

# RECURSIVIDAD Y BACKTRACKING

---

75.41 - ALGORITMOS Y PROGRAMACIÓN II



# RECURSIVIDAD

**Recurrencia, recursión o recursividad** es la forma en la cual se especifica un proceso basado en su propia definición.

Se formula el concepto de recursión de la siguiente manera: Un problema que pueda ser definido en función de su tamaño, sea este **N**, pueda ser dividido en instancias más pequeñas ( $< N$ ) del mismo problema y se conozca la solución explícita a las instancias más simples, lo que se conoce como casos base, se puede aplicar inducción sobre las llamadas más pequeñas y suponer que estas quedan resueltas.

---

## recursivo, va

Del lat. recursus, part. pas. de recurrĕre 'recurrir', e -ivo.

1. **adj.** Sujeto a reglas o pautas recurrentes.
  2. **adj. Gram.** Dicho especialmente de un proceso: Que se aplica de nuevo al resultado de haberlo aplicado previamente. La subordinación es un proceso recursivo.
  3. **adj. Gram.** Dicho de una unidad o una estructura: Que puede contener como constituyente otra del mismo tipo.
-

Decimos que algo es **RECURSIVO** si se contiene a sí mismo en una versión mas pequeña.

---

## FACTORIAL

$$n! = n * (n-1) * (n-2) * (n-3) * \dots * 1$$

---

## FACTORIAL

$$n! = n * (n-1) * (n-2) * (n-3) * \dots * 1$$

$$n! = n * (n-1)!$$

---

## FACTORIAL

$$n! = n * (n-1) * (n-2) * (n-3) * \dots * 1$$

$$n! = n * (n-1)!$$

$$(n-1)! = (n-1) * (n-2)!$$

$$(n-2)! = (n-2) * (n-3)!$$

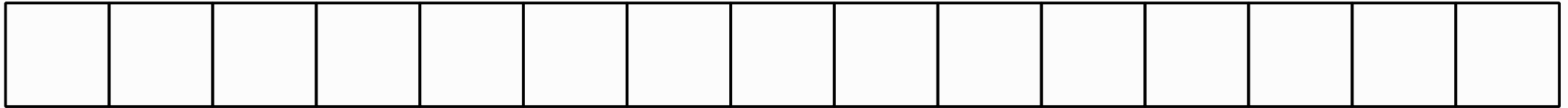
$$0! = 1$$

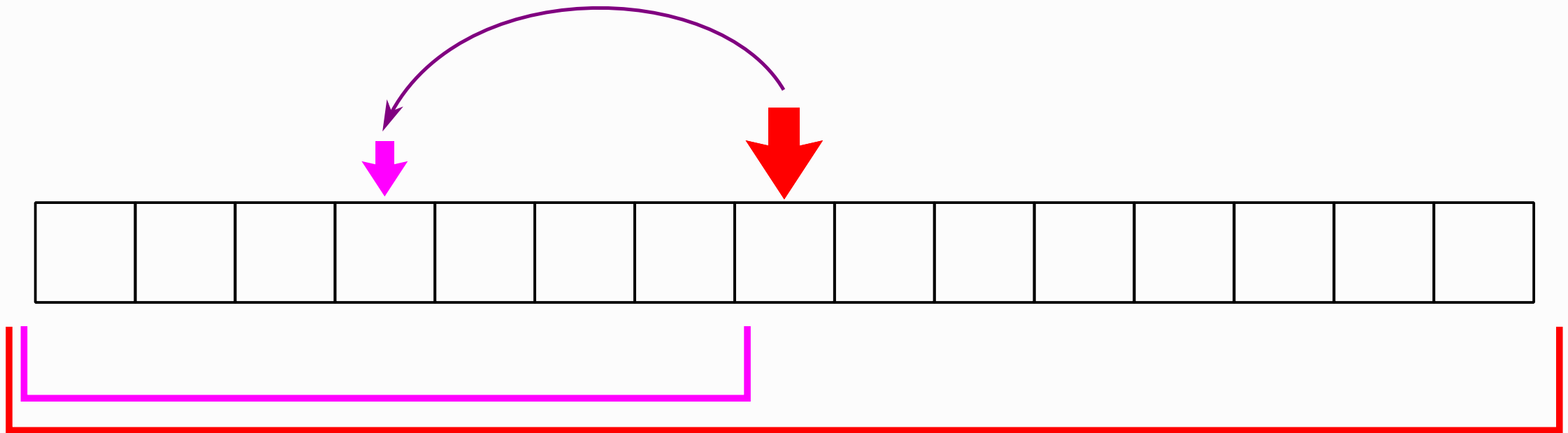
---

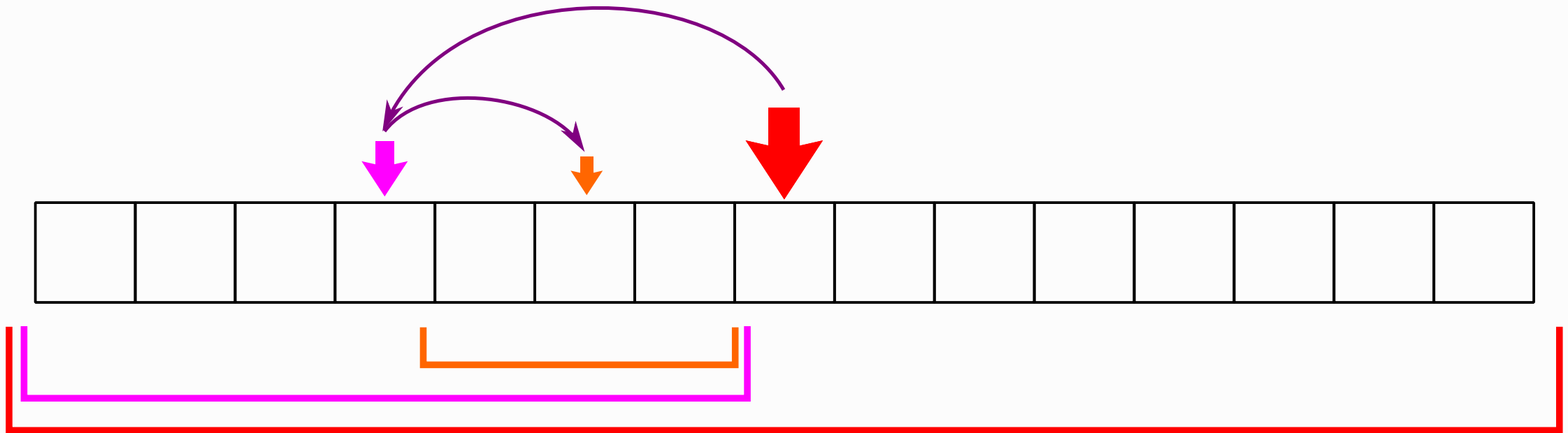


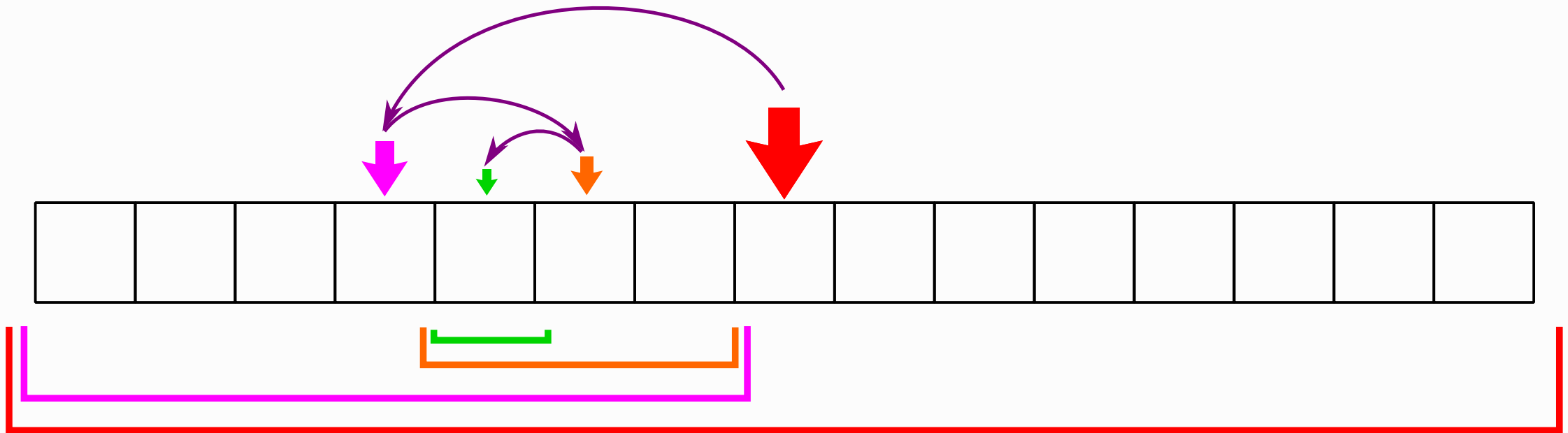
```
unsigned factorial(unsigned n){  
    if(n==0)  
        return 1;  
    return n*factorial(n-1);  
}
```

[illegible]









Sea **R** un problema recursivo

Sea **N** el tamaño del problema y **M < N**

$$R(N) = f(R(M))$$

Sean **R**, **Q** problemas recursivos

Sea **N** el tamaño del problema y  $0 < M < N$

$$R(N) = f(Q(M))$$

$$Q(M) = f(R(0))$$

---



# RECURSIVIDAD DE COLA (TAIL RECURSIVE)

Una función se dice **RECURSIVA DE COLA** si la llamada recursiva a si misma es la última operación dentro de dicha función.

---

```
unsigned factorial(unsigned n){  
    if(n==0)  
        return 1;  
    return n*factorial(n-1);  
}
```

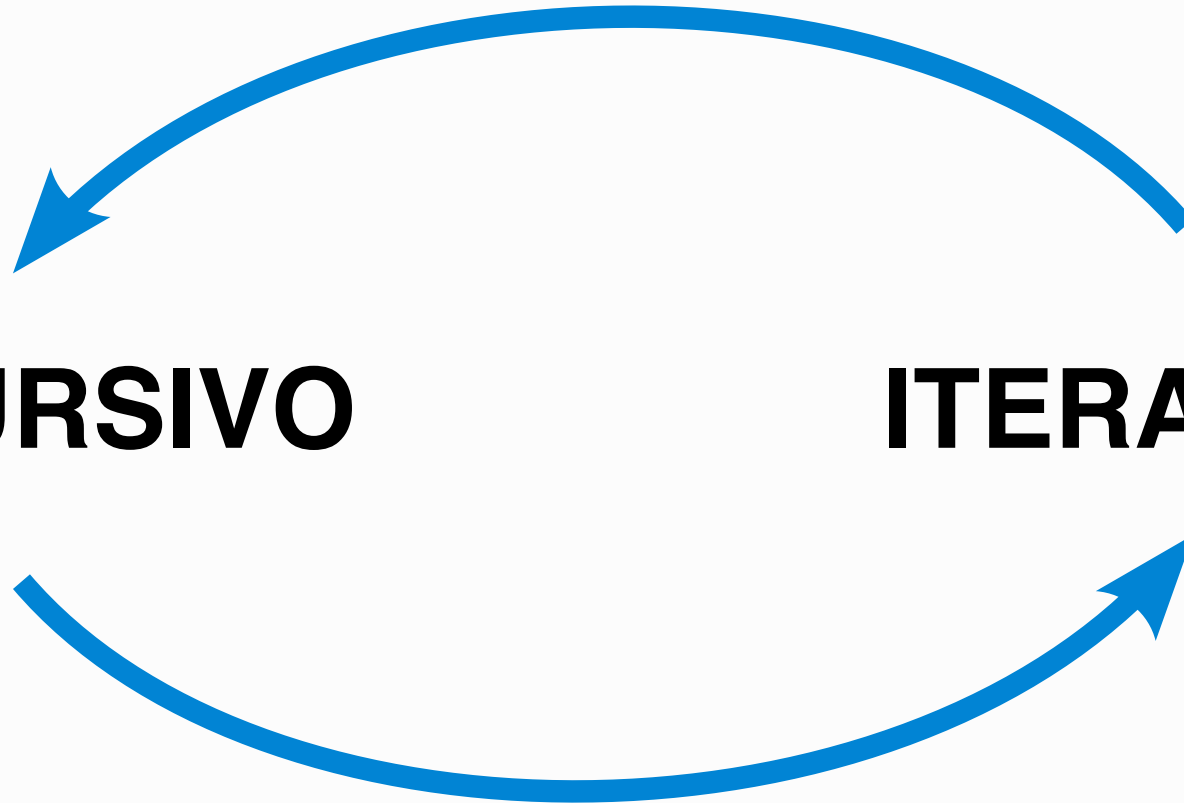
=> factorial(4)  
4\*factorial(3)  
4\*(3\*factorial(2))  
4\*(3\*(2\*factorial(1)))  
4\*(3\*(2\*(1\*factorial(0))))  
4\*(3\*(2\*(1\*1))) => 24

```
unsigned factorial_rec(unsigned n, unsigned parcial){  
    if(n==0)  
        return parcial;  
    return factorial_rec(n-1, n*parcial);  
}  
unsigned factorial(unsigned n){  
    return factorial_rec(n,1);  
}
```

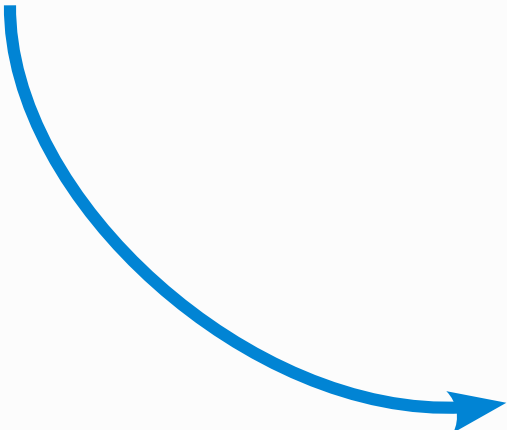
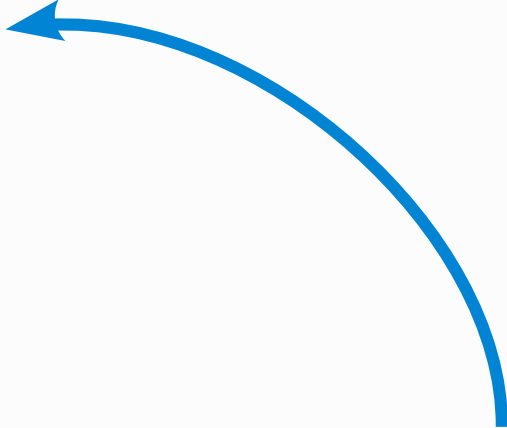
```
=> factorial(4)  
    factorial_rec(4,1)  
    factorial_rec(3,4)  
    factorial_rec(2,12)  
    factorial_rec(1,24)  
    factorial_rec(0,24) => 24
```

**RECURSIVO**

**ITERATIVO**



```
unsigned factorial_rec(unsigned n, unsigned parcial){  
    if(n==0)  
        return parcial;  
    return factorial_rec(n-1, n*parcial);  
}  
unsigned factorial(unsigned n){  
    return factorial_rec(n,1);  
}
```



```
unsigned factorial(unsigned n){  
    unsigned parcial=1;  
    for(unsigned i=n; i>1; i--){  
        parcial *= i;  
    }  
    return parcial;  
}
```

```
unsigned factorial(unsigned n){  
    if(n==0)  
        return 1;  
    return n*factorial(n-1);  
}
```

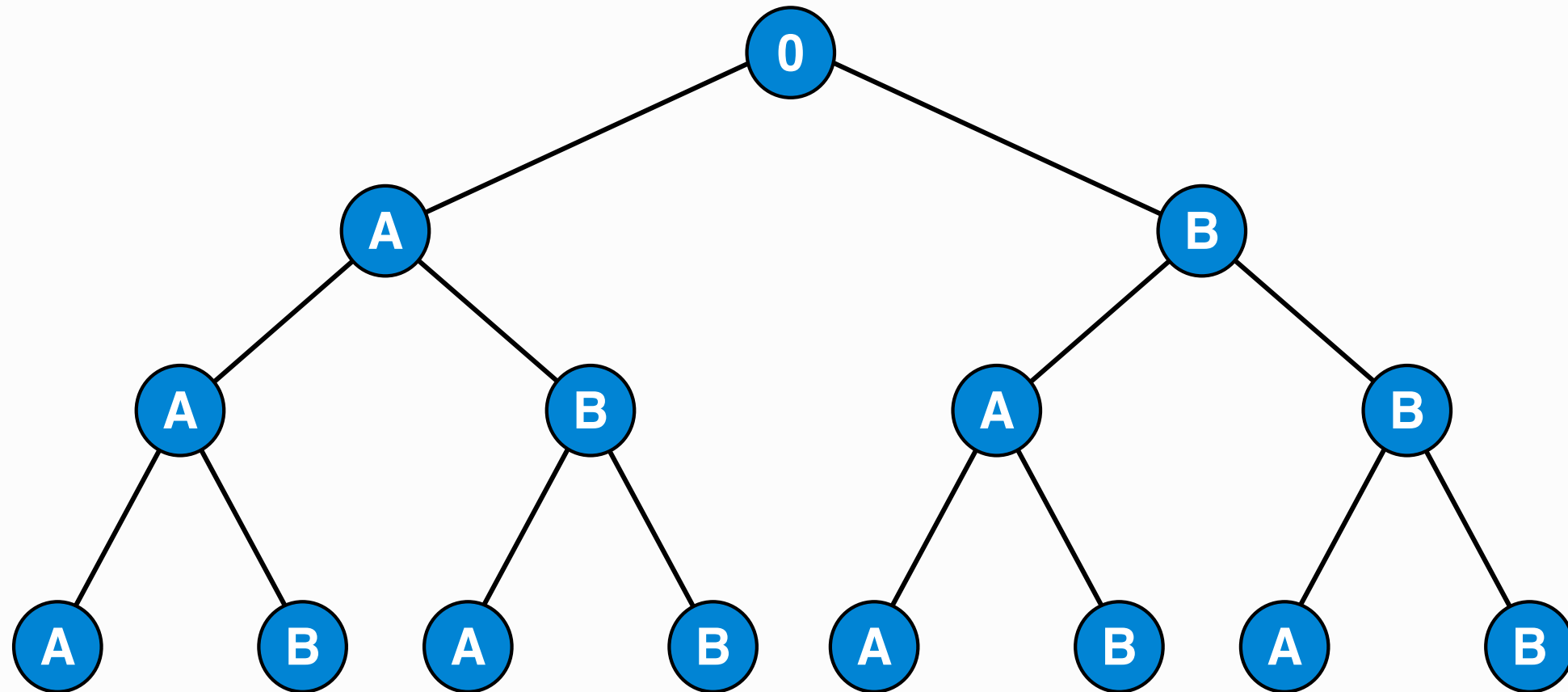
```
unsigned factorial(unsigned n){  
    unsigned resultado=1;  
    Pila pila;  
    if(n==0) return 1;  
  
    for(unsigned i=n; i>1; i--)  
        pila.push(i);  
  
    while(!pila.empty())  
        resultado *= pila.pop();  
  
    return resultado;  
}
```

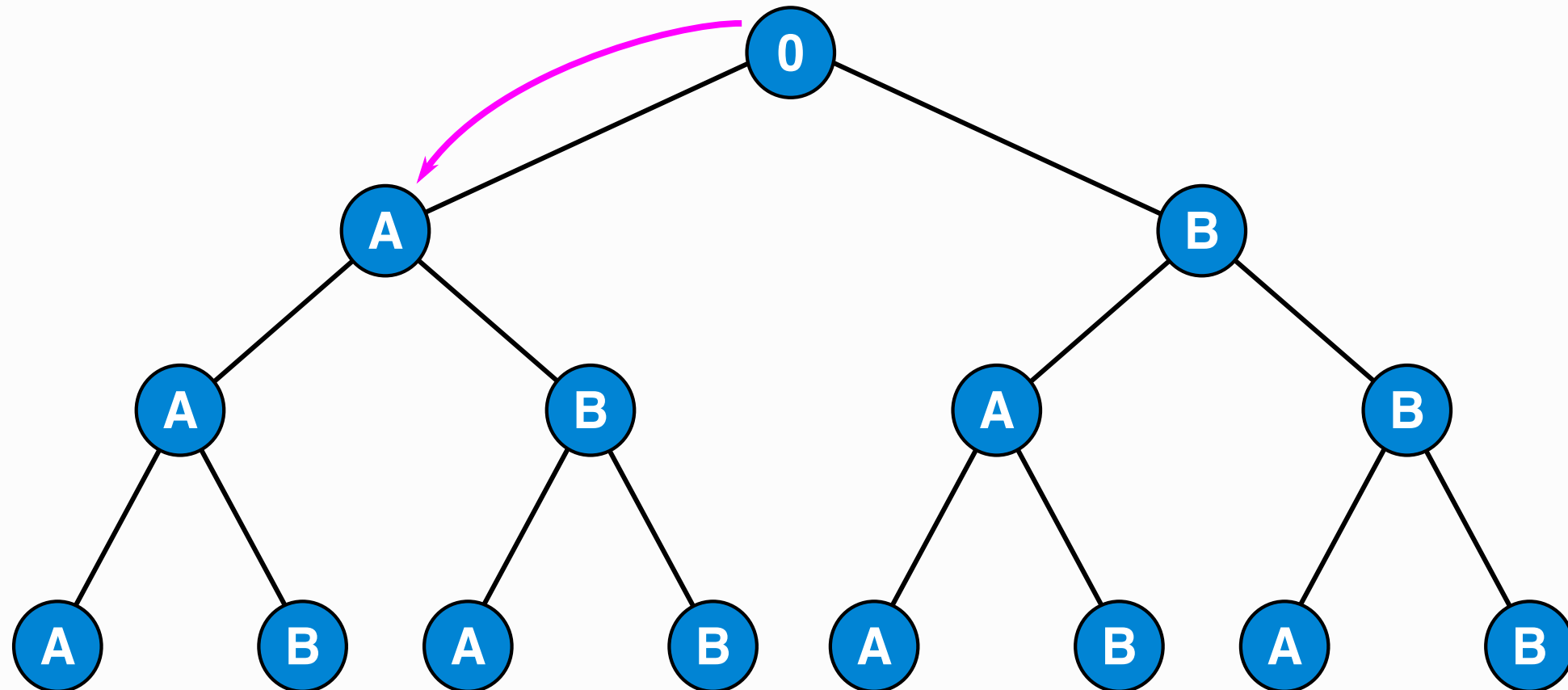
# BACKTRACKING



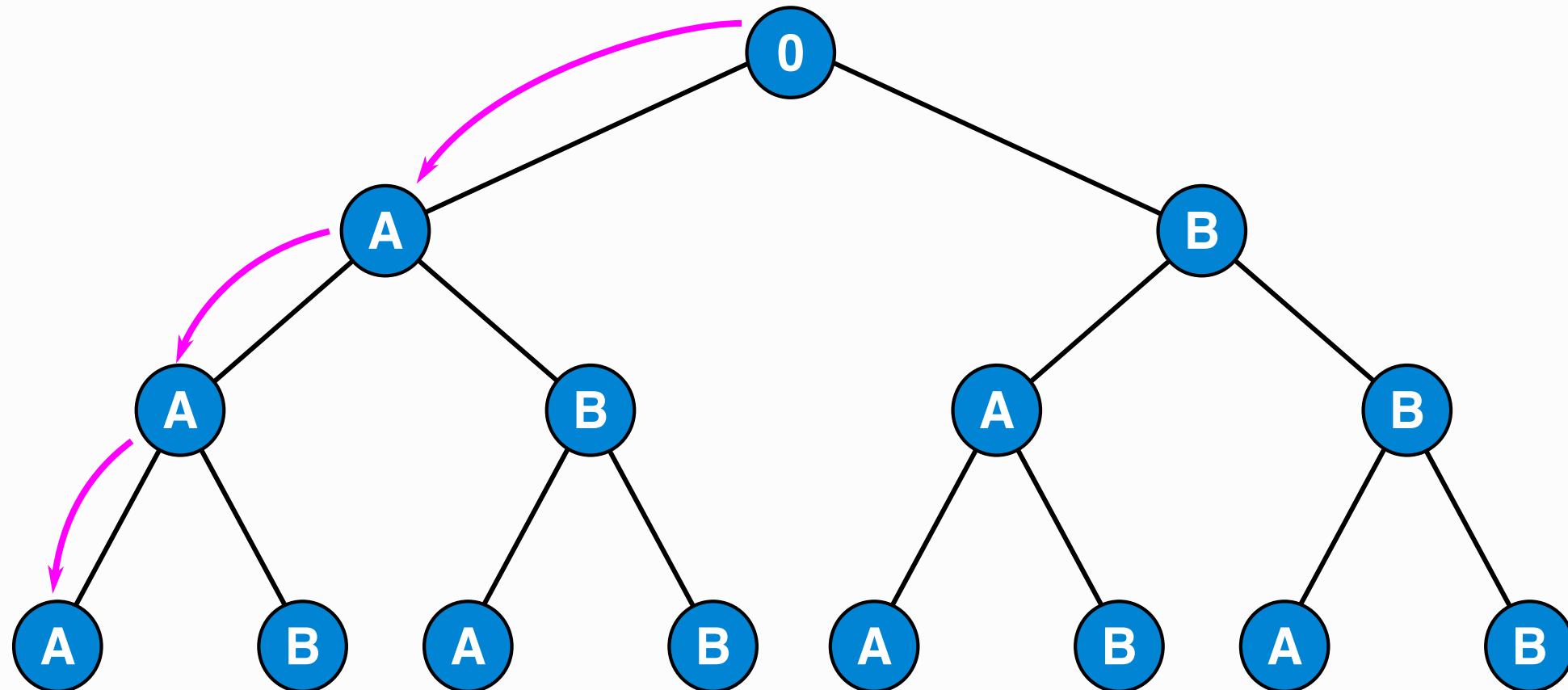
El **BACKTRACKING** es una técnica de resolución de problemas que explora recursivamente todas las soluciones posibles.

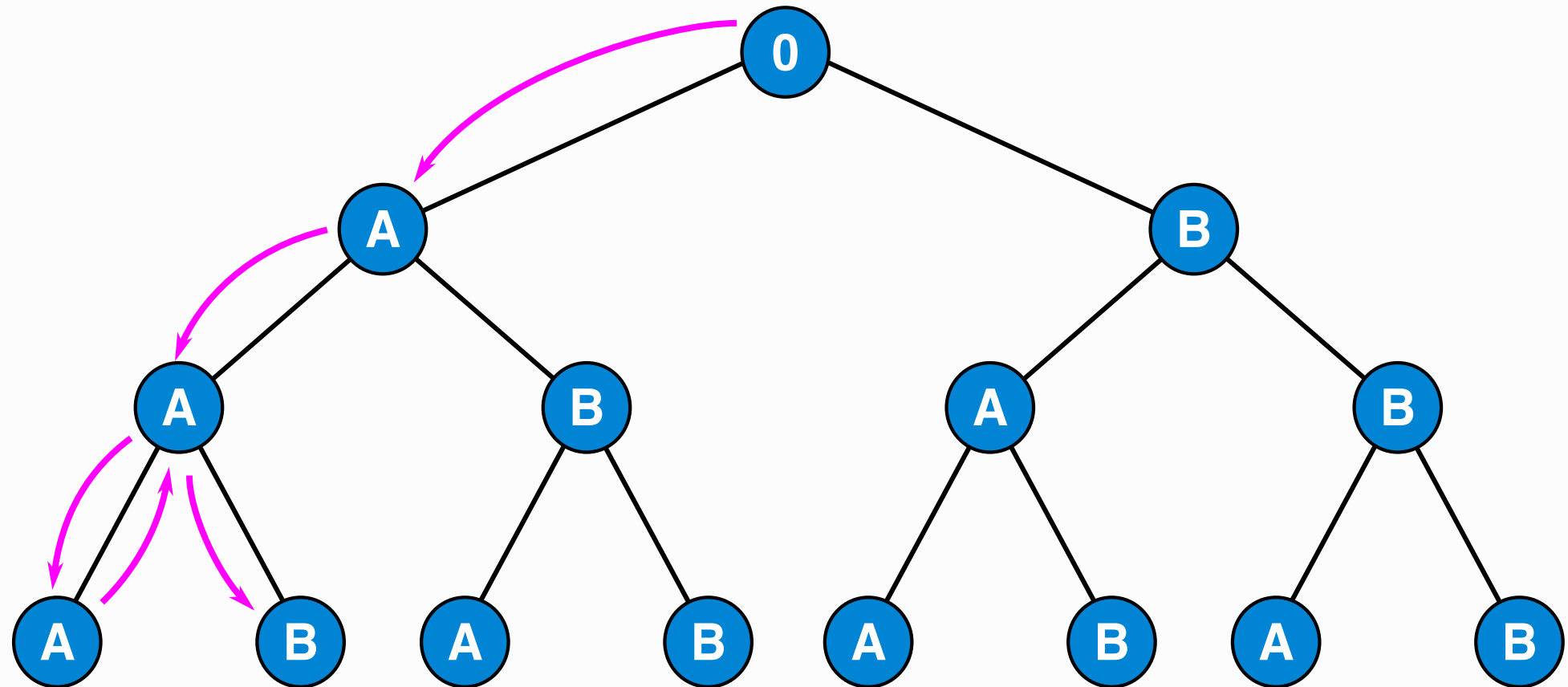
---

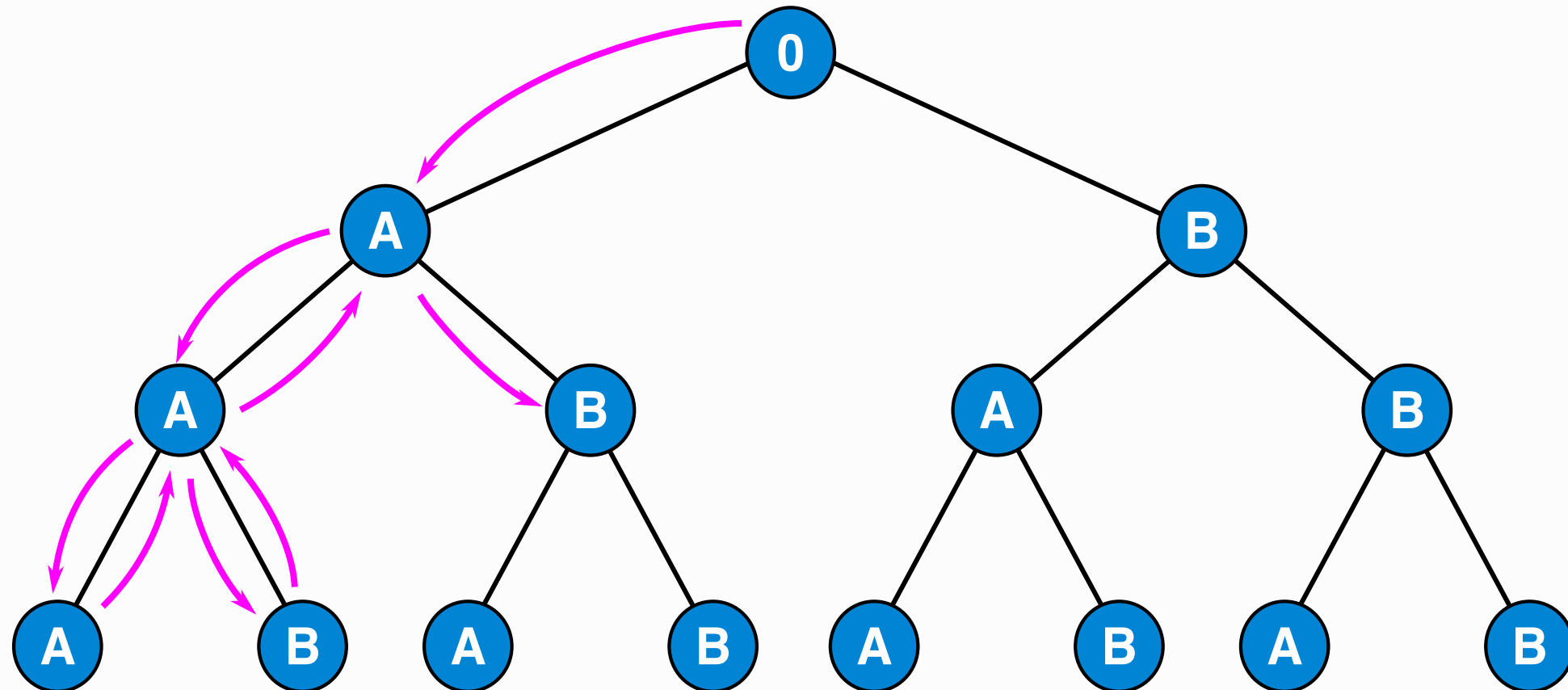


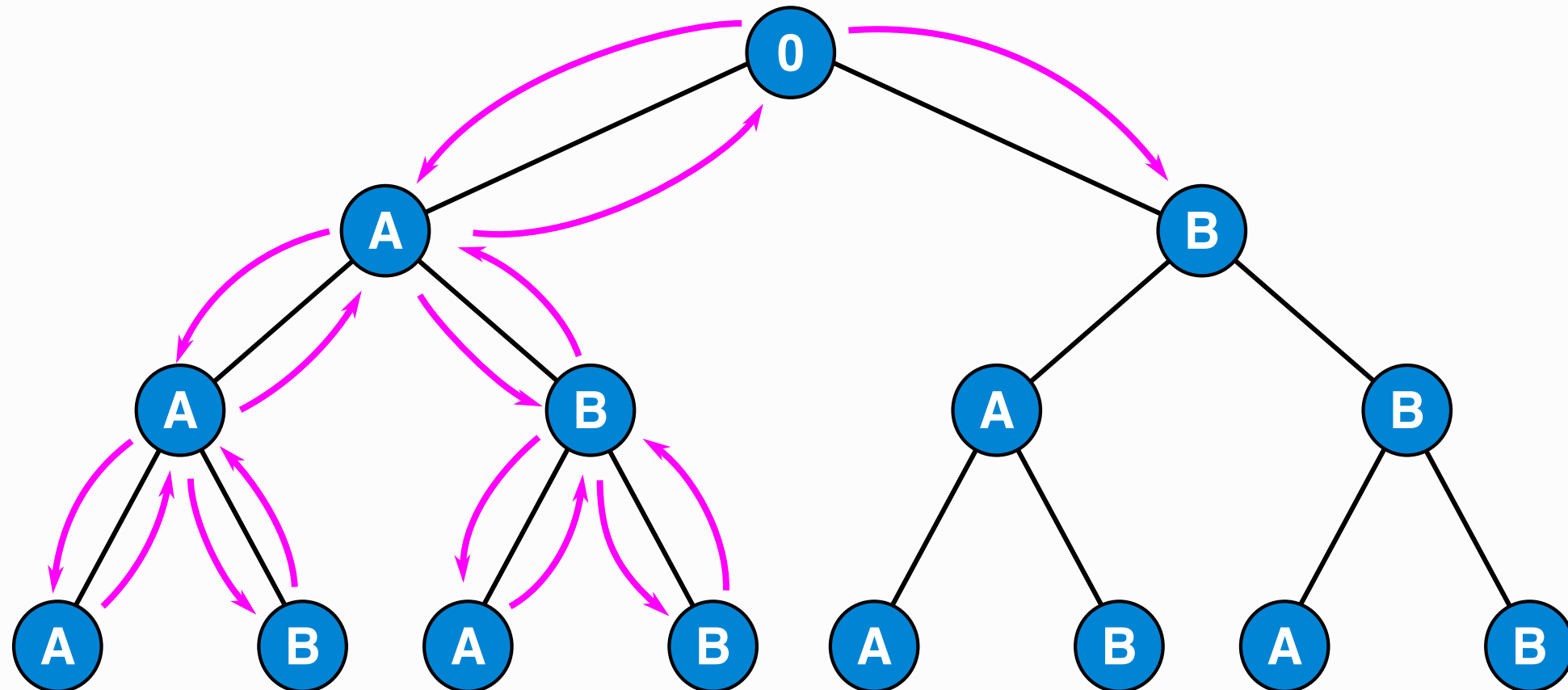




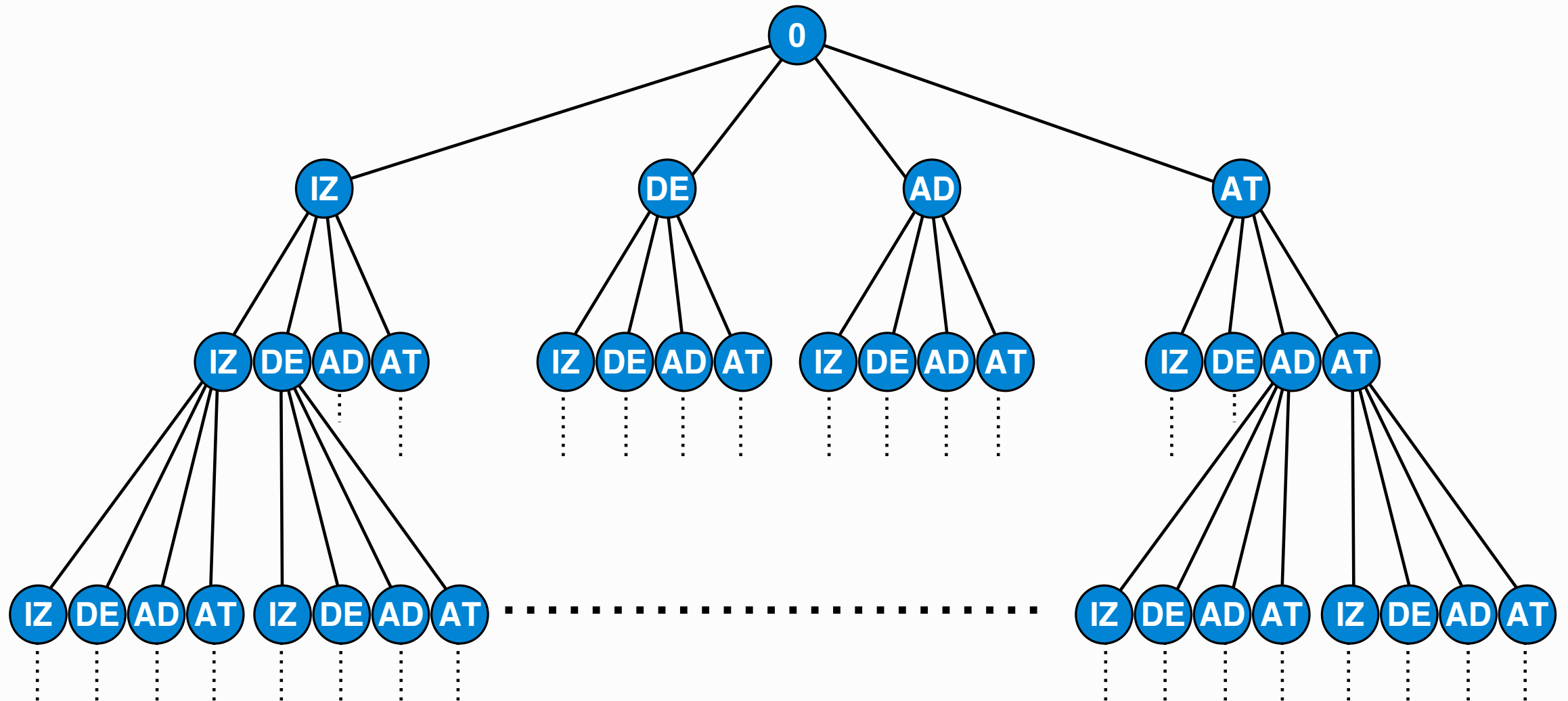


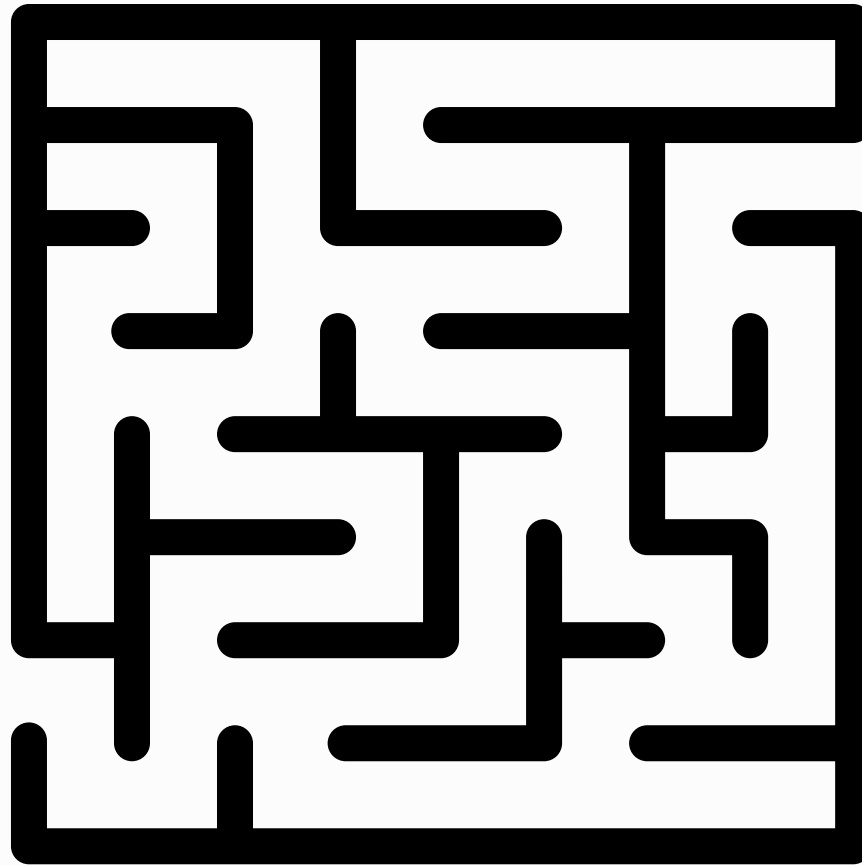


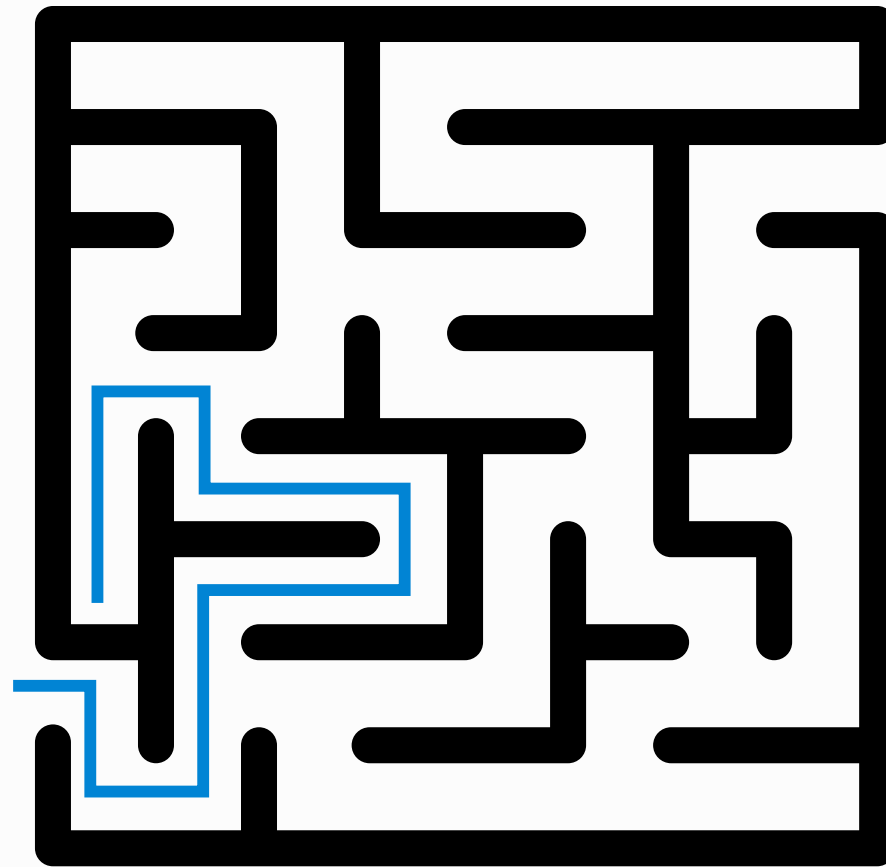


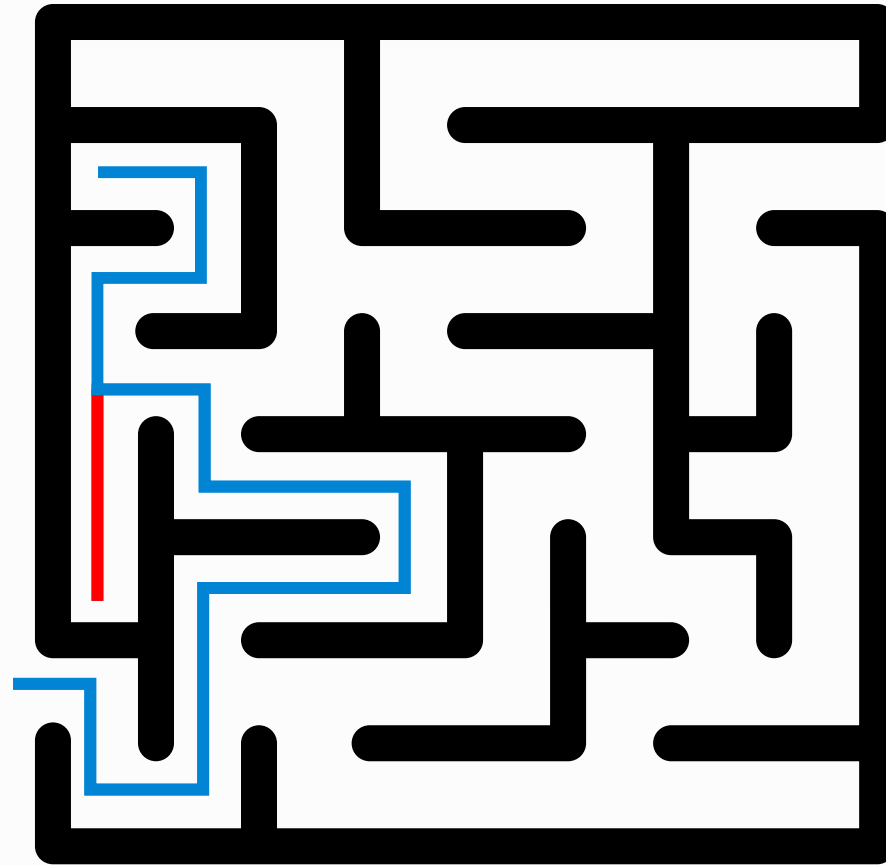


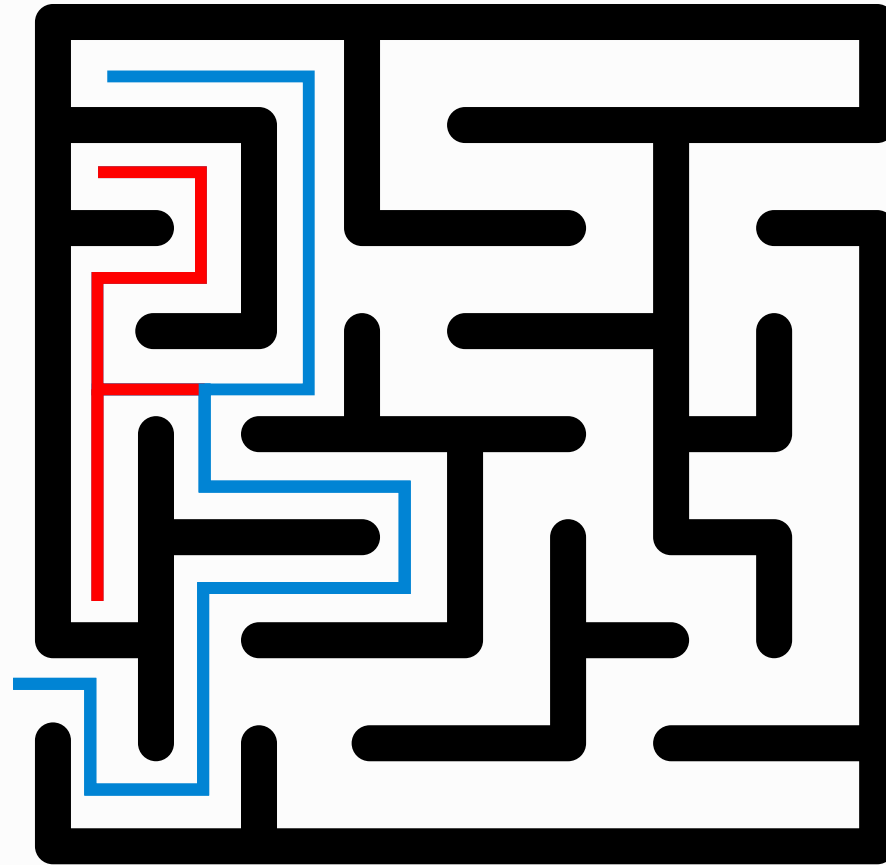


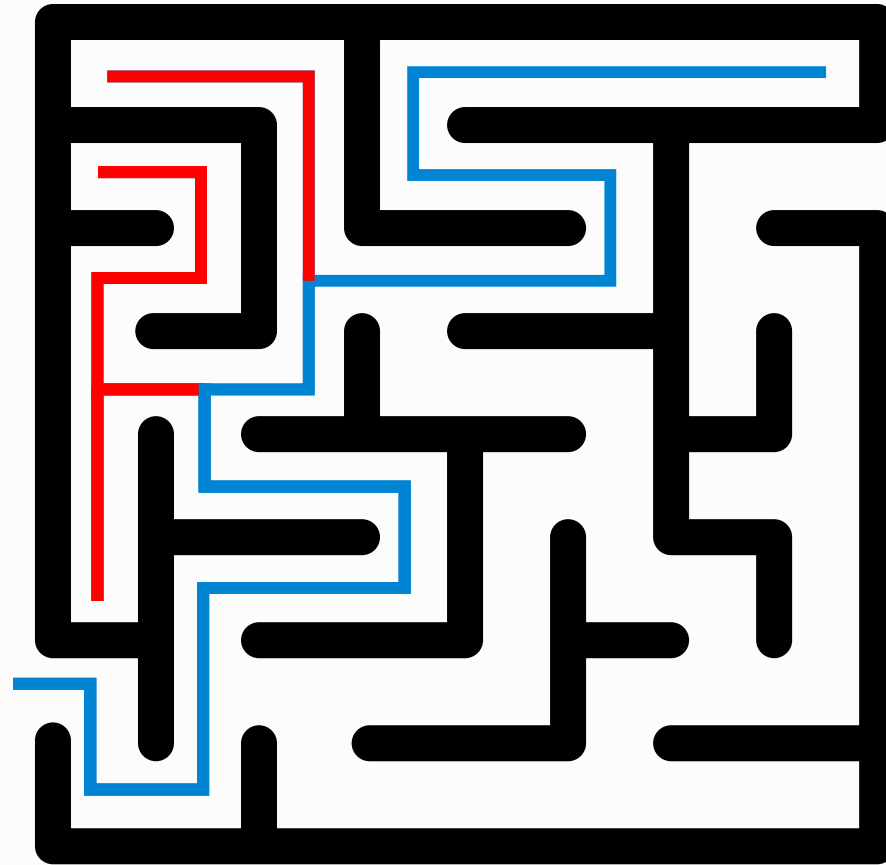












# ¿PREGUNTAS?

# FIN



**.UBA**fiuba



FACULTAD DE INGENIERÍA