Chapter 1

General notions

Definition. **Time-constructible functions.** A function $T : \mathbb{N} \to \mathbb{N}$ is time-constructible if $T(n) \geq n$ and there is a TM M that given input 1^n computes the binary representation of $T(n)$ in time $T(n)$.

Examples for time-constructible functions: n, $n \log n$, n^2, 2^n. Allowing time bounds that are not time constructible can lead to anomalous results. The restriction $T(n) \geq n$ is to allow the algorithm time to read its input.

Definition. **Space-constructible functions.** A function $S : \mathbb{N} \to \mathbb{N}$ is space-constructible if there is a TM M that given input 1^n computes the binary representation of $S(n)$ using $O(S(n))$ space.

Examples for space-constructible functions: $\log n$, n, $n \log n$, n^2, 2^n. Intuitively, if S is space-constructible, then the machine "knows" the space bound it is operating under.

Definition. **Implicitly logspace computable functions.** A function $f : \{0,1\}^* \to \{0,1\}^*$ is implicitly logspace computable if f is polynomially bounded (i.e., there's some c such that $|f(x)| \leq |x|^c$ for every $x \in \{0,1\}^*$) and the languages $L_f = \{\langle x, i\rangle | f(x)_i = 1\}$ and $L'_f = \{\langle x, i\rangle | i \leq |f(x)|\}$ are in **L**.

———

Definition. **Oblivious Turing machines.** A TM M with the following property is called oblivious: Its head movements do not depend on the input but only depend on the input length. That is, for every input $x \in \{0,1\}^*$ and $i \in \mathbb{N}$, the location of M' s head at the i-th step of execution is only a function of $|x|$ and i.

Theorem. For every time-constructible $T : \mathbb{N} \to \mathbb{N}$, any $O(T(n))$ time TM M can be simulated by an $O(T(n)^2)$ time oblivious TM M'.

Proof. For every a in the alphabet of M, the alphabet of M' includes a and $\hat{a}$. Throughout the execution, exactly one symbol will be of hat-type, indicating the corresponding location of the head of M. M' first compute $T(|x|)$ and write a special symbol on the $T(|x|)$-th cell to designate the "right end" of its tape. To simulate one step of M, M' scan from left to right to find the hat-type symbol and update its record of the state of M, then move back from right to left. In the course of the scanning process, M' can modify the cell next to the hat-type symbol if required.

Comment.

(i) This result can be improved to $O(T(n) \log T(n))$.

Definition. **Pairwise independent hash functions.** Let $\mathcal{H}_{n,k}$ be a collection of functions from $\{0,1\}^n$ to $\{0,1\}^k$. We say that $\mathcal{H}_{n,k}$ is pairwise independent if for every $x, x' \in \{0,1\}^n$ with $x \neq x'$, when we choose h at random from $\mathcal{H}_{n,k}$, then the random variable $\langle h(x), h(x') \rangle$ is distributed uniformly on $\{0,1\}^k \times \{0,1\}^k$. In other words, $\forall y, y' \in \{0,1\}^k, \Pr_{h \in_R \mathcal{H}_{n,k}} [h(x) = y \wedge h(x') = y'] = 2^{-2k}$.

Valiant-Vazirani Lemma. Let $\mathcal{H}_{n,k}$ be a pairwise independent hash function collection from $\{0,1\}^n$ to $\{0,1\}^k$ and $S \subseteq \{0,1\}^n$ such that $2^{k-2} \leq |S| \leq 2^{k-1}$. Then

$$\Pr_{h \in_R \mathcal{H}_{n,k}} \left[\left| \left\{ x \in S : h(x) = 0^k \right\} \right| = 1 \right] \geq \frac{1}{8}$$

Comment. This result can be improved to $\frac{3}{16}$.

Chapter 2

NP

Cook-Levin Theorem.

1. SAT is NP-complete.

2. 3SAT is NP-complete.

Theorem.

1. For every $i \geq 1$, we have: $\Sigma_i\text{SAT} \in \Pi_i^p \iff \Sigma_i^p \subseteq \Pi_i^p \iff \Pi_i\text{SAT} \in \Sigma_i^p \iff \Pi_i^p \subseteq \Sigma_i^p \iff \mathbf{PH} = \Pi_i^p = \Sigma_i^p$ (i.e. the polynomial hierarchy collapses to the i-th level.)

2. If $\mathbf{P} = \mathbf{NP}$ then $\mathbf{PH} = \mathbf{P}$ (i.e. the polynomial hierarchy collapses to $\mathbf{P}$.)

Theorem. $\mathbf{P} \neq \mathbf{EXP}$

2.1 Decision problems & search problems

It turns out that for **NP**-complete complexity classes decision problems and search problems are equivalent in the sense that if the decision problem can be solved in polynomial time, then the search version of any **NP** problem can also be solved in polynomial time.

Theorem. Suppose that $\mathbf{P} = \mathbf{NP}$. Then, for every **NP** language L and a verifier TM M for L, there is a polynomial-time TM M' that on input $x \in L$ outputs a certificate for x (i.e. $M(x, M'(x)) = 1$ for $x \in L$).

Chapter 3

Boolean Circuits

3.1 Basic definitions

For every $n, m \in \mathbb{N}$, an n-input m-output **Boolean circuit** is a DAG with n sources (i.e. vertices with no incoming edges) standing for input variables and m sinks (i.e. vertices with no outgoing edges) standing for output gates. All nonsource vertices with r incoming edges are called **gates** of fan-in r, functioning as a logical operation $\{0,1\}^r \to \{0,1\}$. Given some input $x \in \{0,1\}^n$, the output of circuit C on x is denoted $C(x)$. The **size** of circuit C is the number of gates in it, denoted $|C|$. The **depth** of C is the number of gates of fan-in greater or equal to 2 on the longest path in the circuit. (Note that NOT gates do not contribute to the depth.)

In some generalizations to the above definition, some input bits of gates are allowed to be hardwired as 0 or 1. In this case, these constant input bits do not contribute to the size of the circuit.

The set of different gate (a.k.a. logical operation) types used in a Boolean circuit is called the **basis** (denoted Ω) for the circuit. The **fan-in of a basis** is the maximal fan-in of gates in the basis. A basis Ω is **complete** if every binary function can be computed by a circuit over Ω.

Some examples of complete bases:

- The **standard basis**, denoted Ω_0, is the set {AND, OR, NOT} in which AND and OR have fan-in 2.
- The **full two-input basis**, denoted B_2, consists of all two-input Boolean functions.
- The **dyadic unate basis**, denoted U_2, consists of all Boolean functions of the form $(x^a \wedge y^b)^c$, where $x^1 = x$ and $x^0 = \overline{x}$.
- The basis consisting of one function, NAND gate.

Example of incomplete bases: The **monotone basis** $\Omega_{\text{mon}} = \{\text{AND}, \text{OR}\}$ in which the fan-in is 2.

3.1.1 monotone circuits

A **monotone circuit** is a circuit over the monotone basis.

Define an ordering of n-bit binary strings: $\mathbf{x} \le \mathbf{y}$ if $\forall 1 \le i \le n$, $x_i \le y_i$, where $0 \le 0, 0 \le 1, 1 \le 1, 1 \not\le 0$. A **monotone (increasing) function** is a binary function $f : \{0,1\}^n \to \{0,1\}^m$ satisfying that $\forall \mathbf{x}, \mathbf{y} \in \{0,1\}^n, \mathbf{x} \le \mathbf{y} \implies f(\mathbf{x}) \le f(\mathbf{y})$.

Theorem. Every monotone function can be computed by a monotone circuit. Conversely, every monotone circuit computes a monotone function.

Proof. Using the expansion for monotone functions: $f_j(x_1, x_2, \cdots, x_n) = f_j(0, x_2, \cdots, x_n) \vee (x_1 \wedge f_j(1, x_2, \cdots, x_n))$.

Comment. Some monotone functions on n variables require monotone circuits with size exponential in n, but may be realized by polynomial-size circuits over the standard basis Ω_0. (proof?)

3.1.2 complexity measures

The **circuit size** of a binary function $f : \{0,1\}^n \to \{0,1\}^m$ w.r.t. the basis Ω, denoted $C_\Omega(f)$, is the minimum size of any circuit computing f over the basis Ω. The

circuit size with fan-out s, denoted $C_{s,\Omega}(f)$, is the circuit size of f when the fan-out of any gate is at most s. The **circuit depth** and the **circuit depth with fan-out** s of f are correspondingly denoted $D_\Omega(f)$ and $D_{s,\Omega}(f)$.

Effect of fan-out on circuit size:

Theorem. Let Ω be a complete basis of fan-in r. Then for any binary function $f : \{0,1\}^n \to \{0,1\}^m$ and integer $s \geq 2$, the following inequality holds:

$$C_\Omega(f) \leq C_{s,\Omega}(f) \leq C_\Omega(f)\left(1 + k_\Omega \frac{r-1}{s-1}\right) + k_\Omega \frac{m}{s-1}$$

where k_Ω is the number of gates from Ω needed to realize the single bit identity function (in fact, $k_\Omega = 1$ or 2 for any complete Ω if one allows for hardwired input bits to gates). Thus, circuit size increases by at most a constant factor when the fan-out of the circuit is reduced to s for $s \geq 2$.

3.2 Complexity classes

For simplicity, we assume that we use the standard basis Ω_0 ={AND, OR, NOT} in the following discussions and that the circuits produce one bit of output.

A **circuit family** is a sequence $\{C_n\}_{n\in\mathbb{N}}$ of Boolean circuits, where C_n has n inputs.

Definition. **class SIZE(**$T(n)$**).** Let $T : \mathbb{N} \to \mathbb{N}$. A language L is in SIZE($T(n)$) if there exists a $T(n)$-sized circuit family $\{C_n\}_{n\in\mathbb{N}}$, i.e. the size of C_n is no larger than $T(n)$, such that for every $x \in \{0,1\}^n$, $x \in L \iff C_n(x) = 1$.

Definition. **class P/poly.** The class of languages that are decidable by polynomial-sized circuit families, i.e., $\mathbf{P/poly} = \bigcup_c \mathbf{SIZE}\,(n^c)$.

Theorem. $\mathbf{P} \subseteq \mathbf{P/poly}$. In fact, for any $O(T(n))$-time oblivious TM M, there exists an $O(T(n))$-sized circuit family $\{C_n\}_{n\in\mathbb{N}}$ such that $C_n(x) = M(x)$ for every x.

Furthermore, for polynomial $T(n)$, such circuit family is P-uniform.

Comment. $\mathbf{P} \neq \mathbf{P/poly}$. In fact, every unary language (i.e. $L \subseteq \{1^n : n \in \mathbb{N}\}$) is in $\mathbf{P/poly}$, but some unary languages are undecidable, e.g. $n \in L$ iff the binary representation of n encodes $\langle M, x\rangle$ such that M halts on input x.

Definition. **P-uniform circuit families.** A circuit family $\{C_n\}$ is **P**-uniform if there exists a polynomial-time TM that on input 1^n outputs the description of the circuit C_n.

Definition. **Logspace-uniform circuit families.** A circuit family $\{C_n\}$ is logspace-uniform if there exists an implicitly logspace computable function mapping 1^n to the description of the circuit C_n.

Theorem. A language L is computable by a logspace-uniform circuit family $\iff$ L is computable by a **P**-uniform circuit family $\iff$ $L \in \mathbf{P}$. (In other words, restricting circuits to be **P**-uniform ”collapses” $\mathbf{P/poly}$ to **P**.)

Theorem. Every $f : \{0,1\}^n \to \{0,1\}$ can be computed by a circuit of size $\frac{2^n}{n}(1 + o(1))$.

Theorem. **Existence of hard functions.** For every $n > 1$, there exists a function $f : \{0,1\}^n \to \{0,1\}$ that cannot be computed by a circuit of size $\frac{2^n}{10n}$. In fact, the portion of all functions from $\{0,1\}^n$ to $\{0,1\}$ that are computable by a circuit of size $\frac{2^n}{10n}$ is at most $2^{-2^n/10}$.

3.3 Turing machines that take advice

Definition. **class $\mathbf{DTIME}(T(n))/a(n)$.** Let $T, a : \mathbb{N} \to \mathbb{N}$. The class of languages decidable by time-$T(n)$ TMs with $a(n)$ bits of advice, denoted $\mathbf{DTIME}(T(n))/a(n)$, contains every L such that there exists a sequence $\{\alpha_n\}_{n\in\mathbb{N}}$ of strings with $\alpha_n \in \{0,1\}^{a(n)}$ and a TM M satisfying $M(x, \alpha_n) = 1 \iff x \in L$ for every $x \in \{0,1\}^n$, where on input (x, α_n) M runs for at most $O(T(n))$ steps.

Theorem. $\mathbf{P/poly} = \bigcup_{c,d} \mathbf{DTIME}(n^c)/n^d$

3.4 Circuit satisfiability

Definition. **language CKT-SAT (circuit satisfiability).** A string representing an n-input circuit C is in CKT-SAT iff there exists $u \in \{0,1\}^n$ such that $C(u) = 1$.

Theorem. CKT-SAT is **NP**-complete.

3.5 Containment relationships concerning P/poly

Since $\mathbf{P} \subseteq \mathbf{P/poly}$, if we ever prove $\mathbf{NP} \not\subseteq \mathbf{P/poly}$, then we will have shown $\mathbf{P} \neq \mathbf{NP}$.

Karp-Lipton Theorem. $\text{SAT} \in \mathbf{P/poly} \iff \mathbf{NP} \subseteq \mathbf{P/poly} \implies \mathbf{PH} = \Sigma_2^p$

Meyer's Theorem. $\mathbf{EXP} \subseteq \mathbf{P/poly} \implies \mathbf{EXP} = \Sigma_2^p$

Corollary. $\mathbf{P} = \mathbf{NP} \implies \mathbf{EXP} \not\subseteq \mathbf{P/poly}$

3.6 Parallel computation

Definition. **class $\mathbf{NC}^d$, NC.** For every d, a language L is in $\mathbf{NC}^d$ if L can be decided by a family of circuits $\{C_n\}$ where C_n has poly(n) size and $O(\log^d n)$ depth. $\mathbf{NC} := \bigcup_{i \geq 1} \mathbf{NC}^i$.

Definition. **class $\mathbf{AC}^d$, AC.** The class $\mathbf{AC}^d$ is defined similarly to $\mathbf{NC}^d$ except that gates are allowed to have unbounded fan-in (i.e., the OR and AND gates can be applied to an arbitrary number of bits). $\mathbf{AC} := \bigcup_{i \geq 1} \mathbf{AC}^i$.

Chapter 4

Complexity of counting

4.1 Complexity classes

Definition. **class FP.** A function $f : \{0,1\}^n \to \mathbb{N}$ is in **FP** if f is computable by a poly-time TM.

Definition. **class #P.** A function $f : \{0,1\}^n \to \mathbb{N}$ is in **#P** (pronounced "sharp P") if there exist a polynomial p and a poly-time TM M such that for every $x \in \{0,1\}^*$:

$$f(x) = \left|\left\{y \in \{0,1\}^{p(|x|)} : M(x,y) = 1\right\}\right|$$

Properties.

(i) The definition implies that $f(x)$ can be expressed using poly($|x|$) bits.

(ii) An equivalent definition: **#P** consists of all functions f such that $f(x)$ is equal to the number of accepting paths in the configuration graph of a poly-time NTM M on input x.

(iii) $f \in \mathbf{\#P} \implies \{x \in \{0,1\}^* : f(x) \neq 0\} \in \mathbf{NP}$

(iv) $\mathbf{FP} \subseteq \mathbf{\#P}$

————

Definition. **class FP^f.** The set of functions that are computable by poly-time

TMs that have oracle access to a function f, i.e. have access to an oracle for language $O = \{\langle x, i\rangle : f(x)_i = 1\}$.

Definition. A function $f : \{0,1\}^n \to \mathbb{N}$ is $\#\mathbf{P}$-complete if $f \in \#\mathbf{P}$ and every $g \in \#\mathbf{P}$ is in $\mathbf{FP}^f$.

Properties.

(i) If any $\#\mathbf{P}$-complete function is in $\mathbf{FP}$, then $\#\mathbf{P} = \mathbf{FP}$.

(ii) #SAT (the function that given a Boolean formula ϕ outputs the number of satisfying assignments for ϕ) is $\#\mathbf{P}$-complete.

4.2 Toda's theorem

Definition. **class $\oplus$P.** A language L is in $\oplus\mathbf{P}$ (pronounced "parity P") if there exist a polynomial p and a poly-time TM M such that for every $x \in \{0,1\}^*$:

$$x \in L \iff \left|\left\{y \in \{0,1\}^{p(|x|)} : M(x,y) = 1\right\}\right| \text{ is odd}$$

Properties. $\oplus\mathbf{P}$ is closed under complementation.

Definition. A language L is $\oplus\mathbf{P}$-complete if $L \in \oplus\mathbf{P}$ and $L' \leq_p L$ for every $L' \in \oplus\mathbf{P}$.

Define the quantifier $\oplus$ as follows: For every Boolean formula φ, $\oplus_x \varphi(x)$ is true iff the number of x's such that $\varphi(x)$ is true is odd. Some properties of quantifier $\oplus$:

(i) $\oplus_{x,y}\varphi(x,y) = \oplus_x \oplus_y \varphi(x,y)$

Theorem. $\oplus$SAT (the language consisting of all the true quantified Boolean formula of the form $\oplus_x\varphi(x)$ where φ is an unquantified Boolean formula) is $\oplus\mathbf{P}$-complete.

Let USAT be the language of Boolean formulas that have a unique satisfying assignment.

Valiant-Vazirani Lemma. There is a probabilistic poly-time algorithm that, given input 1^n, produces a Boolean formula $\tau(x,y)$ where x is a vector of n Boolean variables and y a vector of poly(n) Boolean variables, such that: for any Boolean function $f:\{0,1\}^n \to \{0,1\}$, define $\alpha(x,y) = \tau(x,y) \wedge (f(x)=1)$, then

$$\exists_x f(x) = 1 \implies \Pr[\alpha \in \text{USAT}] \geq \frac{1}{8n}$$
$$\neg\exists_x f(x) = 1 \implies \Pr[\alpha \in \text{SAT}] = 0$$

Proof. The algorithm choose $k \in_R \{2, \cdots, n+1\}$ and $h \in_R \mathcal{H}_{n,k}$ where $\mathcal{H}_{n,k}$ is an efficiently computable pairwise independent hash function collection from $\{0,1\}^n$ to $\{0,1\}^k$. Then $(f(x)=1) \wedge \left(h(x) = 0^k\right)$ satisfies the desired probability results. Using the Cook-Levin transformation, we can write $\tau(x,y)$ to express the computation of $h(x)$, where y reprensents the snapshots of TM computing h.

Consequence 1. There exists a probabilistic poly-time algorithm A such that for every n-variable Boolean formula φ:

$$\varphi \in \text{SAT} \implies \Pr[A(\varphi) \in \text{USAT}] \geq \frac{1}{8n}$$
$$\varphi \notin \text{SAT} \implies \Pr[A(\varphi) \in \text{SAT}] = 0$$

Consequence 2. There exists a probabilistic poly-time algorithm A such that for every n-variable Boolean formula φ: (The success probability can be boosted to $1 - 2^{-O(m)}$ by running the procedure $O(mn)$ times.)

$$\varphi \in \text{SAT} \implies \Pr[A(\varphi) \in \oplus\text{SAT}] \geq \frac{1}{8n}$$
$$\varphi \notin \text{SAT} \implies \Pr[A(\varphi) \in \oplus\text{SAT}] = 0$$

Consequence 3. There exists a probabilistic algorithm that, given as input an (unquantified) Boolean formula φ and 1^m, runs in poly$(|\varphi|, m)$ time, and produces a quantified Boolean formula $\oplus_x \psi(x)$ where ψ is an unquantified Boolean formula, such that:

$$\exists_x \varphi(x) \implies \Pr\left[\oplus_x \psi(x) \in \oplus\mathrm{SAT}\right] \geq 1 - 2^{-m}$$

$$\neg\exists_x \varphi(x) \implies \Pr\left[\oplus_x \psi(x) \in \oplus\mathrm{SAT}\right] = 0$$

Consequence 4. There exists a probabilistic algorithm that, given as input an (unquantified) Boolean formula φ and 1^m, runs in poly$(|\varphi|, m)$ time, and produces a quantified Boolean formula $\oplus_x \psi(x)$ where ψ is an unquantified Boolean formula, such that:

$$\forall_x \varphi(x) \implies \Pr\left[\oplus_x \psi(x) \in \oplus\mathrm{SAT}\right] = 1$$

$$\neg\forall_x \varphi(x) \implies \Pr\left[\oplus_x \psi(x) \in \oplus\mathrm{SAT}\right] \leq 2^{-m}$$

Chapter 5

Quantum Computing

5.1 Quantum Circuits

Definition. **Elementary quantum gates.** Quantum gates acting on three or less qubits of the register. (Note: Replacing the constant 3 with any constant greater or equal to 2 leads to an equivalently powerful model.)

Definition. Let $T : \mathbb{N} \to \mathbb{N}$ be a function. A language L is **decidable with bounded error using $T(n)$-size uniform quantum circuits** if there is a polynomial-time TM that on input $(1^n, 1^{T(n)})$ for any $n \in \mathbb{N}$ outputs the descriptions of elementary quantum gates $U_1, \cdots, U_{T(n)}$ such that for every $x \in \{0,1\}^n$, the following process outputs 1 (0) with probability at least 2/3 if $x \in L$ ($x \notin L$).

1. Initialize an m-qubit quantum register to the state $|x0^{m-n}\rangle$ (m is the number of qubits the quantum gates are to be applied to).

2. Apply one by one the elementary quantum gates $U_1, \cdots, U_{T(n)}$ to the register.

3. Measure the first qubit of the register and output the result.

Definition. **BQP.** A Language L is in **BQP** if there is some polynomial such that L is decidable with bounded error using $p(n)$-size uniform quantum circuits.

Theorem. **BPP** $\subseteq$ **BQP** $\subseteq$ **PSPACE**.

Theorem. Every 2×2 unitary with real entries can be written as either a rotation matrix $\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$ or a reflextion matrix $\begin{pmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{pmatrix}$ for some $\theta \in [0, 2\pi)$. Equivalently, Every 2×2 unitary with real entries can be written as a rotation matrix, possibly preceded and followed by Z gates, i.e. $U = \begin{pmatrix} 1 & 0 \\ 0 & s_1 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & s_2 \end{pmatrix}$ for some $\theta \in [0, 2\pi)\,, s_1, s_2 \in \{1, -1\}$.

5.2 Shallow Quantum Circuits

Definition. **class $\mathbf{QNC}^0$.** A language L is in $\mathbf{QNC}^0$ if L can be decided by a family of quantum circuits $\{C_n\}$ with bounded fan-in where C_n has poly(n) size and $O(1)$ (constant) depth.

Problem. **Hidden linear function (HLF) problem.** Given an $n \times n$ symmetric binary matrix A specifying a quadratic form $q(x) = x^\top A x \pmod 4$, find a binary vector $z \in \{0,1\}^n$ such that $q(x) = 2z^\top x \pmod 4$ for all $x \in \ker A$, where $\ker A = \{x \in \{0,1\}^n \mid Ax = 0 \pmod 2\}$ is the binary null-space of A.

Theorem. Such a binary string z always exists for any A. Moreover, if the dimension of $\ker A$ is k, then the number of solutions is 2^{n-k}.

Problem. **2D Hidden linear function (2D HLF) problem.** The off-diagonal entries of A can be interpreted as the adjacency matrix of a graph $G(A)$.

2D HLF are special instances of HLF, where $n = N^2$ for some integer N and $G(A)$ is a subgraph of the square grid with $N \times N$ vertices.

Quantum algorithm for 2D HLF problem. Prepare the state:

$$|\Psi_q\rangle = H^{\otimes n} U_q H^{\otimes n} |0\rangle^{\otimes n} = 2^{-n} \sum_{x,z \in \{0,1\}^n} i^{q(x)} (-1)^{z^T x} |z\rangle$$

where U_q is defined as $U_q|x\rangle = i^{q(x)}|x\rangle$. U_q can be realized using only CZ gates and S gates: apply CZ_{ij} for $i \neq j$ whenever $A_{ij} = 1$; apply S_i whenver $A_{ii} = 1$. In fact, this is done by adding ancillary qubits that store the input matrix A (A_{ij} is provided at the edge $\{i, j\}$ and A_{ii} at the vertex i) and replacing each gate of U_q by its controlled version.

Then, measuring the qubits in the computational basis produces a uniformly random bit string z from all solutions to the problem.

Comments.

a. This shows that 2D HLF can be solved with certainty using $\mathbf{QNC}^0$ circuit.

b. The above algorithm can be alternatively described as a sequence of single-qubit Pauli measurements performed on the graph state $|\Psi_{G(A)}\rangle = \prod_{i<j} \mathrm{CZ}_{ij}^{A_{ij}} H^{\otimes n}|0\rangle^{\otimes n}$. Qubit i is measured in the X (Y) basis if $A_{ii} = 0$ (1).

Theorem. For all sufficiently large N: If an $\mathbf{NC}^0\mathbf{/rpoly}$ classical circuit (an $\mathbf{NC}^0$ circuit with access to polynomially many additional input bits that are sampled from any probability distribution) $\{C_n\}$ with fan-in at most K solves all $N \times N$ instances of the 2D HLF problem with probability greater than 7/8, then the depth of C_n is at least $\dfrac{\log N}{8 \log K}$.