UNM Physics 452/581: Introduction to Quantum Information, Solution Set 4, Fall 2007

4.1 ΘΩO frat initiation

- (a) First apply Stirling's approximation $n! \approx \sqrt{2\pi n}(n/e)^n$ to rewrite the last four functions. We find that

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \tag{1}$$

$$\binom{2n}{n} = \frac{4^n}{\sqrt{n\pi}} \tag{2}$$

$$\ln(n!) = \ln\sqrt{2\pi} + (n+\frac{1}{2})\ln n - n \tag{3}$$

$$\ln\binom{2n}{n} = n\ln 4 - \frac{1}{2}\ln n\pi \tag{4}$$

The final ordering is

$$n = \ln\binom{2n}{n} \leq \ln n! = n\log_2 n \leq n^{\ln n} \leq 2^n \leq \binom{2n}{n} \leq (\ln n)^n \leq (\log_2 n)^n \leq n! \tag{5}$$

Rather than providing explicit constants, the following discussion attempts to categorize the classes of asymptotic growth.

  - n (Linear Growth): We see that n and $\ln\binom{2n}{n}$ are to highest order linear in n. While $\ln\binom{2n}{n}$ does have the $-\frac{1}{2}\ln n\pi$ component, it is dominated by $n\ln 4$ for large n. Constants can be easily chosen to account for the $\ln 4$ factor difference allowing us bound either function by the other.
  - $n\log n$ (Log-Linear Growth): For large n, $\ln n!$ and $n\log_2 n$ are dominated by terms of the form $n\log_b n$. Logarithms of different bases are related by $\log_a b = \frac{\log_c b}{\log_c a}$, indicating that they are equivalent under the asymptotic notations (since we can always multiply by $\log_c a$). It is also readily seen that these functions are not in the same class as the linear ones, as we cannot find a constant k such that

$$kn \geq n\log n \tag{6}$$

$$k \geq \log n \tag{7}$$

    for all $n > n_0$.
  - $n^{\ln n}$ (Exponential Logarithmic Growth): Due to the exponential, it is clear that this function grows faster than any linear or log-linear one. Since the term in the exponent is a logarithm (which grows more slowly than a linear function), this function grows more slowly than the next class of functions.
  - c^n (Constant Exponential Growth): Although $2^n \leq \binom{2n}{n}$, the two functions have similar asymptotic forms. We note that $\binom{2n}{n} \approx 4^n = (2^2)^n = 2^{2n}$, so this function will grow faster than 2^n.

- $(\log_b n)^n$ (Logarithmic Exponential Growth): The next two functions are similar, but not in the same class. We have that

$$(\log_2 n)^n = (\ln 2)^{-n} (\ln n)^n \tag{8}$$

Since $\ln 2 < 1$, we see that the $\log_2$ term will grow more than a constant faster than the ln term.

- $n!$ (Factorial Growth): Not surprisingly, the factorial function grows the fastest. In the asymptotic form, it goes as n^n, which is clearly faster than when either the exponent or the base is logarithmic.

- (*b*) Let $f(n) = (1 + \sin n)$ and $g(n) = (1 + \cos n)$. Considering the $f = \mathcal{O}(g)$ case

$$(1 + \sin n) \leq k(1 + \cos n) \tag{9}$$

$$\frac{(1 + \sin n)}{(1 + \cos n)} \leq k \tag{10}$$

and the $g = \mathcal{O}(f)$

$$(1 + \cos n) \leq k'(1 + \sin n) \tag{11}$$

$$\frac{(1 + \cos n)}{(1 + \sin n)} \leq k' \tag{12}$$

In either case, the ratio of f and g oscillates between 0 and ∞ and is periodic in n. There is no finite positive constant which can satisfy either inequality.

- (*c*) $f = \mathcal{O}(g)$ means

$$f(N) \leq kg(N) \tag{13}$$

$$\implies f(f(N)) \leq kg(f(N)) \tag{14}$$

$$\leq kg(kg(N)) \tag{15}$$

If we are able to satisfy the inequality of Eq. 15, we show that $f = \mathcal{O}(g)$. Calculating the right hand side for the given functions (starting from the slowest growing ones) indicates that $f(n) = \mathcal{O}(2^n)$ and $f(n) = \Omega(n^{\ln n})$. Intuitively, this makes sense because the four slowest growing functions do not involve an exponential in n, so even upon composing the function with itself, we cannot get an exponent. All the functions greater than $f(n)$ do have exponential parts, so composing will make them doubly exponential.

Explicitly, we have for $g_1(n) = n^{\ln n}$

$$kg_1(kg_1(n)) = kg_1(kn^{\ln n}) = k\left(kn^{\ln n}\right)^{\ln\left(kn^{\ln n}\right)} \tag{16}$$

Choosing any k gives a growth that is slower than 2^n in the asymptotic limit.

For $g_2(n) = 2^n$

$$kg_2(kg_2(n)) = kg_2(k2^n) = k(2^{k2^n}) \tag{17}$$

Choosing any k, we see that 2^{k2^n} grows more quickly than 2^n.

4.2 Dr. Falken, Would you like to play a game?

- (*a*) Use *de Morgan's laws* $\neg(x \vee y) = \overline{x} \wedge \overline{y}$ and $\neg(x \wedge y) = \overline{x} \vee \overline{y}$
  - (*i*) Noting that $a = \neg\neg a$

$$\begin{aligned}(a \wedge b) \vee (c \wedge d) &= \neg\left[\neg\left((a \wedge b) \vee (c \wedge d)\right)\right] &(18)\\ &= \neg\left[\neg(a \wedge b) \wedge \neg(c \wedge d)\right] &(19)\\ &= (a \overline{\wedge} b) \overline{\wedge} (c \overline{\wedge} d) &(20)\end{aligned}$$

  - (*ii*)

$$\begin{aligned}\neg\left((\overline{a} \vee \overline{b}) \wedge (\overline{c} \vee \overline{d})\right) &= \neg(\overline{a} \vee \overline{b}) \vee \neg(\overline{c} \vee \overline{d}) &(21)\\ &= (a \wedge b) \vee (c \wedge d) &(22)\\ &= (a \overline{\wedge} b) \overline{\wedge} (c \overline{\wedge} d) \quad \text{from part } (i) &(23)\end{aligned}$$

Lets look at a sample game tree for $k = 2$ to get a feel for how NAND can be used in either case to evaluate the tree. In Fig. 1, we see a sample game tree in which Alice

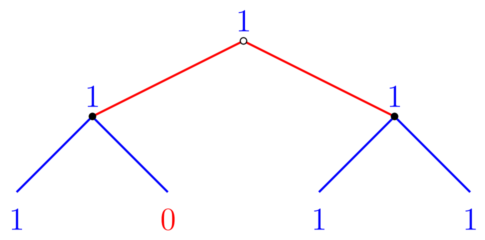

Figure 1: A sample game with $n = 2$ and $k = 2$ with players Alice and Bob

will lose. The leaves are given at the bottom, where a 1 indicates a win for Bob and a 0 indicates a win for Alice. The edges are colored with the person who gets to play for that round. Starting from the bottom up, we see that since Bob goes last, he can always pick a branch with a 1 in it. Moving up a level,we see that Alice will always lose because no matter which of her two branches she chooses Bob is guaranteed to win.

To evaluate this using the NAND formulas, first recall that for bits, NAND evaluates to 0 if all bits are one and is 1 if any bit is zero. Notice that I did not color these 0s and 1s as we still need to *interpret* what they represent with regard to Alice and Bob. Given the above example, we see that when it is a given player's turn, they have a winning strategy if at least one branch will result in a win for that player. Since we have decided to let 0 represent a win for Alice, on her turn, she will win if at least one branch is 0. This means she evaluates a function which returns 0 if at least one input is 0. In terms of the NAND function, we see that negating the output gives an AND function which is 0 if at least one input is 0. In the example, this is just ¬NAND(1, 1)=¬0=1.

Since Bob's strategy is the same as Alice's (can win if at least one branch indicates a win for him), he should apply the exact same operation—*except* he needs to negate the inputs, since his definition of a win is opposite of Alice. In the example, Bob evaluates NAND(¬1,¬0)=NAND(0,1)=1 and NAND(¬1,¬1)=NAND(0,0)=1.

- (b) Given that there is some level of equivalence in evaluating successive rounds of the tree, the problem asks us to analyze the difficulty of just evaluating NAND for a tree using the Las Vegas *depth-first pruning* algorithm. We consider each case individually.

  - $C_0(n,k)$ In order for NAND to evaluate to 0, all k inputs must be 1. This is the only case, so it is trivially the worst case. To evaluate this function, we have to look at all k inputs and ensure each is 1. Thus

$$C_0(n,k) = kC_1(n-1,k) \tag{24}$$

  - $C_1(n,k)$ In order for NAND to evaluate to 1, at least one input must be 0. In the worst case, we have $k-1$ inputs as 1 and one input as 0. In order to evaluate the expected cost, define the cost of having to make x queries before seeing 0 as

$$C(x;n,k) = (x-1)C_1(n-1,k) + C_0(n-1,k) \tag{25}$$

    and similarly, the probability of having to make x queries before seeing a 0 as

$$P(x;n,k) = \left(\frac{k-1}{k}\right)\left(\frac{k-2}{k-1}\right)\cdots\left(\frac{k+1-x}{k+2-x}\right)\left(\frac{1}{k+1-x}\right) = \frac{1}{k} \tag{26}$$

    The expected cost is then

$$\begin{aligned}
\sum_{x=1}^{k} P(x;n,k)C(x;n,k) &= \frac{1}{k}\sum_{x=1}^{k}\left[(x-1)C_1(n-1,k) + C_0(n-1,k)\right] &(27)\\
&= \frac{1}{k}\left[\left(\sum_{x=1}^{k}(x-1)\right)C_1(n-1,k) + kC_0(n-1,k)\right] &(28)\\
&= \frac{1}{k}\left[\left(\sum_{x'=0}^{k-1}x'\right)C_1(n-1,k) + kC_0(n-1,k)\right] &(29)\\
&= \frac{1}{k}\left[\frac{k(k-1)}{2}C_1(n-1,k) + kC_0(n-1,k)\right] &(30)\\
&= C_0(n-1,k) + \frac{k-1}{2}C_1(n-1,k) &(31)
\end{aligned}$$

  - (c) Using the above definitions, we plug in $C_0(n-1,k) = kC_1(n-2,k)$ and then use the ansatz $C_1(n,k) = Ar^n$

$$\begin{aligned}
0 &= C_1(n,k) - \frac{k-1}{2}C_1(n-1,k) - kC_1(n-2,k) &(32)\\
&= Ar^n - \frac{k-1}{2}Ar^{n-1} - kAr^{n-2} &(33)\\
&= r^2 - \frac{k-1}{2}r - k &(34)
\end{aligned}$$

Solving this quadratic equation and taking the positive root gives

$$C_1(n,k) = \left(\frac{k-1+\sqrt{k^2+14k+1}}{4}\right)^n \tag{35}$$

$$C_0(n,k) = k\left(\frac{k-1+\sqrt{k^2+14k+1}}{4}\right)^{n-1} \tag{36}$$

The total cost is the maximum of these two. Since $k > r$, this means that $C_0(n,k)$ is larger. Using the logarithm identity, $a^{\log_b c} = c^{\log_b a}$, we can bound the expression as follows

$$\begin{aligned} kr^{n-1} &= kr^{(n-1)\log_k k} && (37)\\ &= kr^{n\log_k (k^{n-1})} && (38)\\ &= kk^{(n-1)\log_k r} && (39)\\ &\leq k^{n\log_k r} && (40)\\ &= \mathcal{O}\left(N^{\log_k\left(\frac{k-1+\sqrt{k^2+14k+1}}{4}\right)}\right) && (41)\end{aligned}$$

– (*d*) Since $k = 2$, we have a binary tree. Consider the first non-trival case, where $n = 2$. A tree which requires examining all nodes is depicted in Fig. 2.

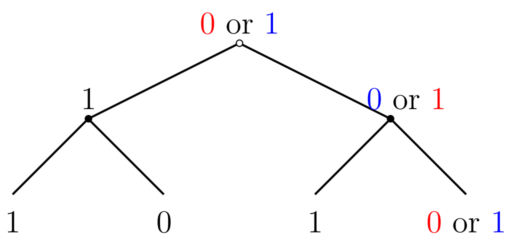

Figure 2: A tree which requires all N bits to be examined

Starting at the top, we know the left child must return 1 in order to have to examine the right branch. The same holds for the left child's left leaf. Since we want this left child to return 1, its right child must be a 0 for NAND(1,0) to give 1. Now walking down the right branch from the top root, we know the left leaf of this child must be 1 in order to have to look at the remaining leaf. However, this final leaf is free to be either 0 or 1 and this value determines the top level value. Generalizing, we can fix the above as the top-levels of an arbitrary depth n binary tree. For any leaf above, we have that if it is a 1, its left and right children are 1 and 0. If the leaf is a 0, its left and right children are 1 and 1. Therefore, we can assign the bottom leaf values using the following "string rewriting" rules. For $n \geq 2$, choose your start string S as either 1011 or 1010. Then define the bit rewrite map $r : \mathbb{B} \to \mathbb{B}^2$ as

$$r(0) := 11 \qquad \text{and} \qquad r(1) := 10 \tag{42}$$

and the string rewrite map $R : \mathbb{B}^n \to \mathbb{B}^{2n}$

$$R(S) := \bigotimes_{k=1}^{n} r(S_k) \tag{43}$$

To find the leaf node values, just evaluate $\overbrace{R(R(\cdots(R(S))))}^{n-2}$, where the rewrite map is applied $n-2$ times.

4.3 Clifford, but not the big red dog

- (a) Recalling the results of problem set 2, we have

$$HIH = H^2 = I \quad HXH = Z \quad HZH = X \quad HYH = -Y \tag{44}$$

A similar calculation for S gives

$$SIS^\dagger = I \quad SXS^\dagger = Y \quad SZS^\dagger = Z \quad SYS^\dagger = -X \tag{45}$$

- (b) As a matrix in the computational basis, $\Lambda(X)$ is

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{46}$$

Using the two factoids to bootstrap our way through the group. In the following, I do not write out the explicit matrix products which can be easily performed on computer. Intuition, or exhaustive search, allows one to recognize the form of the resulting product

1. $\Lambda(X) \cdot II \cdot \Lambda(X) = \Lambda(X)^2 = II$
2. $\Lambda(X) \cdot IX \cdot \Lambda(X) = IX$
3. $\Lambda(X) \cdot IY \cdot \Lambda(X) = ZY$
4. $\Lambda(X) \cdot XI \cdot \Lambda(X) = XX$
5. $\Lambda(X) \cdot YI \cdot \Lambda(X) = YX$

We can use the second factoid to construct any missing elements. The first factoid indicates that we are only missing $IZ, ZZ, ZI, XY, XZ, YY, YZ, ZX$. Using the Pauli algebra to take products, we find that

$$\begin{aligned}\Lambda(X) \cdot IZ \cdot \Lambda(X) &= -i\,(\Lambda(X) \cdot IX \cdot \Lambda(X))\,(\Lambda(X) \cdot IY \cdot \Lambda(X)) \\ &= -i(I \otimes X)(Z \otimes Y) = -iY \otimes iZ = Z \otimes Z\end{aligned} \tag{47}$$

$$\begin{aligned}\Lambda(X) \cdot ZI \cdot \Lambda(X) &= -i\,(\Lambda(X) \cdot XI \cdot \Lambda(X))\,(\Lambda(X) \cdot YI \cdot \Lambda(X)) \\ &= -i(X \otimes X)(Y \otimes X) = -iY \otimes iZ = Z \otimes I\end{aligned} \tag{48}$$

$$\begin{aligned}\Lambda(X)\cdot XY\cdot\Lambda(X) &= (\Lambda(X)\cdot XI\cdot\Lambda(X))\,(\Lambda(X)\cdot IY\cdot\Lambda(X)) \\ &= (X\otimes X)(Z\otimes Y) = -iY\otimes iZ = Y\otimes Z \end{aligned} \tag{49}$$

$$\begin{aligned}\Lambda(X)\cdot XZ\cdot\Lambda(X) &= (\Lambda(X)\cdot XI\cdot\Lambda(X))\,(\Lambda(X)\cdot IZ\cdot\Lambda(X)) \\ &= (X\otimes X)(Z\otimes Z) = -iY\otimes -iY = -Y\otimes Y \end{aligned} \tag{50}$$

$$\begin{aligned}\Lambda(X)\cdot ZX\cdot\Lambda(X) &= (\Lambda(X)\cdot ZI\cdot\Lambda(X))\,(\Lambda(X)\cdot IX\cdot\Lambda(X)) \\ &= (Z\otimes I)(I\otimes X) = Z\otimes X \end{aligned} \tag{51}$$

Tabulating these results, we have

P	$\Lambda(X)\cdot P\cdot\Lambda(X)$
II	II
IX	IX
IY	ZY
IZ	ZZ
XI	XX
XX	XI
XY	YZ
XZ	$-YY$
YI	YX
YX	YI
YY	$-XZ$
YZ	XY
ZI	ZI
ZX	ZX
ZY	IY
ZZ	IZ

- (c) Explicitly

$$TXT^{-1} = \begin{pmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{pmatrix}\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}1 & 0\\ 0 & e^{-i\pi/4}\end{pmatrix} \tag{52}$$

$$= \begin{pmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{pmatrix}\begin{pmatrix}0 & e^{-i\pi/4}\\ 1 & 0\end{pmatrix} \tag{53}$$

$$= \begin{pmatrix}0 & e^{-i\pi/4}\\ e^{i\pi/4} & 0\end{pmatrix} \tag{54}$$

$$= \frac{1}{\sqrt{2}}\begin{pmatrix}0 & 1-i\\ 1+i & 0\end{pmatrix} \tag{55}$$

$$= \frac{1}{\sqrt{2}}(X+Y) \tag{56}$$

- (d) Using the Pauli decomposition given in the hint, we can exhaustively search to find

$$\Lambda(S)(XI)\Lambda(S)^{-1} = \frac{1}{2}\left(XI + XZ + YI - YZ\right) \tag{57}$$

- (e) Again, using the hint (and noting $\Lambda^2(X)$ is its own inverse)

$$\Lambda^2(X)(IIZ)\Lambda^2(X) = \frac{1}{2}\left(IIZ + IZZ + ZIZ - ZZZ\right) \tag{58}$$

  You can use the fact that the resulting matrix is diagonal to limit yourself to considering only the Pauli products involving Z and I.