute force which takes time \( O(\sum_j p_j a_j) = O(F(m)) \). We now bound the number of samples needed till we have an interpolating set modulo \( p^a \). Let \( p^b \) be the smallest power of \( p \) such that \( p^b > k(p^a) \). By Lemma 8, \( p^b < p^{2a} \). Let \( S = \{\alpha_0, \ldots, \alpha_{k(p^a)-1}\} \) where \( \alpha_i \equiv i \mod p^b \).

**Lemma 23. The set \( S \) is an interpolating set for \( \mathbb{Z}_{p^a} \).**

**Proof.** By Corollary 7, \( T = \{0, \ldots, k(p^a)-1\} \) is an interpolating set for \( \mathbb{Z}_{p^a} \). By the choice of \( \alpha_i, \alpha_j \)

\[ \alpha_i - \alpha_j \equiv i - j + cp^b \mod p^a \]

Since \( 0 \leq i \neq j < p^b \), \( \text{val}_p(\alpha_i - \alpha_j) = \text{val}_p(i - j) \). Hence

\[ \sum_{i<j \leq k(p^a)} (\alpha_i - \alpha_j) = \sum_{i<j \leq k(p^a)} (i - j) \]

So by Theorem 30, \( S \) is an interpolating set. \( \square \)

**Lemma 24. Algorithm 6 requires \( O(\log^2 F(m)) \) samples with high probability.**

**Proof.** The uniform distribution over \( \mathbb{Z}_m \) induces the uniform distribution over congruence classes modulo \( p_j^{a_j} \) since \( b_j < a_j \). By the coupon collector’s problem,
in time $O(p_j^b \log(p_j^b))$ we will see a sample from each congruence class with high probability. By Lemma 23, this gives an interpolating set modulo $p_j^a$. Overall the number of samples needed can be bounded by $O(\log^2 F(m))$ w.h.p. □

Theorem 25. Algorithm 6 learns the polynomial $P(X)$ exactly under the uniform distribution. It runs in time $O(\log^2 F(m))$ with high probability.

We needed to factor $m$ to check whether the set of points seen so far is an interpolating set for $\mathbb{Z}_m$. Is there an algorithm to check if $S$ is an interpolating set for $\mathbb{Z}_m$ that does not need to factor $m$?

5.2. PAC Learning. Next we consider the problem of PAC-learning polynomials. We refer the reader to the book of Kearns and Vazirani for details of the PAC-learning model [29].

Problem 5. PAC Learning: Given samples $(x, f(x))$, where $x$ is drawn from an unknown distribution $D$ and $f$ is a polynomial function, find a polynomial $P(X)$ that computes $f$ over $D$ with probability $1 - \varepsilon$.

We show that polynomial functions are PAC learnable under an arbitrary distribution in polynomial time. Once we have drawn the set of samples, the problem reduces to one of general interpolation. The number of samples to be drawn can be determined from $F(m)$ using Occam’s Razor [29]. We first compute $F(m)$ using Equation 5.1. This can be done in time $O(\log F(m))$. We then draw $\frac{1}{\varepsilon} \log \frac{F(m)}{\delta}$ samples from $D$ and solve the interpolation problem on these inputs using Algorithm 5. The proof that this suffices for PAC-learning is standard [29].

Theorem 26. Polynomial over $\mathbb{Z}_m$ are PAC learnable in polynomial time using $\frac{1}{\varepsilon} \log \frac{F(m)}{\delta}$ queries.


In this section we study the algebraic properties of interpolating sets modulo prime powers. We give alternate algebraic characterizations of such sets (Theorems 29 and 30). In this section and the next, we use $p$-adic distance as opposed to valuations.

Recall that by the definition of Interpolating sets, every polynomial which is non-zero over $I$ is in fact non-zero over some point in $S$. The next Lemma generalizes this to show that in fact, the norm of every polynomial is maximized over $I$ at some point in $S$.

Lemma 27. A set $S$ is an interpolating set iff for any polynomial $P(X)$, there exists $\alpha \in S$ such that

$$|P(\alpha)|_p \geq |P(x)|_p \quad \forall x \in I$$ (6.1)

Proof. ($\Rightarrow$) Assume there exists $P(X) \in \mathbb{Z}_{p^e}$ such that $|P(x)|_p > |P(\alpha)|_p \quad \forall \alpha \in S$. But then for an appropriately chosen $e$, $p^e P(X)$ is non-zero at $x$, but 0 everywhere in $S$. Hence $S$ cannot be an interpolating set.

($\Leftarrow$) Assume that $S$ satisfies equation 6.1. There cannot exist a polynomial $P(X)$ which is 0 on $S$, but $P(x) \neq 0$ for some $x \in \mathbb{Z}_{p^e}$, since this implies that $|P(x)|_p > |P(\alpha)|_p \quad \forall \alpha \in S$. Hence $S$ is an interpolating set. □

This property of interpolating sets allows us to order its elements in a natural manner. Given an ordered set $S = \{\alpha_0, \alpha_1, \cdots\}$, let $N^S_j(X) = \prod_{i<j} (X - \alpha_i)$. 
By our choice of $\alpha$, we bound polynomial $P$.

To prove the other direction, simply take $Z$ computed by a unique polynomial of the form $I$.

Assume that the above procedure outputs an ordered set $S = \{\alpha_0, \cdots, \alpha_{k-1}\}$ of size $k$. Let $e_j = \text{val}_p(N_j^S(\alpha_j))$ for $i \leq j \leq k - 1$. Observe that $0 \leq e_j < a$. Using the argument of Theorem 16, we can show that every polynomial function over $I$ is computed by a unique polynomial of the form

$$P(X) = \sum_{j=0}^{k} c_j N_j^S(X) \quad 0 \leq c_j < p^{a-e_j}$$

Using the canonical form above, one can show that all minimal interpolating sets over $\mathbb{Z}_{p^n}$ have the same size. The proof is similar to that of Theorem 17.

**Corollary 28.** The set $S = \{\alpha_0, \cdots, \alpha_{k-1}\}$ is an interpolating set if and only if $|N_j^S(\alpha_j)|_p \geq |N_j^S(x)|_p$ for all $x \in I$.

**Proof.** Clearly an interpolating set with the canonical ordering has this property. To prove the other direction, simply take $T = I$ in Algorithm 7. Since $|N_j^S(x)|_p$ is maximized at $\alpha_j$, we can add $\alpha_j$ to $S$ at step $j$, giving the interpolating set $S = \{\alpha_0, \cdots, \alpha_{k-1}\}$.

Henceforth we will assume that interpolating sets are canonically ordered and that polynomials are in the canonical form. Lemma 27 states that for any polynomial function $P(X)$, $|P(x)|_p$ is maximized at some point $\alpha \in S$. We strengthen this to show that if the degree of $P(X)$ is $d$, such an $\alpha$ can be found among the first $d + 1$ elements in $S$.

**Theorem 29.** The set $S = \{\alpha_0, \cdots, \alpha_{k-1}\}$ is an interpolating set if for every polynomial $P(X)$ of degree $d$, there exists $\alpha \in \{\alpha_0, \cdots, \alpha_d\}$ such that $|P(\alpha)|_p \geq |P(x)|_p \quad \forall x \in I$.

**Proof.** Clearly a set with this property is an interpolating set by Lemma 27. We prove the other direction. The proof is by induction on $d$. The base case when $d = 0$ is trivial. Assume the claim holds for $d - 1$. Let $P(X) = Q(X) + c_d N_d^S(X)$ where $\deg(Q) \leq d - 1$.

$$|P(x)|_p \leq \max(|Q(x)|_p, |c_d N_d^S(x)|_p) \quad \text{(Ultrametric inequality)} \quad (6.2)$$

We bound $|Q(x)|_p$ using the inductive hypothesis. Since $Q(X)$ has degree $d - 1$,

$$|Q(x)|_p \leq \max(|Q(\alpha_0)|_p, \cdots, |Q(\alpha_{d-1})|_p) \quad (6.3)$$

By our choice of $\alpha_d$,

$$|c_d N_d^S(x)|_p \leq |c_d N_d^S(\alpha_d)| = |P(\alpha_d) - Q(\alpha_d)|_p \leq \max(|Q(\alpha_0)|_p, \cdots, |Q(\alpha_{d-1})|_p) \quad (6.4)$$
Hence from Equations 6.2, 6.3 and 6.4 we get
\[ |P(x)|_p \leq \max(|Q(\alpha_0)|_p, \ldots, |Q(\alpha_{d-1})|_p, |P(\alpha_d)|_p) \]
Since \( N^S_d(\alpha_j) = 0 \) for \( j < d \), we have \( Q(\alpha_j) = P(\alpha_j) \) for \( j < d \). Hence
\[ |P(x)|_p \leq \max(|P(\alpha_0)|_p, \ldots, |P(\alpha_{d-1})|_p, |P(\alpha_d)|_p) \quad (6.5) \]

We use this to show that interpolating sets are greedy solutions for the problem of maximizing the \( p \)-adic norm of the Vandermonde determinant. Since the determinant could vanish mod \( p^a \), we define the norm of the Vandermonde determinant as follows

Let \( |V(\alpha_0, \ldots, \alpha_{k-1})|_p = \prod_{0 \leq i < j \leq k-1} |(\alpha_i - \alpha_j)|_p = \prod_{j=1}^{k-1} |N^S_j(\alpha_j)|_p \)

This is equivalent to regarding the determinant as an integer and taking its norm.

**Theorem 30.** The set \( S = \{\alpha_0, \ldots, \alpha_{k-1}\} \) is an interpolating set for \( I \) iff for all subsets \( \{x_0, \ldots, x_{k-1}\} \) of \( I \),
\[ |V(\alpha_0, \ldots, \alpha_{k-1})|_p \geq |V(x_0, \ldots, x_{k-1})|_p \quad (6.6) \]

**Proof.** (\( \Rightarrow \)). We will show a stronger statement: for \( 1 \leq j \leq k-1 \), for any subset \( \{x_0, \ldots, x_j\} \) of \( I \),
\[ |V(\alpha_0, \ldots, \alpha_j)|_p \geq |V(x_0, \ldots, x_j)|_p \quad (6.7) \]
Consider the polynomial \( Q(X) = \prod_{i<j}(X-x_i) \). By Theorem 29, there exists \( \alpha_i \in \{\alpha_0, \ldots, \alpha_j\} \) such that \( |Q(\alpha_i)|_p \geq |Q(x_j)|_p \). Hence one can replace \( x_j \) by \( \alpha_i \) without decreasing the norm of the Vandermonde determinant. Now repeat the same argument for the set \( \{x_0, \ldots, x_{j-1}, \alpha_i\} \) and the element \( x_{j-1} \) and so on. We get
\[ |V(\alpha_0, \ldots, \alpha_j)|_p \geq \cdots \geq |V(x_0, \ldots, x_{j-1}, \alpha_i)|_p \geq |V(x_0, \ldots, x_j)|_p \]

(\( \Leftarrow \)). Assume we have a set \( S \) satisfying Equation 6.6. Assume that the \( \alpha_s \) are ordered canonically. We will show that \( |N^S_j(\alpha_j)|_p \geq |N^S_j(x)|_p \) for all \( x \in I \). This implies \( S \) is an interpolating set by Corollary 28.
Assume there exists \( \beta \) such that \( |N^S_j(\alpha_j)|_p < |N^S_j(\beta)|_p \). Pick \( \alpha_i \in \{\alpha_j, \ldots, \alpha_{k-1}\} \) such that \( |\beta - \alpha_i|_p \) is minimized. We will show that replacing \( \alpha_i \) by \( \beta \) will increase the norm of the Vandermonde determinant. Observe that for any \( \ell \neq i \) and \( \ell \geq j \),
\[ |\alpha_\ell - \alpha_i|_p \leq \max(|\beta - \alpha_\ell|_p, |\beta - \alpha_i|_p) \]
But \( |\alpha_\ell - \beta|_p \geq |\beta - \alpha_i|_p \) by choice of \( \alpha_i \)
Hence \( |\alpha_i - \alpha_\ell|_p \leq |\beta - \alpha_\ell|_p \)
\[ \Rightarrow \prod_{\ell \geq j, \ell \neq i} |\alpha_i - \alpha_\ell|_p \leq \prod_{\ell \geq j, \ell \neq i} |\beta - \alpha_\ell|_p \quad (6.8) \]
We also have
\[ |N^S_j(\alpha_i)|_p \leq |N^S_j(\alpha_j)|_p < |N^S_j(\beta)|_p \]
The first inequality is because $S$ is ordered canonically, the second is by the definition of $\beta$. Hence from the definition of $N_S^j(X)$,

$$\prod_{\ell<j}(\alpha_i - \alpha_\ell)_p < \prod_{\ell<j}(\beta - \alpha_\ell)_p$$

(6.9)

Combining Equations 6.8 and 6.9, we get

$$\prod_{\ell\neq i}(\alpha_i - \alpha_\ell)_p < \prod_{\ell\neq i}(\beta - \alpha_\ell)_p$$

(6.10)

Hence replacing $\alpha_i$ with $\beta$ increases the norm of the Vandermonde determinant, which contradicts the assumption that the norm is maximized at $S$. 

**Corollary 31.** The parameters $e_1, \cdots, e_{k-1}$ are independent of choice of interpolating set.

**Proof.** Note that

$$|N^S_j(\alpha_j)| = p^{-e_j}$$

and

$$|V(\alpha_0, \cdots, \alpha_j)|_p = \prod_{i\leq j}|N^S_i(\alpha_i)|_p = p^{-\sum_{i\leq j} e_i}$$

This quantity is maximized over subsets of $I$ at every interpolating set. So $\sum_{i\leq j} e_i$ and hence $e_i$ is the same for every interpolating set of $I$. 

**7. Some Combinatorial Properties of Ultrametric Spaces.** We show that many of our results for interpolating sets can be translated into properties of general ultrametric spaces. Further, the proof of these properties for general ultrametric spaces follows directly from the proof for polynomials over $\mathbb{Z}_p$.

**Definition 3.** Let $T$ be a tree rooted at a vertex $r$, such that the distances of all leaves from the root $r$ are equal. The metric space whose points are the leaves of the tree and distance is the shortest path in the tree is called an equidistant tree and denoted by $(T, d)$.

It is easy to show that $(T, d)$ is an ultrametric. In fact, the converse is also true.

**Fact 32.** Every finite ultrametric space embeds isometrically into an equidistant tree.

Every equidistant tree can in turn be associated with $I \subseteq \mathbb{Z}_p$ for appropriate choices of $p, a$ and $I$.

**Lemma 33.** There is a mapping from any equidistant tree $T$ to $I \subseteq \mathbb{Z}_p$ for some $p, a$ such that

$$|x - y|_p = p^{d(x, y)/2 - a} \quad \text{for } x \neq y$$

**Proof.** There is a natural way to associate $\mathbb{Z}_{p^a}$ with an equidistant tree of degree $p$ and depth $a$ [6]. The root is at depth 0. The edges from each vertex to its descendants are labeled $\{0, \cdots, p - 1\}$. Given a point $x = \sum_i x_i p^i \in \mathbb{Z}_{p^a}$, we associate it with a leaf of the tree as follows: Start from the root. At depth $i$, follow the edge labeled $x_i$. Thus the leaf nodes correspond to points in $\mathbb{Z}_{p^a}$, while nodes at depth $d$ correspond
to congruence classes modulo \( p^d \). If \( d(x,y) \) is the tree distance between the points, then \( |x - y|_p = p^{\frac{d(x,y)}{2} - a} \).

Given an equidistant tree \( T \), we take \( p \) to be a prime larger than the maximum degree of \( T \), and \( a \) to be the depth of the tree. For any node, we arbitrarily label the edges to its descendants with \( \{0, \ldots, p-1\} \). This can be done since there are at most \( p \) of them. This will map the leaves of \( T \) to \( I \subseteq \mathbb{Z}_{p^a} \) and it is easy to verify that the distance satisfies the desired condition.

Based on this correspondence, we can translate properties of interpolating sets to properties of ultrametric spaces. We consider the following NP-hard optimization problem.

**Problem 6. Max-Dist-k:** Given a metric space \( (X, d) \), pick a subset \( S \) of \( k \) points such that the sum of pairwise distances is maximized.

For ultrametrics, the problem can be solved greedily.

**Algorithm 8. Greedy Algorithm for Max-Dist-k**

*Input:* An \( n \) point ultrametric space \( (T, d) \).

*Output:* A subset \( S \) of size \( k \) maximizing the sum of pairwise distances.

1. Pick \( \alpha_0 \in T \) arbitrarily.
2. For \( j \leq k-1 \), pick \( \alpha_j \) so that \( \sum_{i<j} d(\alpha_j, \alpha_i) \) is minimized.
3. Output \( S = \{\alpha_0, \ldots, \alpha_{k-1}\} \).

**Lemma 34.** The greedy algorithm solves Max-Dist-k over Ultrametric Spaces.

*Proof.* Associate \( T \) with \( I \subseteq \mathbb{Z}_{p^a} \). For any subset \( (x_0, \ldots, x_{k-1}) \) of size \( k \),

\[
\prod_{i<j} |x_i - x_j|_p = p^{\sum_{i<j} \frac{d(x_i, x_j)}{2} - \binom{k}{2} a}
\]

Hence Max-Dist-k on an ultrametric reduces to choosing \( k \) points in \( I \) such that \( |V(x_0, \ldots, x_{k-1})|_p \) is maximized, by Theorem 30 this can be done by choosing the points greedily.

Next we consider the problem of finding a point in a metric space that is farthest from a given set of points.

**Problem 7. Farthest-Point:** Given a metric space \( (X, d) \), and a set of points \( \{y_1, \ldots, y_{k-1}\} \) of size \( k-1 \), find the point \( x \in X \) that maximizes \( \sum_{i<k} d(x, y_i) \).

This problem is easy to solve for arbitrary metric spaces, we can just try every point in \( X \) and pick the best. However, ultrametric spaces admit a more efficient solution with some pre-processing. In the pre-processing step, we find a solution \( S \) to Max-Dist-k using the greedy algorithm. This step is oblivious of the \( y_j \)s. We then return the point \( x \in S \) that maximizes \( \sum_{i<k} d(x, y_i) \). The running time of this step depends only on \( k \), it is independent of the number of points \( n \).

**Lemma 35.** Let \( \{y_1, \ldots, y_{k-1}\} \) be any subset of size \( k-1 \) in \( X \). Let \( S = \{\alpha_0, \ldots, \alpha_{k-1}\} \) be a greedy solution to Max-Dist-k. The quantity \( \sum_{i<k} d(x, y_i) \) is maximized over \( X \) at a point \( \alpha \in S \).

*Proof.* Associate \( X \) with \( I \subseteq \mathbb{Z}_{p^a} \). Given points \( y_i \), consider the polynomial

\[
P(X) = \prod_{i<k} (X - y_i).
\]
Note that 

\[ |P(x)|_p = \sum_{d(x,y_i)}^{d(x,y_i)} \cdot a(k-1) \]

Hence, maximizing the distance is equivalent to maximizing \(|P(x)|_p\). Since \( P(X) \) is of degree \( k - 1 \), by Theorem 29 its norm over \( I \) is maximized at some point in \( \{\alpha_0, \ldots, \alpha_{k-1}\} \).

The case \( k = 2 \) of this Lemma is a direct consequence of the ultrametric inequality. We are unaware of a direct combinatorial proof of Lemma 35 for \( k \geq 3 \) and higher.

Acknowledgments. We thank Sophie Frisch, Igor Shparlinski and an anonymous referee for many useful references and Saugata Basu and Ernie Croot for helpful discussions. We thank T. S Jayram, Ravi Kumar and Satyen Kale for their comments which greatly improved the presentation. We thank Dick Lipton for numerous illuminating discussions on this and related problems.

REFERENCES