第７回量子ソフトウェアハンズオン（産学協働ゼミ）

テンソルネットワークを用いた量子回路シミュレーション

東大理　大久保毅

目次

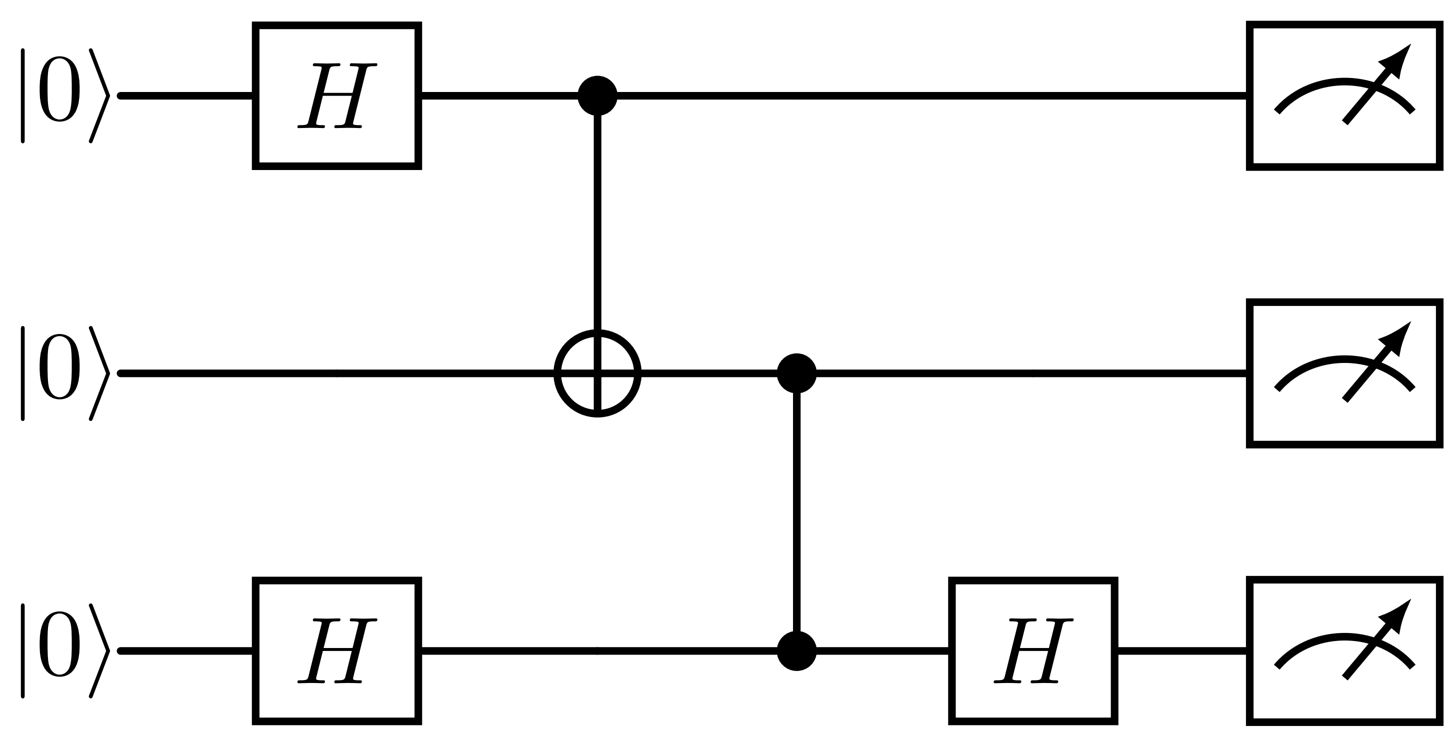
- 量子計算の概要
  - 量子回路の復習
  - 演習で扱う量子アルゴリズム
- テンソルネットワークによるシミュレーション
  - 量子回路とテンソルネットワーク
  - 計算コストと縮約順序
  - 多量子ビットゲートの行列積分解
  - 量子状態のサンプリング

量子計算の概要：量子回路の復習

量子回路

量子回路：**量子ビット**に演算する**ゲート操作**の設計図

- 量子ビットの初期状態を準備
- 左から順番に「量子ゲート」を演算する
  - 量子ゲートはユニタリ行列 $UU^\dagger = U^\dagger U = I$
  - 1qubit、または2qubitに作用するものが基本
- 最後に「測定」して情報（計算結果）を得る



典型的な量子ゲート

$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ の二つのベクトルで状態を表す

1 qubit ゲート

量子回路での記号

- *X* ゲート (NOT ゲート）

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

ビットを反転

$$X|0\rangle = |1\rangle$$
$$X|1\rangle = |0\rangle$$

$$-\boxed{X}-$$

- *H*（Hadamard）ゲート

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

重ね合わせ状態を生成

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

$$H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

$$-\boxed{H}-$$

典型的な量子ゲート

$$|\Psi\rangle = C_{00}|00\rangle + C_{01}|01\rangle + C_{10}|10\rangle + C_{11}|11\rangle = \begin{pmatrix} C_{00} \\ C_{01} \\ C_{10} \\ C_{11} \end{pmatrix}$$

2-qubit ゲート

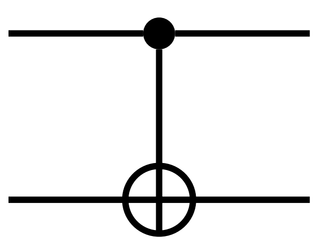
量子回路での記号

- *CX* (Controlled-NOT) ゲート

1番目のビットに依存して
2番目のビットを反転

$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$CX|00\rangle = |00\rangle$$
$$CX|01\rangle = |01\rangle$$
$$CX|10\rangle = |11\rangle$$
$$CX|11\rangle = |10\rangle$$

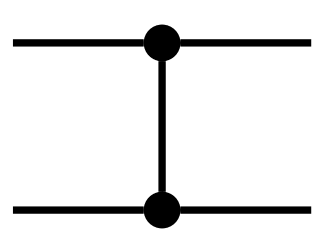


- *CZ*（Controlled-*Z*）ゲート

|11>にマイナス符号がつく

$$CZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

$$CZ|00\rangle = |00\rangle$$
$$CZ|01\rangle = |01\rangle$$
$$CZ|10\rangle = |10\rangle$$
$$CZ|11\rangle = -|11\rangle$$



量子回路における量子的な演算

- **並列性**
  - 量子的な重ね合わせ状態を入力すれば、最終状態は対応する出力の重ね合わせになる
- **分岐**
  - *H*ゲートなどにより、状態が"分岐"（|0>, |1>で見た場合）
- **干渉**
  - 重ね合わせの係数は、場合によってはキャンセルし、消える
- **測定による収縮**
  - 測定した基底のいずれか一つの状態になる

量子アルゴリズム：

これらを組み合わせ、欲しい答えが高確率で実現するようにする

（補足）量子状態と確率

1 qubit状態：$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix}\alpha \\ \beta\end{pmatrix}$

➡ qubitが状態 $|0\rangle$ にいるか $|1\rangle$ にいるかを観測すると

確率 $P(|0\rangle) = \dfrac{|\alpha|^2}{|\alpha|^2 + |\beta|^2}$ で状態 $|0\rangle$

確率 $P(|1\rangle) = \dfrac{|\beta|^2}{|\alpha|^2 + |\beta|^2}$ で状態 $|1\rangle$

が観測される

$= 1 - P(|0\rangle)$

＊以降、量子状態は規格化されているとする

$$\langle\Psi|\Psi\rangle \equiv \left(\alpha^* \beta^*\right)\begin{pmatrix}\alpha \\ \beta\end{pmatrix} = \underline{|\alpha|^2 + |\beta|^2 = 1}$$

N-qubit状態：$|\Psi\rangle = \sum_{\{i_1, i_2, \ldots i_N\}} \Psi_{i_1 i_2 \ldots i_N} |i_1 i_2 \ldots i_N\rangle$

➡ 2^N種類の$\underline{\text{古典bit状態}}$がそれぞれ確率的に観測される

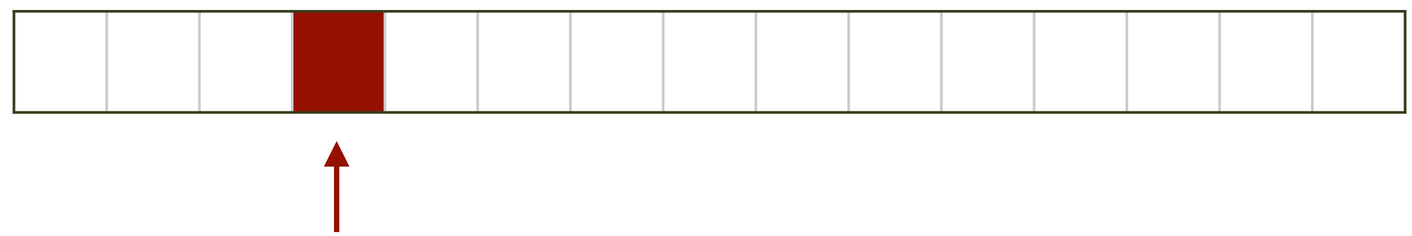
$|010111\cdots\rangle$

演習で扱う量子アルゴリズム

Groverのアルゴリズム

Groverのアルゴリズム (L. K. Grover, 1996)

非構造化データの探索： *N*個のデータ中で特定の"アタリ"はどこ？



古典コンピュータ：

基本的には一つ一つ"箱"をチェックしていく $\sim O(N)$

量子コンピュータでのGrover のアルゴリズムを使うと、

$\sim O(\sqrt{N})$ の計算時間で探索可能

古典コンピュータ

$\sim O(N)$

量子コンピュータ

$\sim O(\sqrt{N})$

"２乗"の計算時間削減！

Quantum bit string comparator

Quantum bit string comparator

与えられた2つのビット列を比較し、それらの大小関係を判定

ビット列：a, b ➡ $a > b, a = b, a < b$ を判定

D. S. Oliveira and R. V. Ramos (2007)の量子アルゴリズム

ビット列：a, b →量子状態とする：$|a\rangle, |b\rangle$

補助量子ビット：*m*+2 個

量子回路：U →2個の補助ビットに判定結果を与える

$$U|a\rangle|b\rangle|0\rangle^{\otimes m}|00\rangle = \underline{|a\rangle|b\rangle|\psi\rangle|xy\rangle}$$

$a > b$ ➡ $|xy\rangle = |10\rangle$

$a = b$ ➡ $|xy\rangle = |00\rangle$

$a < b$ ➡ $|xy\rangle = |01\rangle$

＊$|a\rangle, |b\rangle$ が重ね合わせ状態でない場合の表現

Quantum bit string comparator

Quantum bit string comparator

$$U|a\rangle|b\rangle|0\rangle^{\otimes m}|00\rangle = |a\rangle|b\rangle|\psi\rangle|xy\rangle$$

$a > b \Rightarrow |xy\rangle = |10\rangle$

$a = b \Rightarrow |xy\rangle = |00\rangle$

$a < b \Rightarrow |xy\rangle = |01\rangle$

例1：2ビットの比較

$|a\rangle = |11\rangle, |b\rangle = |01\rangle \Rightarrow |xy\rangle = |10\rangle$

$|a\rangle = |11\rangle, |b\rangle = |11\rangle \Rightarrow |xy\rangle = |00\rangle$

例2：重ね合わせ状態

$|a\rangle = |11\rangle, |b\rangle = \alpha|01\rangle + \beta|11\rangle$

$$|abxy\rangle = \alpha\underbrace{|11\rangle|01\rangle|10\rangle}_{|a\rangle|b\rangle|xy\rangle} + \beta\underbrace{|11\rangle|11\rangle|00\rangle}_{|a\rangle|b\rangle|xy\rangle}$$

確率 $|\alpha|^2$ で $a > b$
確率 $|\beta|^2$ で $a = b$

$\Rightarrow$ $|xy\rangle$ を測定すると

確率 $|\alpha|^2$ で $|10\rangle$
確率 $|\beta|^2$ で $|00\rangle$

量子的な重ね合わせ状態に対しては、出力結果も重ね合わせ状態になる

＊出力される状態は $|a\rangle, |b\rangle$ と $|xy\rangle$ が強くエンタングルした状態

テンソルネットワークによるシミュレーション

量子回路の古典シミュレーション

Schrödinger シミュレーション（State vector simulation）

量子回路を量子状態の時間発展とみなして計算する

n量子ビットの状態：2^n次元の複素数ベクトル

m量子ビットゲート：$2^m \times 2^m$次元の行列

→ 量子ビット数に関して指数関数のコスト

テンソルネットワークシミュレーション

量子回路をテンソルネットワークとみなして計算

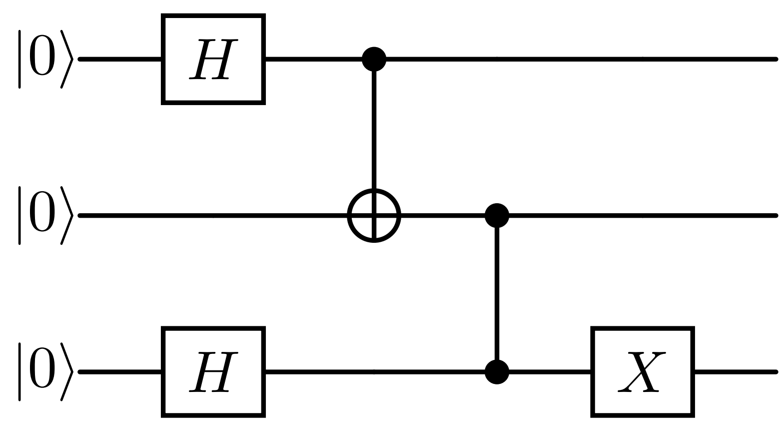
- 縮約順序の工夫による計算量削減
- 量子ゲート、量子“状態”のテンソル分解

Cf. Feynmann シミュレーション

シミュレーションで求めたいもの

1. 量子回路の最終的な量子状態

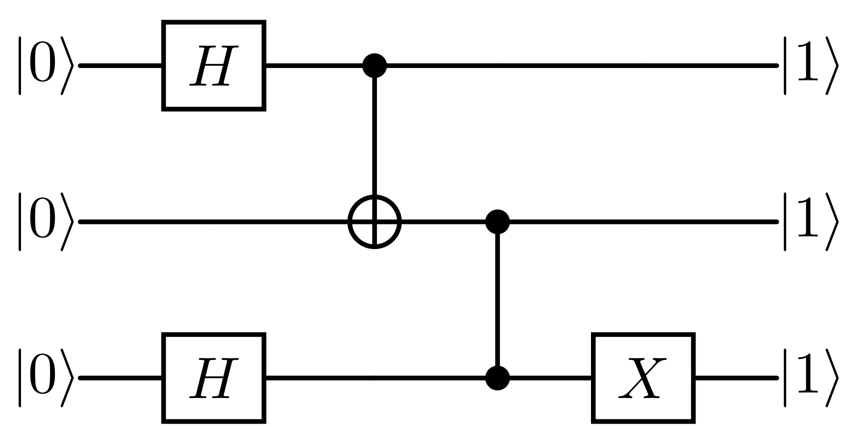
$|\psi\rangle =$



2. 特定の状態を測定する確率

例：$|111\rangle$　を測定する確率 = $|\underbrace{\langle 111|\psi\rangle}_{\text{確率振幅}}|^2$

$\langle 111|\psi\rangle =$



シミュレーションで求めたいもの：密度行列

3. （縮約）密度行列

n量子ビット状態：$|\psi\rangle$

$\boxed{|i_1 i_2 \cdots i_n\rangle \text{ の観測確率}}$

$$P(i_1, i_2, \cdots, i_n) = |\langle i_1 i_2 \cdots i_n|\psi\rangle|^2$$
$$= \langle i_1 i_2 \cdots i_n|\psi\rangle\langle\psi|i_1 i_2 \cdots i_n\rangle$$

$\boxed{i_1 \text{が観測される確率（周辺分布）}}$

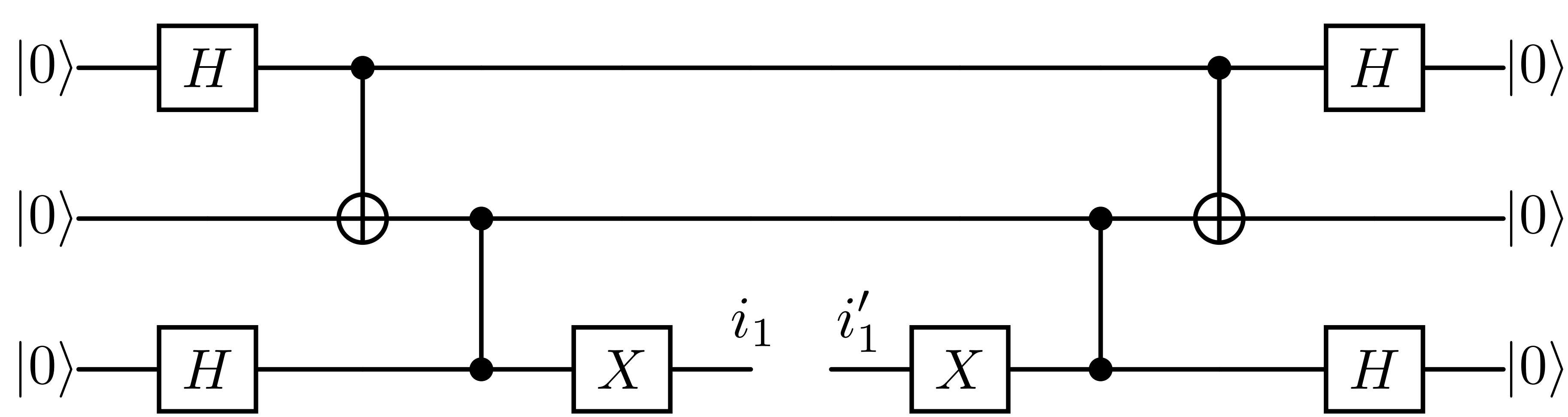
$$P(i_1) = \sum_{i_2, i_3, \ldots, i_n = 0,1} P(i_1, i_2, \ldots, i_n)$$

この周辺分布は、縮約密度行列

$$\rho_{i_1, i_1'} = \sum_{i_2, i_3, \ldots, i_n} \langle {\color{red} i_1} i_2 \cdots i_n|\psi\rangle\langle\psi|{\color{red} i_1'} i_2 \cdots i_n\rangle$$

の対角成分になっている

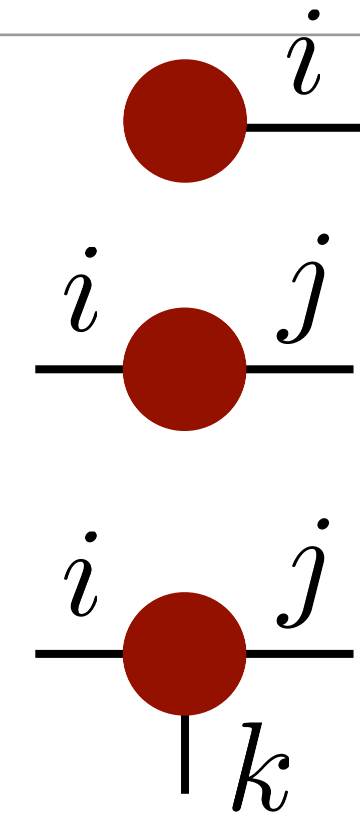
$\rho_{i_1, i_1'} =$



量子回路とテンソルネットワーク

ダイアグラムを用いたテンソル表記

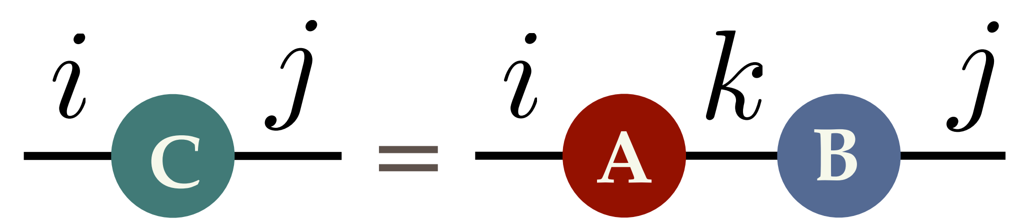
- ベクトル　$\vec{v} : v_i$
- 行列　$M : M_{i,j}$
- テンソル　$T : T_{i,j,k}$



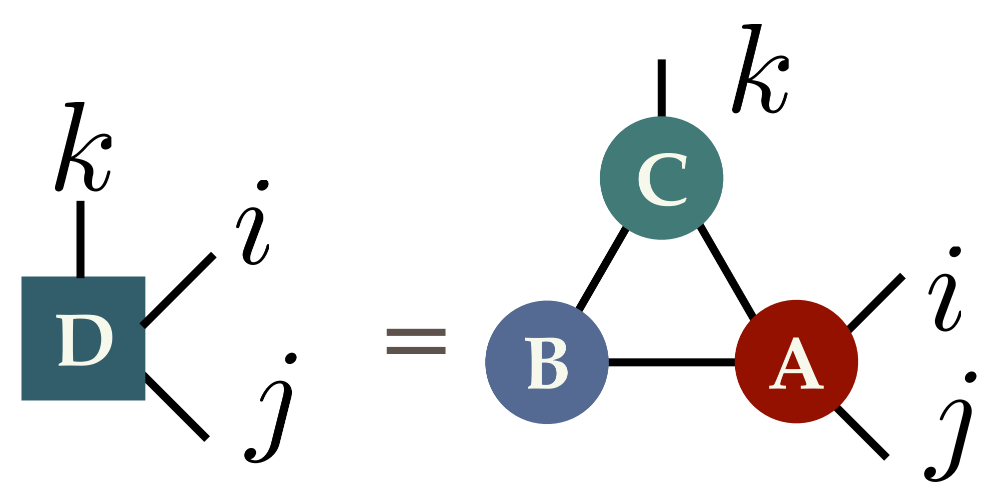
＊n階のテンソル＝n本の足

テンソルの積（縮約）の表現

$$C_{i,j} = (AB)_{i,j} = \sum_k A_{i,k} B_{k,j}$$

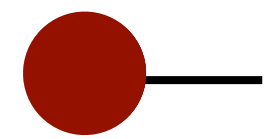


$$D_{i,j,k} = \sum_{\alpha,\beta,\gamma} A_{i,j,\alpha,\beta} B_{\beta,\gamma} C_{\gamma,k,\alpha}$$



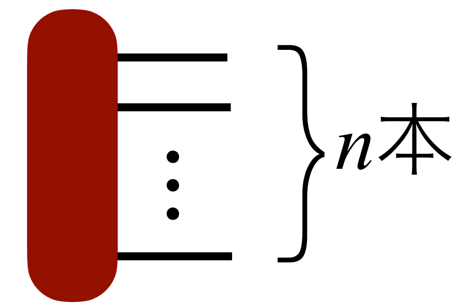
量子回路のテンソルネットワーク表現

1量子ビット状態：次元2のベクトル



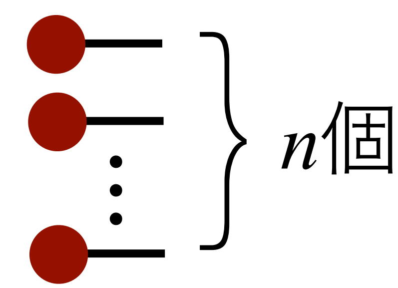
n量子ビット状態 ：次元2^nのベクトル= n本足のテンソル

＊各足の次元は2



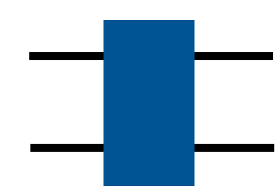
（通常の）初期状態 $|00\cdots0\rangle$：

次元2のベクトルの直積テンソル

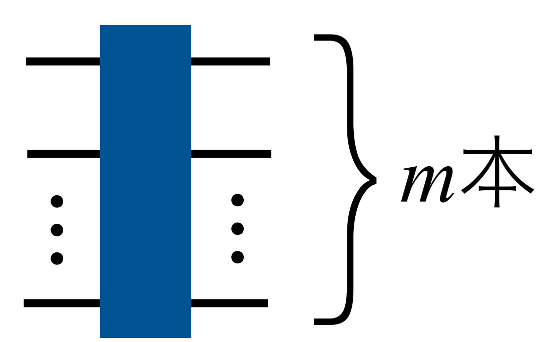


1量子ビットゲート ：2×2の行列

2量子ビットゲート ：$2^2\times2^2$の行列= 2×2×2×2の4本足テンソル

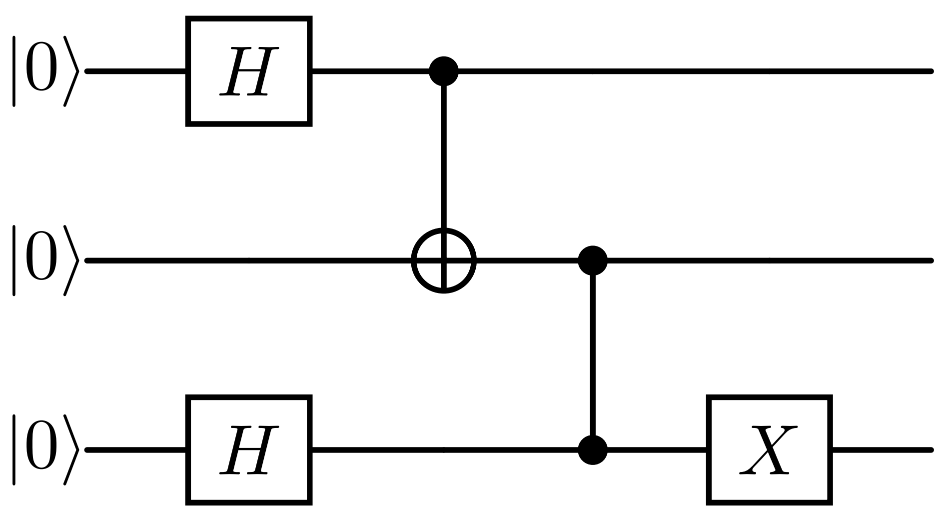


m量子ビットゲート ：2m本足のテンソル



量子回路のテンソルネットワーク表現

量子回路

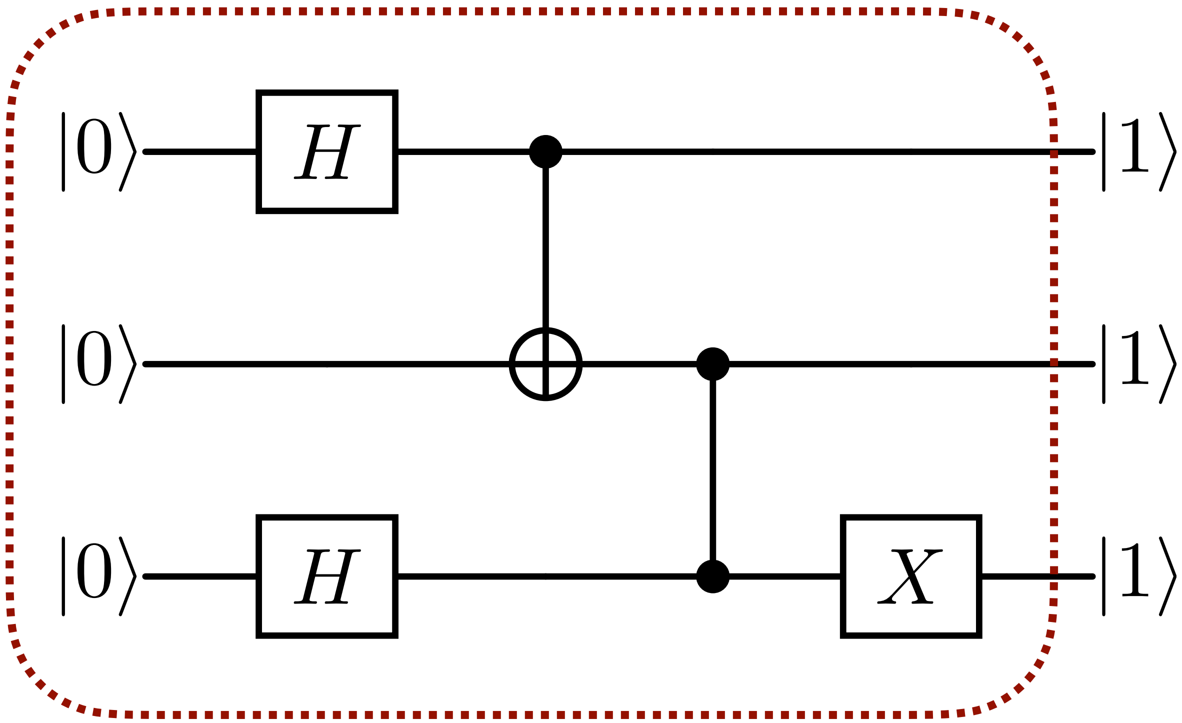


テンソルネットワーク表現

確率振幅とテンソルネットワーク

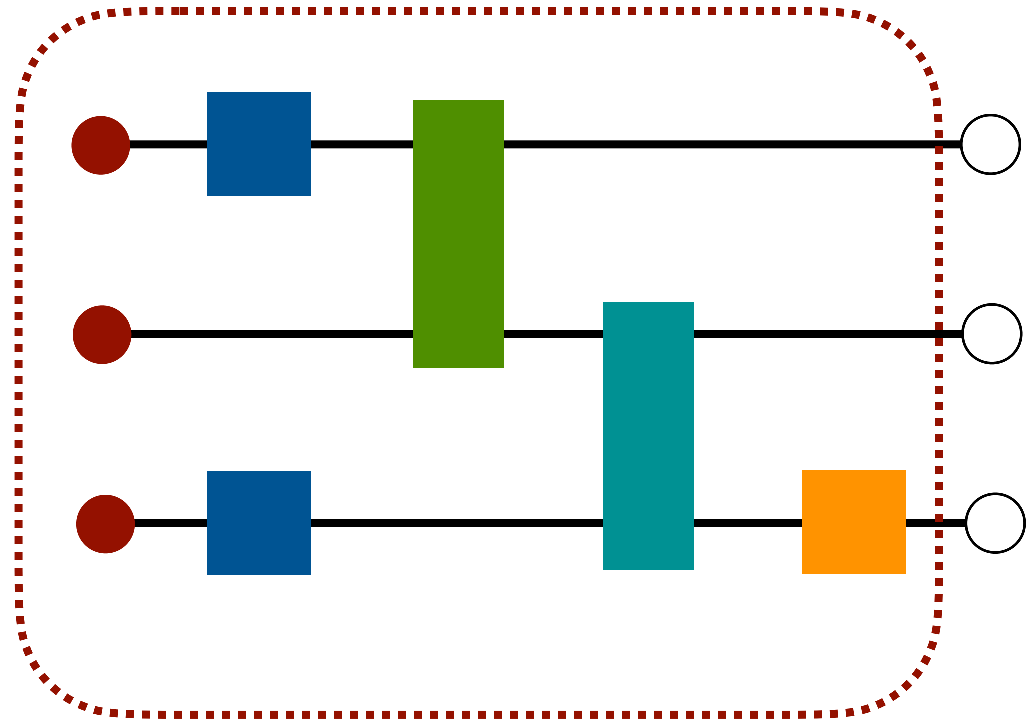
確率振幅：$\langle 111|\psi\rangle$

量子回路



$|\psi\rangle$

テンソルネットワーク表現

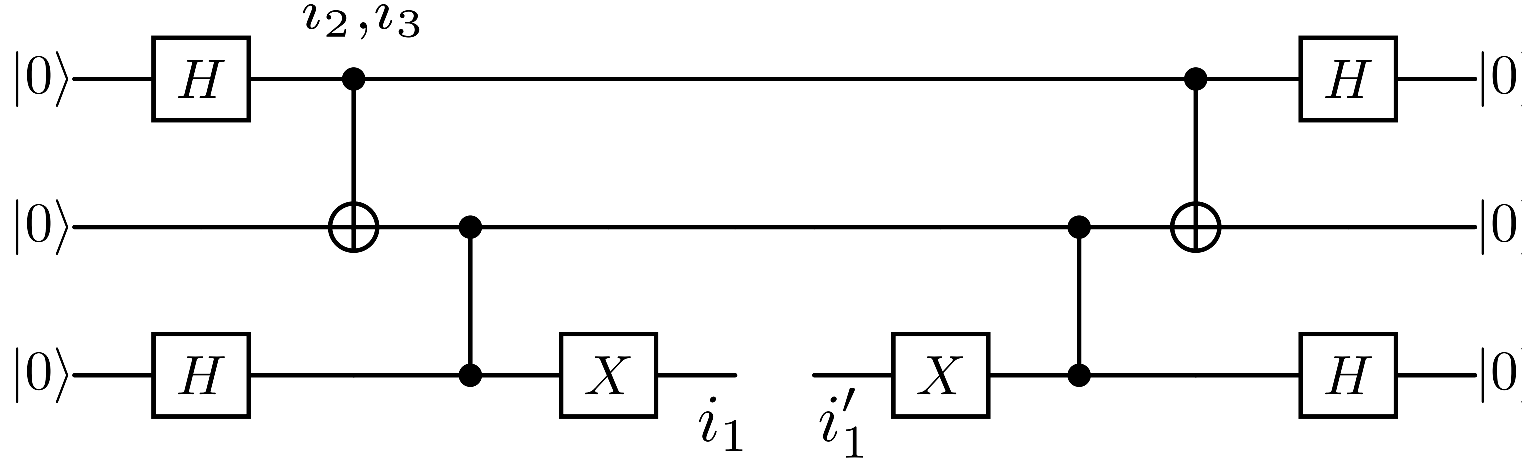


$|\psi\rangle$

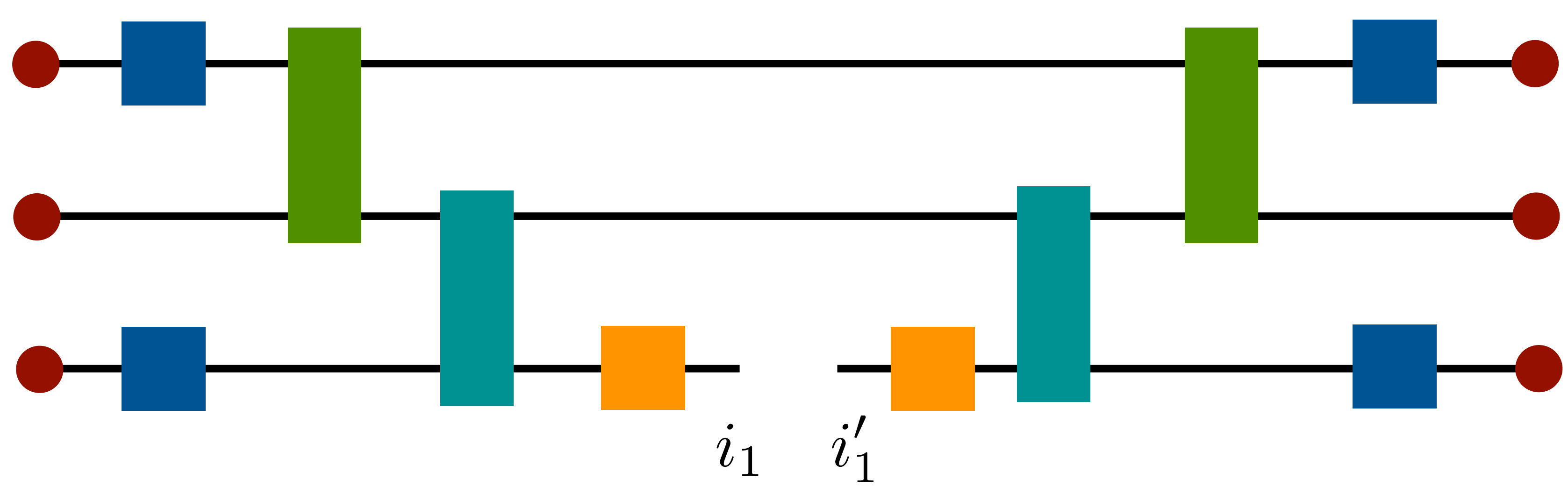
縮約密度行列とテンソルネットワーク

縮約密度行列 $\rho_{i_1,i_1'} = \sum_{i_2,i_3} \langle i_1 i_2 i_3 | \psi \rangle \langle \psi | i_1' i_2 i_3 \rangle$

量子回路



テンソルネットワーク表現



テンソルネットワーク表現のメリット？

- 縮約順序の工夫
- 多量子ビットの効率的な表現

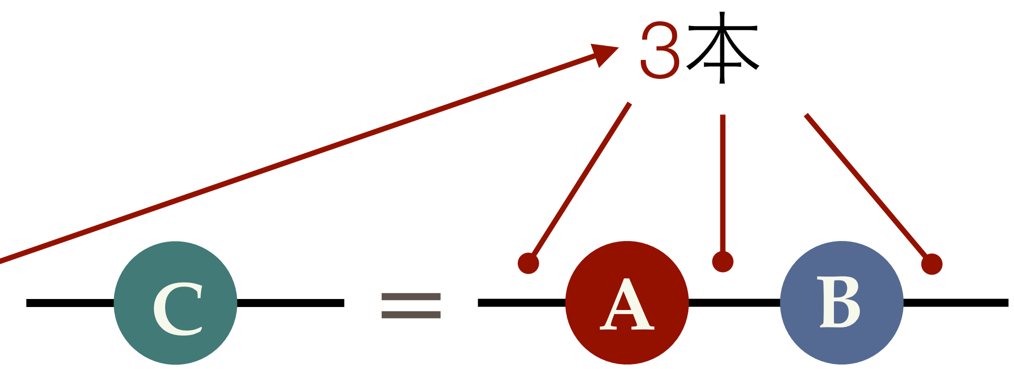
計算コストと縮約順序

縮約の計算量

行列積： $A, B = \chi \times \chi$

$$C = AB$$

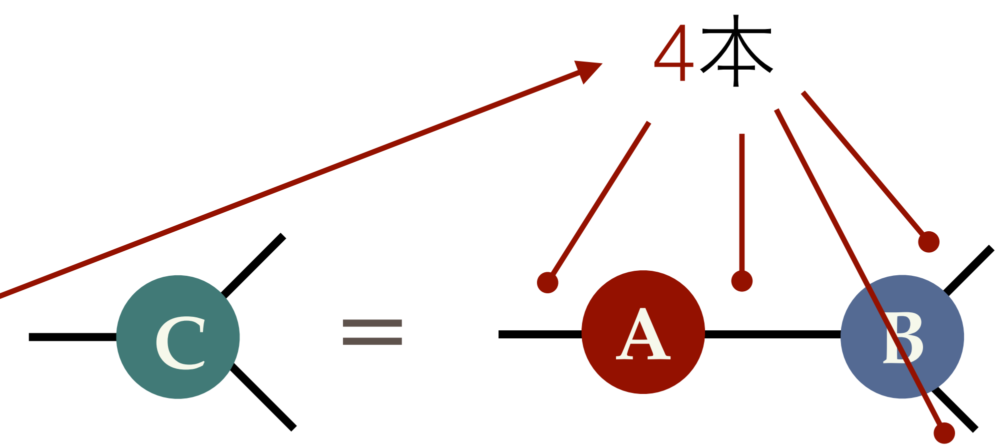
の計算量= $O(\chi^3)$



テンソル縮約： $A = \chi \times \chi$
$B = \chi \times \chi \times \chi$

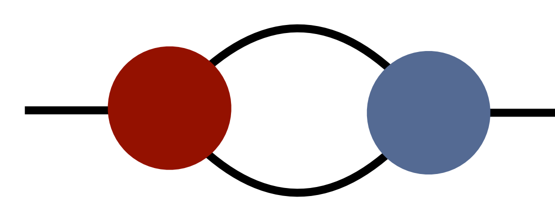
$$C = \sum_{\alpha} A_{:,\alpha} B_{\alpha,:,:}$$

の計算量= $O(\chi^4)$



ダイアグラムとの対応

- 縮約の計算量はダイアグラムの足の数で分かる
- （メモリ使用量も分かる）



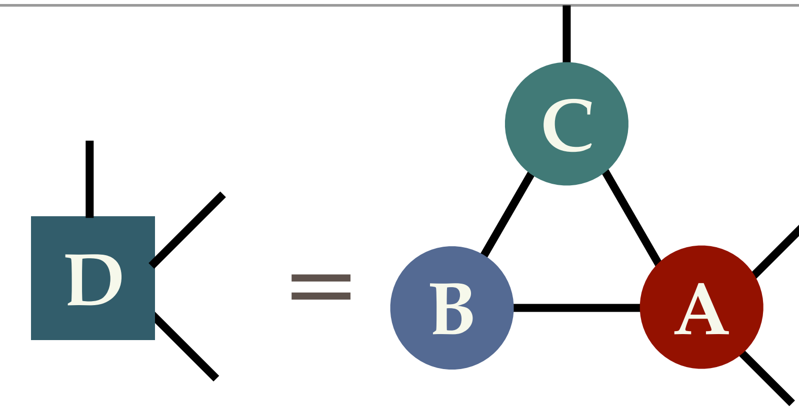
4本 $\longrightarrow O(\chi^4)$

縮約の計算量と計算順

テンソル縮約：

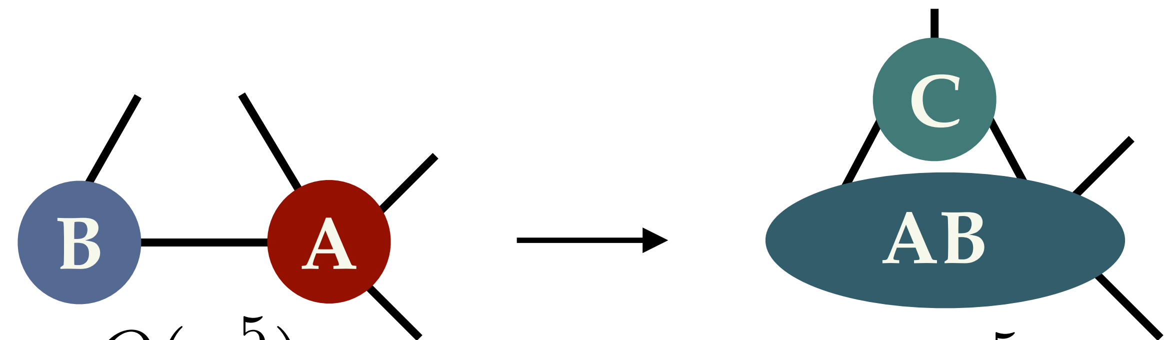
$$A = \ \chi \times \chi \times \chi \times \chi$$
$$B = \ \chi \times \chi$$
$$C = \ \chi \times \chi \times \chi$$

$$D = \sum_{\alpha,\beta,\gamma} A_{:,:,\alpha,\beta} B_{\beta,\gamma} C_{\gamma,:,\alpha}$$



Case 1: $D = (AB)C$

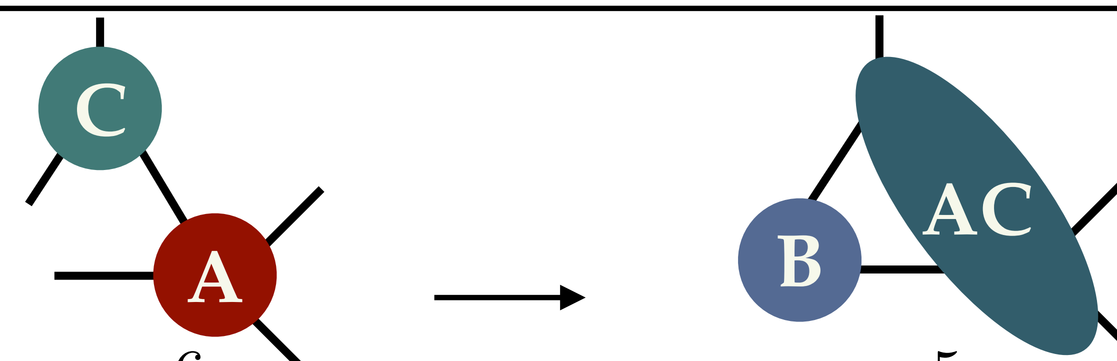
の計算量= $O(\chi^5) + O(\chi^5)$



$O(\chi^5)$ $O(\chi^5)$

Case 2: $D = (AC)B$

の計算量= $O(\chi^6) + O(\chi^5)$



$O(\chi^6)$ $O(\chi^5)$

縮約の評価順で計算量が変わる！

＊最適順序の決定はNP困難。実用的なアルゴリズム例

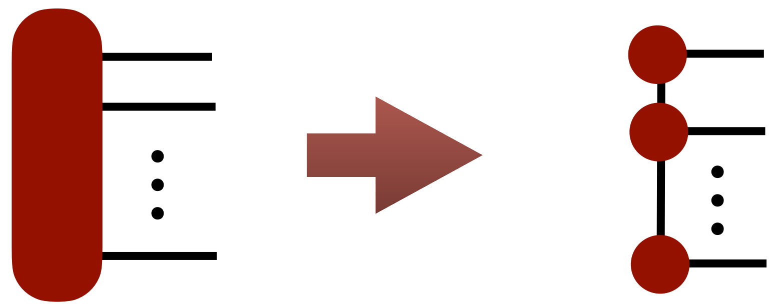
R.N.C. Pfeifer, *et al.*, Phys. Rev. E **90**, 033315 (2014).

多量子ビットゲートの行列積分解

多量子ビットゲートのテンソルネットワーク表現

行列積分解 ：テンソルを小さいテンソルの一次元的なつながりで表現

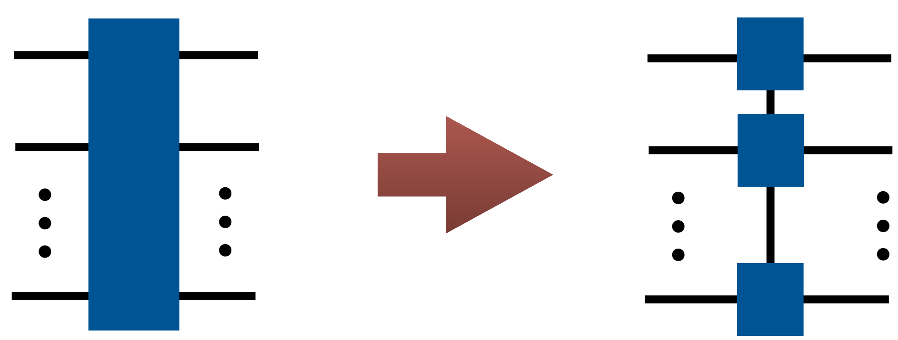
n量子ビット状態



行列積状態
(Matrix Product State)
*各テンソルをつなぐ足の次元
＝ボンド次元 χ

独立要素の数： 2^n $\sim 2n\chi^2$

n量子ビットゲート



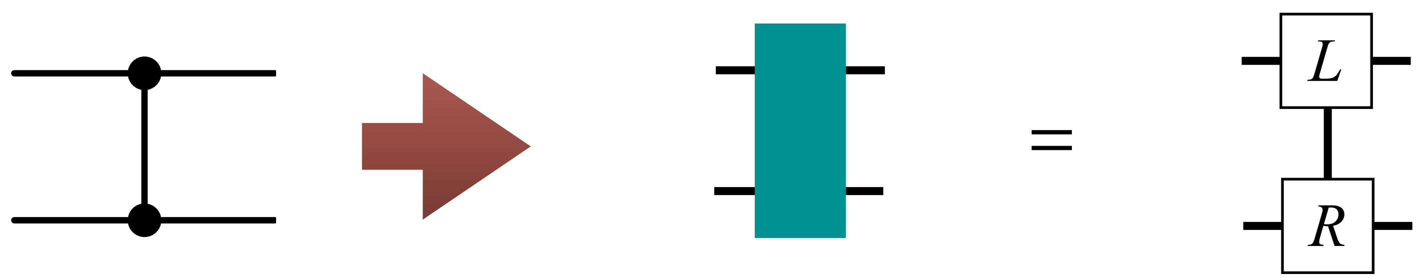
行列積演算子
(Matrix Product Operator)

独立要素の数： 2^{2n} $\sim 4n\chi^2$

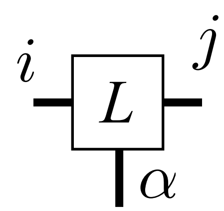
$\boxed{\text{行列積分解は}\chi\text{が小さいと効率的に状態・ゲートを表現できる！}}$

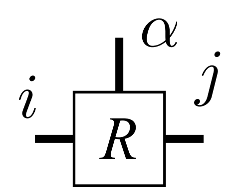
CZゲートの行列積分解

CZゲートはボンド次元χ=2の行列積分解が可能



$$CZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

$L_\alpha^{ij} =$ 

$R_\alpha^{ij} =$ 

$$L_\alpha^{ij} = I_{ij}\underline{\delta_{i\alpha}}$$

αは量子ビットiと同じ状態

$$R_\alpha^{ij} = \underline{I_{ij}\delta_{\alpha 0}} + \underline{Z_{ij}\delta_{\alpha 1}}$$

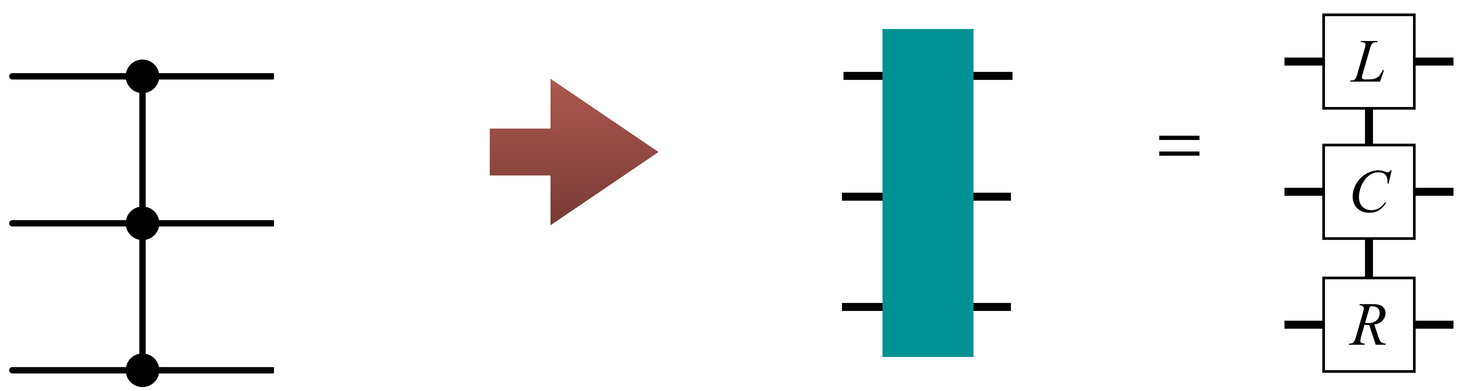
αが0だとIを演算

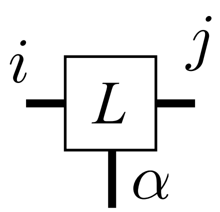
αが1だとZを演算

二つのテンソルをつなぐindex αで、上の量子ビットの情報を運ぶ

3qubit CZゲートの行列積分解

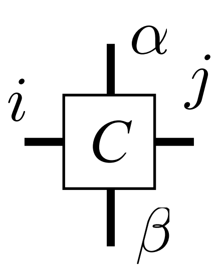
3qubit CZゲートもボンド次元χ=2の行列積分解が可能



$L^{ij}_{\alpha} =$ 

$$L^{ij}_{\alpha} = I_{ij}\delta_{i\alpha}$$

αは量子ビットiと同じ状態

$C^{ij}_{\alpha\beta} =$ 

$$C^{ij}_{\alpha\beta} = I_{ij}\left(\delta_{i1}\delta_{\alpha 1}\delta_{\beta 1} + [1 - \delta_{i1}\delta_{\alpha 1}]\delta_{\beta 0}\right)$$

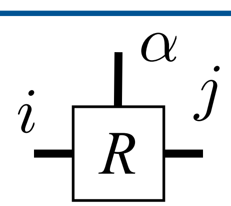
αとiの双方が1だとβも1

(α,i) = (1,1) 以外はβ=0

$\beta = 1$

$\Updownarrow$

上二つの量子ビット状態は$|11\rangle$

$R^{ij}_{\alpha} =$ 

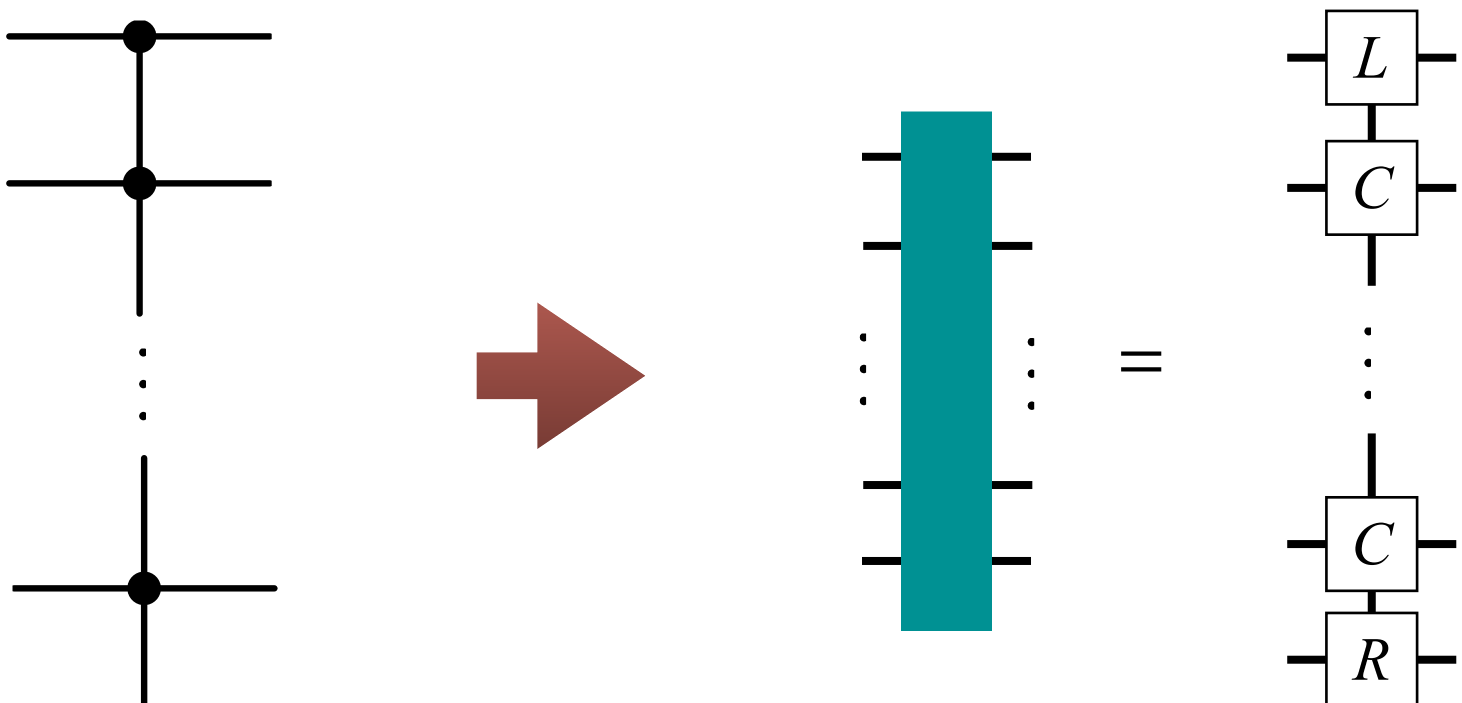
$$R^{ij}_{\alpha} = I_{ij}\delta_{\alpha 0} + Z_{ij}\delta_{\alpha 1}$$

αが0だとIを演算

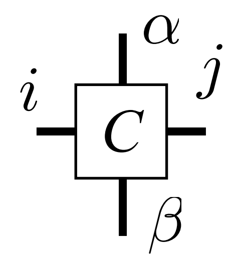
αが1だとZを演算

n qubit CZゲートの行列積分解

n qubit CZゲートはこれまでのL, C, Rを使ってχ=2の行列積分解が可能



$C^{ij}_{\alpha\beta} =$



βが1だとそれより上のビットは全て1

$$R^{ij}_{\alpha} = \underline{I_{ij}\delta_{\alpha 0}} + \underline{Z_{ij}\delta_{\alpha 1}}$$

αが0だとIを演算

αが1だとZを演算

2^{2n} の情報が $\sim 4n\chi^2 = 16n$ の独立成分に圧縮できた！

量子状態のサンプリング

量子ビットの逐次サンプリング

$|i_1 i_2 \cdots i_n\rangle$ の観測確率　$P(i_1, i_2, \cdots, i_n)$

$$P(i_1, i_2, \ldots, i_n) = \underbrace{P(i_1)}_{\text{周辺分布}} \underbrace{P(i_2, i_3, \ldots, i_n | i_1)}_{i_1\text{が決まった時の条件付き確率}}$$

右辺の式を使って状態をサンプリング = まず周辺分布でi_1をサンプリングした後に、残りの量子ビットを条件付き確率で決める

*確率分布に従って状態を生成

条件付き確率も同様に分解

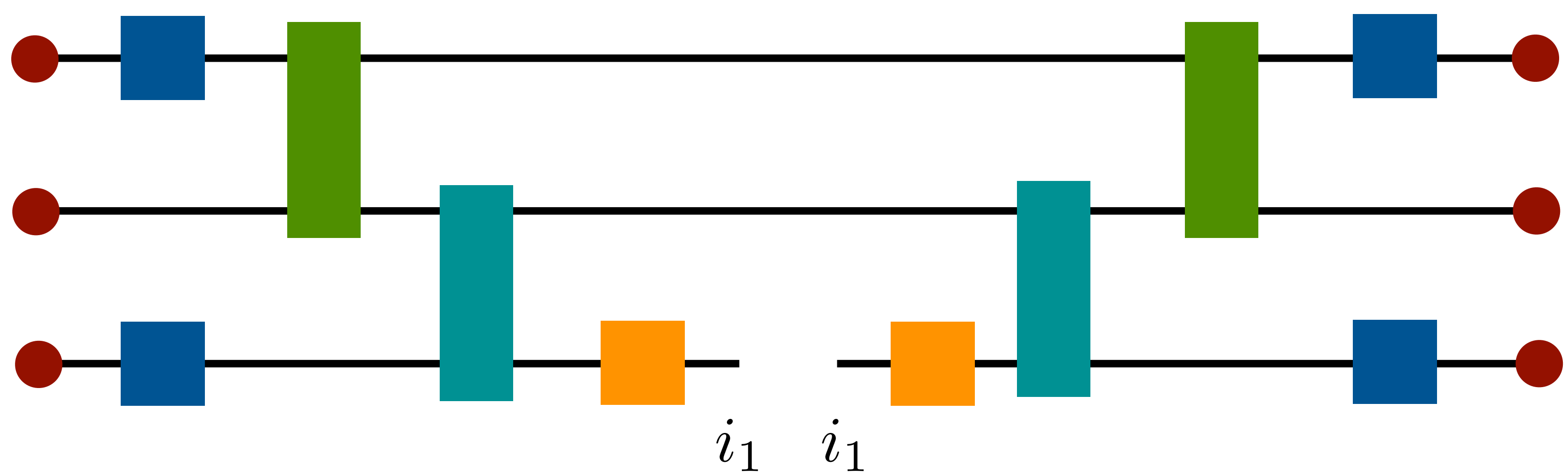
$$\begin{aligned}
P(i_1, i_2, \ldots, i_n) &= P(i_1)P(i_2, i_3, \ldots, i_n | i_1) \\
&= P(i_1)P(i_2|i_1)P(i_3, i_4 \ldots, i_n | i_1, i_2) \\
&= P(i_1)P(i_2|i_1)P(i_3|i_1, i_2)P(i_4, i_5 \ldots, i_n | i_1, i_2, i_3) \\
&= \cdots
\end{aligned}$$

この分解により量子ビットを一つずつ順番にサンプリングすることが可能 → TNの縮約で計算コストを下げられる可能性

量子ビットの逐次サンプリング例：3量子ビット

$$P(i_1, i_2, i_3) = P(i_1)P(i_2|i_1)P(i_3|i_1, i_2)$$

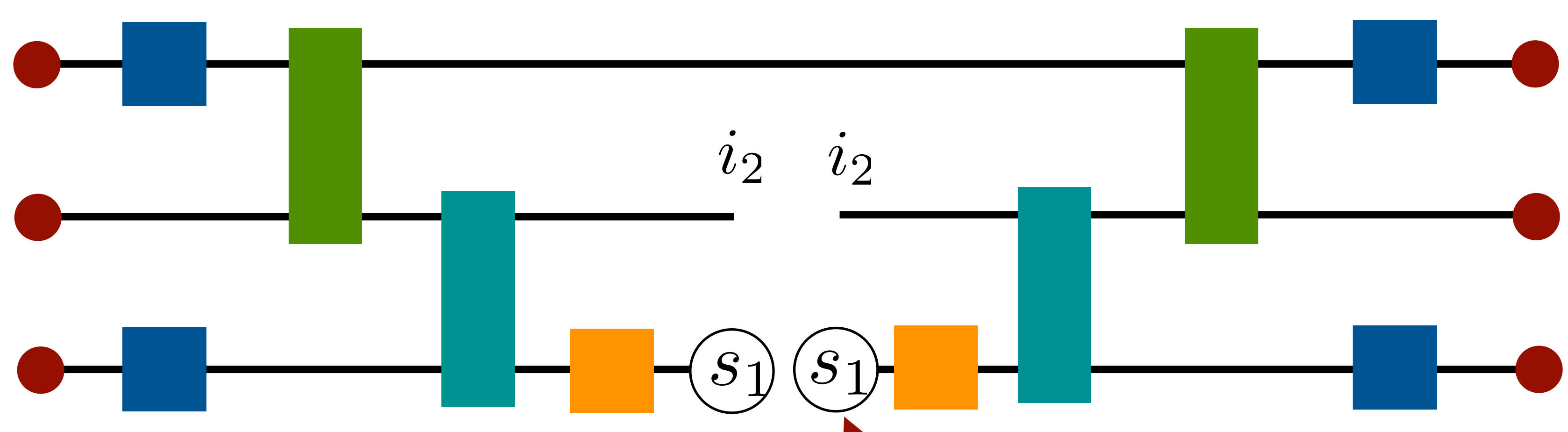
$P(i_1) =$



周辺分布から
s_1が得られた

$s_1 = 0$ or 1

$P(i_2|s_1) =$



この条件付き確率
で2番目のビット
をサンプリング

1番目のビットのサンプリングで
得られた状態に対応するテンソル

まとめ

- 量子計算の概要
  - 典型的な量子ゲートを復習
  - 演習で扱う量子アルゴリズム→Grover & Quantum bit string comparator
- テンソルネットワークによるシミュレーション
  - 量子回路はテンソルネットワークに変換可能
  - 量子状態、確率振幅、縮約密度行列のTNダイアグラム
  - テンソルネットワークの計算コストは縮約順序に依存
  - 多量子ビットゲートを行列積分解にすることでコストが下がる場合がある
  - 量子状態のサンプリングもテンソルネットワーク表現により低コストででできる可能性