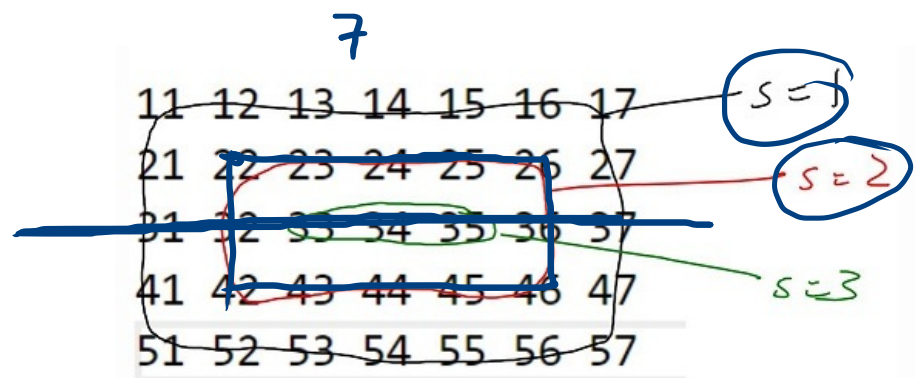


shell $\rightarrow 1$

shell $\rightarrow 2$



$s \rightarrow 2$
 $2 \rightarrow 1$

anti clock

11	12	13	14	15	16	17
21	23	24	25	26	36	27
31	22	33	34	35	46	37
41	32	42	43	44	45	47
51	52	53	54	55	56	57

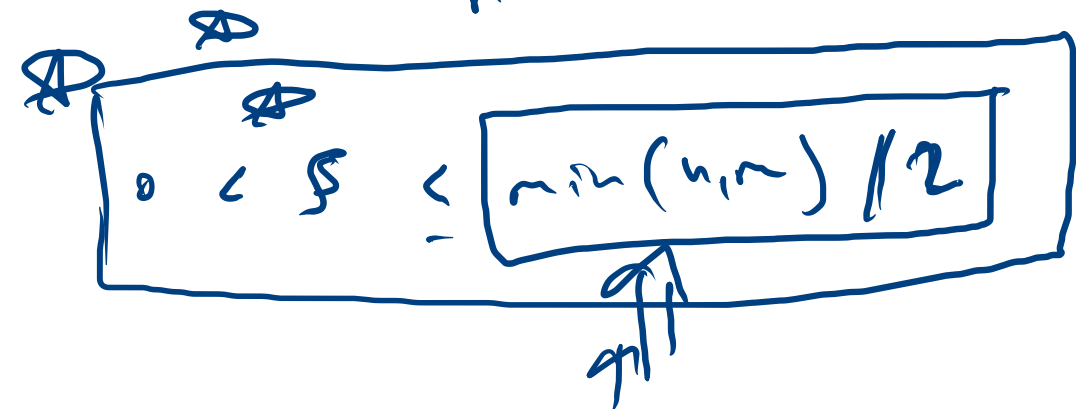
11	12	13	14	15	16	17
21	24	25	26	36	46	27
31	23	33	34	35	45	37
41	22	32	42	43	44	47
51	52	53	54	55	56	57

$r=2$

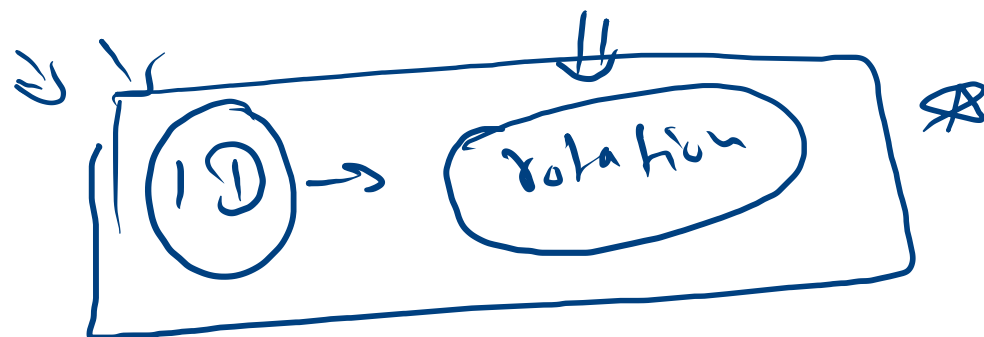
$r=2$

$+r$ anti-clock
 $-r$ clock

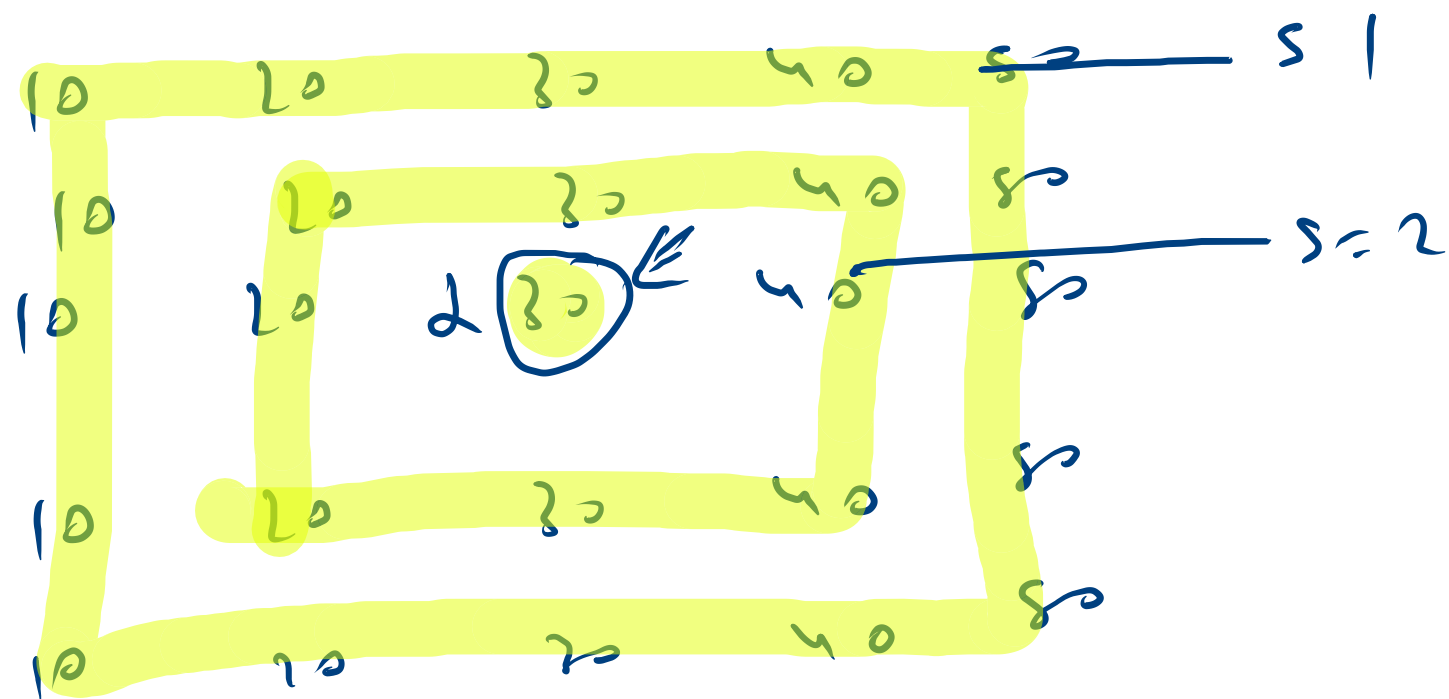
bits rotation



write
read



r/o
-


$$5/2 \rightarrow 2$$

even

even

even

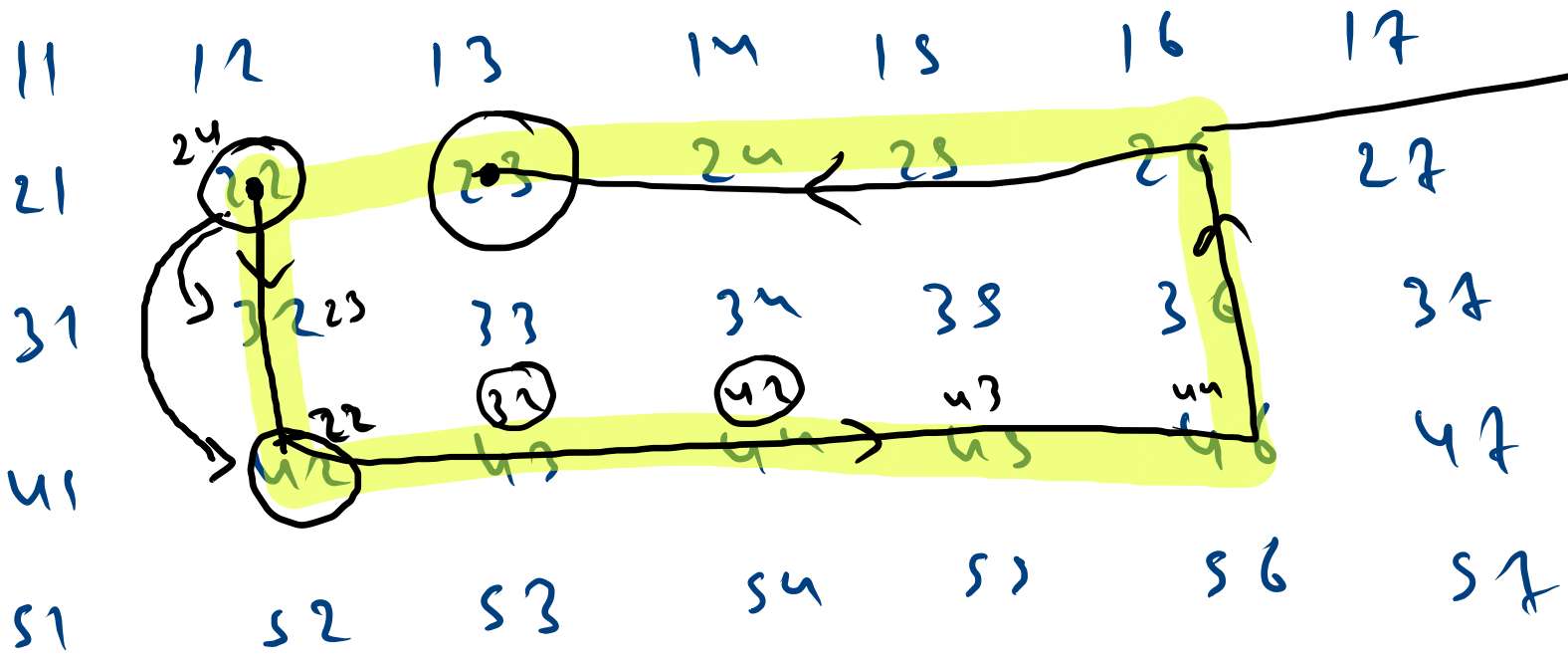
022 2

012

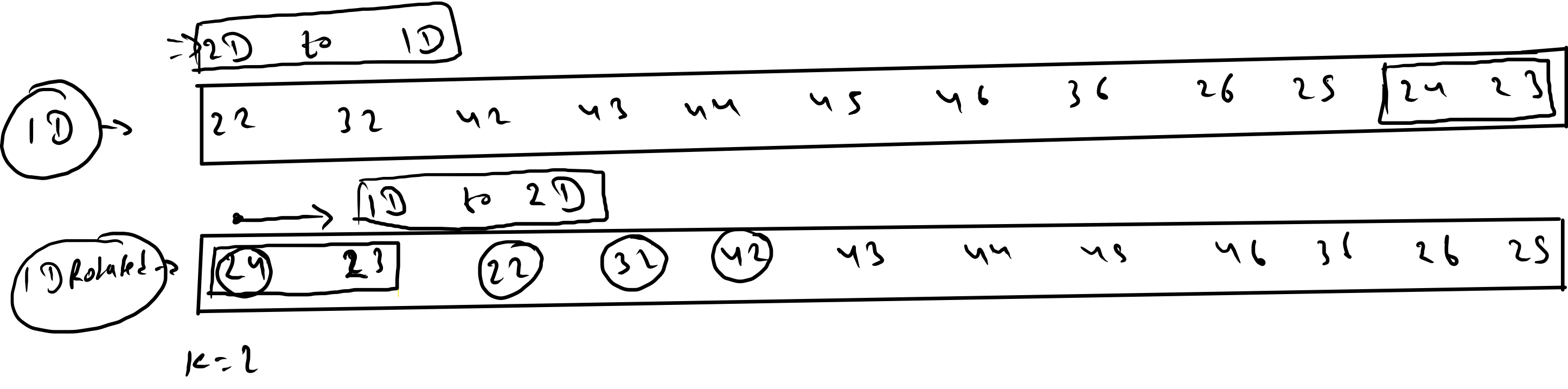
even \cup

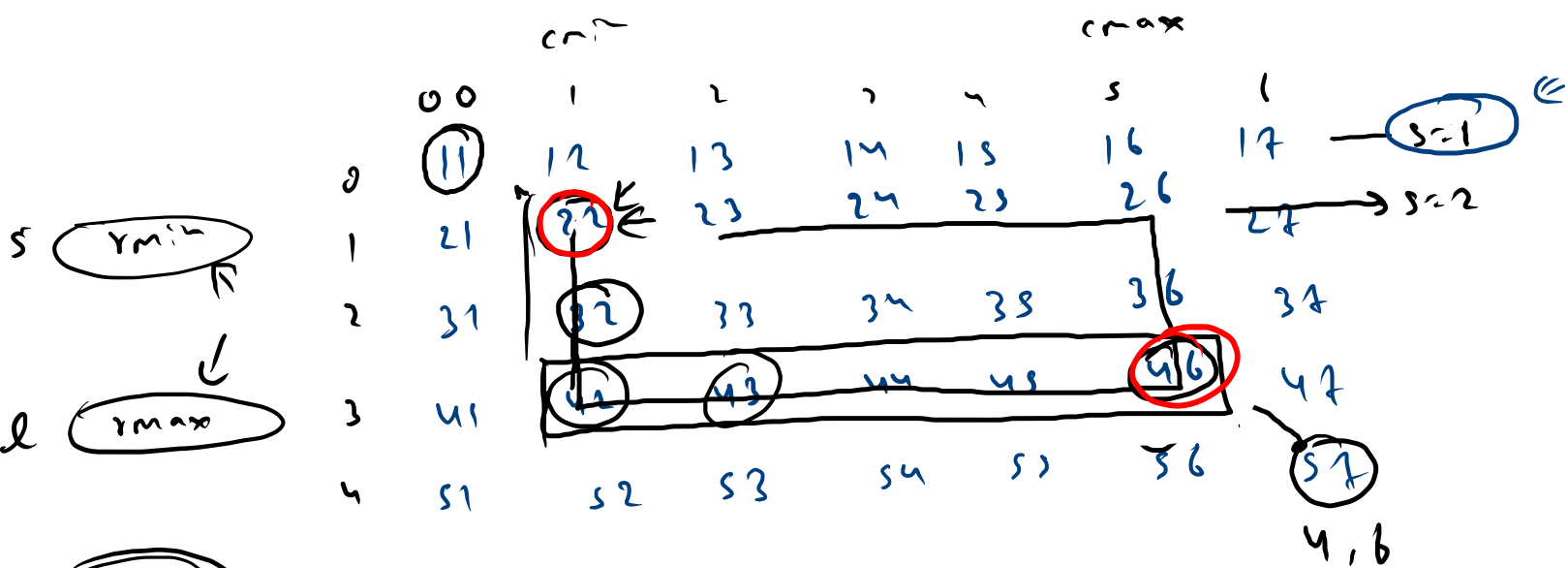
021

011 L



$s = 2$
 $r = ? \rightarrow 1 \text{ D}$
 $-\infty \leq r \leq \infty$
 $r = 1$
 Jen
 3 skip





shell

$r_{min} \rightarrow s-1$
$c_{min} \rightarrow s-1$
$r_{max} \rightarrow n-s$
$c_{max} \rightarrow m-s$

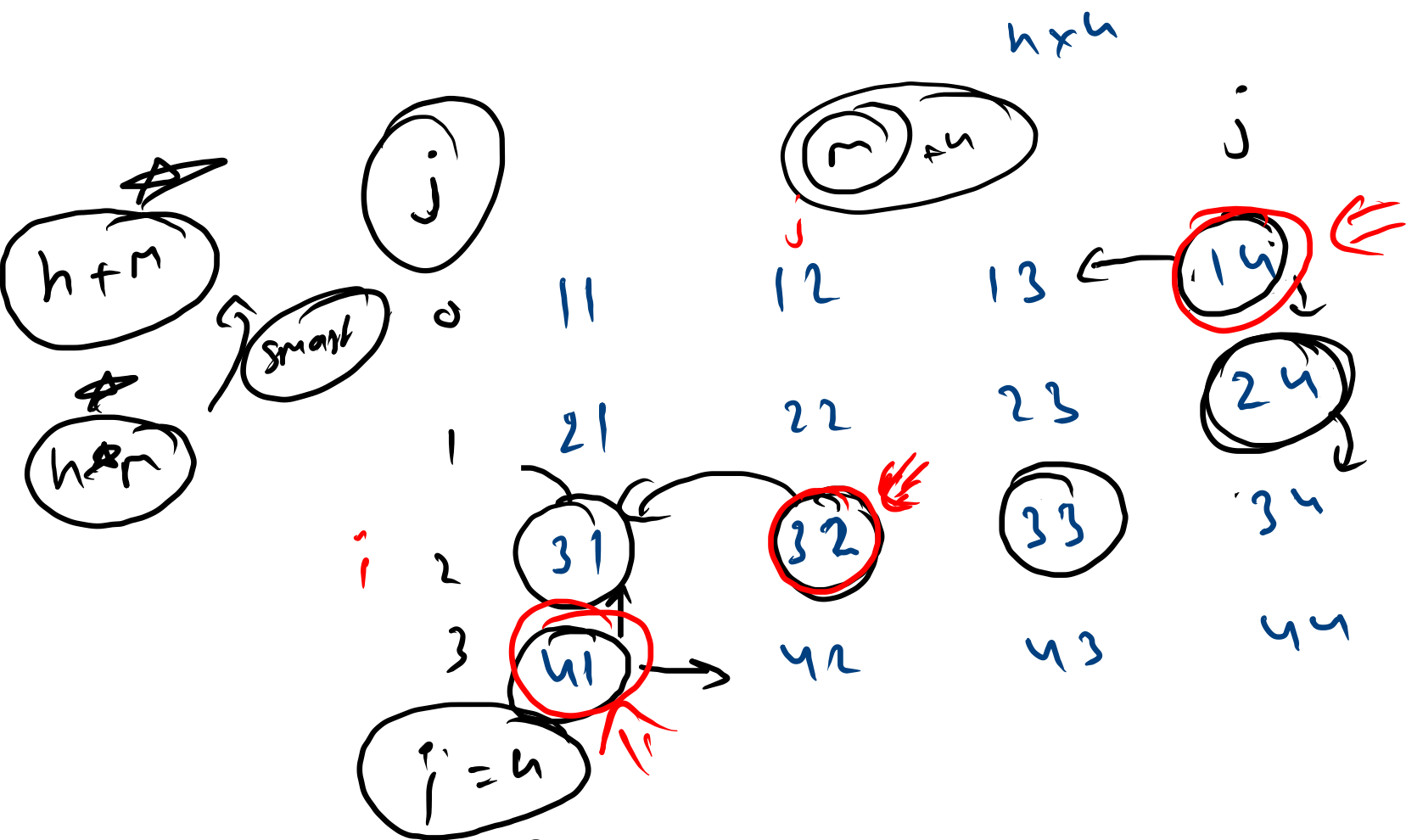
- 1D size $\rightarrow 2(r_{max} + c_{max} - r_{min} - c_{min})$
- 2D \rightarrow 1D spiral display
- 1D \rightarrow rotate
- 1D \rightarrow 2D spiral display

s r_{min}

l r_{max}

$l-s+1$
 $3-1+1 \Rightarrow 3$





$n < ar[i][j]$

$j--$

$n > ar[i][j]$

$i++$

$n ==$

2
1

30

55

2
1
2

$n = 45$ Not Fun 2

$n = 32 \rightarrow 2$
1

rule

had low 2

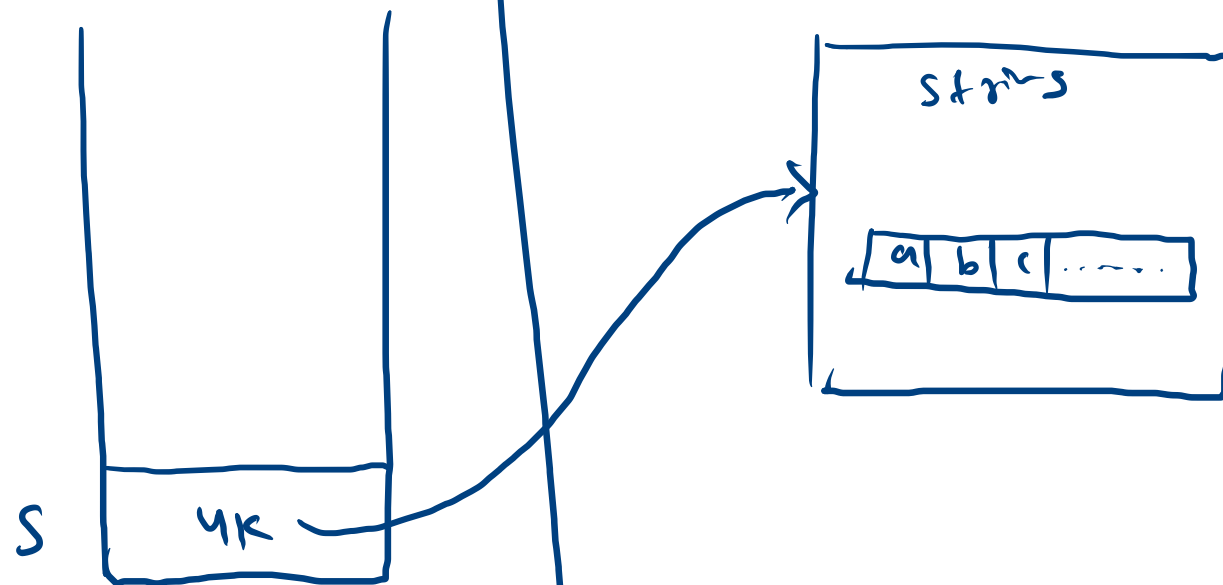
$j < 0, i = 4$

declaration
int a = 23;

String s = "abc12?#";
initialization

Array character

⇒ mutable



array a

a[i] = 10;

int n = a[i] ← read



charAt(i) and
charAt(i)

$s_1 \rightarrow "abc"$

$s_2 \rightarrow "def"$

`String s3 = s1 + s2 ; // abcdef`

Subarray

str \rightarrow "a b c d"
0 1 2 3

str.length()
str.substring(i, j)
(i)

0 1	a	1 2	b
0 2	ab	1 3	bc
0 3	abc	1 4	bcd
0 4	abcd		

2 3 \rightarrow c
2 4 \rightarrow cd
3 4 \rightarrow d

i j

i j

i+1 / length

i j
0 i+1
length

i+1
length

i+1
length

i
0
length-1

s = "abcd" ,

	0	1	2	3
s	a	b	c	d

s.substr(1) bc d
(0) abc d
(2) → cd

substr(i) \swarrow $j = \text{len}$
substr(i, j)

reverse

palindrome

str = reverse(str)

while(i <= j) {

if (s.charAt(i) != s.charAt(j))
not palindrome

i++
j--
}

str.charAt(i)

i	i	i	i	j	j	j	j
a	b	c	d	d	c	b	a

"a b c d d c b a"

"a b c d c b a"

"a" ← single character

+

a += 'c' ←

" " ← palindrome

```

public static boolean isPalindrome(String str){
    int i=0;
    int j = str.length()-1;

    while(i<j){
        if(str.charAt(i) != str.charAt(j)){
            return false;
        }
        i++;
        j--;
    }
    return true;
}

```

Handwritten annotations on the code:

- An arrow points to `i=0`.
- A circle is drawn around `i<j` in the `while` loop condition, with a downward arrow pointing to it.
- An arrow points to `str.charAt(j)` in the `if` statement.
- A bracket `]α` is drawn next to the `if` statement.
- Two arrows point to `i++` and `j--` respectively.

str → " a b c d e d d e d c b a "

Indices: 0 1 2 3 4 5 6 7 8 9 10 11 12

Handwritten annotations on the string:

- The string is enclosed in quotes.
- Indices 0 through 12 are written above the characters.
- The characters at indices 5 and 6 (both 'd') are circled, with an arrow pointing to the first 'd' at index 5.
- The characters at indices 7 and 8 (both 'e') are circled, with an arrow pointing to the first 'e' at index 7.

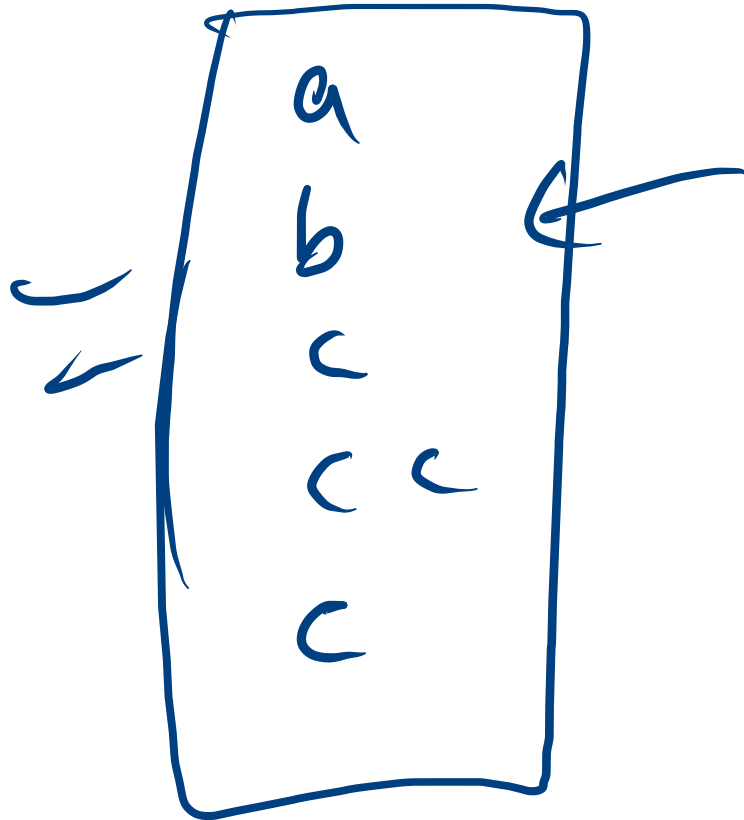
abcc

a ✓
a b ✓
a b c ✓
a b c c ✓

b ✓
b c ✓
b c c ✓

c ✓
c c ✓
c c

c ✓



aaabbbccdee a a a e

a b c d e f

comp1 → a b c d e a

a b c d e f

comp 2 →

a 3 b 2 c 2 d e 2 a 4

a b c d e f