Quantum Information Set Decoding

장경배

https://youtu.be/YmyEfLjExt8

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

Information Set Decoding(ISD)

- 코드기반 암호에서 가장 효율적이라고 알려진 공격법
  - 주어진 암호문 $\boldsymbol{c}$ 와 공개키$\boldsymbol{G'}$을 이용하여 원본메세지 $\boldsymbol{m}$ 을 복구함

1. n – bit 의 길이 암호문 $\boldsymbol{c}$ 에서 k – bit 의 벡터 $\boldsymbol{c_k}$ 를 랜덤하게 선택

   → 오류 위치를 모르는 상태에서 자신이 선택한 k – bit 벡터에 오류가 포함되지 않아야 함

2. 선택한 열의 index에 맞춰 $\boldsymbol{G'}$ 으로부터 $\boldsymbol{G'_k}$ 를 뽑아낸다. 이때, $\boldsymbol{G'_k}$ 는 invertible
3. $\boldsymbol{c} + \boldsymbol{c_k}\boldsymbol{G'_k}^{-1}\boldsymbol{G}$ 의 weight가 t 와 같거나 더 적은지 확인한다. 그렇지 않으면 1단계 부터 다시 반복
4. 앞의 조건이 만족한다면 원본메세지 $\boldsymbol{m} = \boldsymbol{c_k}\boldsymbol{G'_k}^{-1}$ 로 복구가 가능하다.

Information Set Decoding Example(ISD)

1. n – bit 의 길이 암호문 $\boldsymbol{c}$ 에서 k – bit 의 벡터 $\boldsymbol{c_k}$ 를 랜덤하게 선택
   → 오류 위치를 모르는 상태에서 자신이 선택한 k – bit 벡터에 오류가 포함되지 않아야 함

$$G' = S\,G\,P = \begin{matrix} 1\ 1\ 1\ 1\ 0\ 0\ 0 \\ 1\ 1\ 0\ 0\ 1\ 0\ 0 \\ 1\ 0\ 0\ 1\ 1\ 0\ 1 \\ 0\ 1\ 0\ 1\ 1\ 1\ 0 \end{matrix} \qquad m = 1\ 1\ 0\ 1$$

$$e = 0\ 0\ 0\ 0\ 1\ 0\ 0$$

$$c = 0\ 1\ 1\ 0\ 1\ 1\ 0$$

c_k = 0 1 1 1

$$G'_k = \begin{matrix} 1\ 1\ 1\ 0 \\ 1\ 1\ 0\ 0 \\ 1\ 0\ 0\ 0 \\ 0\ 1\ 0\ 1 \end{matrix}$$

$$G'_k{}^{-1} = \begin{matrix} 0\ 0\ 1\ 0 \\ 0\ 1\ 1\ 0 \\ 1\ 1\ 0\ 0 \\ 0\ 1\ 1\ 1 \end{matrix}$$

밑의 결과 벡터의 weight를 확인한다. (t or less)

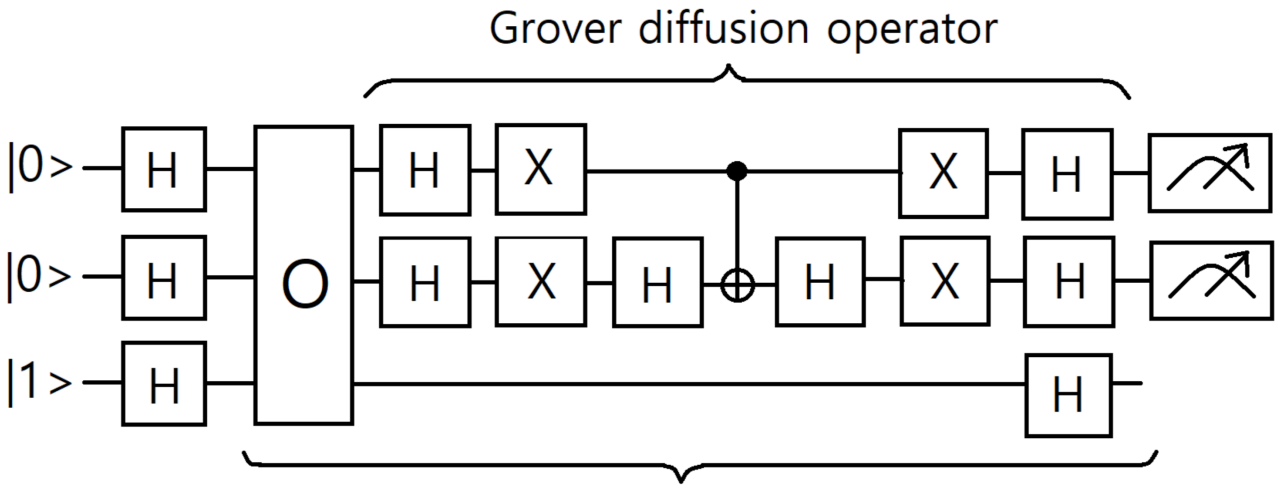
$c + c_k G'_k{}^{-1} G$ = 0 1 1 0 1 1 0 + 0 1 1 0 0 1 0 = 0 0 0 0 1 0 0

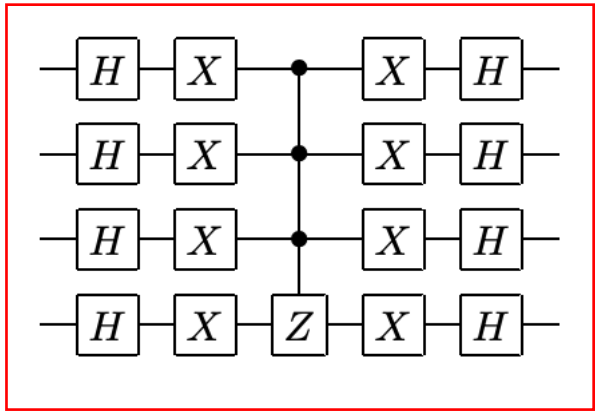
마지막으로 최종 메세지 m = $c_k G'_k{}^{-1}$ = 1 1 0 1

$n = 1024,\quad k = 524$ 의 $Goppa$ 코드에 대하여
이 경우의 수는 $2^{80.7}$ 결과의 작업 계수

Grover

- 정렬되지 않은 데이터베이스로부터 특정 데이터를 찾는 양자 알고리즘
- Classic 한 알고리즘에선 최대 N 번의 시도가 필요하지만, Grover 알고리즘은 $\sqrt{N}$ 번이면 가능하다.





diffusion operator

Grover

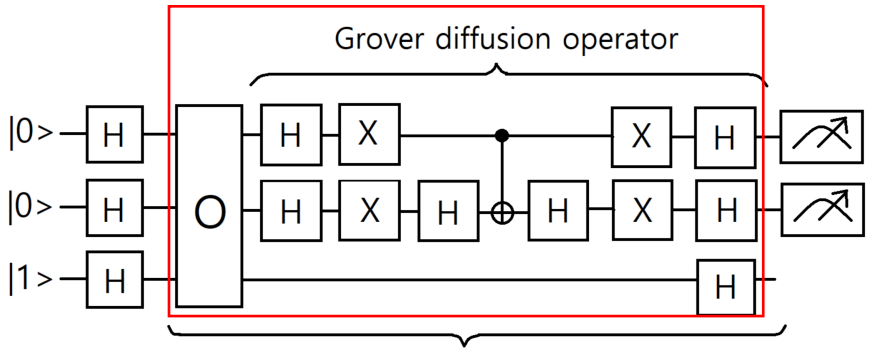
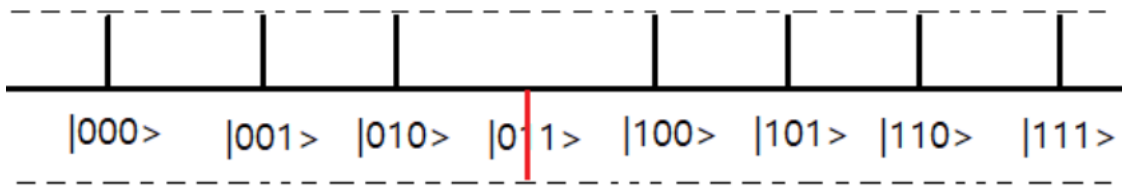
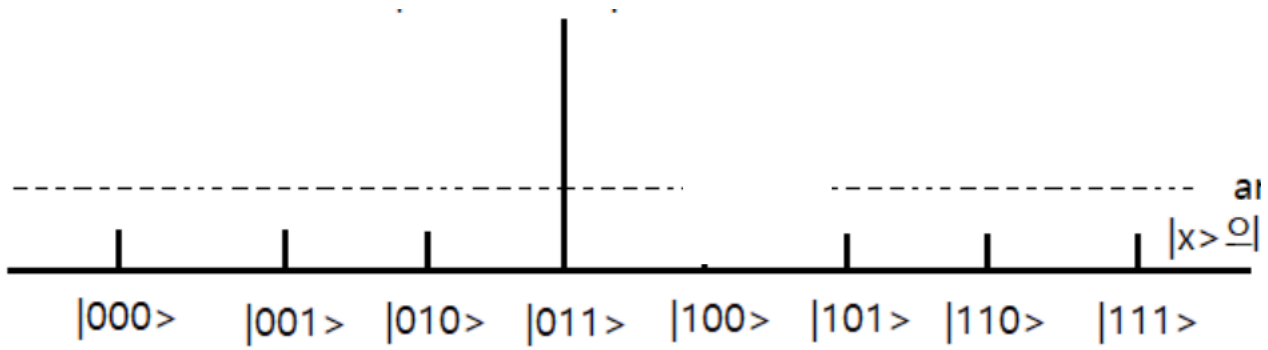


Fig. Grover algorithm (n=2)



Oracle 적용 (n=3)



Diffusion operator (n=3) 적용

Grover

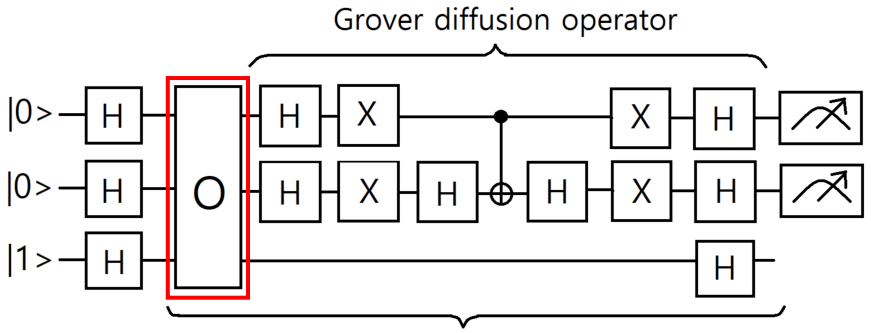
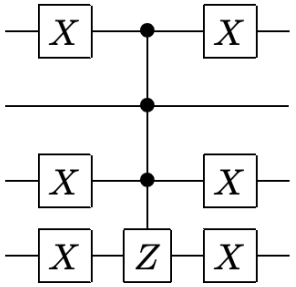


Fig. Grover algorithm (n=2)

Oracle

The oracle function performs a phase flip on the marked state. The phase flip inverts the amplitude α_{0010} of the state, making it $-\frac{1}{4} = -0.25$. Below the oracle for the state $|0010\rangle$ is shown. The corresponding QISKit code can be found in Appendix A.2, and a list of all oracles in Appendix B.1.



Quantum ISD

- Grover Oracle + ISD

Challenge (Classic Mceliece encoding)

$C_0 = He^T$ 라는 신드롬 계산 식에서 C_0 와 H 가 주어진다 해도

low – weight 벡터 e 를 찾아내기 매우 어려움→ Finding low-weight codeword problem

$$G' = SGP = \begin{matrix} 1\ 1\ 1\ 1\ 0\ 0\ 0 \\ 1\ 1\ 0\ 0\ 1\ 0\ 0 \\ 1\ 0\ 0\ 1\ 1\ 0\ 1 \\ 0\ 1\ 0\ 1\ 1\ 1\ 0 \end{matrix} \qquad m = 1\ 1\ 0\ 1$$

$$e = 0\ 0\ 0\ 0\ 1\ 0\ 0$$

$$c = 0\ 1\ 1\ 0\ 1\ 1\ 0$$

$c_k = 0\ 1\ 1\ 1$

$$G'_k = \begin{matrix} 1\ 1\ 1\ 0 \\ 1\ 1\ 0\ 0 \\ 1\ 0\ 0\ 0 \\ 0\ 1\ 0\ 1 \end{matrix}$$

Quantum ISD

Challenge

$C_0 = He^T$ 라는 신드롬 계산 식에서 C_0 와 H 가 주어진다 해도

low – weight 벡터 e 를 찾아내기 매우 어려움→ Finding low-weight codeword problem

ISD Challenge

1 1 1 0
1 1 0 0
1 0 0 0
0 1 0 1

H_k

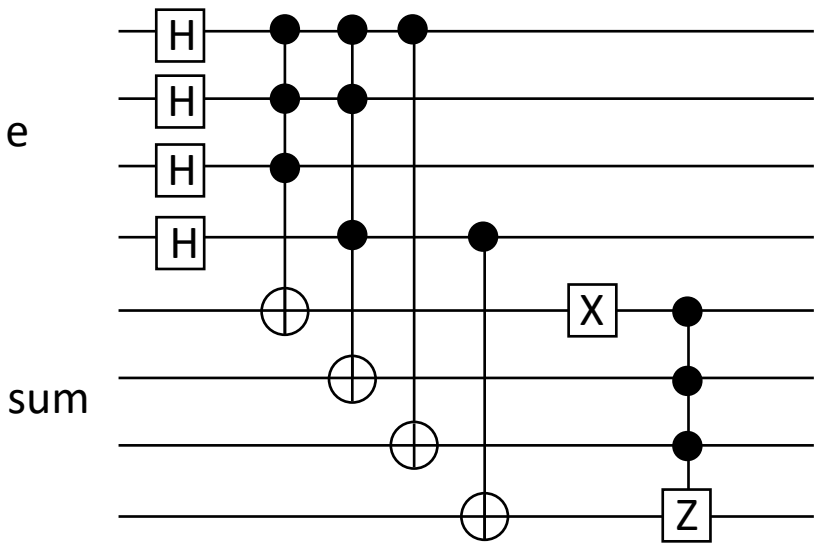
c_k = 0 1 1 1

$c_k = H_k\, e$ 를 만족하는 특정 weight의 벡터 e 찾기

Grover Oracle

1 1 1 0
1 1 0 0
1 0 0 0
0 1 0 1

H_k



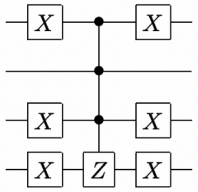
$c_k = 0\,1\,1\,1$

$c_k\, H_k^{-1} = e$

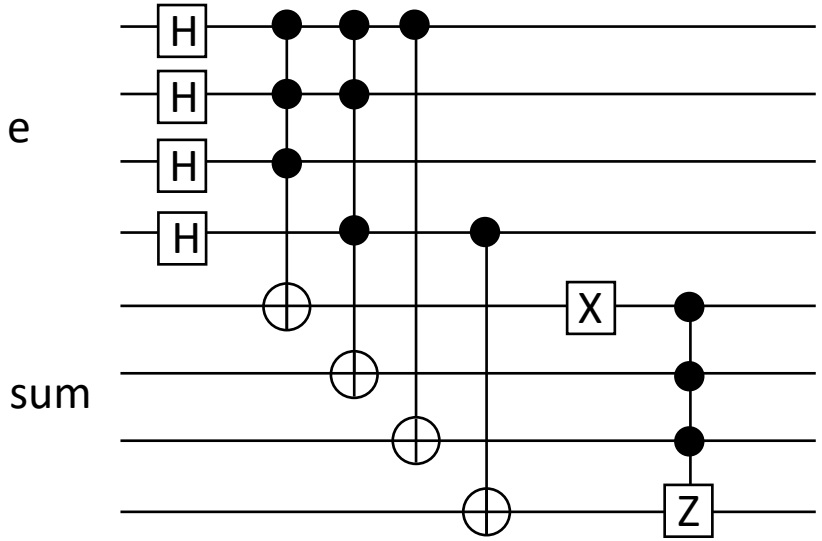
*

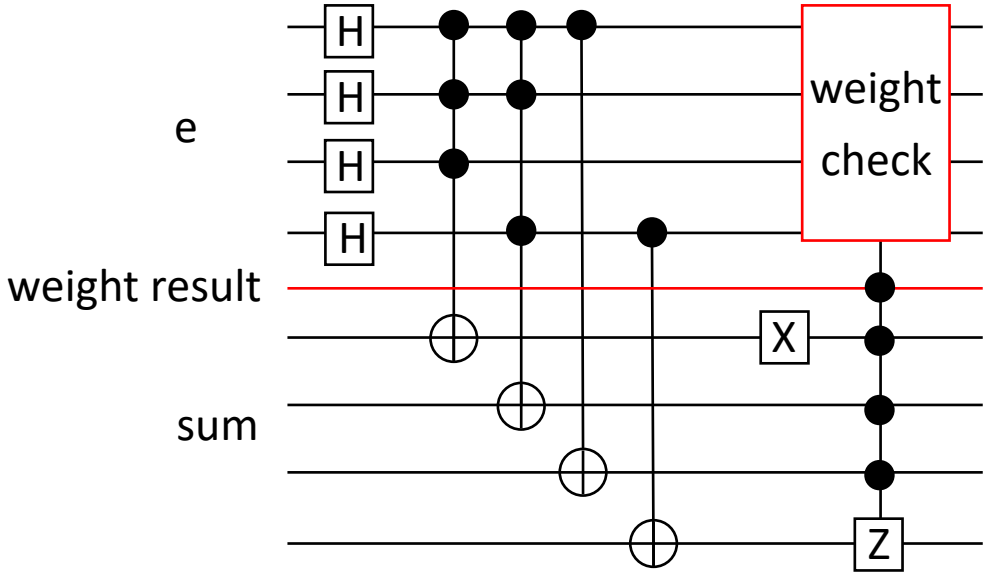
Oracle

The oracle function performs a phase flip on the marked state. The phase flip inverts the amplitude α_{0010} of the state, making it $-\frac{1}{4} = -0.25$. Below the oracle for the state $|0010\rangle$ is shown. The corresponding QISKit code can be found in Appendix A.2, and a list of all oracles in Appendix B.1.



Grover Oracle

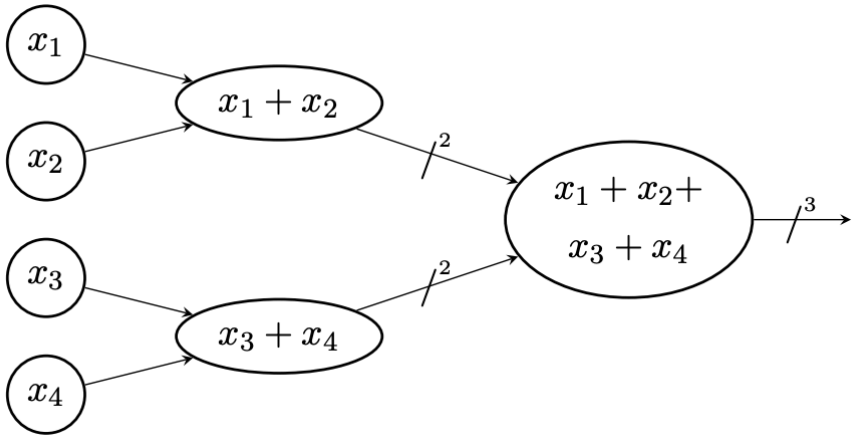


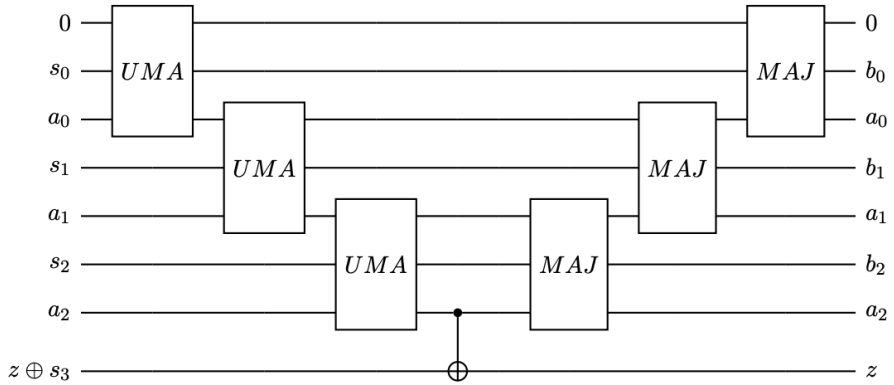


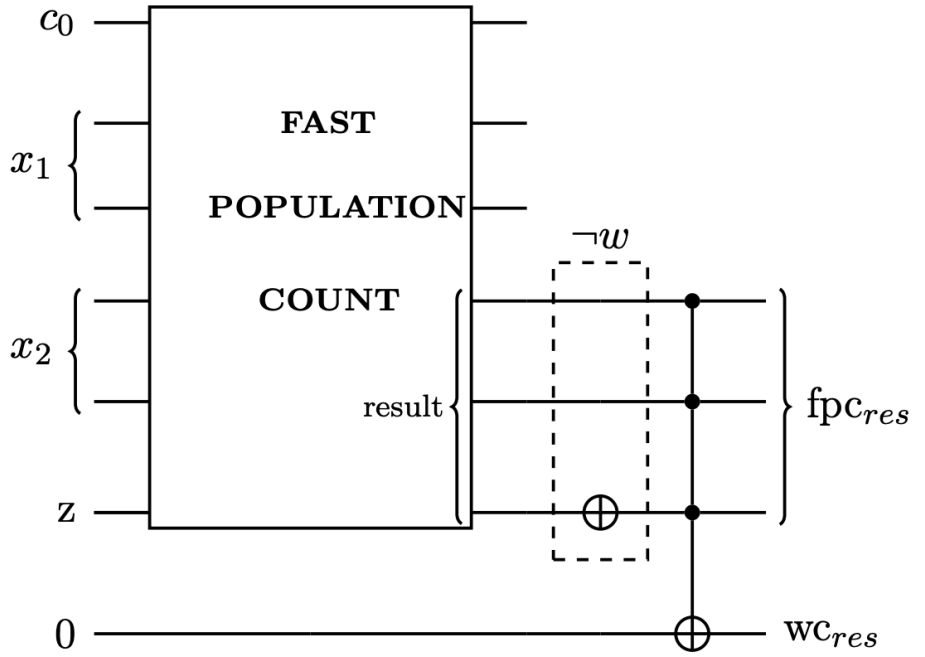
$c_k = H_k\ e$ 를 만족하는 특정 weight의 벡터 e 찾기

Grover Oracle

Hamming weight 확인 (4-bit)

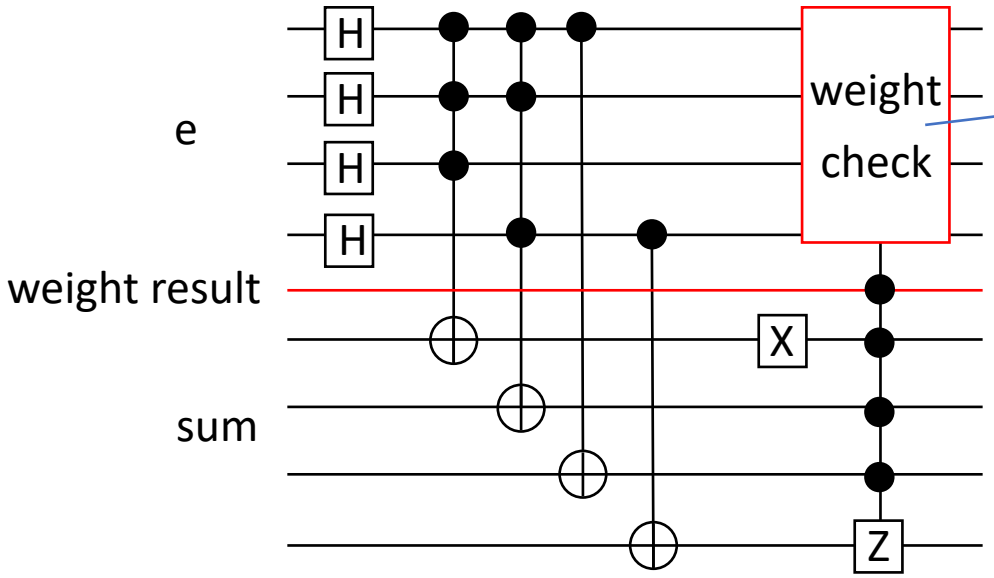


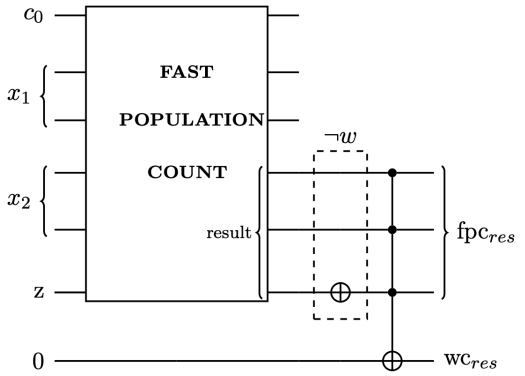




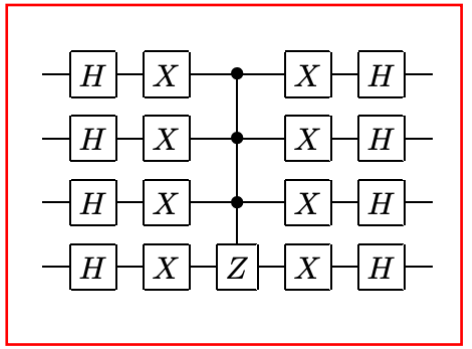
weight 가 011 (3) 인지 확인하는 회로

Grover Oracle + Diffusion





+



diffusion operator

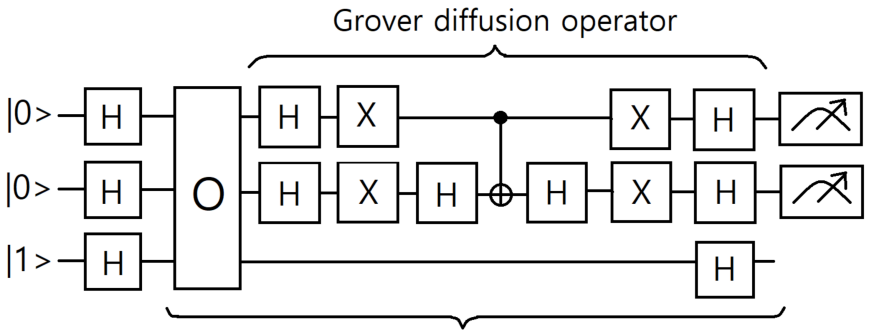
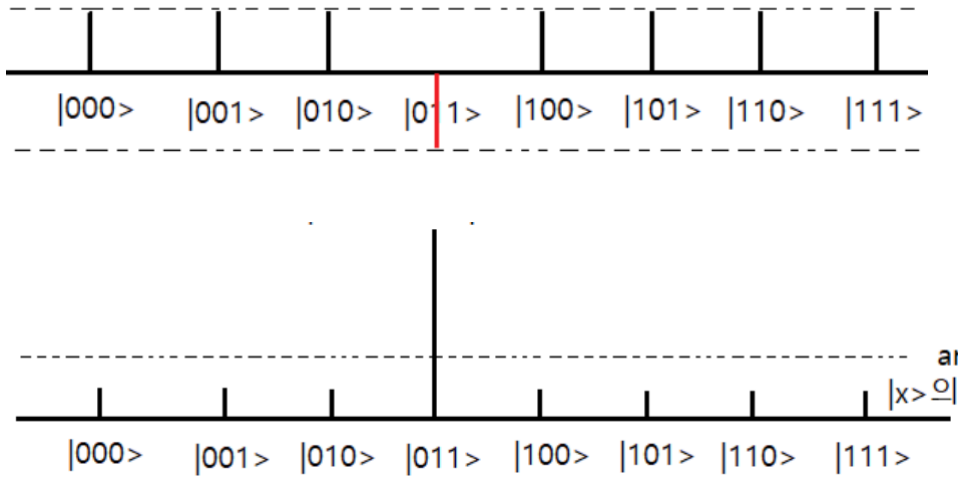


Fig. Grover algorithm (n=2)

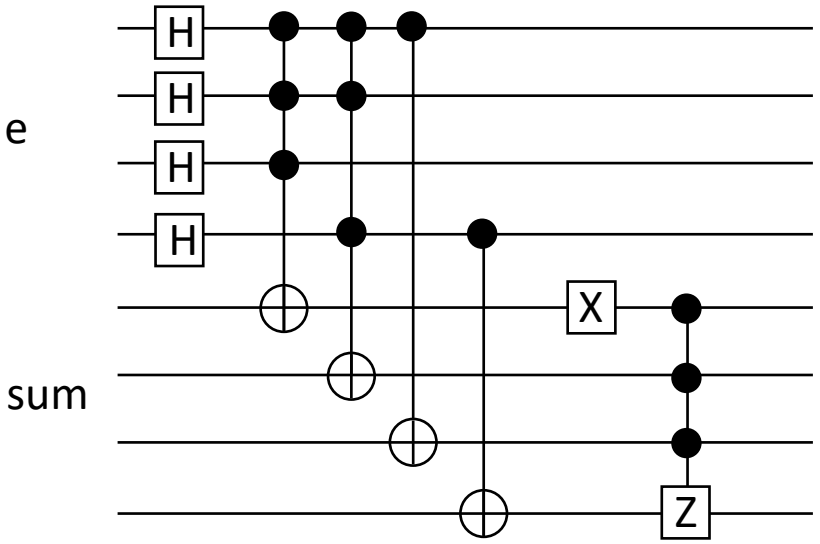


Grover Oracle

ISD Challenge

$$H_k = \begin{matrix} 1\,1\,1\,0 \\ 1\,1\,0\,0 \\ 1\,0\,0\,0 \\ 0\,1\,0\,1 \end{matrix}$$

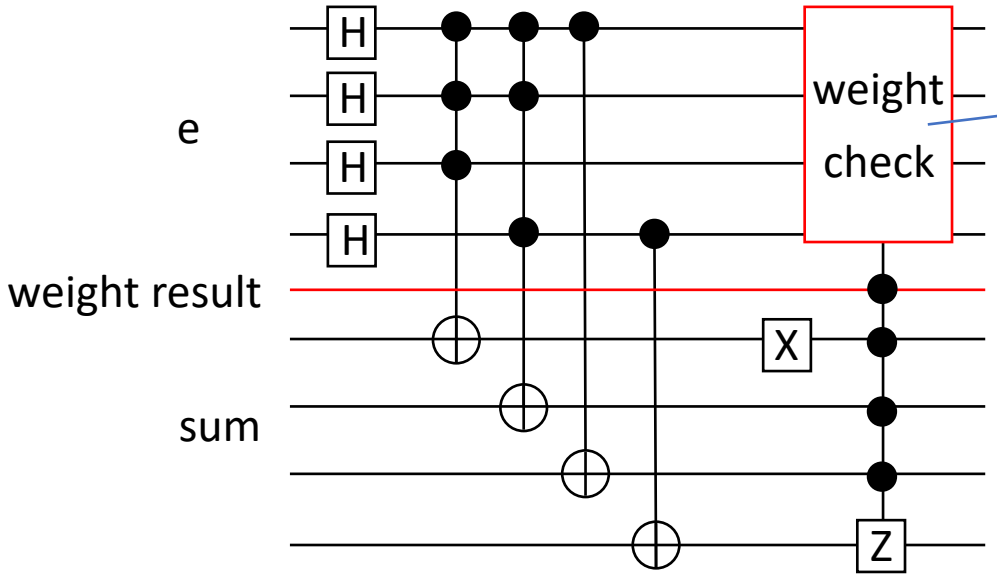
$$c_k = 0\,1\,1\,1$$

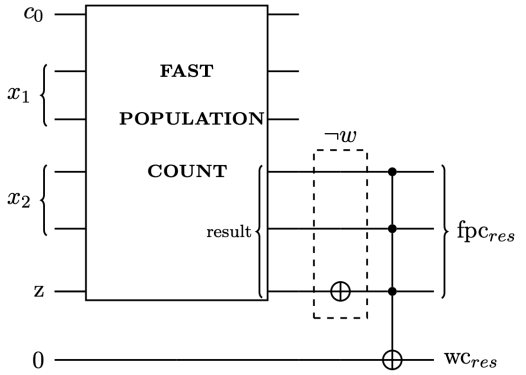


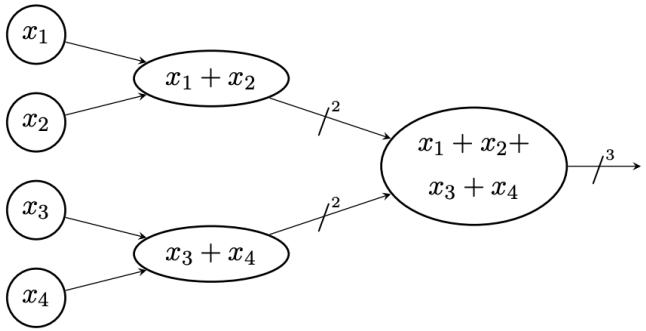

```
def ISD(eng):

    mat = eng.allocate_qureg(4) # vector e
    target = eng.allocate_qureg(4) # syndrome s

    c = eng.allocate_qubit()  # ripple carry bit
    z = eng.allocate_qureg(3) # 4-bit vector for weight check

    w_res = eng.allocate_qubit() # weight result
    n = 15

    All(H) | mat

    with Loop(eng, n):

        # oracle
        Set_matrix(eng, mat, target, c, z, w_res)
```

```
def Set_matrix(eng, mat, target,c, z,w_res):

    with Compute(eng):
        CNOT | (mat[0], target[0]) #1
        CNOT | (mat[1], target[0]) #0
        CNOT | (mat[2], target[0]) #0

        CNOT | (mat[0], target[1]) #1
        CNOT | (mat[1], target[1]) #0
        CNOT | (mat[3], target[1]) #1

        CNOT | (mat[0], target[2]) #1

        CNOT | (mat[3], target[3]) #1

        X | (target[0])

        H_weight(eng, mat, c, z, w_res)

    with Control(eng, w_res):
        with Control(eng, target[0:-1]):
            Z | target[-1]

    Uncompute(eng)
```

Grover Oracle

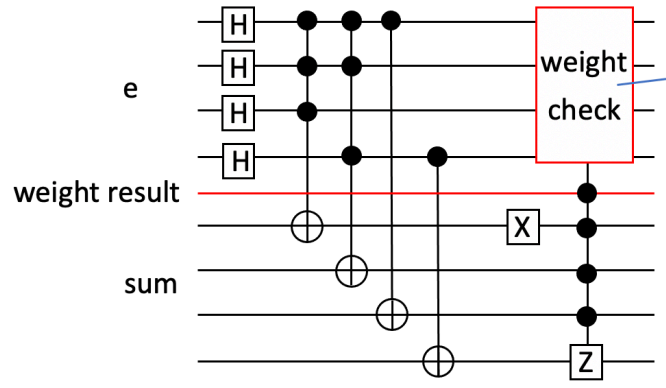


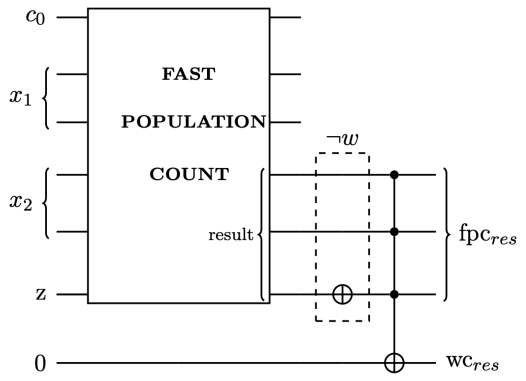




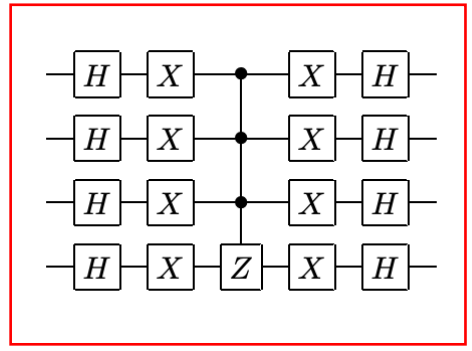

```
def H_weight(eng, vector, c, z, w_res):
    Toffoli | (vector[0], vector[1], z[0])
    CNOT | (vector[0], vector[1])  # z1, vector[1] -> 0 1

    Toffoli | (vector[2], vector[3], z[1])
    CNOT | (vector[2], vector[3])  # z2, vector[3] -> 0 1

    MAJ(eng, vector[1], vector[3], c)
    MAJ(eng, z[0], z[1], vector[1])
    CNOT | (z[0], z[2])

    UMA(eng, z[0], z[1], vector[1])
    UMA(eng, vector[1], vector[3], c)      #z2 z1 v[3]

    X | z[2] # set weight condition : 011

    with Control(eng, z[2]):
        with Control(eng, z[1]):
            with Control(eng, vector[3]):
                X | w_res
```

Quantum ISD





+



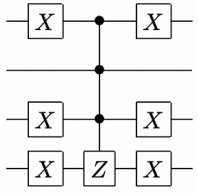
diffusion operator

```
def ISD(eng):
    mat = eng.allocate_qureg(4) # vector e
    target = eng.allocate_qureg(4) # syndrome s

    c = eng.allocate_qubit()  # ripple carry bit
    z = eng.allocate_qureg(3) # 4-bit vector for weight check

    w_res = eng.allocate_qubit() # weight result
    n = 15

    All(H) | mat

    with Loop(eng, n):

        # oracle
        Set_matrix(eng, mat, target, c, z, w_res)

        # Diffusion
        with Compute(eng):
            All(H) | mat
            # All(H) | target
            All(X) | mat
            # All(X) | target

        with Control(eng, mat[0:-1]):
            # with Control(eng, target[0:-1]):
            Z | mat[-1]

        Uncompute(eng)
```

```
def Set_matrix(eng, mat, target,c, z,w_res):

    with Compute(eng):
        CNOT | (mat[0], target[0]) #1
        CNOT | (mat[1], target[0]) #0
        CNOT | (mat[2], target[0]) #0

        CNOT | (mat[0], target[1]) #1
        CNOT | (mat[1], target[1]) #0
        CNOT | (mat[3], target[1]) #1

        CNOT | (mat[0], target[2]) #1

        CNOT | (mat[3], target[3]) #1

        X | (target[0])

        H_weight(eng, mat, c, z, w_res)

    with Control(eng, w_res):
        with Control(eng, target[0:-1]):
            Z | target[-1]

    Uncompute(eng)
```

* **Oracle**

The oracle function performs a phase flip on the marked state. The phase flip inverts the amplitude α_{0010} of the state, making it $-\frac{1}{4} = -0.25$. Below the oracle for the state $|0010\rangle$ is shown. The corresponding QISKit code can be found in Appendix A.2, and a list of all oracles in Appendix B.1.



Quantum ISD

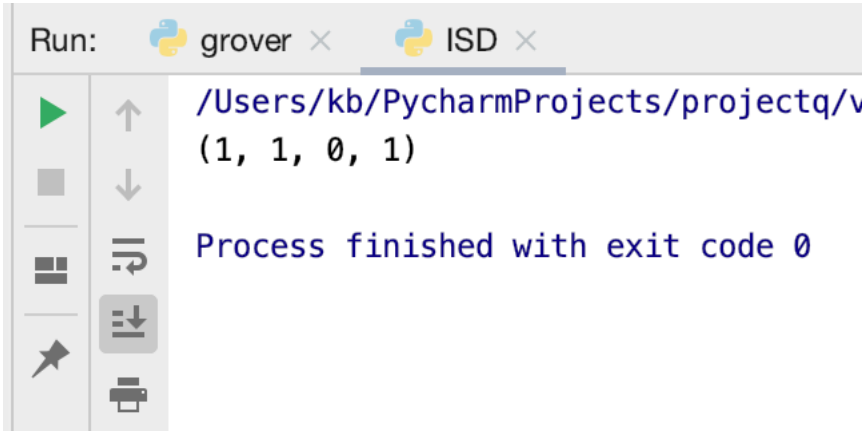
$$G' = S\,G\,P = \begin{matrix} 1\ 1\ 1\ 1\ 0\ 0\ 0 \\ 1\ 1\ 0\ 0\ 1\ 0\ 0 \\ 1\ 0\ 0\ 1\ 1\ 0\ 1 \\ 0\ 1\ 0\ 1\ 1\ 1\ 0 \end{matrix} \qquad m = 1\ 1\ 0\ 1$$

$$e = 0\ 0\ 0\ 0\ 1\ 0\ 0 \qquad c = 0\ 1\ 1\ 0\ 1\ 1\ 0$$

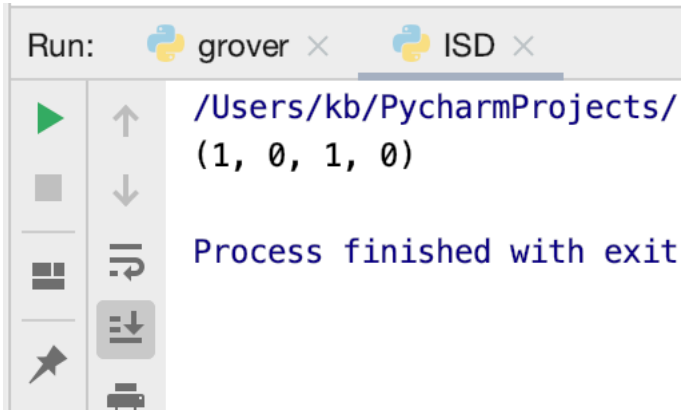
$c_k = 0\ 1\ 1\ 1$

$$G'_k = \begin{matrix} 1\ 1\ 1\ 0 \\ 1\ 1\ 0\ 0 \\ 1\ 0\ 0\ 0 \\ 0\ 1\ 0\ 1 \end{matrix}$$

결과 : 1 1 0 1 (weight = 3, n=3)



결과 : n=1 → 1 1 0 1 이 자주 나오나, 다른 값도 나옴



감사합니다