

# MazeSolver

Nicola Cappello





# Introduzione

- Cos'è un labirinto?
- Cos'è MazeSolver?





# Introduzione

- Cos'è un labirinto?
- Cos'è MazeSolver?





# Introduzione

- Cos'è un labirinto?
- Cos'è MazeSolver?





# Specifica PEAS

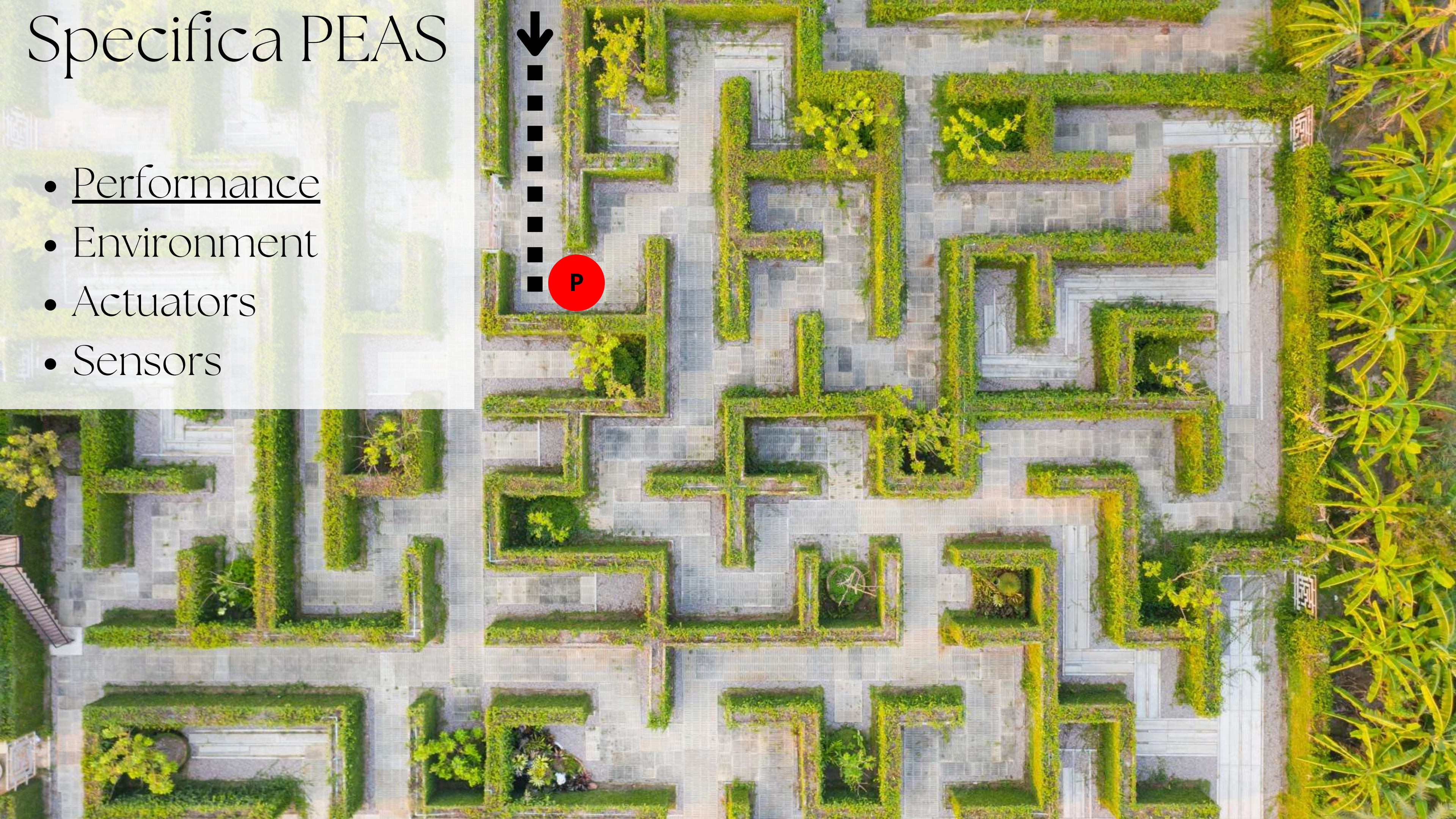
- Performance
- Environment
- Actuators
- Sensors





# Specifica PEAS

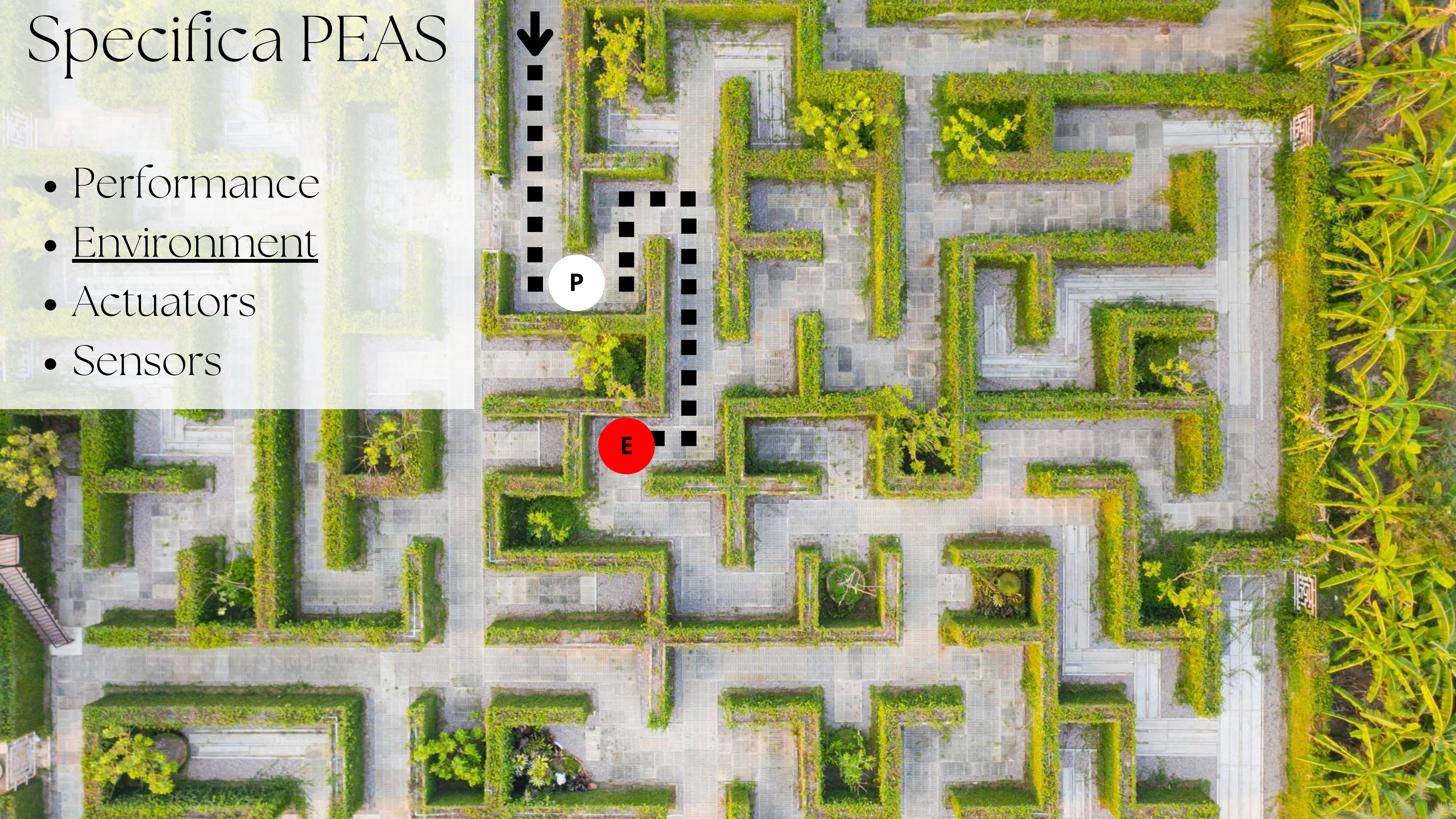
- Performance
- Environment
- Actuators
- Sensors





# Specifica PEAS

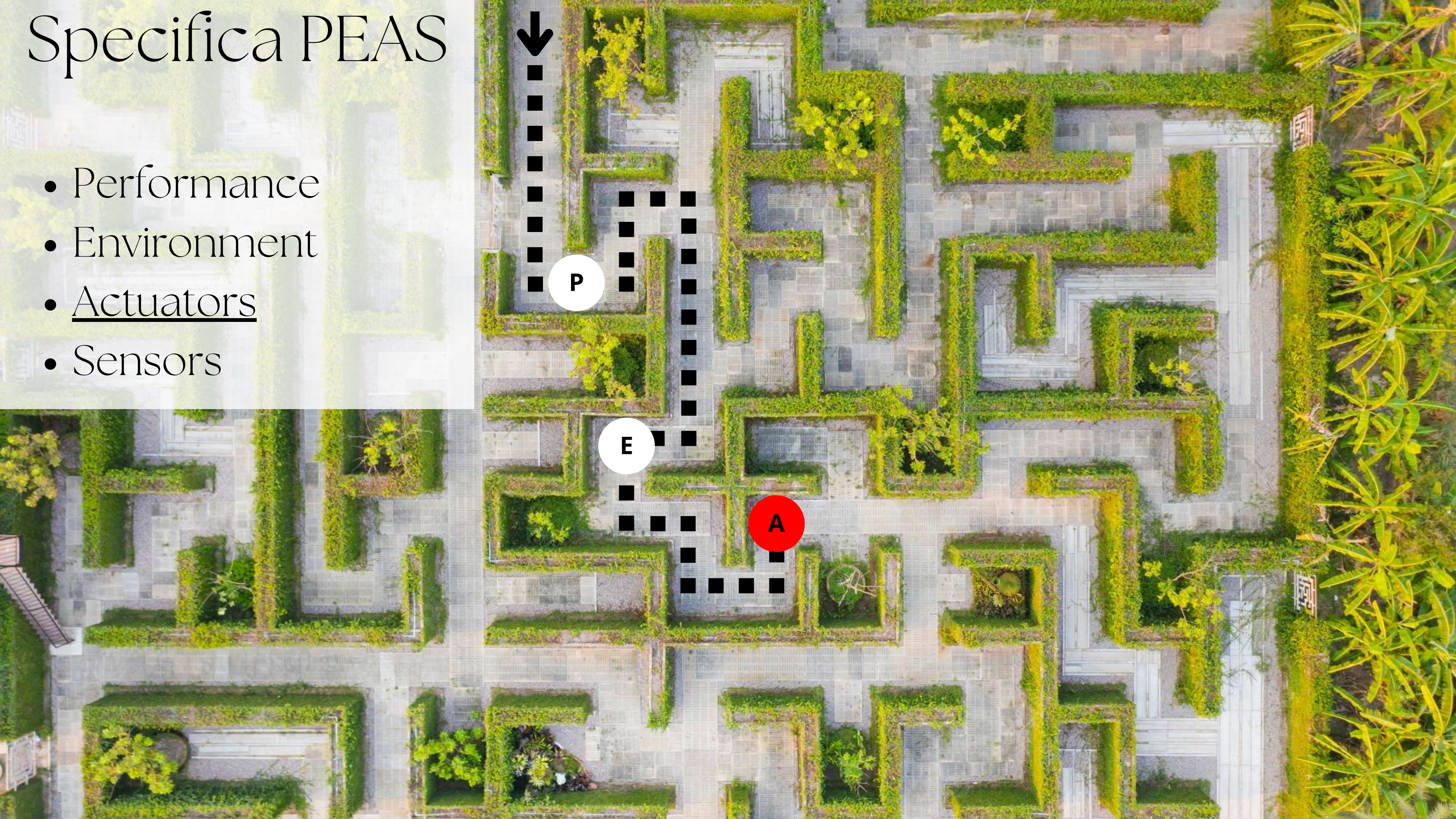
- Performance
- Environment
- Actuators
- Sensors





# Specifica PEAS

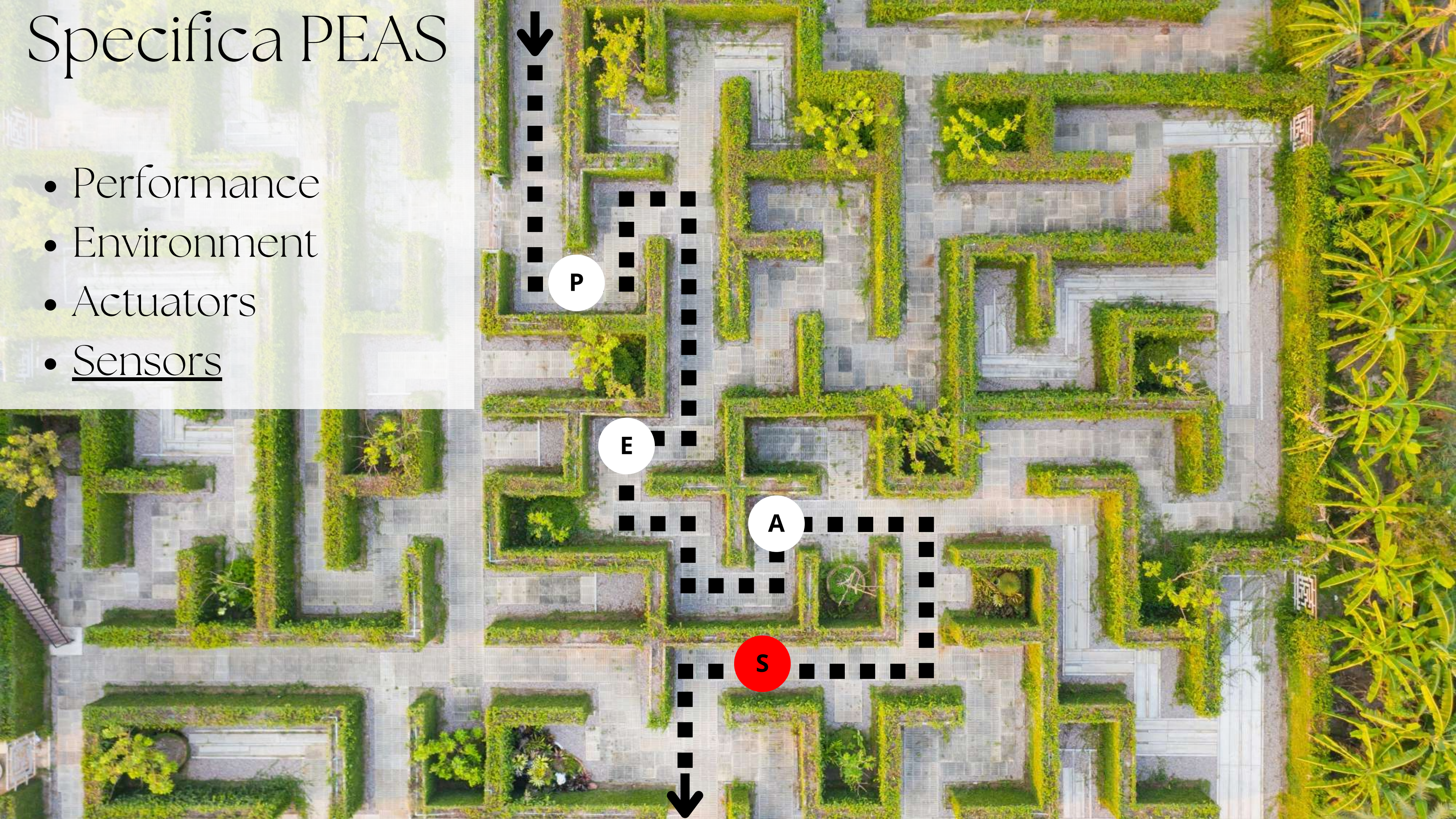
- Performance
- Environment
- Actuators
- Sensors





# Specifica PEAS

- Performance
- Environment
- Actuators
- Sensors





# Specifiche Ambiente

- Singolo agente
- Completamente osservabile
- Deterministico
- Statico
- Sequenziale
- Discreto





# Specifiche Ambiente

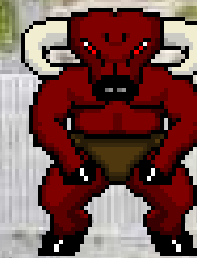
- Singolo agente
- Completamente osservabile
- Deterministico
- Statico
- Sequenziale
- Discreto





# Specifiche Ambiente

- Singolo agente
- Completamente osservabile
- Deterministico
- Statico
- Sequenziale
- Discreto



*maze[][] =  
{A1 , A2, A3...  
B1, B2, B3...}*



# Specifiche Ambiente

- Singolo agente
- Completamente osservabile
- Deterministico
- Statico
- Sequenziale
- Discreto



Sono in **E12**;  
*Risultato( $\text{in}(\text{E12}), \text{goEst}()$ )*  
=  
 *$\text{in}(\text{E13})$*



# Specifiche Ambiente

- Singolo agente
- Completamente osservabile
- Deterministico
- Statico
- Sequenziale
- Discreto

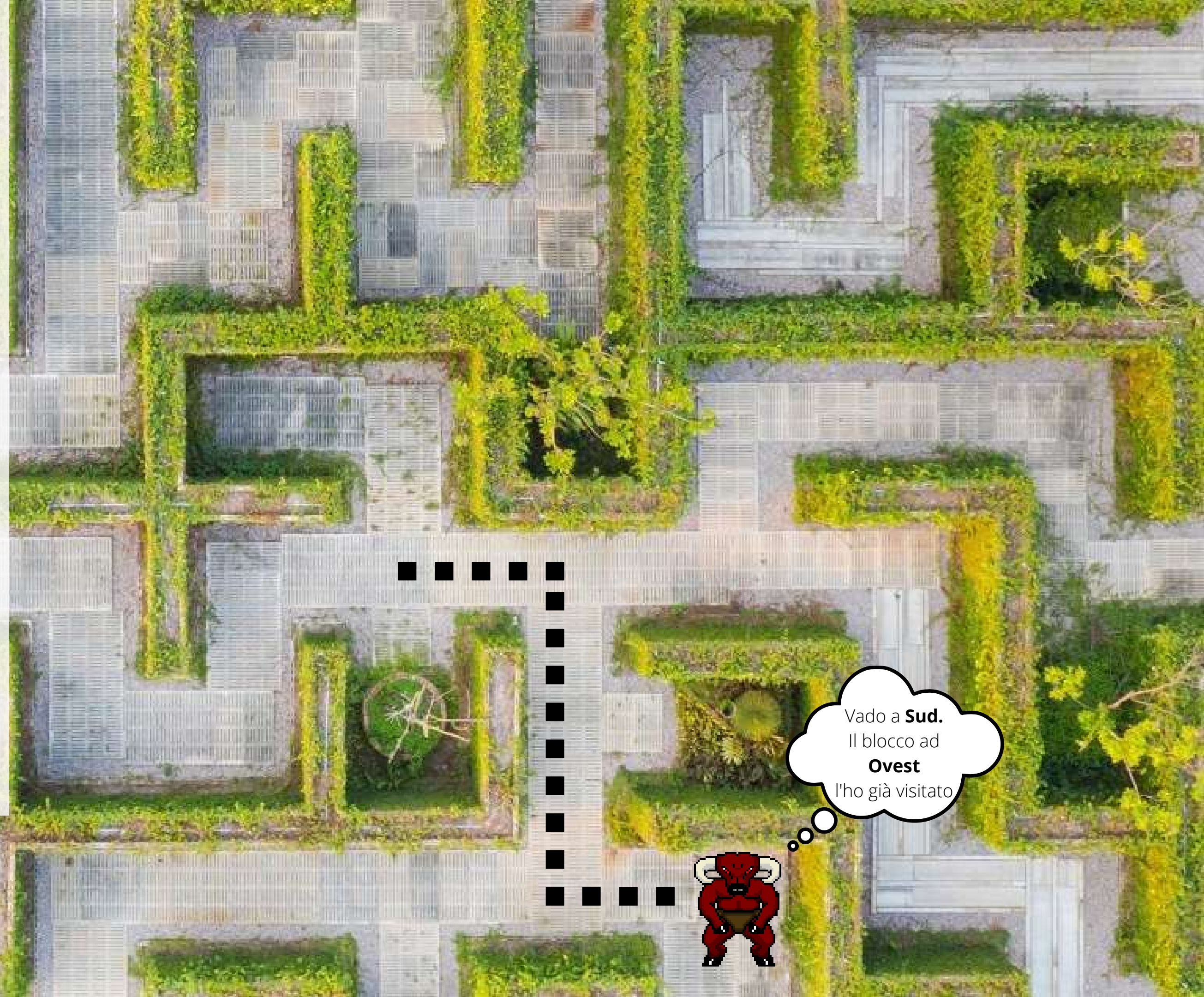


*\*Angry  
Minotaur  
Noises\**



# Specifiche Ambiente

- Singolo agente
- Completamente osservabile
- Deterministico
- Statico
- Sequenziale
- Discreto





# Specifiche Ambiente

- Singolo agente
- Completamente osservabile
- Deterministico
- Statico
- Sequenziale
- Discreto





# Analisi del problema

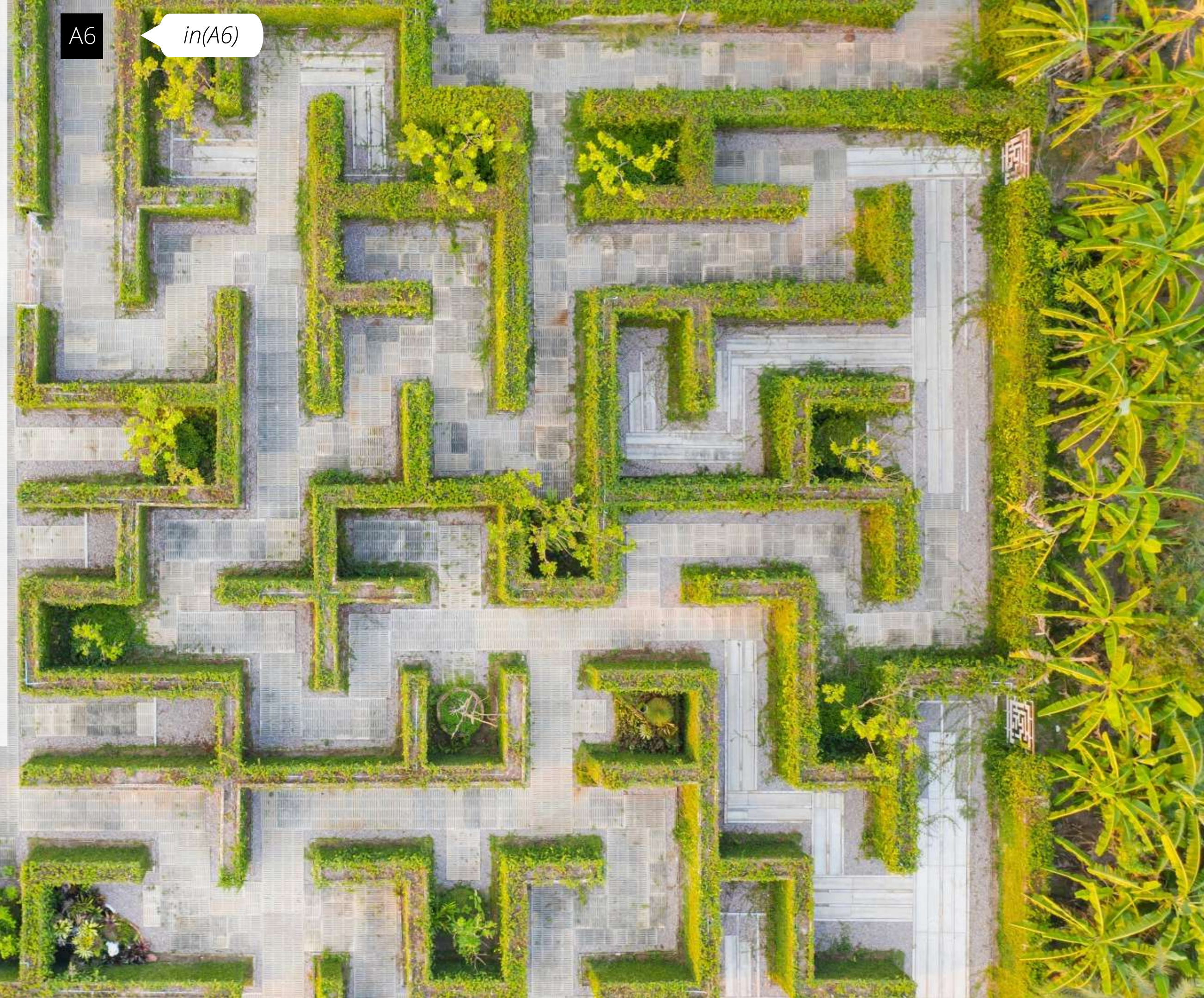
- Stato iniziale
- Azioni
- Modello di transizione
- Test obiettivo
- Costo di cammino





# Analisi del problema

- Stato iniziale
- Azioni
- Modello di transizione
- Test obiettivo
- Costo di cammino



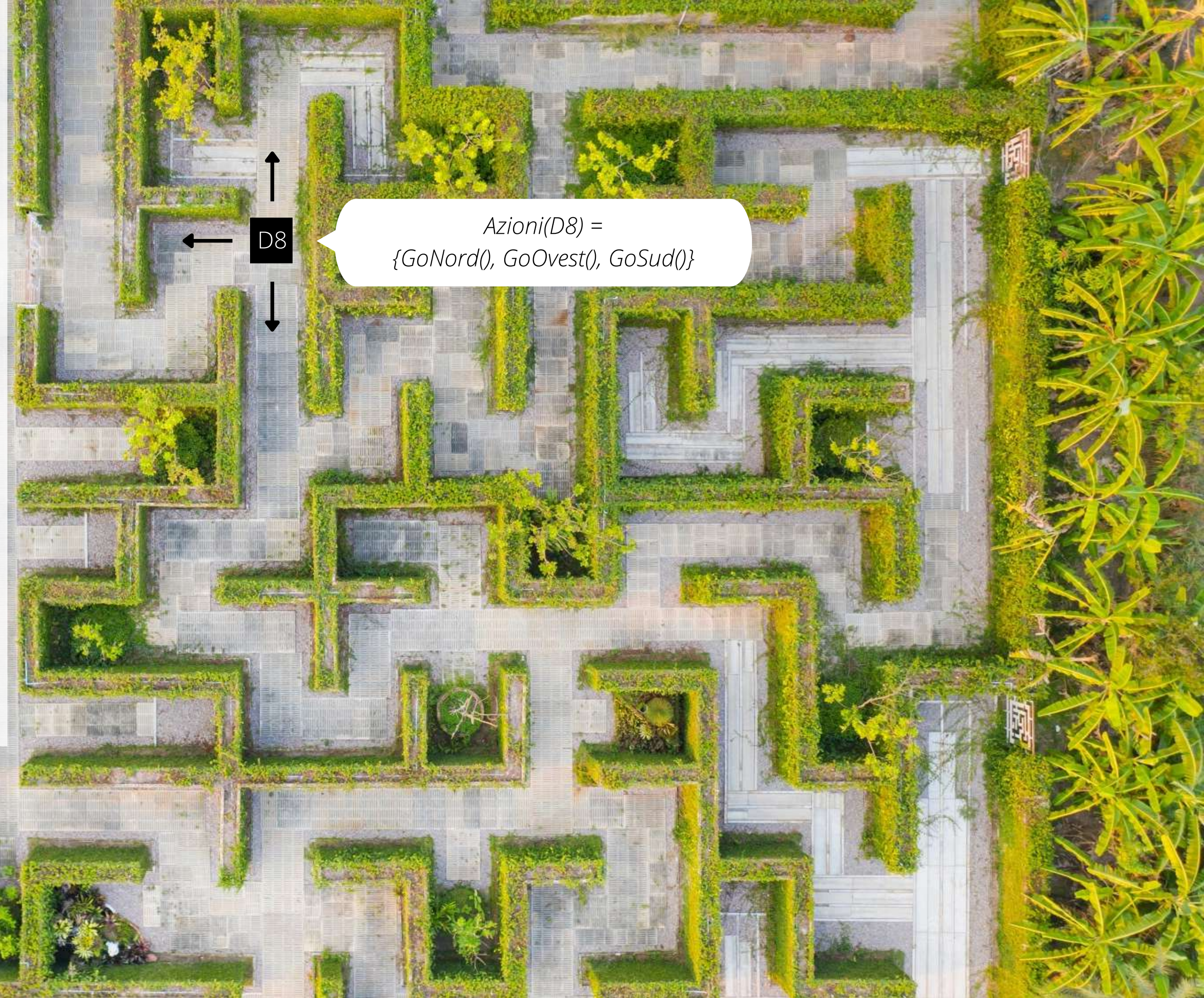
A6

in(A6)



# Analisi del problema

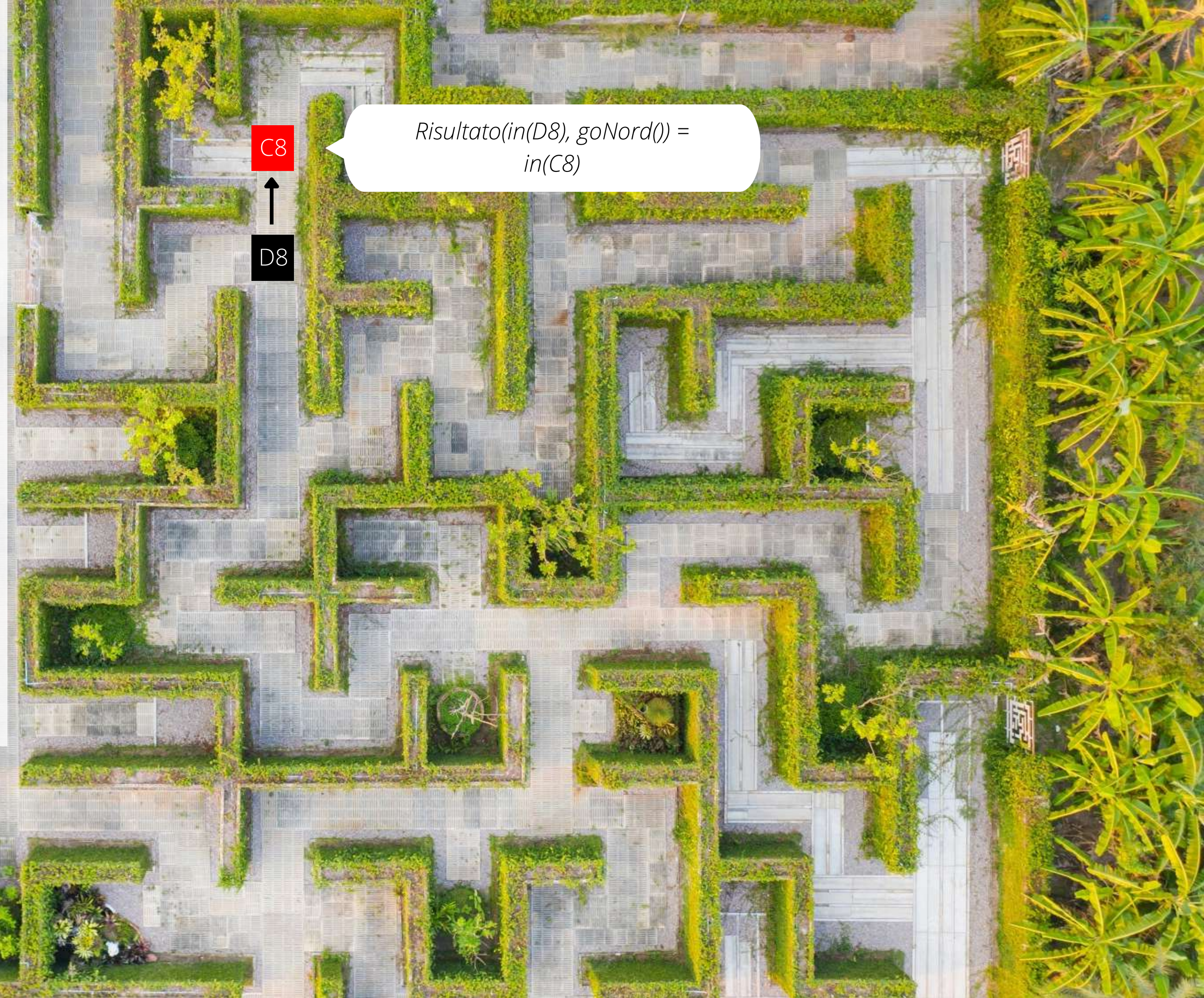
- Stato iniziale
- Azioni
- Modello di transizione
- Test obiettivo
- Costo di cammino





# Analisi del problema

- Stato iniziale
- Azioni
- Modello di transizione
- Test obiettivo
- Costo di cammino





# Analisi del problema

- Stato iniziale
- Azioni
- Modello di transizione
- Test obiettivo
- Costo di cammino



M8

{in(M8)}



# Analisi del problema

- Stato iniziale
- Azioni
- Modello di transizione
- Test obiettivo
- Costo di cammino





# Ricerca non informata

- Ricerca in Ampiezza
- Ricerca in Profondità





# Ricerca non informata

- Ricerca in Ampiezza
- Ricerca in Profondità

*stato iniziale*

D8

**nodo = labirinto.statoiniziale**

```
if (testoObiettivo(nodo))  
    return soluzione(nodo)  
frontieraFIFO.add(nodo)  
esplorati = [ ]
```

loop:

```
if (frontieraFIFO.isEmpty())  
    return fallimento  
nodo = frontieraFIFO.pop()  
esplorati.add(nodo)
```

for each azione in Azioni(nodo)

figlio = creaFiglio(nodo,azione)

if(!frontieraFIFO.contains(figlio) && !esplorati.contains(figlio))

if(testObiettivo(figlio)) return soluzione(figlio)

frontieraFIFO.add(figlio)



# Ricerca non informata

- Ricerca in Ampiezza
- Ricerca in Profondità

D8

```
nodo = labirinto.statoiniziale
if (testoObiettivo(nodo))
    return soluzione(nodo)
frontieraFIFO.add(nodo)
esplorati = [ ];

loop:
if (frontieraFIFO.isEmpty())
    return fallimento
nodo = frontieraFIFO.pop()
esplorati.add(nodo)
```

```
for each azione in Azioni(nodo)
    figlio = creaFiglio(nodo,azione)
    if(!frontieraFIFO.contains(figlio) && !esplorati.contains(figlio))
        if(testObiettivo(figlio)) return soluzione(figlio)
        frontieraFIFO.add(figlio)
```



# Ricerca non informata

- Ricerca in Ampiezza
- Ricerca in Profondità

D8

```
nodo = labirinto.statoIniziale  
if (testoObiettivo(nodo))  
    return soluzione(nodo)  
frontieraFIFO.add(nodo)  
esplorati = [ ]
```

```
loop:  
if (frontieraFIFO.isEmpty())  
    return fallimento  
nodo = frontieraFIFO.pop()  
esplorati.add(nodo)
```

```
for each azione in Azioni(nodo)  
    figlio = creaFiglio(nodo,azione)  
    if(!frontieraFIFO.contains(figlio) && !esplorati.contains(figlio))  
        if(testObiettivo(figlio)) return soluzione(figlio)  
        frontieraFIFO.add(figlio)
```

```
frontieraFIFO = [D8]  
esplorati = [ ]
```



# Ricerca non informata

- Ricerca in Ampiezza
- Ricerca in Profondità

D8

```
nodo = labirinto.statoIniziale  
if (testoObiettivo(nodo))  
    return soluzione(nodo)  
frontieraFIFO.add(nodo)  
esplorati = [ ]
```

```
loop:  
if (frontieraFIFO.isEmpty())  
    return fallimento
```

```
nodo = frontieraFIFO.pop()  
esplorati.add(nodo)
```

```
for each azione in Azioni(nodo)  
    figlio = creaFiglio(nodo,azione)  
    if(!frontieraFIFO.contains(figlio) && !esplorati.contains(figlio))  
        if(testObiettivo(figlio)) return soluzione(figlio)  
        frontieraFIFO.add(figlio)
```

```
frontieraFIFO = [D8]  
esplorati = [ ]
```



# Ricerca non informata

- Ricerca in Ampiezza
- Ricerca in Profondità

D8

```
nodo = labirinto.statoIniziale  
if (testoObiettivo(nodo))  
    return soluzione(nodo)  
frontieraFIFO.add(nodo)  
esplorati = [ ]
```

```
loop:  
if (frontieraFIFO.isEmpty())  
    return fallimento
```

```
nodo = frontieraFIFO.pop()  
esplorati.add(nodo)
```

```
for each azione in Azioni(nodo)  
    figlio = creaFiglio(nodo,azione)  
    if(!frontieraFIFO.contains(figlio) && !esplorati.contains(figlio))  
        if(testObiettivo(figlio)) return soluzione(figlio)  
        frontieraFIFO.add(figlio)
```

```
frontieraFIFO = [ ]  
esplorati = [D8]
```



# Ricerca non informata

- Ricerca in Ampiezza
- Ricerca in Profondità



```
nodo = labirinto.statoIniziale  
if (testoObiettivo(nodo))  
    return soluzione(nodo)  
frontieraFIFO.add(nodo)  
esplorati = [ ]
```

```
loop:  
if (frontieraFIFO.isEmpty())  
    return fallimento  
nodo = frontieraFIFO.pop()  
esplorati.add(nodo)
```

**for each azione in Azioni(nodo)**

**figlio = creaFiglio(nodo,azione)**

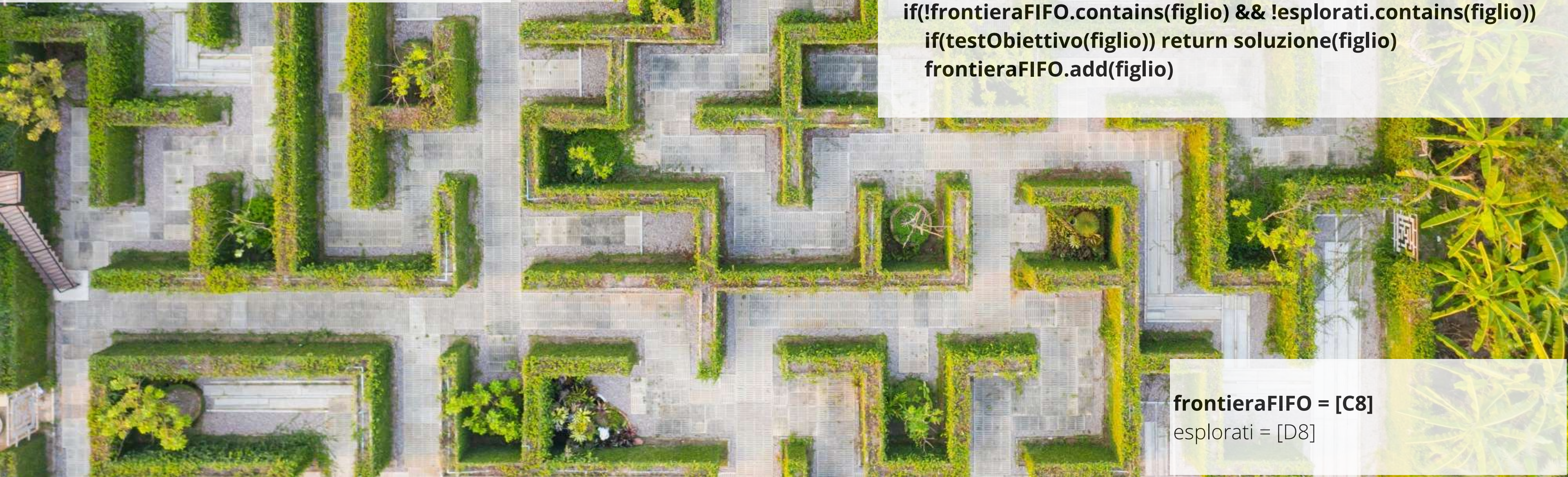
```
if(!frontieraFIFO.contains(figlio) && !esplorati.contains(figlio))  
    if(testObiettivo(figlio)) return soluzione(figlio)  
    frontieraFIFO.add(figlio)
```

```
frontieraFIFO = []  
esplorati = [D8]
```



# Ricerca non informata

- Ricerca in Ampiezza
- Ricerca in Profondità



```
nodo = labirinto.statoIniziale  
if (testoObiettivo(nodo))  
    return soluzione(nodo)  
frontieraFIFO.add(nodo)  
esplorati = [ ]
```

```
loop:  
if (frontieraFIFO.isEmpty())  
    return fallimento  
nodo = frontieraFIFO.pop()  
esplorati.add(nodo)
```

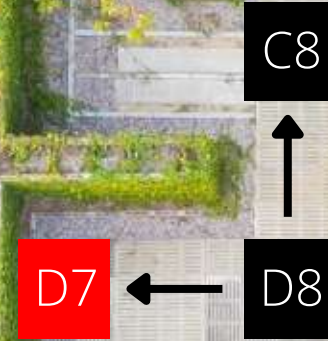
```
for each azione in Azioni(nodo)  
    figlio = creaFiglio(nodo,azione)  
    if(!frontieraFIFO.contains(figlio) && !esplorati.contains(figlio))  
        if(testObiettivo(figlio)) return soluzione(figlio)  
        frontieraFIFO.add(figlio)
```

**frontieraFIFO = [C8]**  
esplorati = [D8]



# Ricerca non informata

- Ricerca in Ampiezza
- Ricerca in Profondità



```
nodo = labirinto.statoIniziale  
if (testoObiettivo(nodo))  
    return soluzione(nodo)  
frontieraFIFO.add(nodo)  
esplorati = [ ]
```

```
loop:  
if (frontieraFIFO.isEmpty())  
    return fallimento  
nodo = frontieraFIFO.pop()  
esplorati.add(nodo)
```

```
for each azione in Azioni(nodo)  
    figlio = creaFiglio(nodo,azione)  
    if(!frontieraFIFO.contains(figlio) && !esplorati.contains(figlio))  
        if(testObiettivo(figlio)) return soluzione(figlio)  
        frontieraFIFO.add(figlio)
```

**frontieraFIFO = [C8, D7]**  
esplorati = [D8]



# Ricerca non informata

- Ricerca in Ampiezza
- Ricerca in Profondità



```
nodo = labirinto.statoIniziale  
if (testoObiettivo(nodo))  
    return soluzione(nodo)  
frontieraFIFO.add(nodo)  
esplorati = [ ]
```

```
loop:  
if (frontieraFIFO.isEmpty())  
    return fallimento  
nodo = frontieraFIFO.pop()  
esplorati.add(nodo)
```

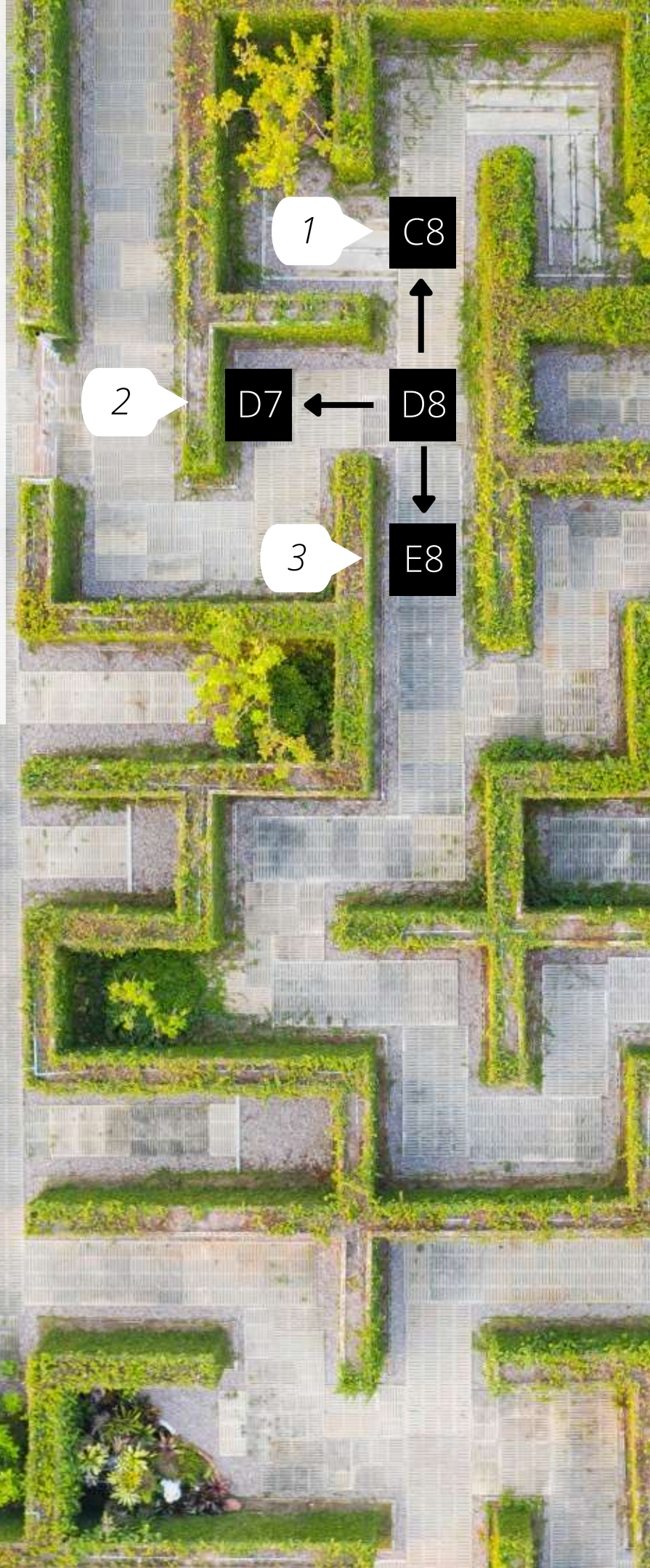
```
for each azione in Azioni(nodo)  
    figlio = creaFiglio(nodo,azione)  
    if(!frontieraFIFO.contains(figlio) && !esplorati.contains(figlio))  
        if(testObiettivo(figlio)) return soluzione(figlio)  
        frontieraFIFO.add(figlio)
```

**frontieraFIFO = [C8, D7, E8]**  
esplorati = [D8]



# Ricerca non informata

- Ricerca in Ampiezza
- Ricerca in Profondità



```
nodo = labirinto.statoiniziale  
if (testoObiettivo(nodo))  
    return soluzione(nodo)  
frontieraFIFO.add(nodo)  
esplorati = [ ]
```

```
loop:  
if (frontieraFIFO.isEmpty())  
    return fallimento  
nodo = frontieraFIFO.pop();  
esplorati.add(nodo)
```

```
for each azione in Azioni(nodo)  
    figlio = creaFiglio(nodo,azione)  
    if(!frontieraFIFO.contains(figlio) && !esplorati.contains(figlio))  
        if(testObiettivo(figlio)) return soluzione(figlio)  
        frontieraFIFO.add(figlio)
```

```
frontieraFIFO = [C8, D7, E8]  
esplorati = [D8]
```



# Ricerca non informata

- Ricerca in Ampiezza
- Ricerca in Profondità

*stato iniziale*

D8

```
nodo = labirinto.statoiniziale  
if (testoObiettivo(nodo))  
    return soluzione(nodo)  
frontieraLIFO.add(nodo)  
esplorati = [ ]
```

```
loop:  
if (frontieraLIFO.isEmpty())  
    return fallimento  
nodo = frontieraLIFO.pop()  
esplorati.add(nodo)
```

```
for each azione in Azioni(nodo)  
    figlio = creaFiglio(nodo,azione)  
    if(!frontieraLIFO.contains(figlio) && !esplorati.contains(figlio))  
        if(testObiettivo(figlio)) return soluzione(figlio)  
        frontieraLIFO.add(figlio)
```



# Ricerca non informata

- Ricerca in Ampiezza
- Ricerca in Profondità



**frontieraLIFO** = [C8]  
esplorati = [D8]



# Ricerca non informata

- Ricerca in Ampiezza
- Ricerca in Profondità



**frontieraLIFO = [D7, C8]**  
esplorati = [D8]



# Ricerca non informata

- Ricerca in Ampiezza
- Ricerca in Profondità



**frontieraLIFO** = [E8, D7, C8]  
esplorati = [D8]



# Ricerca non informata

- Ricerca in Ampiezza
- Ricerca in Profondità



**frontieraLIFO** = [D7, C8]  
esplorati = [D8]



# Ricerca non informata

- Ricerca in Ampiezza
- Ricerca in Profondità



**frontieraLIFO** = [F8, D7, C8]  
esplorati = [D8]



# Ricerca non informata

- Ricerca in Ampiezza
- Ricerca in Profondità



**frontieraLIFO** = [F8, D7, C8]  
esplorati = [D8]



# Ricerca informata

- Ricerca A\*





# Ricerca informata

- Ricerca A\*

A6

stato iniziale

$g(n) = 7$

D8

$f(D8) = g(D8) + h(D8)$   
 $= 7 + 7 = 14$

$h(n) = d(D8, M8) = 7$

M8

nodo obiettivo

$f(n) = g(n) + h(n)$

$h(n) = d(n, \text{nodoObiettivo})$

$g(n)$  rappresenta il costo di  
cammino dallo stato iniziale ad  $n$

$d$  rappresenta la distanza euclidea



# MazeSolver

Nicola Cappello



Grazie per l'attenzione