به نام خدا

توابع بازگشتی

یادآوری

- **ایده‌ی کلی:** برای حل یک مسئله، سعی کنیم آن را به مسئله‌های مشابه ولی به اندازه‌ی کوچک‌تر بشکنیم.

- **دست‌گرمی:**

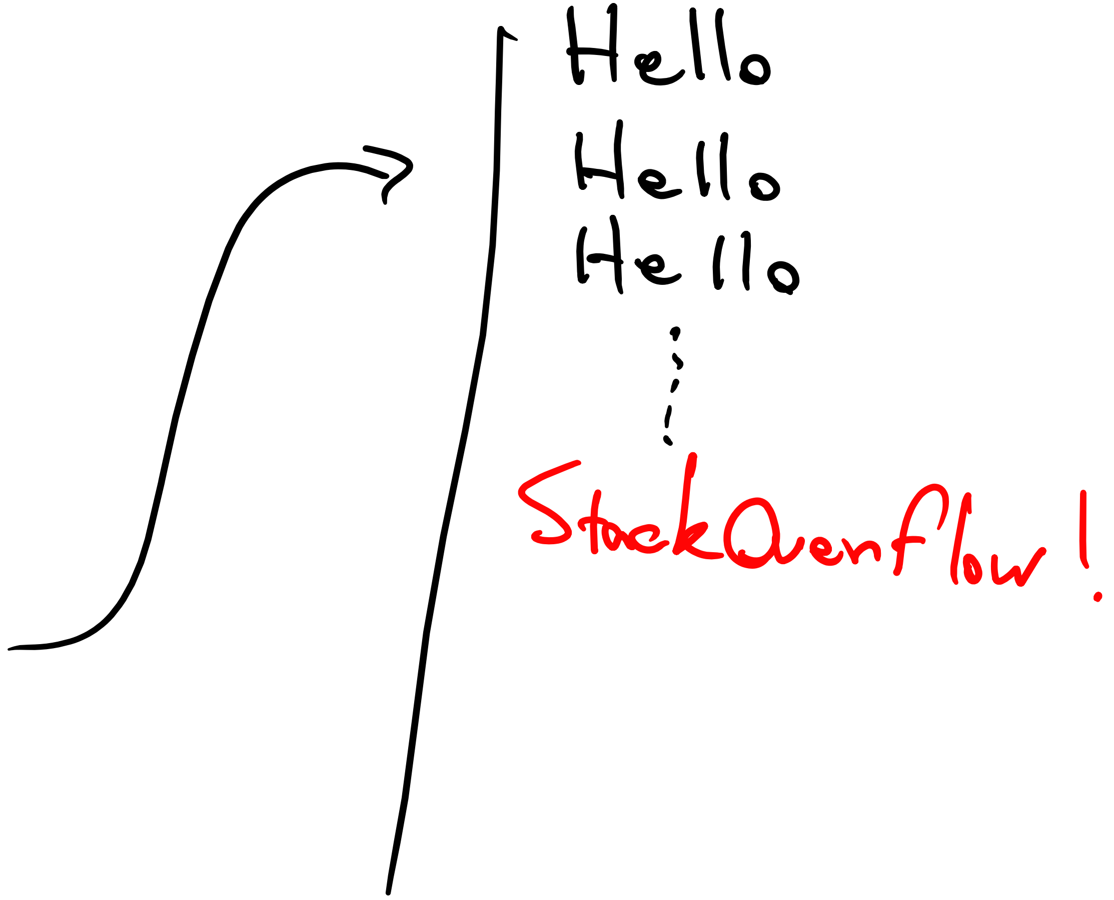
  برای n بار سلام کردن، اگر یک بار سلام کنیم، در ادامه مسئله‌ی کوچک‌ترِ n-1 بار سلام کردن را داریم!

```java
public static void sayHellos(int n) {
    // Say Hello:
    System.out.println("Hello");

    sayHellos(n-1);
}
```

یک بار سلام کن

مسئله‌ی کوچکتر رو حل کن: n-1 سلام → sayHellos(n-1);



```
public static void sayHellos(int n) {
        if (n == 0) {
                // do nothing!
        } else {
                // Say Hello:
                System.out.println("Hello");
                sayHellos(n-1);
        }
}
```

حالت پایه

اجزای یک تابع یا الگوریتم بازگشتی

- تبدیل مسئله، به مسئله‌های کوچک‌تر
- حالت یا حالت‌های پایه‌ای برای توقف الگوریتم
- حل مسئله‌ی اصلی براساس جواب مسئله‌های کوچک‌تر

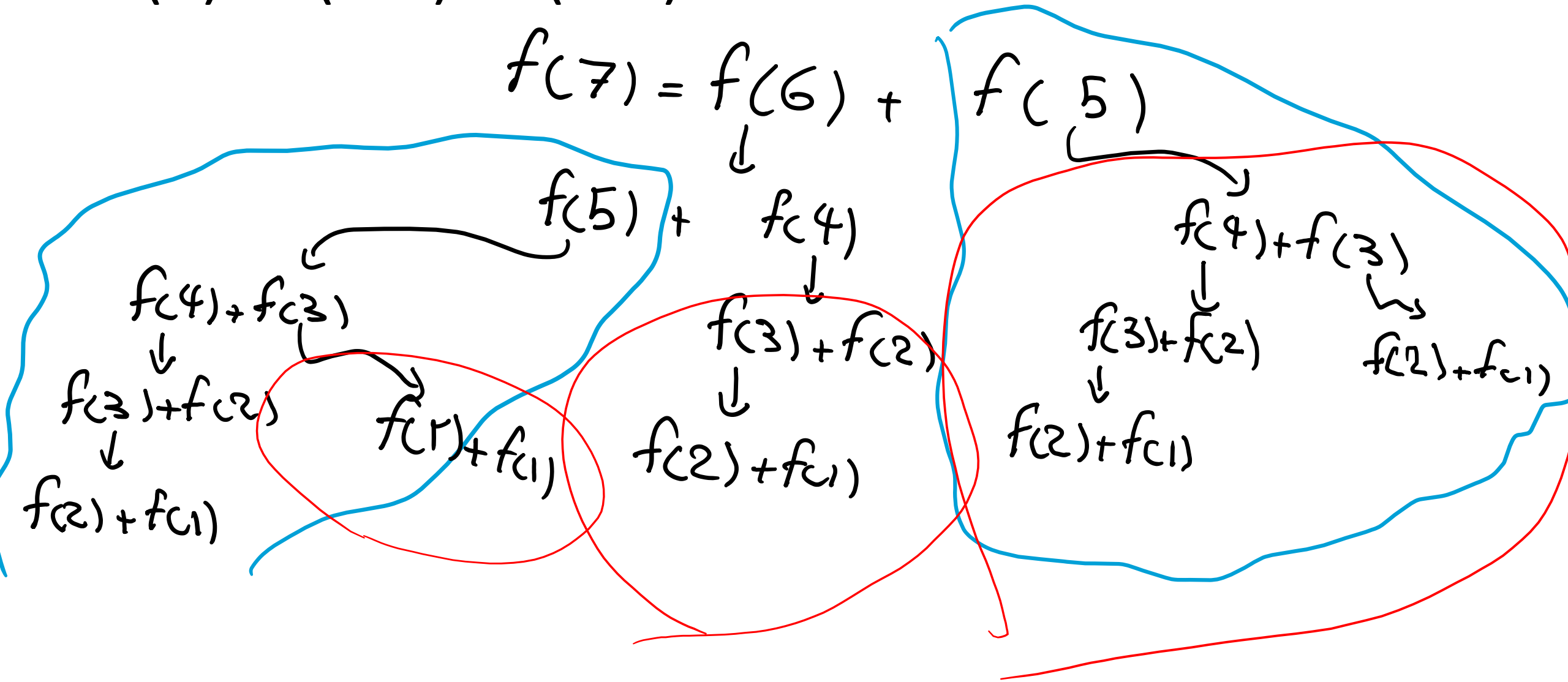
مثال: تعداد روش‌های بالا رفتن از پله

- یک فرد می‌خواهد n قدم از پله بالا رود. در هر قدم **یک** یا **دو** پله بالا می‌رود. چند حالت وجود دارد که فرد دقیقا n پله بالا رود؟
- **f(n):** تعداد حالت‌هایی که فرد با قدم‌های یک یا دو پله‌ای می‌تواند n پله بالا رود.

- **f(1) = 1, f(2) = 2, f(n) = f(n-1) + f(n-2)**

```
public int f(int n) {
        if (n == 1) {
                return 1;
        } else if (n == 2) {
                return 2;
        } else {
                return f(n-1) + f(n-2);
        }
}
```

$$f(n) = f(n-1) + f(n-2)$$


f(7) = f(6) + f(5)
f(6) = f(5) + f(4)
f(5) = f(4) + f(3)
f(4) = f(3) + f(2)
f(3) = f(2) + f(1)
f(4) = f(3) + f(2)
f(3) = f(2) + f(1)
f(3) = f(2) + f(1)
f(2) = f(1) + f(0)

مسئله‌ی اجرای چندباره‌ی کد

- در تابع f اسلاید قبل، اگر n را زیاد کنیم، زمان اجرای برنامه خیلی زیاد می‌شود!

- راه حل؟

راه ۱: باز نویسی کد بدون تابع بازگشتی!

```
int n = ...;
int[] f = new int[n+1];
f[1] = 1;
f[2] = 2;
for (int i = 3; i <= n; i++) {
        f[i] = f[i-1] + f[i-2];
}
```

```
Int[] history = new int[10000];
public int f(int n) {
        if (n < history.length && history[n] != 0) {
                return history[n];
        }
        if (n == 1) {
                return 1;
        } else if (n == 2) {
                return 2;
        } else {
                int n = return f(n-1) + f(n-2);
        }
}
                history[n] = n;
                return n;
```

این جلسه!

مثال‌های بیشتر

رشته‌های به طول n از حروف ۰ و ۱

- **ورودی:** عدد طبیعی n
- **خروجی:** تمام رشته‌ها به طول n که از حروف 0 و 1 تشکیل شده باشند.
- مثال: n=3
- 000, 001, 010, 011, 100, 101, 110, 111

ایده:

- void print10s(int n, String s)
- تابع قبل، تمام رشته‌ها از 0 و 1 به طول s.length()+n را چاپ کند طوری که پیشوند همه‌ی آن‌ها رشته‌ی s باشد.

ایده:

- print01s(n, s)، تمام رشته‌ها از 0 و 1 به طول s.length()+n را چاپ کند طوری که پیشوند همه‌ی آن‌ها رشته‌ی s باشد.

- print01s(2, “001”):

00100, **001**01, **001**10, **001**11

رشته‌های به طول n از حروف ۰ و ۱

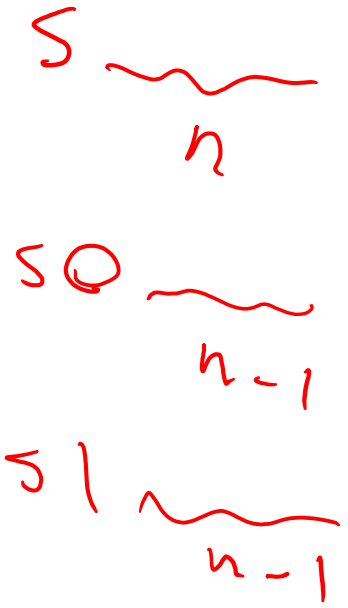
- **ورودی:** عدد طبیعی n
- **خروجی:** تمام رشته‌ها به طول n که از حروف 0 و 1 تشکیل شده باشند.

- **سوال ۱:** مسئله‌ی بالا چطور با print01s حل شود؟

رشته‌های به طول n از حروف ۰ و ۱

- **سوال ۲:** چطور print01s را حل کنیم؟

```
n, s
if (n == 0) {
    System.out.println (s);
}
```



افزونه‌ی ۱:

رشته‌های به طول n از حروف ۰ و ۱ و ۲

افزونه‌ی ۲:

رشته‌های به طول n از حروف ۰ و ۱ با حداکثر k تا ۱

رشته‌های به طول n از حروف ۰ و ۱ با حداکثر k تا یک

- **ورودی:** عدد طبیعی n و k
- **خروجی:** تمام رشته‌ها به طول n که از حروف 0 و 1 تشکیل شده باشند و حداکثر k کاراکتر آن‌ها 1 باشد.
- مثال: n=3, k = 1
- 000, 001, 010, 100

افزونه‌ی ۳:

رشته‌های به طول n از حروف ۰ و ۱ با دقیقا k تا ۱؟

در خانه فکر کنید و حل کنید ;)

مرتب‌سازی آرایه!

- **ورودی:** آرایه‌ی n عضوی از اعداد
- **هدف:** اعضای این آرایه را به شکل صعودی مرتب کنیم.

- مثال:
- ورودی: 7, 3, 4, 9, 2
- خروجی: 2, 3, 4, 7, 9

مرتب‌سازی آرایه!

- **ورودی:** آرایه‌ی n عضوی از اعداد
- **هدف:** اعضای این آرایه را به شکل صعودی مرتب کنیم.

void sortInPlace(int[] array, int l, int r)

در آرایه، عناصر array[l], array[l+1] تا ... array[r-1] را مرتب کند.

مرتب‌سازی آرایه!

- **ورودی:** آرایه‌ی n عضوی از اعداد
- **هدف:** اعضای این آرایه را به شکل صعودی مرتب کنیم.

void sortInPlace(int[] array, int l, int r)

در آرایه، عناصر array[l], array[l+1] تا ... array[r-1] را مرتب کند.

اگر این تابع را داشته باشیم، جواب مسئله؟

مرتب‌سازی آرایه!

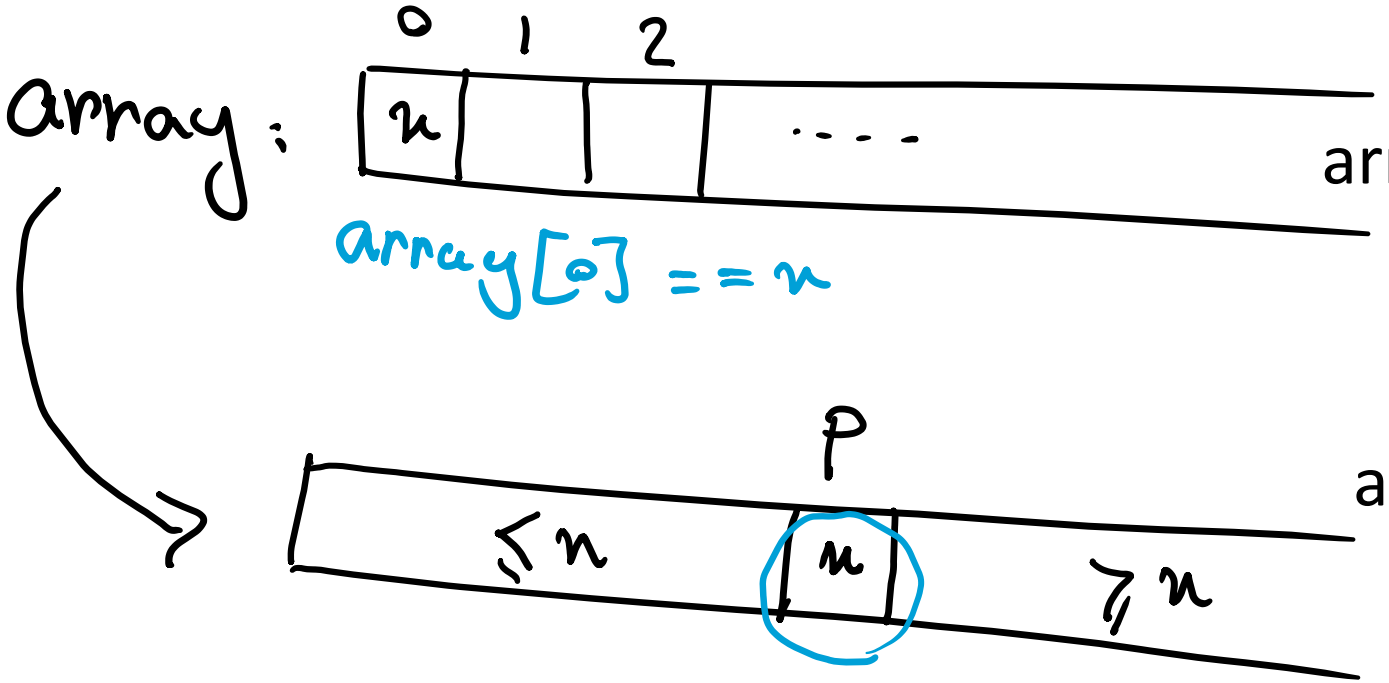
void sortInPlace(int[] array, int l, int r)

در آرایه، عناصر array[l], array[l+1] تا ... array[r-1] را مرتب کند.

قدم ۱: چطوری array[0] را در جایگاه مناسب خودش قرار بدیم؟

مرتب‌سازی آرایه!

قدم ۱: چطوری array[0] را در جایگاه مناسب خودش قرار بدیم؟



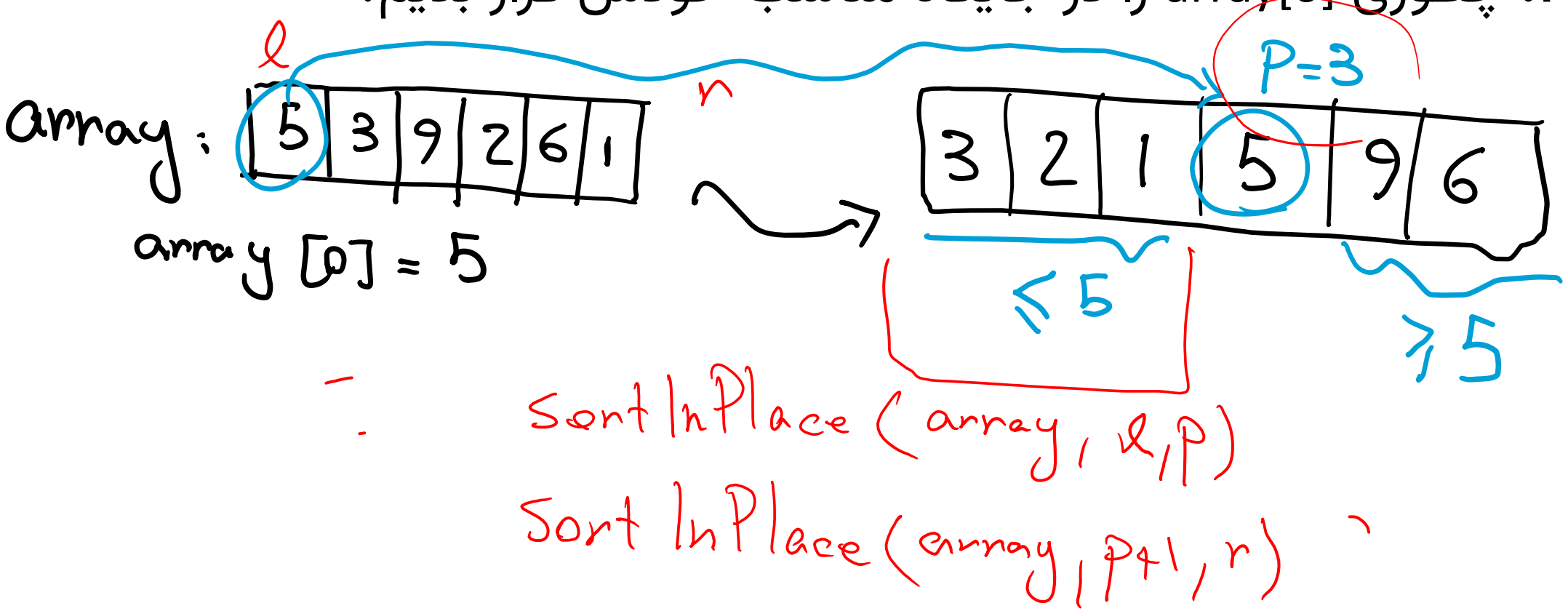
array[0] بیاد توی جایگاه p

هر چی قبل از p هست از array[0] کوچکتر یا مساوی باشه.

هرچی بعد از p هست از array[0] بزرگتر باشه.

مرتب‌سازی آرایه!

قدم ۱: چطوری array[0] را در جایگاه مناسب خودش قرار بدیم؟



مرتب‌سازی آرایه!

قدم ۱: array[0] را در جایگاه مناسب خودش قرار دادیم.

قدم ۲: حالا چطور بازگشتی مسئله رو حل کنیم؟

پیاده‌سازی!

کوچیتر هستند array[l] از عدد ها

p = l

for(int i = l+1 ; i < r ; i++) {

}

خصومت!

- اگر کسی که ورودی را می‌دهد با ما دشمنی نداشته باشد و ورودی به ترتیب شانسی بدهد، الگوریتم خیلی سریع کار می‌کند!

«مرتب‌سازی سریع»

- به الگوریتمی که پیاده‌سازی کردیم Quick Sort می‌گویند!

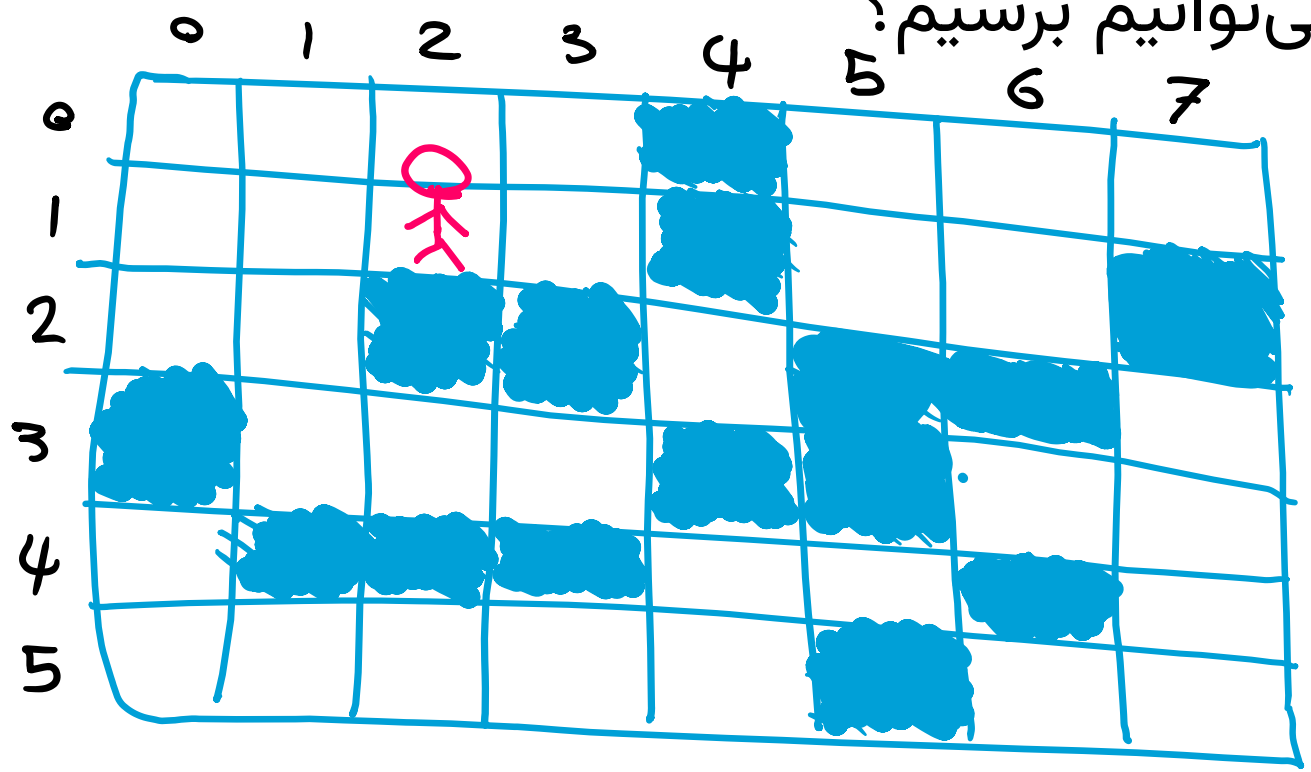
حل ماز!

- یک جدول n در m ورودی می‌گیریم، هر خانه از جدول یکی از دو حالت زیر را دارد:
- یا خالی است
- یا بسته شده و قابل دسترس نیست

- **سوال؟** با شروع از خانه‌ی (x, y) و حرکت‌های بالا، پایین، چپ یا راست، به کدام خانه‌های جدول می‌توانیم برسیم؟

حل ماز!

- **سوال؟** با شروع از خانه‌ی (x, y) و حرکت‌های بالا، پایین، چپ یا راست، به کدام خانه‌های جدول می‌توانیم برسیم؟



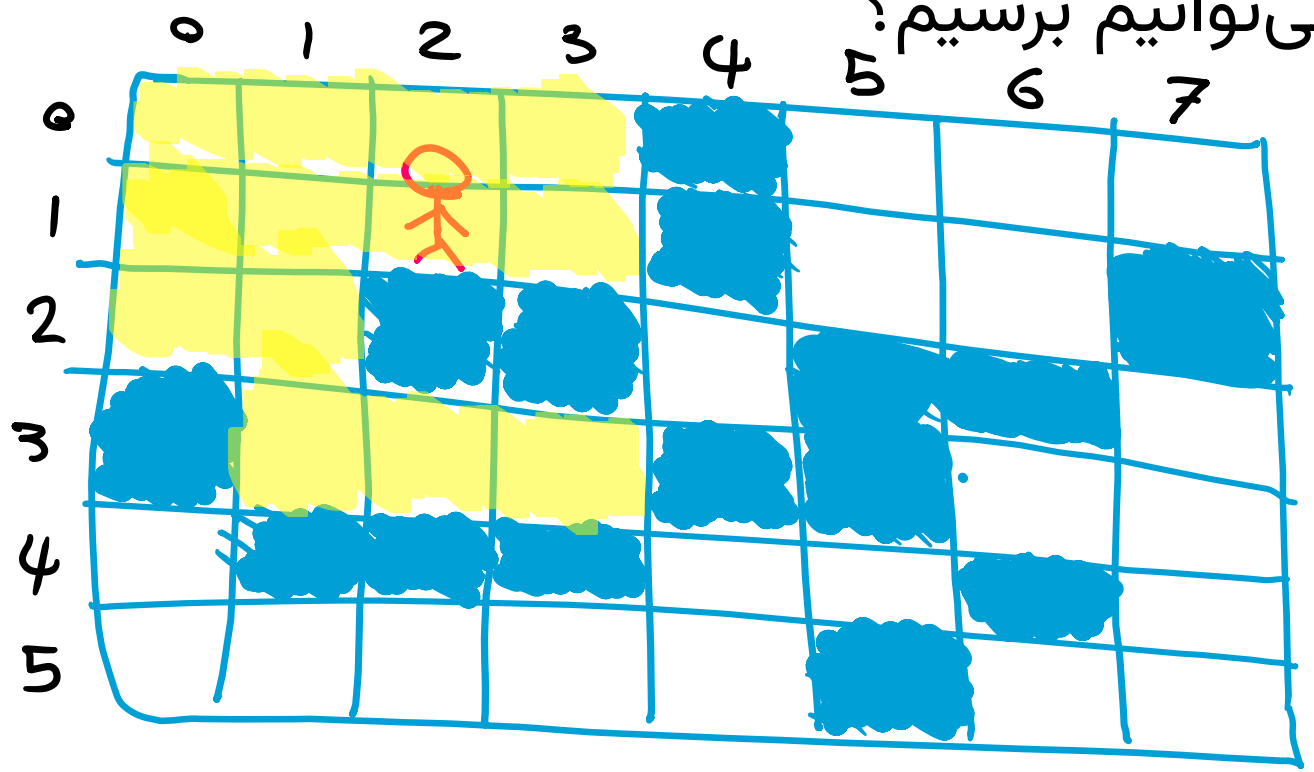
$n = 6 \quad m = 8$

$x = 1$

$y = 2$

حل ماز!

- **سوال؟** با شروع از خانه‌ی (x, y) و حرکت‌های بالا، پایین، چپ یا راست، به کدام خانه‌های جدول می‌توانیم برسیم؟



n = 6 m = 8

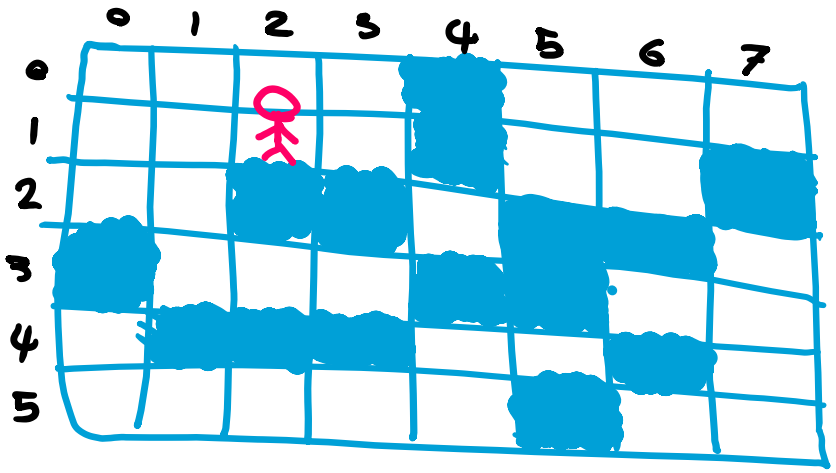
x = 1

y = 2

حل ماز!

- **سوال؟** با شروع از خانه‌ی (x, y) و حرکت‌های بالا، پایین، چپ یا راست، به کدام خانه‌های جدول می‌توانیم برسیم؟

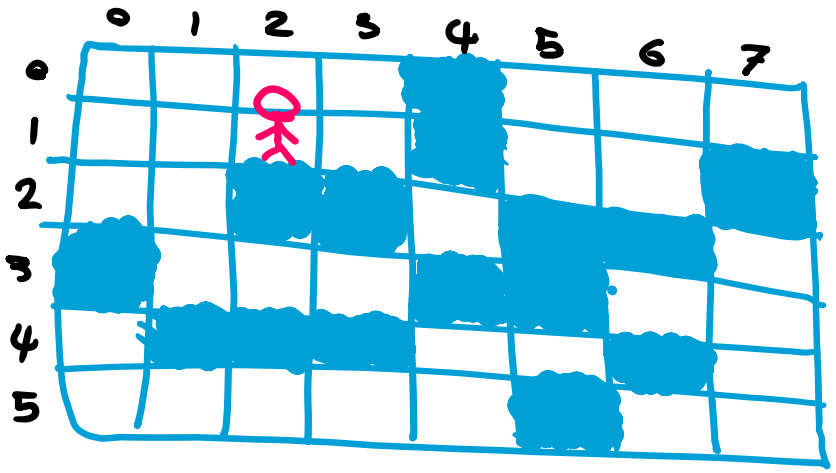
- **ایده؟**



حل ماز!

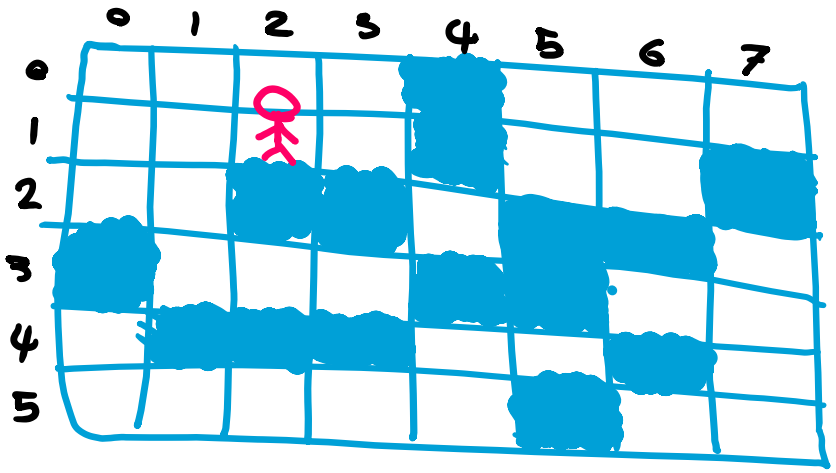
- **سوال؟** با شروع از خانه‌ی (x, y) و حرکت‌های بالا، پایین، چپ یا راست، به کدام خانه‌های جدول می‌توانیم برسیم؟

- **ایده؟** به طور بازگشتی خانه‌ها را زرد کرده و حرکت کنیم.



حل ماز!

- **قدم ۱:** خانه‌ای که در آن هستی را رنگ کن.
- **قدم ۲:** به ازای هر چهار طرف، اگر خانه خالی بود، بازگشتی مسئله را روی آن حل کن!



حل ماز!

- **قدم ۱:** خانه‌ای که در آن هستی را رنگ کن.
- **قدم ۲:** به ازای هر چهار طرف، اگر خانه خالی بود، بازگشتی مسئله را روی آن حل کن!