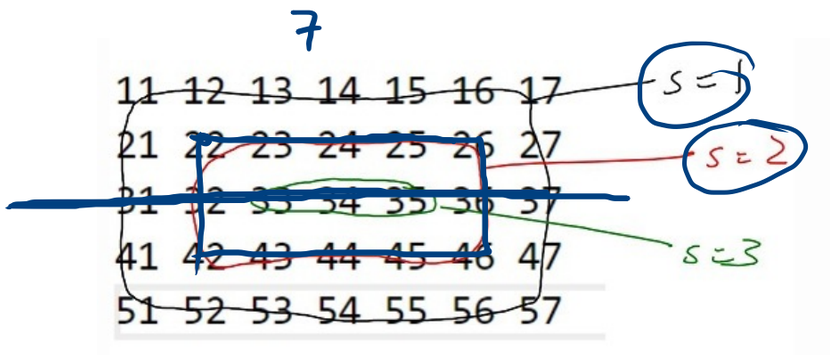
Shell -> 1

shell -> 2



s → 2

2 r → 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

r=22

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	anti-clo
-r	clock

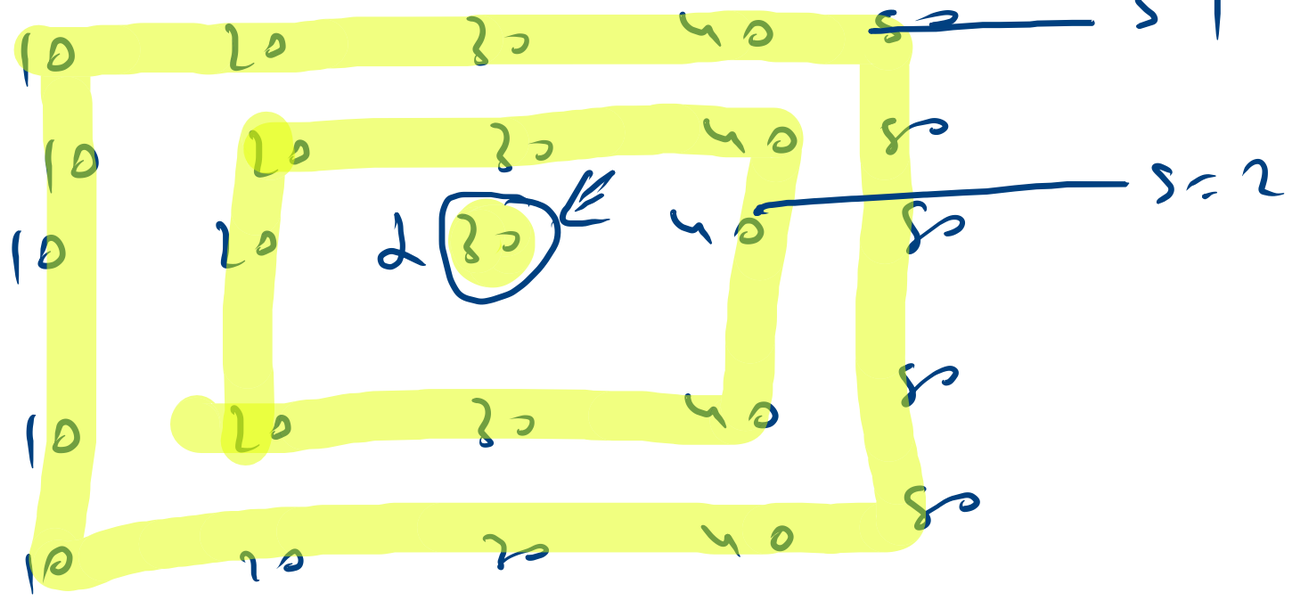
min row

$0 < s \leq \min(m,n)/2$

write read

r/o

5



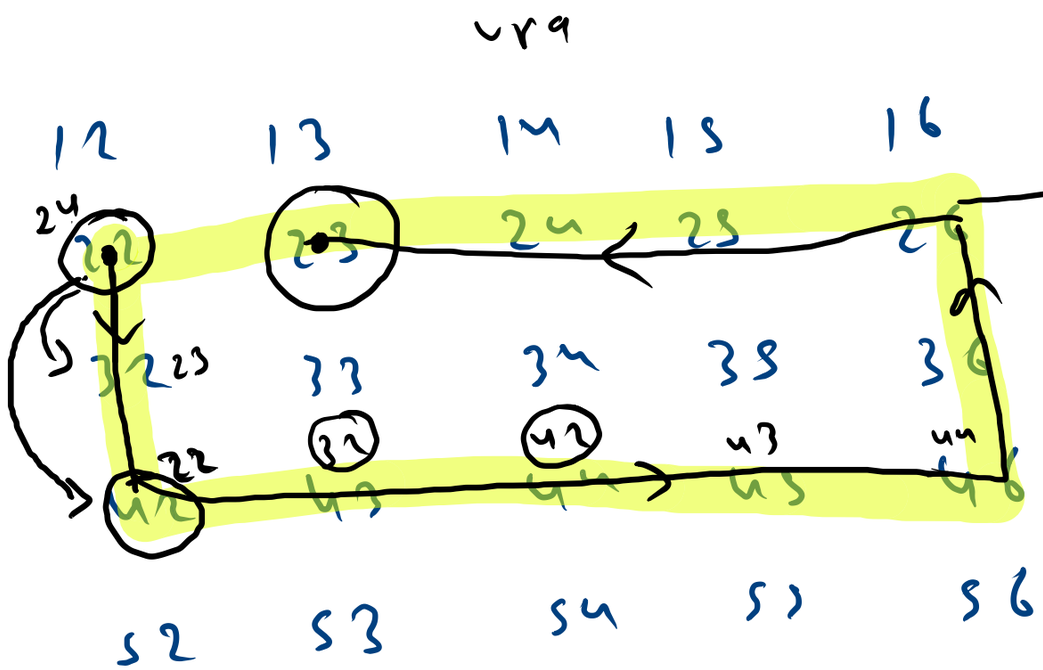
s = 1

s = 2

5

even	even	
even	odd	
odd	even	
odd	odd	

5/2 → 2



s = 2

r = ? → 1 D

$-\infty \leq r \leq \infty$

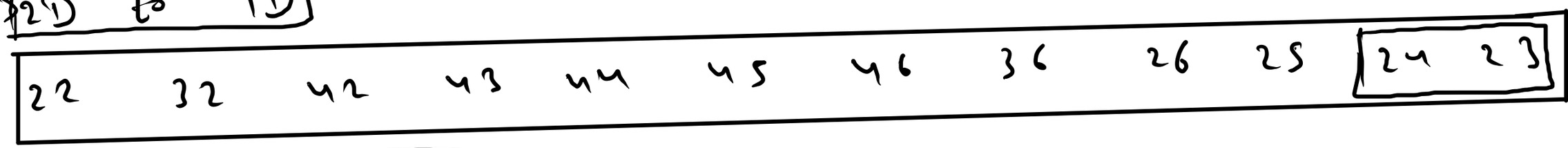
r = 2

3 skip

dan

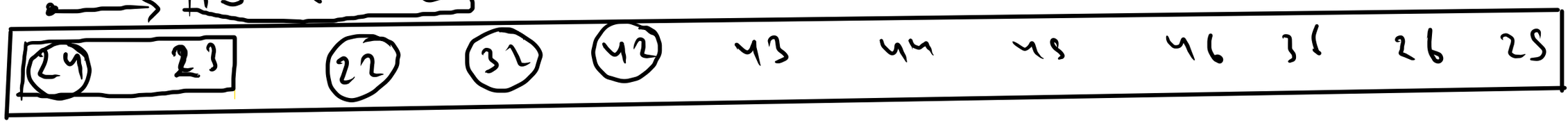
=> 2D to 1D

1D →

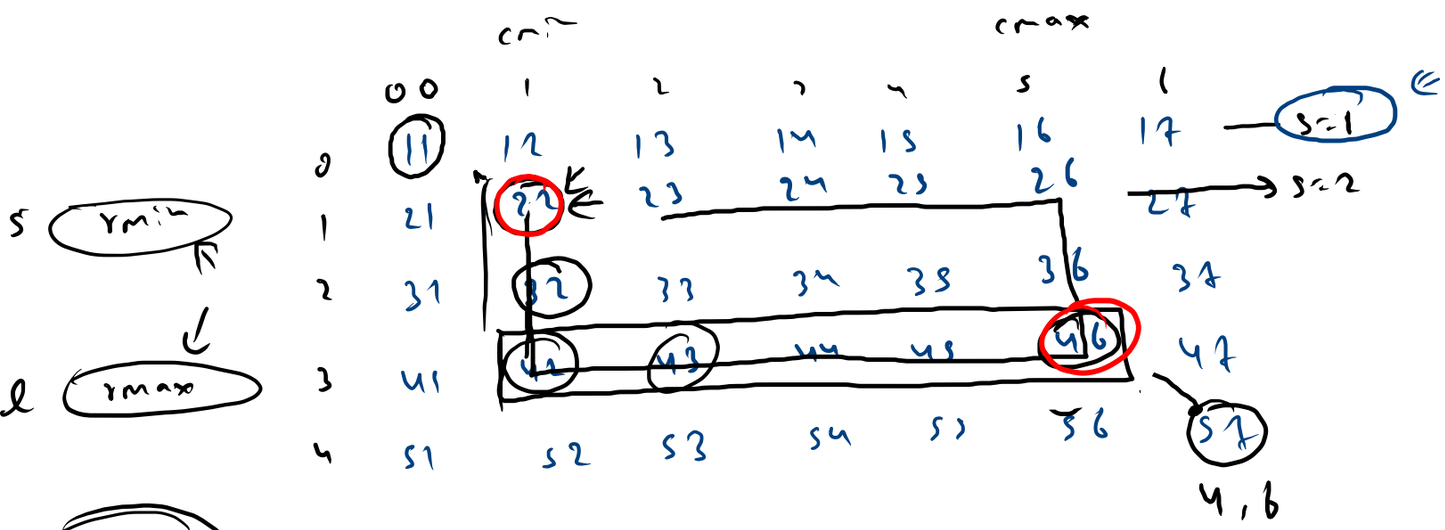


1D to 2D

1D Rotate →



k = 2



shell

rmin -> s-1
cmin -> s-1
rmax -> n-s
cmax -> m-s

1D size -> 2(rmax + cmax - rmin - cmin)

2D -> 1D spiral display

1D -> rotate

1D -> 2D spiral display

n x n

h+m

h*r

smart

m * n

j

0	11	12	13	14
1	21	22	23	24
2	31	32	33	34
3	41	42	43	44

i

j = n

x = 45 Not found

x = 32 → 2

1

rule

x < ar[i][j]

j--

x > ar[i][j]

i++

x ==

2

1

2

3

30

55

not found

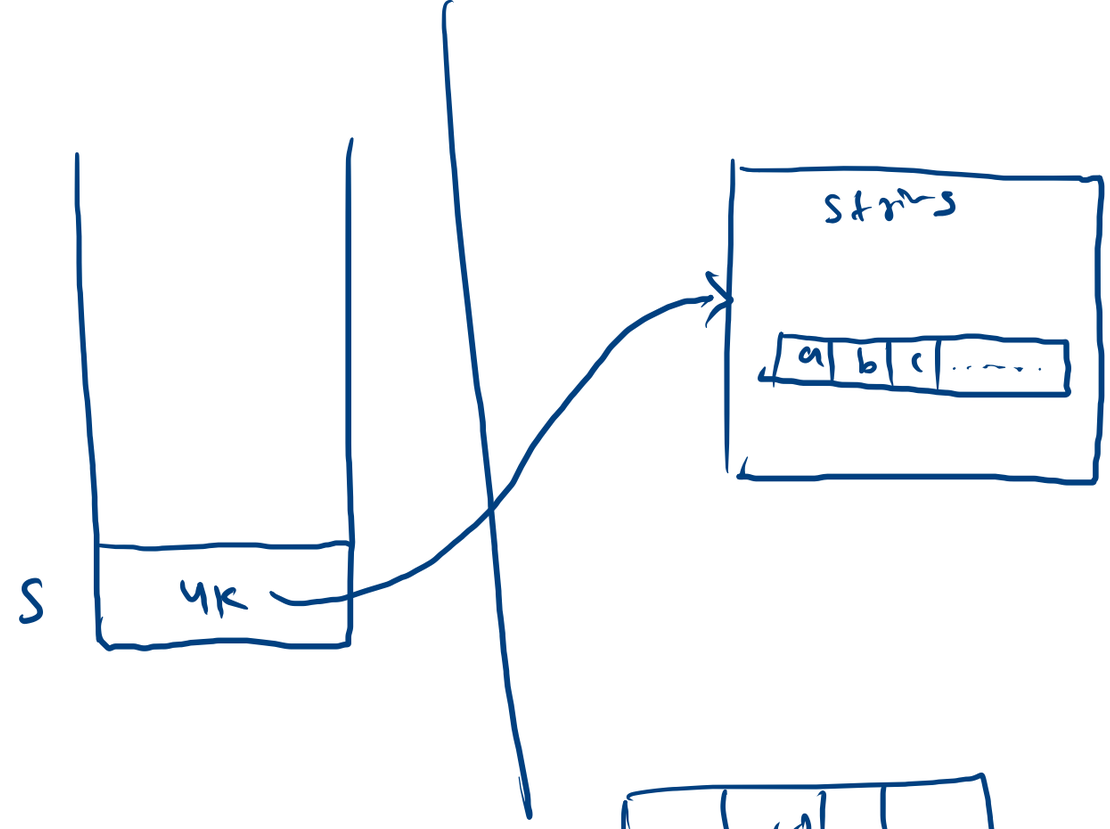
j < 0 , i == n

declaration

```
int a = 23;
```

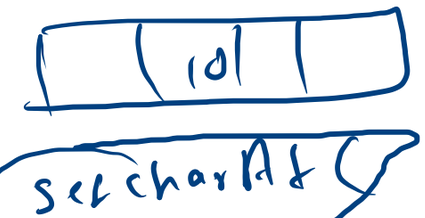
```
String s = "abc12?#";
```

initialization



Array character

=> Immutable



array a

a[i]: 10;

int n: a[i] <- read

setCharAt( )

charAt(i)

s1 -> "abc"

s2 -> "def"

String s3 = s1+s2 ; //abcdef

Subarrays

str → "abcd"

0 1 2 3

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

0 1	a	1 2	b	2 3 → c	3 4 → d
0 2	ab	1 3	bc	2 4 → cd	
0 3	abc	1 4	bcd		
0 4	abcd				

i j

i j

i j

i+1 / length

i

0

j-1

i+1

i+1

length

length

length

0 1 2 3

s = " a b c d "

s.substring(1) bcd

(0) abcd

(2) → cd

substring(i) ← j = len

substring(i, j)

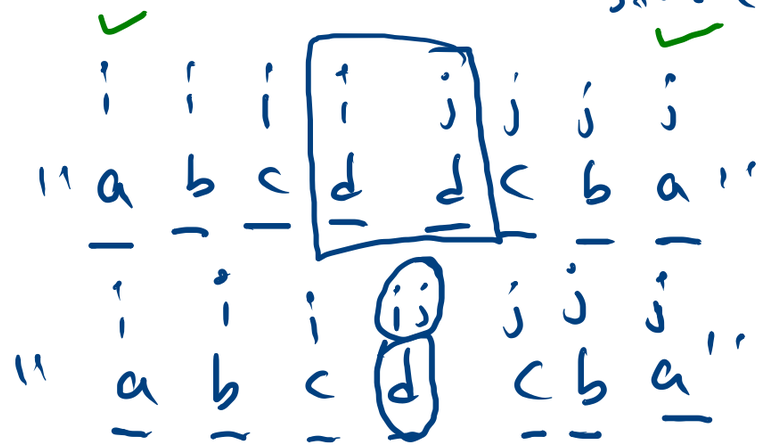
reverse

palindrome

str = reverse(str)

while(i ≤ j) {

str.charAt(i)



"a" → single character

+

a += 'c'

if(s.charAt(i) != s.charAt(j)) "" → palindrome

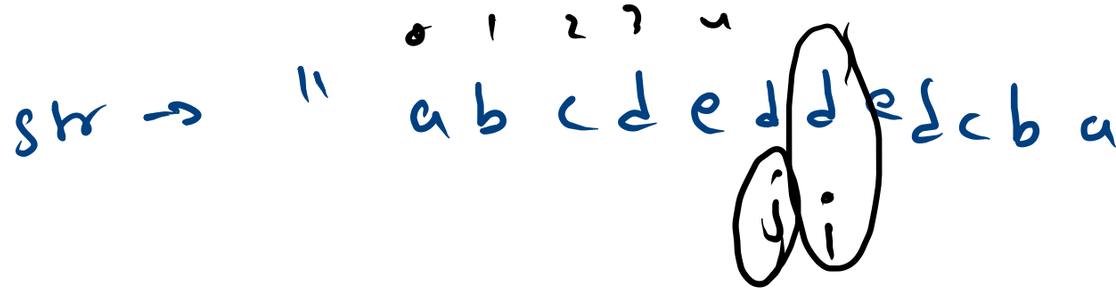
not palindrome

i++ j--

}

```java
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;
}
```



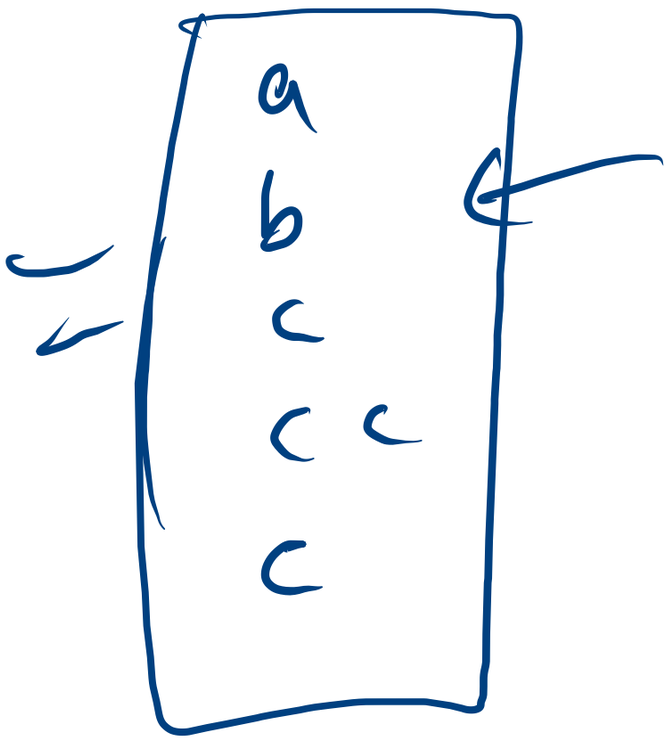
abcc

a ✓
ab✗
abc✗
abcc✗

b ✓
bc✗
bcc✗

c ✓
cc ✓

c ✓



aaabbccdee aaaa

abcdef

compress → abcdea

abcdef

comp 2 → a3b2c2de2a4

abcdef