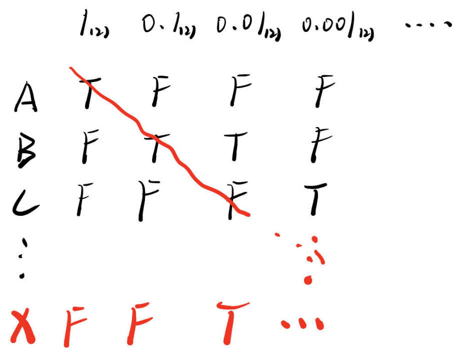
Math Logic Assignment 2

— 饶翔云 520030910366

— 2022 9 28

Q1

Proof: Assume $\mathcal{R}$ is countable $\Rightarrow \mathcal{R}$ is enumerble $\Rightarrow \exists$ a non-repetitive listing of $\mathcal{R}$.
Suppose A,B,C... are items of the listing and we can find write it down in the binary form:



T or F convey whether the corresponding bit has value. And we can find from the above picture that we can always construct a X$\in \mathcal{R}$ but $\notin$ the listing of $\mathcal{R}$ in the way of diagonal argument.
Therefore, $\mathcal{R}$ is uncountable.

Q2

```
input i:
if i == 1
    print yes.
    halt
for j in 2,3……,i-1:
    if i % j ==0:
        print no.
        halt
print yes
halt
```

Q3

```
initial P as empty sets.
for i in 1,2,3……:
    if i % j != 0 for j in P and j < i:
        print i
        add i to P as a new member.
    continue
```

Q4

Proof: Suppose $\mathcal{A}$ is a algorithm for computing f. From ppt p63 we know that $rng(f)$ is an effectively enumerable set. Obviously $rng(f)$ is a subset of $\mathbb{N}$.
We can construct a algorithm $\mathcal{B}$:

```
input a
for i in 1,2,3……:
    if A(i) has no output:
        continue
    if A(i) < a:
        continue
    if A(i) = a:
        print yes
    if A(i) > a:
        print no
```

For total function:
Because of the strict increasing property of f, we can always find A(i) $\geq$ a in finite steps(less than a).
~~**For partial function:**~~
~~Suppose the domain of f is {a_1,a_2,a_3,a_4,……}, and the biggest distance between neighbers is d. Then we can we can always find A(i) $\geq$ a in finite steps(less than a*d)~~
~~Then we can decide whether a $\in rng(f)$ in finite steps.~~
Therefore, $rng(f)$ is effectively decidable.

Q5

Proof: We can adjust the algorithm $\mathcal{B}$ for listing $\mathcal{A}$ in this way($\mathcal{C}$ for listing $\mathbb{N}\setminus \mathcal{A}$)

```
for input n
for i in 1,2,3……,n:
    conduct B and C:
    check which list n is in:
    if n is in listing of B:
        print yes
    else print no
```

Since $\mathcal{B}$ and $\mathcal{C}$ can list every member in finite step, we first run $\mathcal{B}$ and $\mathcal{C}$ then we just judge which list n is in. This process costs finite steps. **Therefore, $\mathcal{A}$ is effectively enumerble.**

Q6

Proof: Suppose the algorithm $\mathcal{A}$ is the listing algorithm of $\mathcal{R}$, then we can construct th e following algorithm:

```
initial R as empty set
print 0
for i in 1,2,3……:
    continue running A until it prints i-th member n.
    add it to R
    for j in 1,2,3……:
        if j not in R:
            break
        if j+1 has been printed:
            return
        print j+1
```

Let the listing of A' be $p_0, p_1, p_2, \ldots\ldots$, and $p_0 = 0$. If exist $p_{i+1} - p_i \geq 1$, then we will have no element $> p_i + 1$, because we can always say $\forall n > p_i + 1$, $\exists p_i \notin$ R. And we can output members of R in finite step. So if we want to output the i_{th} member of P, we will first list all members in R and < n. Since this process is finite, we can easily know whether i_{th} member exists by judging the forward elements whether they are ccontinuous. If it exists we can just output it in finite steps as above.
Therefore, P is effectively enumerble.