양자 컴퓨팅 소개

오학주

고려대학교 정보대학 컴퓨터학과

2021.6.8

소프트웨어 오류 문제

- SW 오류 = 사회 모든 영역에서 발생

금융거래SW결함(2012)

항공전산SW결함(2017)

자율주행SW결함(2017)

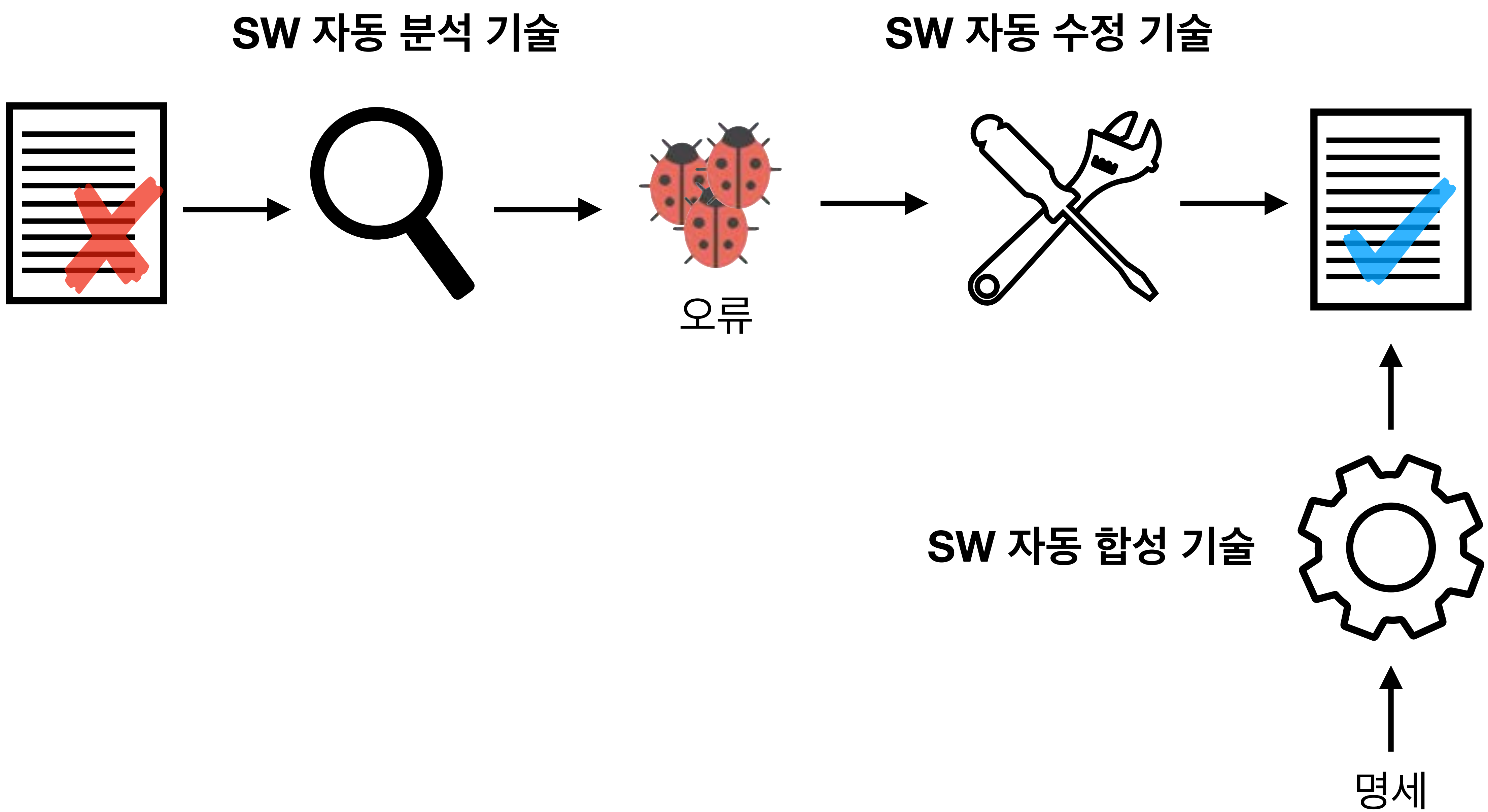
의료SW결함(2018)

- SW 오류 = 사회경제적 비용 1.7조 달러/년

Software fail watch (5th edition). 2017

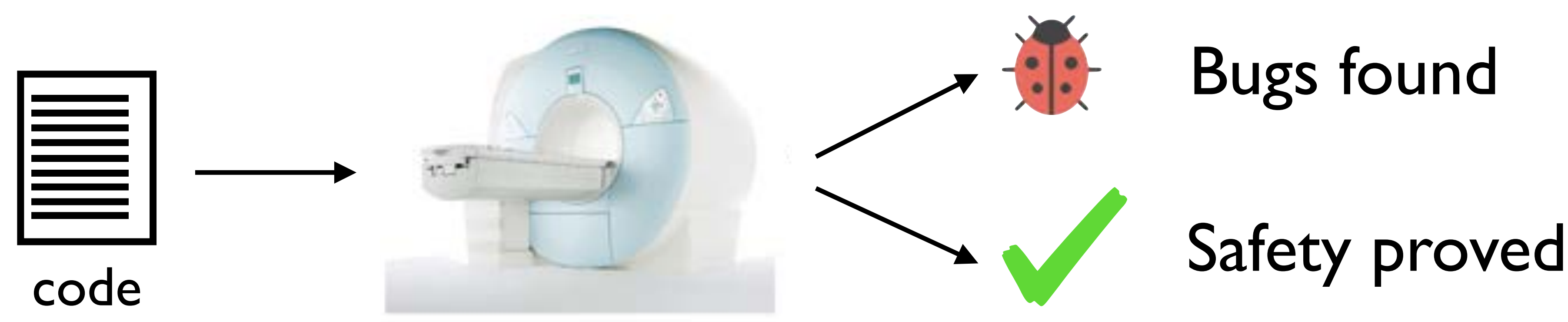
연구 분야

- SW 오류 **자동 분석** + **자동 수정** + **자동 합성**



소프트웨어 자동 분석 기술

"소프트웨어 MRI"



- 소프트웨어의 실행 성질을 엄밀히 확인하는 기술
  - 정적 분석: 실행 전 확인 (요약 해석, 모델 체킹 등)
  - 동적 분석: 실행 중 확인 (퍼징, 기호 실행 등)
- 소프트웨어 산업에서 적극적으로 활용되기 시작

프로그램 분석 사례 (Linux Kernel)

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {

  goto err;
}

out = malloc(2);
if (out == NULL) {
  free(in);

  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

프로그램 분석 사례 (Linux Kernel)

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {

  goto err;
}

out = malloc(2);
if (out == NULL) {
  free(in);

  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

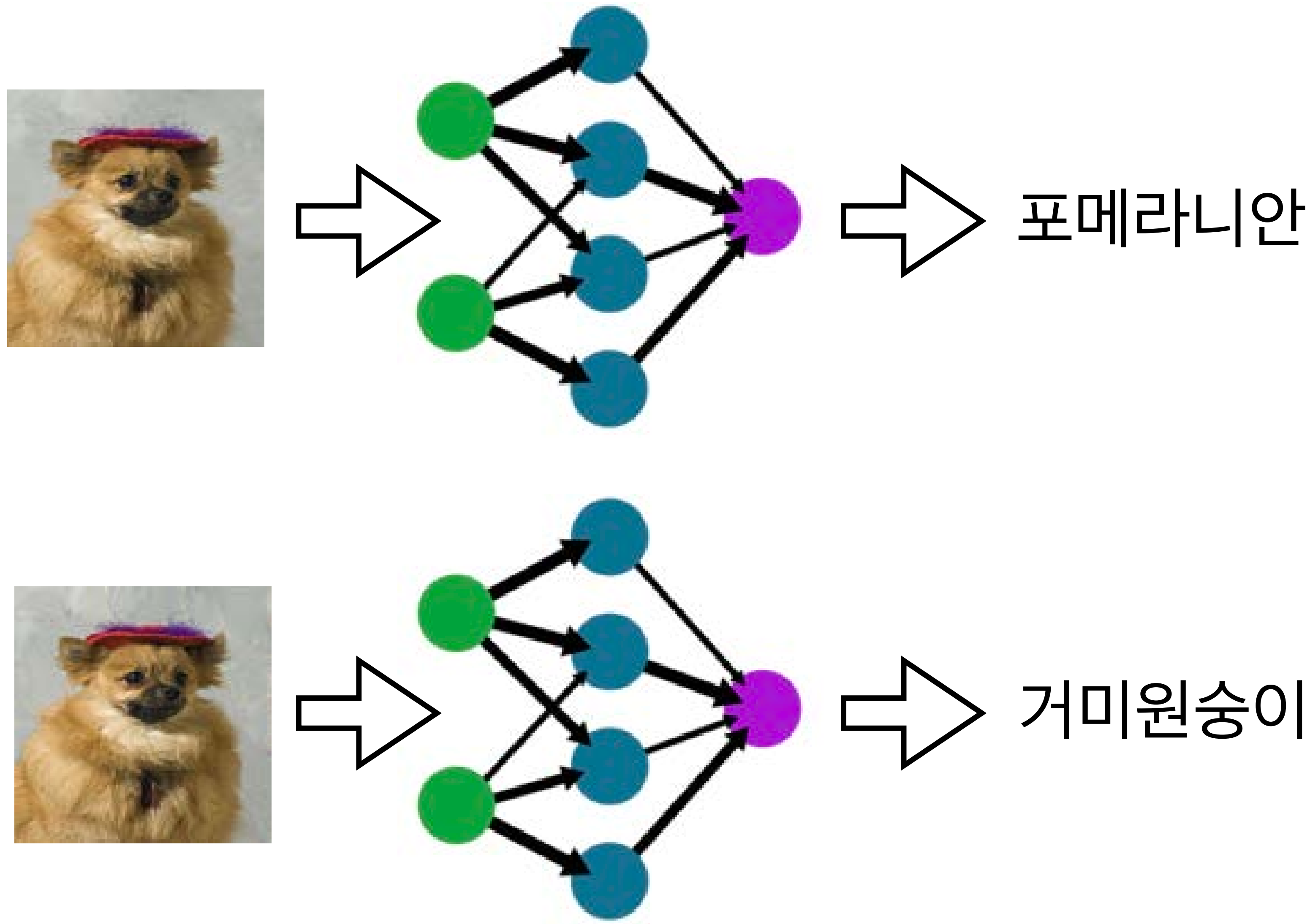
메모리 할당

메모리 해제

double-free

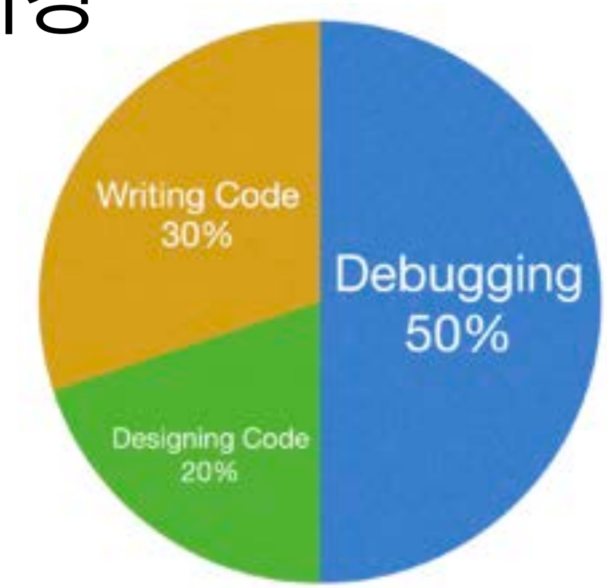
메모리 중복 해제 (double-free)

프로그램 분석 사례 (Neural Network)



SW 자동 수정 기술의 필요성

- 개발 생산성은 소프트웨어 산업 경쟁력을 결정하는 주 요인
- 소프트웨어개발에서 가장 생산성이 낮은 단계는 디버깅
  - 개발자들이 가장 부담스러워 하는 단계[1)]
  - 개발자들은 전체 시간의 절반을 디버깅에 사용[2)]
  - 상용 소프트웨어 오류를 수정하는데 평균 200일 소요[3)]



1) The art of software testing (3rd edition). 2012
2) Reversible debugging software. University of Cambridge. 2013
3) How long did it take to fix bugs? Mining Software Repositories. 2006

오류 자동 수정 기술 필요성 (Linux Kernel 사례)

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {

  goto err;
}

out = malloc(2);
if (out == NULL) {
  free(in);

  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

메모리 할당

메모리 해제

double-free

메모리 중복 해제 (double-free)

오류 자동 수정 기술 필요성 (Linux Kernel 사례)


USB: fix double frees in error code paths of ipaq driver

the error code paths can be enter with buffers to freed buffers.
Serial core would do a kfree() on memory already freed.

Signed-off-by: Oliver Neukum <oneukum@suse.de>
Signed-off-by: Greg Kroah-Hartman <gregkh@suse.de>

master v4.15-rc1 ... v2.6.24-rc1

Oliver Neukum committed with **gregkh** on 18 Sep 2007 1 par

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {
  out = NULL;
  goto err;
}

out = malloc(2);
if (out == NULL) {
  free(in);
  in = NULL;
  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

오류 자동 수정 기술 필요성 (Linux Kernel 사례)


USB: fix double frees in error code paths of ipaq driver

the error code paths can be enter with buffers to freed buffers.
Serial core would do a kfree() on memory already freed.

Signed-off-by: Oliver Neukum <oneukum@suse.de>
Signed-off-by: Greg Kroah-Hartman <gregkh@suse.de>

master v4.15-rc1 ... v2.6.24-rc1

Oliver Neukum committed with **gregkh** on 18 Sep 2007 1 par

수동 디버깅의 문제 1:
오류가 제거되었는지 확신하기 어려움

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {
  out = NULL;
  goto err;
}

out = malloc(2);
if (out == NULL) {
  free(in);
  in = NULL;
  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

오류 자동 수정 기술 필요성 (Linux Kernel 사례)

9개월 후에 다시 오류 수정을 시도

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {
  out = NULL;
  goto err;
}
free(out);
out = malloc(2);
if (out == NULL) {
  free(in);
  in = NULL;
  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

오류 자동 수정 기술 필요성 (Linux Kernel 사례)

memory leak

수동 디버깅의 문제 2:
오류 수정 과정에서 새로운 오류가 발생

9개월 후에 다시 오류 수정을 시도

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {
  out = NULL;
  goto err;
}
free(out);
out = malloc(2);
if (out == NULL) {
  free(in);
  in = NULL;
  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

오류 자동 수정 기술 필요성 (Linux Kernel 사례)

오류 발견에서 수정까지 총 10개월 소요

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);
out = NULL;
in = malloc(2);
if (in == NULL) {
  out = NULL;
  goto err;
}
free(out);
out = malloc(2);
if (out == NULL) {
  free(in);
  in = NULL;
  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

오류 자동 수정 기술 필요성 (Linux Kernel 사례)

수동 디버깅의 문제 3:
오류는 제거했지만 코드 품질이 떨어짐

오류 발견에서 수정까지 총 10개월 소요

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);
out = NULL;
in = malloc(2);
if (in == NULL) {
  out = NULL;
  goto err;
}
free(out);
out = malloc(2);
if (out == NULL) {
  free(in);
  in = NULL;
  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

SAVER: 메모리 오류 자동 수정기

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {

  goto err;
}

out = malloc(2);
if (out == NULL) {
  free(in);

  goto err;
}
... // use in, out
err:
  free(in); // double-free
  free(out);// double-free
  return;
```

SAVER

✓개발생산성↑
✓SW품질↑

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {

  goto err;
}
free(out);
out = malloc(2);
if (out == NULL) {
  free(in);

  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

프로그램 합성

- 사람은 프로그램이 만족해야 하는 성질(명세)을 기술

$$\text{reverse}(12) = 21,\ \text{reverse}(123) = 321$$

- 컴퓨터가 명세를 만족하는 코드를 합성

```
reverse(n) {
  r := 0;
  while (          ) {



  };
  return r;
}
```

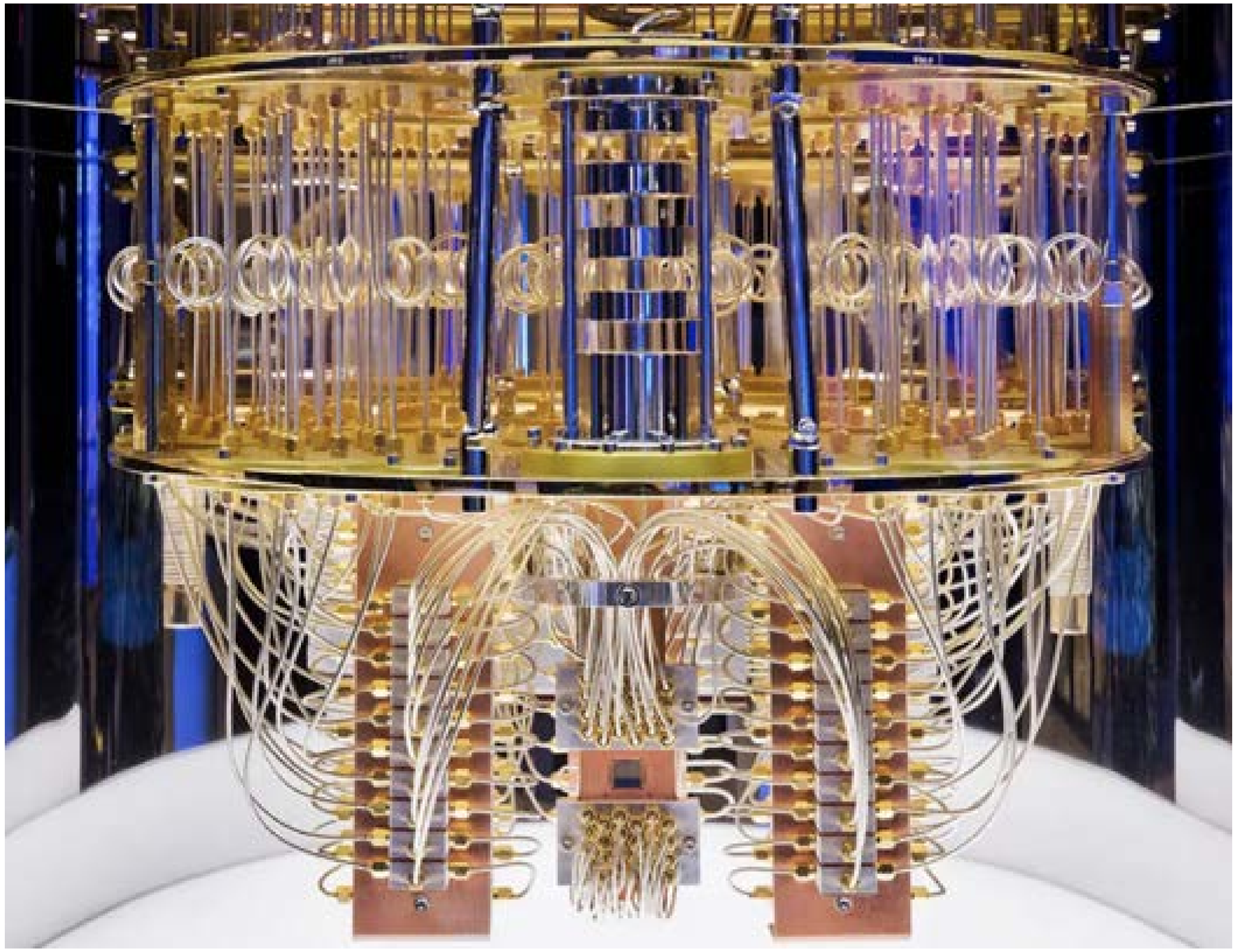
2.5s

```
reverse(n) {
  r := 0;
  while ( n > 0 ) {
    x := n % 10;
    r := r * 10;
    r := r + x;
    n := n / 10;
  };
  return r;
}
```

양자 컴퓨팅 소개

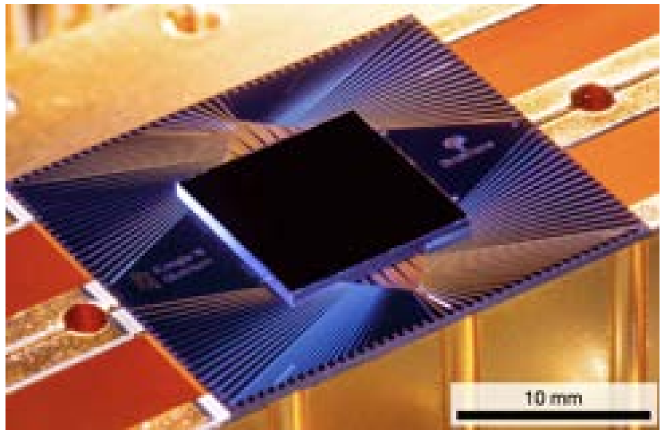
양자 컴퓨터?

- IBM에서 개발중인 양자 컴퓨터



The Promise of Quantum Computers

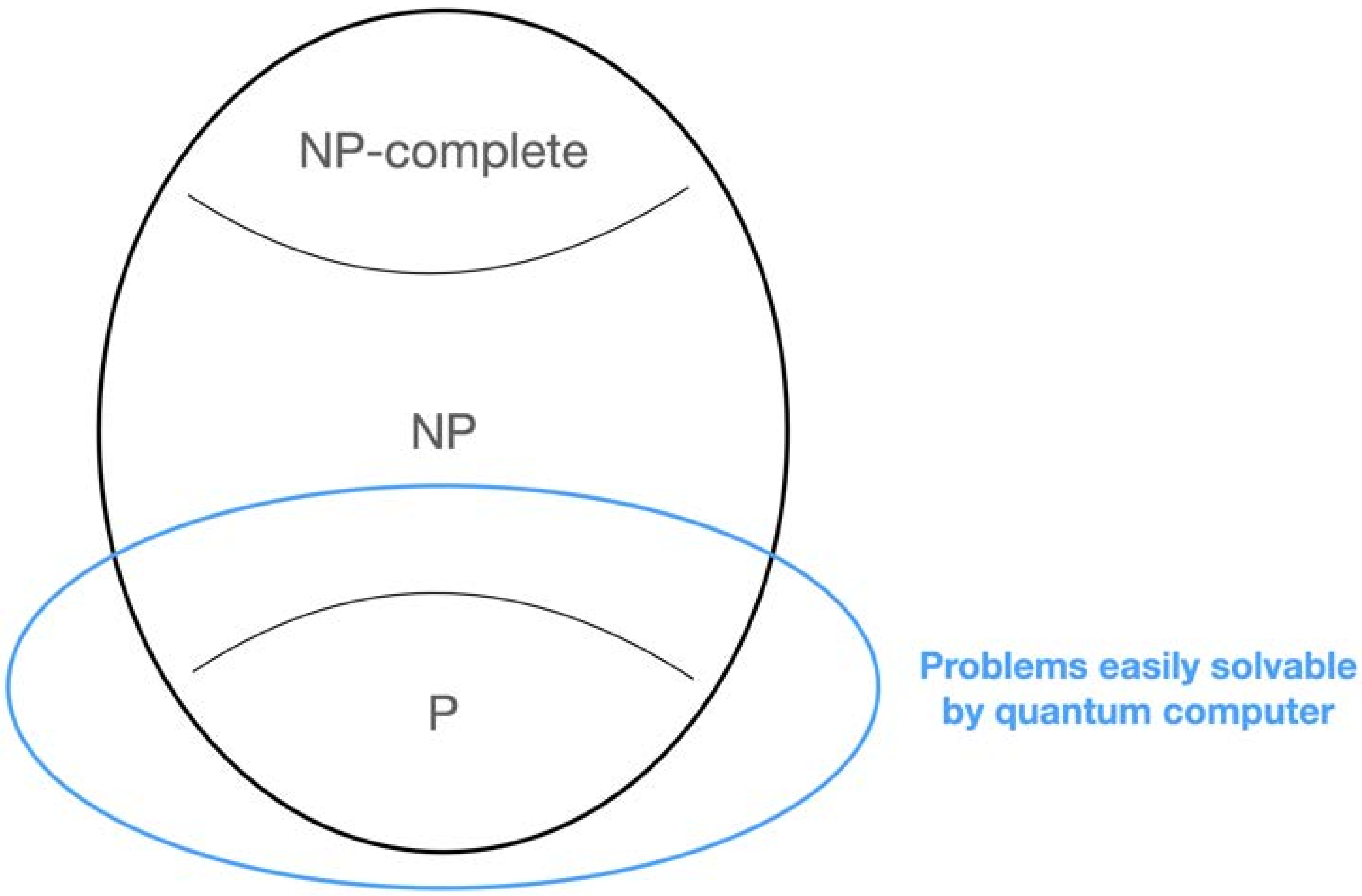
In 2019, Google demonstrated that a computationally hard problem can be easily solved on a real quantum computer.



- IBM Summit: 10,000 years
- Google Sycamore: 200 seconds

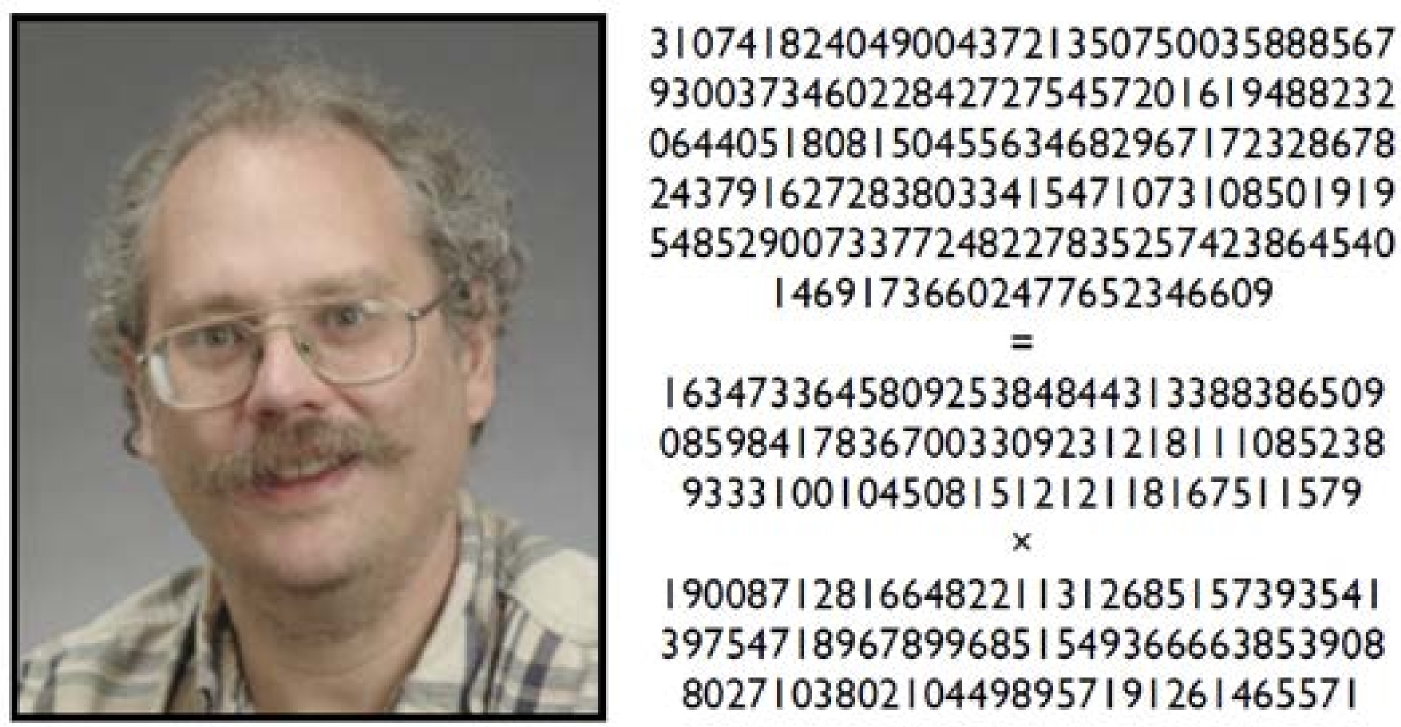
The Promise of Quantum Computers

Certain computational tasks can be much faster on a quantum computer than on a classical computer.



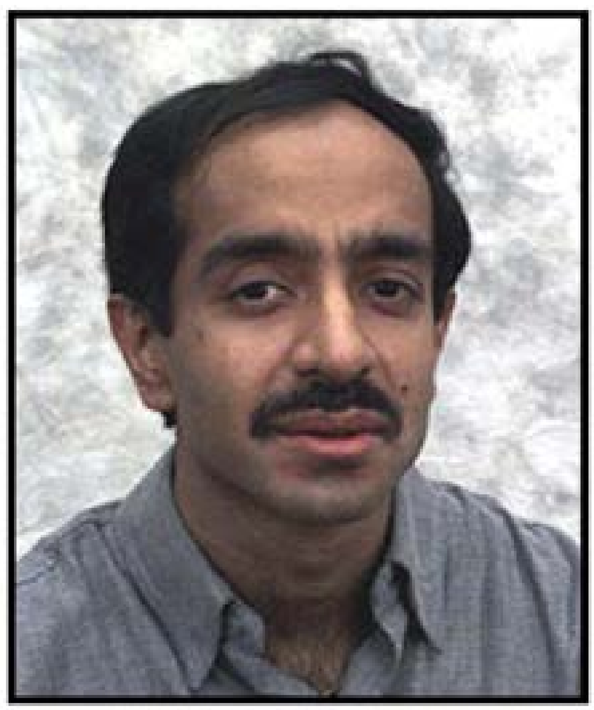
Killer Applications of Quantum Computer

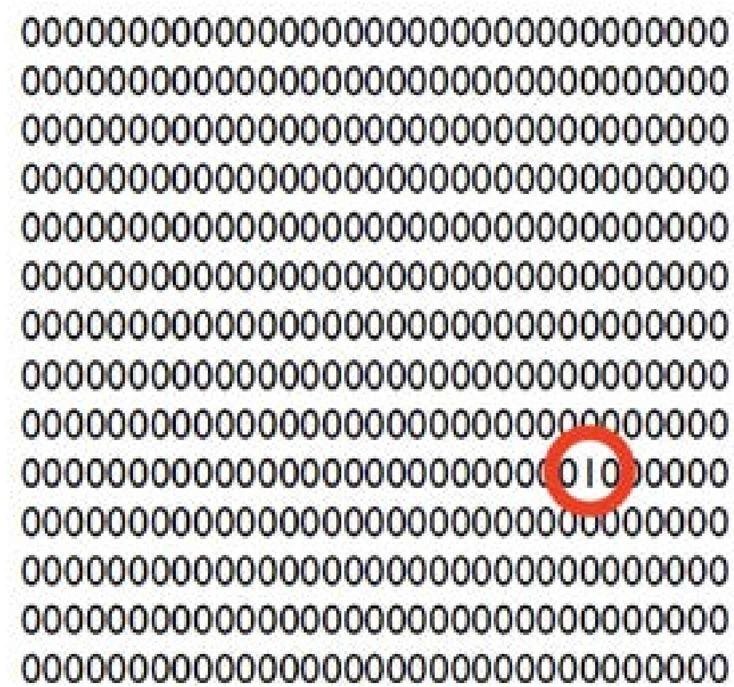
- Shor's algorithm
  - Polynomial-time algorithm for integer factorization
  - Exponentially faster than the most efficient classical factoring algorithm



Killer Applications of Quantum Computer

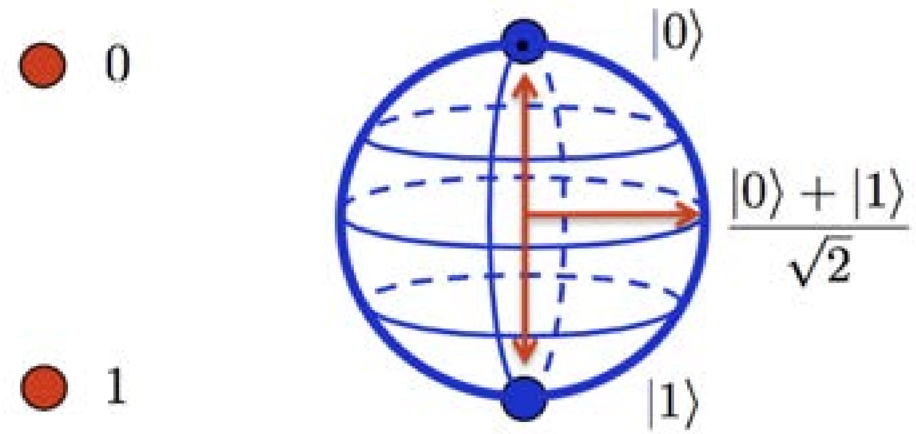
- Grover's algorithm for unstructured search
  - Given a function $f : \{1, 2, \ldots, N\} \rightarrow \{0, 1\}$, find x such that $f(x) = 1$.
  - Can be solved with $O(\sqrt{N})$ evaluations of f.
  - E.g., when $N = 1000$, only 25 evaluations are needed.





Classical vs. Quantum Computing

- A classical bit is either 0 or 1, but a quantum bits (qubit) is a superposition of 0 and 1:



- Quantum computers generalize classical computers, and we can exploit the generality to devise a new set of algorithms.

$$\boxed{\text{Computer Science} = \text{Quantum Computing}}$$

Qubits

- Classical bits are represented by $|0\rangle$ and $|1\rangle$ in quantum computing.
- The state of a qubit is a two-dimensional vector of the form

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

  - $|0\rangle$ and $|1\rangle$ are *computational basis states*:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

  - α and β are complex numbers ($\mathbb{C}$) such that

$$|\alpha|^2 + |\beta|^2 = 1.$$

- Examples:
  - $|0\rangle$, $|1\rangle$
  - $\frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle$, $\frac{1}{\sqrt{2}}\,|0\rangle - \frac{1}{\sqrt{2}}\,|1\rangle$
  - $\frac{1}{\sqrt{2}}\,|0\rangle + \frac{i}{\sqrt{2}}\,|1\rangle$, $\frac{1}{\sqrt{2}}\,|0\rangle - \frac{i}{\sqrt{2}}\,|1\rangle$

Multiple Qubits

- A 2-qubit system is a superposition of $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$.

$$|\psi\rangle = \alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle$$

- A 3-qubit system is a superposition of $|000\rangle, |001\rangle, \ldots, |111\rangle$.

$$|\psi\rangle = \alpha_{000}\,|000\rangle + \alpha_{001}\,|001\rangle + \cdots + \alpha_{110}\,|110\rangle + \alpha_{111}\,|111\rangle$$

- An N-qubit system is a superposition of classical N-bit states.

$$\sum_{x \in \{0,1\}^N} \alpha_x\,|x\rangle$$

$\boxed{\text{Exponential cost in classical computing!}}$

Quantum Parallelism

- Classical programs:

$$x \longrightarrow \boxed{f(x)} \longrightarrow y$$

- Quantum programs:

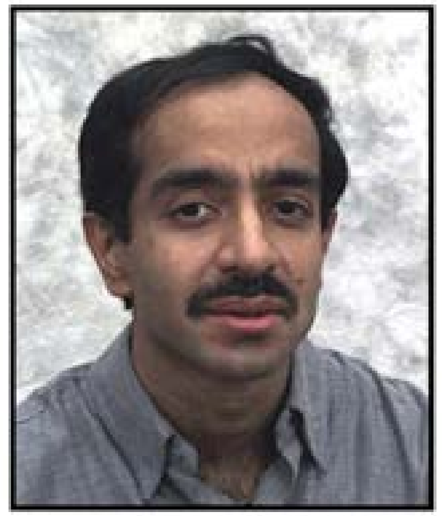
$$\sum_{x \in \{0,1\}^N} \alpha_x |x\rangle \longrightarrow \boxed{f(x)} \longrightarrow \text{모든 가능한 출력}$$

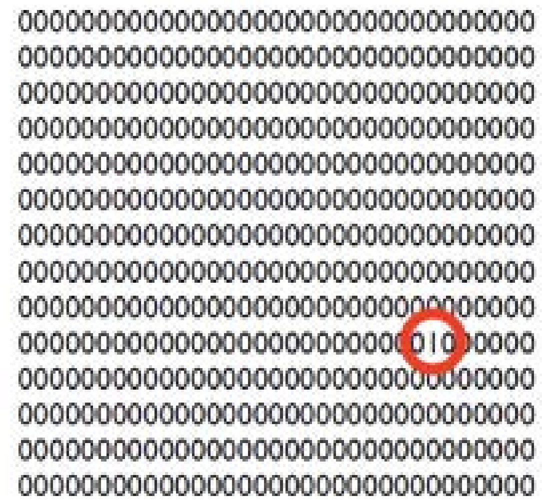
Measurement

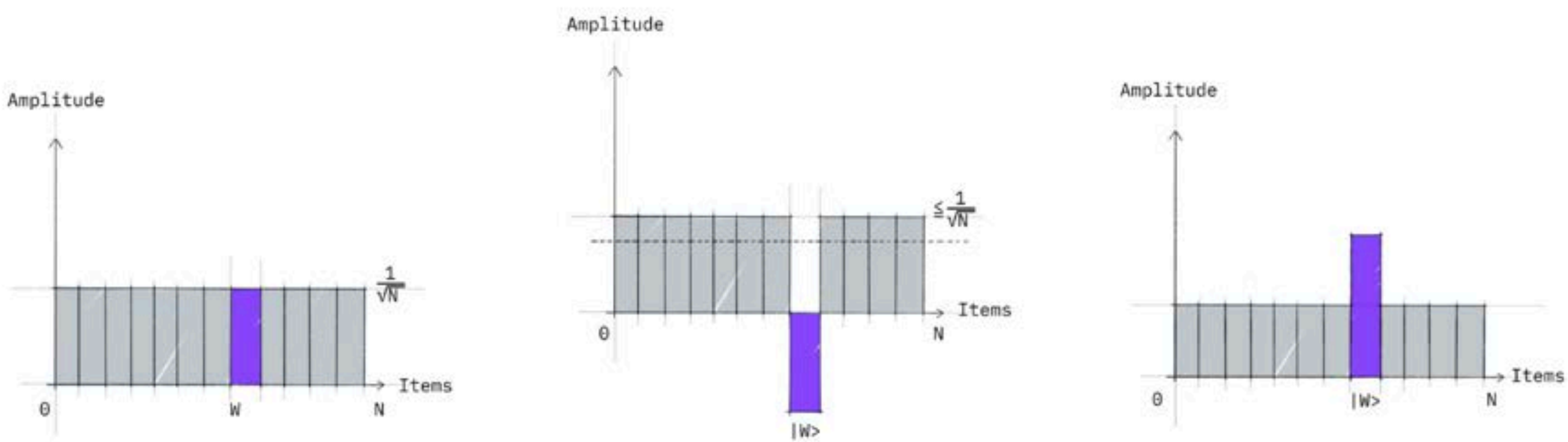
$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

- We cannot directly observe the state (α, β) of a qubit. We can only *measure* it.
- When we measure $|\psi\rangle$,
  - it jumps to $|0\rangle$ with probability $|\alpha|^2$ and we read 0, or
  - it jumps to $|1\rangle$ with probability $|\beta|^2$ and we read 1.
- Examples:
  - $|0\rangle$, $|1\rangle$
  - $\frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle$, $\frac{1}{\sqrt{2}}\,|0\rangle - \frac{1}{\sqrt{2}}\,|1\rangle$
  - $\frac{1}{\sqrt{2}}\,|0\rangle + \frac{i}{\sqrt{2}}\,|1\rangle$, $\frac{1}{\sqrt{2}}\,|0\rangle - \frac{i}{\sqrt{2}}\,|1\rangle$
  - $\frac{1}{2}\,|0\rangle + \frac{\sqrt{3}}{2}\,|1\rangle$

Grover's Algorithm

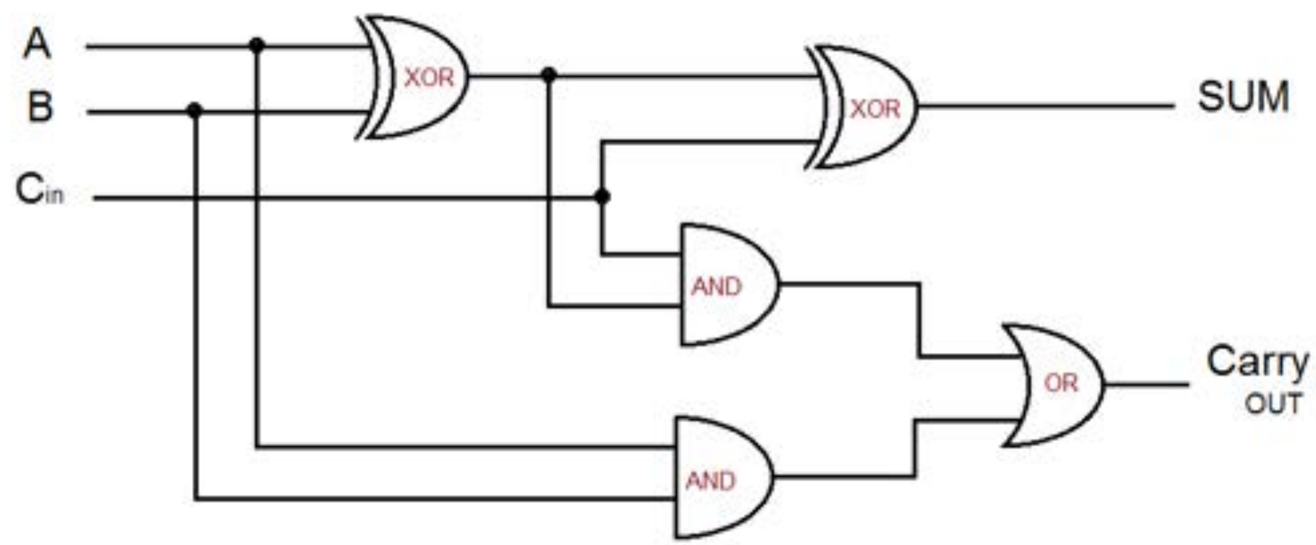




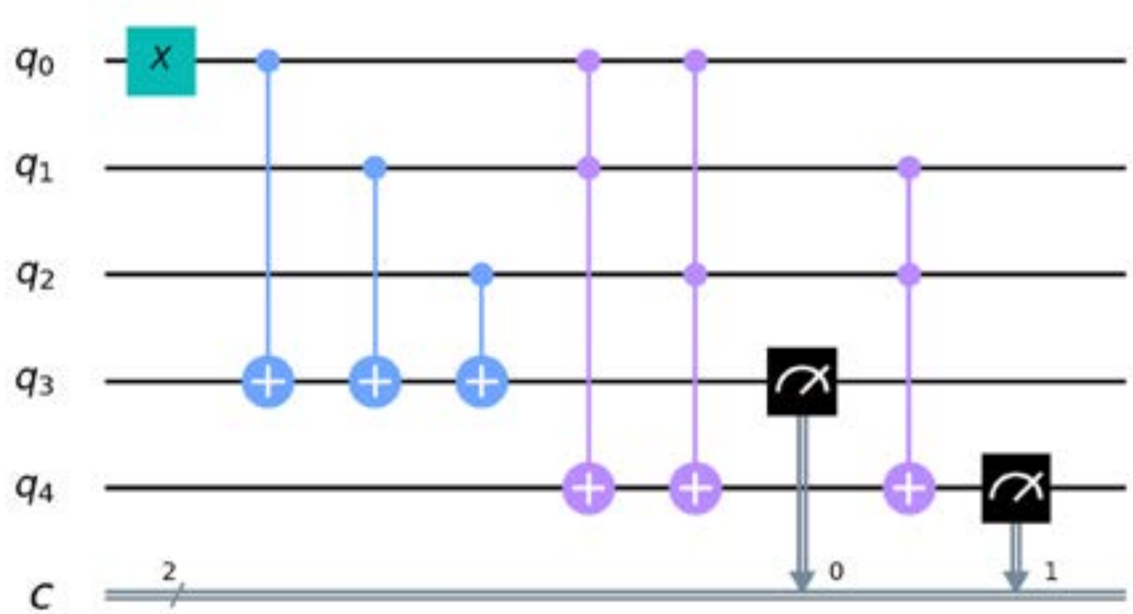


양자 회로 (양자 프로그램)

- Classical circuit:



- Quantum circuit:



양자 회로 (양자 프로그램)

- In quantum programming language:

```
print('\n Quantum Full Adder')
print('---------------------')

from qiskit import QuantumRegister
from qiskit import ClassicalRegister
from qiskit import QuantumCircuit, execute,IBMQ
from qiskit.tools.monitor import job_monitor

IBMQ.enable_account('INSERT API TOKEN HERE')
provider = IBMQ.get_provider(hub='ibm-q')

######## A ############################
q = QuantumRegister(5,'q')
c = ClassicalRegister(2,'c')

circuit = QuantumCircuit(q,c)
circuit.x(q[0])
circuit.cx(q[0],q[3])
circuit.cx(q[1],q[3])
circuit.cx(q[2],q[3])
circuit.ccx(q[0],q[1],q[4])
circuit.ccx(q[0],q[2],q[4])
circuit.ccx(q[1],q[2],q[4])

circuit.measure(q[3],c[0])
circuit.measure(q[4],c[1])
```

양자 게이트 예

- $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$$\alpha|0\rangle + \beta|1\rangle \;\text{——}\boxed{I}\text{——}\; \alpha|0\rangle + \beta|1\rangle$$

- $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

$$\alpha|0\rangle + \beta|1\rangle \;\text{——}\boxed{X}\text{——}\; \beta|0\rangle + \alpha|1\rangle$$

- $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

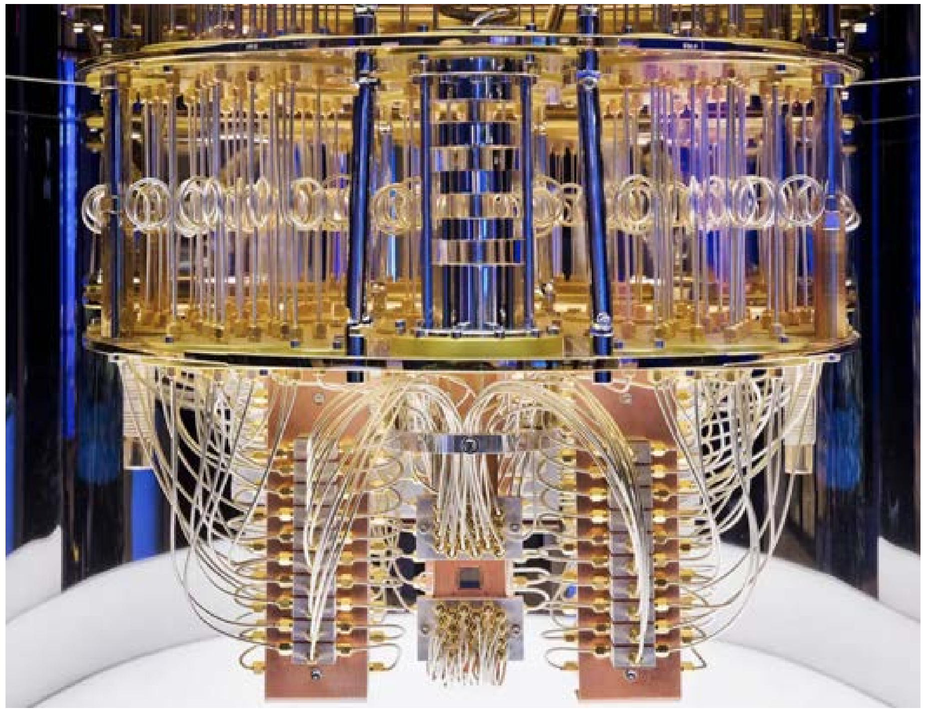
$$\alpha|0\rangle + \beta|1\rangle \;\text{——}\boxed{Z}\text{——}\; \alpha|0\rangle - \beta|1\rangle$$

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

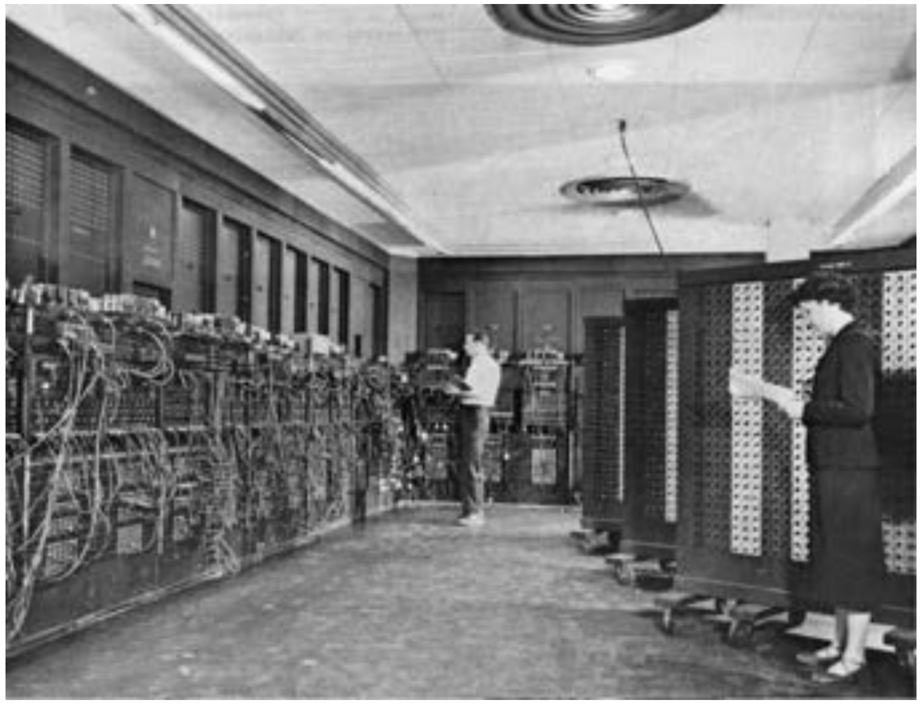
$$\alpha|0\rangle + \beta|1\rangle \;\text{——}\boxed{H}\text{——}\; \alpha\frac{|0\rangle+|1\rangle}{\sqrt{2}} + \beta\frac{|0\rangle-|1\rangle}{\sqrt{2}}$$

양자 컴퓨팅의 현재 수준

- 1950~60s of classical computers

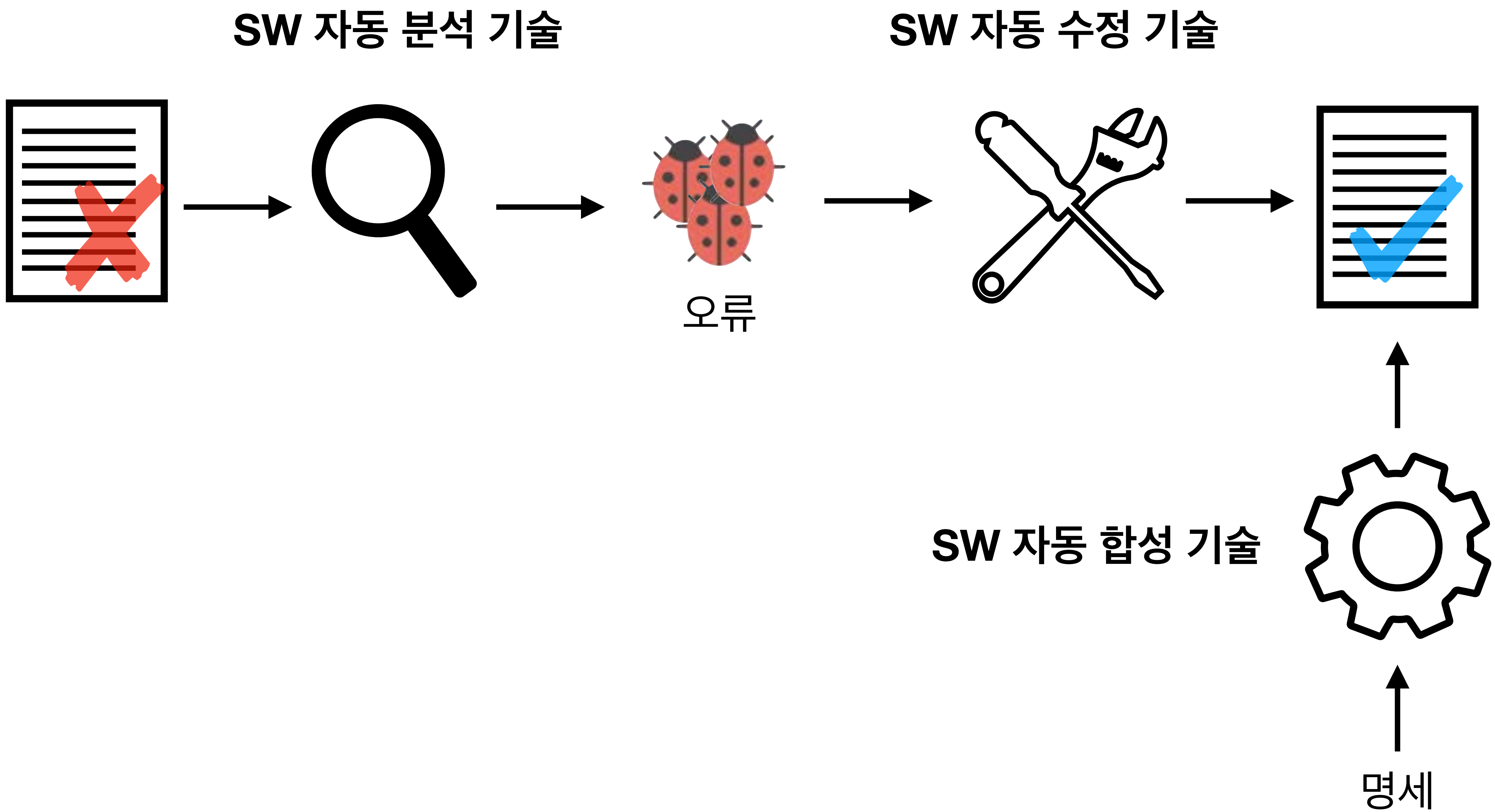


~



양자 프로그램으로의 확장

- 올바른 양자 프로그램 작성의 어려움
- 양자 프로그램을 위한 **자동 분석** + **자동 수정** + **자동 합성**



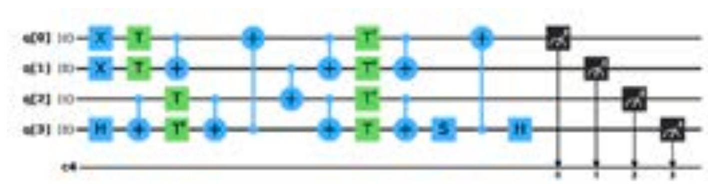
양자 회로 자동 합성

입출력 예제

```
0000 -> 0000
0100 -> 1111
```

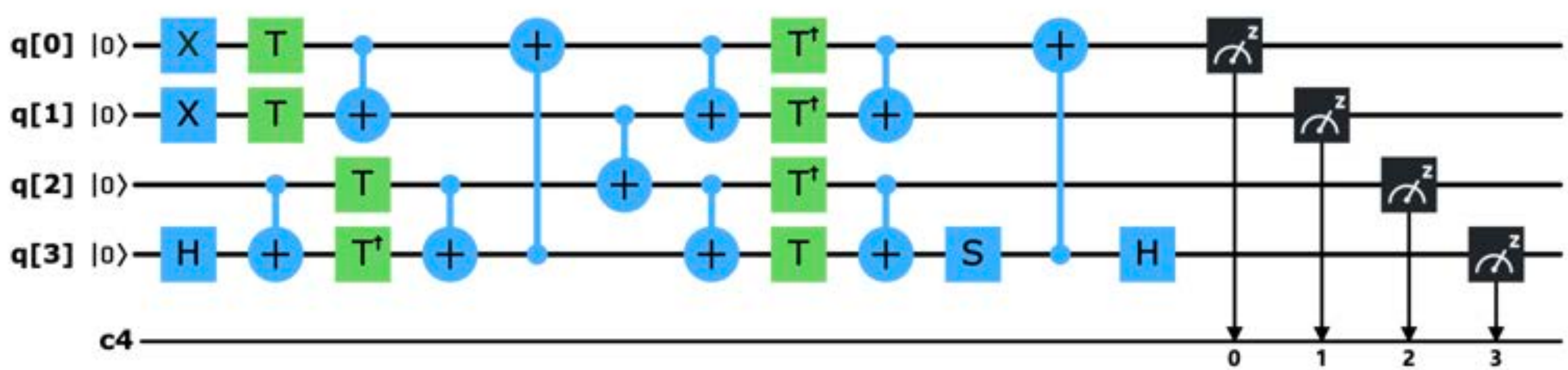
…

양자 회로 합성기



양자 회로 검증

- Writing correct quantum programs is challenging.
- How can we ensure that quantum programs are correct?



Summary

- Introduction to software analysis, repair, and synthesis
- Introduction to quantum computing and opportunities
- If you're intereseted in research in quantum program analysis, repair, and synthesis, send email to me (hakjoo_oh@korea.ac.kr)

소프트웨어 분석 연구실
Software Analysis Lab.

- Research areas: programming languages, software engineering, software security
    - program analysis and testing
    - program synthesis and repair
- Publication: top-venues in PL, SE, and Security:
    - PLDI('12,'14,'20),OOPSLA('15,'17a,'17b,'18a,'18b,'19,'20), TOPLAS('14,'16,'17,'18,'19), ICSE('17,'18,'19,'20,'21), FSE('18,'19,'20), ASE'18, ISSTA'20, IEEE S&P('17,'20), USENIX Security'21, etc

http://prl.korea.ac.kr