Answer

一切虚张声势的狐假虎威。

Choices

Q1: make n-qubits GHZ state

用级联的 CNOT 门产生一串连续 |1>

```
from tiny_q import *

def make_GHZ(n:int=3):
  assert n > 0

  u = H @ get_I(n-1)
  for j in range(n-1):
    u = (get_I(j) @ CNOT @ get_I(n-j-2)) * u

  return u


for j in range(1, 10):
  u = make_GHZ(j)
  q = v('0' * j)
  res = u | q > Measure()
  print({k: v for k, v in res.items() if v > 0})

# => outputs:
# {'0': 509, '1': 491}
# {'00': 480, '11': 520}
# {'000': 487, '111': 513}
# {'0000': 497, '1111': 503}
# {'00000': 500, '11111': 500}
# {'000000': 495, '111111': 505}
# {'0000000': 499, '1111111': 501}
# {'00000000': 522, '11111111': 478}
# {'000000000': 476, '111111111': 524}
```

Q2: read the bio-sim circuit

这啥玩意儿啊？——但是4个编码qubit，1个辅助qubit

```
def. sin(theta_j) = 1 / sqrt(d - j)

=> d = 4
=> sin(theta_2) = 1 / sqrt(4 - 2)
                = 1 / sqrt(2)
```

Q3: entanglement condiftion

纠缠态要求系数分布列行列不独立，即不对应成比例

```
    |psi> = (|0> + b1*e^(i*p1)|1>) (|0> + b0*e^(i*p0)|1>)                       (ignore global phase)
          = |00> + b1*e^(i*p1)|10> + b0*e^(i*p0)|01> + b1*b0*e^(i*(p1+p0))|11>
CNOT|psi> = |00> + b1*e^(i*p1)|11> + b0*e^(i*p0)|01> + b1*b0*e^(i*(p1+p0))|10>    (|1,x> -> |1,~x>)
          = |00> + b0*e^(i*p0)|01> + b1*b0*e^(i*(p1+p0))|10> + b1*e^(i*p1)|11>    (reorder)

[coeff matrix]
q0\q1         |0>               |1>
 |0>           1             b0*e^(i*p0)
 |1>  b1*b0*e^(i*(p1+p0))   b1*e^(i*p1)

b0*e^(i*p0) * b1*b0*e^(i*(p1+p0)) != 1 * b1*e^(i*p1)
           b0^2 * e^(i*(p1+2*p0)) != e^(i*p1)
              b0^2 * e^(i*(2*p0)) != 1
                 (b0*e^(i*p0)) ^2 != 1
                      b0*e^(i*p0) != ±1       (e^α > 0)
                      b0*e^(i*p0) != 1
                       => b0 != 1 and e^(i*p0) != 1
                          b0 != 1 and p0 != 0
```

Q4: read the arithmetic module

跑一下可见 `f(x, y) = x | y`

```python
from tiny_q import *

u = (X @ 3) * CCNOT * (X @ X @ I)

for s in ['00', '01', '10', '11']:
  q = v(s) @ v0
  res = u | q > Measure(300)
  cnt_val = [(cnt, val) for val, cnt in res.items()]
  cnt_val.sort(reverse=True)
  top_val = cnt_val[0][1]
  print(f'f({s}) = {top_val[2:]}')

# => outputs:
# f(00) = 0
# f(01) = 1
# f(10) = 1
# f(11) = 1
```

Q5: inversible computation

可逆计算要求函数是双射，所以做一下单射性检验

对 B 有反例 `f(0, 1) = f(1, 0) = (1, 0)`
对 D 有反例 `f(0, 1) = f(1, 0) = (1, 2)`

```python
D = [0, 1, 2]

f_A = lambda x, y: (y, x)
f_B = lambda x, y: ((x + y) % 3, (x * y) % 3)
f_C = lambda x, y: ((x + y) % 3, (x - y) % 3)
f_D = lambda x, y: ((x + y) % 3, (2*x - y) % 3)

print('The non-invertibles are:')
for name in ['A', 'B', 'C', 'D']:
  f = globals()[f'f_{name}']

  res = {}
  is_dup = False
  for x in D:
    if is_dup: break
    for y in D:
      if is_dup: break

      o = f(x, y)
      if o in res:
        is_dup = True
        break
      else:
        res[o] = (x, y)

  if is_dup: print(name)

# => outputs
# B
# D
```

Q6: measure eigen-what of Y

构造一个门，它的行向量是 Y 的特征向量，这样的测量即以Y的本征态为基。

```
Y = [
  [0, -i],
  [i,  0],
]
eigen value:
  m0 = +1
  m2 = -1
eigen vector:
  y0 = [1,  i]
  y1 = [1, -i]

Take Y's as eigen vector new basis:
  [y0, y1] = [
    [1,  i],
    [1, -i],
  ] / sqrt(2)
This new gate equals to H*S:
  [1  1][1 0]    [1  i]
  [1 -1][0 i] -> [i -i]
```

Q7: QFT unitary matrix

让我康康？但是首先，它必是个对称矩阵。

```
from tiny_q import *

u = QFT(2, False)
print(u.v * 2)

# => outputs:
# [[ 1.+0.j  1.+0.j  1.+0.j  1.+0.j]
#  [ 1.+0.j  0.+1.j -1.+0.j -0.-1.j]
#  [ 1.+0.j -1.+0.j  1.-0.j -1.+0.j]
#  [ 1.+0.j -0.-1.j -1.+0.j  0.+1.j]]
```

Q8: noise & error correcting code

嗯算就是了 😃

```
Obviously, this noise U is actually a Z gate:
  a|0> + b|1> ==[U]==> a|0> - b|1>

Z = [
  [1,  0]   # [1,  0][a] = [ a]
  [0, -1]   # [0, -1][b] = [-b]
]

For any |phi> in the sub-space with base {|s1>, |s2>}:
  |phi> = a|s1> + b|s2>
This noise will turn it to be a mixed state:
  |psi> = ((Z @ I)|phi> + (I @ Z)|phi>) / 2
        = (Z @ I + I @ Z)|phi> / 2
        = N|phi>
  where the gate N unitary is:
    [[1, 0, 0, 0],
     [0, 0, 0, 0],
     [0, 0, 0, 0],
     [0, 0, 0, 1]]

[choice_A]
  |phi> = [a, 0, 0, b]
  |psi> = N|phi> = [a, 0, 0, b] = |phi>

[choice_B]
  |phi> = [a, 0, b, 0]
  |psi> = N|phi> = [a, 0, 0, 0]

[choice_C]
  |phi> = [0, a, 0, b]
  |psi> = N|phi> = [0, 0, 0, b]

[choice_D]
  |phi> = [0, a, b, 0]
  |psi> = N|phi> = [0, 0, 0, 0]
```

Q9: QAOA pauli operator

也还是嗯算就是了 😉

```
The formula:
  xi = (I - Zi) / 2
Hence:
  x1 | x2 = ~(~x1 & ~x2)                (De Morgan's laws)
          = 1 - (1 - x1) * (1 - x2)
          = 1 - (1 - x1 - x2 + x1*x2)
          = x1 + x2 - x1*x2
          = (I - Z1) / 2 + (I - Z2) / 2 - ((I - Z1) / 2) * ((I - Z2) / 2)
          = I - (Z1 + Z2) / 2 - (I - Z1 - Z2 + Z1 * Z2) / 4
          = (3/4) * I - (1/4) * (Z1 + Z2 + Z1 * Z2)
```

Q10: read the Hadamard test circuit

正确的，但是，怎么会事呢？

```
[original]
q = (H @ I) * CU * (H @ Ub) | v('00')
  = (H @ I) * CU * (H @ Ub) |00>
  = (H @ I) * CU * (H|0> @ Ub|0>)
  = (H @ I) * CU * (|+> @ |b>)
  = (H @ I) * CU * ((|0> + |1>) @ |b>)      (ignore global coeff)
  = (H @ I) * CU * (|0b> + |1b>)
  = (H @ I) * (|0b> + |1>@U|b>)
  = (H @ I)|0b> + (H @ I)|1>@U|b>
  = |0b> + |1b> + |0>@U|b> - |1>@U|b>       (ignore global coeff)
  = |0b> + |0>@U|b> + |1b> - |1>@U|b>       (reorder)
  = |0>(|b> + U|b>) + |1>(|b> - U|b>)
  = |0>(I + U)|b> + |1>(I - U)|b>
density(q) = |q><q|
  = (|0>(I + U)|b> + |1>(I - U)|b>)(<b|(I + U)'<0| + <b|(I - U)'<1|)
  = |0>(I + U)|b><b|(I + U)'<0| + |1>(I - U)|b><b|(I - U)'<1|
  = ?

[choice_A]
q = (H @ I) * (X @ I) * C(Ub) * (X @ I) * CU * C(Ua) * (H @ I) | v('00')
  = |0>(|b> + U|a>) + |1>(|b> - U|a>)       (see choice_D)

[choice_B]
q = (H @ I) * C(Ua) * (X @ I) * CU * C(Ub) * (X @ I) * (H @ I) | v('00')
  = (H @ I) * C(Ua) * (X @ I) * CU * C(Ub) * (X @ I) * (H @ I) |00>
  = (H @ I) * C(Ua) * (X @ I) * CU * C(Ub) * (X @ I) * (|00> + |10>)      (ignore global coeff)
  = (H @ I) * C(Ua) * (X @ I) * CU * C(Ub) * (|10> + |00>)
  = (H @ I) * C(Ua) * (X @ I) * CU * (|1>Ub|0> + |00>)
  = (H @ I) * C(Ua) * (X @ I) * CU * (|1b> + |00>)
  = (H @ I) * C(Ua) * (X @ I) * (|1>@U|b> + |00>)
  = (H @ I) * C(Ua) * (|0>@U|b> + |10>)
  = (H @ I) * (|0>@U|b> + |1>Ua|0>)
  = (H @ I) * (|0>@U|b> + |1a>)
  = (H @ I) * |0>@U|b> + (H @ I) * |1a>
  = |0>@U|b> + |1>@U|b> + |0a> - |1a>       (ignore global coeff)
  = |0>@U|b> + |0a> + |1>@U|b> - |1a>       (reorder)
  = |0>(U|b> + |a>) + |1>(U|b> - |a>)

[choice_C]
q = (H @ I) * C(Ua) * CU * C(Ub) * (H @ I) | v('00')
  = (H @ I) * C(Ua) * CU * C(Ub) * (H @ I) |00>
  = (H @ I) * C(Ua) * CU * C(Ub) * (|00> + |10>)     (ignore global coeff)
  = (H @ I) * C(Ua) * CU * (|00> + |1>Ub|0>)
  = (H @ I) * C(Ua) * CU * (|00> + |1b>)
  = (H @ I) * C(Ua) * (|00> + |1>@U|b>)
  = (H @ I) * (|00> + |1>@Ua*U|b>)
  = (H @ I) * |00> + (H @ I) * |1>@Ua*U|b>
  = |00> + |10> + |0>@Ua*U|b> - |1>@Ua*U|b>   (ignore global coeff)
  = |00> + |0>@Ua*U|b> + |10> - |1>@Ua*U|b>   (reorder)
  = |0>(|0> + Ua*U|b>) + |1>(|0> - Ua*U|b>)
```

```
[choice_D]
q = (H @ I) * (X @ I) * C(Ua) * (X @ I) * CU * C(Ub) * (H @ I) | v('00')
  = (H @ I) * (X @ I) * C(Ua) * (X @ I) * CU * C(Ub) * (H @ I) |00>
  = (H @ I) * (X @ I) * C(Ua) * (X @ I) * CU * C(Ub) * (|00> + |10>)      (ignore global coeff)
  = (H @ I) * (X @ I) * C(Ua) * (X @ I) * CU * (|00> + |1>@Ub|0>)
  = (H @ I) * (X @ I) * C(Ua) * (X @ I) * CU * (|00> + |1b>)
  = (H @ I) * (X @ I) * C(Ua) * (X @ I) * (|00> + |1>@U|b>)
  = (H @ I) * (X @ I) * C(Ua) * (|10> + |0>@U|b>)
  = (H @ I) * (X @ I) * (|1>@Ua|0> + |0>@U|b>)
  = (H @ I) * (X @ I) * (|1a> + |0>@U|b>)
  = (H @ I) * (|0a> + |1>@U|b>)
  = (H @ I) * |0a> + (H @ I) * |1>@U|b>
  = |0a> + |1a> + |0>@U|b> - |1>@U|b>       (ignore global coeff)
  = |0a> + |0>@U|b> + |1a> - |1>@U|b>       (reorder)
  = |0>(|a> + U|b>) + |1>(|a> - U|b>)
```

Programs

P1: bell state & teleport circuit

=> see solutions/P1.md

P2: HHL circuit

=> see solutions/P2.md

by Armit

2023/04/01