HMC Senior Theses

HMC Student Scholarship

2007

Special Cases of Carry Propagation

Alexander Izsak

Harvey Mudd College

Special Cases of Carry Propagation

Alexander Izsak

Nick Pippenger, Advisor

Ran Libeskind-Hadas, Reader

May, 2007

HARVEY MUDD
COLLEGE

Department of Mathematics

Abstract

The average time necessary to add numbers by local parallel computation is directly related to the length of the longest carry propagation chain in the sum. The mean length of longest carry propagation chain when adding two independent uniform random n bit numbers is a well studied topic, and useful approximations as well as an exact expression for this value have been found. My thesis searches for similar formulas for mean length of the longest carry propagation chain in sums that arise when a random n-digit number is multiplied by a number of the form $1 + 2^d$.

Letting $C_{n,d}$ represent the desired mean, my thesis details how to find formulas for $C_{n,d}$ using probability, generating functions and linear algebra arguments. I also find bounds on $C_{n,d}$ to prove that $C_{n,d} = \log_2 n + O(1)$, and show work towards finding an even more exact approximation for $C_{n,d}$.

Contents

List of Figures

Acknowledgments

Prof. Pippenger's insight, encyclopedic math knowledge and lucid explanations has made this project a joy. Craig Weidert's and Steven Ehrlich's LaTeX skills have helped me pass more than a few frustrations. I join my praise with all math majors to Claire Connelly for her clear guides to making posters and thesis documents and for printing our posters.

Chapter 1

Introduction

1.1 A Question Explained

Add two numbers x and y in base b by the following formula

$$x + y = S_{xy} + C_{xy}.$$

Here C_{xy} are the carry digits. According to the rules of addition as taught in countless elementary schools, we carry a 1 to the i^{th} digits if the $(i-1)^{\text{th}}$ digits of x and y add to b or more. So in this case the i^{th} digit of C_{xy} is 1. If instead the $(i-1)^{\text{th}}$ digits of x and y add to less than b, then the i^{th} digit of C_{xy} is 0.

In the formula defining addition, S_{xy} are the sum digits. The i^{th} digit of this number is the sum of the i^{th} digits of x and y modulo b. The value for $S_{xy} + C_{xy}$ is simply S_{xy} if $C_{xy} = 0$. If not, we then find $S_{xy} + C_{xy}$ by repeating this process. Each iteration of this algorithm can be completed in constant time by local parallel computing. The question of how many operations are necessary to compute the sum of two numbers then directly relates to the number of times we must repeat this algorithm.

Below is an example of the summation process described. Note in this paper we only consider operations in binary, although it would not be difficult to extend these results to other bases.

$$\begin{array}{lr}
 & 01011 \\
 & +\ 10110 \\
\hline
\text{Carry} & 00100 \\
\text{Sum} & +\ 11101 \\
\hline
\text{Carry} & 01000 \\
\text{Sum} & +\ 11001 \\
\hline
\text{Carry} & 10000 \\
\text{Sum} & +\ 10001 \\
\hline
 & 100001
\end{array}$$

In the above example the second bit position causes a carry to occur in the third position. This carry subsequently creates another carry, which then causes another. We may then call bit positions 2 through 5 a *carry propagation chain* of length 4, although we will consider positions 2 though k, $2 \leq k < 5$ as carry propagation chains as well.

From the example we see that the length of the longest carry propagation chain is the number of times we must repeat the described adding algorithm. This is true since we must complete another new addition when a carry digit causes another carry. The question of how long on average is the longest carry chain of two random n-bit numbers when summed together then is of particular interest, since we may derive from the answer the amount of time on average needed to sum. Accordingly carry propagation is one of the oldest problems studied in the analysis of algorithms. Over decades several formulas of varying accuracy and utility have been discovered that answered the question described above.

My thesis deviates from established research by instead considering the type of addition that arises from multiplying a random n-bit number by $1 + 2^d$. So I am investigating the average length of the longest carry propagation chain when adding a random n-bit numbers of the form $x_n x_{n-1} ... x_1$ to another with bits $x_n x_{n-1} ... x_1 00...0$. The end goal is to find exact and approximate formulas for the mean length of the longest carry propagation chain from such an addition. This mean length is denoted by $C_{n,d}$.

My research has been split into the following goals.

- Determine lower and upper bounds for $C_{n,d}$ using only first- and second-moment probabilistic arguments. From these bounds I found that
$$C_{n,d} = \log_2 n + O(1).$$
- Using a recursive argument, derive a generating function that describes $C_{n,d}$ and from that generating function find a more precise

asymptotic formula. This work led to a proof that

$$C_n = \log_2 n + \gamma \log_2 e - \frac{3}{2} - F(\log_2 n) + O\left(\frac{(\log n)^4}{n}\right) \qquad (1.1)$$

with $|F(v)| \leq 1.573... \times 10^{-6}$.

- Discover exact formulas for $C_{n,d}$. I ended up finding for example that

$$\begin{aligned} C_{n,1} = \quad & \textstyle\sum_{k>0,k \text{ even}} \sum_{j\geq 1} \binom{n-jk}{j} \frac{(-1)^{j-1}}{2^{j(k+1)}} + \binom{n-jk}{j-1} \frac{1}{2^{(j-1)(k+1)+k}} \\ & + \textstyle\sum_{k>0,k \text{ odd}} \sum_{j\geq 1} \sum_{l\geq 0} (-1)^{j-1} \left(\begin{array}{c} n-jk \\ j-l \end{array} \right) \left(\left(\begin{array}{c} j-l \\ l \end{array} \right) \right) / 2^{j(k+1)-l} \quad . \end{aligned}$$

  I also created a method by which such exact formulas could be found for any d, although it takes time exponential in d. I wrote a program in Mathematica that outputs long although easy to solve generating functions for these formula.

1.2 Terminology and Useful Basics

A bit position for two summed numbers *generates* a carry if both summands are 1 in this position. If the summands are a 1 and a 0, we say that bit position *propagates* a carry. Then a carry propagation chain is some consecutive bit positions such that the rightmost position generates a carry, and the rest propagate a carry. We call a set of k conescutive bit positions a *k-block*, and if a k-block is also a carry propagation chain, then we refer to it as an *active* k-block.

Suppose x has binary digit representation $x_n x_{n-1}...x_1$. Then the multiplication $x(2^d + 1)$ produces a sum with an active k-block starting in the l bit position only if $x_l = 1$ and $x_{l-d} = 1$. This ensures the active k-block starts with a generator. To ensure that every other bit position propagates a carry, then $x_{l+j} = \overline{x_{l+j-d}}$ must hold for $1 \geq j \geq k$. If some $k + d$ consecutive bits in x when multiplied by $1 + 2^d$ gives rise to an active k-block, then we refer to those $k + d$ digits as an active k-block with regard to multiplication by $1 + 2^d$.

According to these rules an active k-block with regard to multiplication by 11 must be $k + 1$ digits of the form ...0101011. More generally, an active k-block with regard to multiplication by $1 + 2^d$ is $k + d$ digits of the form $...0\overline{x_{d-1}x_{d-2}}...\overline{x_1}1x_{d-1}x_{d-2}...x_1 0\overline{x_{d-1}x_{d-2}}...\overline{x_1}1x_{d-1}x_{d-2}...x_1 1$. Here x_j may be 1 or 0 and $\overline{x_j}$ is the opposite number. So there are 2^{d-1} different possible active k-blocks with regard to this multiplication. The probability that $k+d$ random ordered digits are an active k-block then is $2^{d-1}/2^{k+d} = 1/2^{k+1}$. Note that almost every time I refer to an active k-block it will be with regard to some multiplication, and I will bring it to the reader's attention when this is not the case.

If we multiply the number 11 by $1+2^1$ we see that this causes one carry which then propagates another carry. So the number 11 should be an active 2-block but according to the rules of active k-blocks above it is just an active 1-block. Notice we do not have this inconsistency if the digits 11 are not the leftmost portion of a number, as 011 is the only set of 3-digits that is an active 2-block. Something is happening at the very end of the number.

When we add the number to itself shifted over d digit, to the left of the original number there are d implicit 0's s. So the number 11 with regard to multiplication by $1+2^1$ we may consider as actually being the number 011. This does not change the results of adding or multiplying the number. The new representation though does allow us to see that the bits 11 at the end of a number are an active 2-block. More generally, appending d zeros allows us to use the definition above to check whether the last few digits of a number are an active k-block. We call an active k-block that includes any of the implicit zeroes at to the left of a number a *truncated* active k-block. We use this term because a truncated active k-block in the original representation of the number is an active k-block but with leftmost positions that ought to be zero removed. We count the length of a truncated k-block as $k+d$ minus the number of digits that aren't in the original representation of the number. We also call an active k-block that is not truncated a *full* active k-block.

Chapter 2

Rough Bounds

2.1 An Upper Bound

In this section we derive decent bounds for $C_{n,d}$ through only very basic probabilistic properties. This proof is due to ideas from J. von Neumann's proof of an upper bound and V. Claus's proof of a lower bound for C_n as cited in [3].

Let $\mathbf{C}_{n,d}$ be the random variable denoting the length of the longest active k-block in a random n-bit number with regard multiplication by $1 + 2^d$. Note that by partial summation

$$C_{n,d} = \sum_{k \geq 0} k \Pr[\mathbf{C}_{n,d} = k] = \sum_{k \geq 1} \Pr[\mathbf{C}_{n,d} \geq k]. \tag{2.1}$$

Now define $\mathbf{B}_{n,d,k}$ to be the random variable denoting the number of active k-blocks when multiplying a uniformly distributed n-bit number by $1 + 2^d$. This variable is useful to consider since

$$\Pr[\mathbf{C}_{n,d} \geq k] = \Pr[\mathbf{B}_{n,d,k} \geq 1]. \tag{2.2}$$

This holds because a carry propagation chain of length at least k occurs if and only if at least one k-block is active.

A full k-block is active with probability $1/2^{k+1}$. Such a k-block requires $k + d$ bits and so there exists $n - k - d + 1$ distinct full k-blocks among n digits. A truncated k-block is active with probability at most $1/2^k$, since for each bit of the block not in the number, we gain a factor of 2 for not having to set that bit to a specific value, but can gain a factor of 1/2 since we have to label the bit d to the left of it as 1. We do not gain the factor of 1/2 only when a bit not in the number but in the active k-block is the $(cd + 1)^{\text{th}}$

with even c and so the bit d to the left must have been a 1 anyway. There are altogether d possible truncated k-blocks in the number, since an active k-block may end in at most d zeros. This all implies that

$$\mathrm{Ex}[\mathbf{B}_{n,d,k}] \leq (n-k-d+1)/2^{k+1} + d/2^k. \tag{2.3}$$

Markov's inequality shows $\Pr[\mathbf{B}_{n,d,k} \geq 1] \leq \min\{1, \mathrm{Ex}[\mathbf{B}_{n,d,k}]\}$. Using (2.2) to rewrite the lefthand side of the inequality, and (2.3) to rewrite the righthand side, we derive

$$\Pr[\mathbf{C}_{n,d} \geq k] \leq \min\{1, (n-k-d+1)/2^{k+1} + d/2^k\}. \tag{2.4}$$

Assuming $\log_2 n \geq d+2$ implies $1 \geq (n-k-d+1)/2^{k+1} + d/2^k$ for k at least $\log_2 n - 1$. So summing over all k and using (2.1) for a substitution leads to

$$\begin{aligned} C_{n,d} &\leq \sum_{1 \leq k \leq \log_2 n - 1} 1 + \sum_{k > \log_2 n - 1} \left(\frac{(n-k-d+1)}{2^{k+1}} + \frac{d}{2^k} \right) \\ &\leq \log_2 n + \frac{3d}{n} + 1. \end{aligned}$$

Our analysis implicitly assumed $n > d$, else we'd be shifting the original number by so much the sum could have no carries. It's important to note this, else the previous inequality would imply $C_{n,d}$ is negative for large enough values of d.

2.2 A Lower Bound

Let $C'_{n,d}$, $\mathbf{C}'_{n,d}$ and $\mathbf{B}'_{n,k,d}$ be defined the same as $C_{n,d}$, $\mathbf{C}_{n,d}$ and $\mathbf{B}'_{n,k,d}$ except with regards to only full active k-blocks. So for example $C'_{n,d}$ is the average length of longest full active k-block in a random n bit number with regard to multiplication by $1+2^d$. We can disregard the $d/2^k$ term from (2.3) contributed by truncated k-blocks to find that

$$\mathrm{Ex}[\mathbf{B}'_{n,d,k}] = (n-k-d+1)/2^{k+1}. \tag{2.5}$$

To find a lower bound for $C'_{n,d}$ we will also estimate the variance of $\mathbf{B}'_{n,k,d}$. Recall

$$\mathrm{Var}[\mathbf{B}'_{n,k,d}] = \sum_{\beta_1,\beta_2} \Pr[\beta_1,\beta_2 \mathrm{active}] - \Pr[\beta_1 \mathrm{active}] \Pr[\beta_2 \mathrm{active}],$$

where β_1 and β_2 represent any possible ordered pair of k-blocks. The $n-k-d+1$ summands where $\beta_1 = \beta_2$ contribute $(n-k-d+1)(1/2^{k+1} - 1/2^{2k+2})$ in total. Disjoint pairs give no contribution since in this case β_1 and β_2 being active are independent events, meaning $\Pr[\beta_1, \beta_2\text{active}] = \Pr[\beta_1\text{active}]\Pr[\beta_2\text{active}]$. Pairs that overlap by more than d contribute a negative amount to the sum since $\Pr[\beta_1, \beta_2\text{active}] = 0$. On the other hand, $\Pr[\beta_1, \beta_2\text{active}]$ is $1/2^{2k+1}$ or 0 if they overlap by d or less. Given o is the number of bit positions shared by the two k-blocks, there are $n-k-d-o+1$ possible positions for β_1 and β_2. Summing o over all values from 1 to d gives less than $2d(n-2k-2d+1)$ places β_1 and β_2 might be. In all then, the overlapping k-blocks contribute no more than $2d(n-2k-2d+1)(1/2^{2k+1} - 1/2^{2k+2}) < d(n+1)/2^{2k+1}$. So

$$\mathrm{Var}[\mathbf{B'}_{n,k,d}] \leq (n-k-d+1)/2^{k+1} + d(n+1)/2^{2k+1}.$$

Applying (2.3) with Chebyshev's Inequality shows

$$\begin{aligned}\Pr[\mathbf{B'}_{n,k,d} = 0] &\leq \mathrm{Var}[\mathbf{B'}_{n,k,d}]/\mathrm{Ex}[\mathbf{B'}_{n,k,d}]^2 \\ &\leq 2^{k+1}/(n-k-d+1) + 4d(n+1)/(n-k-d+1)^2.\end{aligned}$$

Since $\Pr[\mathbf{C'}_{n,d} \geq k] = \Pr[\mathbf{B'}_{n,k,d} \geq 1] = 1 - \Pr[\mathbf{B'}_{n,k,d} = 0]$, we know that

$$\Pr[\mathbf{C'}_{n,d} \geq k] = \max\{0, 1 - 2^{k+1}/(n-k-d+1) + 4d(n+1)/(n-k-d+1)^2\}.$$

If $k + d \leq \log_2 n - 3$ then $n-k-d+1 \geq n/2$. In that case $4d(n+1)/(n-k-d+1)^2 \leq 16d/n + 16d/n^2$, which is less than $1/2$ for $n \geq 33d$. Similarly, $2^{k+1}/(n-k-d+1) \leq 2^{-d-1} < 1/2$. Since $C_{n,d} = \sum_{k\geq 1}\Pr[\mathbf{C}_{n,d} \geq k]$, we have

$$\begin{aligned}C'_{n,d} \geq &\sum_{1\leq k\leq \log_2 n-d-3} 1 - 2^{k+1}/(n-k-d+1) + 4d(n+1)/(n-k-d+1)^2 \\ &\geq \lfloor \log_2 n - d - 3\rfloor - 2^{-d-1} - 2.\end{aligned}$$

Assuming $n \geq 16d$ we then have

$$C'_{n,d} \geq \log_2 n - d - 7.$$

Every number has at least as long an active k-block as the longest full k-block in that number. This implies that the average $C_{n,d} \geq C'_{n,d}$. So we have

$$C_{n,d} \geq \log_2 n - d - 7.$$

Chapter 3

A Better Asymptotic Formula

3.1 The Objective

In this chapter I apply the analysis from [3] to the problem of finding a better asymptotic formula for $C_{n,d}$. Let $\mathbf{D}_\lambda$ be a random variable over the nonnegative integers such that

$$\Pr[\mathbf{D}_\lambda \geq k] = 1 - e^{\lambda/2^k}. \tag{3.1}$$

To reuse Pippenger's analysis we first must show that

$$\Pr[\mathbf{C}_{n,d} \geq k] = \Pr[\mathbf{D}_{n/2} \geq k] + O\left(\frac{(\log n)^3}{n}\right), \tag{3.2}$$

which will be done by a similar process as in his paper. From (2.4) we have the estimate

$$\Pr[\mathbf{C}_{n,d} \geq k] = O(n/2^k) \tag{3.3}$$

mirroring the estimate

$$\Pr[\mathbf{D}_\lambda \geq k] = O(\lambda/2^k) \tag{3.4}$$

which comes from (3.1) and the fact that as x approaches 0 we have $e^x = 1 + O(x)$. Now set

$$D_\lambda = \sum_{k \geq 1} \Pr[\mathbf{D}_\lambda \geq k].$$

For $k \leq 3\log_2 n$ we apply the estimate (3.3) and for $k > 3\log_2 n$ we apply (3.4) to (3.2) showing that

$$C_{n,d} = D_{n/2} + O\left(\frac{(\log n)^3}{n}\right). \tag{3.5}$$

3.2 The Recurrence

Let $q_{n,k,d}$ be the probability that a uniformly distributed random n-bit number has no full active k-blocks with regard to multiplication by $1 + 2^d$. This definition allows a truncated k-block to exist in the number.

So $q_{n-1,k,d}$ is the probability that there is no active k-block in the $n - 1$ lowest bits of a n-bit multiplication. This event implies that there is either no full active k-blocks in all n bits, which has probability $q_{n,k,d}/$ of occuring, or there is a full active k-block in the leftmost possible position and no other active k-blocks, which has probability $q_{n-k-d,k,d}/2^k + 1$ of occuring. Thus we have for $n \geq k + d$ that

$$q_{n,k,d} = q_{n-1,k,d} - q_{n-k-d,k,d}/2^{k+1}. \tag{3.6}$$

For $n < k + d$ notice that an n bit number does not contain enough bits to house a full active k-blocks. So $q_{n,k,d} = 1$ in this case.

3.3 The Generating Function and Its Roots

Consider the generating function

$$Q_{k,d}(z) = \sum_{n \geq 0} q_{n,k,d} z^n.$$

Then (3.6) shows after multiplying by z^n, summing over $n \geq k + d$ and then adding $\sum_{0 \leq n < k+d} q_{n,k,d} z^n$ to both sides that

$$Q_{k,d}(z) = zQ_{k,d}(z) + z^{k+d} Q_{k,d}(z)/2^{k+1} + 1.$$

After further formula manipulation, this leads to

$$Q_{k,d}(z) = \frac{1}{P_{k,d}(z)},$$

where

$$P_{k,d}(z) = 1 - z + z^{k+d}/2^{k+1}. \tag{3.7}$$

We have singled out $P_{k,d}(z)$ from the equation since the asymptotic behavior of the generating function is determined by the zeroes of this polynomial.

For $0 \leq j \leq 1$, $P_{k,d}(j) > 0$ and for k large enough, $P_{k,d}(1 + 1/k) < 0$, so there exists a root $\zeta = 1 + O(1/k)$. Writing $\zeta = 1 + \epsilon$, we have from (3.7) that

$$\epsilon = (1 + \epsilon)^{k+d}/2^{k+1} = \exp((k + d)\log(1 + \epsilon) - (k + 1)\log 2). \tag{3.8}$$

Since $\log(1+\epsilon) = O(\epsilon)$, letting $\epsilon = O(1/k)$ in (3.8) produces the better estimate $\epsilon = O(1/2^k)$. Substituting in this new formula for ϵ shows $\epsilon = 1/2^{k+1} + O\left(\frac{k}{2^{2k}}\right)$ by utilizing the fact that $\exp(x) = 1 + O(x)$ for x approaching 0. We now have

$$\zeta = 1 + 1/2^{k+1} + O\left(\frac{k}{2^{2k}}\right).$$

The remaining roots of $P_{k,d}(z)$ are also the roots of $2^{k+1}P_{k,d}(z)$ divided by $z - \zeta$ which we write as

$$E_{k,d}(z) = z^{k+d-1} + \zeta z^{k+d-2} + ... + \zeta^{k+d-2}z + \zeta^{k+d-1} - 2^{k+1}.$$

For k large enough, we have $\zeta \leq 3/2$. Suppose as well that $|z| \leq 2^{\frac{k+1}{k+d-1}}$ and for simplicity let us label $2^{\frac{k+1}{k+d-1}}$ as c. We then have

$$\begin{aligned} |z^{k+d-1} & +\zeta z^{k+d-2} + ... + \zeta^{k+d-2}z + \zeta^{k+d-1}| \\ & \leq c^{k+d-1}\left(1 + \zeta/c + ... + (\zeta/c)^{k+d-2} + (\zeta/c)^{k+d-1}\right) \\ & = 2^{k+1}(1 - (\zeta/c)^{k+d})/(1 - \zeta/c) \\ & \leq 2^{k+1}. \end{aligned}$$

This implies $E_{k,d}(z)$ is positive for $|z| \leq c$ and so there are no roots within that circle. If the roots of $P_{k,d}(z)$ are $\zeta_1, ..., \zeta_{k+d}$ and ζ_1 is ζ we know $|\zeta_j| \geq c$ for all $2 \leq j \leq k+d$.

Now note that any root of $P_{k,d}(z)$ with multiplicity greater than one must also be a root of the derivative $P'_{k,d}(z) = -1 + (k+d)z^{k+d-1}/2^{k+1}$. But any root ζ_p of $P'_{k,d}(z)$ must have the property

$$|\zeta_p| = \left(2^{k+1}/(k+d)\right)^{1/(k+d-1)} \leq c.$$

The only root of $P_{k,d}(z)$ that exists in that region is the simple root ζ_1. Thus every root of $P_{k,d}(z)$ is simple.

We may express the generating function as

$$Q_{k,d}(z) = \sum_{1 \leq j \leq k+d} \frac{-1}{(1 - z/\zeta_j)\zeta_j P'_{k,d}(\zeta_j)}$$

because $P_{k,d}(z)$ has only simple roots. This means that

$$\begin{aligned} q_{n,k,d} &= \sum_{1 \leq j \leq k+d} \frac{-1}{\zeta_j^{n+1} P'_{k,d}(\zeta_j)} \\ &= \frac{1}{\zeta_j^{n+1}(1 - (k+d)\zeta_j^{k+d-1}/2^{k+1})}. \end{aligned}$$

The $j = 1$ term is

$$\frac{1}{\zeta^{n+1}(1-(k+d)\zeta^{k+d-1}/2^{k+1})} = \frac{1}{(1+1/2^{k+1}+O(k/2^{2k}))^n\,(1+O(k/2^k))},$$

by employing the estimates for the first root we found earlier. Since all roots besides the first have absolute value greater than c, we know for $j > 1$ that $|1-(k+d)\zeta_j^{k+d-1}/2^{k+1}| \geq k+d-1$. So we may express every term where $j > 1$ in the sum as

$$\frac{1}{\zeta_j^{n+1}(1-(k+d)\zeta_j^{k+d-1}/2^{k+1})} = O\left(\frac{1}{c^n(k+d)}\right).$$

The $j > 1$ terms sum together into $O\left(1/c^n\right)$. In total the sum gives us that

$$q_{n,k,d} = \frac{1}{(1+1/2^{k+1}+O(k/2^2k))^n\,(1+O(k/2^k))} + O\left(\frac{1}{c^n}\right).$$

Since $q_{n,k,d}$ represents the probability that there is no full active k-block,

$$\begin{aligned}\Pr[\mathbf{C}_{n,d} \geq k] &= 1 - q_{n,k,d} + \Pr[\text{some truncated } k-\text{block is active}] \\ &= 1 - q_{n,k,d} + O\left(\frac{1}{2^k}\right).\end{aligned}$$

So if $1 \leq k \leq \log_2 n - \log_2(4\log_2 n)$, we get by substituting for $q_{n,k,d}$ that

$$\Pr[\mathbf{C}_{n,d} \geq k] = 1 + O\left(\frac{1}{n^2}\right).$$

For the same values of k equation (3.1) gives us that

$$\Pr[\mathbf{D}_n \geq k] = 1 + O\left(\frac{1}{n^2}\right).$$

So (3.2) holds for these values of k. If $k > \log_2 n - \log_2(4\log_2 n)$ then

$$\Pr[\mathbf{C}_{n,d} \geq k] = 1 - e^{n/2^{k+1}}\left(1 + O\left(\frac{nk}{2^{2k}}\right)\right),$$

which means (3.2) holds for all k. Then (3.2) being true implies (3.5) holds by the arguments from the beginning of this chapter.

3.4 The Asymptotic Formula

Equation (3.5) is an exact mirror of the formula found in [3] for average length of the longest active k-block when adding two random n-bit numbers. By virtue of the analysis in that paper of D_λ we already know that

$$C_{n,d} = \log_2 n + \gamma \log_2 e - \frac{3}{2} - F(\log_2 n) + O\left(\frac{(\log n)^4}{n}\right) \tag{3.9}$$

Here

$$F(v) = \int_0^\infty \left(\{v - \log_2 y\} - \frac{1}{2}\right) e^{-y} \mathrm{d}y$$

with $\{x\} = x - \lfloor x \rfloor$ being the fractional part of x. This function is oscillating but by a miniscule magnitude. In fact, the function is bound by $|F(v)| \leq 1.573... \times 10^{-6}$.

Chapter 4

An Exact Formula

This chapter proves an exact formula for $C_{n,1}$ by a direct combinatorial argument. Although we can extend this combinatorial arguments to find a formula for $C_{n,d}$ with constant d greater than 1, the arguments become overly convoluted for d larger than one. So this chapter will also detail a generating function method for finding $C_{n,d}$ that is more manageable for large values of d. The direct combinatorial proof for the $d = 1$ case relies highly on the proof given in [3] of the similar formula for C_n, the mean length of longest active k-block when adding two independently chosen unformly random n-bit numbers. This section exposes the equation for C_n using methods from Pippenger's article

4.1 Finding C_n

First see that

$$C_n = \sum_{k\geq 0} k \Pr[\mathbf{C}_n = k] = \sum_{k\geq 1} \Pr[\mathbf{C}_n \geq k] \tag{4.1}$$

because of partial summation. A carry propagation chain of length k occurs if and only is some k block is active, and so

$$\Pr[\mathbf{C}_n \geq k] = \Pr[\text{some } k\text{-block is active}].$$

Then 4.1 implies

$$C_n = \sum_{k\geq 1} \Pr[\text{some } k\text{-block is active}]. \tag{4.2}$$

Using an inclusion-exclusion argument, we have

$$\Pr[\text{some } k\text{-block is active}] = \sum_{j\geq 1} (-1)^{j-1} \sum_{\beta_1,\ldots,\beta_j} \Pr[\beta_1, ..., \beta_j \text{active}]. \tag{4.3}$$

Here the inner sum is indexed by all unordered sets $\{\beta_1, ..., \beta_j\}$ of distinct k-blocks.

If two k-blocks are active they must be disjoint. Also the events of two disjoint k-blocks being active are independant. These facts allow us to rewrite (4.3) as

$$\Pr[\text{some } k\text{-block is active}] = \sum_{j\geq 1}(-1)^{j-1}\sum_{\beta_1,...,\beta_j}\Pr[\beta_1\text{active}]...[\beta_j\text{active}], \tag{4.4}$$

where $\beta_1, ..., \beta_j$ are pairwise disjoint k-blocks.

We have shown when we first defined active k-blocks that $\Pr[\beta_i \text{ active}] = 1/2^{k+1}$. So every term of the inner sum in (4.4) is $1/2^{j(k+1)}$. By collecting all of these terms we have

$$\Pr[\text{some } k\text{-block is active}] = \sum_{j\geq 1} A_{n,j,k}\frac{(-1)^{j-1}}{2^{j(k+1)}}, \tag{4.5}$$

with $A_{n,j,k}$ being the number of ways to place j distinct k-blocks among n bits. To place these blocks, we only need to consider placing the rightmost bit of each block. The other $k-1$ bits in each block are positions where we may not place another active k-block. So we remove $k-1$ potential bit positions where we may choose to have an active k-block's first bits for each active k-block in our number. Then the number of ways to place the heads is

$$A_{n,j,k} = \binom{n-j(k-1)}{j}.$$

Substituting this back into (4.5) gives

$$\Pr\text{ some } k\text{-block is active}] = \sum_{j\geq 1}\binom{n-j(k-1)}{j}\frac{(-1)^{j-1}}{2^{j(k+1)}}, \tag{4.6}$$

and after using this form for Pr[some k-block is active] in 4.2 we have

$$C_n = \sum_{k\geq 1}\sum_{j\geq 1}\binom{n-j(k-1)}{j}\frac{(-1)^{j-1}}{2^{j(k+1)}}. \tag{4.7}$$

4.2 A Combinatorial Solution for $C_{n,1}$

To appropriate the last proof about C_n's value to finding $C_{n,1}$, first we must notice the differences between active k-blocks in the two cases. First for

$C_{n,1}$ active k-blocks require one more bit. There may also be truncated active k-blocks, which require fewer bits and may only exist at the end of the number. Having more effect on our analysis though is the fact that the probability that two k-blocks are active is not independrnt with regar to this or any multiplication of the form we are examining. To see this consider the number 10101101011. The first and last 6 digits are an active 5-block with regard to multiplication by 3, overlapping on their lowest and highest bit positions respectively. Due to the form of active k-blocks under multiplication by 3, that is ...01011, active k-blocks may only overlap by one bit and when k is odd.

Even with these differences, we can use the same arguments from the last section almost verbatim to show for all positive d that

$$C_{n,d} = \sum_{k \geq 1} \Pr[\text{some } k\text{-block is active}] \tag{4.8}$$

for any multiplication $1 + 2^d$. We also still have that

$$\Pr[\text{some } k\text{-block is active}] = \sum_{j \geq 1} (-1)^{j-1} \sum_{\beta_1, \ldots, \beta_j} \Pr[\beta_1, \ldots, \beta_j \text{active}]. \tag{4.9}$$

Combining these two formulas, we may split the sum in (4.8) for k even and k odd to produce

$$\begin{aligned} C_{n,d} = & \sum_{k \geq 1,\ k \text{ odd}} \sum_{j \geq 1} (-1)^{j-1} \sum_{\beta_1, \ldots, \beta_j} \Pr[\beta_1, \ldots, \beta_j \text{active}] \\ & + \sum_{k \geq 1,\ k \text{ even}} \sum_{j \geq 1} (-1)^{j-1} \sum_{\beta_1, \ldots, \beta_j} \Pr[\beta_1, \ldots, \beta_j \text{active}] \end{aligned} \tag{4.10}$$

Notice the inner sum of (4.10) is over all unordered set of j distinct k-blocks.

4.2.1 The Even Case

When k is even then two overlapping k-blocks cannot both be active and two distinct k-blocks being active are still independent events. So the arguments leading to (4.6) still apply almost exactly. One change now is since our full k-blocks are $k+1$ bits long, we must remove k digits for each full active k-block from our list of possible starting bits for an active k-block. So we have that if each β_i is a full k-block that

$$\sum_{\beta_1, \ldots, \beta_j} \Pr[\beta_1, \ldots, \beta_j \text{active}] = \binom{n - jk}{j} \frac{1}{2^{j(k+1)}}.$$

Now suppose some β_i is a truncated k-block. We know this happens with probability $1/2^k$ In that case we also know no other active k-block may include the last k digits of the number. So, since we must merely count all the ways to place other active k-blocks and the probability those k-blocks are all active, we have

$$\sum_{\beta_1,\ldots,\beta_j} \Pr[\beta_1, ..., \beta_j \text{active}] = \binom{n-jk}{j-1} \frac{1}{2^{(j-1)(k+1)+k}}$$

if some β_i is a truncated k-block. The value for $\sum_{\beta_1,\ldots,\beta_j} \Pr[\beta_1, ..., \beta_j\text{active}]$ with no restrictions on whether there exists some β_i that is truncated is merely the sum $\sum_{\beta_1,\ldots,\beta_j} \Pr[\beta_1, ..., \beta_j\text{active}]$ when each β_i must be full, and $\sum_{\beta_1,\ldots,\beta_j} \Pr[\beta_1, ..., \beta_j\text{active}]$ given some β_i is truncated. Thus, for k even we conclude that

$$\begin{aligned} &\sum_{j\geq 1} (-1)^{j-1} \sum_{\beta_1,\ldots,\beta_j} \Pr[\beta_1, ..., \beta_j\text{active}] \qquad (4.11) \\ &\qquad = \sum_{j\geq 1} \binom{n-jk}{j} \frac{(-1)^{j-1}}{2^{j(k+1)}} + \binom{n-jk}{j-1} \frac{(-1)^{j-1}}{2^{(j-1)(k+1)+k}}. \end{aligned}$$

4.2.2 The Odd Case

For odd k, we may still tackle the problem in a similar way. To find $\Pr[\beta_1, ..., \beta_j\text{active}]$ first suppose $l \geq 0$ where l represents the number of bits where successive active k-blocks overlap. Then

$$\Pr[\beta_1, ..., \beta_j\text{active}] = 1/2^{j(k+1)-l}, \qquad (4.12)$$

since the number of bits that must be specific values is now $k+1$ for each active k-block, minus l to account for bits we double counted. Summing over all combinations of j active k-blocks is equivalent to counting the number of possible placements of j active k-blocks which have l total overlapping digits, summed over all l. So we have

$$\sum_{\beta_1,\ldots,\beta_j} \Pr[\beta_1, ..., \beta_j\text{active}] = \sum_{l\geq 0} A_{n,j,k,l}/2^{j(k+1)-l}. \qquad (4.13)$$

Here $A_{n,j,k,l}$ is the number of ways to arrange j active k-blocks in an n-bit number such that there are l bit positions occupied by more than one k-block.

To determine $A_{n,j,k,l}$, define a set of active k-blocks $\{\beta_1, ..., \beta_i\}$ as a *body* if it abides by the following rules. For $0 < h < i$, β_h and $\beta_h + 1$ must share a bit position and any k-block in the body shares bit positions only with other k-blocks in the same body. We refer to the rightmost k-block of a body as the *head*. All other k-blocks in the body are known as *tails*. So 10101101011 is a body composed of active 5-blocks where the first 6 bits are its head and the last 6 are its tail.

Now determining $A_{n,j,k,l}$ is as simple as choosing where to place the head k-blocks and counting the ways to attach tails onto these heads. To place the heads, we need simply choose where the rightmost bit goes. In total there are jk positions we may not place these bits since the rightmost digit of each active k-block is either also the leftmost digit of another active k-block or is one of the digits we wish to place. So even though each of the j active k-blocks is $k+1$ digits long, we may subtract one digit for each active k-block from our count of positions we may not place the rightmost digit of a head. The number of heads is $j - l$; in other words the number of active k-blocks that are not tails since l also counts number of tails. So there are $\binom{n-jk}{j-l}$ ways of positioning each head. Since we may place however many tails onto each head, the number of ways to do this is $\left(\!\!\binom{j-l}{l}\!\!\right)$. So

$$A_{n,j,k,l} = \binom{n-jk}{j-l}\left(\!\!\binom{j-l}{l}\!\!\right),$$

which along with (4.13) gives

$$\sum_{\beta_1,...,\beta_j} \Pr[\beta_1, ..., \beta_j \text{active}] = \sum_{l \geq 0} \binom{n-jk}{j-l}\left(\!\!\binom{j-l}{l}\!\!\right) / 2^{j(k+1)-l}. \tag{4.14}$$

We now use (4.11) and (4.14) to substitute in the sums over odd and even k in (4.10), arriving at our exact expression for $C_{n,1}$,

$$\begin{aligned} C_{n,1} = & \sum_{k>0,\, k \text{ even}} \sum_{j \geq 1} \binom{n-jk}{j} \frac{(-1)^{j-1}}{2^{j(k+1)}} + \binom{n-jk}{j-1} \frac{1}{2^{(j-1)(k+1)+k}} \\ & + \sum_{k>0,\, k \text{ odd}} \sum_{j \geq 1} \sum_{l \geq 0} (-1)^{j-1} \binom{n-jk}{j-l}\left(\!\!\binom{j-l}{l}\!\!\right) / 2^{j(k+1)-l}. \end{aligned} \tag{4.15}$$

From (4.15) we see what is most likely the simplest formula for any $C_{n,d}$. Clearly the difficulty in calculating such values necessitates the simpler although not exact formulas for $C_{n,d}$ given in the previous sections.

4.3 Finding Any $C_{n,d}$

4.3.1 Split Up the Problem

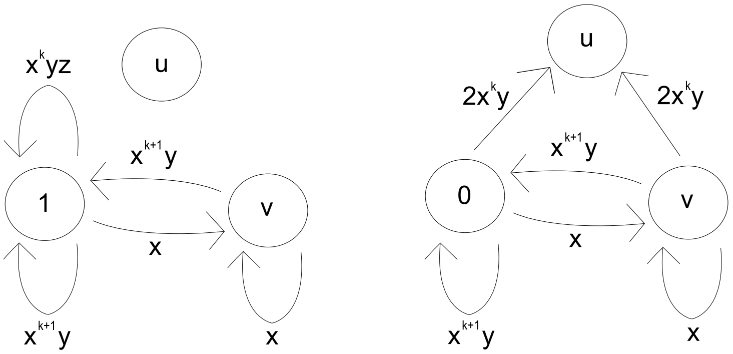
Our goal is to show the methods by which an exact expression for $C_{n,d}$ may be found. To do this, consider the $2d$ possible cases $k = 1 \pmod{2d}$, ..., $k = 2d \pmod{2d}$ seperately. Why? The value of $k \bmod 2d$ determines how active k-blocks may overlap and how many truncated active k-blocks there may be. For example when $d = 1$ we saw that depending on whether k was even or odd, active k-blocks may or may not overlap with one another. Combining (4.8) and (4.9) while grouping together values of k that are in the same congruence class gives

$$C_{n,d} = \sum_{l=1}^{2d} \sum_{k\geq 1, k\equiv_{2d} l} \sum_{j\geq 1} \sum_{\beta_1,\ldots,\beta_j} (-1)^{j-1} \Pr[\beta_1, ..., \beta_j \text{active}]. \tag{4.16}$$

4.3.2 Overlap Graphs

First notice that with regard to multiplication by $1 + 2^d$, up to d bits may be shared by two active k-blocks. This is because, if two active k-block overlap by some digits, the left active k-block will have 1's in its first and $(d+1)^{\text{th}}$ positions, to generate a carry. So if the right active k-block overlaps by more than d positions, it will generate a carry where it should propagate a carry. This fact allows us to see that each bit of a number multiplied by $1 + 2^d$ is in at most 2 active k-blocks, because active k-blocks are $k + d$ bits long.

For $k = l \pmod{2d}$, we may represent all the possible ways to arrange k-blocks among n bits using a directed graph with weighted edges. Vertices represents active k-blocks ending on the left with a specific d bits and single bits that are not necessarily in any active k-blocks. An edge to a vertex represents a way we may append that active k-block or a single bit to the left of some block of bits. The weights, of the form $2^e x^a y^b z^c$, give information on how these digits are added as follows. Here e is the number of bits of the appended active k-block that are not contained in the original binary representation of the number. So e is nonzero only for edges that represent appending a truncated active k-block. The power of x is number of new bits we gain by appending those bits. The value b is the number of new active k-blocks. Finally, c is the number of bits of the new active k-block that already were in the original block of bits, or 0 if we did not append a k-block.



(a) Overlap graph for odd k.

(b) Overlap graph for even k.

Figure 4.1: Overlap graphs corresponding to $C_{n,1}$.

We call such a graph an *overlap graph* and we can make one in the following manner. Create 2^{d-1} vertices and label each uniquely with the d bits a k-block may end in. Note one of those bits is a fixed value, and so we need only 2^{d-1} vertices. Let one vertex called v represent bit positions that are not within any active k-blocks. Every vertex has an edge to all vertices other than v of weight $x^{k+d}y$, and every vertex has an edge to v of weight x. Suppose vertices x and y correspond to active k-blocks ending in 01 and 11 respectively. Also imagine we may overlap an active k-block ending in 11 onto the left-hand side of an active k-block ending in 01. If the amount of overlap between the two active k-blocks is o bits, then our graph would have a directed edge from x to y with weight $x^{k+d-o}yz^o$.

Finally create a vertex u that represents a truncated active k-block. Since we may have no more bits after forming a truncated active k-block, there exists no edges from u to any other vertex in the graph. If any other vertex in the graph s has an edge of weight $x^{k+d}yz^c$ to a vertex whose label has at least e 0-bits from the leftmost end, we add an edge from s to u of weight $2^e x^{k+d-e}yz^c$. This corresponds to the fact that adding this truncated k-block with c bits not in the actual number is possible and requires c less bits than if this were a full k-block.

4.3.3 Deriving Formulas from the Graphs

Let $V(p)$ be the product of the weights of all edges along some path p. For p starting at v and $V(p) = x^n y^j z^o$, p represents an n bit number built by adding active k-blocks and single bits in the order of vertices visited by p after the first vertex v. This number has at least j active k-blocks and o digits shared among successive active k-blocks. Path p also uniquely represents how we may place those j active k-blocks in the number. Notice as well the overlap graph gives all ways in which we may create n bit numbers with at least j active k-blocks from all possible $V(p)$ with p starting at v.

Substituting $y = -2^{-(k+d)}$ and $z = 2$, then if there exists an edge from vertex s to t with weight px^a this then means that there is probability $|p|$ appending a random bits to the left of the bits that s represents creates the bits that t represents. Here p is negative if and only if t represents an active k-block. This is true since there are $k+d$ bits in an active k-block, and so appending an active k-block to the left of some number requires that $k+d$ bits are set to certain values. The bits of the appended active k-block that were also in the original number we do not need to set though, and we also do not need to set zeroes at the end of a truncated active k-block that will not be in the binary representaion of the number.

So we may simplify the innermost two sums from (4.16) as

$$\sum_{j\geq 1}(-1)^{j-1}\sum_{\beta_1,\ldots,\beta_j}\Pr[\beta_1,\ldots,\beta_j\text{active}] = -[x^n]\sum_p V(p).$$

Note the second sum is over all paths p that start with v.

Let T be a transition matrix of the graph such that the first row corresponds to vertex v. T^m then contains in the i^{th} row and j^{th} column the sum of $V(p)$ over all paths p that start at the vertex represented by the i^{th} row and end at the j^{th} column vertex. Then since $(I-T)^{-1} = \sum_{m\geq 0} T^m$, the first row contains the weights of all paths that start at v and end at any vertex. This implies

$$\text{Pr some } k\text{-block is active}] = [x^n][100\ldots0](I-T)^{-1}\begin{bmatrix}1\\1\\\vdots\\1\end{bmatrix},$$

since all n digit numbers are represented uniquely by a path from v to any other vertex in the graph.

4.3.4 The $C_{n,1}$ Example

From the graph (4.1a), when k is odd we have

$$T = \begin{bmatrix} x & x^{k+1}y & 2x^k y \\ x & x^{k+1}y & 2x^k y \\ 0 & 0 & 0 \end{bmatrix}.$$

So

$$\Pr[\text{some k-block is active}] = [x^n]\frac{1+2yx^k}{1-x-x^{k+1}y}.$$

Note $\frac{1}{1-x-x^{k+1}y} = \sum_{m\geq 0}(x+x^{k+1}y)^m$. This then implies

$$\begin{aligned}\Pr[\text{some } k\text{-block is active}] = &\sum_{n=c_1+c_2(k+1)} \binom{c_1+c_2}{c_2}(-1)^{c_2-1}/2^{(k+1)c_2} \\ &- 2^{-k} \sum_{n-k=c_1+c_2(k+1)} \binom{c_1+c_2}{c_2}(-1)^{c_2-1}/2^{(k+1)c_2}\end{aligned}$$

We may rewrite this sum in the same form as given by (4.11) found by the direct combinatorial approach. Finding the exact formula for odd k the same way then leads to a formula for $C_{n,1}$.

4.3.5 The Program

The method above for finding exact expressions for $C_{n,d}$ requires creating $2d$ overlap graphs each with $2^{d-1}+2$ vertices, and then placing an edge potentially between every pair of vertices. This is extremely tedious if done by hand, and one can easily create an error in the overlap graph that would be difficult to notice. For this reason, I have created a two functions in Mathematica to handle all of this. One creates a transition matrix if you specify k mod $2d$ and d for that overlap graph. This information is useful to examine the structure of the overlap graph. The other function takes the same input and outputs the generating function for the probability that such a k-block is active. This generating function is always in the form of a fraction with polynomial numerator and denominator. So we can always easily extract from these generating functions exact expressions for the probability a k-block is active. We may then apply (4.16) to arrive at the expression for $C_{n,d}$.

Here are the program for finding the transition matrix.

```
gettransitionmatrix[k_, d_] := (
    (*gettransitionmatrix takes k of values 1 through 2d of k
mod 2d, and it takes d which corresponds to d as
defined in my thesis. It then outputs the transition
matrix of the overlap graph for those values.*)

    (*Trans becomes the transition matrix we desire.
        lastdigits takes on the value of all possible bits in
the last d positions.
        We get all possible values for the last d digits
from i, which equal to all possible base 10
representations of lastdigits. Note one of the
  last d digits must be a specific value since the
d + 1 digit must be a 1,
        the 2d + 1 digit must be a 0 and so on.*)

    Trans = IdentityMatrix[2^(d - 1) + 2]  *0;
    For[ i = 0, i < 2^(d - 1), i++, lastdigits =
    IntegerDigits[i, 2];

      While[Length[lastdigits] < d - 1, lastdigits =
      Prepend[lastdigits, 0]];
      If[ k < d + 1, lastdigits = Insert[lastdigits, 1,
Mod[k - 1, d] + 1], lastdigits = Insert[lastdigits,
0, Mod[k - 1, d] + 1]];

      (*append represents the last d digits of an active
k-block that shares p digits with the k - block ending
with lastdigits.*)

      For[ p = 1, p <= d, p++,
        If[lastdigits[[p]] == 1,
          append = Table[3, {d}];
      If[k = d, append[[Mod[k - 1, d] + 1]] = 1,
        append[[Mod[k - 1, d] + 1]] = 0];

          (*j represents a number of digits to the left of the
              pth digit of lastdigits. We use the digit digit
              at the p - j position in lastdigits to set the
      appropriate digit in append*)

          For[j = 1, j < p, j++, placer = Mod[k - j, 2d];
             If[placer < d , If[placer != 0,
                append[[Mod[k - j,
                    d]]] = Mod[lastdigits[[p - j]] + 1, 2],
append[[d]] = lastdigits[[p - j]]],
If[placer != d,  append[[Mod[k -
```

```
                 j, d]]] = lastdigits[[p - j]],
         append[[d]] = Mod[lastdigits[[p - j]] + 1, 2]]]];

         (*Some digits
             in append still are not set to 1 or 0. These digits
     may be replaced by any set of 1’s and 0’s.
     freedigitslist is a list of bits we will replace
             those not-set bits in append with. freedigitslist
     is determined by the base 10 number freedigits,
             which takes on values so that we end up replacing
             those not set bits with all possible lists of 1’s
     and 0’s of the correct size.*)

          For[freedigits = 0, freedigits < 2^Count[append, 3],
           freedigits = freedigits + 1,
           freedigitslist = IntegerDigits[freedigits, 2];
           While[Length[freedigitslist] < Count[append, 3]  ,
             freedigitslist = Prepend[
             freedigitslist, 0]]; replacelist =
     Position[append, 3];
           nofreeappend = append;

           (*replacelist is a
                list of all positions in append that have not
been given set values. replacecounter tells us
                how many of those digits have already been
                replaced. nofreeappend becomes
                append except all digits are now 1 or 0.*)

           For[replacecounter = 1,
           replacecounter
             = Count[append, 3], replacecounter++,
           freeposition = replacelist[[replacecounter]][[1]];
             nofreeappend[[freeposition]] =
               freedigitslist[[replacecounter]]];

           (*After doing all this, we find that a k - block
               ending in lastdigits may have a k - block
               ending in append to the left of it and
               sharing p of its bits. After every
```

```
                iteration, we will know all ways two
                k-blocks may overlap. This allows us to put
                appropriate entries into
                Trans.*)

            Trans = ReplacePart[Trans, x^(kv + d -p)*y*z^p +
            Trans[[i + 1]][[1 + FromDigits[Delete[nofreeappend,
              1 + Mod[k - 1, d]], 2]]], {i + 1,
                1 + FromDigits[Delete[nofreeappend, 1 +
                   Mod[k - 1, d]], 2]}];

            (*zeroesatend counts the number of zeroes at the
end of nofreeappend. This info is used to
                edit Trans so we include edges to u*)

            zeroesatend = 0;
            While[nofreeappend[[1]] == 0,
              zeroesatend++;
              Trans = ReplacePart[Trans,
                  2^zeroesatend*x^(kv + d - p - zeroesatend)
                   *y*z^p+ Trans[[1 + i]][[2^(d - 1) + 2]],
                  { 1 + i, 2^(d - 1) + 2} ];
              nofreeappend = Delete[nofreeappend, 1];
              ];
            ]]]];

    (*We now add to Trans entries corresponding to edges not
        having to do with overlapping k - blocks. singlebits,
        ro and col are all indices to help us add these
        entries everywhere necessay.*)

    For[singlebits = 1, singlebits < 2^(d -
                1) + 2, singlebits++, Trans = ReplacePart[
                  Trans, x^(kv + d)y, { 2^(d - 1) + 1,
                  singlebits}];
      Trans = ReplacePart[Trans, x, { singlebits,
        2^(d - 1) + 1}]];
    For[ro = 1, ro = 2^(d - 1), ro++, For[col = 1,
     col = 2^(d - 1), col++,
     Trans = ReplacePart[Trans, x^(kv + d)y +
       Trans[[ro]][[col]],
     {ro, col}]]];
Trans)
```

The next function outputs the generating function for the probability that such a k-block is active. Note this function calles `gettransitionmatrix`.

```
getgenfunction[k_, d_] := (
    (*getgenfunction takes k of values 1 through 2d of k mod 2d, and
          it takes d which corresponds to d as defined in my thesis.
          It then outputs the generating function
          for the probability such a k - block is active.*)

    (*genfunction will become the generating function, and allpaths
        is the inverse of I minus the transition matrix.*)

    genfunction = 0;
    allpaths =
      Inverse[IdentityMatrix[d + 2] - gettransitionmatrix[k, d]];

    (*We create the generating function by summing over all the
        columns in the 2^(d - 1) + 1, since this row corresponds
        to vertex v.*)

    For[col = 1, col = 2^(d - 1) + 2, col++,
      genfunction = genfunction +
        allpaths[[2^(d - 1) + 1]][[col]]];
    Simplify[genfunction])
```

Even though these functions would take time exponential in d to run, they are more reliable and faster than calculations by hand.

Chapter 5

Conclusion

My thesis sought expressions for $C_{n,d}$, the average length of the longest active k-block with regard to multiplaction by $1+2^d$ in a random n bit number. This value directly related to the average time necessary to multiply an n bit number by $1+2^d$ using local parallel computation. First we found upper and lower bounds for $C_{n,d}$ using first- and second-moment probabilistic arguments. These gave the asymptotic formula

$$C_{n,d} = \log_2 n + O(1).$$

Then by finding a recurrence that described the probability that there does not exist full active k-blocks, we discovered the even tighter asymptotic formula

$$C_n = \log_2 n + \gamma \log_2 e - \frac{3}{2} - F(\log_2 n) + O\left(\frac{(\log n)^4}{n}\right). \tag{5.1}$$

Finally we created a method that allows us to derive exact formula for $C_{n,d}$.

These results are all interesting since they answer with varying detail a basic question concerning a basic algorithm. These results also allow comparisons of varying precision between the runtime of the addition algorithm we considered and other algorithms. They can be used to derive running time of algorithms that multiply by $1+2^d$ as well. Perhaps most usefully, the analyses used in my thesis are applicable to problems besides the one considered. Different problems concerning average running time of other special cases or variants of the this algorithm may rely on the work done here, as I relied on my sources [1] and [3]. Even more generally though, this analysis can be used to determine expressions for average length of the longest appearence of certain patterns in random n bit numbers. The need to find such an average could arise in problems from many different fields, and so we see utility for the work accomplished.

Bibliography

[1] Knuth, Donald E. “The Average Time for Carry Propagation.” *Selected Papers on Analysis of Algorithms.* Stanford: Leland Stanford Junior University, 2000.

[2] Knuth, Donald E.“The Average Time for Carry Propagation.” *Nederl. Akad. Wettensch. Indag. Math.* 40 (1978): 238-242.

[3] Pippenger, Nicholas. “Analysis of Carry Propagation: An Elementary Approach.” *Journal of Algorithms.* 42 (2002): 317-333.

[4] Wolfram, Stephen. *The Mathematica Book.* Fourth ed. Wolfram Media/Cambridge University Press, 1999.