

22

20

16

12

11

10

7

16

11

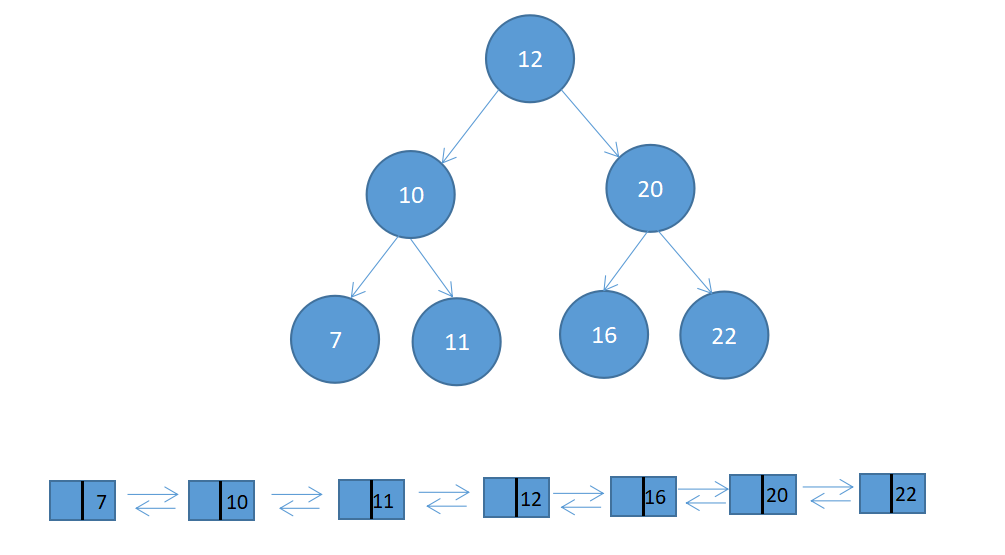
20

10

22

7

12



f(7)=5

f(6)=3

f(5)=1

f(4)=-7

f(3)=8

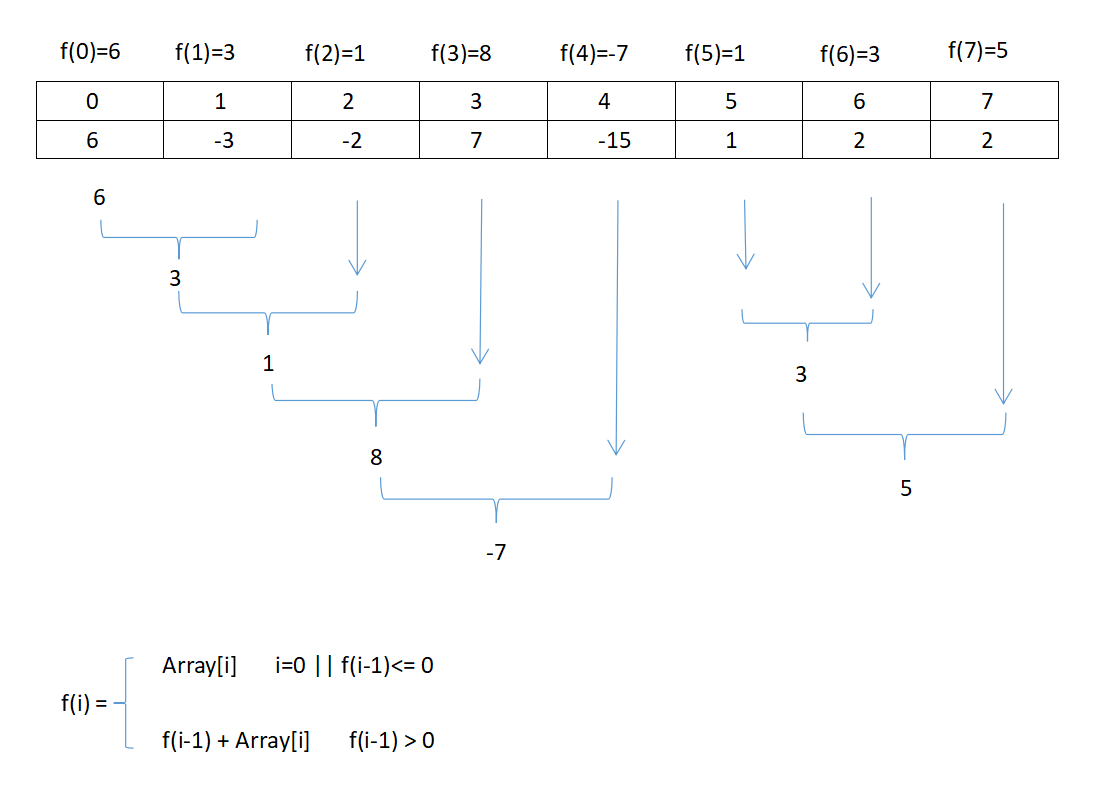
f(2)=1

f(1)=3

f(0)=6

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 6 | -3 | -2 | 7 | -15 | 1 | 2 | 2 |

6

Array[i] i=0 || f(i-1)<= 0  
f(i) =   
 f(i-1) + Array[i] f(i-1) > 0  
  
  
 

5

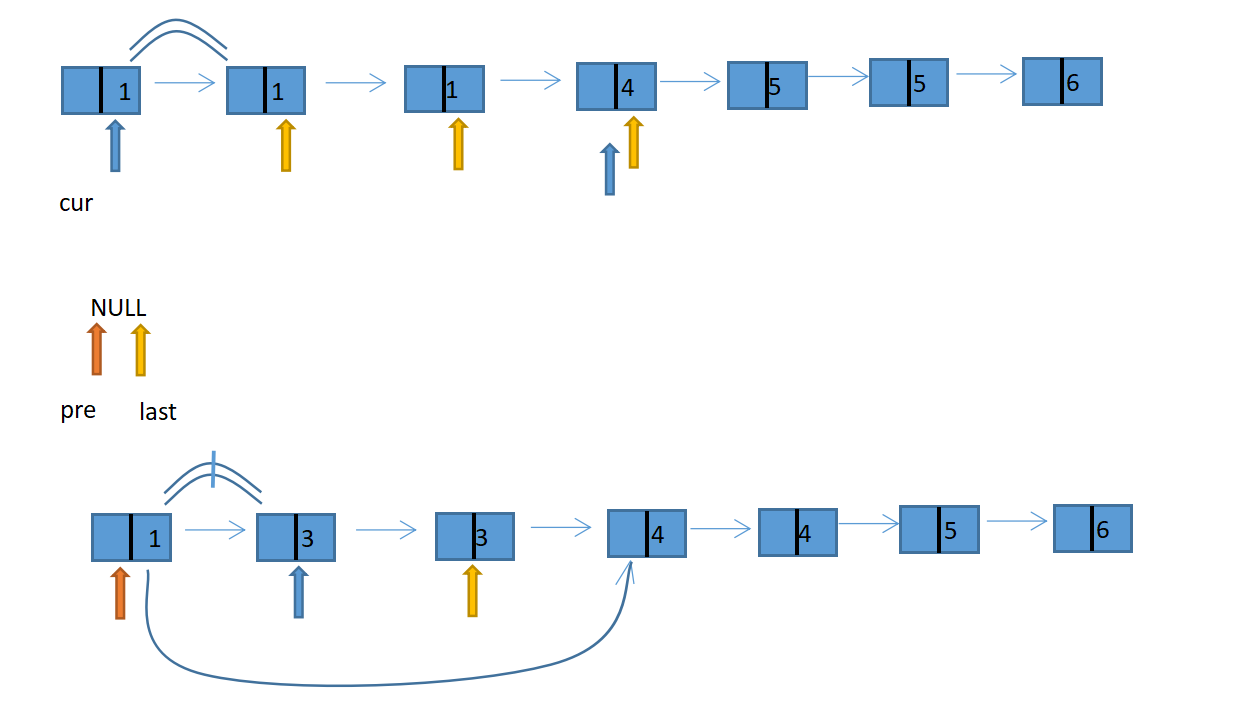
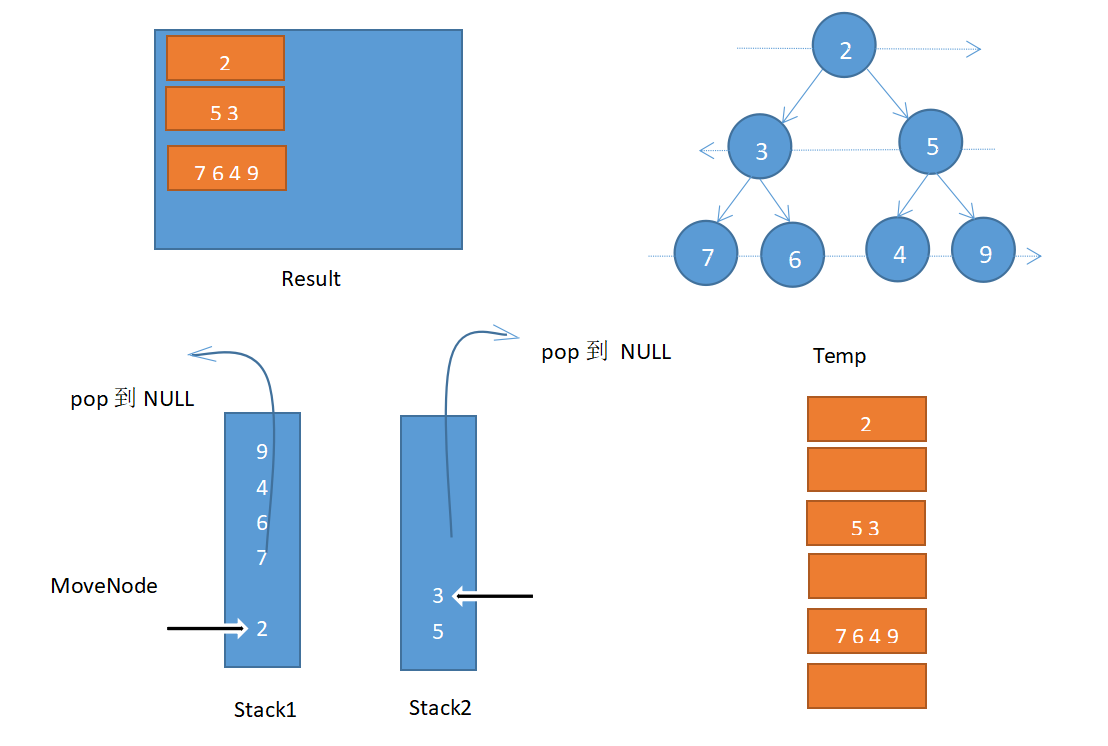
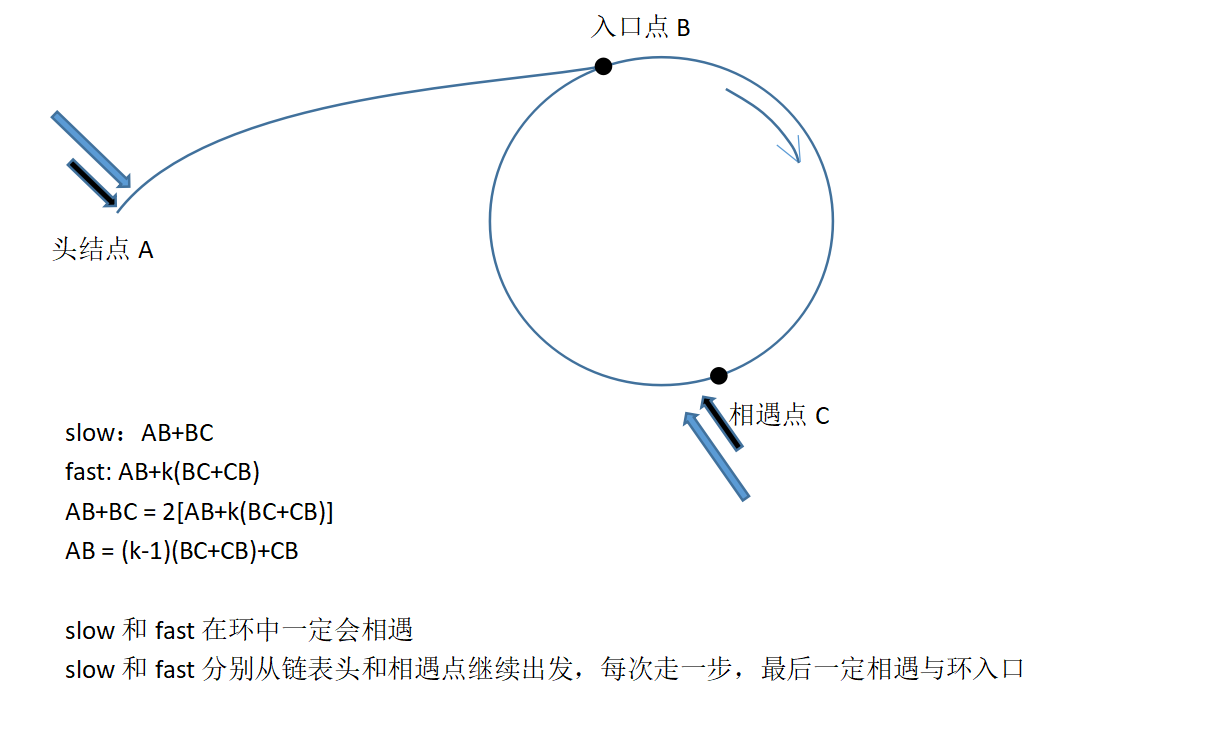
3

3

-7

8

1

NULL  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
slow：AB+BC  
fast: AB+k(BC+CB)  
AB+BC = 2[AB+k(BC+CB)]  
AB = (k-1)(BC+CB)+CB  
  
slow和fast在环中一定会相遇  
slow和fast分别从链表头和相遇点继续出发，每次走一步，最后一定相遇与环入口  


6

5

4

4

3

3

1

pre

last

cur

6

5

5

4

1

1

1

50

30

46

61

92

7

20

pop到NULL

7 6 4 9

5 3

7 6 4 9

pop到 NULL

5 3

MoveNode

Temp

2

Result

Stack2

Stack1

2

3

5

9

4

6

7

2

头结点A

相遇点C

入口点B

依次拿出个位、十位、百位...

令当前位为CurNum；

以百位为例：

找出CurNum的高位 HightNum 和 LowNum

CurNum = (n / digit) % 10 (digit 个十百千)

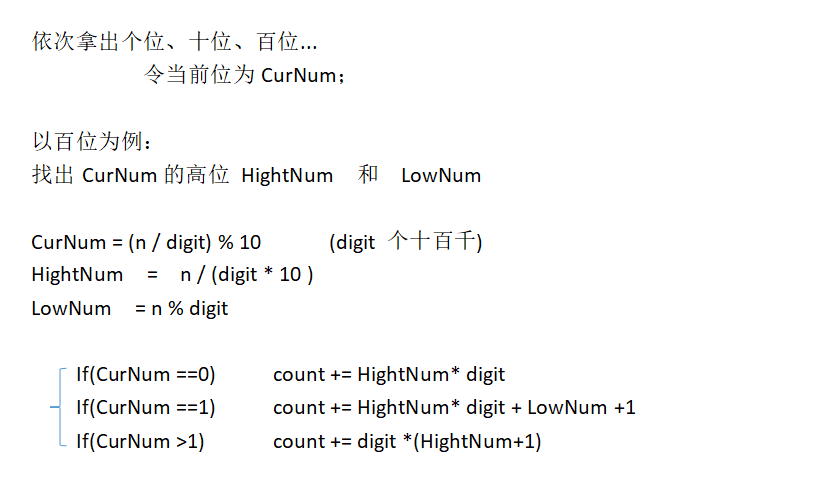
HightNum = n / (digit \* 10 )

LowNum = n % digit

If(CurNum ==0) count += HightNum\* digit

If(CurNum ==1) count += HightNum\* digit + LowNum +1

If(CurNum >1) count += digit \*(HightNum+1)



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 丑数数组 | 1 | 2 | 3 | 4 | 5 |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| \*2 | 2 | 4 | 6 | 8 | 10 |  |  |  |  |  |

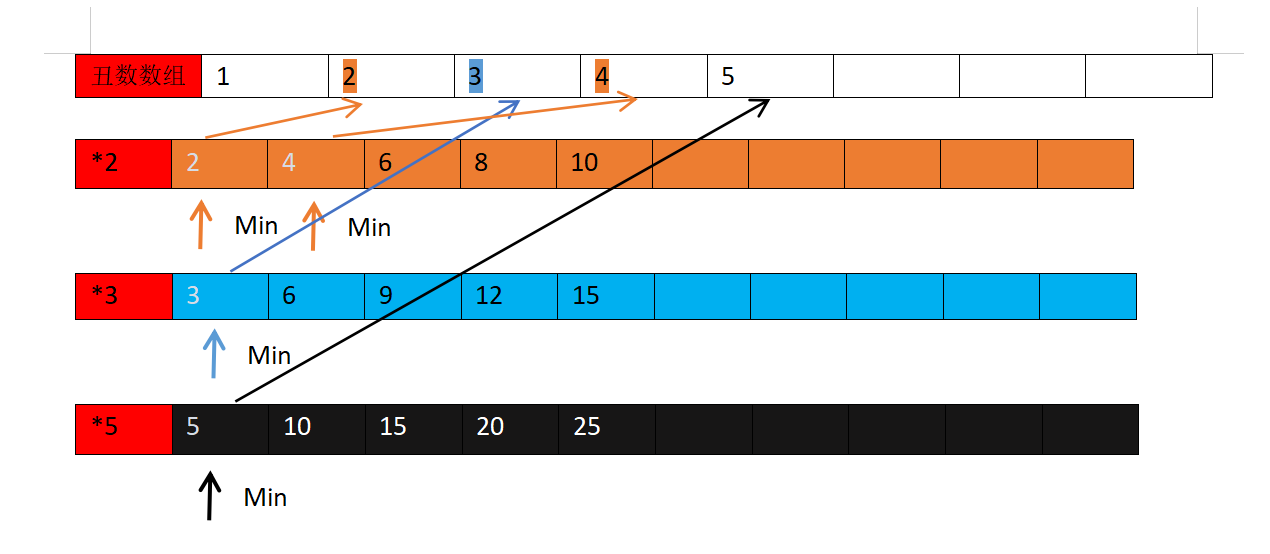
Min

Min

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| \*3 | 3 | 6 | 9 | 12 | 15 |  |  |  |  |  |

Min

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| \*5 | 5 | 10 | 15 | 20 | 25 |  |  |  |  |  |



Min

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 7 | 3 | 7 | 6 | 1 | 2 | 4 |

1^3^4^7^3^7^6^1^2^4 = 1^1^3^3^4^4^7^7^2^6 = 6^2

0001 0001 0010 0011 0011 0100 0100 0110 0111 0111

0100 0100 0111 0111 0110 0110 ^ 0010 = 0100 & 0001 = 0000

0001 0001 0011 0011 0010 0100右移0010&0001 =0000

0010右移0001&0001 =0001

0110 ^ 0010 可以确定1在不同的位置 右移了两次分组

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 4 | 7 | 7 | 6 |

4^4^7^7^6 = 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1 | 3 | 3 | 2 |

1^1^3^3^2 =2  
  
右移，高位补0，低位舍弃

左移，右补0

  
  
  
  
  
  
  
  
result tempArray  
  
result tempArray  
  
result tempArray

a

“ ”Student.

Student.

Student. a am I

“ ”a“ ”Student.

result tempArray  
  
result tempArray

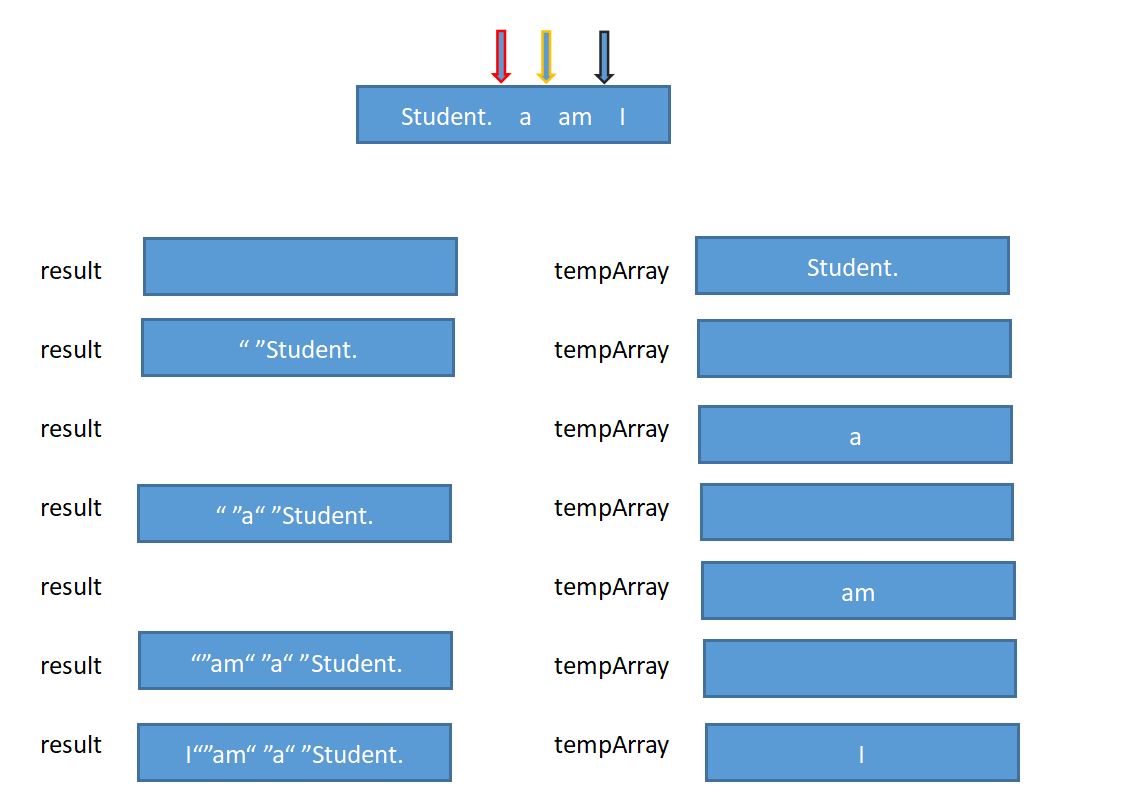
“”am“ ”a“ ”Student.

am

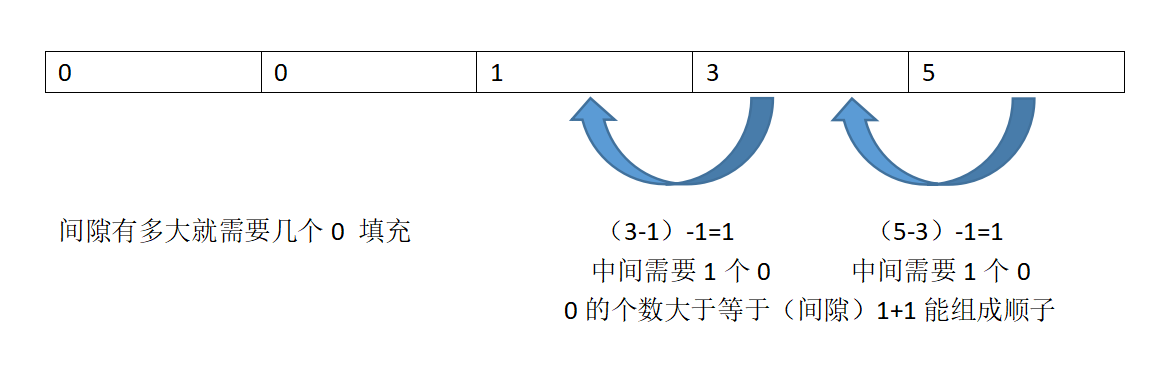
result tempArray

I“”am“ ”a“ ”Student.

I

result tempArray  
  
  


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 3 | 5 |

间隙有多大就需要几个0 填充 （3-1）-1=1 （5-3）-1=1  
 中间需要1个0 中间需要1个0  
 0的个数大于等于（间隙）1+1能组成顺子  
  
  
  
  
  
  
  
  
  
5 3 2 4 6

|  |
| --- |
|  |
| 2 |
| 3 |
| 5 |

当插入4 的时候，我们

发现2 左边3，右边是4。2出栈

2（3,4）

发现3左边5，右边是4。3出栈

3（5,4）

当插入6的时候

③

②

①

pClone

pClone

pCloned

pNode