本源
量子

课程二：泡利运算符

QAOA | 泡利算符

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \sigma_{\mathrm{y}} = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

每个泡利矩阵有两个特征值，+1和−1，其对应的归一化特征向量为

$$\psi_{x+} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \psi_{y+} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ \mathrm{i} \end{bmatrix} \qquad \psi_{z+} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\psi_{x-} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad \psi_{y-} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -\mathrm{i} \end{bmatrix} \qquad \psi_{z-} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$|+\rangle \equiv \psi_{x+} \qquad |0\rangle \equiv \psi_{\mathrm{z}+}$$

$$|-\rangle \equiv \psi_{x-} \qquad |1\rangle \equiv \psi_{z-}$$

本 源
量 子

QAOA | 泡利算符

运算规则

$$\sigma_x I = I\sigma_x = \sigma_x$$

$$\sigma_y I = I\sigma_y = \sigma_y$$

$$\sigma_z I = I\sigma_z = \sigma_z$$

$$\sigma_x\sigma_x = \sigma_y\sigma_y = \sigma_z\sigma_z = I$$

$$\sigma_x\sigma_y = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = i\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = i\sigma_z$$

$$\sigma_y\sigma_x = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = -i\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = -i\sigma_z$$

$$\sigma_y\sigma_z = i\sigma_x$$

$$\sigma_z\sigma_y = -i\sigma_x$$

$$\sigma_z\sigma_x = i\sigma_y$$

$$\sigma_x\sigma_z = -i\sigma_y$$

本源量子

QAOA | 泡利算符类

PauliOperator

$$X \equiv \sigma_x \qquad Y \equiv \sigma_y \qquad Z \equiv \sigma_z$$

$$\{"X0", 2\} \equiv 2\sigma_x^0$$

$$\{"", 2\} \equiv 2I$$

$$\{"Z0\ Z1",\ 3\} \equiv 3\sigma_z^0 \otimes \sigma_z^1$$

$$\{"X0\ Y1\ Z2\ Z3",\ 4\} \equiv 4\sigma_x^0 \otimes \sigma_y^1 \otimes \sigma_z^2 \otimes \sigma_z^3$$

本 源
量 子

QAOA | 泡利算符类

PauliOperator

泡利运算符类构造方式

```cpp
#include "Operator/PauliOperator.h"
int main()
{
    using namespace QPanda;
    PauliOperator p1;
    PauliOperator p2({{“Z0 Z1”, 2},{“X1 Y2”, 3}});
    PauliOperator p3(“Z0 Z1”, 2);
    PauliOperator p4(2); // PauliOperator p4(“”, 2);
    PauliOperator p5(p2);

    return 0;
}
```

```python
from pyqpanda import *

if __name__=="__main__":

    p1 = PauliOperator()
    p2 = PauliOperator({'Z0 Z1': 2, 'X1 Y2': 3})
    p3 = PauliOperator('Z0 Z1', 2)
    p4 = PauliOperator(2)
    p5 = p2
```

本 源
量 子

QAOA | 泡利算符类

PauliOperator

运算操作

加、减、乘等常规运算操作

```cpp
#include "Operator/PauliOperator.h"
int main()
{
    using namespace QPanda;
    PauliOperator a("Z0 Z1", 2);
    PauliOperator b("X5 Y6", 3);
    auto plus = a + b;
    auto minus = a - b;
    auto muliply = a * b;

    return 0;
}
```

```python
from pyqpanda import *

if __name__=="__main__":

    a = PauliOperator('Z0 Z1', 2)
    b = PauliOperator('X5 X6', 3)
    plus = a + b
    minus = a - b
    muliply = a * b
```

本源量子

QAOA | 泡利算符类

PauliOperator

打印功能

泡利运算符可以直接被打印出来

```cpp
#include "Operator/PauliOperator.h"
int main()
{
    using namespace QPanda;
    PauliOperator a("Z0 Z1", 2);
    PauliOperator b("X5 Y6", 3);
    auto plus = a + b;
    auto minus = a - b;
    auto multiply = a * b;

    std::cout << "a + b = " << plus << std::endl;
    std::cout << "a - b = " << minus << std::endl;
    std::cout << "a * b = " << multiply << std::endl;

    return 0;
}
```

```python
from pyqpanda import *

if __name__=="__main__":

    a = PauliOperator('Z0 Z1', 2)
    b = PauliOperator('X5 X6', 3)
    plus = a + b
    minus = a - b
    multiply = a * b

    print("a + b = {}".format(plus))
    print("a - b = {}".format(minus))
    print("a * b = {}".format(multiply))
```

```
a + b = {
"X5 X6" : 3.000000
"Z0 Z1" : 2.000000
}
a - b = {
"X5 X6" : -3.000000
"Z0 Z1" : 2.000000
}
a * b = {
"Z0 Z1 X5 X6" : 6.000000
}
```

本源
量子

QAOA | 泡利算符类

PauliOperator

getMaxIndex()

获得泡利运算符使用的最大索引值。如果为空则返回0，否则返回最大下标索引值+1的结果。

```cpp
#include "Operator/PauliOperator.h"
int main()
{
    using namespace QPanda;
    PauliOperator a("Z0 Z1", 2);
    PauliOperator b("X5 Y6", 3);

    auto muliply = a * b;

    std::cout << "a * b = " << muliply << std::endl;
    std::cout << "Index : " << muliply.getMaxIndex();

    return 0;
}
```

```python
from pyqpanda import *

if __name__=="__main__":

    a = PauliOperator('Z0 Z1', 2)
    b = PauliOperator('X5 X6', 3)

    muliply = a * b

    print("a * b = {}".format(muliply))
    print("Index : {}".format(muliply.getMaxIndex()))
```

```
a * b = {
"Z0 Z1 X5 X6" : 6.000000
}
Index : 7
```

本 源
量 子

QAOA | 泡利运算符类

remapQubitIndex()

对泡利运算符中索引从0开始分配映射，并返回新的泡利运算符。

```
#include "Operator/PauliOperator.h"
int main()
{
    using namespace QPanda;
    PauliOperator a("Z0 Z1", 2);
    PauliOperator b("X5 Y6", 3);

    auto muliply = a * b;

    std::map<size_t, size_t> index_map;
    auto remap_pauli = muliply.remapQubitIndex(index_map);
    std::cout << "remap_pauli : " << remap_pauli << std::endl;
    std::cout << "Index : " << remap_pauli.getMaxIndex();

    return 0;
}
```

```
from pyqpanda import *

if __name__=="__main__":

    a = PauliOperator('Z0 Z1', 2)
    b = PauliOperator('X5 X6', 3)

    muliply = a * b

    index_map = {}
    remap_pauli = muliply.remapQubitIndex(index_map)

    print("remap_pauli = {}".format(remap_pauli))
    print("Index : {}".format(remap_pauli.getMaxIndex()))
```

```
remap_pauli = {
"Z0 Z1 X2 X3" : 6.000000
}
Index : 4
```

QAOA | 泡利运算符类

PauliOperator

其它功能

- isEmpyt()　　　// 判空
- dagger()　　　// 返回共轭泡利算符
- isAllPauliZorI() // 判断是否全为泡利“Z”或“I”
- toString()　　　// 返回字符串形式
- data()　　　　// 返回泡利运算符内部维护的数据结构

本 源
量 子

QAOA | 代码演示

本 源
量 子

QAOA | 练习题

$\{"Z0\ Z1", 2\} + \{"Z0\ Z1", 4\} = ?$

$\{"Z0\ Z1", 2\} - \{"Z0\ Z1", 4\} = ?$

$\{"Z0\ Z1", 2\} * \{"Z0\ Z1", 4\} = ?$

$\{"Z0\ Z1", 2\} * \{"X0\ X1", 4\} = ?$

$\{"Z0\ Y1", 2\} * \{"X0\ X1", 4\} = ?$

追本溯源 高掌远跖

支 持 与 交 流

https://github.com/OriginQ/QPanda-2

https://www.originqc.com.cn