本日の演習の概要

- Python上で使えるライブラリ「TensorNetwork」を用いて量子コンピュータのテンソルネットワークシミュレーションを実際に動かしてもらいます
- 演習を通じて、以下の目標を達成することを目指します

本日の目標

1. Python上でのテンソルネットワークによる量子回路シミュレーションの実装方法を学ぶ
2. 実際にいくつかのアルゴリズムを、テンソルネットワークを用いてシミュレーションすることで、テンソルネットワークシミュレーションの強み/弱みについて考察する
3. 量子コンピュータが役立ちそうな場所はどのような場所か（古典コンピュータでは解けない問題は何か）を考え、ディスカッションする

演習の内容

- 前述の目標を達成するため、本日は3つの演習に取り組んでいただきます
- 分からない点などがあれば、積極的に周囲の人と相談しながら取り組んでください
  - 講師側でも適宜サポートに回ります

#	演習テーマ	概要
1	TensorNetworkの使い方	• 最初に、簡単な回路を題材に、ライブラリの使い方の解説を行います。 • その後、実際に自分の手でテンソルネットワークを組み、量子回路のシミュレーションを行う演習課題に取り組んでもらいます。
2	量子アルゴリズムのシミュレーション	• 量子コンパレータ、Groverのアルゴリズムの二つのアルゴリズムについて、通常のシミュレーション（Qiskitを利用）と、テンソルネットワークシミュレーションの違いを体験してもらいます。 • 量子ビット数などのパラメータを変化させながら、各シミュレーション手法の特徴を考察してもらいます。
3	ディスカッション	• 2.で行ったシミュレーション結果を元に、テンソルネットワークの活用や、量子コンピュータの活用について、グループ内でディスカッションをしてもらいます。 • その後、議論した内容を全体に共有してもらいます。

演習環境の準備

- 今回の演習はGoogle Colaboratoryを使って、演習・解説を行います
- 各自のPCのローカル環境上で行っていただいても構いませんが、ご自身の環境に依存するエラー対応については、自己責任でお願いいたします

今回の講義のgithubページに以下の3つのノートブックを掲載してあります。

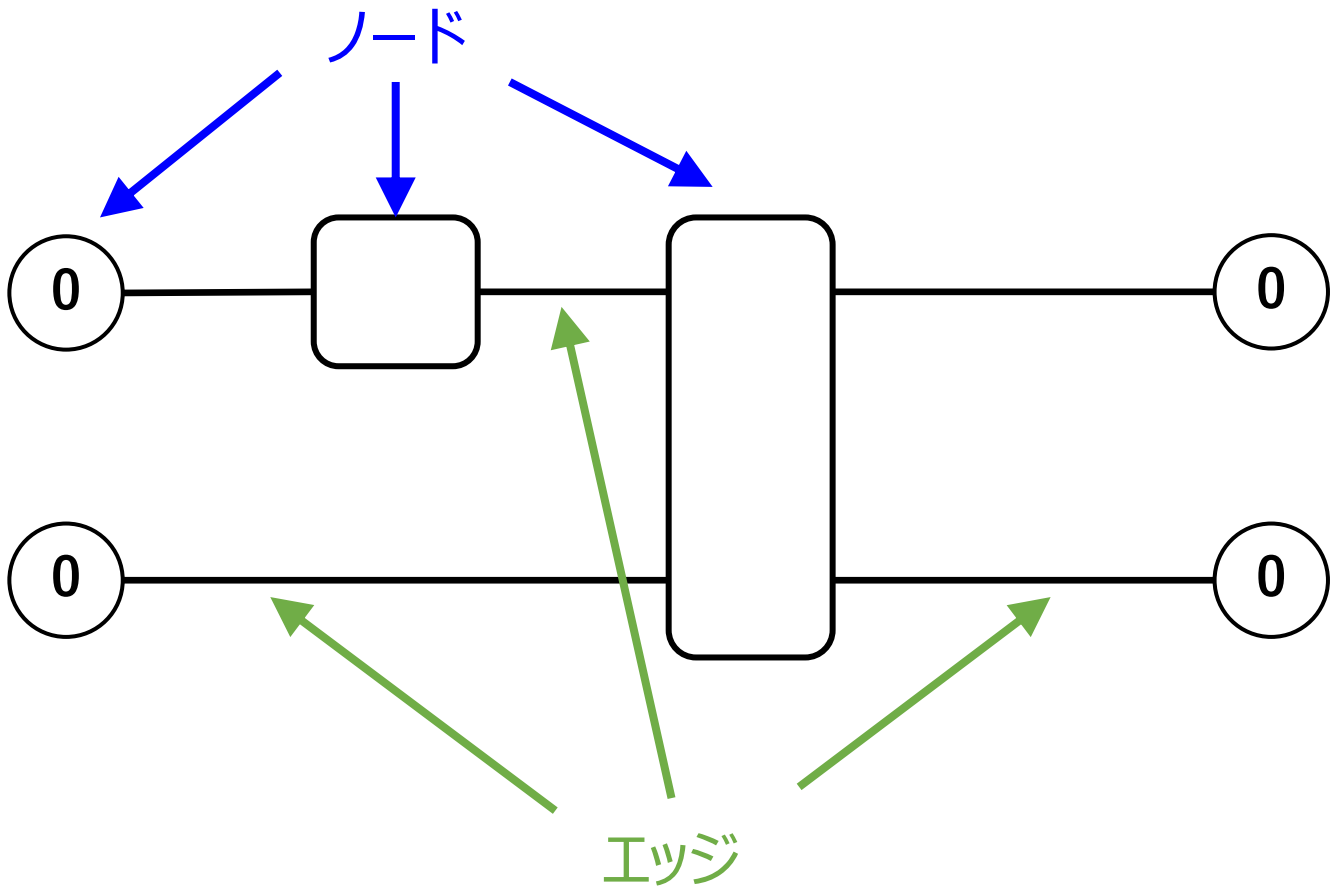
- 「01_tensornetworkの使い方.ipynb」
- 「02_comparatorのシミュレーション.ipynb」
- 「03_groverのシミュレーション.ipynb」

Google Colab環境で「01_tensornetworkの使い方.ipynb」を開いて、エラーなく課題部分の前まで実行できることを確認してください。

ノートブック内の課題部分については、後ほど説明します。

演習1．TensorNetworkライブラリについて

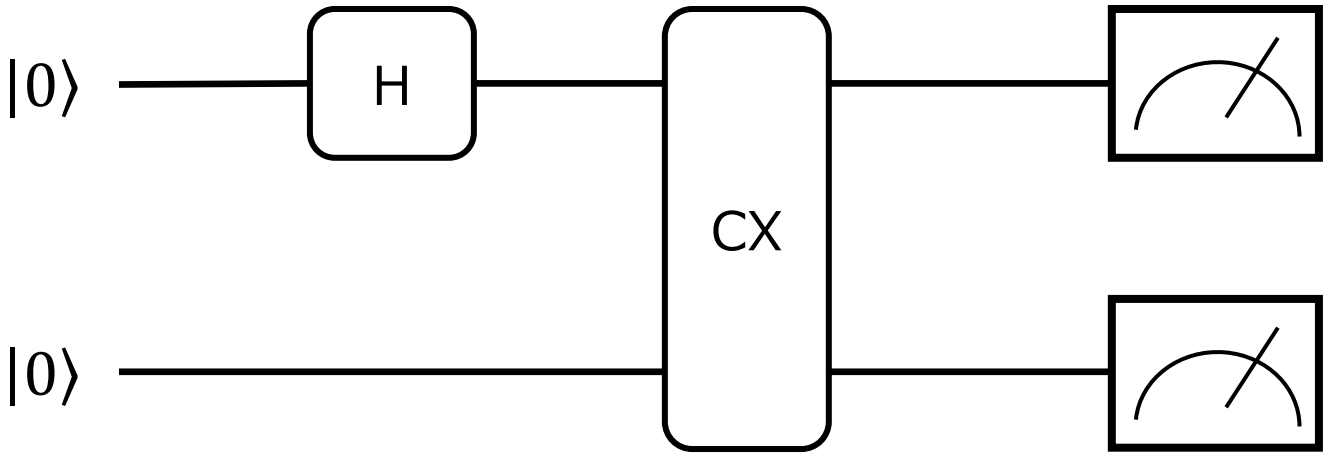
- 「TensorNetwork」はGoogleの作成したテンソルネットワークの計算を行うライブラリ
- ノードとエッジを定義することで、ネットワークの形状を指定することができる
- その後、用意されている関数を利用するだけで、自動で縮約計算を行ってくれる



演習1．簡単な回路の実装

- ベル状態を作成する回路（下図参照）を題材に、実際に実装例を確認する
- この回路を実行すると、|00⟩と|11⟩が等確率で出現する
- まずは特定の状態の振幅を計算する手法を用いて|00⟩が出現する確率が50%であることを確認する

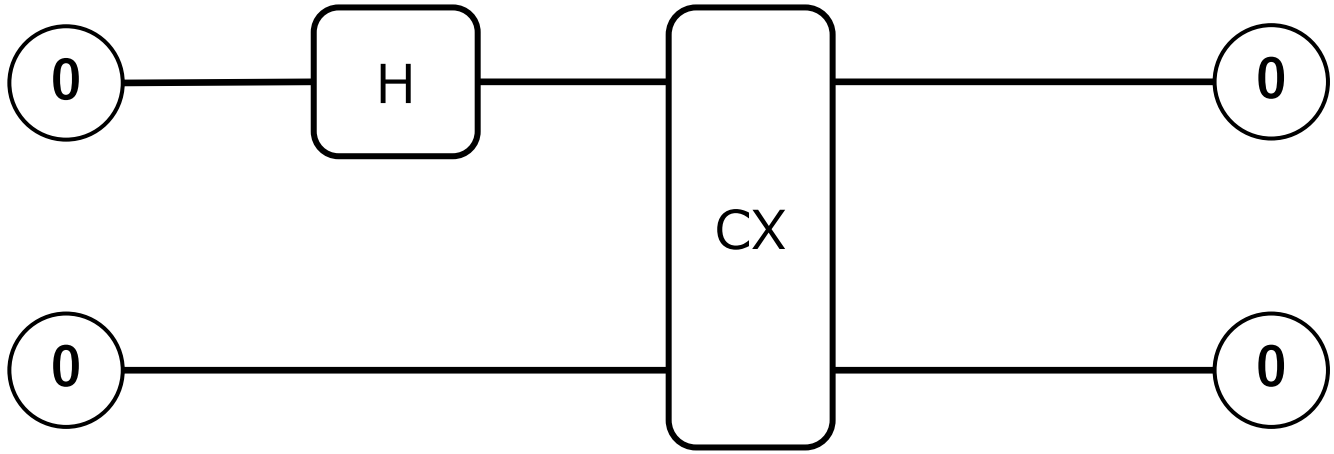
＜ベル状態を作成する回路＞



演習1．簡単な回路の実装

- テンソルネットワークを用いて、|00⟩の振幅を計算する際には、以下のようなテンソルネットワークを構築する
- このテンソルネットワークの縮約を計算することで、振幅が得られる
- 実際にJupyter Notebook上でこのテンソルネットワークを作成し、動作を確認してみる

＜ベル状態を作成する回路に対応するテンソルネットワーク＞



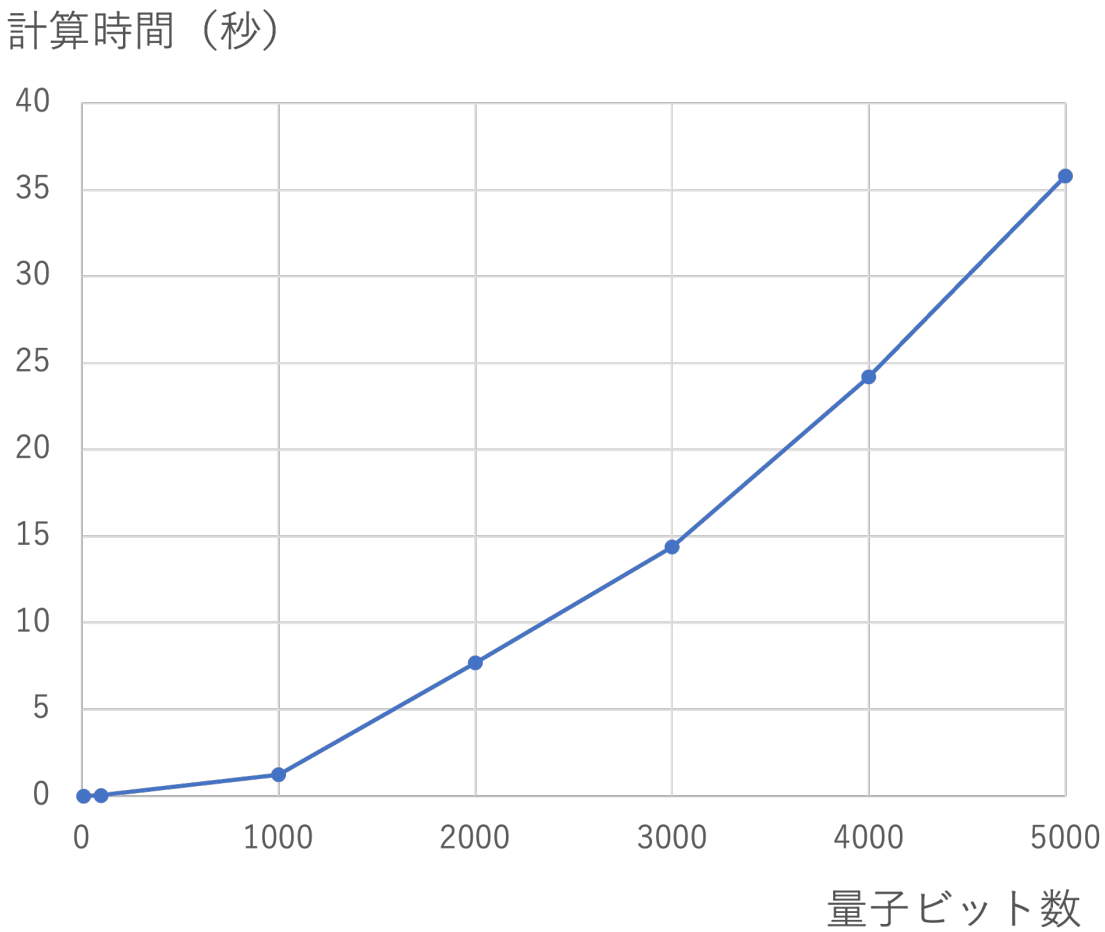
演習1. TensorNetworkの実装演習

- 前述の実装例を参考に、以下の3つの課題に順次取り組む
- 実装の方針に悩むことがあれば、遠慮なく質問してください

#	課題	概要
1	計算する状態の変更	• $\|00\rangle$の出現確率を求める回路を元に、$\|11\rangle$や$\|01\rangle$の出現確率を求めてみましょう。
2	GHZ状態のシミュレーション	• ベル状態を作成する回路のテンソルネットワークを参考に、GHZ状態を作成する回路のテンソルネットワークシミュレーションをやってみましょう。 • GHZ状態：3量子ビットのもつれ状態
3	より多くの量子ビットを用いたもつれ状態作成のシミュレーション	• より一般に、n量子ビットに対して、もつれ状態を作成できるような回路を作成し、テンソルネットワークシミュレーションをやってみましょう。 • 量子ビット数が非常に多くなっても計算できることを確認しましょう。

演習1．TensorNetworkの実装演習

- 量子ビット数を変更しながら、計算時間を計測した結果を以下に記載
- 通常の状態ベクトルシミュレーションでは動かすことのできないビット数でも高速に計算ができている



演習2. 量子アルゴリズムのシミュレーション

- 次に、もう少し複雑な量子アルゴリズムのシミュレーションをテンソルネットワークで行う
- 今回はComparatorとGroverの二つのアルゴリズムを題材に、演習を実施する

Comparator

- 与えられた二つのビット列の大小関係を判定するアルゴリズム
- 重ね合わせ状態にも対応しているため、確率的に大小関係がどうなるかが判別できる

Groverのアルゴリズム

- データの探索（全探索）を行うアルゴリズム
- 古典コンピュータよりも高速に探索ができることが計算理論上証明されている
  - $O(n) \rightarrow O(\sqrt{n})$の二乗加速
- 組合せ探索などにも利用できる可能性がある
  - ただし、二乗加速なので、指数時間かかるものに対しては指数時間のままである

演習2. 量子アルゴリズムのシミュレーション

- 今回は時間の関係もあり、Qiskit、tensornetworkを用いたアルゴリズムの実装はこちらで用意
- 入力するデータや、各種パラメータを変動させながら、計算時間等がどのように変化するかを観察する
- 演習1で扱ったような簡単な回路との違いを確認し、各シミュレーション手法の特徴を考察・ディスカッションする

使用するノートブック

- 「02_comparatorのシミュレーション.ipynb」
- 「03_groverのシミュレーション.ipynb」

それぞれのノートブックを開き、エラーなく実行できることを確認してください。

ディスカッションについて

- Comparator、Groverのアルゴリズムについて、以下の観点で考察し、近くの人とディスカッションしてみましょう

演習のポイント

- 量子ビット数を変化させることによる、計算時間の変化などを確認してみましょう
- Groverのアルゴリズムの反復回数を変化させることで、回路の深さと計算時間の関係を確認してみましょう

ディスカッションのポイント

1. 通常のシミュレーターとテンソルネットワークシミュレーションの向き/不向きについて考察してみましょう
2. 将来、量子コンピューターがどのような問題に対して役に立ちそうか、考えてみましょう