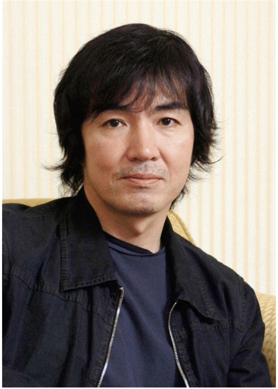
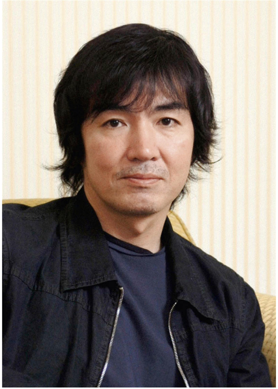
4-11 P and NP

Hengfeng Wei

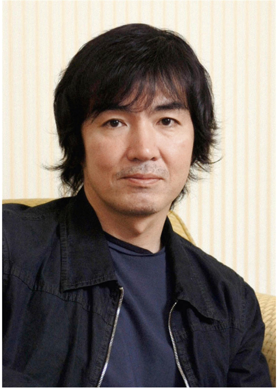
hfwei@nju.edu.cn

May 20, 2019






东野圭吾
嫌疑人X的献身



“对于数学问题，自己想出解答，
和判断别人说的解答是否正确，何者比较简单”


CONCEPT


decide.

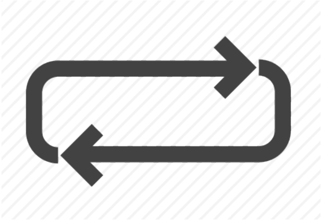

ACCEPT


decide.


ACCEPT


Yes
No
I·DECIDE

Always terminate.



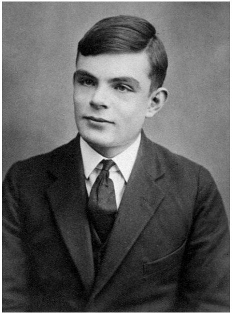
Always terminate.

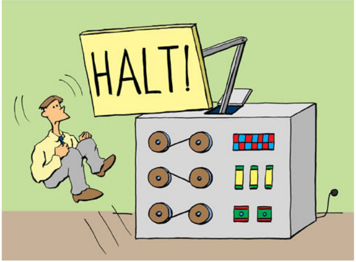
May loop forever for “NO” instance.

Definition (Halting Problem)

Input: An arbitrary program and input

Output: Will the program eventually halt?



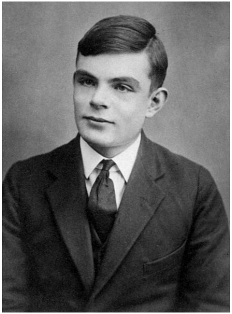


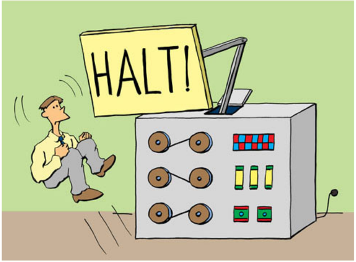
Alan designed the perfect computer

Definition (Halting Problem)

Input: An arbitrary program and input

Output: Will the program eventually halt?





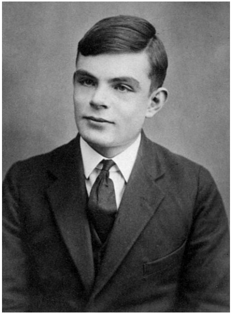
Alan designed the perfect computer

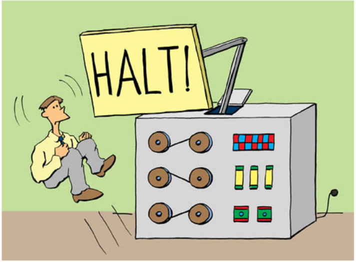
Undecidable

Definition (Halting Problem)

Input: An arbitrary program and input

Output: Will the program eventually halt?





Alan designed the perfect computer

Undecidable

But Acceptable (Semi-decidable)

$$\mathrm{P} = \Big\{L : L \text{ is decided by a poly. time algorithm}\Big\}$$

$$\mathrm{P} = \Big\{ L : L \text{ is } \textcolor{red}{\text{decided}} \text{ by a poly. time algorithm} \Big\}$$

Theorem (Theorem 34.2)

$$P = \Big\{ L : L \textit{ is } \textcolor{red}{\textit{accepted}} \textit{ by a poly. time algorithm} \Big\}$$

$$\mathrm{P} = \Big\{ L : L \text{ is } \textcolor{red}{\text{decided}} \text{ by a poly. time algorithm} \Big\}$$

Theorem (Theorem 34.2)

$$P = \Big\{ L : L \textit{ is } \textcolor{red}{\textit{accepted}} \textit{ by a poly. time algorithm} \Big\}$$

You can safely forget "semi-decidable" in computational complexity theory.

Definition (NP)

$$L \in \mathrm{NP}$$

$$\iff$$

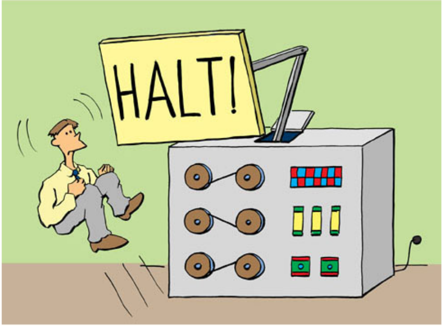
$\exists$ poly. time *verifier* $V(x, c)$ such that

$$\forall x \in \{0,1\}^* : x \in L \iff \exists c \text{ with } |c| = O(|x|^k), V(x,c) = 1.$$

NP-problems has short certificates that are easy to verify.

$$\exists L : L \notin \text{NP?}$$

$$\exists L : L \notin \mathrm{NP}?$$



Alan designed the perfect computer

$\exists L : L \notin \mathrm{NP} \wedge L$ is decidable?

$$\exists L : L \notin \text{NP} \land L \text{ is decidable?}$$

Theorem (Deterministic Time Hierarchy Theorem (1965))

$$f(n) \log f(n) = o\big(g(n)\big) \implies \mathit{DTIME}\big(f(n)\big) \subsetneq \mathit{DTIME}\big(g(n)\big)$$

$$\exists L : L \notin \text{NP} \wedge L \text{ is decidable?}$$

Theorem (Deterministic Time Hierarchy Theorem (1965))

$$f(n) \log f(n) = o(g(n)) \implies DTIME(f(n)) \subsetneq DTIME(g(n))$$

$$\text{P} \subsetneq \text{EXP}$$

$$\exists L : L \notin \text{NP} \land L \text{ is decidable?}$$

Theorem (Deterministic Time Hierarchy Theorem (1965))

$$f(n) \log f(n) = o(g(n)) \implies DTIME(f(n)) \subsetneq DTIME(g(n))$$

$$\text{P} \subsetneq \text{EXP}$$

Theorem (Non-deterministic Time Hierarchy Theorem (Cook, 1972))

$$f(n+1) = o(g(n)) \implies NTIME(f(n)) \subsetneq NTIME(g(n))$$

$$\exists L : L \notin \mathrm{NP} \land L \text{ is decidable?}$$

Theorem (Deterministic Time Hierarchy Theorem (1965))

$$f(n) \log f(n) = o(g(n)) \implies DTIME(f(n)) \subsetneq DTIME(g(n))$$

$$\mathrm{P} \subsetneq \mathrm{EXP}$$

Theorem (Non-deterministic Time Hierarchy Theorem (Cook, 1972))

$$f(n+1) = o(g(n)) \implies NTIME(f(n)) \subsetneq NTIME(g(n))$$

$$\mathrm{NP} \subsetneq \mathrm{NEXP}$$

$\exists L : L \notin \text{NP} \land L$ is decidable?

$$\exists L : L \notin \text{NP} \wedge L \text{ is decidable?}$$

"Equivalence of Regular Expressions with Squaring" is NEXP-complete:

$$e_1 \cup e_2, \quad e_1 \cdot e_2, \quad e^2$$

Closure of NP (CLRS 34.2-4)

NP is closed under $\cup, \cap, \cdot, \star$.

$$L_1 \in \text{NP}, L_2 \in \text{NP} \implies L = L_1 \circ L_2 \in \text{NP}$$

$$L_1 \in \mathrm{NP}, L_2 \in \mathrm{NP} \implies L = L_1 \cup L_2 \in \mathrm{NP}$$

$$L_1 \in \mathrm{NP}, L_2 \in \mathrm{NP} \implies L = L_1 \cup L_2 \in \mathrm{NP}$$

procedure V(x, c)

  if $c \neq c_1 \# c_2$ **then**

    return 0

  return $V(x, c_1) \lor V(x, c_2)$

$$L_1 \in \mathrm{NP}, L_2 \in \mathrm{NP} \implies L = L_1 \cup L_2 \in \mathrm{NP}$$

```
1: procedure V(x, c)
2:     if c ≠ c1#c2 then
3:         return 0

4:     return V(x, c1) ∨ V(x, c2)
```

$$x \in L_1 \cup L_2 \iff \exists c, V(x, c) = 1$$

$$L_1 \in \mathrm{NP}, L_2 \in \mathrm{NP} \implies L = L_1 \cap L_2 \in \mathrm{NP}$$

$$L_1 \in \mathrm{NP}, L_2 \in \mathrm{NP} \implies L = L_1 \cap L_2 \in \mathrm{NP}$$

```
procedure V(x, c)
    if c ≠ c1#c2 then
        return 0

    return V(x, c1) ∧ V(x, c2)
```

$$L_1 \in \text{NP}, L_2 \in \text{NP} \implies L = L_1 \cap L_2 \in \text{NP}$$

```
procedure V(x, c)
    if c ≠ c1#c2 then
        return 0

    return V(x, c1) ∧ V(x, c2)
```

$$x \in L_1 \cap L_2 \iff \exists c, V(x, c) = 1$$

$$L_1 \in \mathrm{NP}, L_2 \in \mathrm{NP} \implies L = L_1 \cdot L_2 \in \mathrm{NP}$$

$$L_1 \in \mathrm{NP}, L_2 \in \mathrm{NP} \implies L = L_1 \cdot L_2 \in \mathrm{NP}$$

```
procedure V(x, c)
    if c ≠ c1#c2&m then
        return 0

    return V(x_{1...m}, c1) ∧ V(x_{m+1...|x|}, c2)
```

$$\textbf{procedure } \mathrm{V}(x, c)$$
$$\quad \textbf{if } c \neq c_1 \# c_2 \& m \textbf{ then}$$
$$\qquad \textbf{return } 0$$
$$\quad \textbf{return } V(x_{1\dots m}, c_1) \land V(x_{m+1\dots|x|}, c_2)$$

$$L_1 \in \mathrm{NP}, L_2 \in \mathrm{NP} \implies L = L_1 \cdot L_2 \in \mathrm{NP}$$

```
procedure V(x, c)
    if c ≠ c_1#c_2&m then
        return 0

    return V(x_{1...m}, c_1) ∧ V(x_{m+1...|x|}, c_2)
```

$$x \in L_1 \cdot L_2 \iff \exists c, V(x, c) = 1$$

$$L \in \mathrm{NP} \implies L^{\star} \in \mathrm{NP}$$

$$L \in \text{NP} \implies L^\star \in \text{NP}$$

```
1: procedure V(x, c)
2:     for k ← 1 to |x| do
3:         m_0 ← 0, m_k ← |x|
4:         if c = c_1#c_2#···#c_k&m_1&m_2&···&m_{k-1} then
5:             return ⋀_{i=1}^{i=k} V(x_{m_{i-1}+1...m_i}, c_i)
```

procedure $\mathrm{V}(x, c)$

for $k \leftarrow 1$ **to** $|x|$ **do**

$m_0 \leftarrow 0, m_k \leftarrow |x|$

if $c = c_1 \# c_2 \# \cdots \# c_k \& m_1 \& m_2 \& \cdots \& m_{k-1}$ **then**

return $\bigwedge_{i=1}^{i=k} V(x_{m_{i-1}+1 \ldots m_i}, c_i)$

$$L \in \mathrm{NP} \implies L^{\star} \in \mathrm{NP}$$

```
procedure V(x, c)
    for k ← 1 to |x| do
        m_0 ← 0, m_k ← |x|
        if c = c_1#c_2# ··· #c_k&m_1&m_2& ··· &m_{k-1} then
            return ⋀_{i=1}^{i=k} V(x_{m_{i-1}+1...m_i}, c_i)
```

$$x \in L^{\star} \iff \exists c, A(x, c) = 1$$

Definition (Polynomial-time Reduction)

$L_1 \leq_p L_2$ if $\exists$ poly. time function f such that

$$\forall x : x \in L_1 \iff f(x) \in L_2.$$

Definition (Polynomial-time Reduction)

$$L_1 \leq_p L_2 \text{ if } \exists \text{ poly. time function } f \text{ such that}$$

$$\forall x : x \in L_1 \iff f(x) \in L_2.$$

$$\forall L \in \text{NP}, L \leq_p L' \implies L' \text{ is NP-hard}$$

$$\text{NP-complete} = \text{NP} \cap \text{NP-hard}$$

CLRS 34.5-6

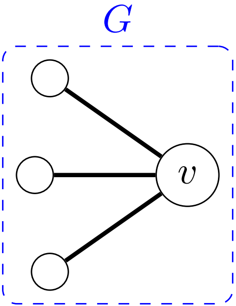
HAM-PATH is NP-complete

HAM-PATH is NP-complete

HAM-CYCLE $\leq_p$ HAM-PATH

HAM-PATH is NP-complete

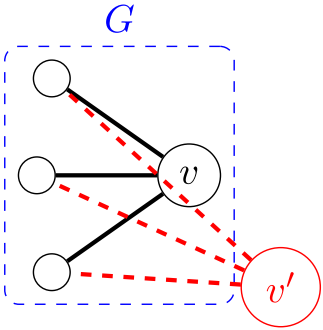
HAM-CYCLE $\leq_p$ HAM-PATH



$$G \in \text{HAM-CYCLE} \iff G' \in \text{HAM-PATH}$$

HAM-PATH is NP-complete

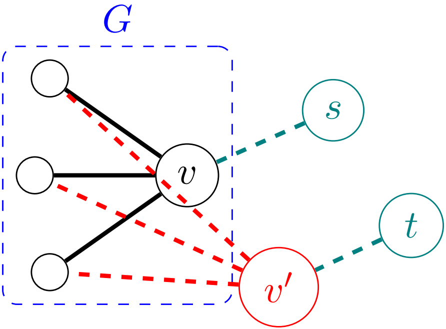
HAM-CYCLE $\leq_p$ HAM-PATH



$$G \in \text{HAM-CYCLE} \iff G' \in \text{HAM-PATH}$$

HAM-PATH is NP-complete

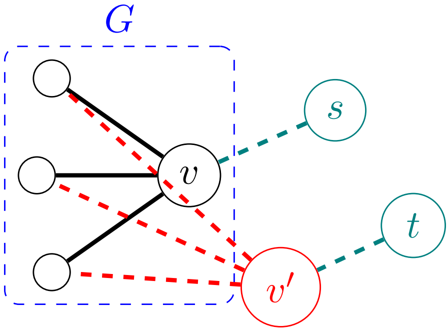
HAM-CYCLE $\leq_p$ HAM-PATH



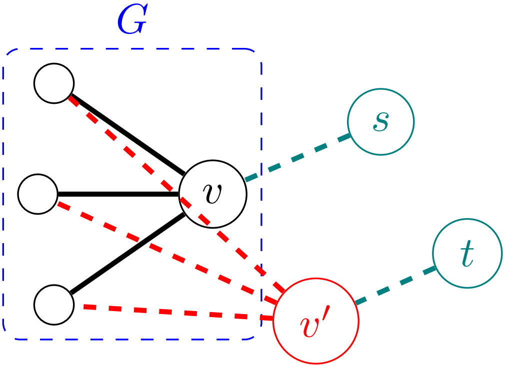
CLRS 34.5-6

HAM-PATH is NP-complete

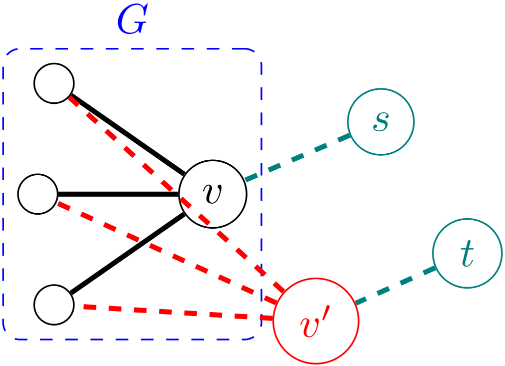
HAM-CYCLE $\leq_p$ HAM-PATH



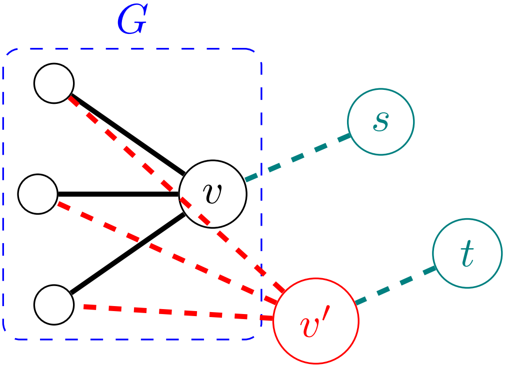
$$G \in \text{HAM-CYCLE} \iff G' \in \text{HAM-PATH}$$



G
v
s
t
v'



$\deg(v) \geq 2$



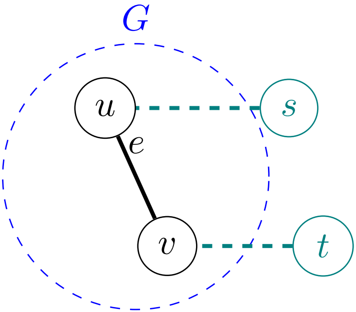
$\deg(v) \geq 2$

$\forall u \in V(G) : \deg(u) \geq 2$

HAM-CYCLE $\leq_p$ HAM-PATH

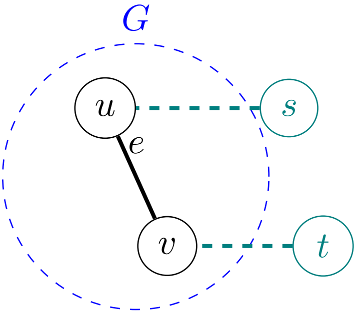
HAM-CYCLE $\leq_p$ HAM-PATH

$\forall e \in G$: Construct G_e



HAM-CYCLE $\leq_p$ HAM-PATH

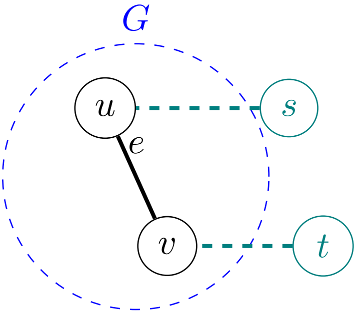
$\forall e \in G$: Construct G_e



G has a HC containing $e \iff G_e$ has a HP

HAM-CYCLE $\leq_p$ HAM-PATH

$\forall e \in G$: Construct G_e



G has a HC containing $e \iff G_e$ has a HP

$G \in$ HAM-CYCLE $\iff \exists G_e : G_e$ has a HP

HAM-CYCLE $\leq_p$ HAM-PATH

$$G \text{ has a HC containing } e \iff G_e \text{ has a HP}$$

$$G \in \text{HAM-CYCLE} \iff \exists G_e : G_e \text{ has a HP}$$

Definition (Polynomial-time Reduction)

$L_1 \leq_p L_2$ if $\exists$ poly. time function f such that

$$\forall x : x \in L_1 \iff f(x) \in L_2.$$

Definition (Polynomial-time Reduction)

$L_1 \leq_p L_2$ if $\exists$ poly. time function f such that

$$\forall x : x \in L_1 \iff f(x) \in L_2.$$

x for $L_1 \mapsto x' = f(x)$ for L_2

Definition (Polynomial-time Reduction)

$L_1 \leq_p L_2$ if $\exists$ poly. time function f such that

$$\forall x : x \in L_1 \iff f(x) \in L_2.$$

x for $L_1 \mapsto x' = f(x)$ for L_2

Call the oracle O_2 for L_2 once

Definition (Polynomial-time Reduction)

$L_1 \leq_p L_2$ if $\exists$ poly. time function f such that

$$\forall x : x \in L_1 \iff f(x) \in L_2.$$

x for $L_1 \mapsto x' = f(x)$ for L_2

Call the oracle O_2 for L_2 once

Answer whatever O_2 returns

Definition (Polynomial-time Reduction)

$L_1 \leq_p L_2$ if $\exists$ poly. time function f such that

$$\forall x : x \in L_1 \iff f(x) \in L_2.$$

x for $L_1 \mapsto x' = f(x)$ for L_2

Call the oracle O_2 for L_2 once

Answer whatever O_2 returns

Definition (Polynomial-time Reduction)

$L_1 \leq_p L_2$ if $\exists$ poly. time function f such that

$$\forall x : x \in L_1 \iff f(x) \in L_2.$$

Definition (Polynomial-time Reduction)

$L_1 \leq_p L_2$ if $\exists$ poly. time function f such that

$$\forall x : x \in L_1 \iff f(x) \in L_2.$$

Karp Reduction

Definition (Polynomial-time Reduction)

$L_1 \leq_p L_2$ if $\exists$ poly. time function f such that

$$\forall x : x \in L_1 \iff f(x) \in L_2.$$

Karp Reduction

Richard M. Karp (1935 ~)

REDUCIBILITY AMONG COMBINATORIAL PROBLEMS

Richard M. Karp

University of California at Berkeley

(1972)

Cook Reduction

Stephen Cook (1939 $\sim$)

The Complexity of Theorem-Proving Procedures

Stephen A. Cook

University of Toronto

(1971)

$$\text{UNSAT} = \Big\{ \varphi : \varphi \text{ is unsatisfiable.} \Big\}$$

Q : Is UNSAT NP-hard?

$$\mathrm{UNSAT} = \Big\{ \varphi : \varphi \text{ is unsatisfiable.} \Big\}$$

Q : Is UNSAT NP-hard?

Proof.

$$\text{UNSAT} = \Big\{ \varphi : \varphi \text{ is unsatisfiable.} \Big\}$$

Q : Is UNSAT NP-hard?

Proof.

$$\text{SAT} \le_p \text{UNSAT}$$

$$\text{UNSAT} = \Big\{\varphi : \varphi \text{ is unsatisfiable.}\Big\}$$

Q : Is UNSAT NP-hard?

Proof.

$$\text{SAT} \le_p \text{UNSAT}$$

$$x \in \text{SAT} \iff x \notin \text{UNSAT}$$

$$\text{UNSAT} = \left\{ \varphi : \varphi \text{ is unsatisfiable.} \right\}$$

Q : Is UNSAT NP-hard?

Proof.

$$\text{SAT} \le_p \text{UNSAT}$$

$$x \in \text{SAT} \iff x \notin \text{UNSAT}$$

□

$$\text{UNSAT} = \left\{ \varphi : \varphi \text{ is unsatisfiable.} \right\}$$

Q : Is UNSAT NP-hard?

Proof.

$$\text{SAT} \le_p \text{UNSAT}$$

$$x \in \text{SAT} \iff x \notin \text{UNSAT}$$

□

$$\boxed{\forall x : x \in L_1 \iff f(x) \in L_2}$$

Theorem (CLRS 34.5-2)

0/1 *ILP is NP-complete.*

Theorem (CLRS 34.5-2)

0/1 ILP is NP-complete.

$$0/1 \text{ ILP} \in \text{NP}$$

Theorem (CLRS 34.5-2)

0/1 ILP is NP-complete.

$$0/1 \text{ ILP} \in \text{NP}$$

$$\text{3-CNF-SAT} \leq_p 0/1 \text{ ILP}$$

Theorem (CLRS 34.5-2)

0/1 ILP is NP-complete.

$$0/1 \text{ ILP} \in \text{NP}$$

$$\text{3-CNF-SAT} \leq_p 0/1 \text{ ILP}$$

$x_1 \lor \overline{x_2} \lor \overline{x_3} \iff$

Theorem (CLRS 34.5-2)

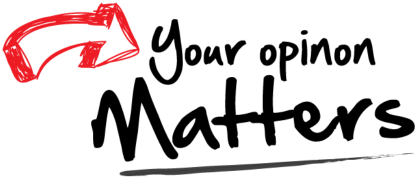
0/1 ILP is NP-complete.

$$0/1 \text{ ILP} \in \text{NP}$$

$$\text{3-CNF-SAT} \le_p 0/1 \text{ ILP}$$

$$x_1 \lor \overline{x_2} \lor \overline{x_3} \iff x_1 + (1 - x_2) + (1 - x_3) \ge 1, \quad x_i \in \{0, 1\}$$


THANK YOU



Office 302

Mailbox: H016

hfwei@nju.edu.cn