PROBLEM 1 :

GIVEN AN ARRAY

$$A = [-1\ 2\ 3\ -4\ 4\ -5\ 8]$$

SORT IT SUCH THAT ALL THE NEGATIVE NUMBERS COME BEFORE ALL POSITIVE NUMBERS IN $O(n)$ TIME & $O(1)$ EXTRA SPACE

$$A = [-1\ -4\ -5\ 2\ 3\ 4\ 8]$$

PROBLEM 1 :

GIVEN AN ARRAY

$$A = [-1\ 2\ 3\ -4\ 4\ -5\ 8]$$

SORT IT SUCH THAT ALL THE NEGATIVE NUMBERS COME BEFORE ALL POSITIVE NUMBERS IN $O(n)$ TIME & $O(1)$ EXTRA SPACE

$$A = [-1\ -4\ -5\ 2\ 3\ 4\ 8]$$

ANS : FIND THE SMALLEST POSITIVE NUMBER, SAY x

PARTITION USING x AS PIVOT.

PROBLEM 1 :

GIVEN AN ARRAY

$$A = [-1\ 2\ 3\ -4\ 4\ -5\ 8]$$

SORT IT SUCH THAT ALL THE NEGATIVE NUMBERS COME BEFORE ALL POSITIVE NUMBERS IN $O(n)$ TIME & $O(1)$ EXTRA SPACE

$$A = [-1\ -4\ -5\ 2\ 3\ 4\ 8]$$

ANS : FIND THE SMALLEST POSITIVE NUMBER, SAY x } $O(n)$

PARTITION USING x AS PIVOT. } $O(n)$

② A SORTING ALGORITHM IS CALLED STABLE IF THE RELATIVE ORDER OF SIMILAR ELEMENT DONOT CHANGE IN THE SORTED OUTPUT

② A SORTING ALGORITHM IS CALLED STABLE IF THE RELATIVE ORDER OF SIMILAR ELEMENT DONOT CHANGE IN THE SORTED OUTPUT

(B, 3) (A, 2) (C, 3)

⇒ (A, 2) (B, 3) (C, 3)

② A SORTING ALGORITHM IS CALLED STABLE IF THE RELATIVE ORDER OF SIMILAR ELEMENT DONOT CHANGE IN THE SORTED OUTPUT

(B, 3) (A, 2) (C, 3)

⇒ (A, 2) (B, 3) (C, 3)

a) INSERTION
b) HEAP
c) MERGESORT
d) QUICKSORT

② A SORTING ALGORITHM IS CALLED STABLE IF THE RELATIVE ORDER OF SIMILAR ELEMENT DONOT CHANGE IN THE SORTED OUTPUT

(B, 3) (A, 2) (C, 3)

⇒ (A, 2) (B, 3) (C, 3)

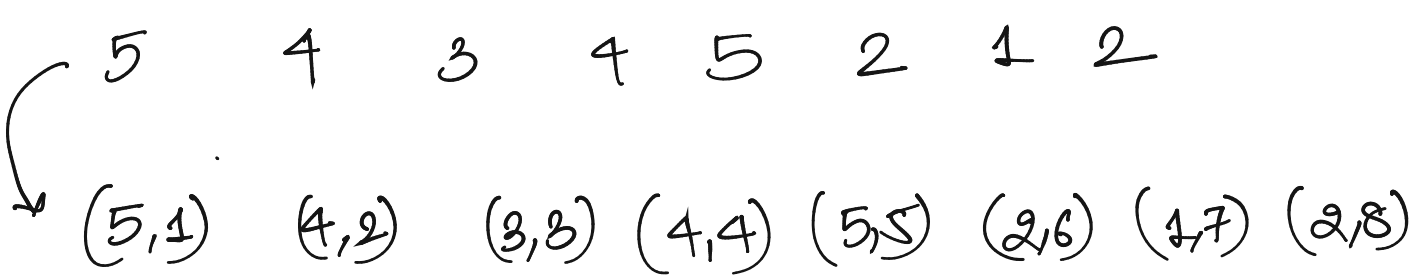
a) INSERTION YES
b) HEAP NO
c) MERGESORT YES
d) QUICKSORT NO

GIVEN A SORTING ALGORITHM WHICH IS NOT STABLE, SHOW HOW YOU CAN CHANGE IT TO MAKE IT STABLE.

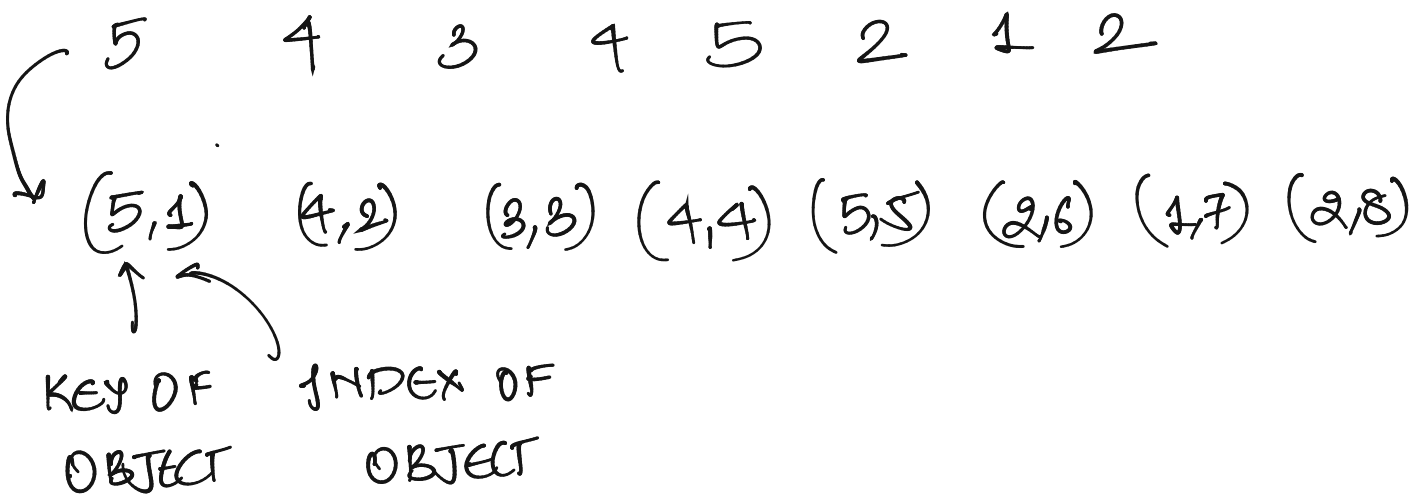
GIVEN A SORTING ALGORITHM WHICH IS NOT STABLE, SHOW HOW YOU CAN CHANGE IT TO MAKE IT STABLE.

5 4 3 4 5 2 1 2

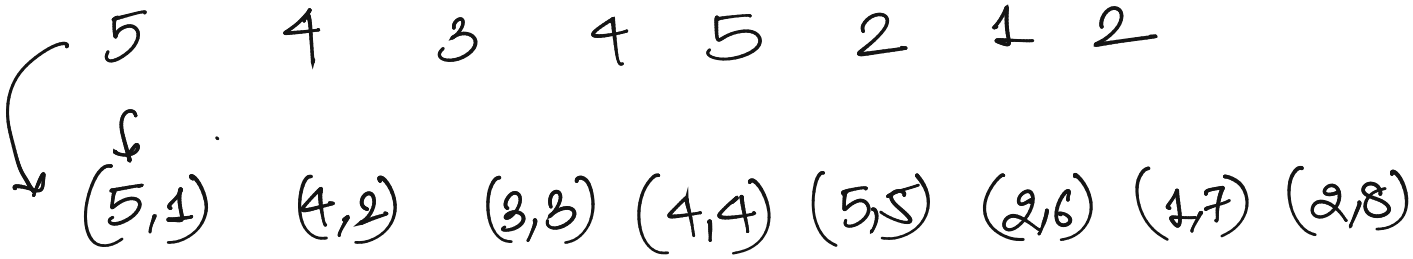
GIVEN A SORTING ALGORITHM WHICH IS NOT STABLE, SHOW HOW YOU CAN CHANGE IT TO MAKE IT STABLE.



GIVEN A SORTING ALGORITHM WHICH IS NOT STABLE, SHOW HOW YOU CAN CHANGE IT TO MAKE IT STABLE.



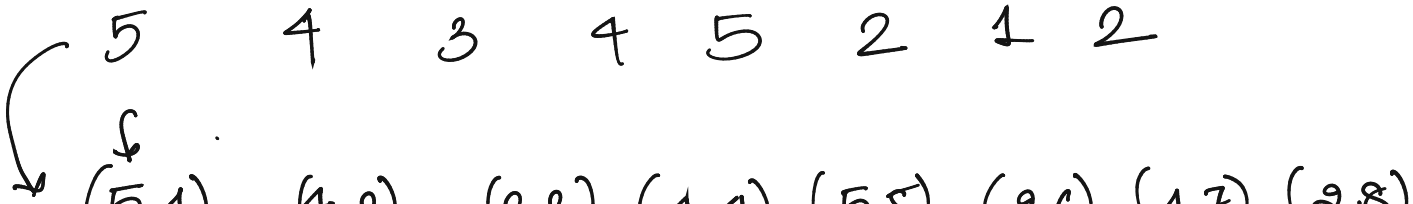
GIVEN A SORTING ALGORITHM WHICH IS NOT STABLE, SHOW HOW YOU CAN CHANGE IT TO MAKE IT STABLE.



SORT USING THE GIVEN ALGO WITH THE FOLLOWING CHANGE

$x < y$ IFF $x.key < y.key$
OR
$x.key = y.key$ f

GIVEN A SORTING ALGORITHM WHICH IS NOT STABLE, SHOW HOW YOU CAN CHANGE IT TO MAKE IT STABLE.



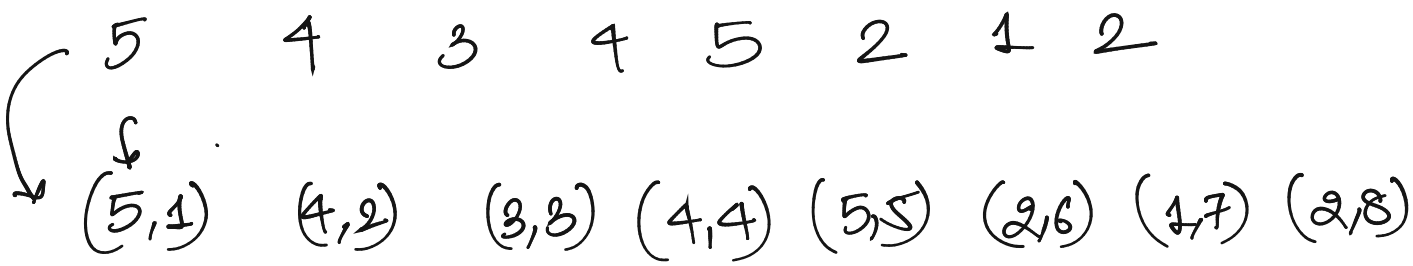
SORT USING THE GIVEN ALGO WITH THE FOLLOWING CHANGE

$x < y$ IFF $x.key < y.key$

OR

$x.key = y.key$ & $x.INDEX < y.INDEX$

GIVEN A SORTING ALGORITHM WHICH IS NOT STABLE, SHOW HOW YOU CAN CHANGE IT TO MAKE IT STABLE.



SORT USING THE GIVEN ALGO WITH THE FOLLOWING CHANGE

$x < y$ IFF $x.key < y.key$
OR
$x.key = y.key$ & $x.INDEX < y.INDEX$

JUST CHANGED THE COMPARISON OPERATION IN THE ALGORITHM.

③

$$A_1 = [1 \quad 5 \quad 9 \quad 12]$$

$$A_2 = [2 \quad 4 \quad 8 \quad 15]$$

③ $A_1 = [1 \quad 5 \quad 9 \quad 12]$

$A_2 = [2 \quad 4 \quad 8 \quad 15]$

MERGE ODD INDEX NUMBERS

$A_{ODD} = [1 \quad 2 \quad 8 \quad 9]$

MERGE EVEN INDEX NUMBERS

$A_{EVEN} = [4 \quad 5 \quad 12 \quad 15]$

③ $A_1 = [1 \quad 5 \quad 9 \quad 12]$

$A_2 = [2 \quad 4 \quad 8 \quad 15]$

MERGE ODD INDEX NUMBERS

$A_{ODD} = [1 \quad 2 \quad 8 \quad 9]$

MERGE EVEN INDEX NUMBERS

$A_{EVEN} = [4 \quad 5 \quad 12 \quad 15]$

ARRANGE THE NUMBERS IN SUCH A WAY THAT ALL ODD NUMBERS COME FROM A_{ODD} & EVEN NUMBERS FROM A_{EVEN}.

$A = [1 \quad 4 \quad 2 \quad 5 \quad 8 \quad 12 \quad 9 \quad 15]$

③ $A_1 = [1 \quad 5 \quad 9 \quad 12]$

$A_2 = [2 \quad 4 \quad 8 \quad 15]$

MERGE ODD INDEX NUMBERS

$A_{ODD} = [1 \quad 2 \quad 8 \quad 9]$

MERGE EVEN INDEX NUMBERS

$A_{EVEN} = [4 \quad 5 \quad 12 \quad 15]$

ARRANGE THE NUMBERS IN SUCH A WAY THAT ALL ODD NUMBERS COME FROM A_{ODD} & EVEN NUMBERS FROM A_{EVEN}.

$A = [1 \quad 4 \quad 2 \quad 5 \quad 8 \quad 12 \quad 9 \quad 15]$

IN GENERAL,

$A_{ODD} = [a_1 \quad a_2 \quad \ldots \quad a_n]$

$A_{EVEN} = [b_1 \quad b_2 \quad \ldots \quad b_n]$

$A = [a_1 \; b_1 \; a_2 \; b_2 \; \ldots \; a_n \; b_n]$

(3) $A_1 = [1 \quad 5 \quad 9 \quad 12]$

$A_2 = [2 \quad 4 \quad 8 \quad 15]$

IN GENERAL,

$A_{ODD} = [a_1 \quad a_2 \quad \ldots \quad a_n]$

$A_{EVEN} = [b_1 \quad b_2 \quad \ldots \quad b_n]$

$A = [a_1 \; b_1 \; a_2 \; b_2 \; \ldots \; a_n \; b_n]$

(1) SHOW THAT a_1 IS THE MINIMUM NUMBER IN $A_1 \cup A_2$

③ $A_1 = [1 \quad 5 \quad 9 \quad 12]$

$A_2 = [2 \quad 4 \quad 8 \quad 15]$

IN GENERAL,

$A_{ODD} = [a_1 \quad a_2 \quad \ldots \quad a_n]$

$A_{EVEN} = [b_1 \quad b_2 \quad \ldots \quad b_n]$

$A = [a_1 \; b_1 \; a_2 \; b_2 \; \ldots \; a_n \; b_n]$

① SHOW THAT a_1 IS THE MINIMUM NUMBER IN $A_1 \cup A_2$

$a_1 = \min \{ A_1[1], A_2[1] \}$

③ $A_1 = [1 \quad 5 \quad 9 \quad 12]$

$A_2 = [2 \quad 4 \quad 8 \quad 15]$

IN GENERAL,

$A_{ODD} = [a_1 \quad a_2 \quad \ldots \quad a_n]$

$A_{EVEN} = [b_1 \quad b_2 \quad \ldots \quad b_n]$

$A = [a_1 \; b_1 \; a_2 \; b_2 \; \ldots \; a_n \; b_n]$

① SHOW THAT a_1 IS THE MINIMUM NUMBER IN $A_1 \cup A_2$

$a_1 = \min \{ A_1[1], A_2[1] \}$

② SHOW HOW WILL YOU SORT A IN n COMPARISONS & SWAPS ?

③ $A_1 = [1 \quad 5 \quad 9 \quad 12]$

$A_2 = [2 \quad 4 \quad 8 \quad 15]$

IN GENERAL,

$A_{ODD} = [a_1 \quad a_2 \quad \ldots \quad a_n]$

$A_{EVEN} = [b_1 \quad b_2 \quad \ldots \quad b_n]$

$A = [a_1 \; b_1 \; a_2 \; b_2 \; \ldots \; a_n \; b_n]$

① SHOW THAT a_1 IS THE MINIMUM NUMBER IN $A_1 \cup A_2$

$a_1 = \min \{ A_1[1], A_2[1] \}$

② SHOW HOW WILL YOU SORT A IN n COMPARISONS & SWAPS?

```
FOR  i = 0 to n-1
    If  A[2i+1] > A[2i+2]
        SWAP( A[2i+1], A[2i+2] )
```

③ $A_1 = [1 \quad 5 \quad 9 \quad 12]$

$A_2 = [2 \quad 4 \quad 8 \quad 15]$

IN GENERAL,

$A_{ODD} = [a_1 \quad a_2 \quad \ldots \quad a_n]$

$A_{EVEN} = [b_1 \quad b_2 \quad \ldots \quad b_n]$

$A = [a_1 \; b_1 \; a_2 \; b_2 \; \ldots \; a_n \; b_n]$

① SHOW THAT a_1 IS THE MINIMUM NUMBER IN $A_1 \cup A_2$

$a_1 = \min\{A_1[1], A_2[1]\}$

② SHOW HOW WILL YOU SORT A IN n COMPARISONS & SWAPS?

```
FOR  i = 0 to n-1
    If  A[2i+1] > A[2i+2]
        SWAP( A[2i+1], A[2i+2] )
```

Q: WHY DOES THIS SORT THE ARRAY?

③ $A_1 = [1 \quad 5 \quad 9 \quad 12]$

$A_2 = [2 \quad 4 \quad 8 \quad 15]$

IN GENERAL,

$A_{ODD} = [a_1 \quad a_2 \quad \ldots \quad a_n]$

$A_{EVEN} = [b_1 \quad b_2 \quad \ldots \quad b_n]$

$A = [a_1 \; b_1 \; a_2 \; b_2 \; \ldots \; a_n \; b_n]$

① SHOW THAT a_1 IS THE MINIMUM NUMBER IN $A_1 \cup A_2$

$a_1 = \min\{A_1[1], A_2[1]\}$

② SHOW HOW WILL YOU SORT A IN n COMPARISONS & SWAPS?

```
FOR  i = 0 to n-1
    If  A[2i+1] > A[2i+2]
        SWAP( A[2i+1], A[2i+2])
```

Q: WHY DOES THIS SORT THE ARRAY?

WHY IS $b_2 > a_2$

LEMMA : $b_2 > a_2$

LEMMA : $b_2 > a_2$

LEMMA : AT LEAST 3 NUMBERS ARE LESS THAN b_2 IN $A_1 \cup A_2$

LEMMA : $b_2 > a_2$

LEMMA : AT LEAST 3 NUMBERS ARE LESS THAN b_2 IN $A_1 \cup A_2$

{ IMPLIES → b_2 CANNOT LIE IN THE SECOND OR THIRD INDEX }

LEMMA : $b_2 > a_2$

LEMMA : AT LEAST 3 NUMBERS ARE LESS THAN b_2 IN $A_1 \cup A_2$

{ IMPLIES → b_2 CANNOT LIE IN THE SECOND OR THIRD INDEX }

PROOF :

$A_1 = [\ b_1\ ]$

$A_2 = [\ b_2\ ]$

LEMMA : $b_2 > a_2$

LEMMA : AT LEAST 3 NUMBERS ARE LESS THAN b_2 IN $A_1 \cup A_2$

{ IMPLIES → b_2 CANNOT LIE IN THE SECOND OR THIRD INDEX }

PROOF :

$A_1 = [\ b_1 \quad]$

$A_2 = [\quad b_2 \quad]$

$b_1 < b_2$ — FOUND A ELEMENT $< b_2$

LEMMA : $b_2 > a_2$

LEMMA : AT LEAST 3 NUMBERS ARE LESS THAN b_2 IN $A_1 \cup A_2$

{ IMPLIES → b_2 CANNOT LIE IN THE SECOND OR THIRD INDEX }

PROOF :

$A_1 = [\, y \; b_1 \quad]$

$A_2 = [\quad x \; b_2 \quad]$

$b_1 < b_2$ — FOUND A ELEMENT $< b_2$

LOOK AT ELEMENT JUST TO THE LEFT OF b_1 & b_2

LEMMA : $b_2 > a_2$

LEMMA : AT LEAST 3 NUMBERS ARE LESS THAN b_2 IN $A_1 \cup A_2$

{ IMPLIES → b_2 CANNOT LIE IN THE SECOND OR THIRD INDEX }

PROOF :

$A_1 = [\ y \ b_1 \ \ \ ]$

$A_2 = [\ \ \ x \ b_2 \ \ \ ]$

$b_1 < b_2$ — FOUND A ELEMENT $< b_2$

LOOK AT ELEMENT JUST TO THE LEFT OF b_1 & b_2

$x < b_2$ — FOUND ANOTHER ELEMENT $< b_2$

LEMMA : $b_2 > a_2$

LEMMA : AT LEAST 3 NUMBERS ARE LESS THAN b_2 IN $A_1 \cup A_2$

{ IMPLIES → b_2 CANNOT LIE IN THE SECOND OR THIRD INDEX }

PROOF :

$$A_1 = [\, y \; b_1 \quad]$$
$$A_2 = [\quad x \; b_2 \quad]$$

$b_1 < b_2$ — FOUND A ELEMENT $< b_2$

LOOK AT ELEMENT JUST TO THE LEFT OF b_1 & b_2

$x < b_2$ — FOUND ANOTHER ELEMENT $< b_2$

$y < b_1$ BUT $b_1 < b_2$

LEMMA : $b_2 > a_2$

LEMMA : AT LEAST 3 NUMBERS ARE LESS THAN b_2 IN $A_1 \cup A_2$

{ IMPLIES → b_2 CANNOT LIE IN THE SECOND OR THIRD INDEX }

PROOF :

$A_1 = [\ y \quad b_1 \quad \ldots \quad]$

$A_2 = [\ \ldots \quad x \quad b_2 \quad \ldots \quad]$

$b_1 < b_2$ — FOUND A ELEMENT $< b_2$

LOOK AT ELEMENT JUST TO THE LEFT OF b_1 & b_2

$x < b_2$ — FOUND ANOTHER ELEMENT $< b_2$

$y < b_1$ BUT $b_1 < b_2$

$\Rightarrow y < b_2$ — FOUND ANOTHER ELEMENT $< b_2$

LEMMA : $b_2 > a_2$

LEMMA : AT LEAST NUMBERS ARE LESS THAN b_i IN $A_1 \cup A_2$

{ IMPLIES → b_2 CANNOT LIE IN THE SECOND OR THIRD INDEX }

LEMMA : $b_2 > a_2$

LEMMA : AT LEAST $2i-1$ NUMBERS ARE LESS THAN b_i IN $A_1 \cup A_2$

{ IMPLIES → b_i CANNOT LIE IN $A[1 \ldots 2i-1]$ }

LEMMA : THERE ARE AT LEAST 3 NUMBERS LESS THAN a_3 IN $A_1 \cup A_2$

LEMMA : $b_2 > a_2$

LEMMA : AT LEAST $2i-1$ NUMBERS ARE LESS THAN b_i IN $A_1 \cup A_2$

{ IMPLIES → b_i CANNOT LIE IN $A[1 \ldots 2i-1]$ }

LEMMA : THERE ARE AT LEAST 3 NUMBERS LESS THAN a_3 IN $A_1 \cup A_2$

{ IMPLIES → a_3 CANNOT LIE IN THE SECOND & THIRD CELL OF A }

LEMMA : $b_2 > a_2$

LEMMA : AT LEAST $2i-1$ NUMBERS ARE LESS THAN b_i IN $A_1 \cup A_2$

{ IMPLIES → b_i CANNOT LIE IN $A[1 \ldots 2i-1]$ }

LEMMA : THERE ARE AT LEAST 3 NUMBERS LESS THAN a_3 IN $A_1 \cup A_2$

{ IMPLIES → a_3 CANNOT LIE IN THE SECOND & THIRD CELL OF A }

PROOF : $A_1 = [\, a_1 \qquad\qquad]$

$A_2 = [\, a_2 \qquad a_3 \qquad]$

LEMMA : $b_2 > a_2$

LEMMA : AT LEAST $2i-1$ NUMBERS ARE LESS THAN b_i IN $A_1 \cup A_2$

{ IMPLIES → b_i CANNOT LIE IN $A[1 \ldots 2i-1]$ }

LEMMA : THERE ARE AT LEAST 3 NUMBERS LESS THAN a_3 IN $A_1 \cup A_2$

{ IMPLIES → a_3 CANNOT LIE IN THE SECOND & THIRD CELL OF A }

PROOF : $A_1 = [\, a_1 \quad\quad\quad\quad\quad\quad]$

$A_2 = [\, a_2 \quad\quad a_3 \quad\quad\quad]$

$a_1 < a_3$ — FOUND A NUMBER $< a_3$

$a_2 < a_3$ — FOUND ANOTHER NUMBER $< a_3$

LEMMA : $b_2 > a_2$

LEMMA : AT LEAST $2i-1$ NUMBERS ARE LESS THAN b_i IN $A_1 \cup A_2$

{ IMPLIES → b_i CANNOT LIE IN $A[1 \ldots 2i-1]$ }

LEMMA : THERE ARE AT LEAST 3 NUMBERS LESS THAN a_3 IN $A_1 \cup A_2$

{ IMPLIES → a_3 CANNOT LIE IN THE SECOND & THIRD CELL OF A }

PROOF : $A_1 = [a_1 \quad]$

$A_2 = [a_2 \quad x \quad a_3 \quad]$

$a_1 < a_3$ — FOUND A NUMBER $< a_3$

$a_2 < a_3$ — FOUND ANOTHER NUMBER $< a_3$

$x < a_3$ — FOUND THIRD NUMBER

LEMMA : $b_2 > a_2$

LEMMA : AT LEAST $2i-1$ NUMBERS ARE LESS THAN b_i IN $A_1 \cup A_2$

{ IMPLIES → b_i CANNOT LIE IN $A[1 \ldots 2i-1]$ }

LEMMA : THERE ARE ATLEAST NUMBERS LESS THAN a_i IN $A_1 \cup A_2$

{ IMPLIES → a_3 CANNOT LIE IN THE SECOND & THIRD CELL OF A }

LEMMA : $b_2 > a_2$

LEMMA : AT LEAST $2i-1$ NUMBERS ARE LESS THAN b_i IN $A_1 \cup A_2$

{ IMPLIES → b_i CANNOT LIE IN $A[1 \ldots 2i-1]$ }

LEMMA : THERE ARE ATLEAST $2i-3$ NUMBERS LESS THAN a_i IN $A_1 \cup A_2$

{ IMPLIES → a_i CANNOT LIE IN THE $A[1 \ldots 2i-3]$ }

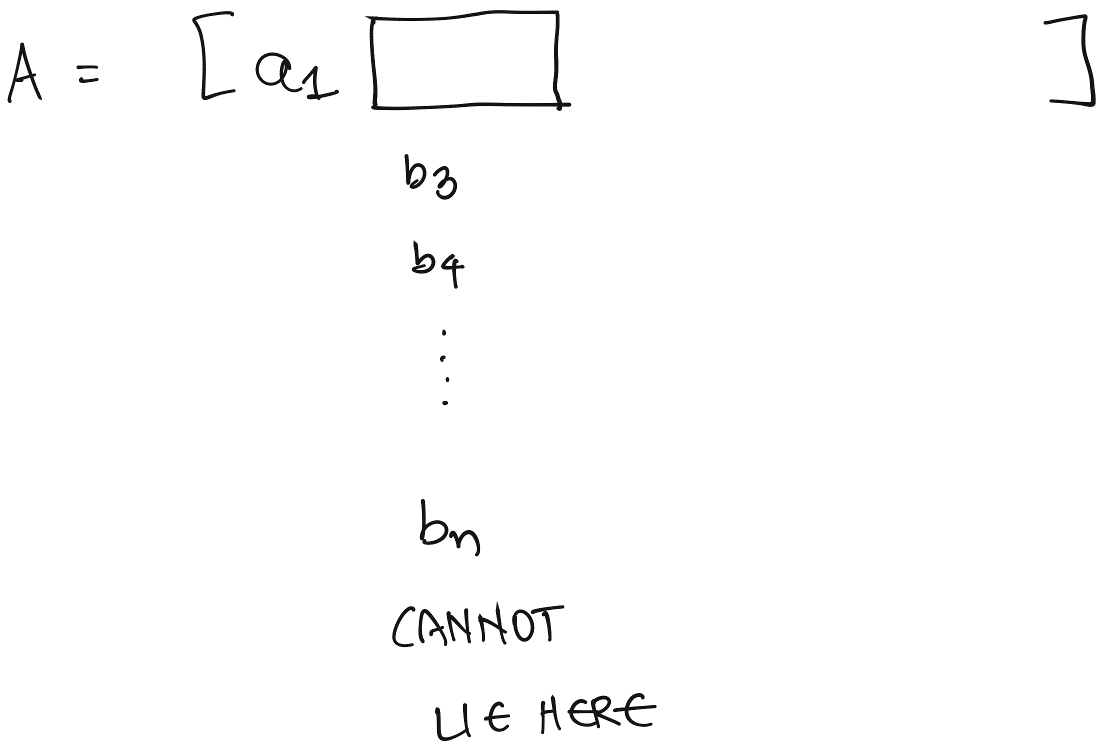
LEMMA: $b_2 > a_2$

LEMMA: AT LEAST $2i-1$ NUMBERS ARE LESS THAN b_i IN $A_1 \cup A_2$

{ IMPLIES → b_i CANNOT LIE IN $A[1 \ldots 2i-1]$ }

LEMMA: THERE ARE ATLEAST $2i-3$ NUMBERS LESS THAN a_i IN $A_1 \cup A_2$

{ IMPLIES → a_i CANNOT LIE IN THE $A[1 \ldots 2i-3]$ }



LEMMA : $b_2 > a_2$

LEMMA : AT LEAST $2i-1$ NUMBERS ARE LESS THAN b_i IN $A_1 \cup A_2$

{ IMPLIES → b_i CANNOT LIE IN $A[1 \ldots 2i-1]$ }

LEMMA : THERE ARE ATLEAST $2i-3$ NUMBERS LESS THAN a_i IN $A_1 \cup A_2$

{ IMPLIES → a_i CANNOT LIE IN THE $A[1 \ldots 2i-3]$ }

$A = [a_1$ [ ] $]$

$b_3 \quad a_3$

$b_4 \quad a_4$

$\vdots \quad \vdots$

$b_n \quad a_n$

CANNOT

LIE HERE

LEMMA : $b_2 > a_2$

LEMMA : AT LEAST $2i-1$ NUMBERS ARE LESS THAN b_i IN $A_1 \cup A_2$

{ IMPLIES → b_i CANNOT LIE IN $A[1 \ldots 2i-1]$ }

LEMMA : THERE ARE ATLEAST $2i-3$ NUMBERS LESS THAN a_i IN $A_1 \cup A_2$

{ IMPLIES → a_i CANNOT LIE IN THE $A[1 \ldots 2i-3]$ }

$A = [\, a_1 \square \quad]$

b_3 a_3

b_4 a_4

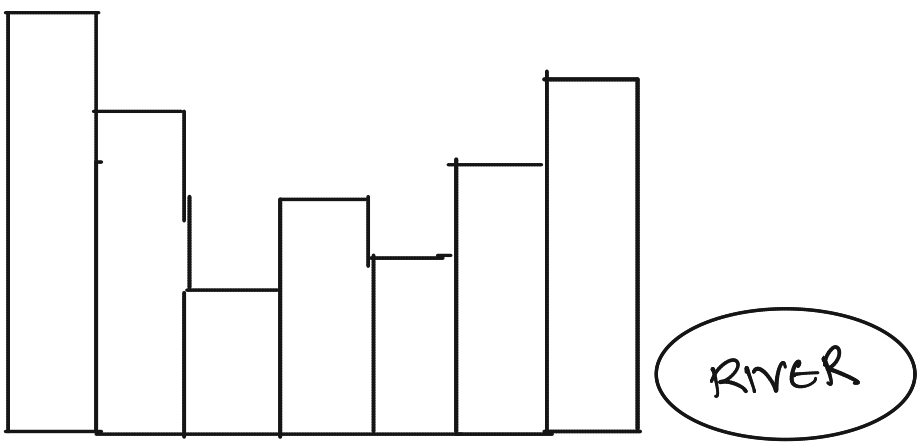
⋮ ⋮

b_n a_n

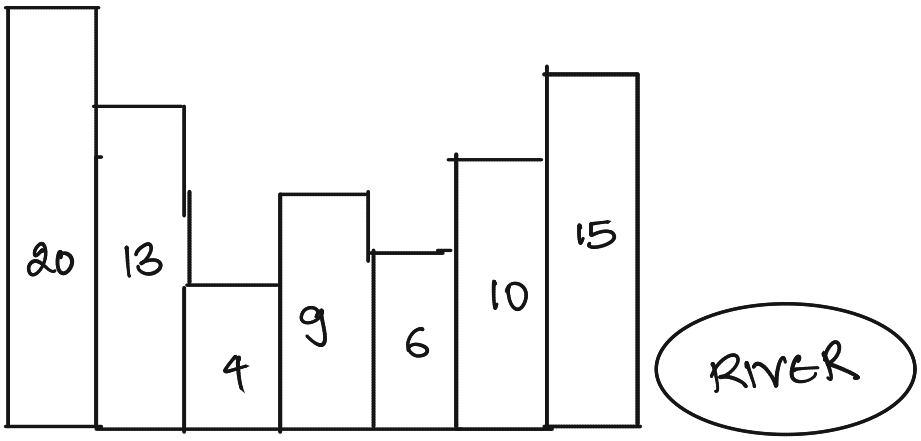
CANNOT LIE HERE

$\Rightarrow$ ONLY b_1 & a_2 CAN LIE IN SECOND & THIRD CELL OF A.

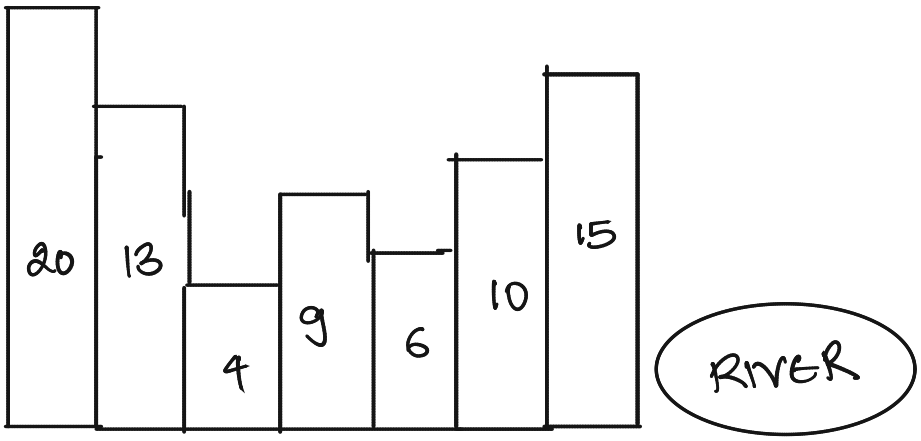
④



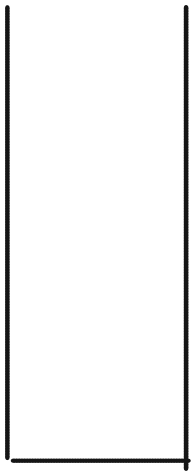
④



④

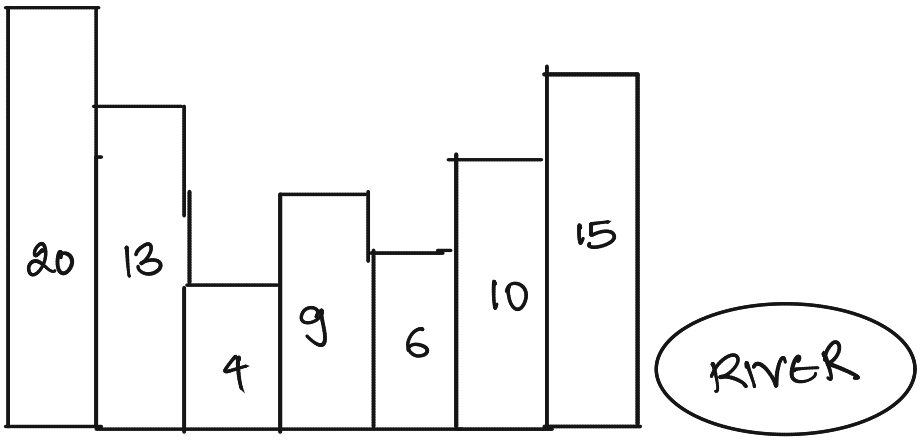


USE A STACK.

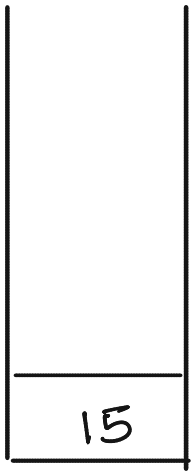


STACK CONTAINS ALL BUILDINGS THAT MAY POTENTIALLY BLOCK SOME BUILDINGS TO ITS LEFT

④

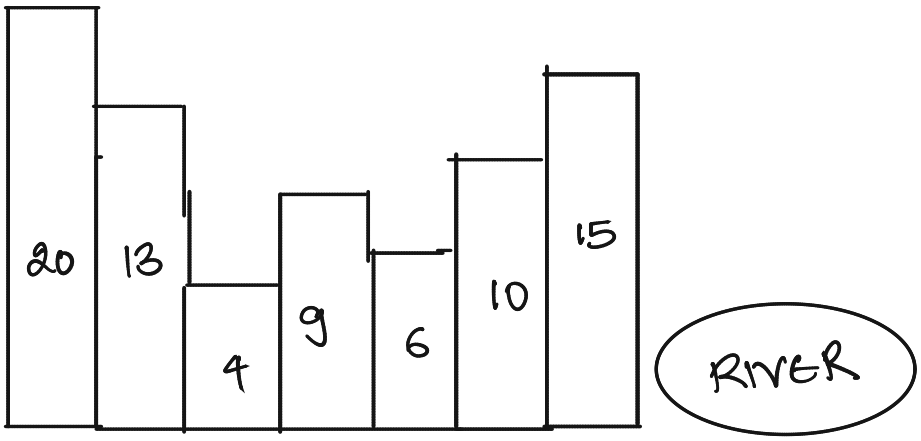


USE A STACK.

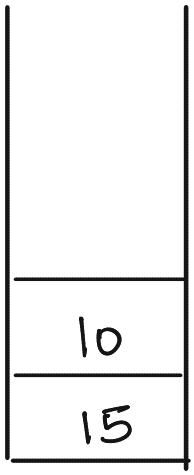


STACK CONTAINS ALL BUILDINGS THAT MAY POTENTIALLY BLOCK SOME BUILDINGS TO ITS LEFT

④

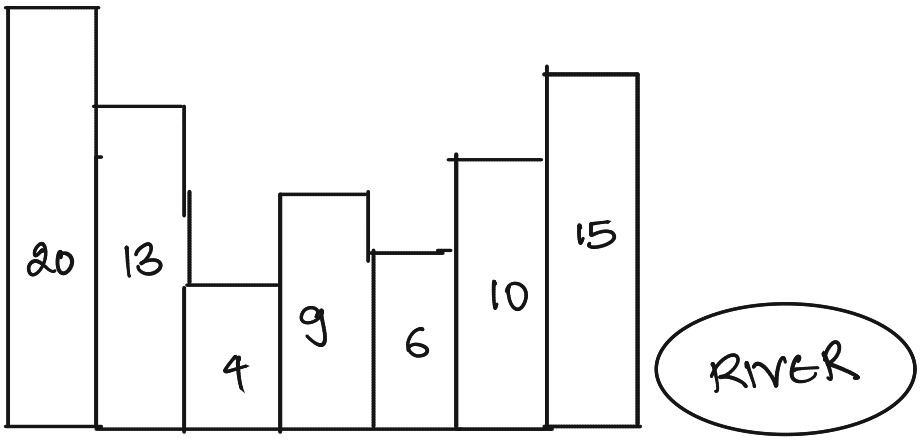


USE A STACK.

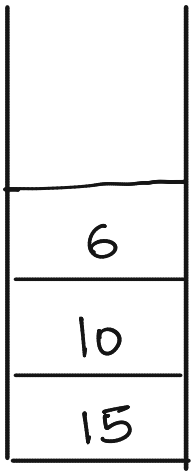


STACK CONTAINS ALL BUILDINGS THAT MAY POTENTIALLY BLOCK SOME BUILDINGS TO ITS LEFT

④

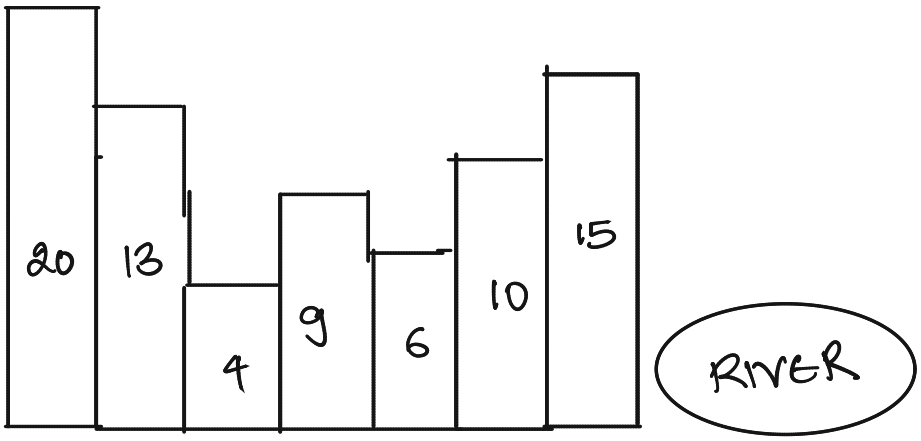


USE A STACK.

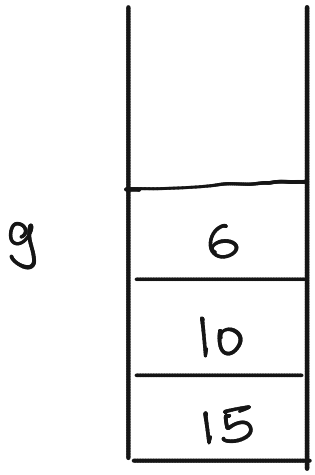


STACK CONTAINS ALL BUILDINGS THAT MAY POTENTIALLY BLOCK SOME BUILDINGS TO ITS LEFT

④

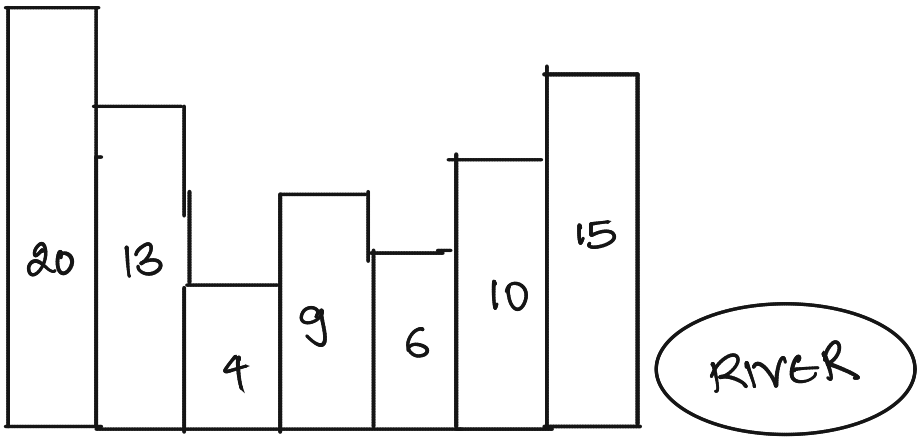


USE A STACK.

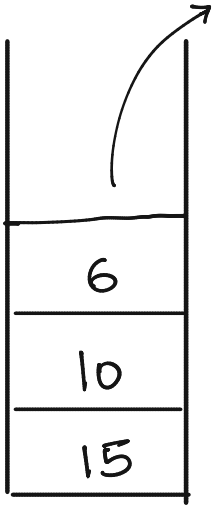


STACK CONTAINS ALL BUILDINGS THAT MAY POTENTIALLY BLOCK SOME BUILDINGS TO ITS LEFT

④

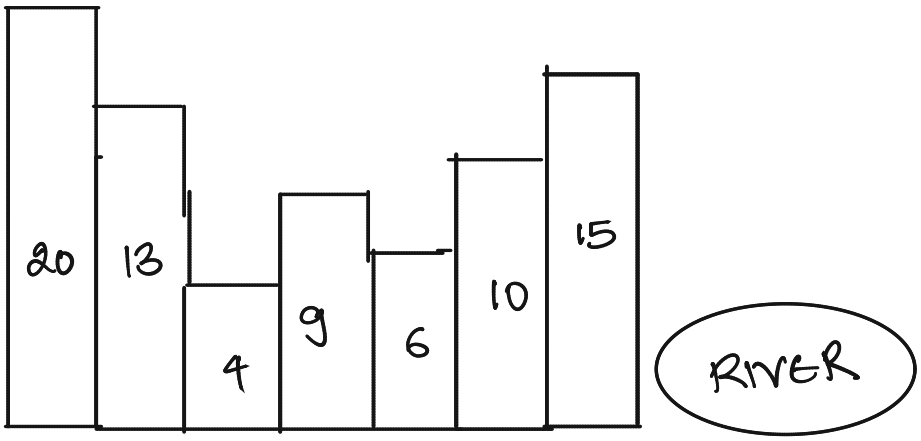


USE A STACK.

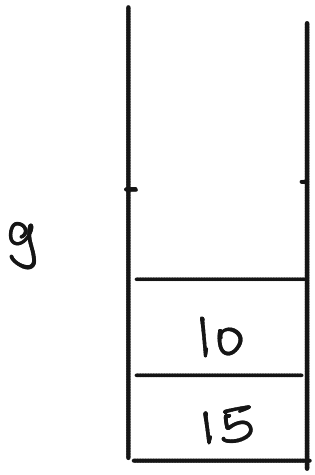


STACK CONTAINS ALL BUILDINGS THAT MAY POTENTIALLY BLOCK SOME BUILDINGS TO ITS LEFT

④

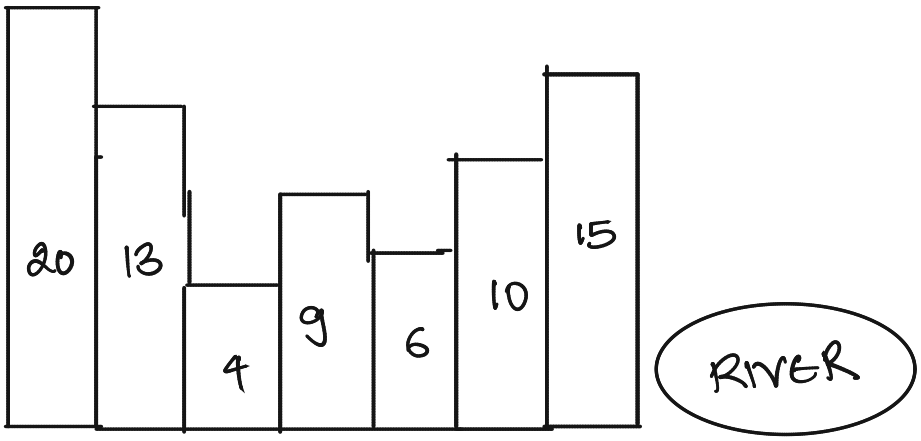


USE A STACK.

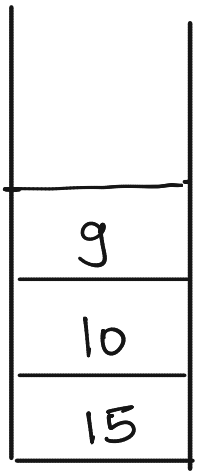


STACK CONTAINS ALL BUILDINGS THAT MAY POTENTIALLY BLOCK SOME BUILDINGS TO ITS LEFT

④

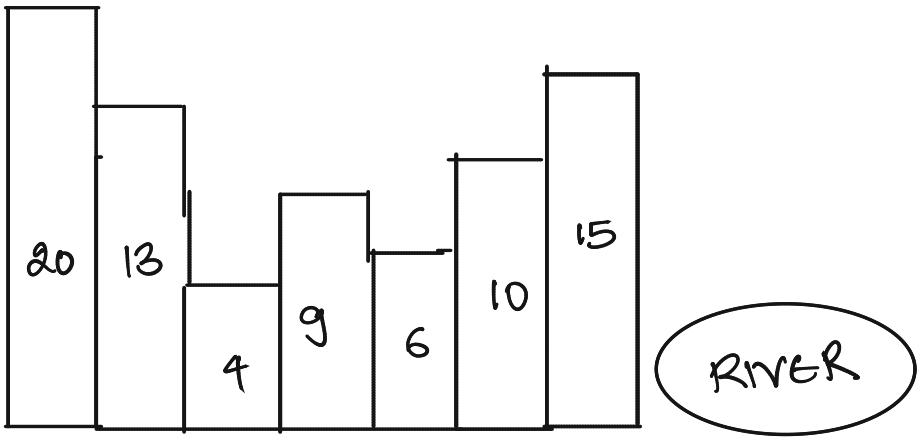


USE A STACK.

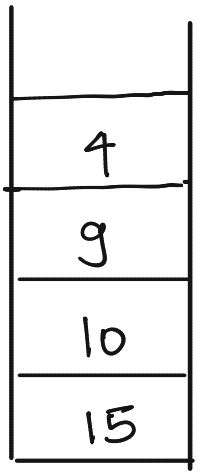


STACK CONTAINS ALL BUILDINGS THAT MAY POTENTIALLY BLOCK SOME BUILDINGS TO ITS LEFT

④

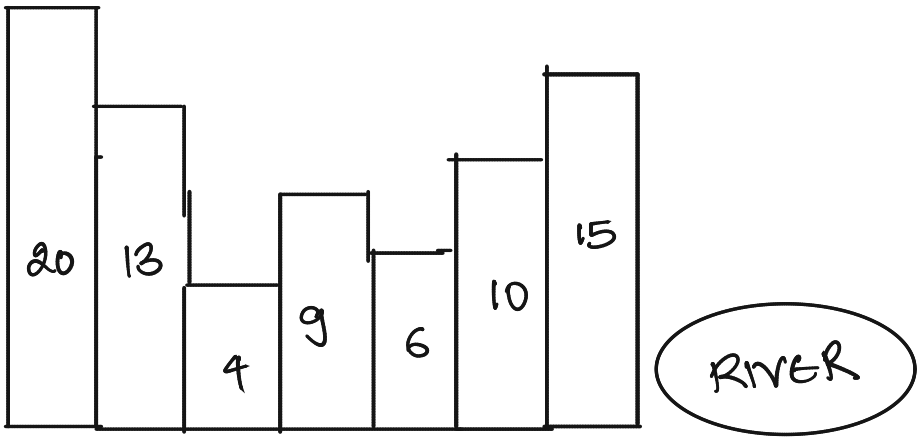


USE A STACK.

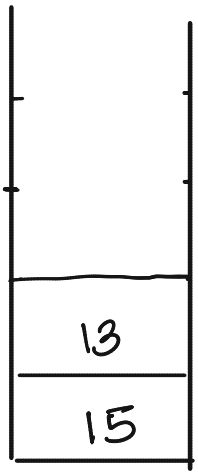


STACK CONTAINS ALL BUILDINGS THAT MAY POTENTIALLY BLOCK SOME BUILDINGS TO ITS LEFT

④

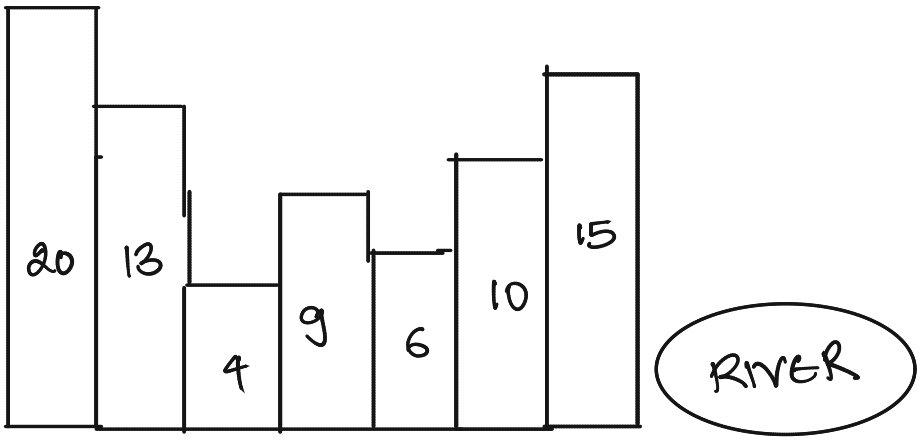


USE A STACK.

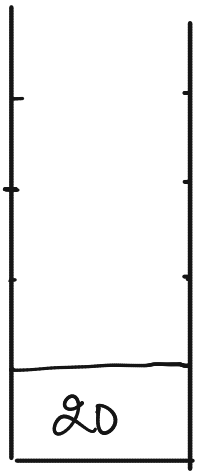


STACK CONTAINS ALL BUILDINGS THAT MAY POTENTIALLY BLOCK SOME BUILDINGS TO ITS LEFT

④



USE A STACK.



STACK CONTAINS ALL BUILDINGS THAT MAY POTENTIALLY BLOCK SOME BUILDINGS TO ITS LEFT

```
S.PUSH(A[n])
PRINT " BUILDING n CAN SEE THE RIVER

FOR ( i= n-1  TO  1)
{
```

```
S.PUSH(A[n])
PRINT " BUILDING n CAN SEE THE RIVER

FOR ( i= n-1  TO  1)
{
      WHILE ( A[i] > S.TOP())
      {   S.POP();
      }
```

```
S.PUSH(A[n])
PRINT " BUILDING n CAN SEE THE RIVER

FOR ( i= n-1  TO  1)
{
      WHILE ( A[i] > S.TOP())
      {   S.POP();
      }

      IF ( S IS EMPTY)
          PRINT " BUILDING  i .  CAN SEE
                        THE RIVER"
      ELSE
          PRINT " BUILDING i CANNOT SEE
            THE RIVER DUE TO BUILDING
             S.TOP() "
```

```
S.PUSH(A[n])
PRINT "BUILDING n CAN SEE THE RIVER

FOR( i= n-1  TO  1)
{
      WHILE ( A[i] > S.TOP())
      {   S.POP();
      }

      IF ( S IS EMPTY)
         PRINT "BUILDING i. CAN SEE
                      THE RIVER"
      ELSE
         PRINT "BUILDING i CANNOT SEE
           THE RIVER DUE TO BUILDING
            S.TOP()"

      S.PUSH(A[i]);
}
```

```
S.PUSH(A[n])
PRINT " BUILDING n CAN SEE THE RIVER

FOR ( i= n-1  TO  1)
{
      WHILE ( A[i] > S.TOP())
      {   S.POP();
      }

      IF ( S IS EMPTY)
         PRINT " BUILDING  i .  CAN SEE
                       THE RIVER"
      ELSE
         PRINT " BUILDING i CANNOT SEE
           THE RIVER DUE TO BUILDING
            S.TOP() "

      S.PUSH (A[i]);
}
```

RUNNING TIME:

```
S.PUSH(A[n])
PRINT " BUILDING n CAN SEE THE RIVER

FOR ( i= n-1  TO  1)
{
      WHILE ( A[i] > S.TOP())
      {   S.POP();
      }

      IF ( S IS EMPTY)
          PRINT " BUILDING  i .  CAN SEE
                        THE RIVER"
      ELSE
          PRINT " BUILDING i CANNOT SEE
            THE RIVER DUE TO BUILDING
             S.TOP() "

      S.PUSH (A[i]);   <- A BUILDING IS
}                        PUSHED JUST ONCE
                         ON THE STACK
```

RUNNING TIME:

```
S.PUSH(A[n])
PRINT "BUILDING n CAN SEE THE RIVER

FOR ( i= n-1  TO  1)
{
      WHILE ( A[i] > S.TOP())
      {   S.POP();        → SO IT CAN BE
      }                     POPPED JUST ONCE
                            OUT OF THE STACK

      IF ( S IS EMPTY)
          PRINT "BUILDING i CAN SEE
                 THE RIVER"
      ELSE
          PRINT "BUILDING i CANNOT SEE
            THE RIVER DUE TO BUILDING
             S.TOP()"

      S.PUSH(A[i]);  ← A BUILDING IS
}                      PUSHED JUST ONCE
                       ON THE STACK
```

RUNNING TIME:

```
S.PUSH(A[n])
PRINT "BUILDING n CAN SEE THE RIVER

FOR ( i= n-1  TO  1)
{
      WHILE ( A[i] > S.TOP())
      {   S.POP();        → SO IT CAN BE
      }                     POPPED JUST ONCE
                            OUT OF THE STACK

      IF ( S IS EMPTY)
          PRINT "BUILDING i. CAN SEE
                 THE RIVER"
      ELSE
          PRINT "BUILDING i CANNOT SEE
            THE RIVER DUE TO BUILDING
             S.TOP()"

      S.PUSH(A[i]);  ← A BUILDING IS
}                      PUSHED JUST ONCE
                       ON THE STACK
```

RUNNING TIME: $O(n)$

CORRECTNESS: TO SHOW THAT FOR i, IF THE NEAREST BUILDING j BLOCKS ITS VIEW OF THE RIVER, THEN OUR ALGO FINDS BUILDING j AT ITERATION i.

ELSE OUR ALGORITHM DECLARES THAT BULDING i CAN SEE THE RIVER (AT ITERATION i)

CORRECTNESS: TO SHOW THAT FOR i, IF THE NEAREST BUILDING j BLOCKS ITS VIEW OF THE RIVER, THEN OUR ALGO FINDS BUILDING j AT ITERATION i.
ELSE OUR ALGORITHM DECLARES THAT BULDING i CAN SEE THE RIVER (AT ITERATION i)

LEMMA: AT THE END OF ITERATION i, STACK S CONTAINS BUILDING j $(j > i)$ IF

CORRECTNESS: TO SHOW THAT FOR i, IF THE NEAREST BUILDING j BLOCKS ITS VIEW OF THE RIVER, THEN OUR ALGO FINDS BUILDING j AT ITERATION i.
ELSE OUR ALGORITHM DECLARES THAT BULDING i CAN SEE THE RIVER (AT ITERATION i)

LEMMA: AT THE END OF ITERATION i, STACK S CONTAINS BUILDING j $(j > i)$ IF $A[i \ldots j]$ DOES NOT CONTAIN ANY BUILDING $> A[j]$.

CORRECTNESS: TO SHOW THAT FOR i, IF THE NEAREST BUILDING j BLOCKS ITS VIEW OF THE RIVER, THEN OUR ALGO FINDS BUILDING j AT ITERATION i.
ELSE OUR ALGORITHM DECLARES THAT BULDING i CAN SEE THE RIVER (AT ITERATION i)

LEMMA: AT THE END OF ITERATION i, STACK S CONTAINS BUILDING j $(j > i)$ IF $A[i \ldots j]$ DOES NOT CONTAIN ANY BUILDING $> A[j]$.

PROOF: BY INDUCTION

$i = n$

BEFORE THE START OF FOR LOOP S CONTAINS BULDING n AND $A[n \ldots n]$ DOES NOT CONTAIN ANY BUILDING $> A[n]$

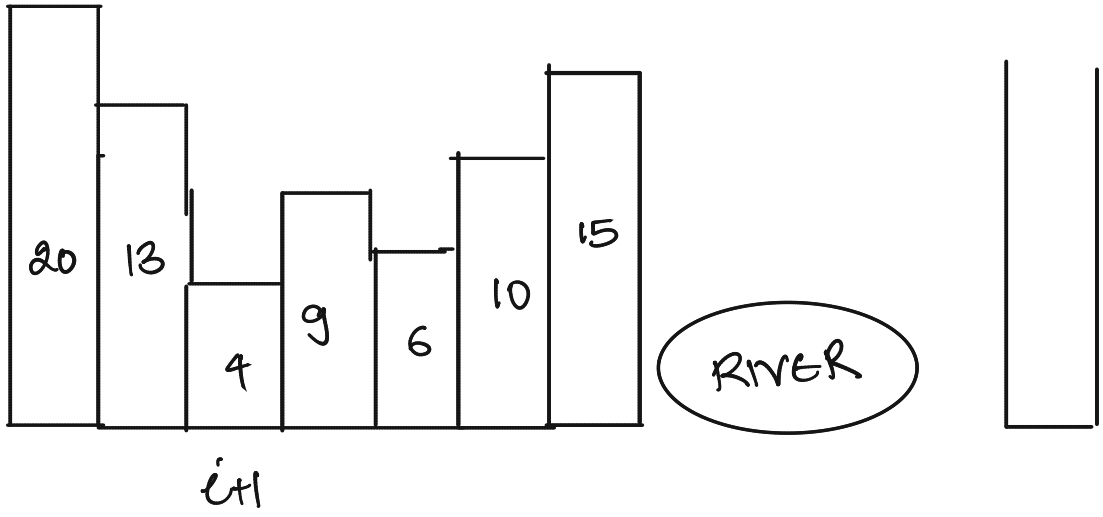
LEMMA: AT THE END OF ITERATION i, STACK S CONTAINS BUILDING j $(j > i)$ IF $A[i \ldots j]$ DOES NOT CONTAIN ANY BUILDING $> A[j]$.

PROOF: BY INDUCTION

$i = n$

BEFORE THE START OF FOR LOOP S CONTAINS BUILDING n AND $A[n \ldots n]$ DOES NOT CONTAIN ANY BUILDING $> A[n]$

USING INDUCTION HYPOTHESIS, ASSUME THAT THE STATEMENT IS TRUE AT THE END OF $i+1$ ITERATION



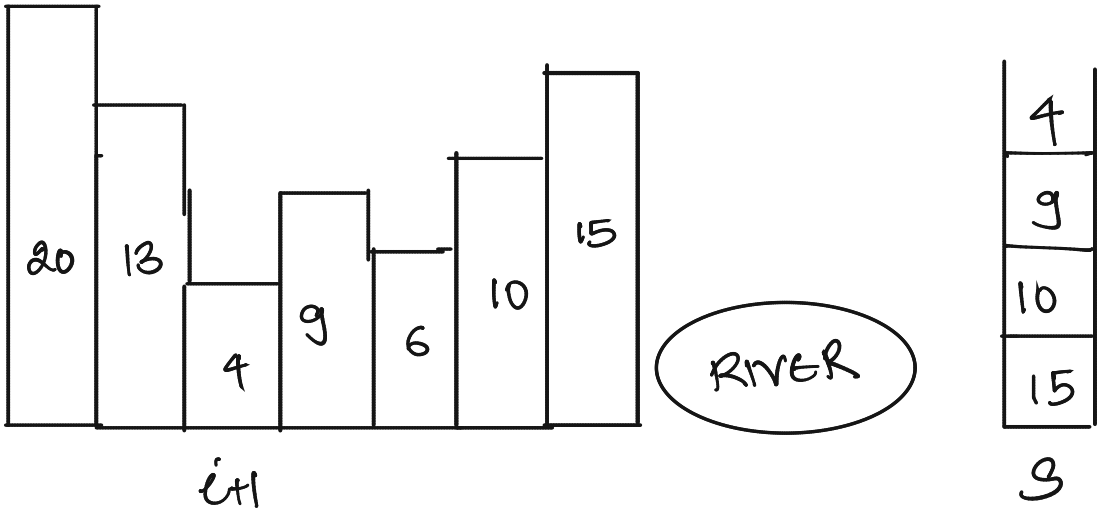
LEMMA: AT THE END OF ITERATION i, STACK S CONTAINS BUILDING j $(j > i)$ IF $A[i \ldots j]$ DOES NOT CONTAIN ANY BUILDING $> A[j]$.

PROOF: BY INDUCTION

$i = n$

BEFORE THE START OF FOR LOOP S CONTAINS BUILDING n AND $A[n \ldots n]$ DOES NOT CONTAIN ANY BUILDING $> A[n]$

USING INDUCTION HYPOTHESIS, ASSUME THAT THE STATEMENT IS TRUE AT THE END OF $i+1$ ITERATION



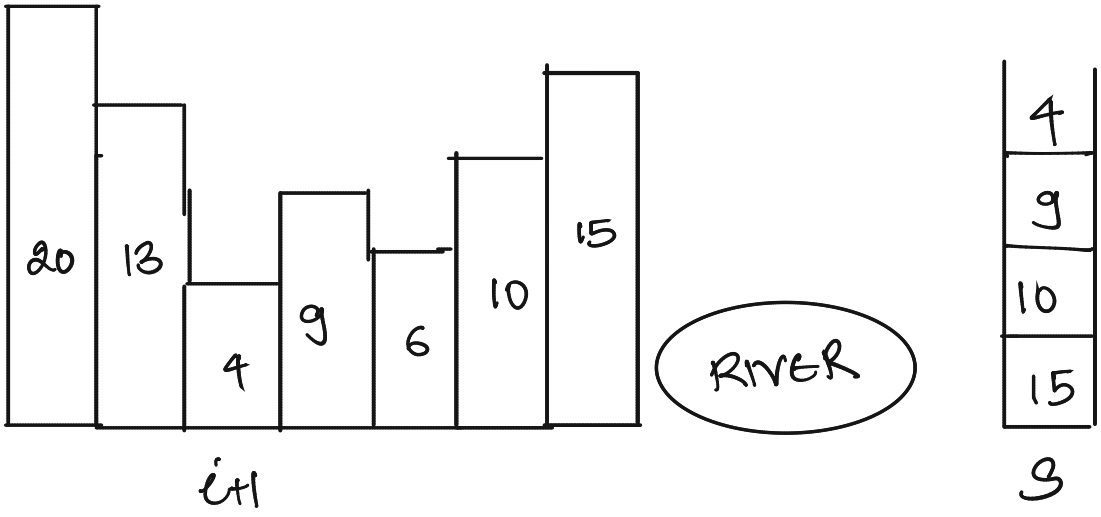
LEMMA: AT THE END OF ITERATION i, STACK S CONTAINS BUILDING j $(j > i)$ IF $A[i \ldots j]$ DOES NOT CONTAIN ANY BUILDING $> A[j]$.

PROOF: BY INDUCTION

$i = n$

BEFORE THE START OF FOR LOOP S CONTAINS BUILDING n AND $A[n..n]$ DOES NOT CONTAIN ANY BUILDING $> A[n]$

USING INDUCTION HYPOTHESIS, ASSUME THAT THE STATEMENT IS TRUE AT THE END OF ith ITERATION



Q: WHAT HAPPENS AT THE i^{th} ITERATION?

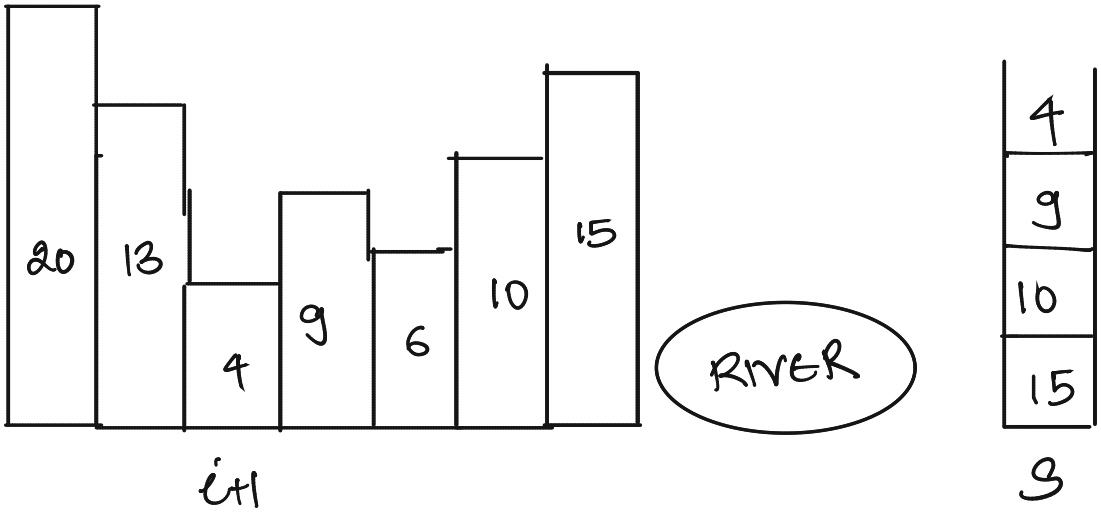
LEMMA: AT THE END OF ITERATION i, STACK S CONTAINS BUILDING j $(j > i)$ IF $A[i \ldots j]$ DOES NOT CONTAIN ANY BUILDING $> A[j]$.

PROOF: BY INDUCTION

$i = n$

BEFORE THE START OF FOR LOOP S CONTAINS BUILDING n AND $A[n..n]$ DOES NOT CONTAIN ANY BUILDING $> A[n]$

USING INDUCTION HYPOTHESIS, ASSUME THAT THE STATEMENT IS TRUE AT THE END OF ith ITERATION



Q: WHAT HAPPENS AT THE i^{th} ITERATION?

A: ALL BUILDINGS WITH HEIGHT $< A[i]$ ARE REMOVED FROM S.

LEMMA: AT THE END OF ITERATION i, STACK S CONTAINS BUILDING j $(j > i)$ IF $A[i \ldots j]$ DOES NOT CONTAIN ANY BUILDING $> A[j]$.

IF BUILDING i CAN SEE THE RIVER, THEN AT THE END OF i^{th} ITERATION S CONTAINS

LEMMA: AT THE END OF ITERATION i, STACK S CONTAINS BUILDING j $(j > i)$ IF $A[i \ldots j]$ DOES NOT CONTAIN ANY BUILDING $> A[j]$.

IF BUILDING i CAN SEE THE RIVER, THEN AT THE END OF i^{th} ITERATION S CONTAINS i

IF BUILDING j IS THE NEAREST BUILDING THAT BLOCKS THE VIEW OF BUILDING i THEN

LEMMA: AT THE END OF ITERATION i, STACK S CONTAINS BUILDING j $(j > i)$ IF $A[i \ldots j]$ DOES NOT CONTAIN ANY BUILDING $> A[j]$.

IF BUILDING i CAN SEE THE RIVER, THEN AT THE END OF i^{th} ITERATION S CONTAINS i

IF BUILDING j IS THE NEAREST BUILDING THAT BLOCKS THE VIEW OF BUILDING i THEN $A[i \ldots j]$ DOES NOT CONTAIN ANY BUILDING $> A[j]$

LEMMA: AT THE END OF ITERATION i, STACK S CONTAINS BUILDING j $(j > i)$ IF $A[i \ldots j]$ DOES NOT CONTAIN ANY BUILDING $> A[j]$.

IF BUILDING i CAN SEE THE RIVER, THEN AT THE END OF i^{th} ITERATION S CONTAINS i

IF BUILDING j IS THE NEAREST BUILDING THAT BLOCKS THE VIEW OF BUILDING i THEN $A[i \ldots j]$ DOES NOT CONTAIN ANY BUILDING $> A[j]$

$\Rightarrow$ $A[i+1 \ldots j]$ DOES NOT CONTAIN ANY BUILDING $> A[j]$

LEMMA: AT THE END OF ITERATION i, STACK S CONTAINS BUILDING j $(j > i)$ IF $A[i \ldots j]$ DOES NOT CONTAIN ANY BUILDING $> A[j]$.

IF BUILDING i CAN SEE THE RIVER, THEN AT THE END OF i^{th} ITERATION S CONTAINS i

IF BUILDING j IS THE NEAREST BUILDING THAT BLOCKS THE VIEW OF BUILDING i THEN $A[i \ldots j]$ DOES NOT CONTAIN ANY BUILDING $> A[j]$

$\Rightarrow$ $A[i+1 \ldots j]$ DOES NOT CONTAIN ANY BUILDING $> A[j]$

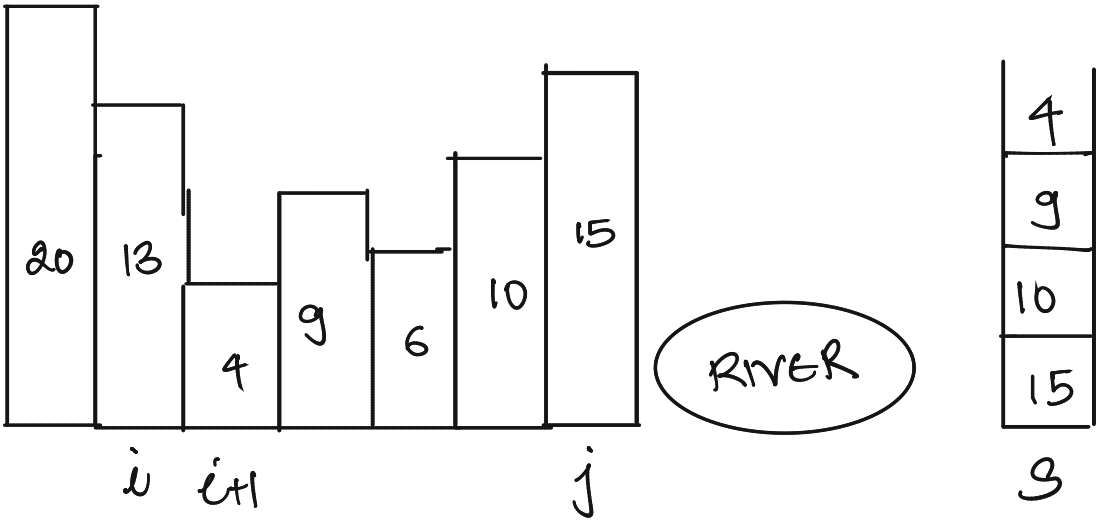
APPLY THE ABOVE LEMMA.

AT THE END OF ITERATION $i+1$, S CONTAINS BUILDING j.

LEMMA: AT THE END OF ITERATION i, STACK S CONTAINS BUILDING j $(j > i)$ IF $A[i \ldots j]$ DOES NOT CONTAIN ANY BUILDING $> A[j]$.

IF BUILDING i CAN SEE THE RIVER, THEN AT THE END OF i^{th} ITERATION S CONTAINS i

IF BUILDING j IS THE NEAREST BUILDING THAT BLOCKS THE VIEW OF BUILDING i THENAT THE END OF ITERATION $i+1$, S CONTAINS BUILDING j.

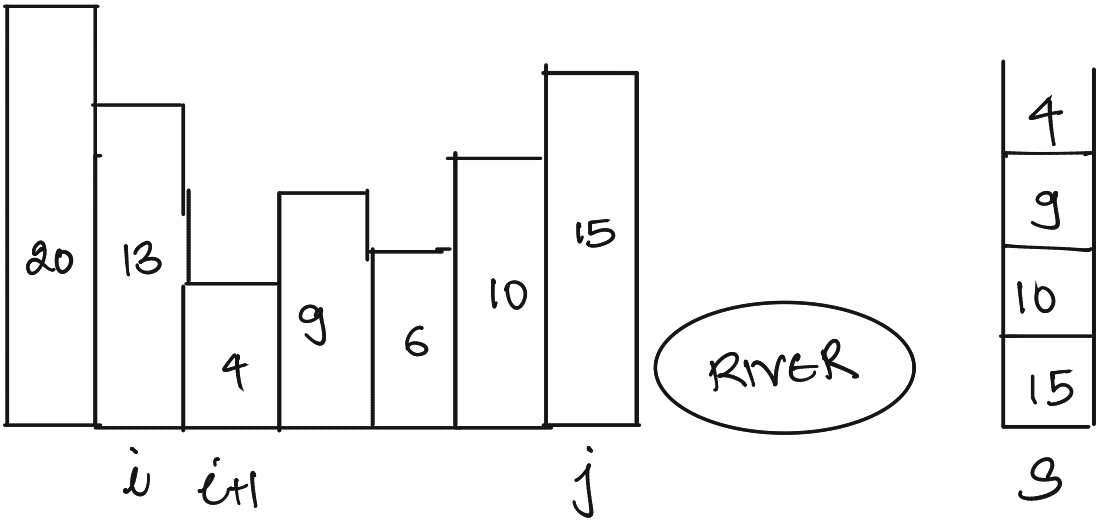


Q: WHAT HAPPENS AT ITERATION i?

LEMMA: AT THE END OF ITERATION i, STACK S CONTAINS BUILDING j $(j > i)$ IF $A[i \ldots j]$ DOES NOT CONTAIN ANY BUILDING $> A[j]$.

IF BUILDING i CAN SEE THE RIVER, THEN AT THE END OF i^{th} ITERATION S CONTAINS i

IF BUILDING j IS THE NEAREST BUILDING THAT BLOCKS THE VIEW OF BUILDING i THENAT THE END OF ITERATION $i+1$, S CONTAINS BUILDING j.



Q: WHAT HAPPENS AT ITERATION i?

A: OUR ALGO POPS OUT ALL INTERMEDIATE BUILDING AND FINDS j.