MazeSolver

Nicola Cappello

Introduzione

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

Introduzione

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

Introduzione

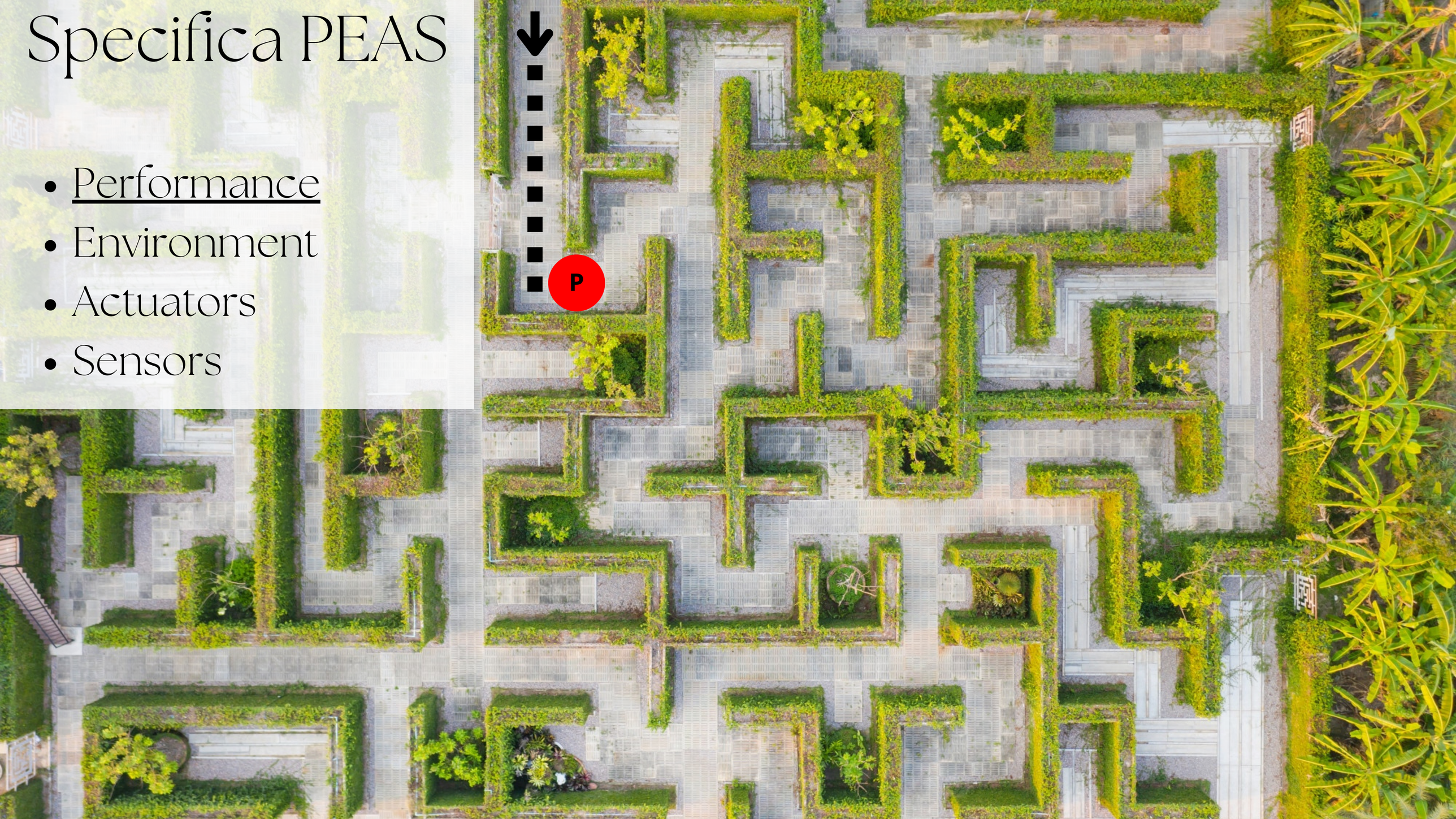
- Cos'è un labirinto?
- Cos'è MazeSolver?

Specifica PEAS

- Performance
- Environment
- Actuators
- Sensors

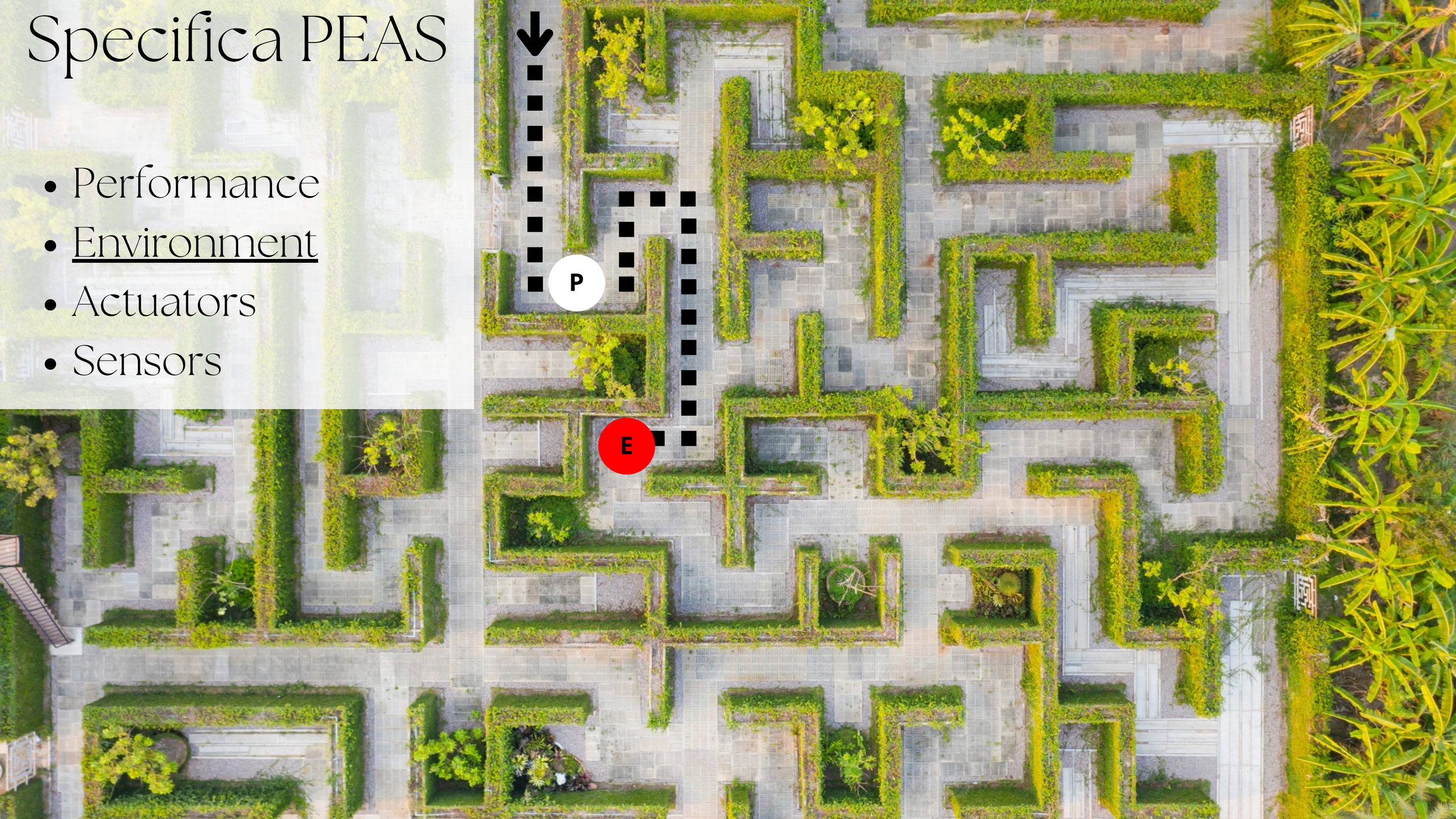
Specifica PEAS

- <u>Performance</u>
- Environment
- Actuators
- Sensors



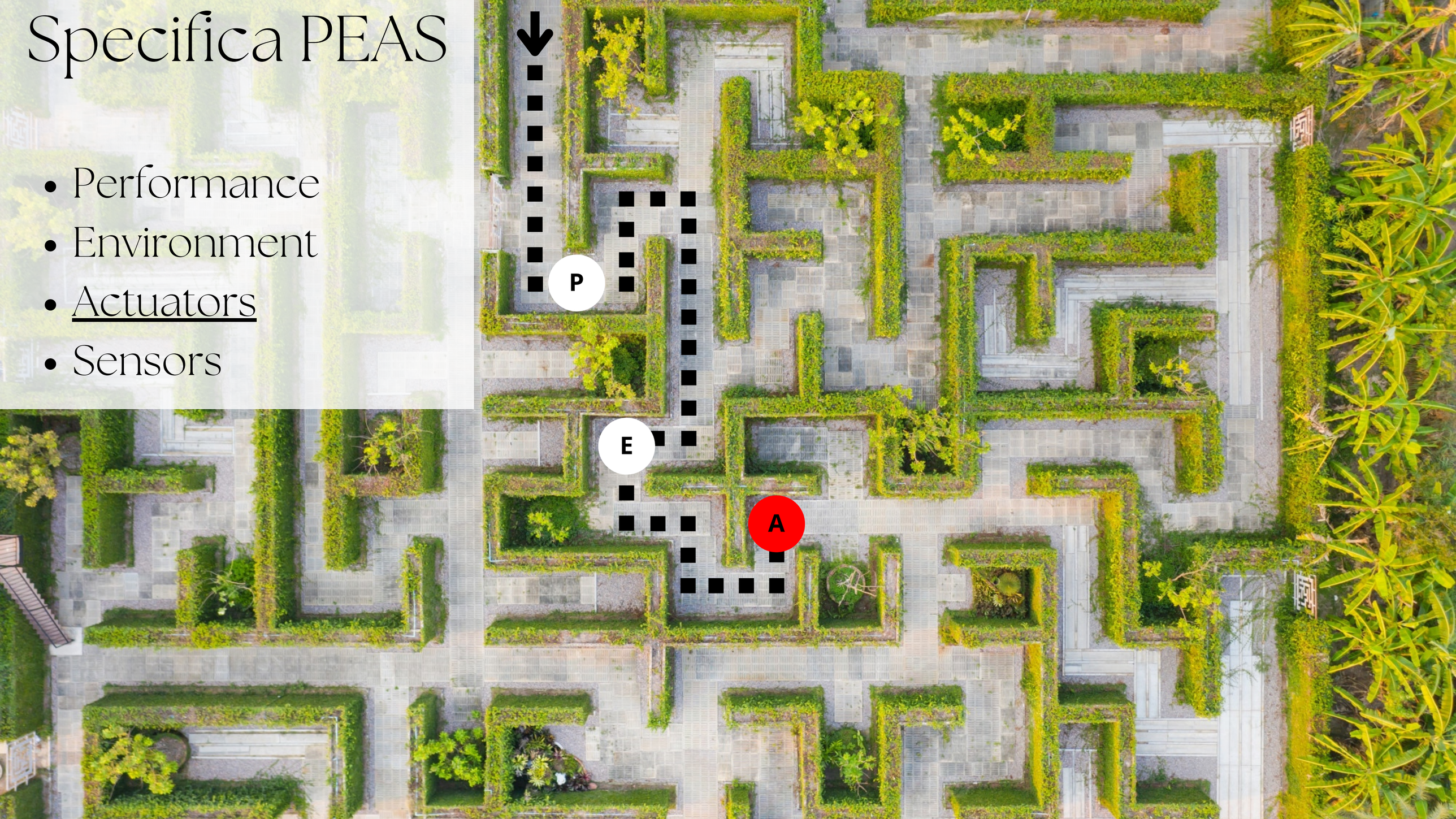
Specifica PEAS

- Performance
- <u>Environment</u>
- Actuators
- Sensors



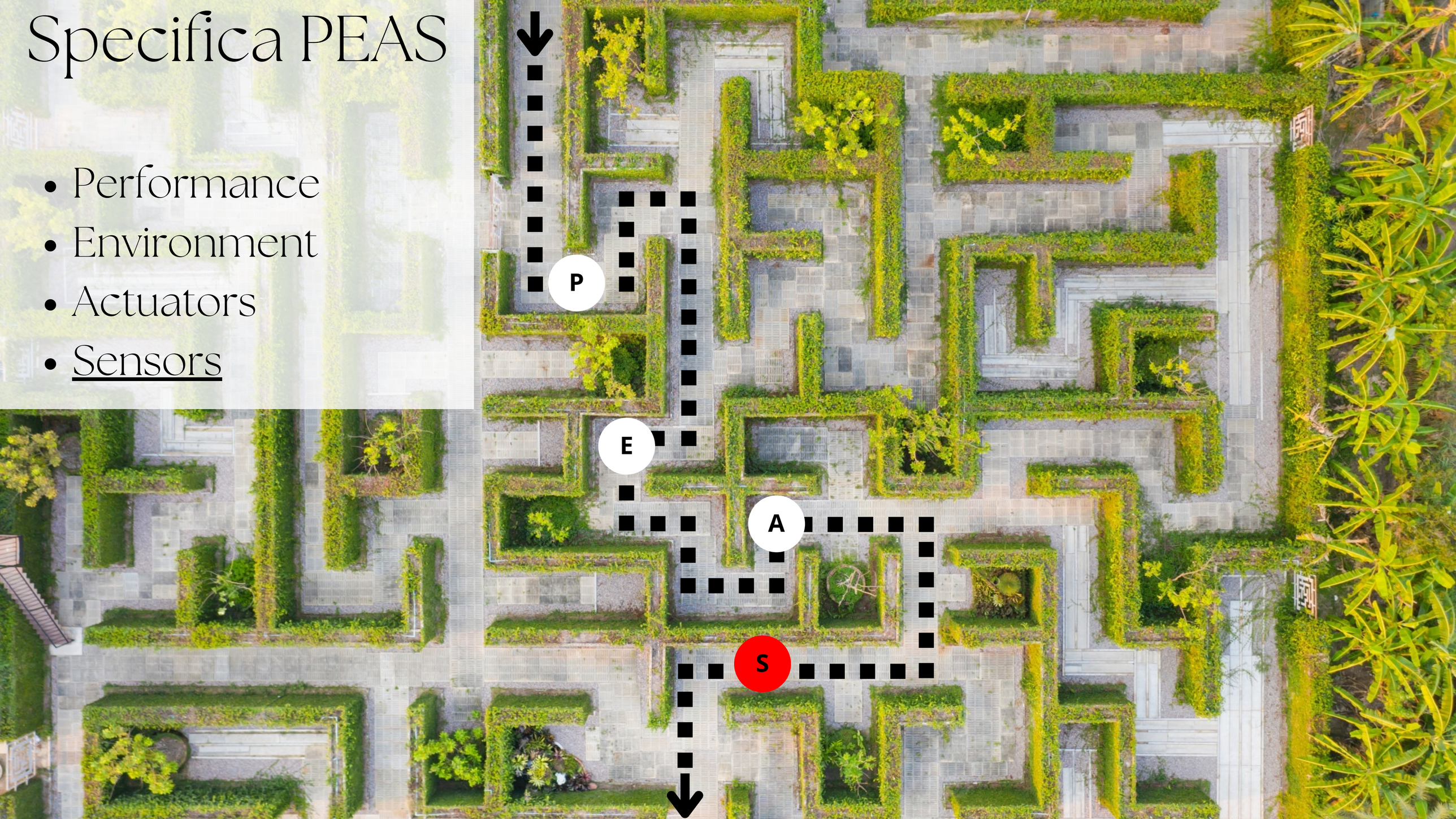
Specifica PEAS

- Performance
- Environment
- <u>Actuators</u>
- Sensors



Specifica PEAS

- Performance
- Environment
- Actuators
- <u>Sensors</u>



Specifiche Ambiente

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

Specifiche Ambiente

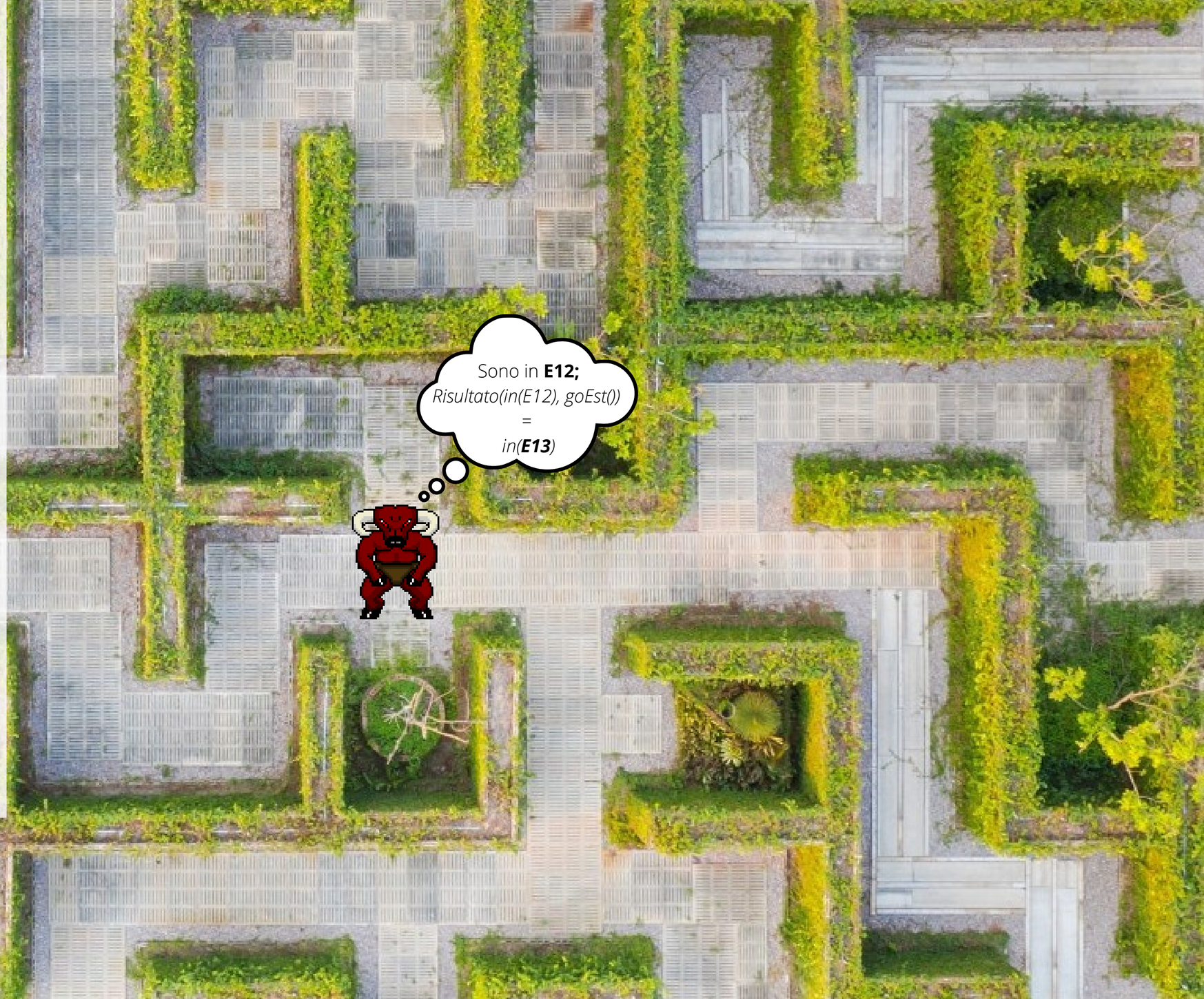
- Singolo agente
- Completamente osservabile
- Deterministico
- Statico
- Sequenziale
- Discreto

Specifiche Ambiente

- Singolo agente
- <u>Completamente osservabile</u>
- Deterministico
- Statico
- Sequenziale
- Discreto

Specifiche Ambiente

- Singolo agente
- Completamente osservabile
- <u>Deterministico</u>
- Statico
- Sequenziale
- Discreto

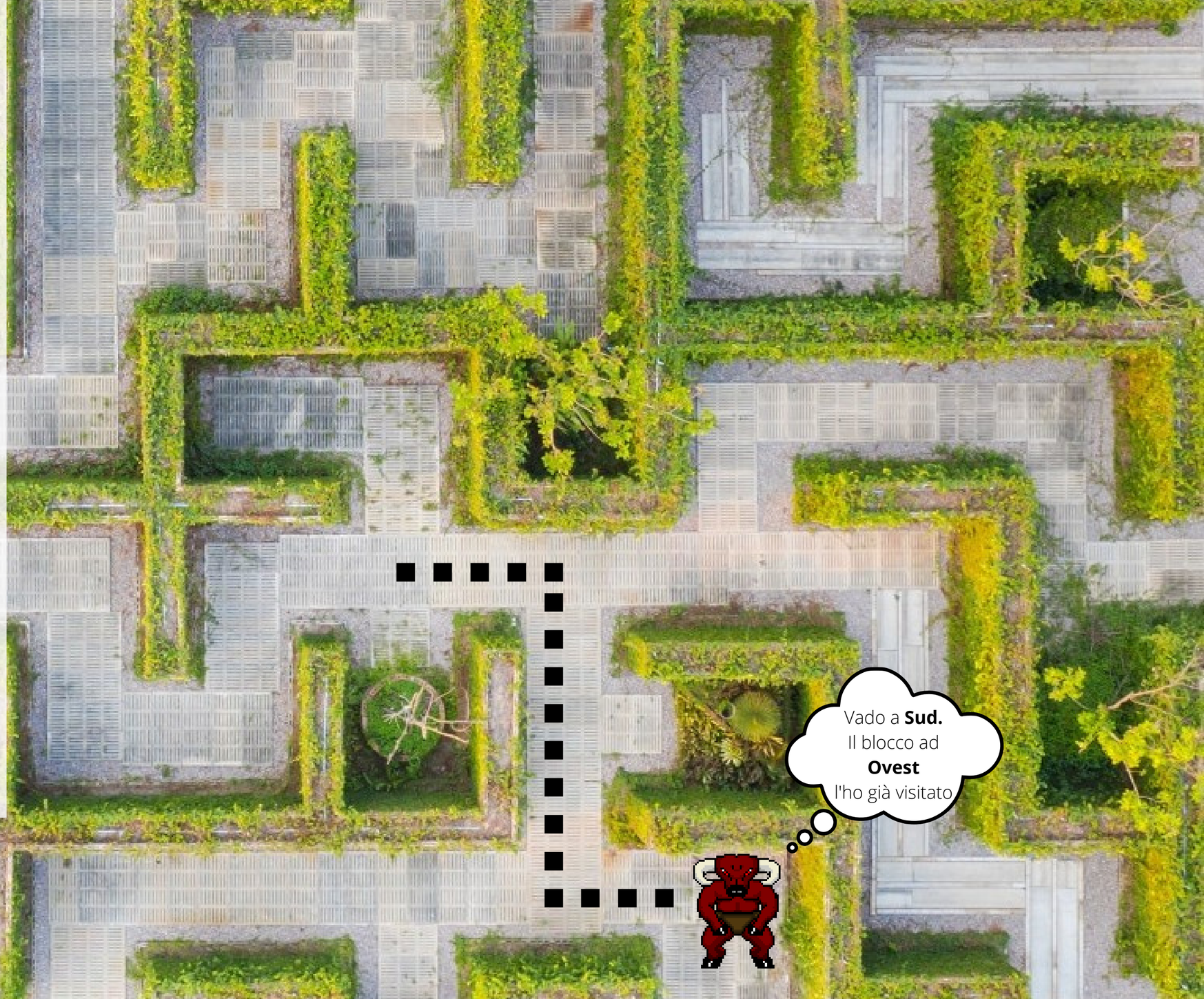


Specifiche Ambiente

- Singolo agente
- Completamente osservabile
- Deterministico
- <u>Statico</u>
- Sequenziale
- Discreto

Specifiche Ambiente

- Singolo agente
- Completamente osservabile
- Deterministico
- Statico
- <u>Sequenziale</u>
- Discreto



Specifiche Ambiente

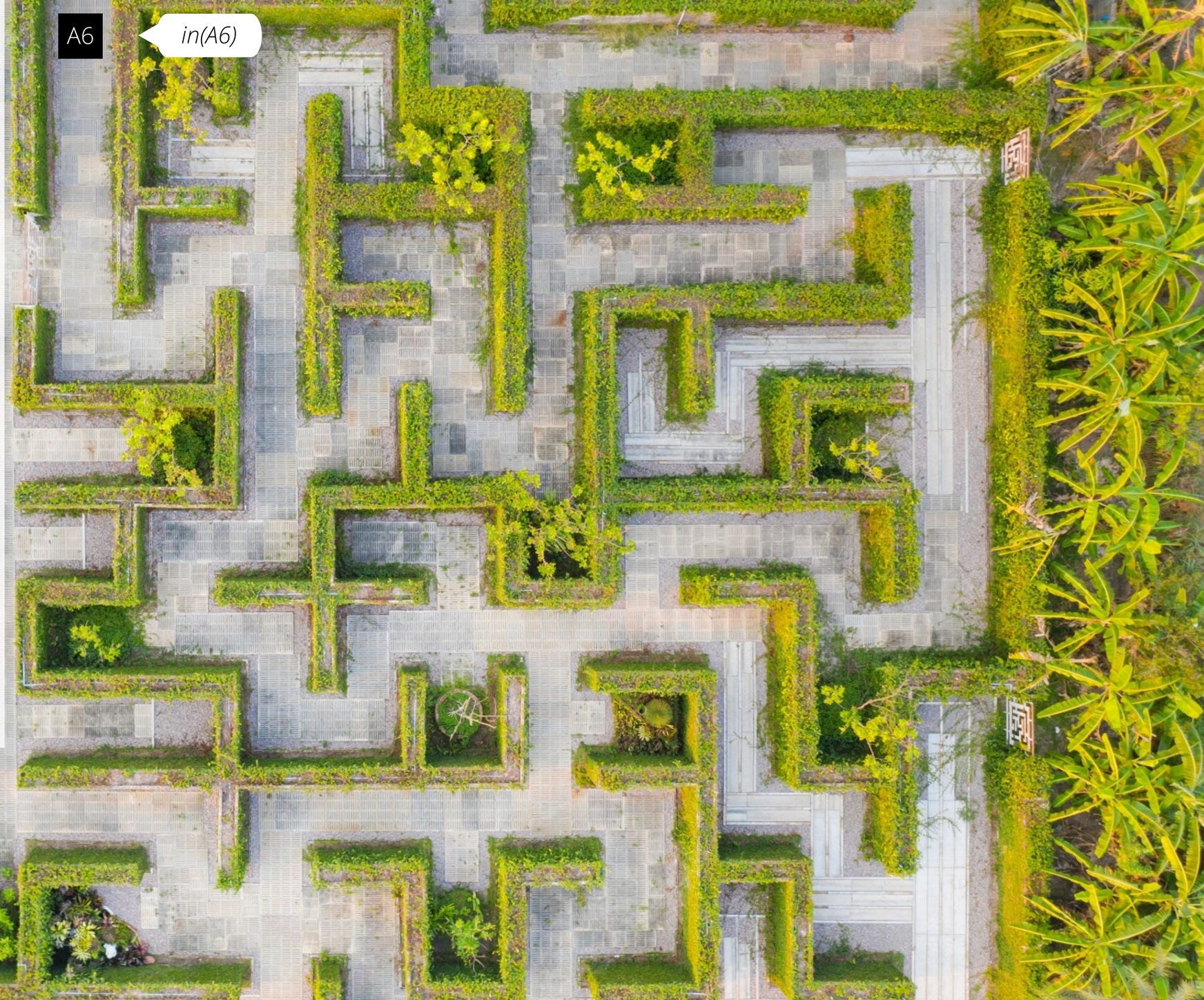
- Singolo agente
- Completamente osservabile
- Deterministico
- Statico
- Sequenziale
- <u>Discreto</u>

Analisi del problema

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

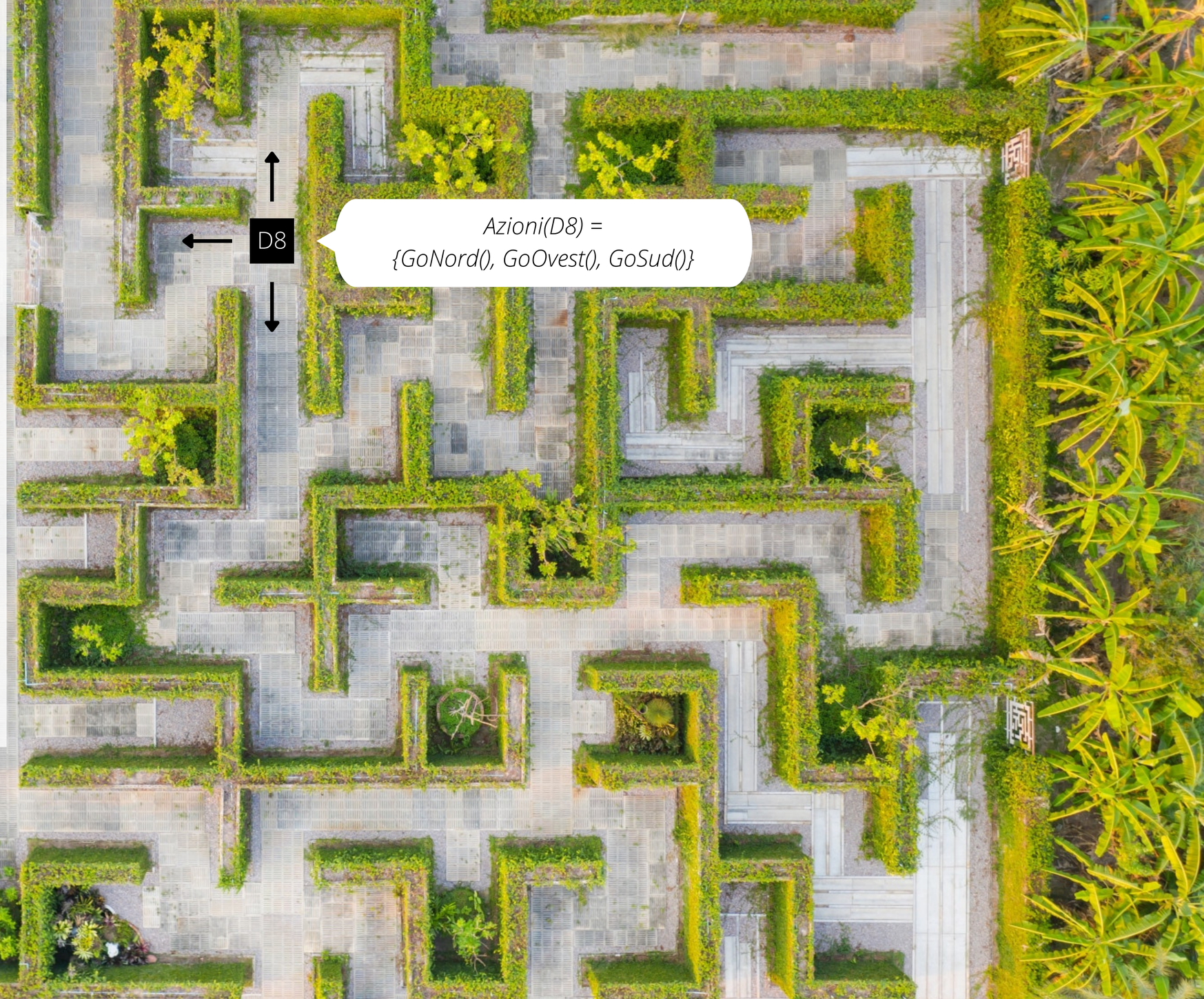
Analisi del problema

- <u>Stato iniziale</u>
- Azioni
- Modello di transizione
- Test obiettivo
- Costo di cammino



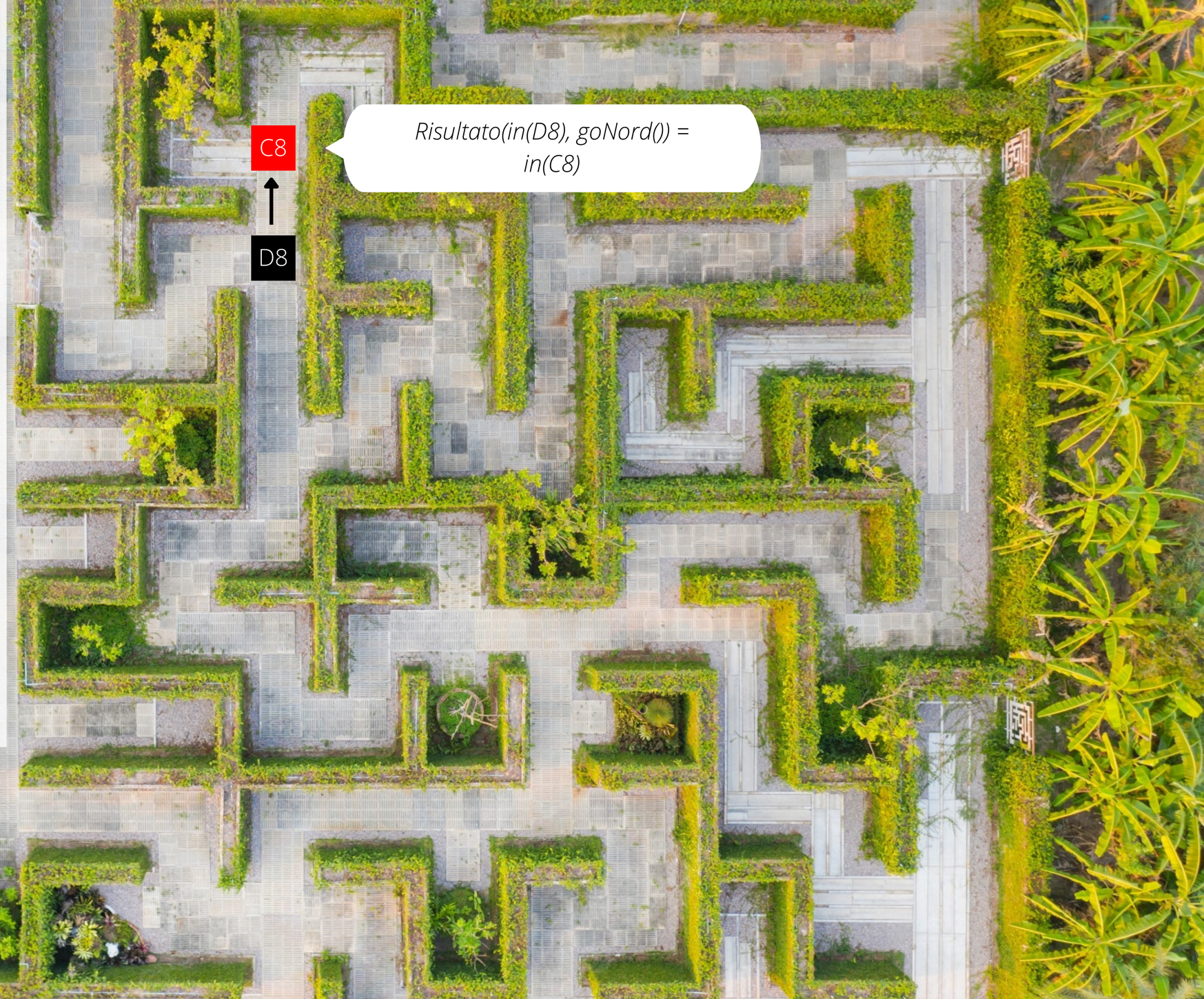
Analisi del problema

- Stato iniziale
- <u>Azioni</u>
- Modello di transizione
- Test obiettivo
- Costo di cammino



Analisi del problema

- Stato iniziale
- Azioni
- <u>Modello di transizione</u>
- Test obiettivo
- Costo di cammino

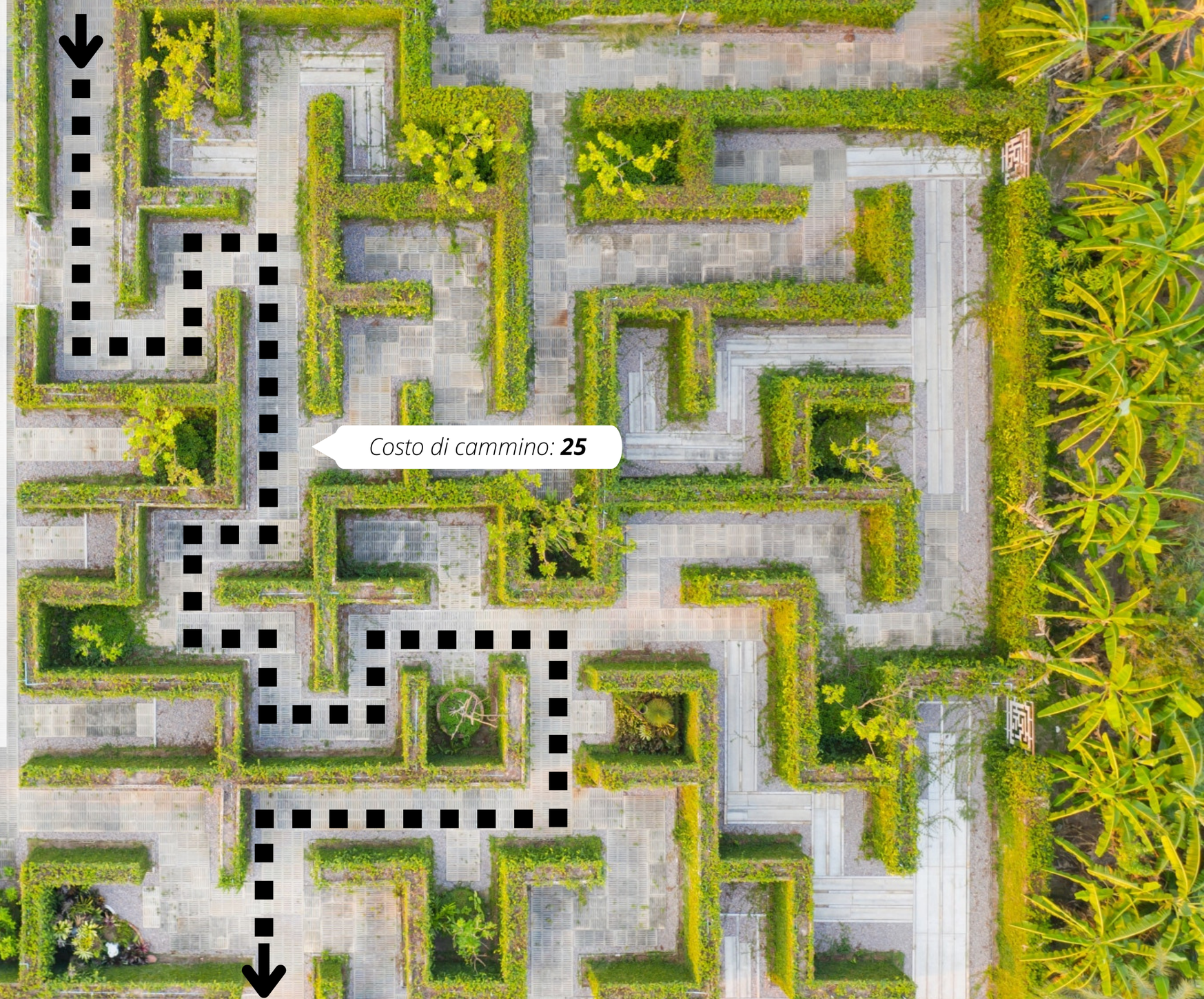


Analisi del problema

- Stato iniziale
- Azioni
- Modello di transizione
- <u>Test obiettivo</u>
- Costo di cammino

Analisi del problema

- Stato iniziale
- Azioni
- Modello di transizione
- Test obiettivo
- <u>Costo di cammino</u>



Ricerca non informata

- Ricerca in Ampiezza
- Ricerca in Profondità
- Ricerca a Costo Uniforme

Ricerca non informata

- <u>Ricerca in Ampiezza</u>
- Ricerca in Profondità
- Ricerca a Costo Uniforme

```
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
- Ricerca a Costo Uniforme

```
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

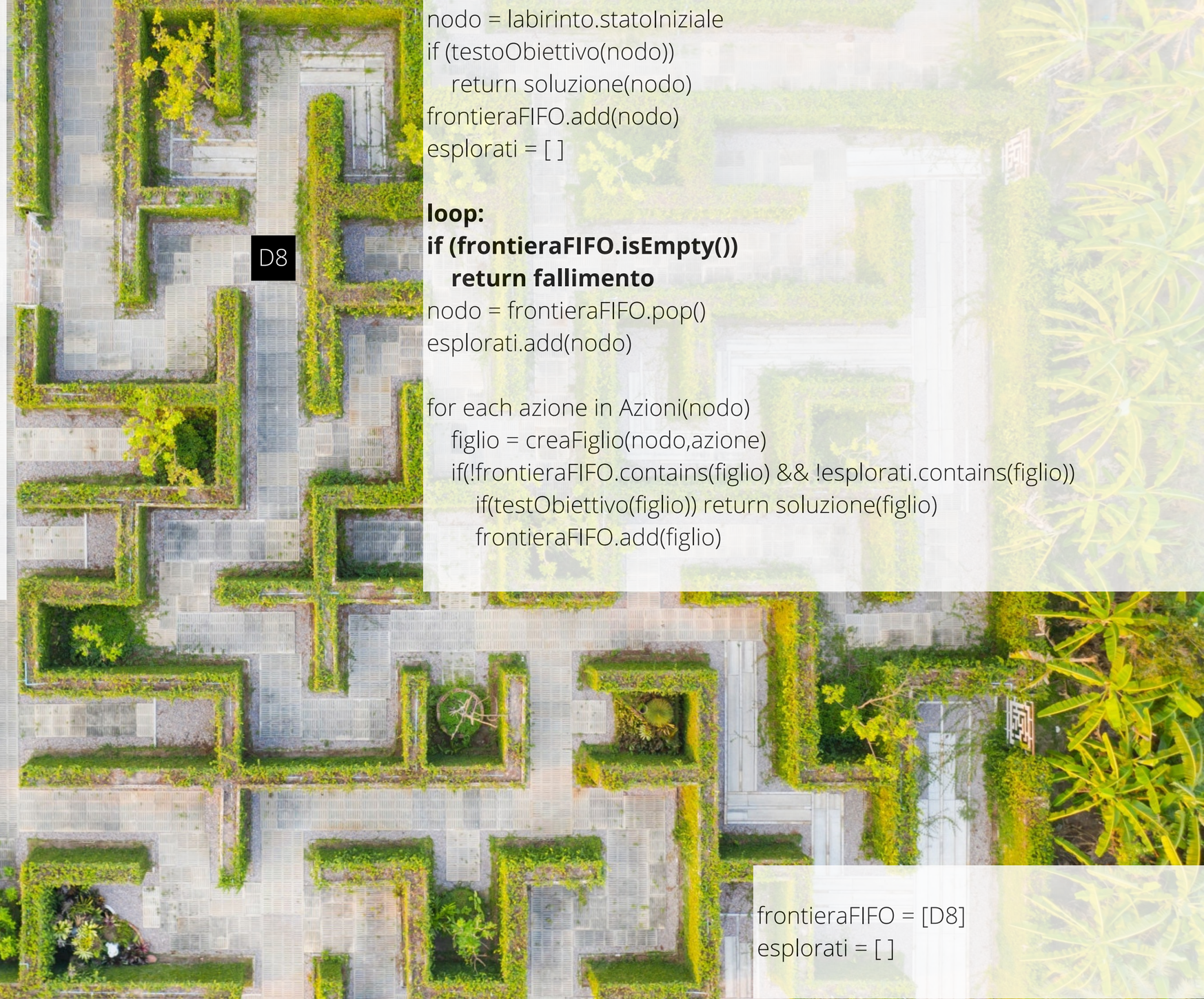
- <u>Ricerca in Ampiezza</u>
- Ricerca in Profondità
- Ricerca a Costo Uniforme

```
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

- <u>Ricerca in Ampiezza</u>
- Ricerca in Profondità
- Ricerca a Costo Uniforme




```
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

- <u>Ricerca in Ampiezza</u>
- Ricerca in Profondità
- Ricerca a Costo Uniforme

```
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
- Ricerca a Costo Uniforme

```
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

- <u>Ricerca in Ampiezza</u>
- Ricerca in Profondità
- Ricerca a Costo Uniforme

```
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

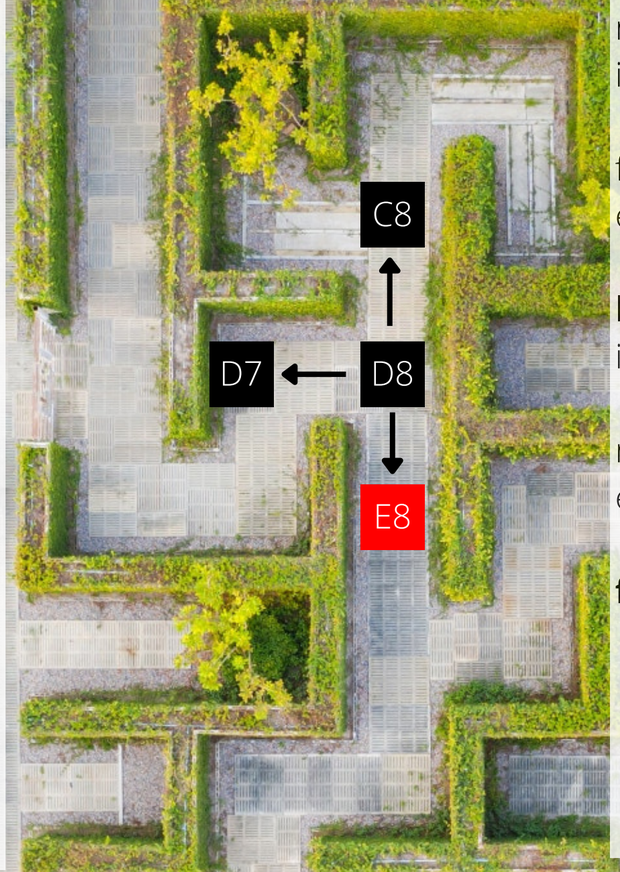
- <u>Ricerca in Ampiezza</u>
- Ricerca in Profondità
- Ricerca a Costo Uniforme

```
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

- <u>Ricerca in Ampiezza</u>
- Ricerca in Profondità
- Ricerca a Costo Uniforme

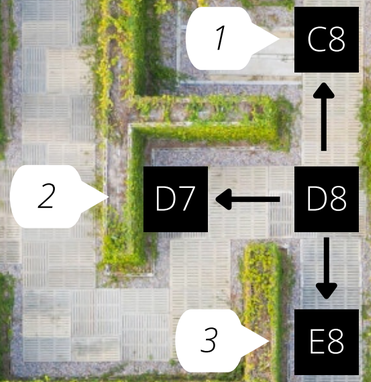



```
nodo = labirinto.statoIniziale
if (testoObiettivo(nodo))
    return soluzione(nodo)
frontieraFIFO.add(nodo)
esplorati = [ ]

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

- <u>Ricerca in Ampiezza</u>
- Ricerca in Profondità
- Ricerca a Costo Uniforme




```
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]

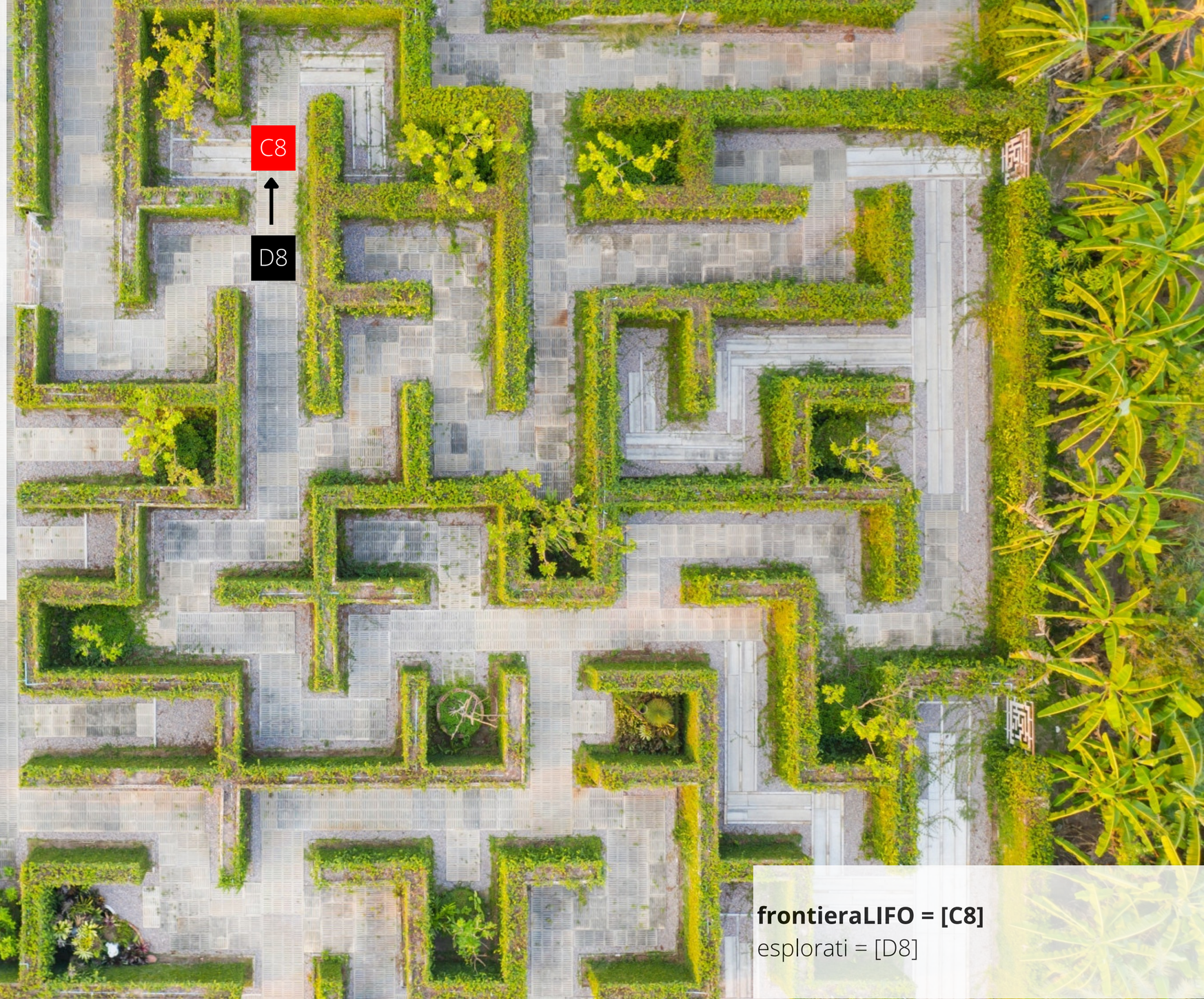
Ricerca non informata

- Ricerca in Ampiezza
- <u>Ricerca in Profondità</u>
- Ricerca a Costo Uniforme

```
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
- <u>Ricerca in Profondità</u>
- Ricerca a Costo Uniforme



frontieraLIFO = [C8]
esplorati = [D8]

Ricerca non informata

- Ricerca in Ampiezza
- <u>Ricerca in Profondità</u>
- Ricerca a Costo Uniforme

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

Ricerca non informata

- Ricerca in Ampiezza
- <u>Ricerca in Profondità</u>
- Ricerca a Costo Uniforme

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

Ricerca non informata

- Ricerca in Ampiezza
- <u>Ricerca in Profondità</u>
- Ricerca a Costo Uniforme

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

Ricerca non informata

- Ricerca in Ampiezza
- <u>Ricerca in Profondità</u>
- Ricerca a Costo Uniforme

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

Ricerca non informata

- Ricerca in Ampiezza
- Ricerca in Profondità
- <u>Ricerca a Costo Uniforme</u>

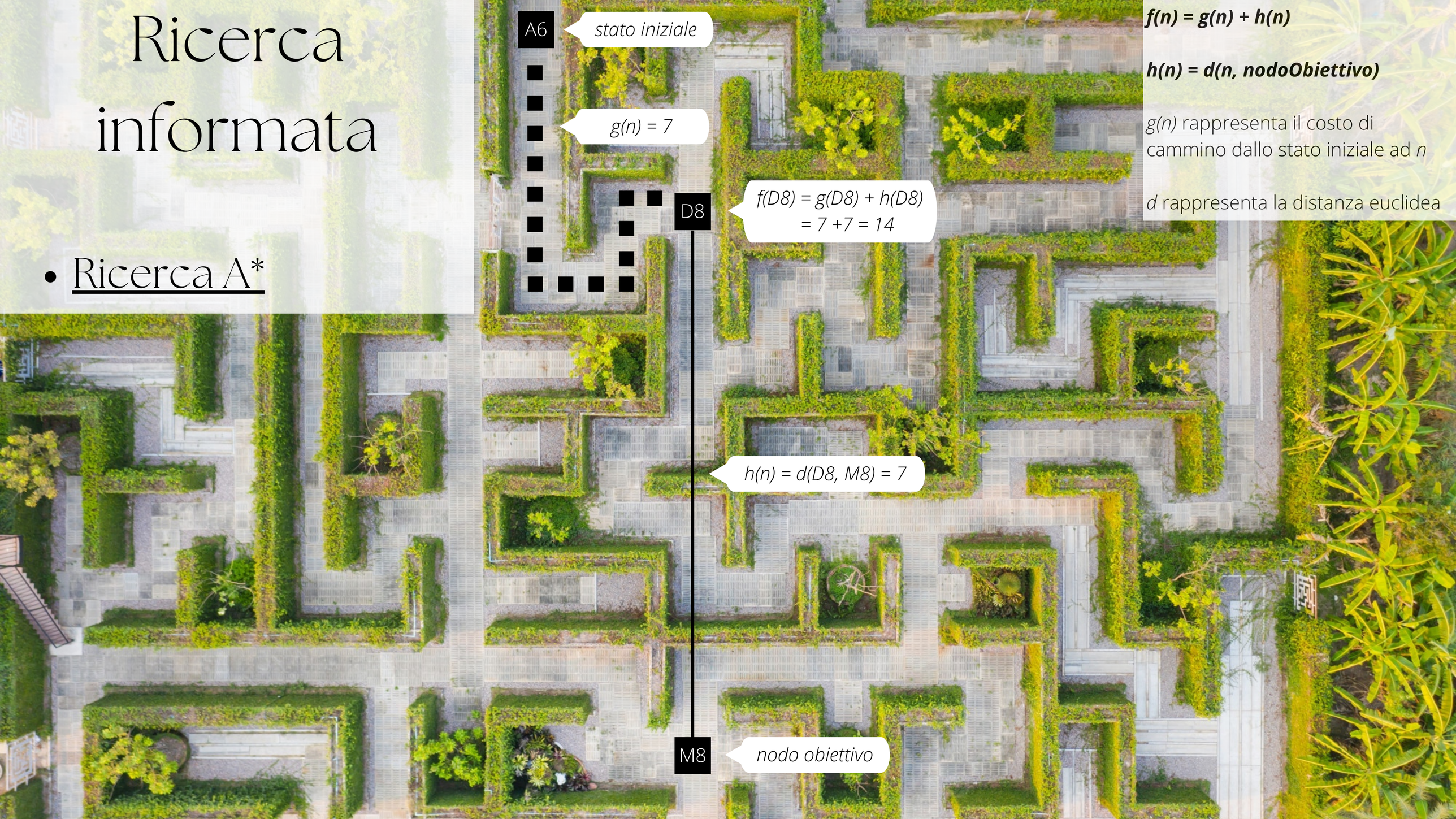
```
nodo = labirinto.statoIniziale
nodo.g = 0
frontieraPQGN.add(nodo)
esplorati = [ ]

loop:
if (frontieraPQGN.isEmpty())
   return fallimento
nodo = frontieraPQGN.pop()
if(testObiettivo(nodo)) return soluzione(nodo)
esplorati.add(nodo)

for each azione in Azioni(nodo)
   figlio = creaFiglio(nodo,azione)
   g(figlio) = nodo.g + 1;
   if(!frontieraPQGN.contains(figlio) && !esplorati.contains(figlio))
      frontieraPQGN.add(figlio)
      figlio.g = g(figlio)
   else if (frontieraPQGN.contains(figlio) && figlio.g > g(figlio))
      figlio.g = g(figlio)
```

Ricerca informata

- Ricerca A*

Ricerca informata

- Ricerca A*



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

h(n) = d(n, 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