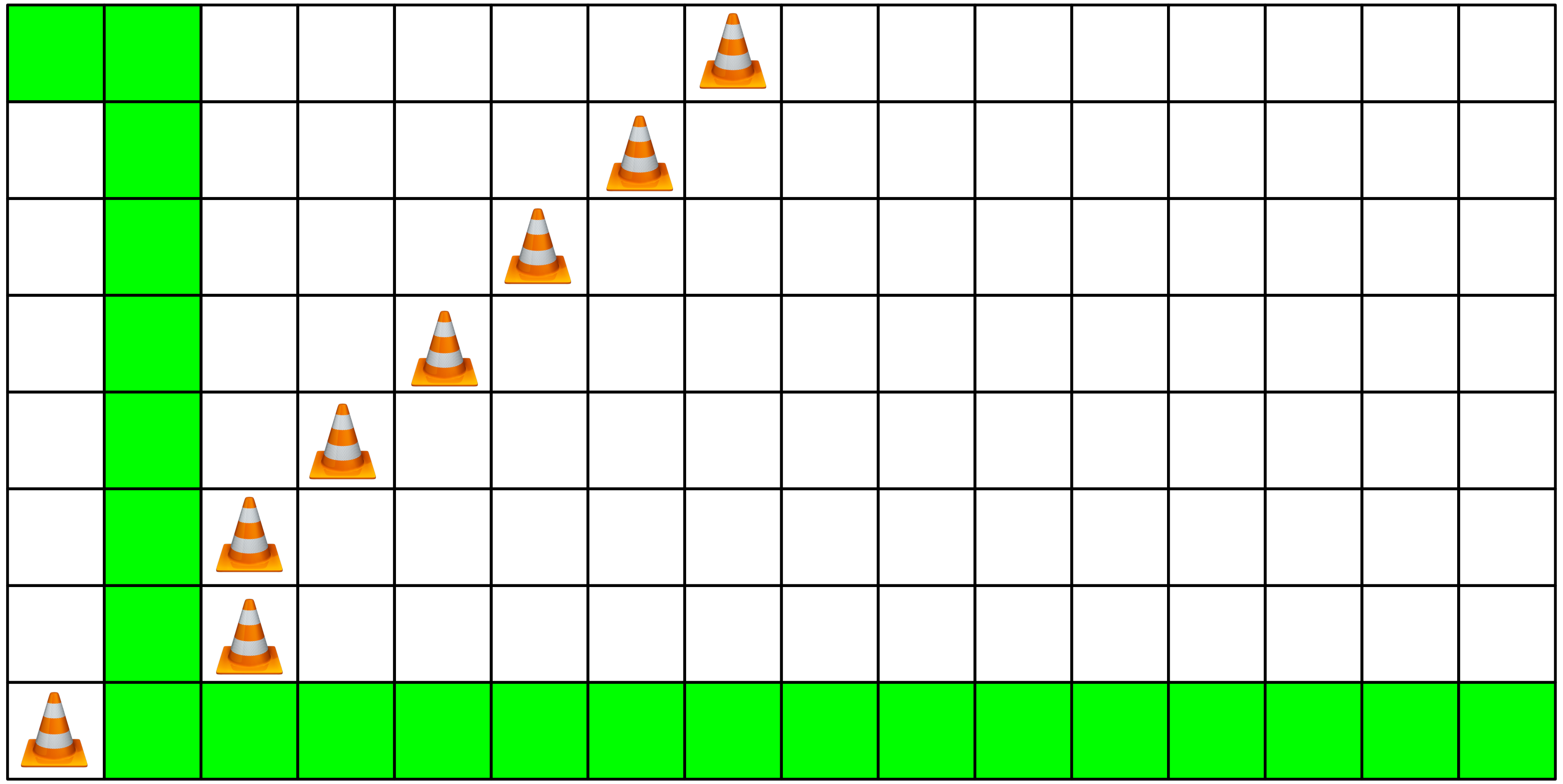
The MOR problem

Minimum Obstacle Removal to reach the Corner

Giordano Colli

Giovanna Melideo



L'Aquila, 20/12/2024

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

DISIM Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica

Minimum Obstacle Removal (Descrizione Informale)

Minimum Obstacle Removal (Descrizione Informale)

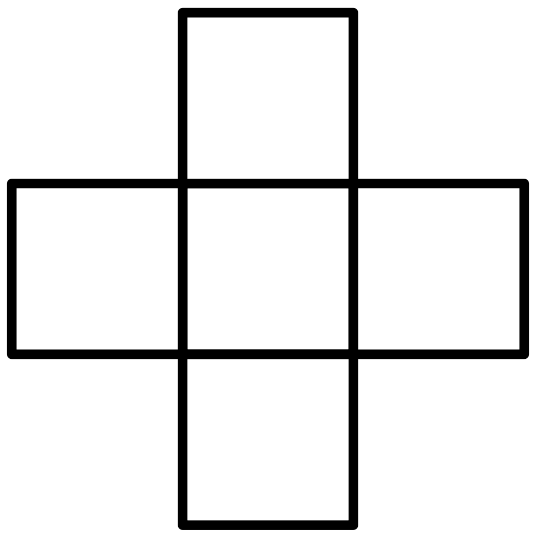
- **Input:** Una griglia $n \times m$ in cui ogni casella può contenere un ostacolo o meno.

Minimum Obstacle Removal (Descrizione Informale)

- **Input:** Una griglia $n \times m$ in cui ogni casella può contenere un ostacolo o meno.
- Partendo da una casella senza ostacoli, è possibile spostarsi sulle caselle adiacenti ai suoi lati (su, destra, giù, sinistra) **se non ci sono ostacoli**.

Minimum Obstacle Removal (Descrizione Informale)

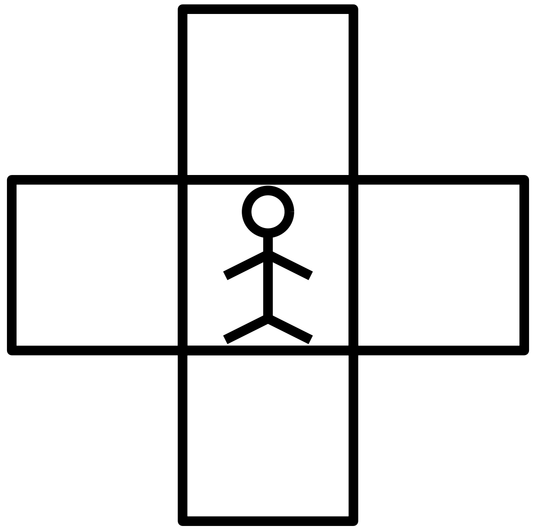
- **Input:** Una griglia $n \times m$ in cui ogni casella può contenere un ostacolo o meno.
- Partendo da una casella senza ostacoli, è possibile spostarsi sulle caselle adiacenti ai suoi lati (su, destra, giù, sinistra) **se non ci sono ostacoli**.



Minimum Obstacle Removal (Descrizione Informale)

- **Input:** Una griglia $n \times m$ in cui ogni casella può contenere un ostacolo o meno.

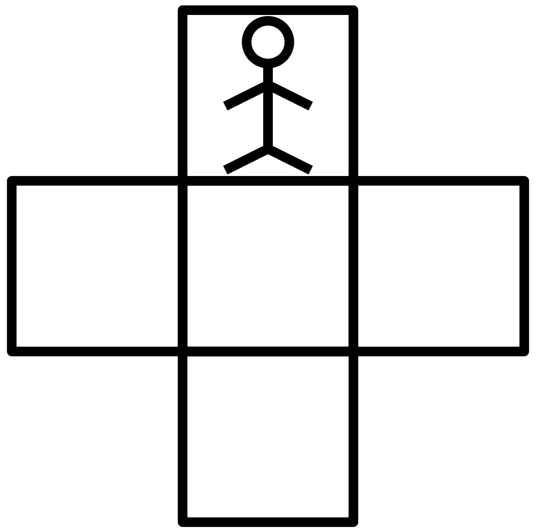
- Partendo da una casella senza ostacoli, è possibile spostarsi sulle caselle adiacenti ai suoi lati (su, destra, giù, sinistra) **se non ci sono ostacoli**.



Minimum Obstacle Removal (Descrizione Informale)

- **Input:** Una griglia $n \times m$ in cui ogni casella può contenere un ostacolo o meno.

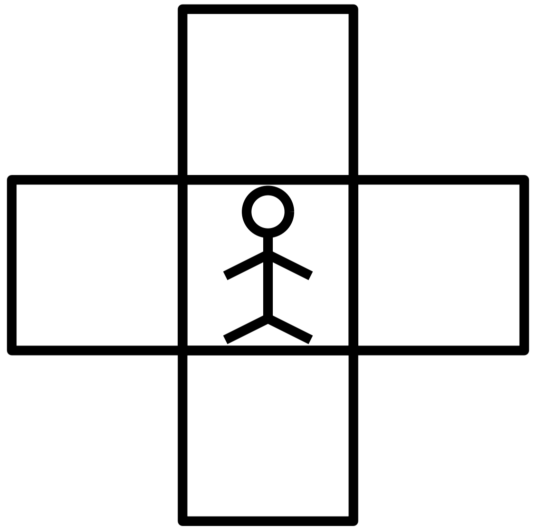
- Partendo da una casella senza ostacoli, è possibile spostarsi sulle caselle adiacenti ai suoi lati (su, destra, giù, sinistra) **se non ci sono ostacoli**.



Minimum Obstacle Removal (Descrizione Informale)

• **Input:** Una griglia $n \times m$ in cui ogni casella può contenere un ostacolo o meno.

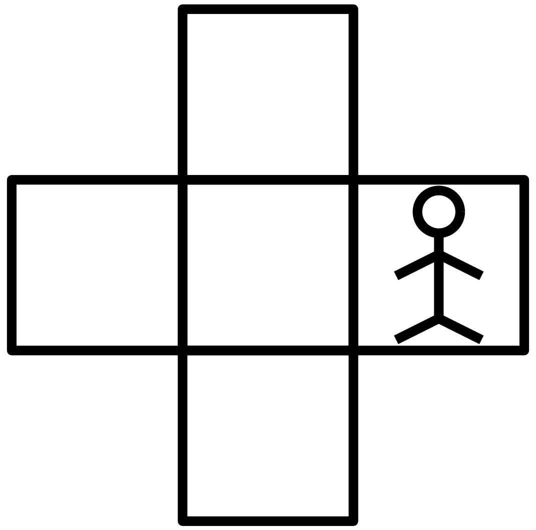
• Partendo da una casella senza ostacoli, è possibile spostarsi sulle caselle adiacenti ai suoi lati (su, destra, giù, sinistra) **se non ci sono ostacoli**.



Minimum Obstacle Removal (Descrizione Informale)

- **Input:** Una griglia $n \times m$ in cui ogni casella può contenere un ostacolo o meno.

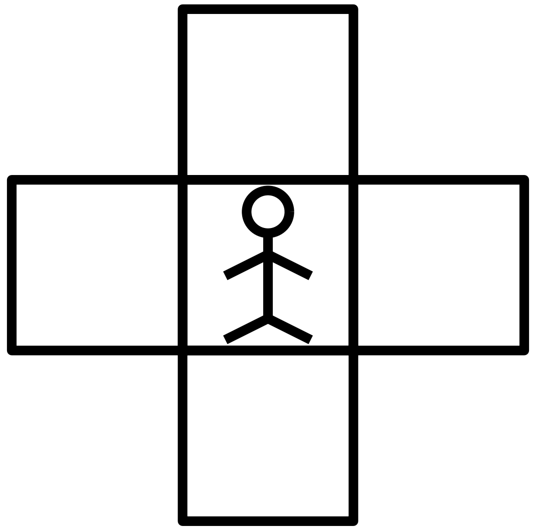
- Partendo da una casella senza ostacoli, è possibile spostarsi sulle caselle adiacenti ai suoi lati (su, destra, giù, sinistra) **se non ci sono ostacoli**.



Minimum Obstacle Removal (Descrizione Informale)

• **Input:** Una griglia $n \times m$ in cui ogni casella può contenere un ostacolo o meno.

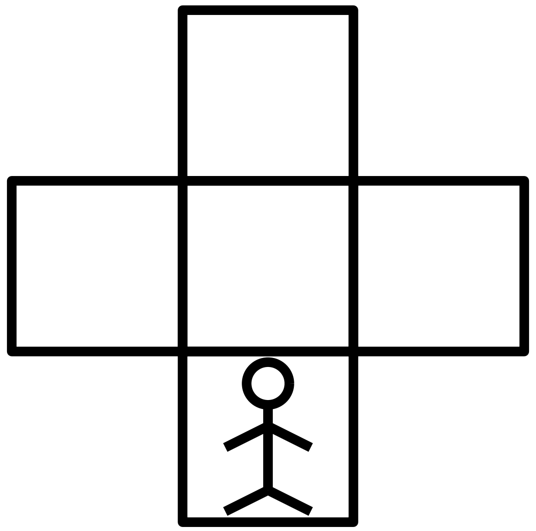
• Partendo da una casella senza ostacoli, è possibile spostarsi sulle caselle adiacenti ai suoi lati (su, destra, giù, sinistra) **se non ci sono ostacoli**.



Minimum Obstacle Removal (Descrizione Informale)

- **Input:** Una griglia $n \times m$ in cui ogni casella può contenere un ostacolo o meno.

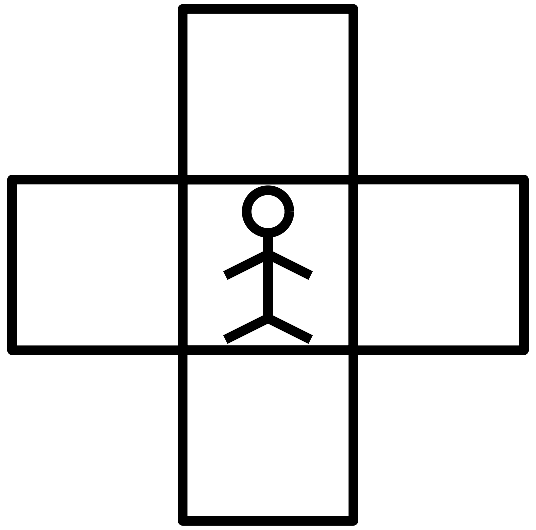
- Partendo da una casella senza ostacoli, è possibile spostarsi sulle caselle adiacenti ai suoi lati (su, destra, giù, sinistra) **se non ci sono ostacoli**.



Minimum Obstacle Removal (Descrizione Informale)

• **Input:** Una griglia $n \times m$ in cui ogni casella può contenere un ostacolo o meno.

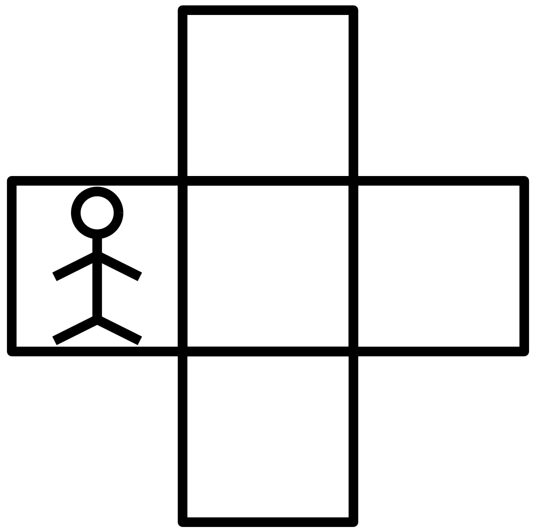
• Partendo da una casella senza ostacoli, è possibile spostarsi sulle caselle adiacenti ai suoi lati (su, destra, giù, sinistra) **se non ci sono ostacoli**.



Minimum Obstacle Removal (Descrizione Informale)

- **Input:** Una griglia $n \times m$ in cui ogni casella può contenere un ostacolo o meno.

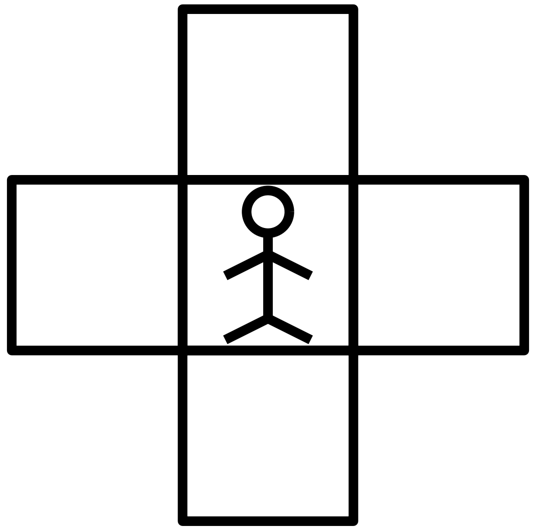
- Partendo da una casella senza ostacoli, è possibile spostarsi sulle caselle adiacenti ai suoi lati (su, destra, giù, sinistra) **se non ci sono ostacoli**.



Minimum Obstacle Removal (Descrizione Informale)

• **Input:** Una griglia $n \times m$ in cui ogni casella può contenere un ostacolo o meno.

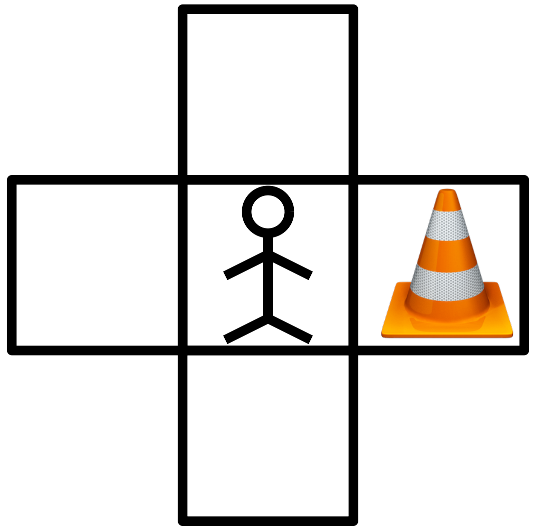
• Partendo da una casella senza ostacoli, è possibile spostarsi sulle caselle adiacenti ai suoi lati (su, destra, giù, sinistra) **se non ci sono ostacoli**.



Minimum Obstacle Removal (Descrizione Informale)

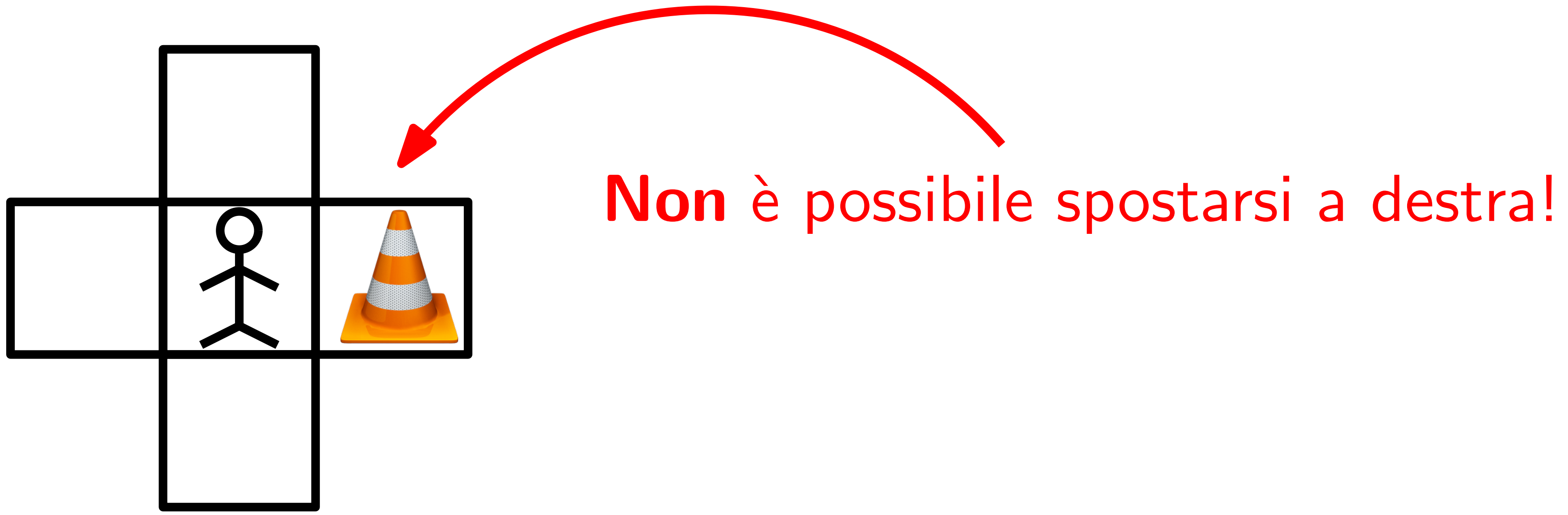
- **Input:** Una griglia $n \times m$ in cui ogni casella può contenere un ostacolo o meno.

- Partendo da una casella senza ostacoli, è possibile spostarsi sulle caselle adiacenti ai suoi lati (su, destra, giù, sinistra) **se non ci sono ostacoli**.



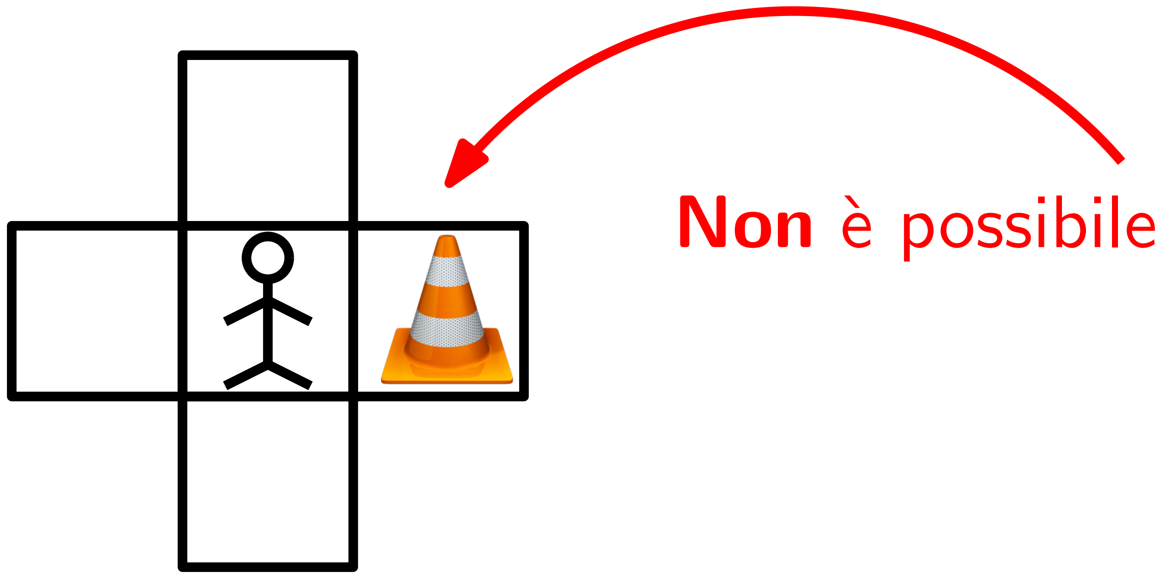
Minimum Obstacle Removal (Descrizione Informale)

- **Input:** Una griglia $n \times m$ in cui ogni casella può contenere un ostacolo o meno.
- Partendo da una casella senza ostacoli, è possibile spostarsi sulle caselle adiacenti ai suoi lati (su, destra, giù, sinistra) **se non ci sono ostacoli**.

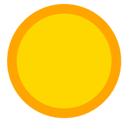


Minimum Obstacle Removal (Descrizione Informale)

- **Input:** Una griglia $n \times m$ in cui ogni casella può contenere un ostacolo o meno.
- Partendo da una casella senza ostacoli, è possibile spostarsi sulle caselle adiacenti ai suoi lati (su, destra, giù, sinistra) **se non ci sono ostacoli**.



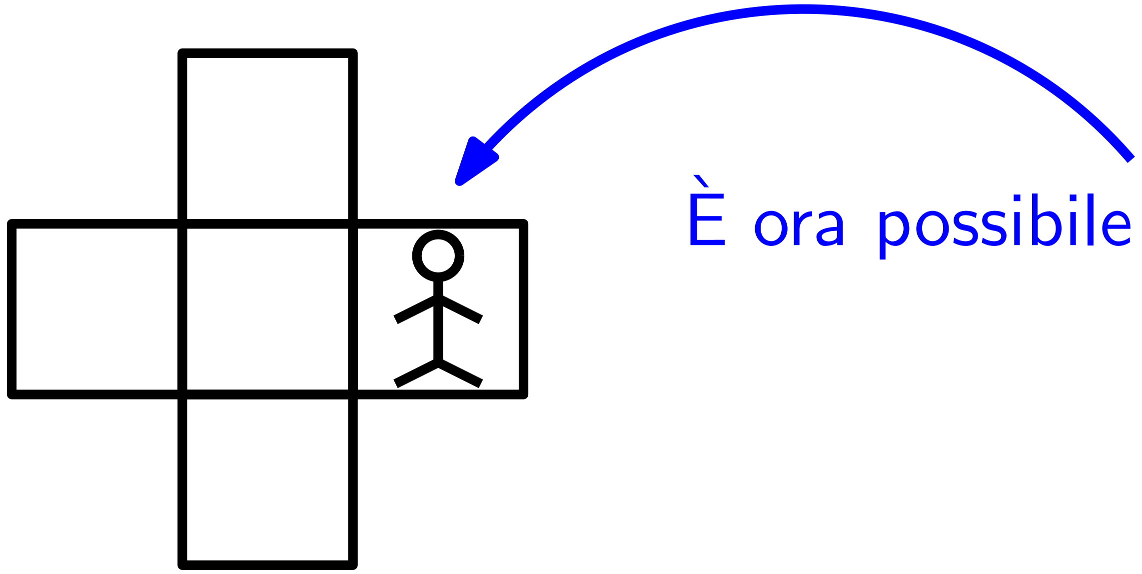
Non è possibile spostarsi a destra!

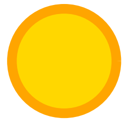
- È possibile rimuovere un ostacolo pagando una moneta. 

Minimum Obstacle Removal (Descrizione Informale)

- **Input:** Una griglia $n \times m$ in cui ogni casella può contenere un ostacolo o meno.

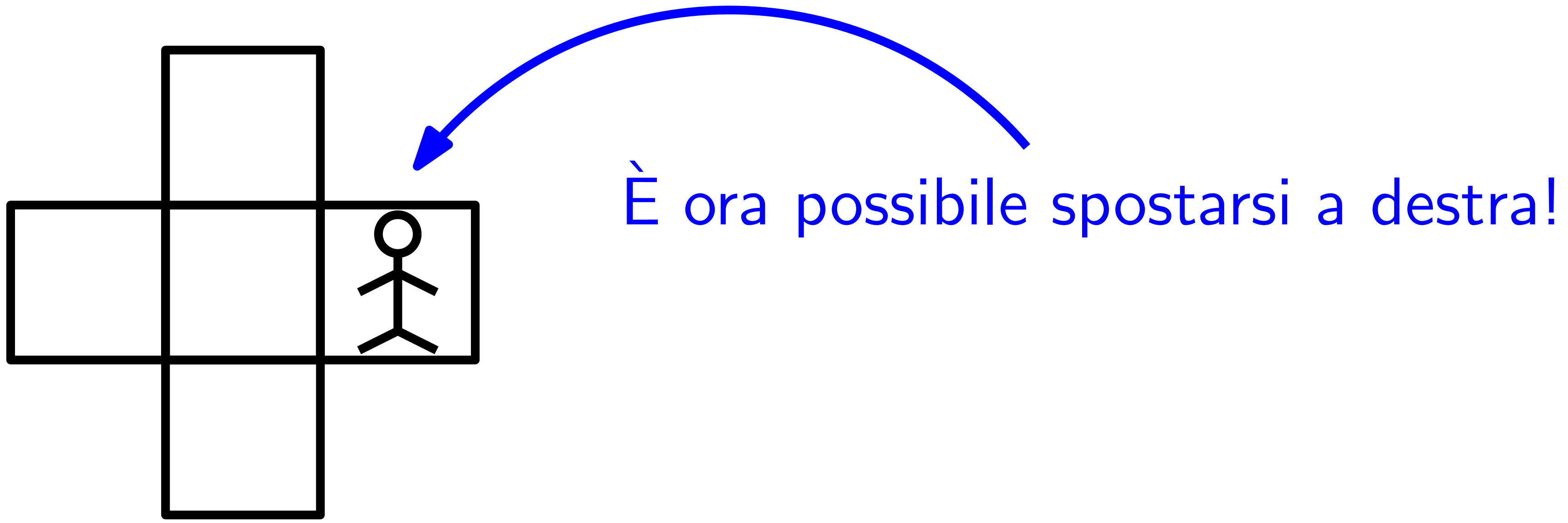
- Partendo da una casella senza ostacoli, è possibile spostarsi sulle caselle adiacenti ai suoi lati (su, destra, giù, sinistra) **se non ci sono ostacoli**.

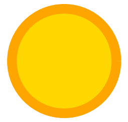


- È possibile rimuovere un ostacolo pagando una moneta. 
- Rendendo **percorribile** la casella da cui l'ostacolo è stato rimosso.

Minimum Obstacle Removal (Descrizione Informale)

- **Input:** Una griglia $n \times m$ in cui ogni casella può contenere un ostacolo o meno.
- Partendo da una casella senza ostacoli, è possibile spostarsi sulle caselle adiacenti ai suoi lati (su, destra, giù, sinistra) **se non ci sono ostacoli**.



- È possibile rimuovere un ostacolo pagando una moneta. 
- Rendendo **percorribile** la casella da cui l'ostacolo è stato rimosso.
- **Output:** Il minimo numero di ostacoli da rimuovere (equivalentemente monete da spendere) che rendono possibile raggiungere la casella (n, m) partendo dalla casella $(1, 1)$.

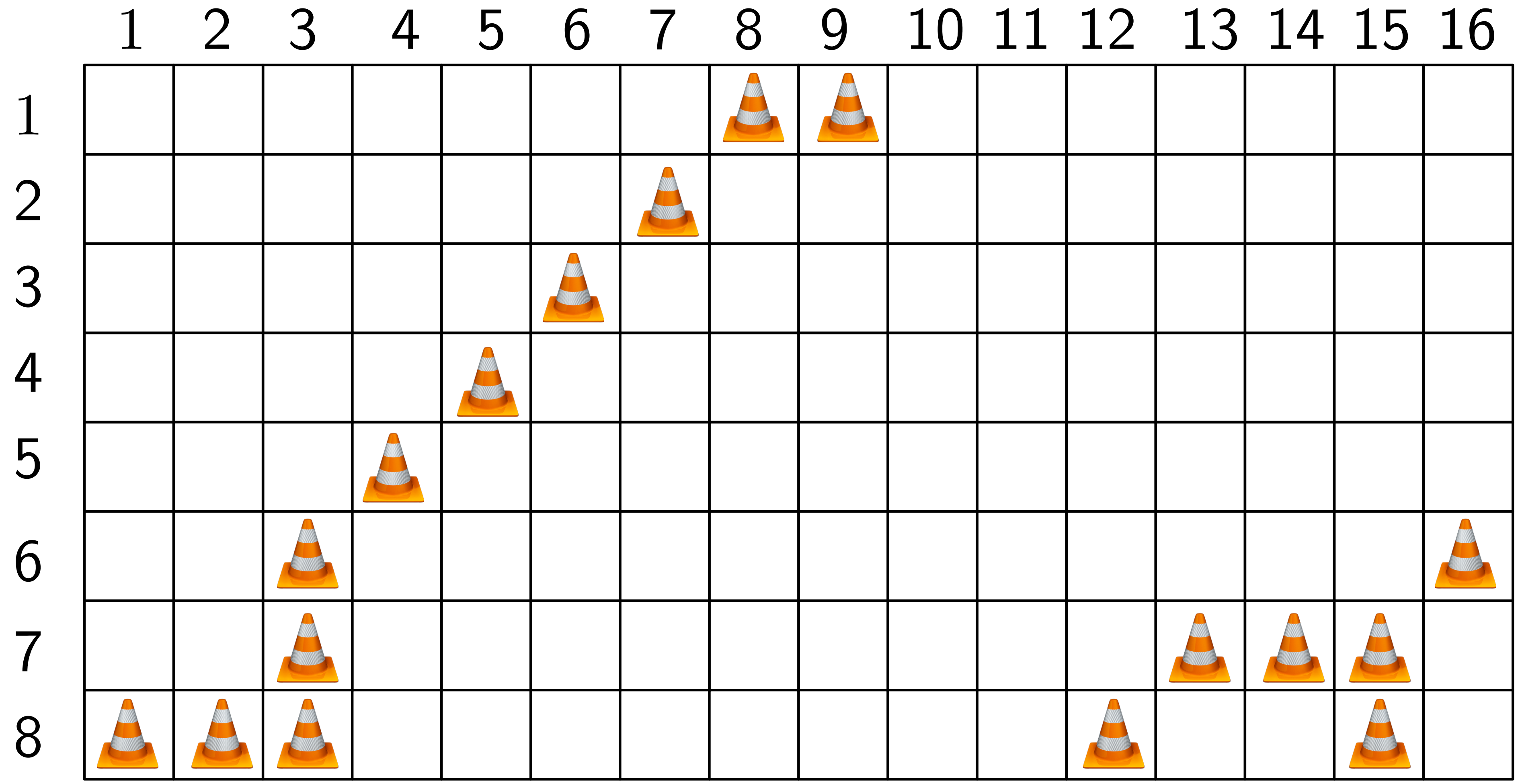
Minimum Obstacle Removal (Esempio)

Minimum Obstacle Removal (Esempio)

- $n = 8, m = 16$

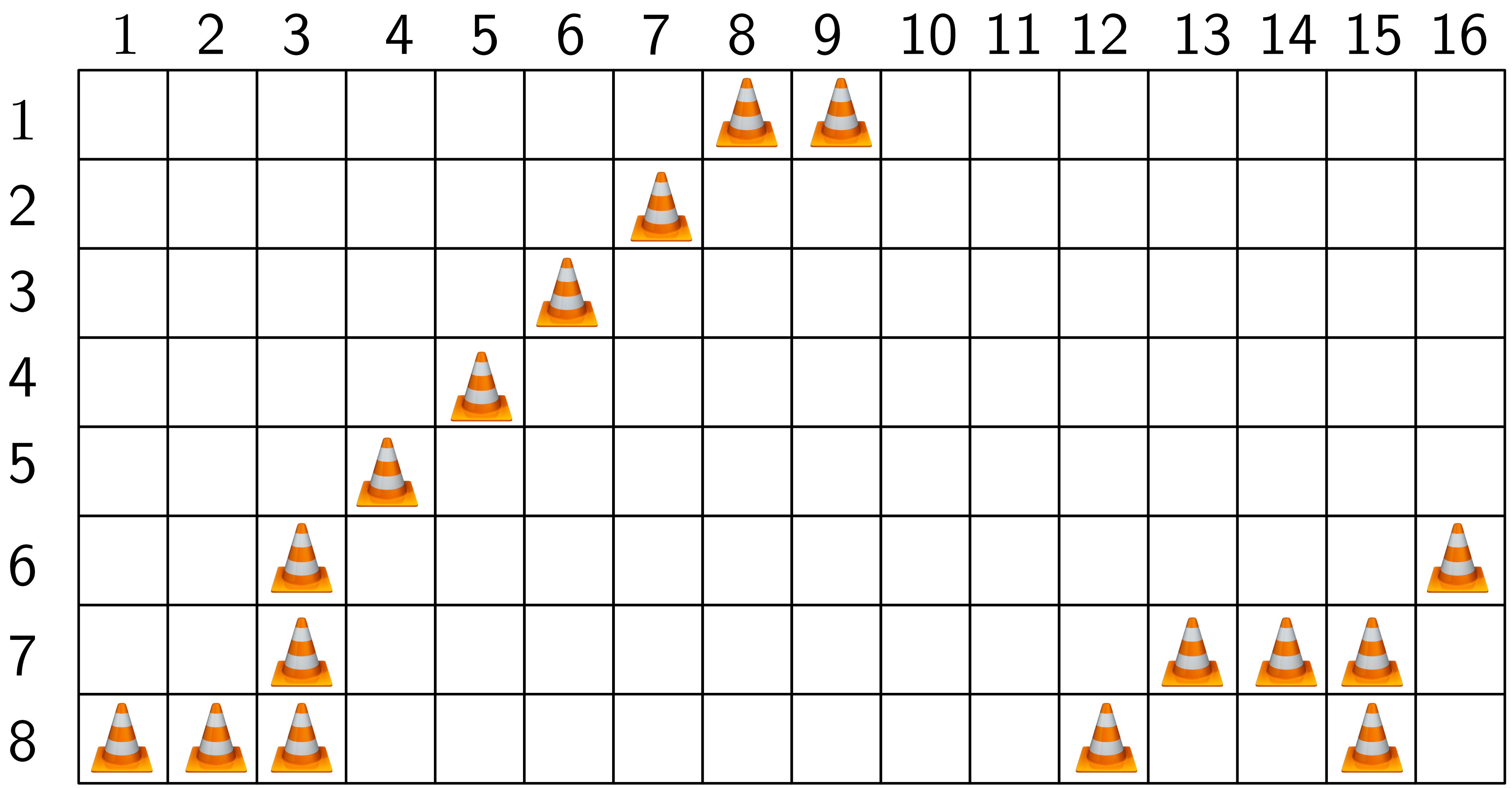
Minimum Obstacle Removal (Esempio)

- $n = 8, m = 16$



Minimum Obstacle Removal (Esempio)

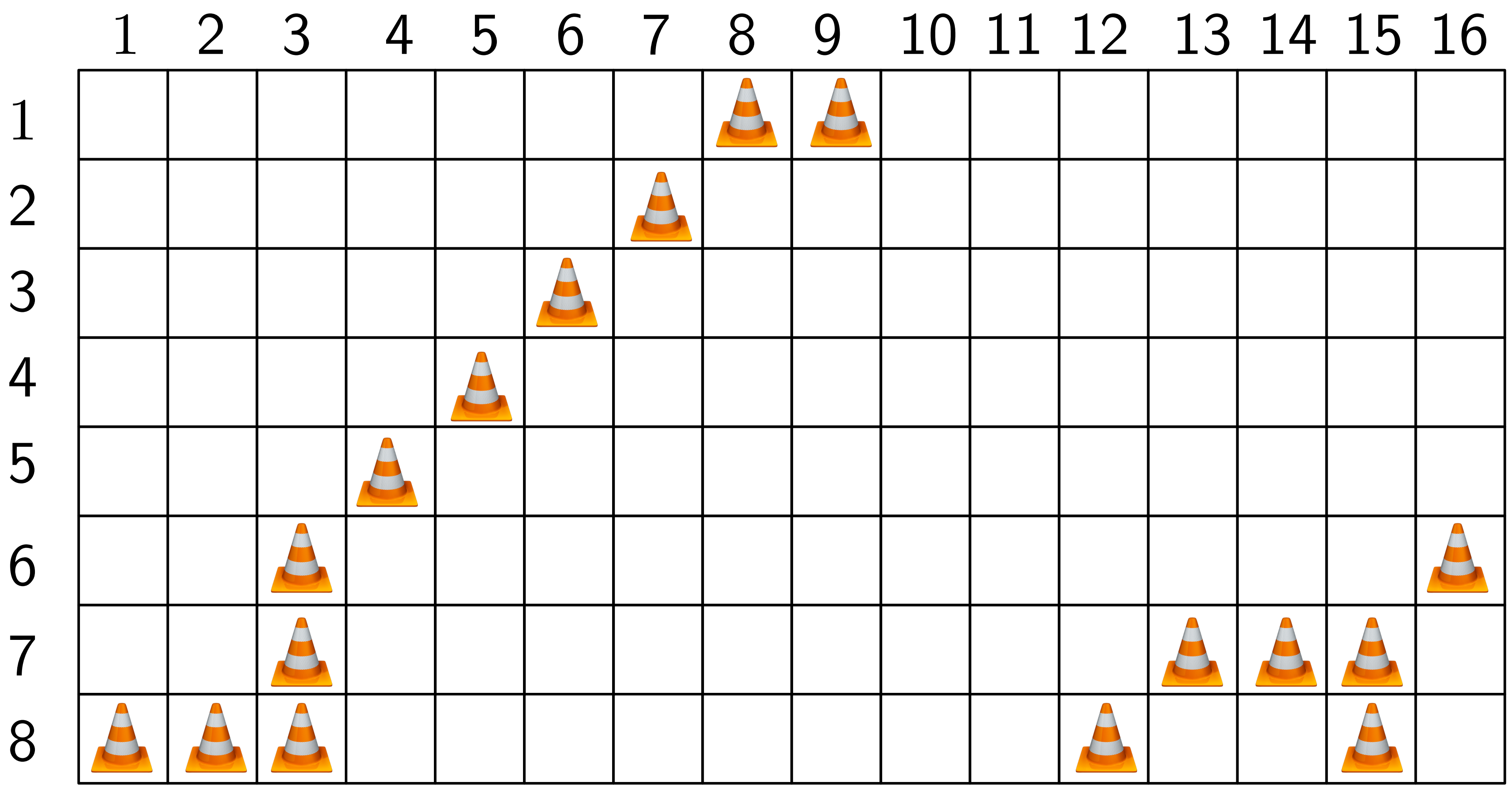
- $n = 8, m = 16$



- È possibile raggiungere la casella $(8, 16)$ rimuovendo **esattamente** 1 ostacolo?

Minimum Obstacle Removal (Esempio)

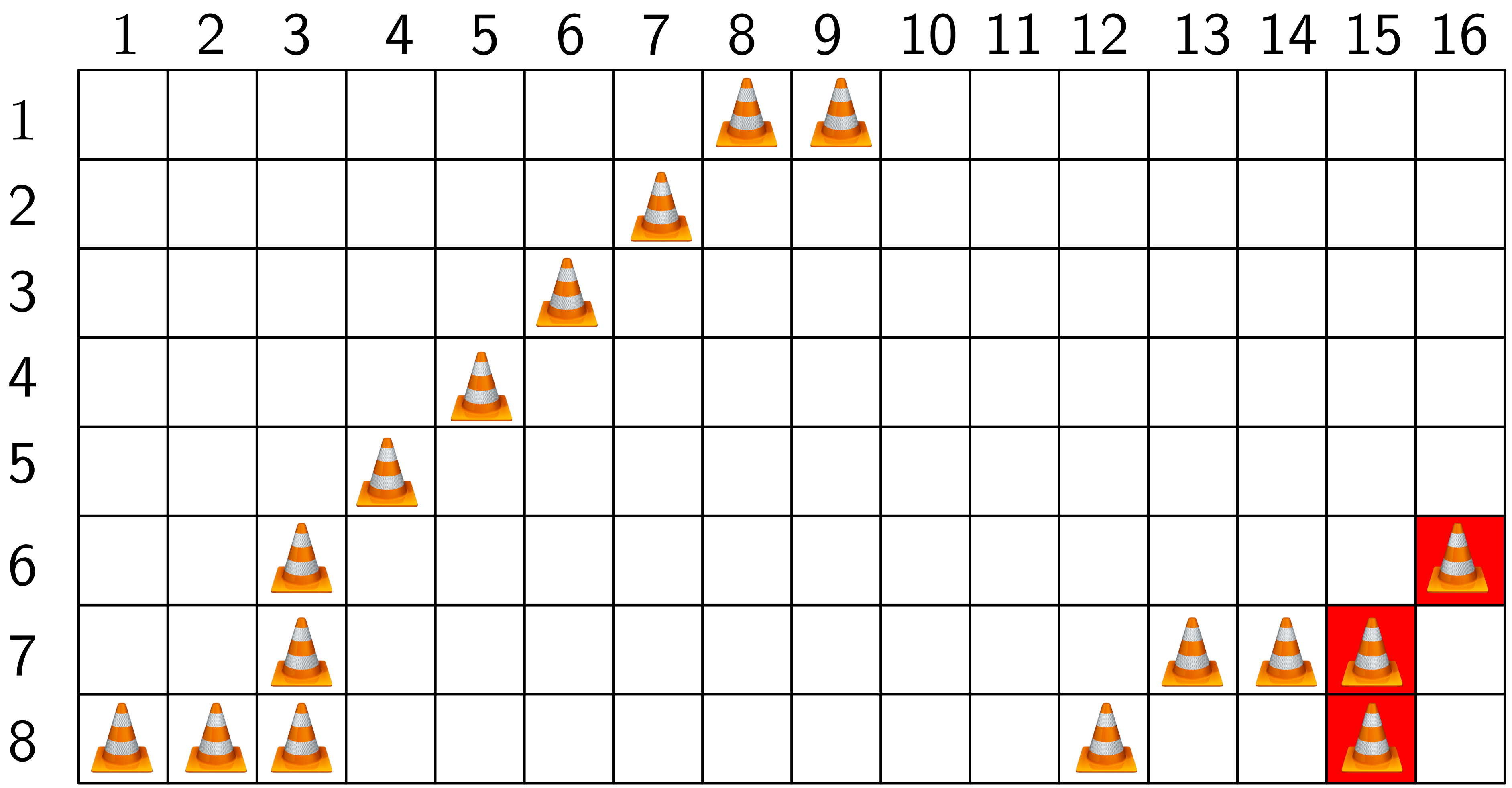
- $n = 8, m = 16$



- È possibile raggiungere la casella $(8, 16)$ rimuovendo **esattamente** 1 ostacolo? **No**.

Minimum Obstacle Removal (Esempio)

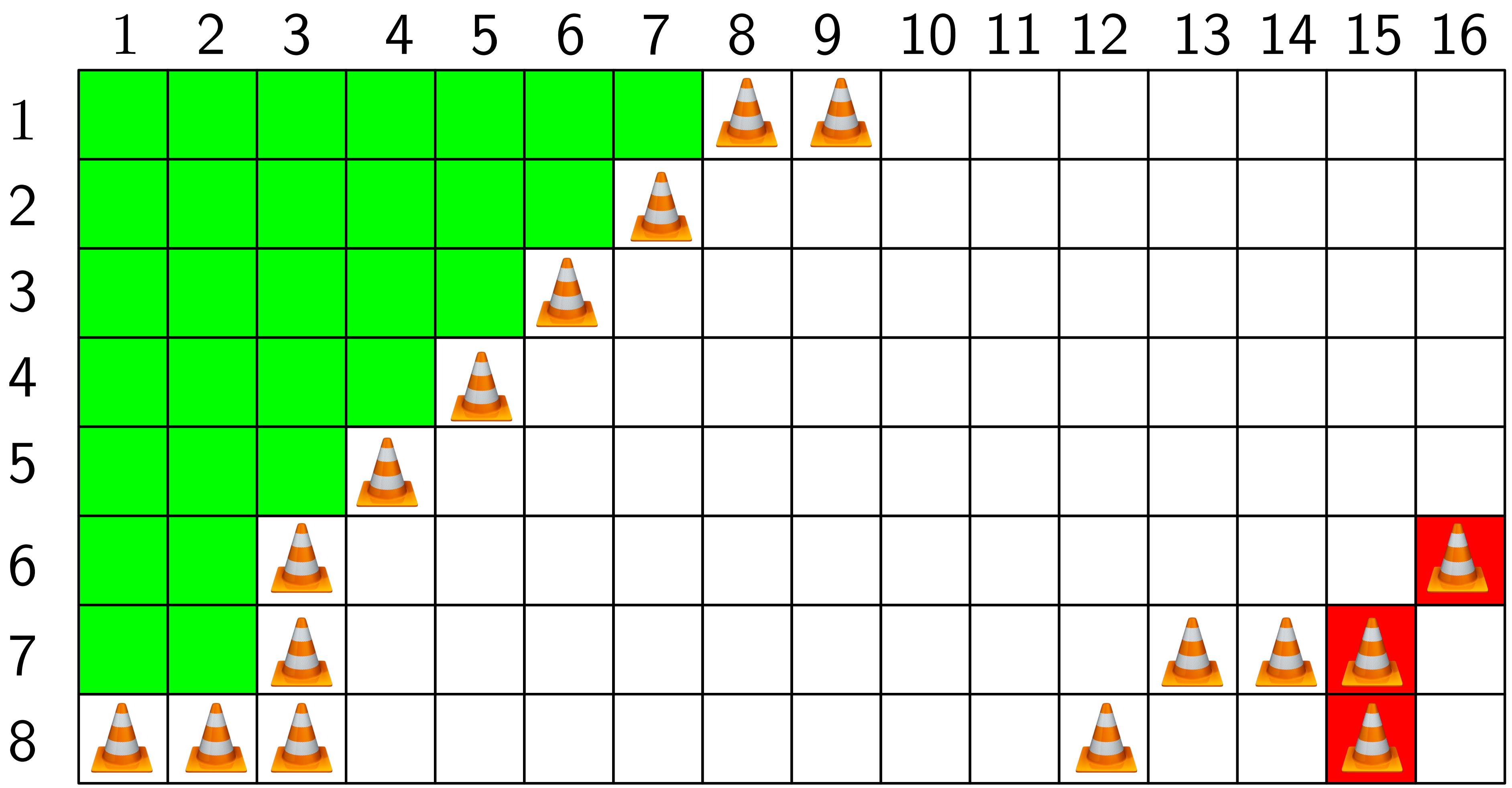
- $n = 8, m = 16$



- È possibile raggiungere la casella $(8, 16)$ rimuovendo **esattamente** 1 ostacolo? **No**.
- Basta osservare che per raggiungere la casella $(8, 16)$ devo rimuovere almeno un ostacolo in $(6, 16), (7, 15), (8, 15)$.

Minimum Obstacle Removal (Esempio)

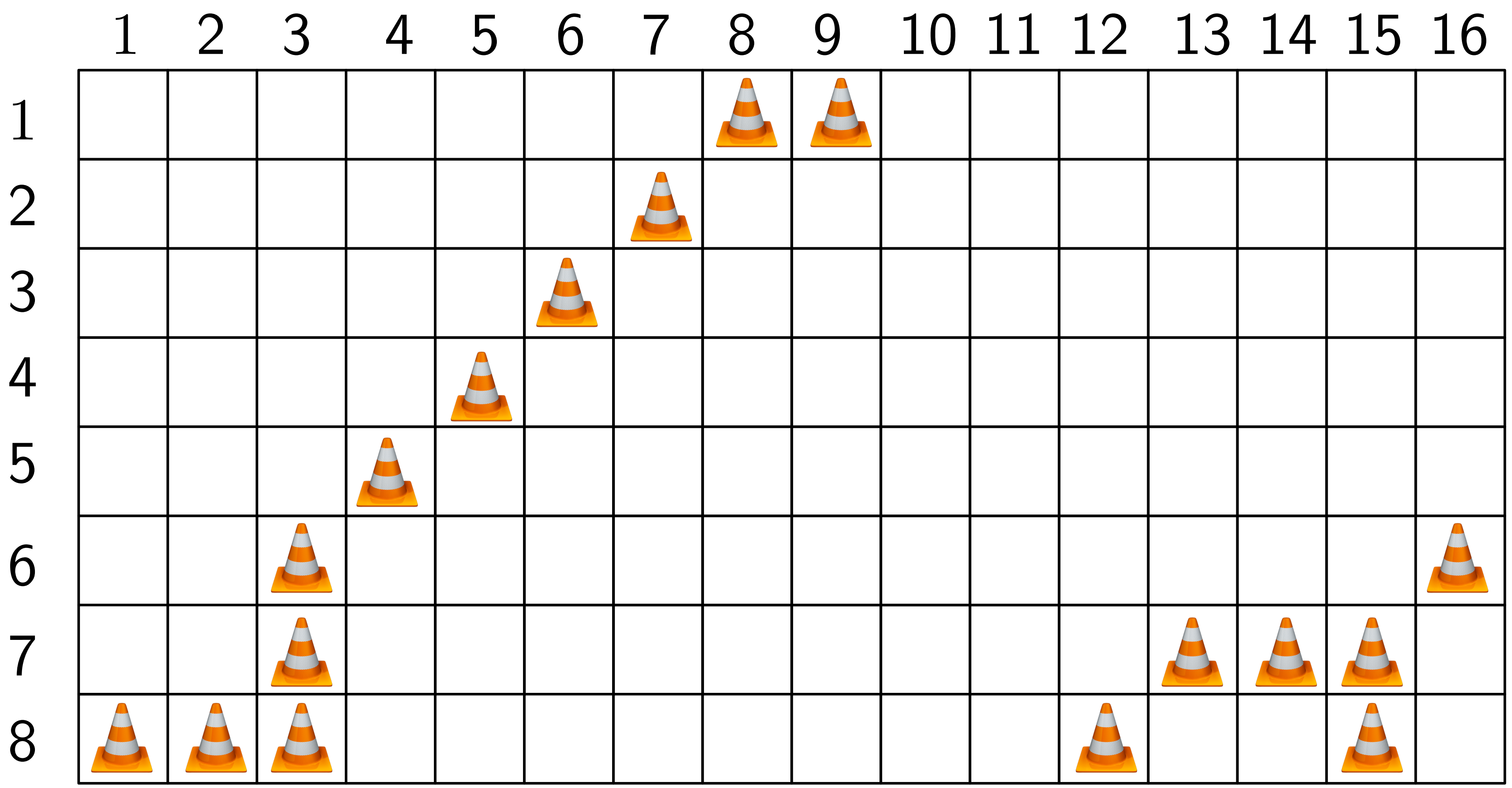
- $n = 8, m = 16$



- È possibile raggiungere la casella $(8, 16)$ rimuovendo **esattamente** 1 ostacolo? **No**.
- Basta osservare che per raggiungere la casella $(8, 16)$ devo rimuovere almeno un ostacolo in $(6, 16), (7, 15), (8, 15)$.
- Successivamente non è possibile "uscire" dalla componente colorata in verde.

Minimum Obstacle Removal (Esempio)

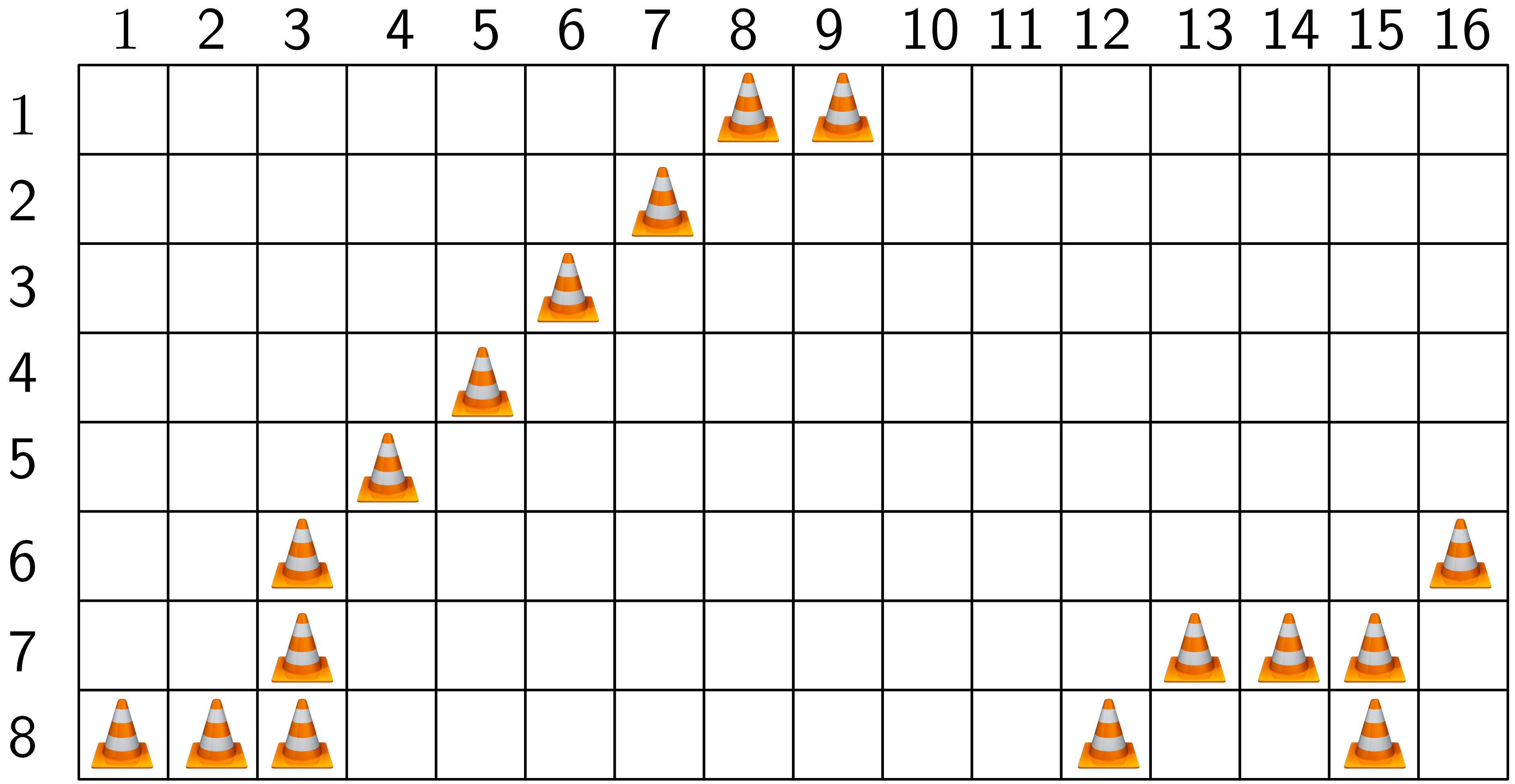
- $n = 8, m = 16$



- È possibile raggiungere la casella $(8, 16)$ rimuovendo **esattamente** 2 ostacoli?

Minimum Obstacle Removal (Esempio)

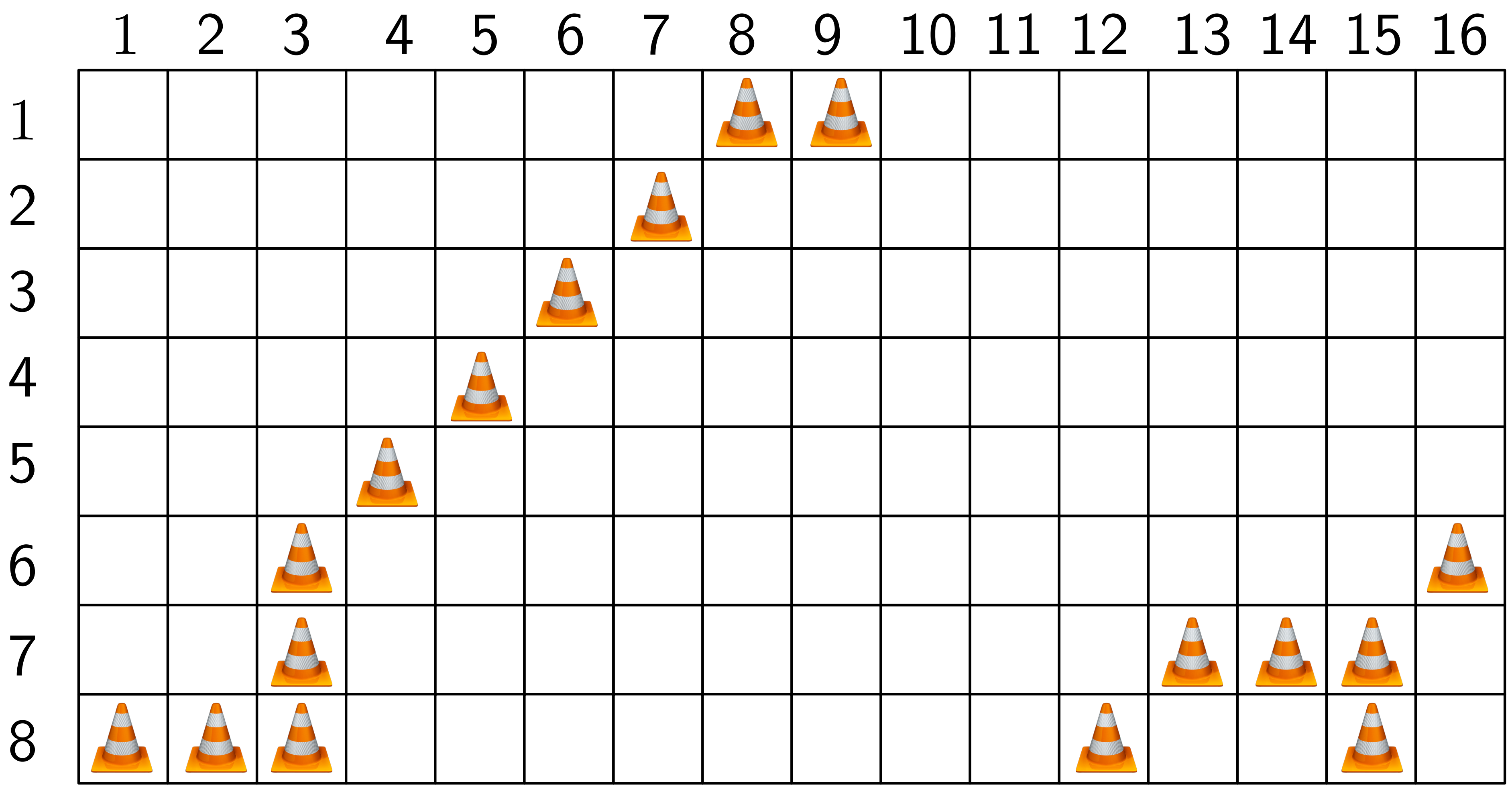
- $n = 8, m = 16$



- È possibile raggiungere la casella $(8, 16)$ rimuovendo **esattamente** 2 ostacoli? **Sì**.

Minimum Obstacle Removal (Esempio)

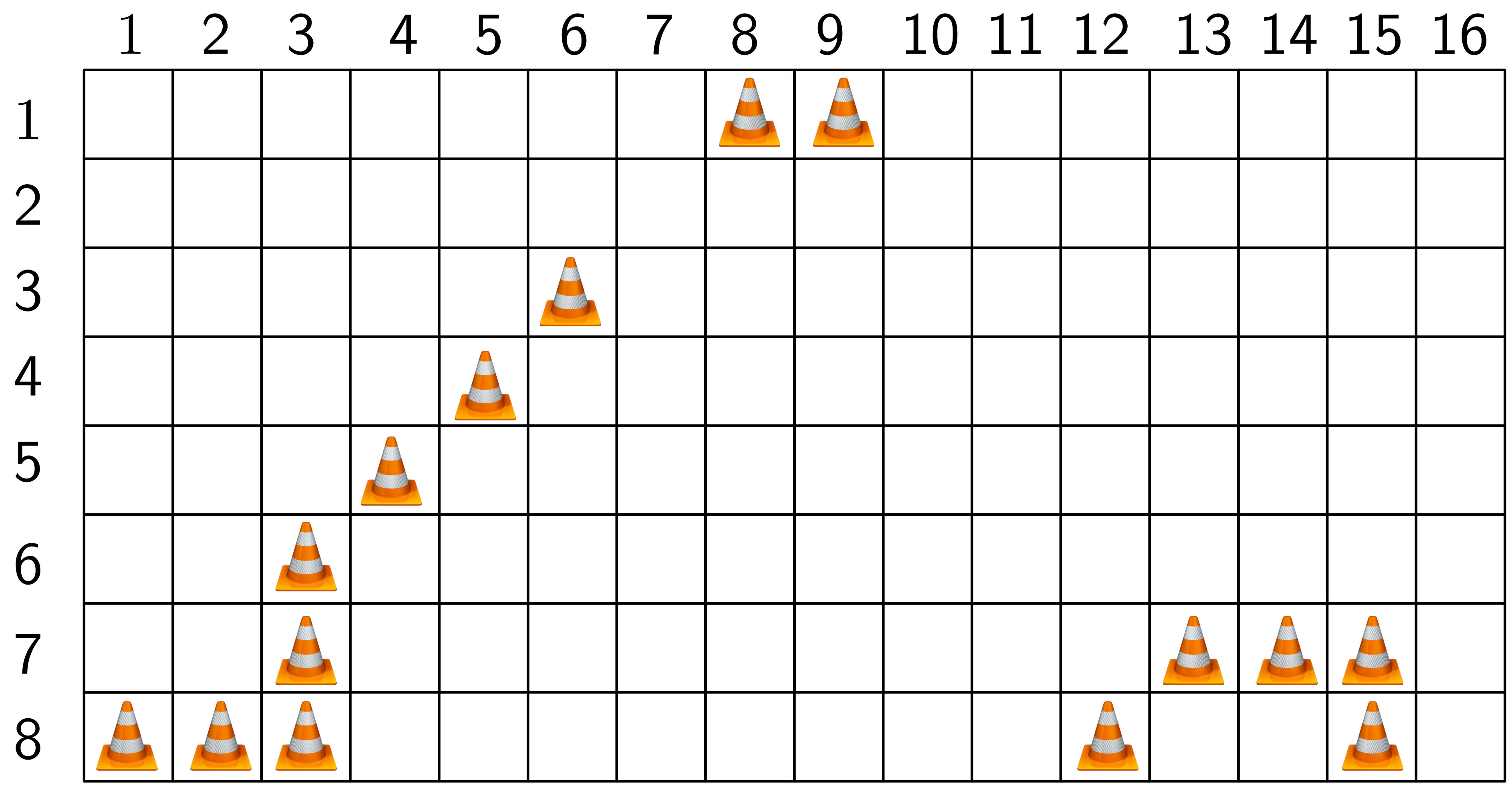
- $n = 8, m = 16$



- È possibile raggiungere la casella $(8, 16)$ rimuovendo **esattamente** 2 ostacoli? **Sì**.
- Per esempio rimuovendo gli ostacoli in $(2, 7)$ e $(6, 16)$.

Minimum Obstacle Removal (Esempio)

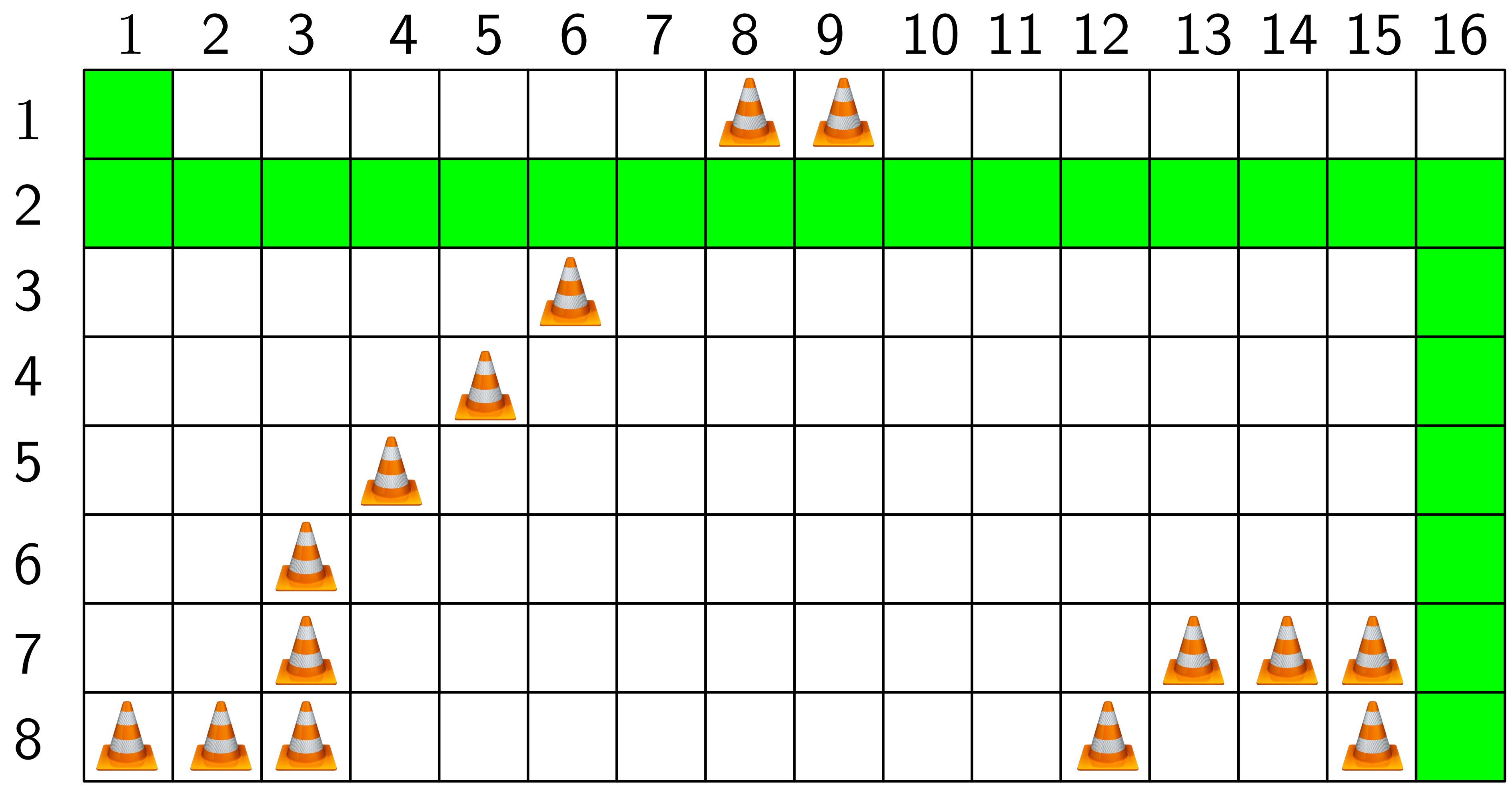
- $n = 8, m = 16$



- È possibile raggiungere la casella $(8, 16)$ rimuovendo **esattamente** 2 ostacoli? **Sì**.
- Per esempio rimuovendo gli ostacoli in $(2, 7)$ e $(6, 16)$.

Minimum Obstacle Removal (Esempio)

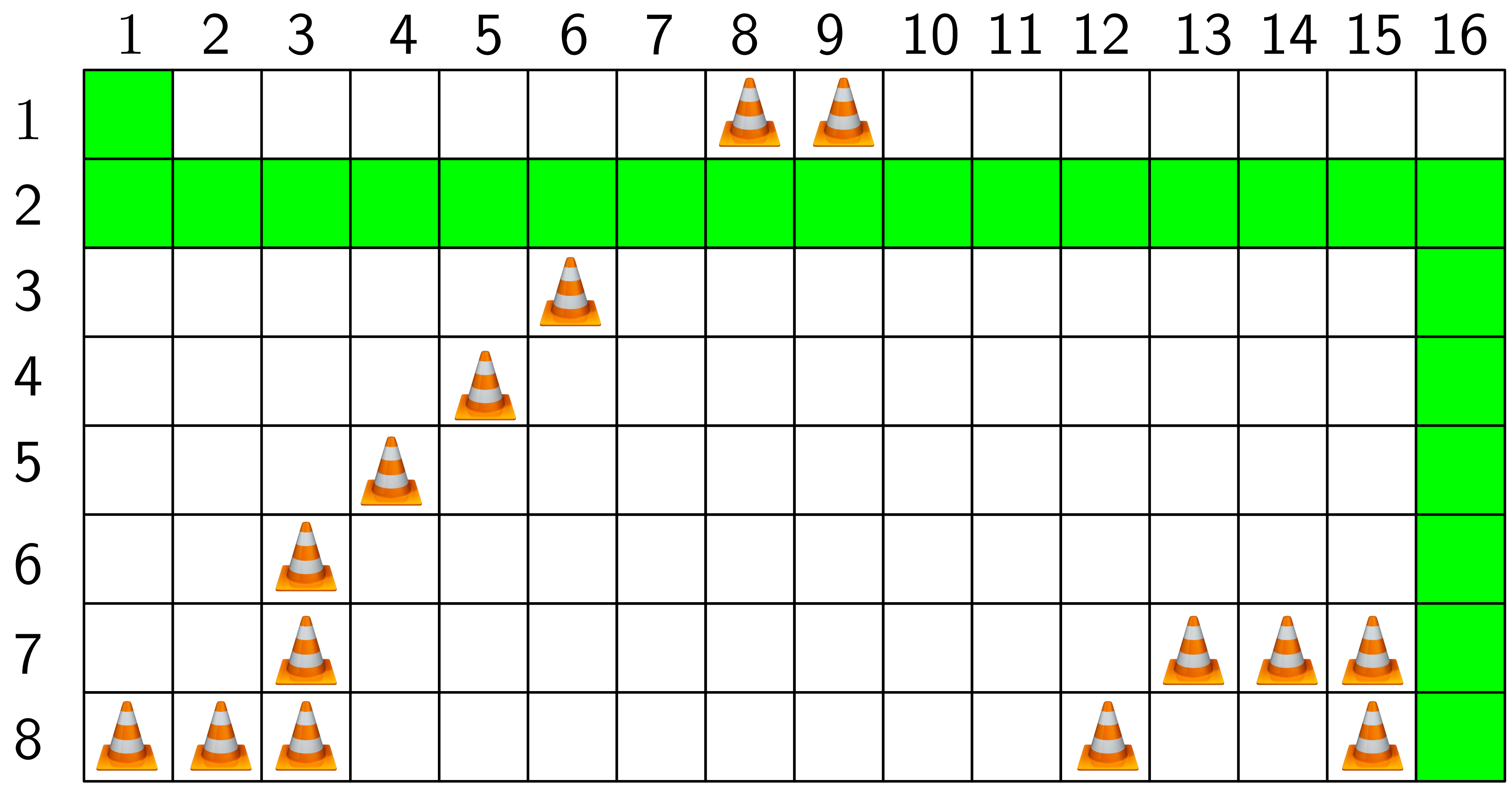
- $n = 8, m = 16$



- È possibile raggiungere la casella $(8, 16)$ rimuovendo **esattamente** 2 ostacoli? **Sì**.
- Per esempio rimuovendo gli ostacoli in $(2, 7)$ e $(6, 16)$.

Minimum Obstacle Removal (Esempio)

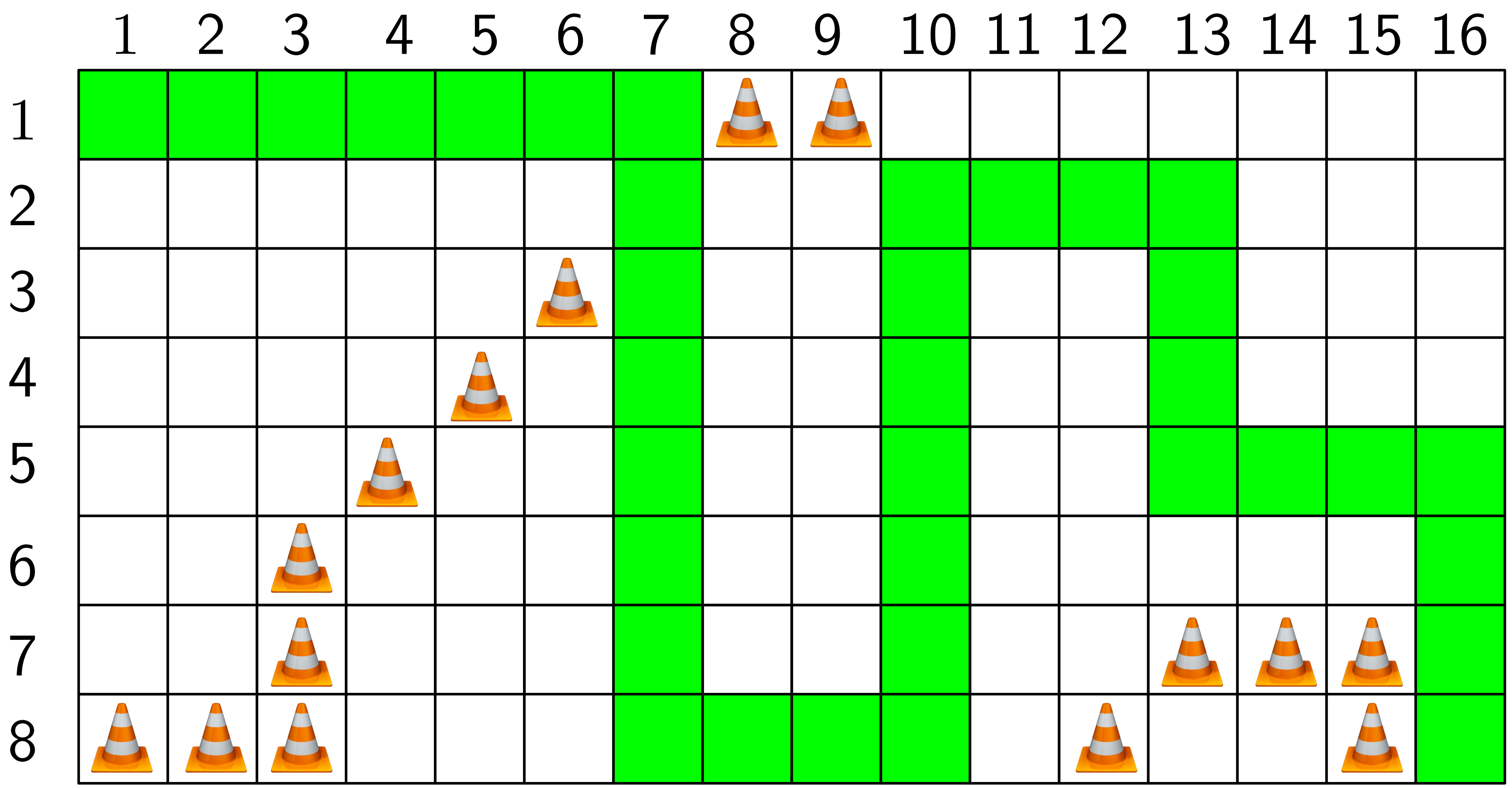
- $n = 8, m = 16$



Osservazione: Non è necessario che il cammino da $(1, 1)$ a (n, m) sia minimo, è sufficiente che (n, m) sia raggiungibile partendo da $(1, 1)$.

Minimum Obstacle Removal (Esempio)

- $n = 8, m = 16$



Osservazione: Non è necessario che il cammino da $(1, 1)$ a (n, m) sia minimo, è sufficiente che (n, m) sia raggiungibile partendo da $(1, 1)$.

Minimum Obstacle Removal

Minimum Obstacle Removal

- **Input:** A binary matrix $A \in \{0,1\}^{n,m} : n, m \in \mathbb{N}^+$.

Minimum Obstacle Removal

- **Input:** A binary matrix $A \in \{0,1\}^{n,m} : n, m \in \mathbb{N}^+$.
- **Output:** $o \in \mathbb{N} : \exists P = \{p_1, \ldots, p_k\} \subseteq (C = \{1,2,\ldots,n\} \times \{1,2,\ldots,m\})$ such that

Minimum Obstacle Removal

- **Input:** A binary matrix $A \in \{0,1\}^{n,m} : n, m \in \mathbb{N}^+$.
- **Output:** $o \in \mathbb{N} : \exists P = \{p_1, \ldots, p_k\} \subseteq (C = \{1, 2, \ldots, n\} \times \{1, 2, \ldots, m\})$ such that
  - $p_1 = (1, 1)$
  - $p_k = (n, m)$
  - $p_{i+1} \in (\{(a+1, b), (a, b+1), (a-1, b), (a, b-1)\} \cap C)$, $p_i = (a, b)$

Minimum Obstacle Removal

- **Input:** A binary matrix $A \in \{0,1\}^{n,m} : n, m \in \mathbb{N}^+$.
- **Output:** $o \in \mathbb{N} : \exists P = \{p_1, \ldots, p_k\} \subseteq (C = \{1, 2, \ldots, n\} \times \{1, 2, \ldots, m\})$ such that
  - $p_1 = (1, 1)$
  - $p_k = (n, m)$
  - $p_{i+1} \in (\{(a+1, b), (a, b+1), (a-1, b), (a, b-1)\} \cap C)$, $p_i = (a, b)$
  - $|\{p_1, \ldots, p_k\} \cap \{(i, j) \in C : A_{ij} = 1\}| \leq o$

Minimum Obstacle Removal

- **Input:** A binary matrix $A \in \{0,1\}^{n,m} : n, m \in \mathbb{N}^+$.
- **Output:** $o \in \mathbb{N} : \exists P = \{p_1, \ldots, p_k\} \subseteq (C = \{1,2,\ldots,n\} \times \{1,2,\ldots,m\})$ such that
  - $p_1 = (1,1)$
  - $p_k = (n,m)$
  - $p_{i+1} \in (\{(a+1,b),(a,b+1),(a-1,b),(a,b-1)\} \cap C)$, $p_i = (a,b)$
  - $|\{p_1,\ldots,p_k\} \cap \{(i,j) \in C : A_{ij} = 1\}| \leq o$
- **Metrica:** o

Minimum Obstacle Removal

- **Input:** A binary matrix $A \in \{0,1\}^{n,m} : n, m \in \mathbb{N}^+$.
- **Output:** $o \in \mathbb{N} : \exists P = \{p_1, \ldots, p_k\} \subseteq (C = \{1, 2, \ldots, n\} \times \{1, 2, \ldots, m\})$ such that
  - $p_1 = (1, 1)$
  - $p_k = (n, m)$
  - $p_{i+1} \in (\{(a+1, b), (a, b+1), (a-1, b), (a, b-1)\} \cap C)$, $p_i = (a, b)$
  - $|\{p_1, \ldots, p_k\} \cap \{(i, j) \in C : A_{ij} = 1\}| \leq o$
- **Metrica:** o

Idee?

Forza bruta

- Sia $O = \{(i, j) \in (\{1, 2, \ldots, n\} \times \{1, 2, \ldots, m\}) : A_{ij} = 1\}$ l'insieme degli ostacoli.

Forza bruta

- Sia $O = \{(i, j) \in (\{1, 2, \ldots, n\} \times \{1, 2, \ldots, m\}) : A_{ij} = 1\}$ l'insieme degli ostacoli.

  - For $i \leftarrow 0$ to $n \cdot m$

Forza bruta

- Sia $O = \{(i,j) \in (\{1,2,\ldots,n\} \times \{1,2,\ldots,m\}) : A_{ij} = 1\}$ l'insieme degli ostacoli.

  - For $i \leftarrow 0$ to $n \cdot m$
    - For each set $O_i \in \binom{O}{i}$ $\longleftarrow$ Se $i > |O|$, $\binom{O}{i} = 0$

Forza bruta

- Sia $O = \{(i,j) \in (\{1,2,\ldots,n\} \times \{1,2,\ldots,m\}) : A_{ij} = 1\}$ l'insieme degli ostacoli.

  - For $i \leftarrow 0$ to $n \cdot m$
    - For each set $O_i \in \binom{O}{i}$ ⟵ Se $i > |O|$, $\binom{O}{i} = 0$

$$A'_{ij} = \begin{cases} 0 & \text{if } (i,j) \in O_i, \\ A_{ij} & \text{otherwise} \end{cases}$$

Forza bruta

- Sia $O = \{(i,j) \in (\{1,2,\ldots,n\} \times \{1,2,\ldots,m\}) : A_{ij} = 1\}$ l'insieme degli ostacoli.

  - For $i \leftarrow 0$ to $n \cdot m$
    - For each set $O_i \in \binom{O}{i}$ $\longleftarrow$ Se $i > |O|$, $\binom{O}{i} = 0$

$$A'_{ij} = \begin{cases} 0 & \text{if } (i,j) \in O_i, \\ A_{ij} & \text{otherwise} \end{cases}$$

se esiste un path **ammissibile** da $(1,1)$ a (n,m) rispetto ad A'_{ij} **return** i

Forza bruta

- Sia $O = \{(i,j) \in (\{1,2,\ldots,n\} \times \{1,2,\ldots,m\}) : A_{ij} = 1\}$ l'insieme degli ostacoli.

  - For $i \leftarrow 0$ to $n \cdot m$
    - For each set $O_i \in \binom{O}{i}$ ← Se $i > |O|$, $\binom{O}{i} = 0$

      $$A'_{ij} = \begin{cases} 0 & \text{if } (i,j) \in O_i, \\ A_{ij} & \text{otherwise} \end{cases}$$

      se esiste un path **ammissibile** da $(1,1)$ a (n,m) rispetto ad A'_{ij} **return** i

- **Complessità temporale?**

Forza bruta

- Sia $O = \{(i,j) \in (\{1,2,\ldots,n\} \times \{1,2,\ldots,m\}) : A_{ij} = 1\}$ l'insieme degli ostacoli.

  - For $i \leftarrow 0$ to $n \cdot m$
    - For each set $O_i \in \binom{O}{i}$ ⟵ Se $i > |O|$, $\binom{O}{i} = 0$

$$A'_{ij} = \begin{cases} 0 & \text{if } (i,j) \in O_i, \\ A_{ij} & \text{otherwise} \end{cases}$$

se esiste un path **ammissibile** da $(1,1)$ a (n,m) rispetto ad A'_{ij} **return** i

- **Complessità temporale?**

$$T(n,m) \geq \sum_{i=0}^{n \cdot m} \binom{n \cdot m}{i} = 2^{n \cdot m}$$ ☹

Una matrice che contiene solo ostacoli.

Forza bruta

- Sia $O = \{(i,j) \in (\{1,2,\ldots,n\} \times \{1,2,\ldots,m\}) : A_{ij} = 1\}$ l'insieme degli ostacoli.

Algoritmo: ForzaBruta

- For $i \leftarrow 0$ to $n \cdot m$
  - For each set $O_i \in \binom{O}{i}$ ⟵ Se $i > |O|$, $\binom{O}{i} = 0$

$$A'_{ij} = \begin{cases} 0 & \text{if } (i,j) \in O_i, \\ A_{ij} & \text{otherwise} \end{cases}$$

se esiste un path **ammissibile** da $(1,1)$ a (n,m) rispetto ad A'_{ij} **return** i

- **Complessità temporale?**

$$T(n,m) \geq \sum_{i=0}^{n\cdot m} \binom{n\cdot m}{i} = 2^{n\cdot m}$$

Come viene implementato il verificatore?

Verificatore lineare

Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.

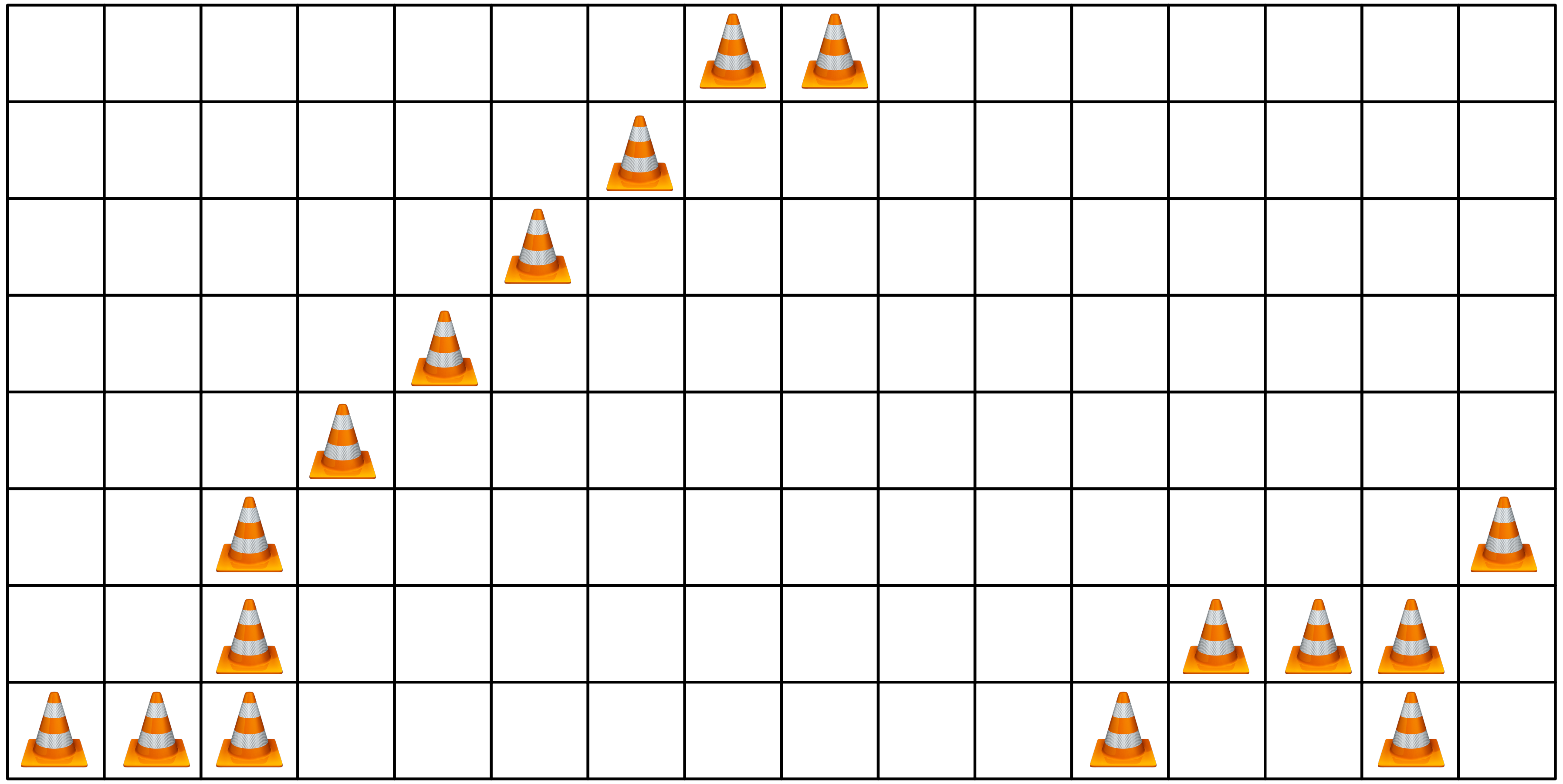
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.

- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.

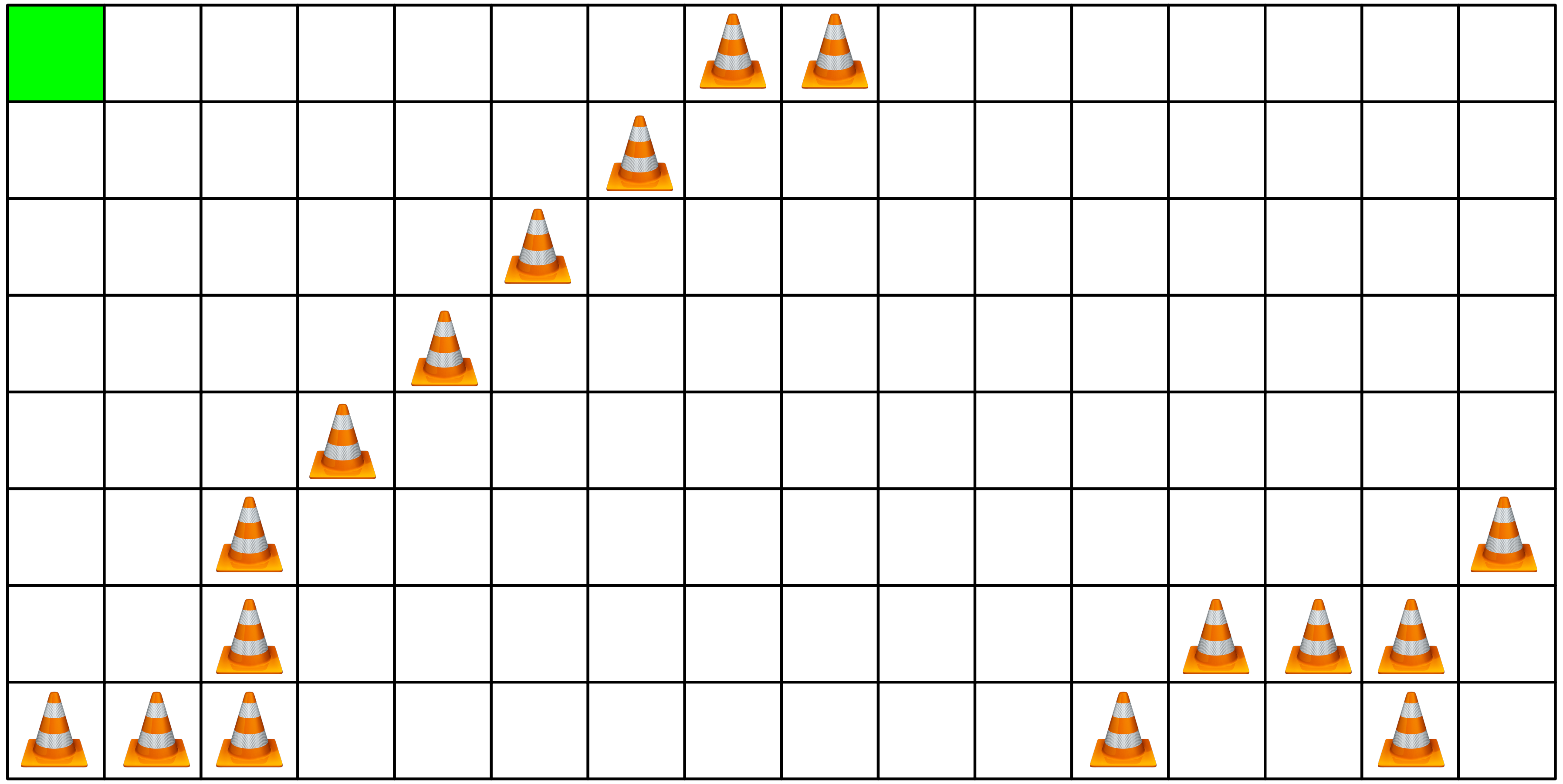
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



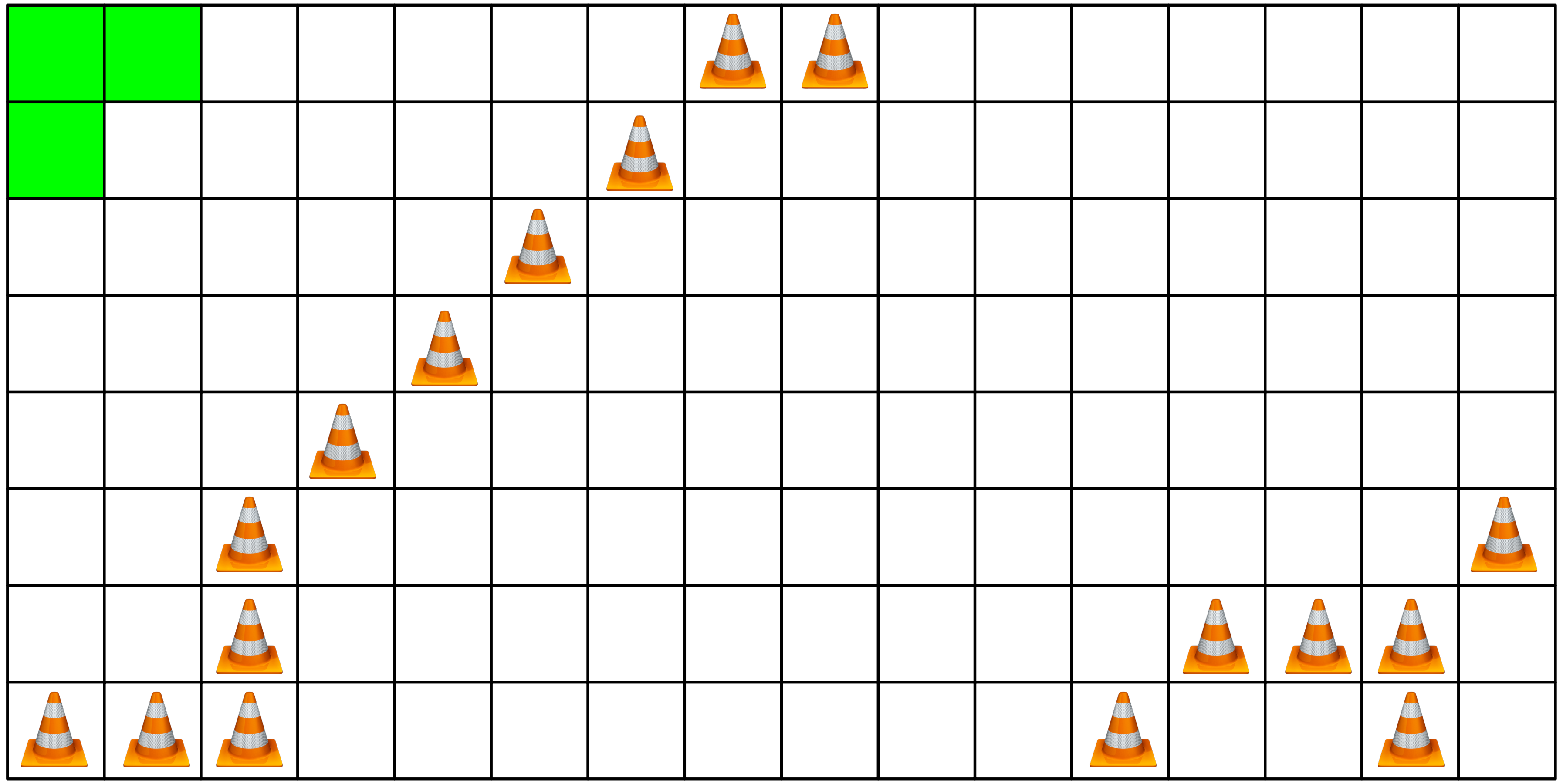
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



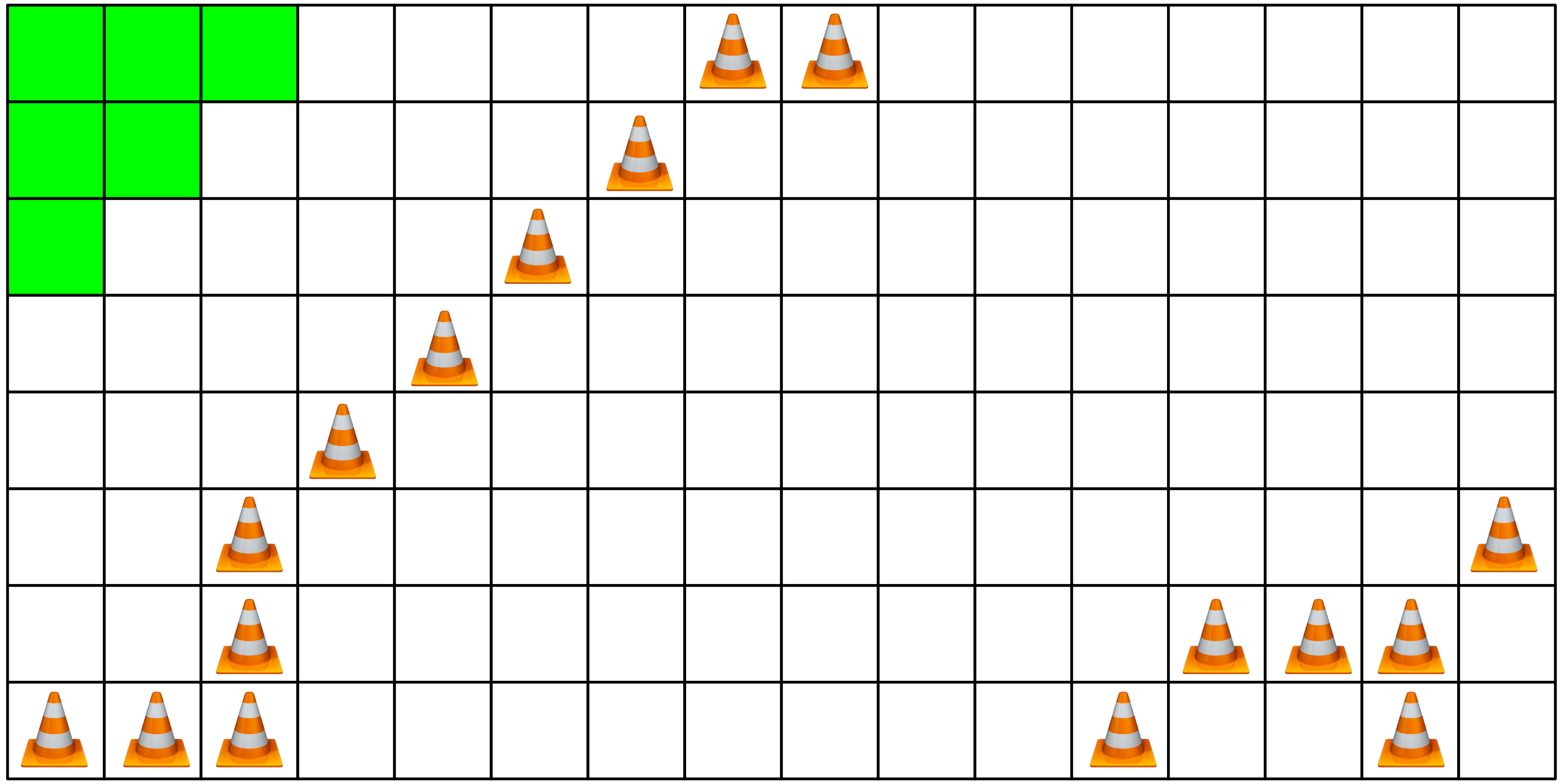
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



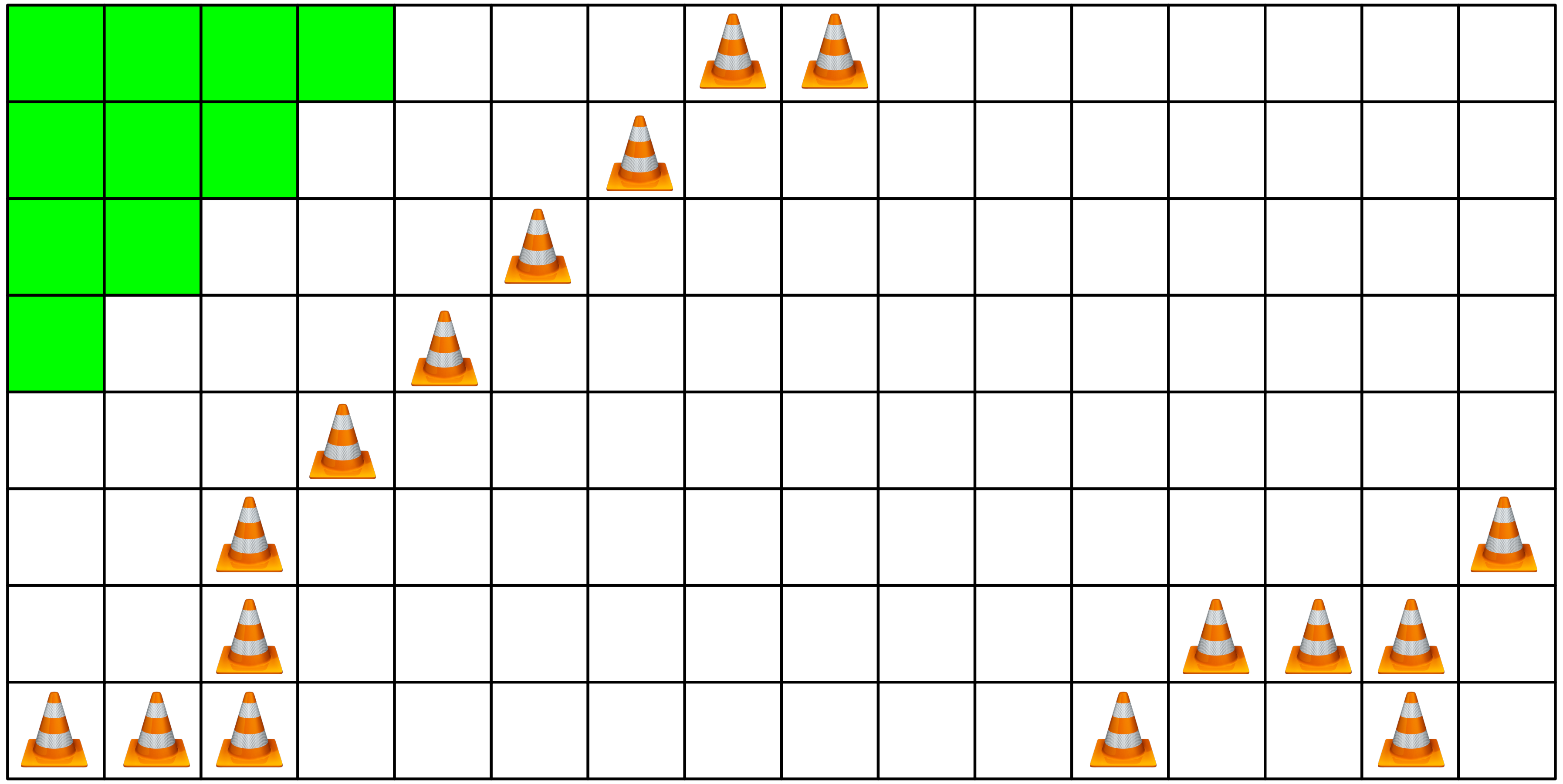
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1,1)$.
- Da ogni casella (i,j) è possibile visitare le caselle in $\{(i+1,j),(i,j+1),(i-1,j),(i,j-1)\}$, se esistono e **non ci sono ostacoli**.



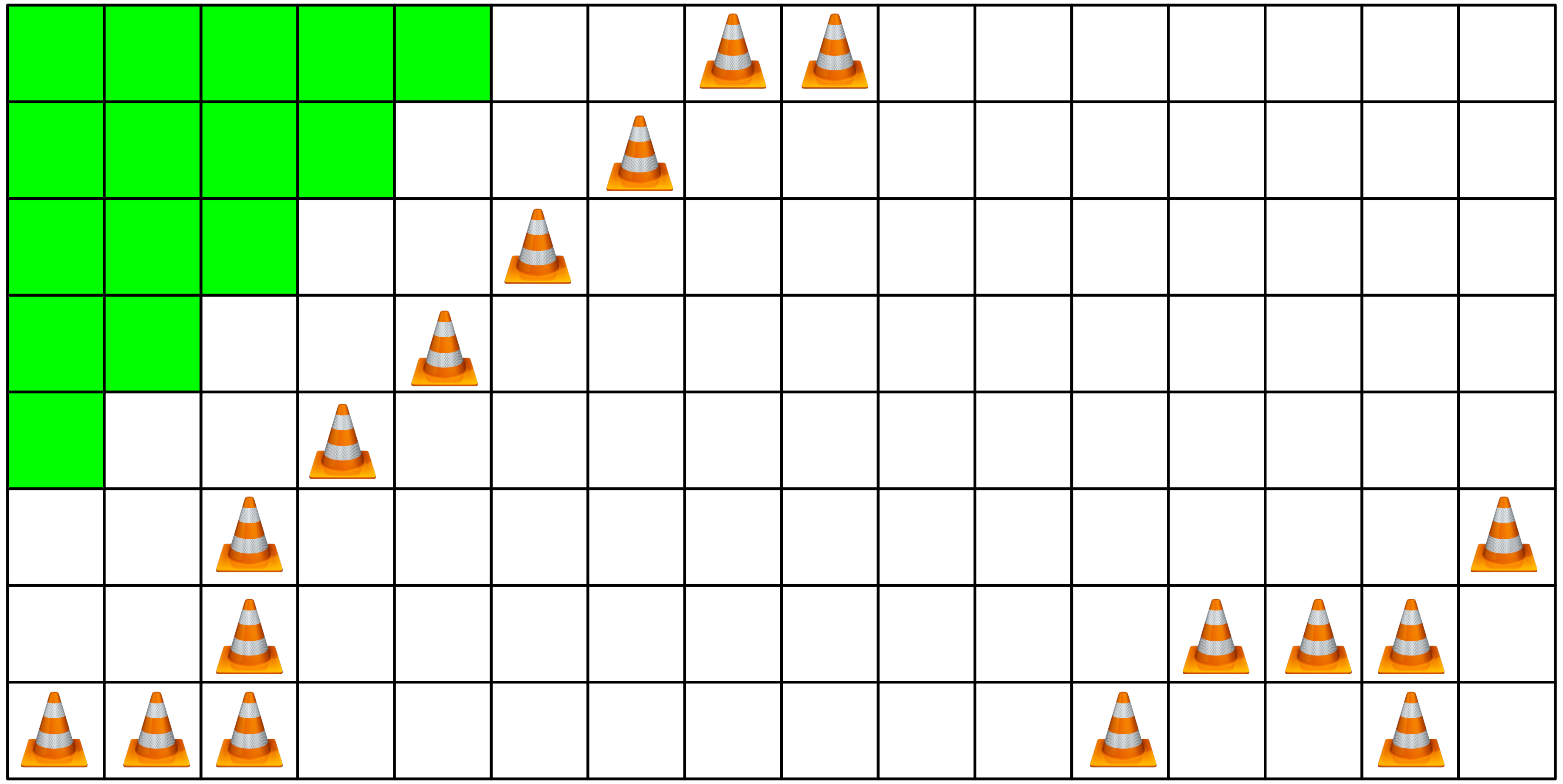
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



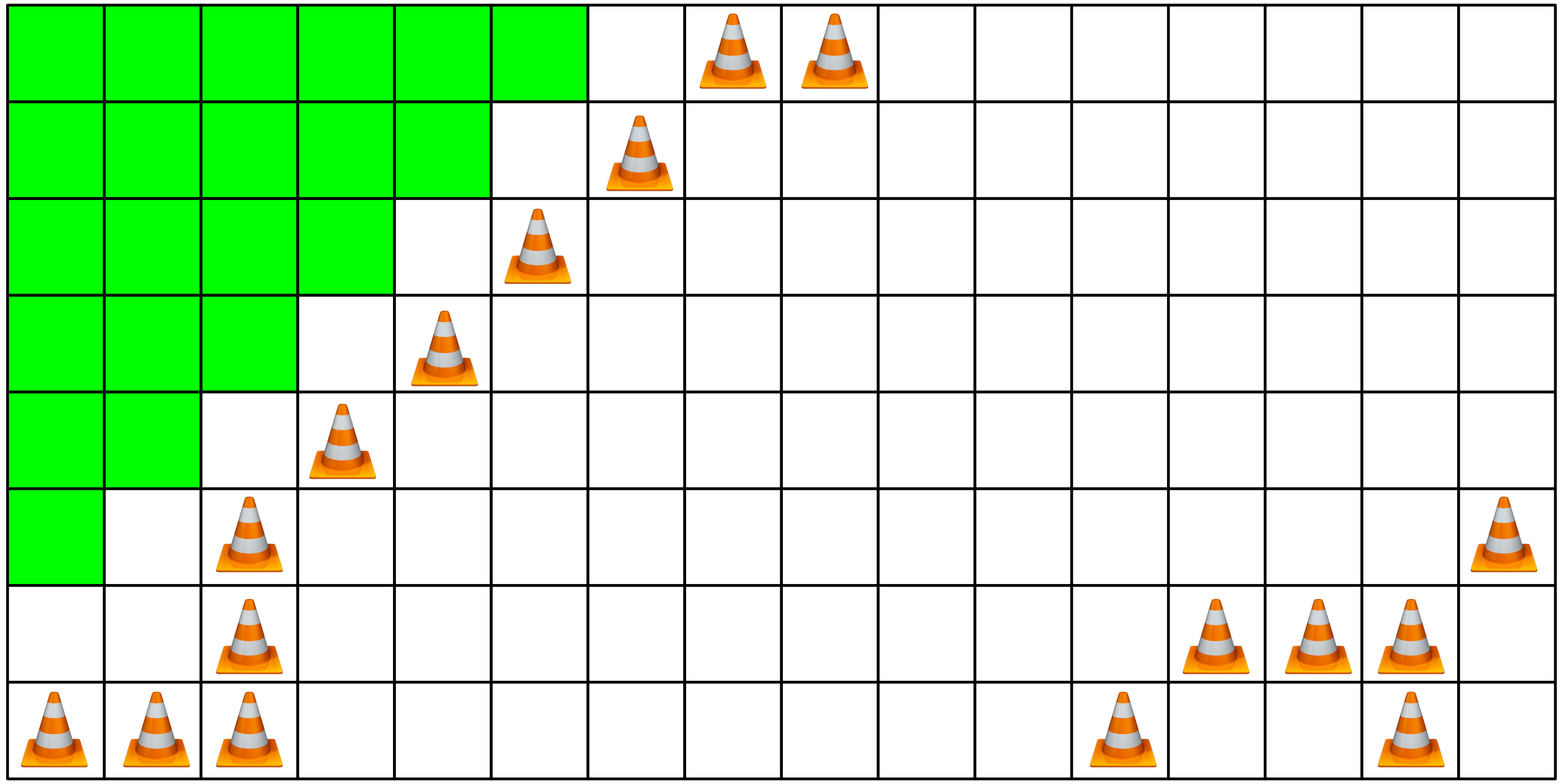
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



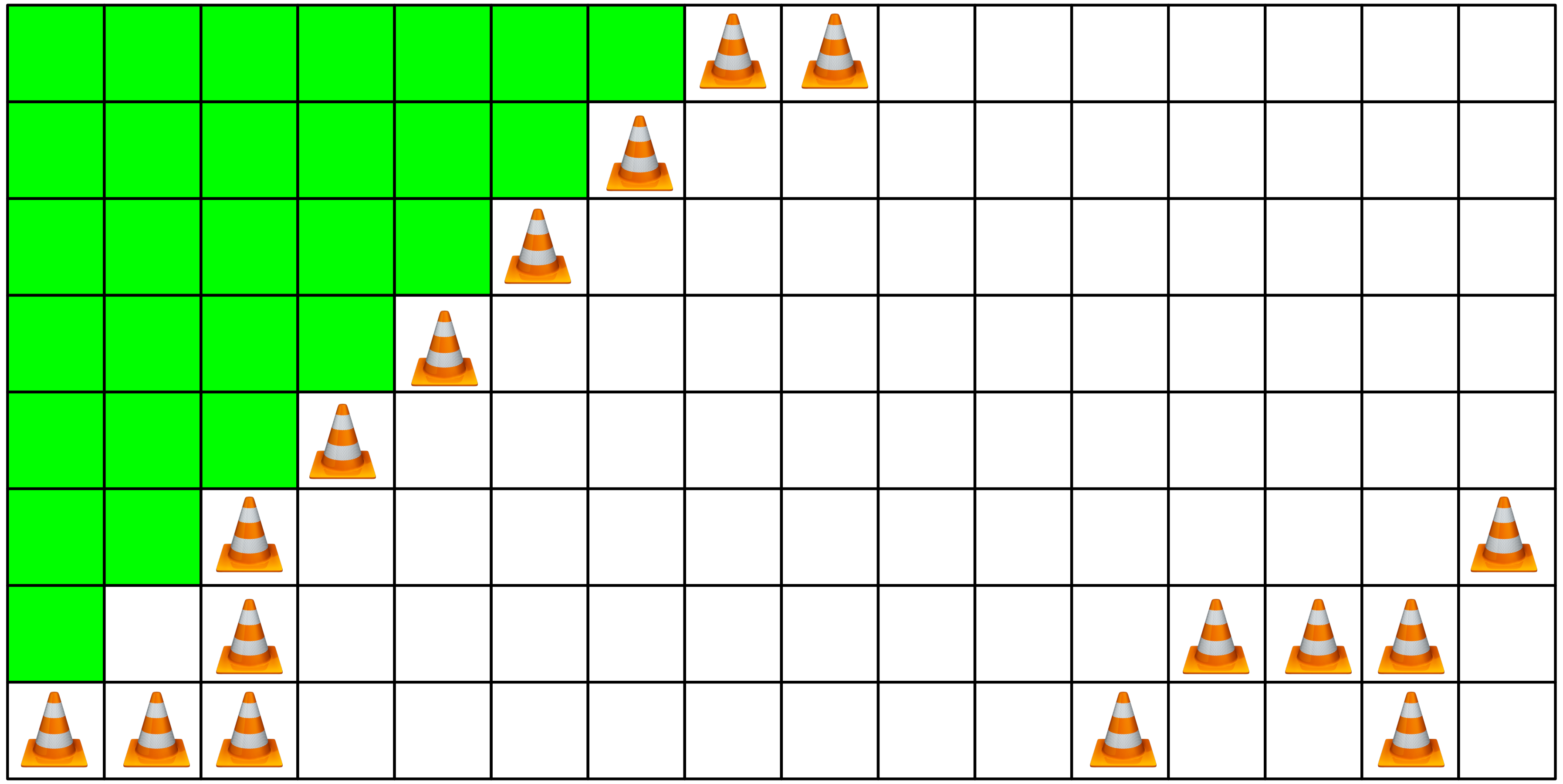
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



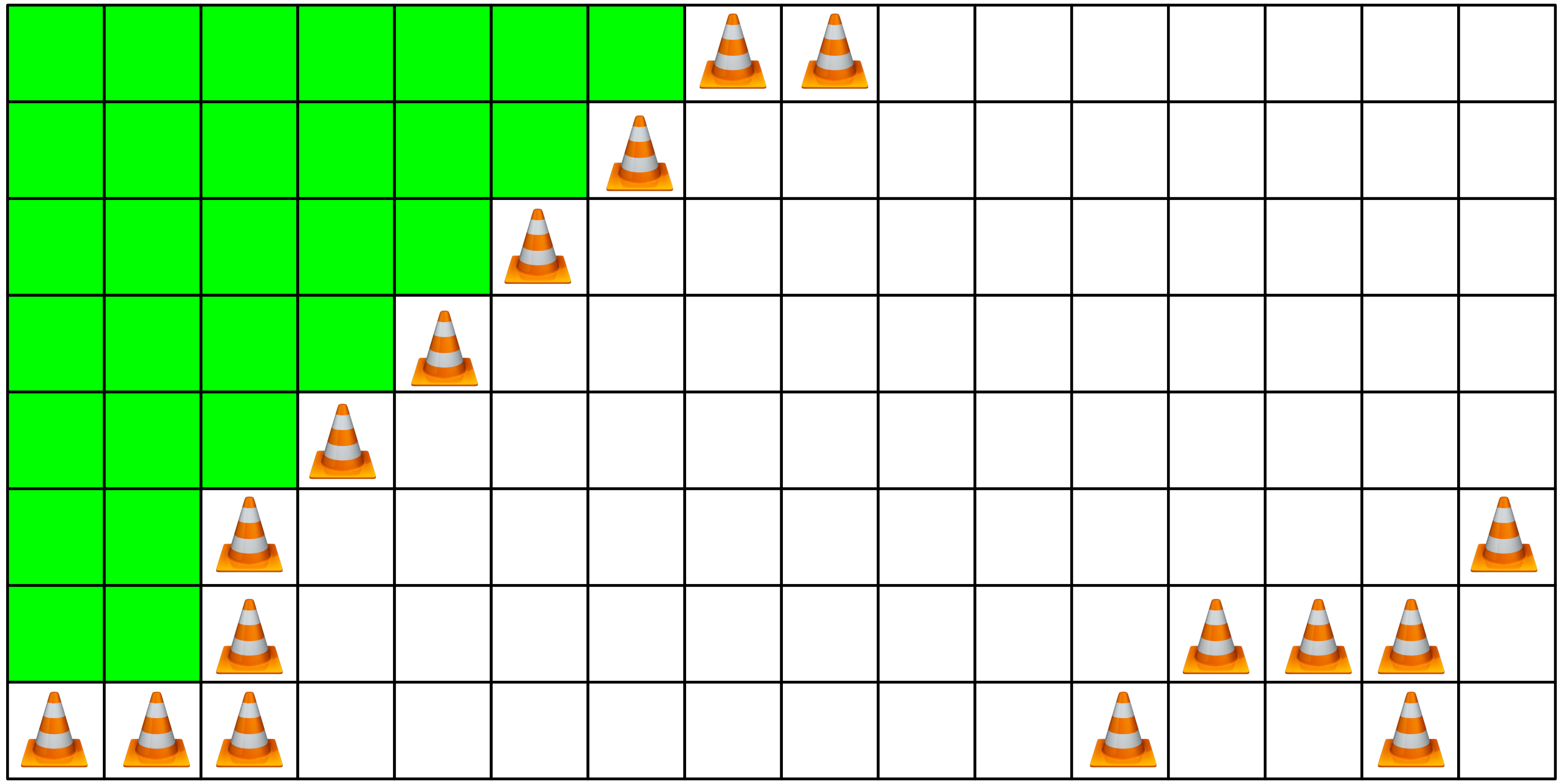
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



Verificatore lineare

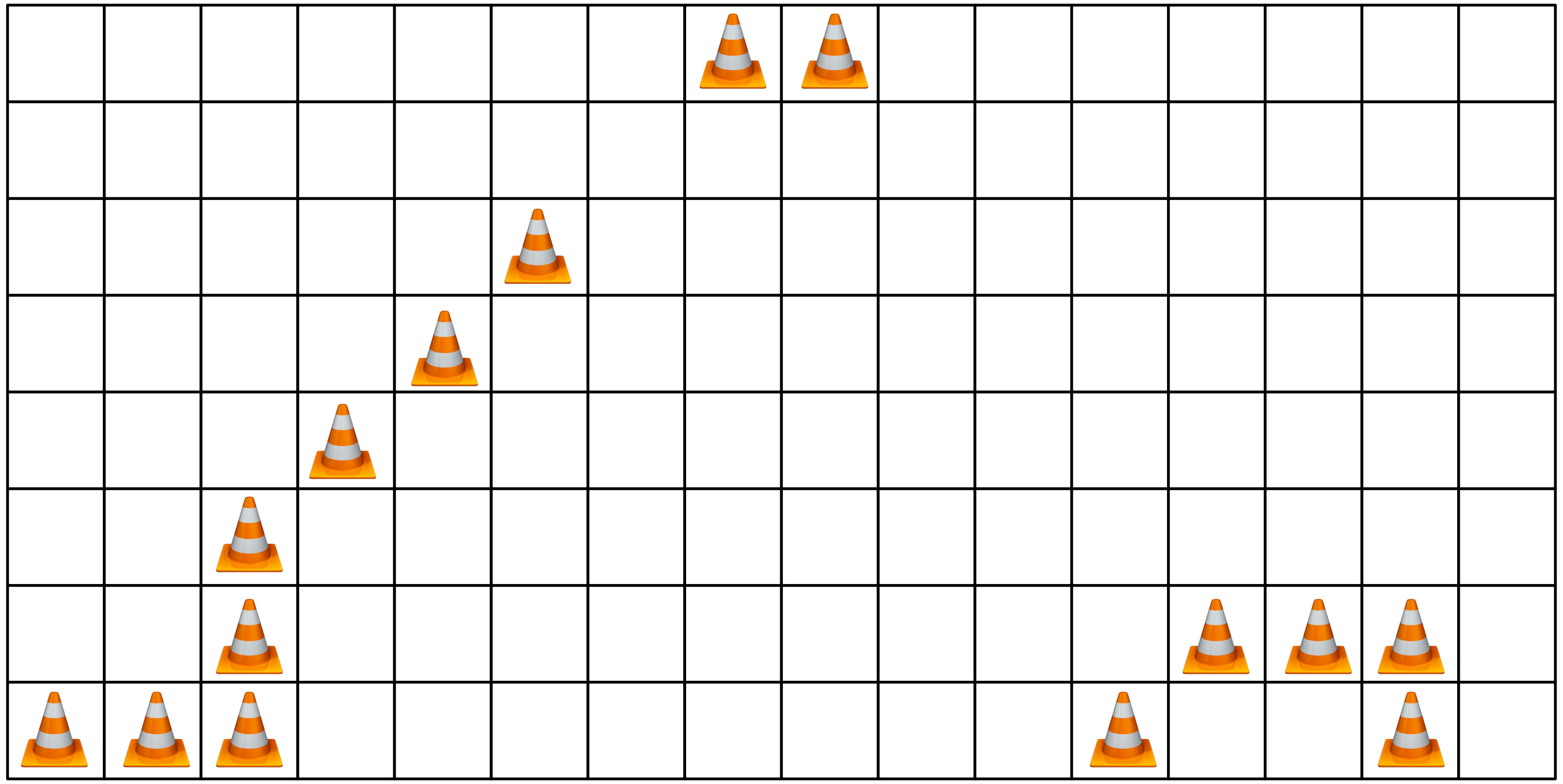
- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1,1)$.
- Da ogni casella (i,j) è possibile visitare le caselle in $\{(i+1,j),(i,j+1),(i-1,j),(i,j-1)\}$, se esistono e **non ci sono ostacoli**.



Non e' possibile raggiungere (n,m) da $(1,1)$.

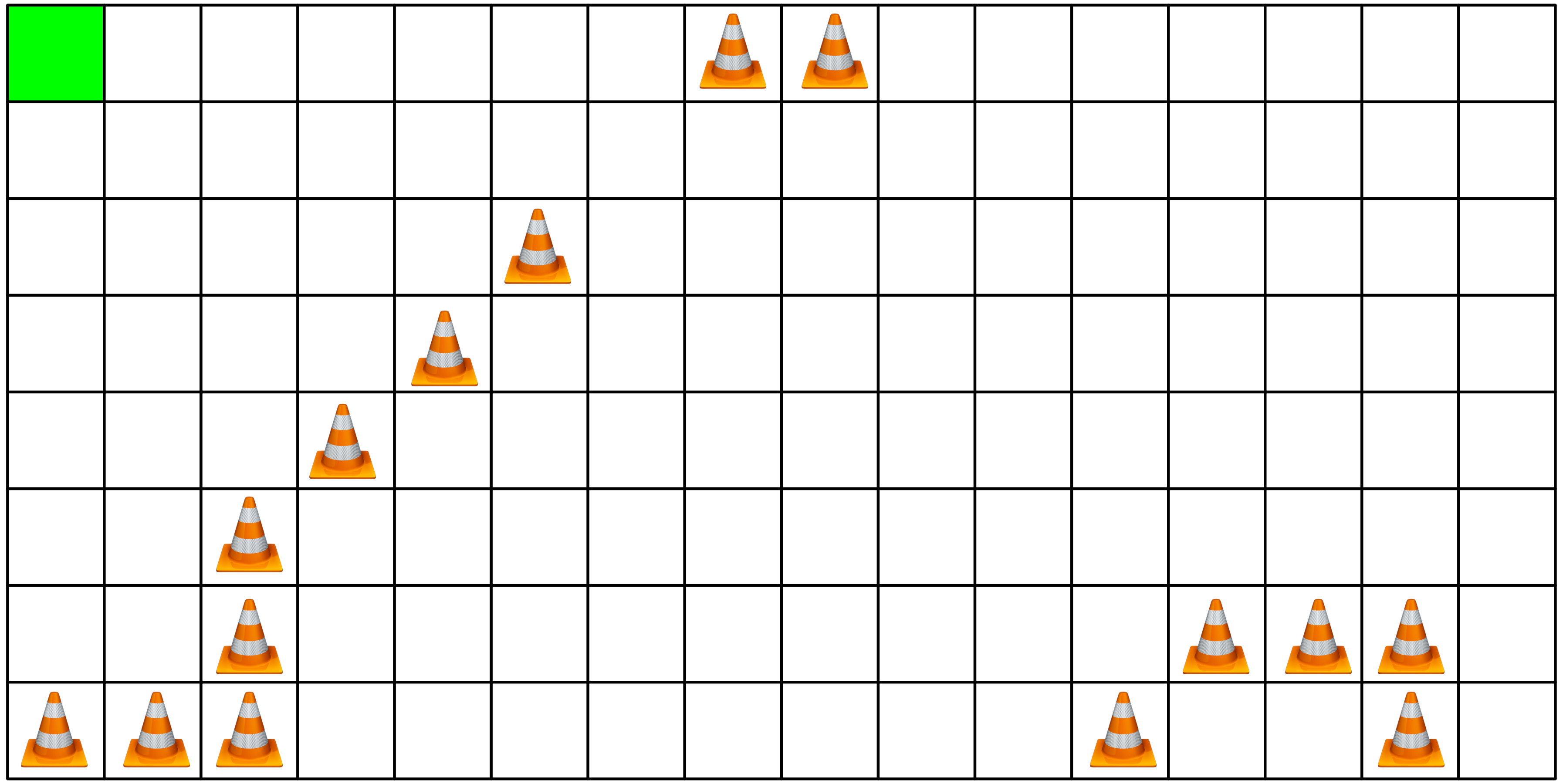
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



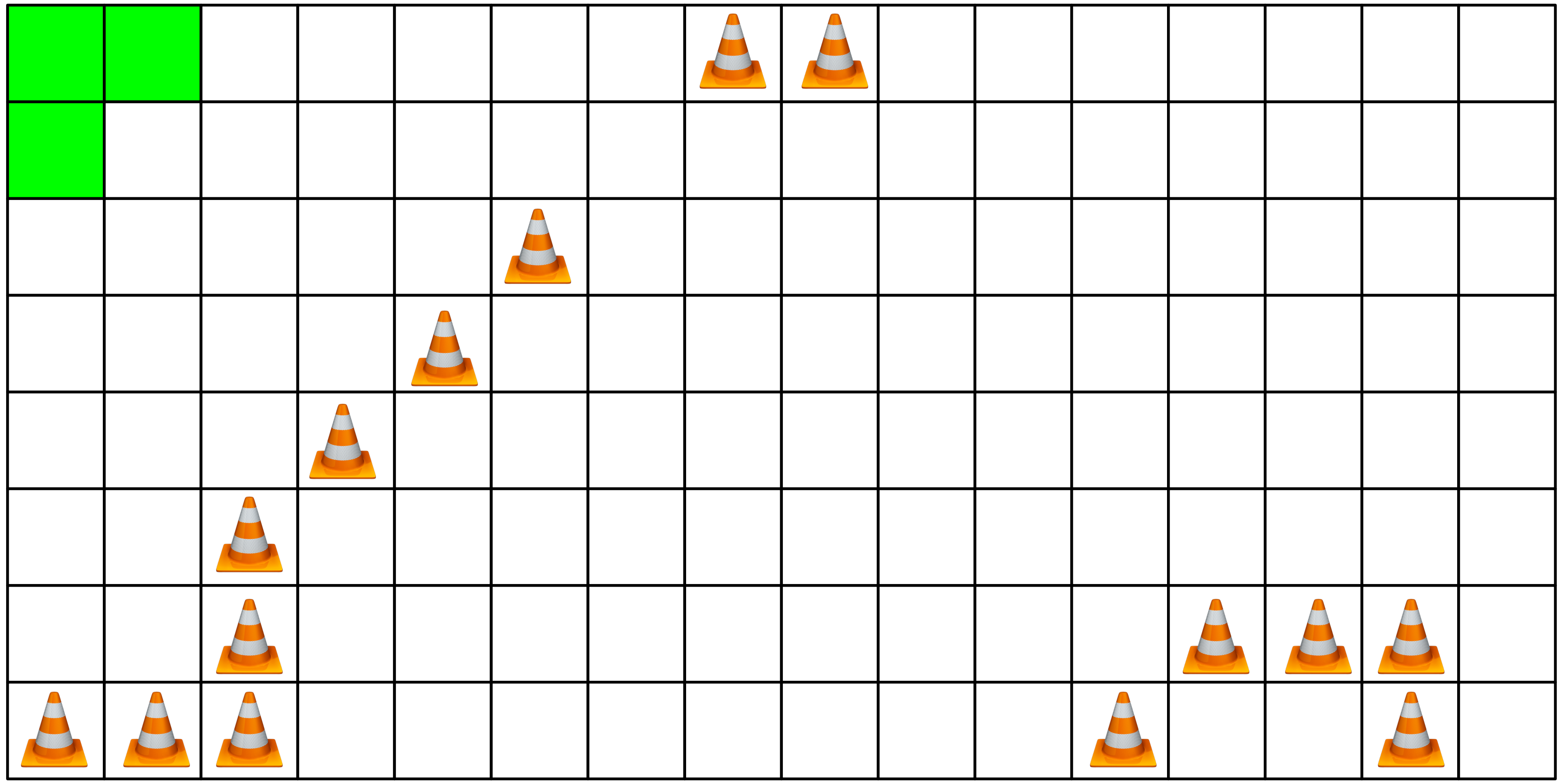
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1,1)$.
- Da ogni casella (i,j) è possibile visitare le caselle in $\{(i+1,j),(i,j+1),(i-1,j),(i,j-1)\}$, se esistono e **non ci sono ostacoli**.



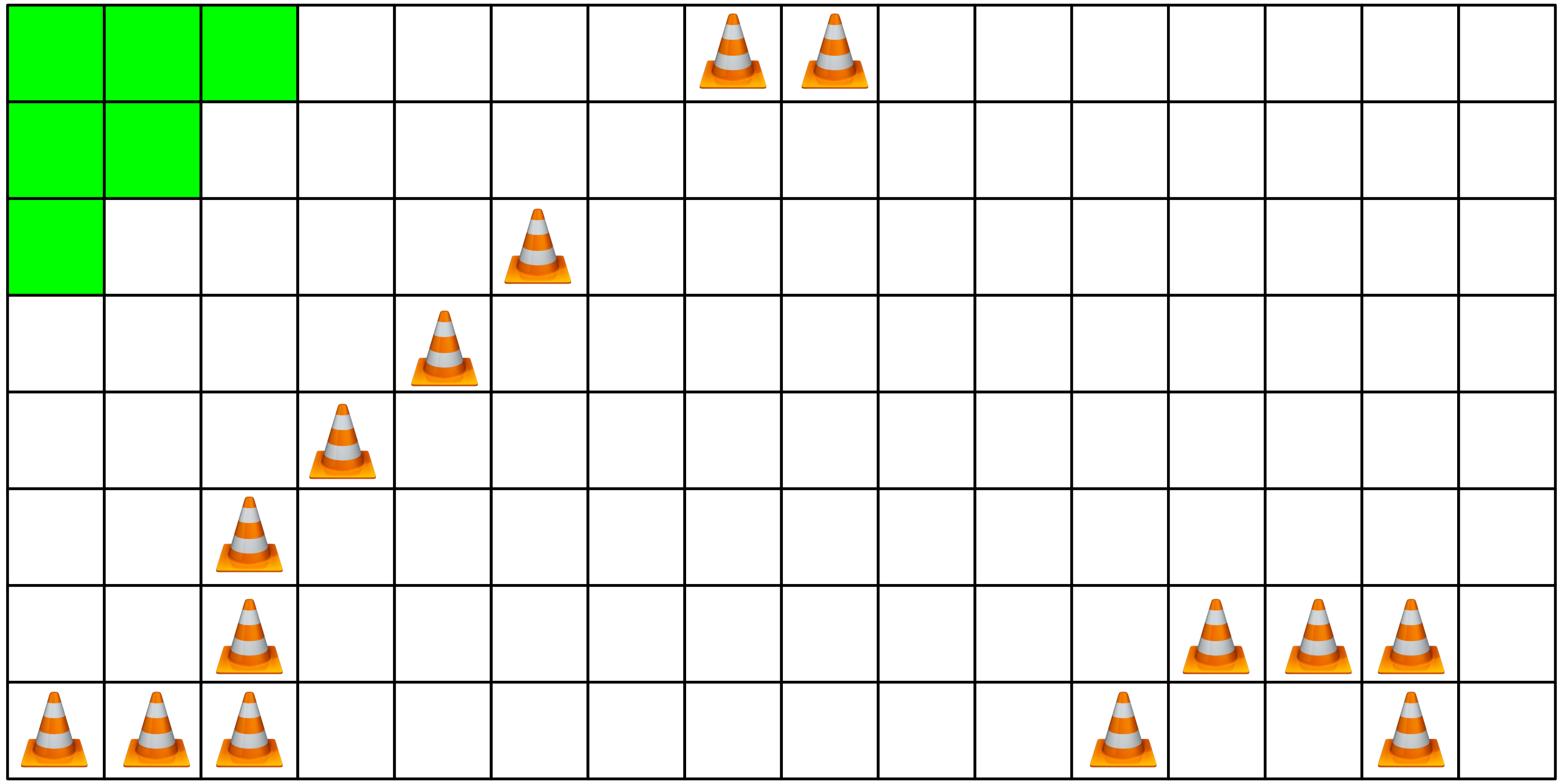
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i + 1, j), (i, j + 1), (i - 1, j), (i, j - 1)\}$, se esistono e **non ci sono ostacoli**.



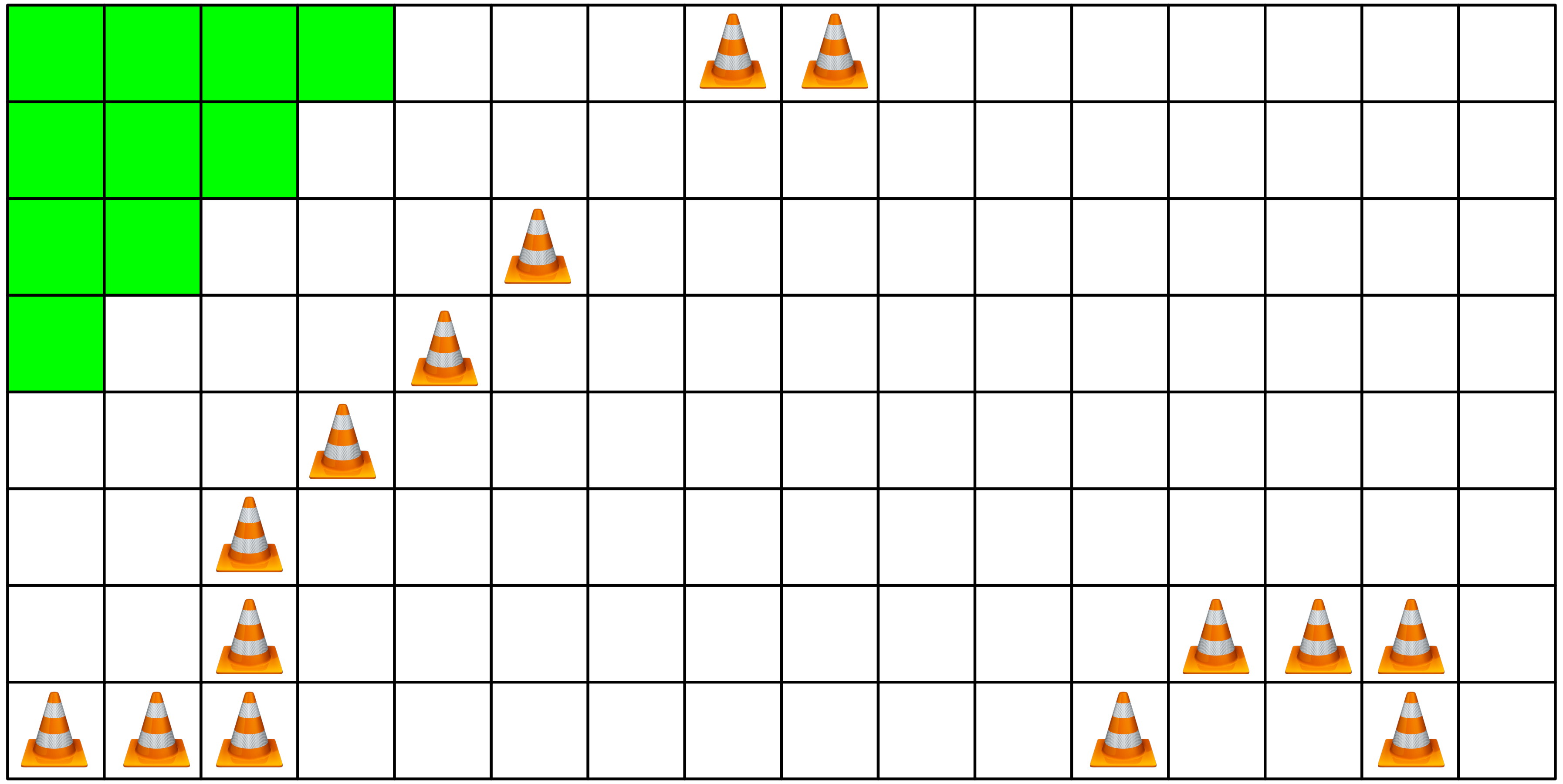
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



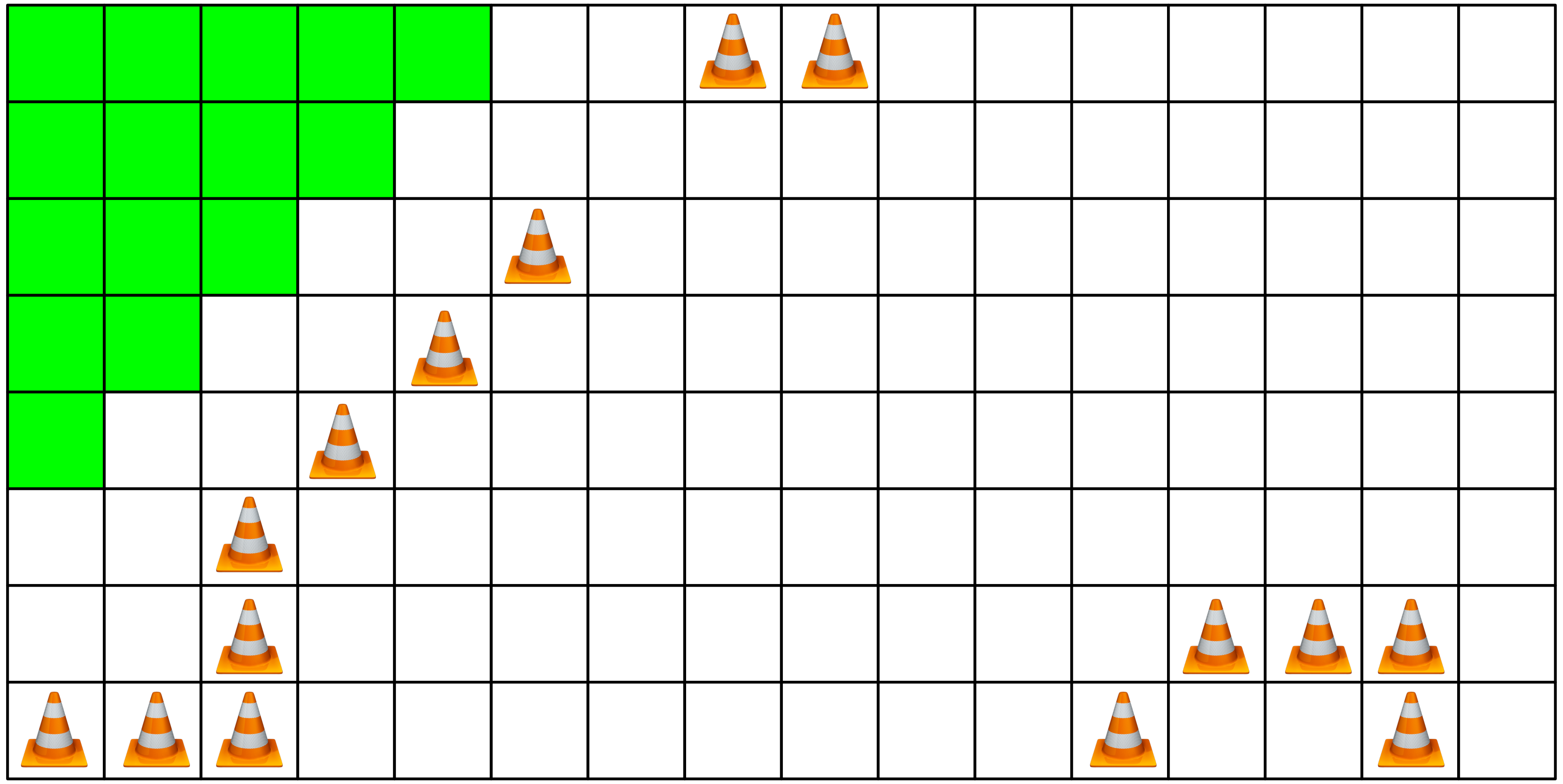
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



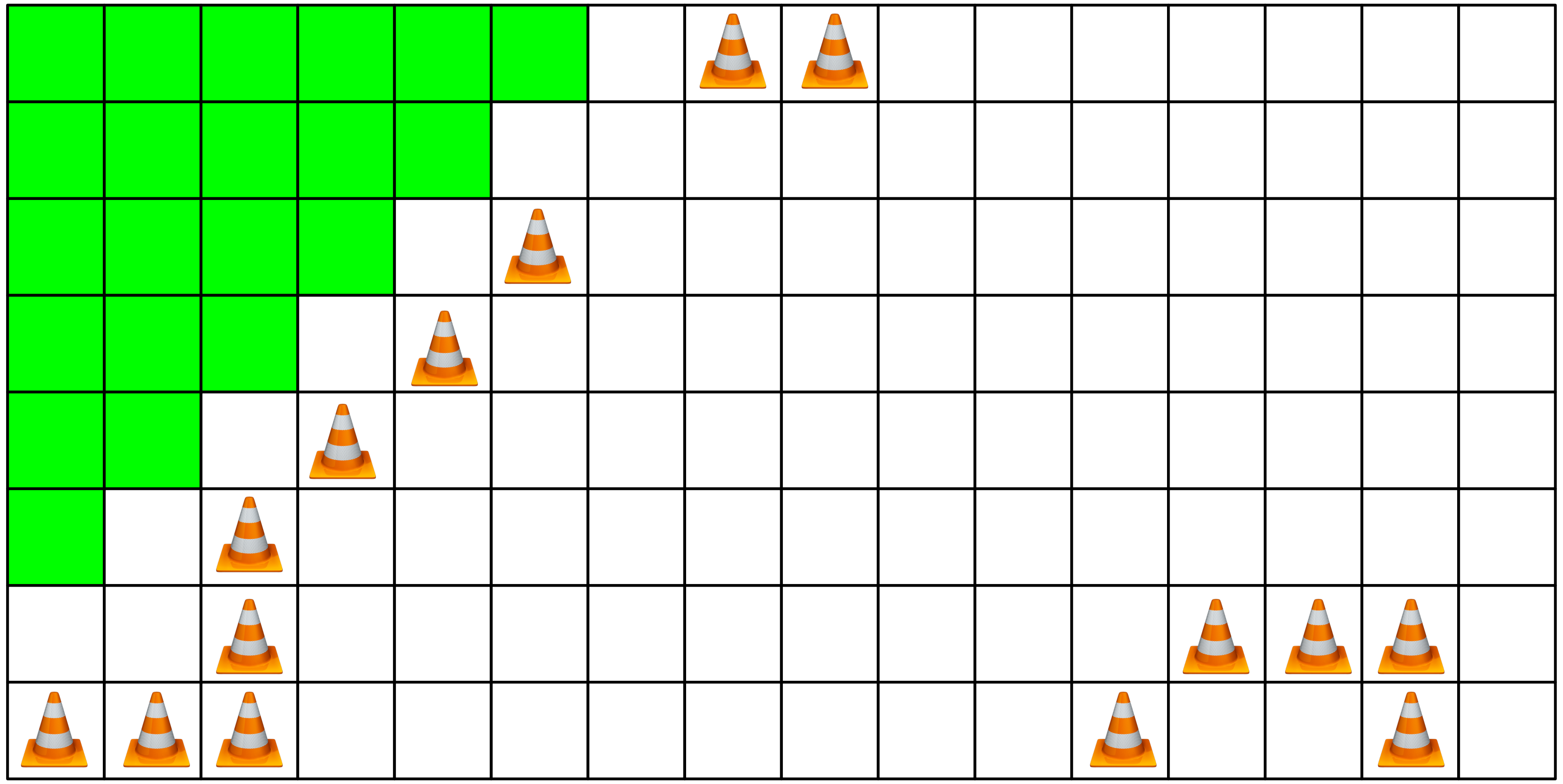
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



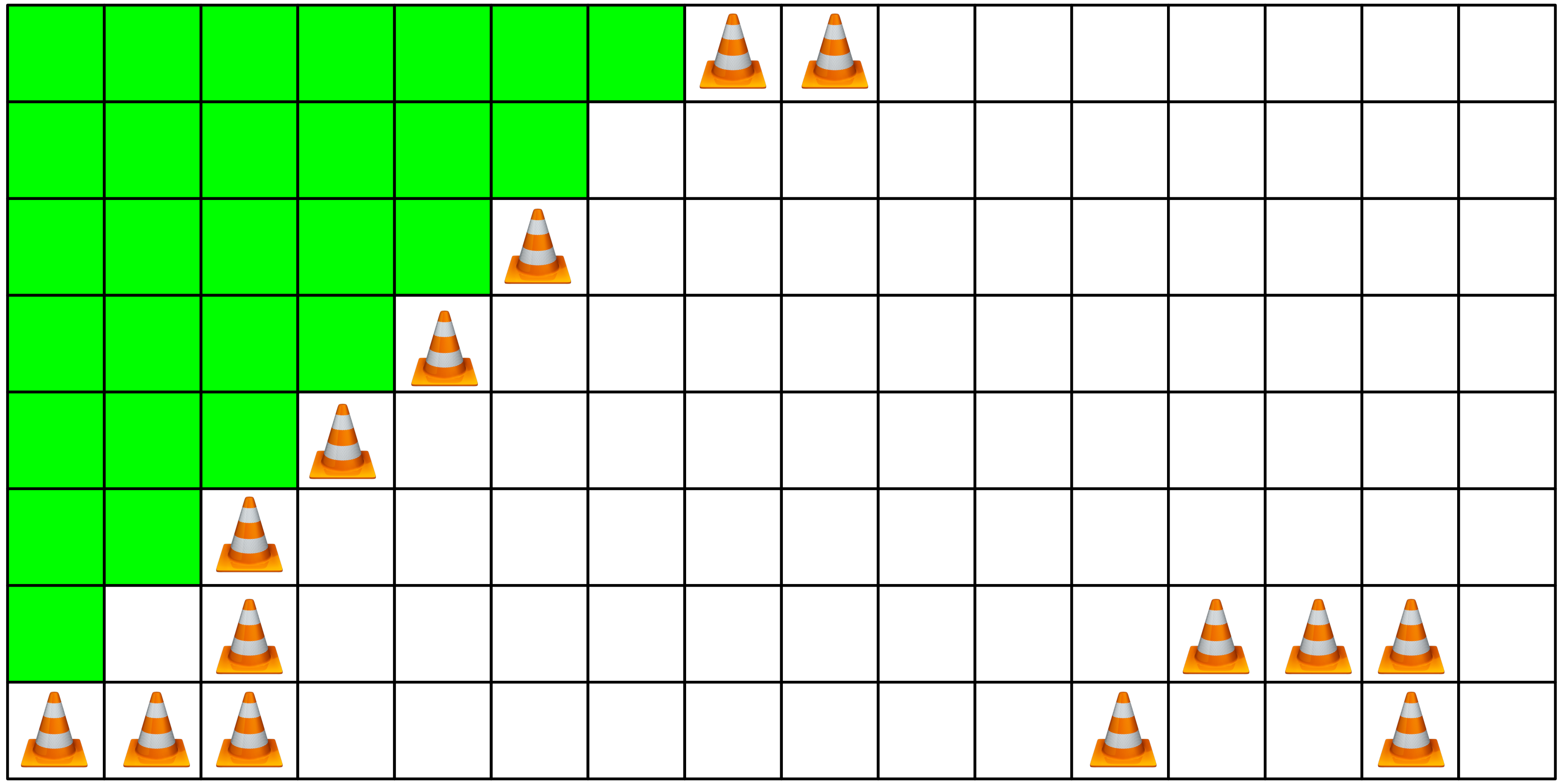
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1,1)$.
- Da ogni casella (i,j) è possibile visitare le caselle in $\{(i+1,j),(i,j+1),(i-1,j),(i,j-1)\}$, se esistono e **non ci sono ostacoli**.



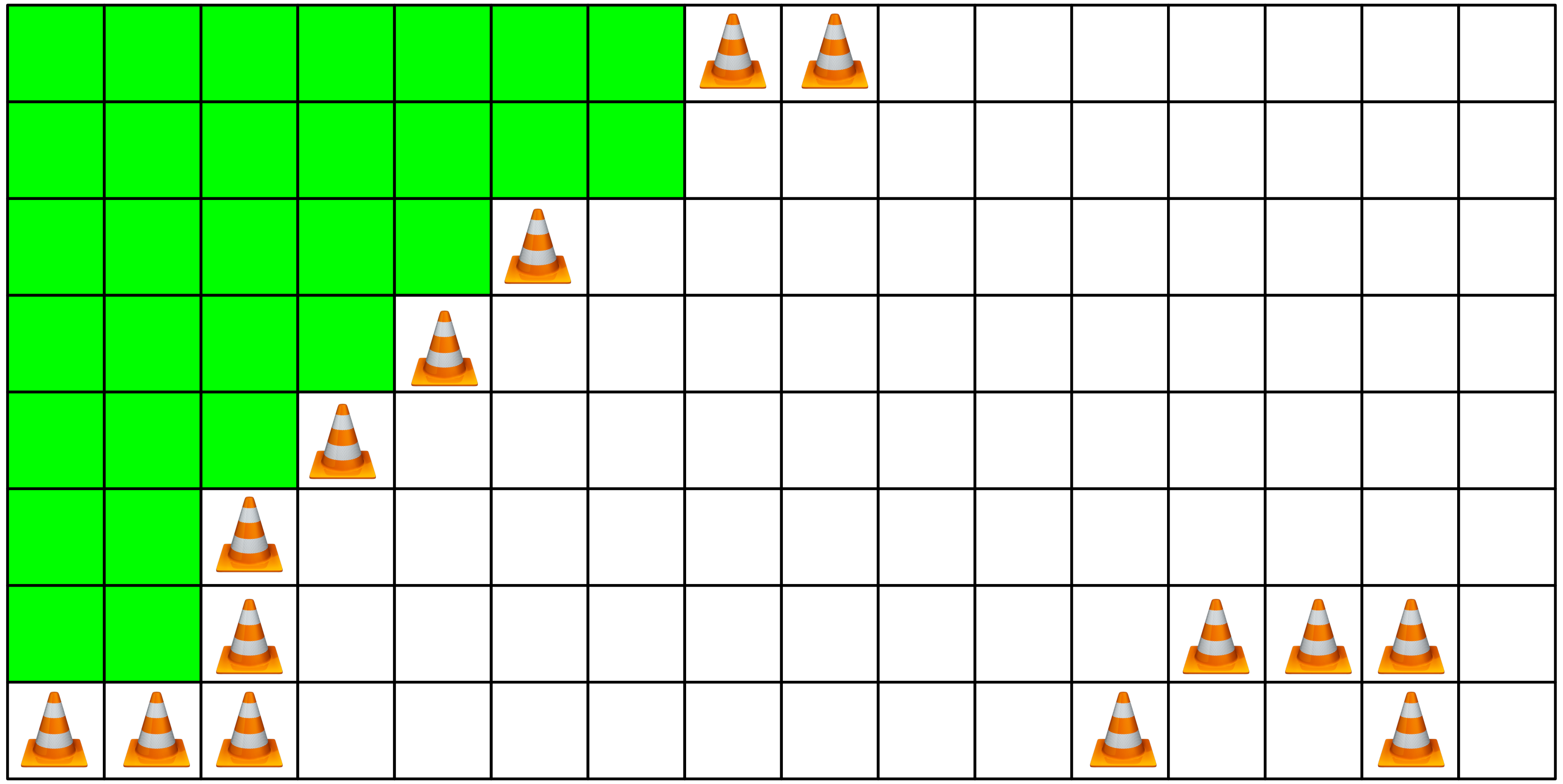
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1,1)$.
- Da ogni casella (i,j) è possibile visitare le caselle in $\{(i+1,j),(i,j+1),(i-1,j),(i,j-1)\}$, se esistono e **non ci sono ostacoli**.



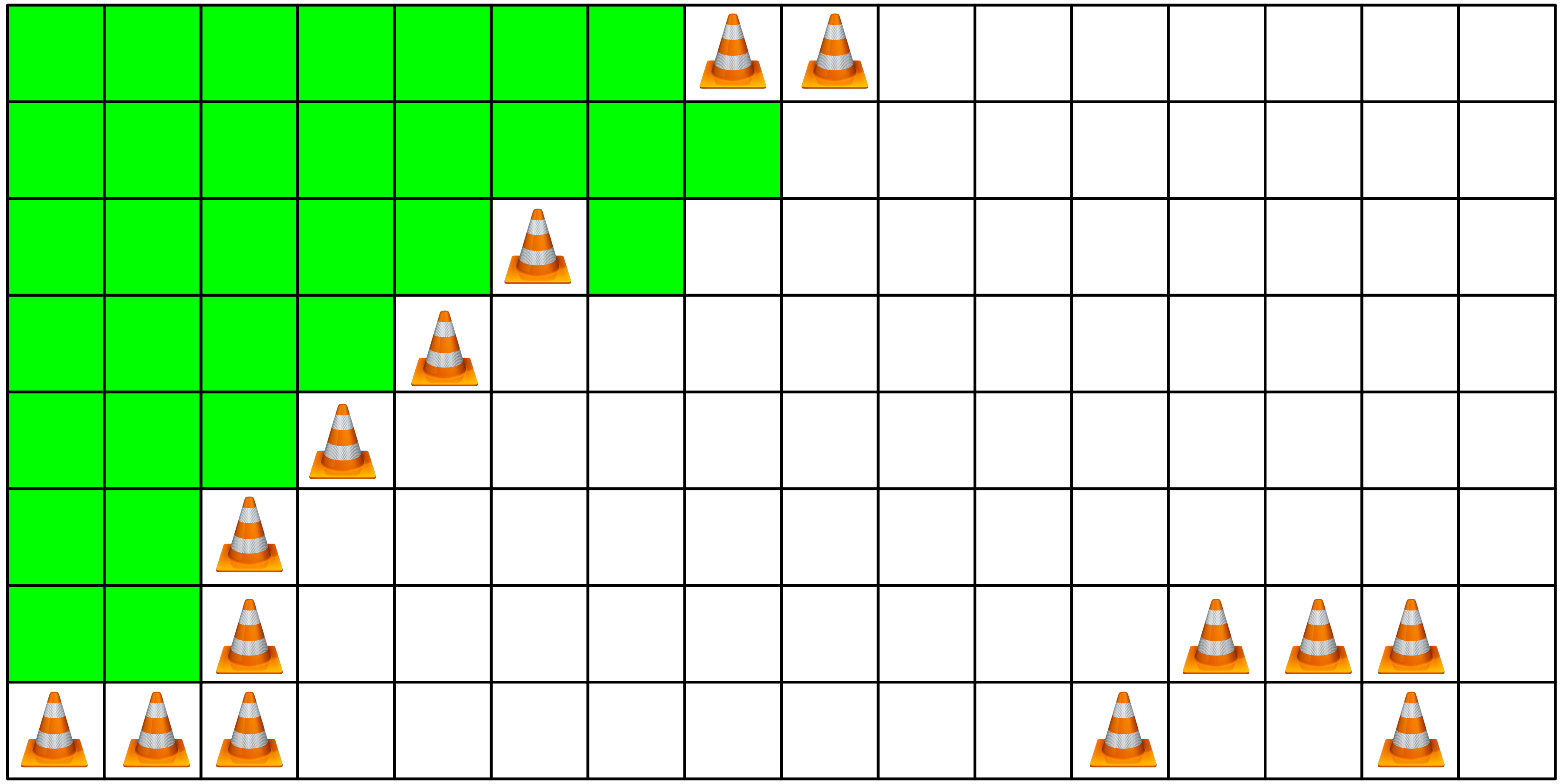
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



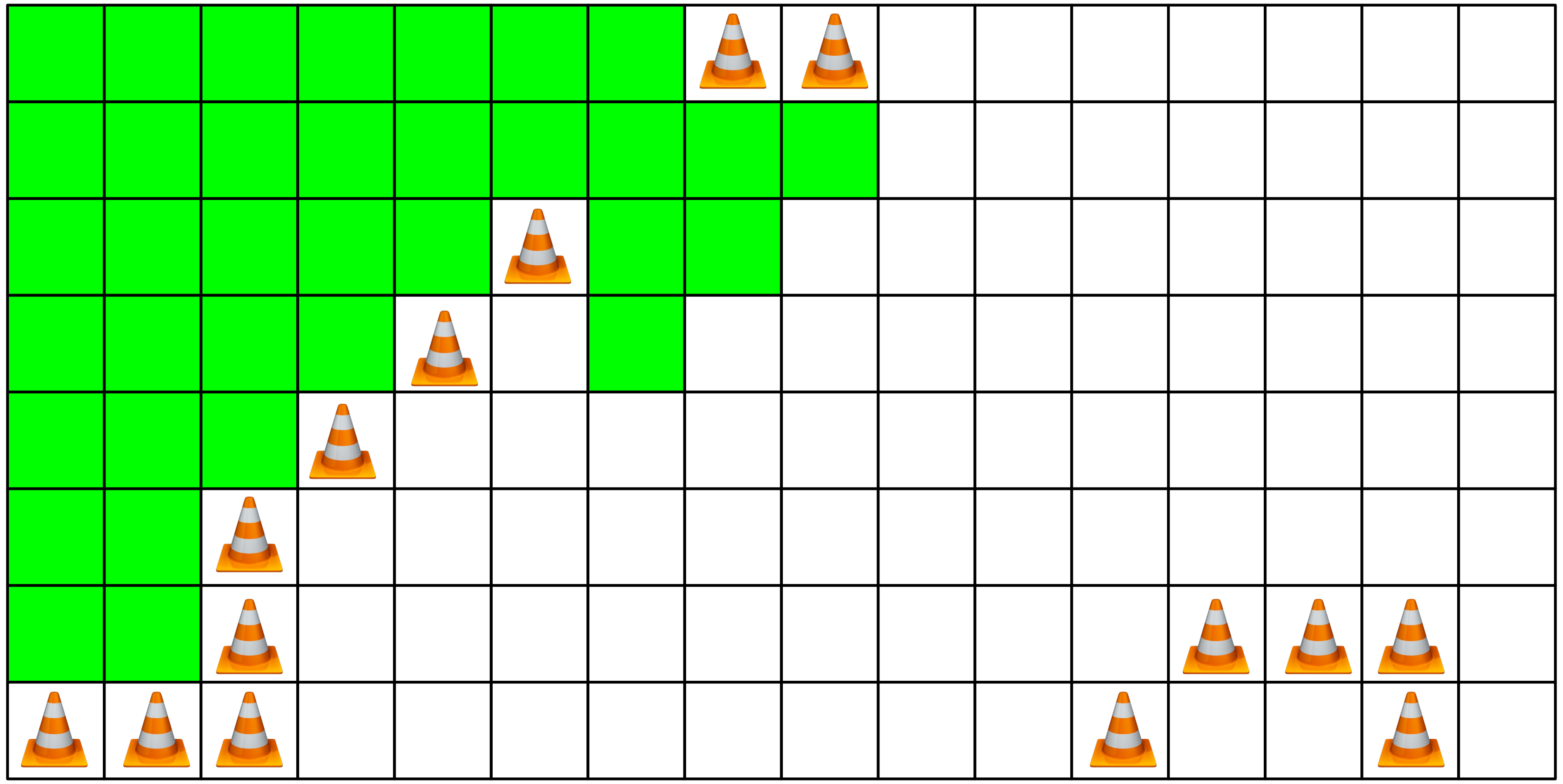
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



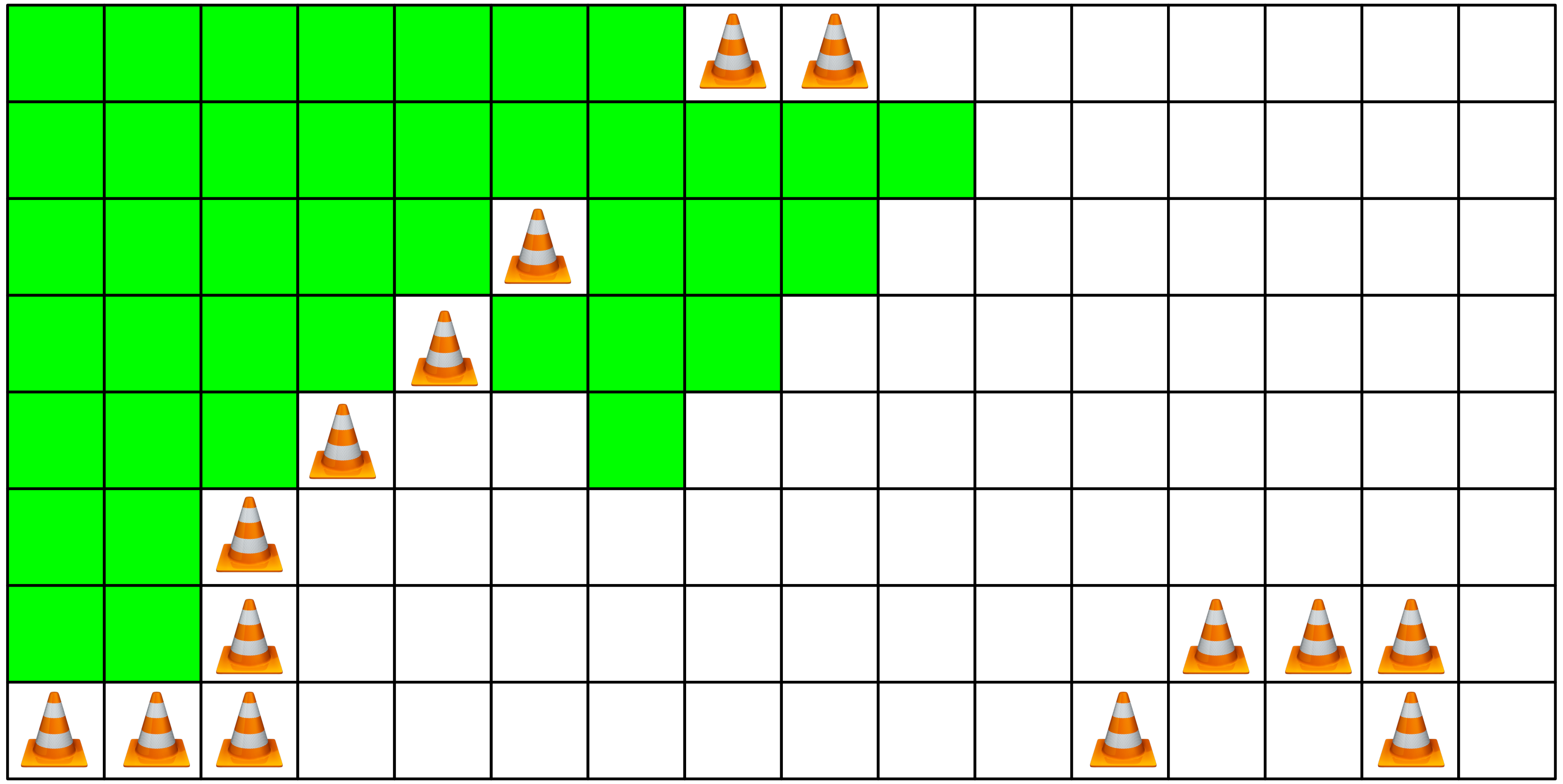
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



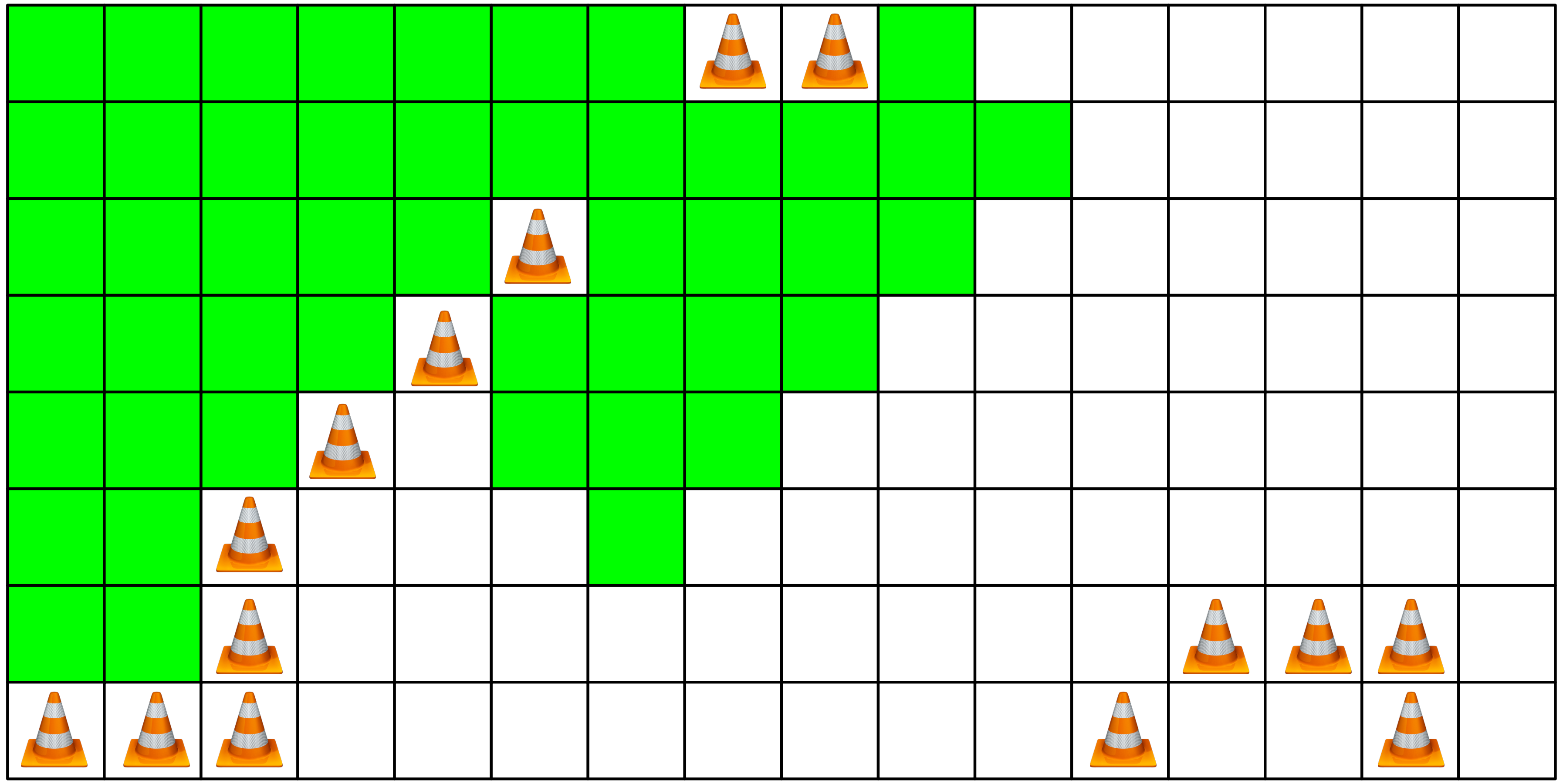
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



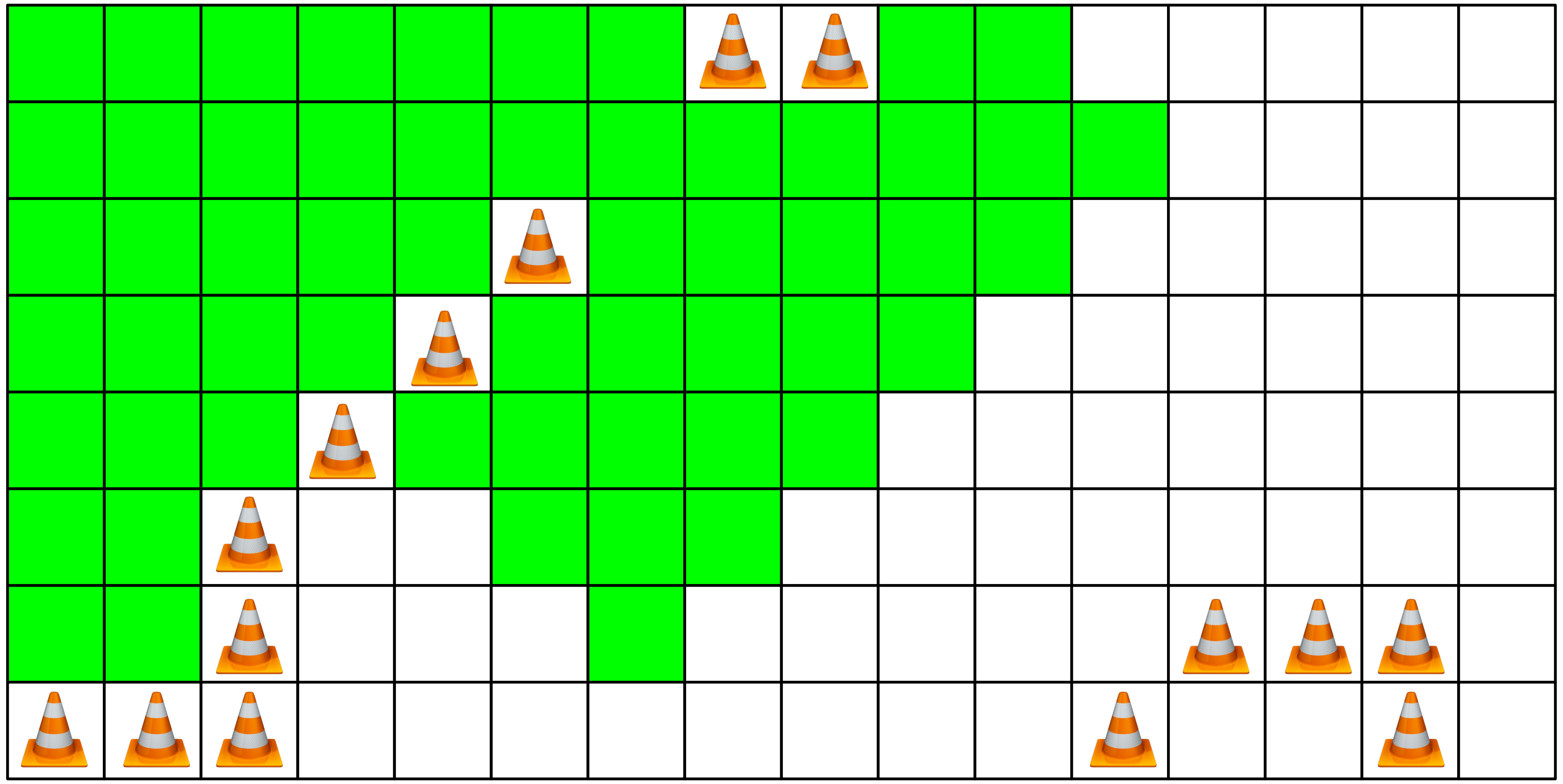
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1,1)$.
- Da ogni casella (i,j) è possibile visitare le caselle in $\{(i+1,j),(i,j+1),(i-1,j),(i,j-1)\}$, se esistono e **non ci sono ostacoli**.



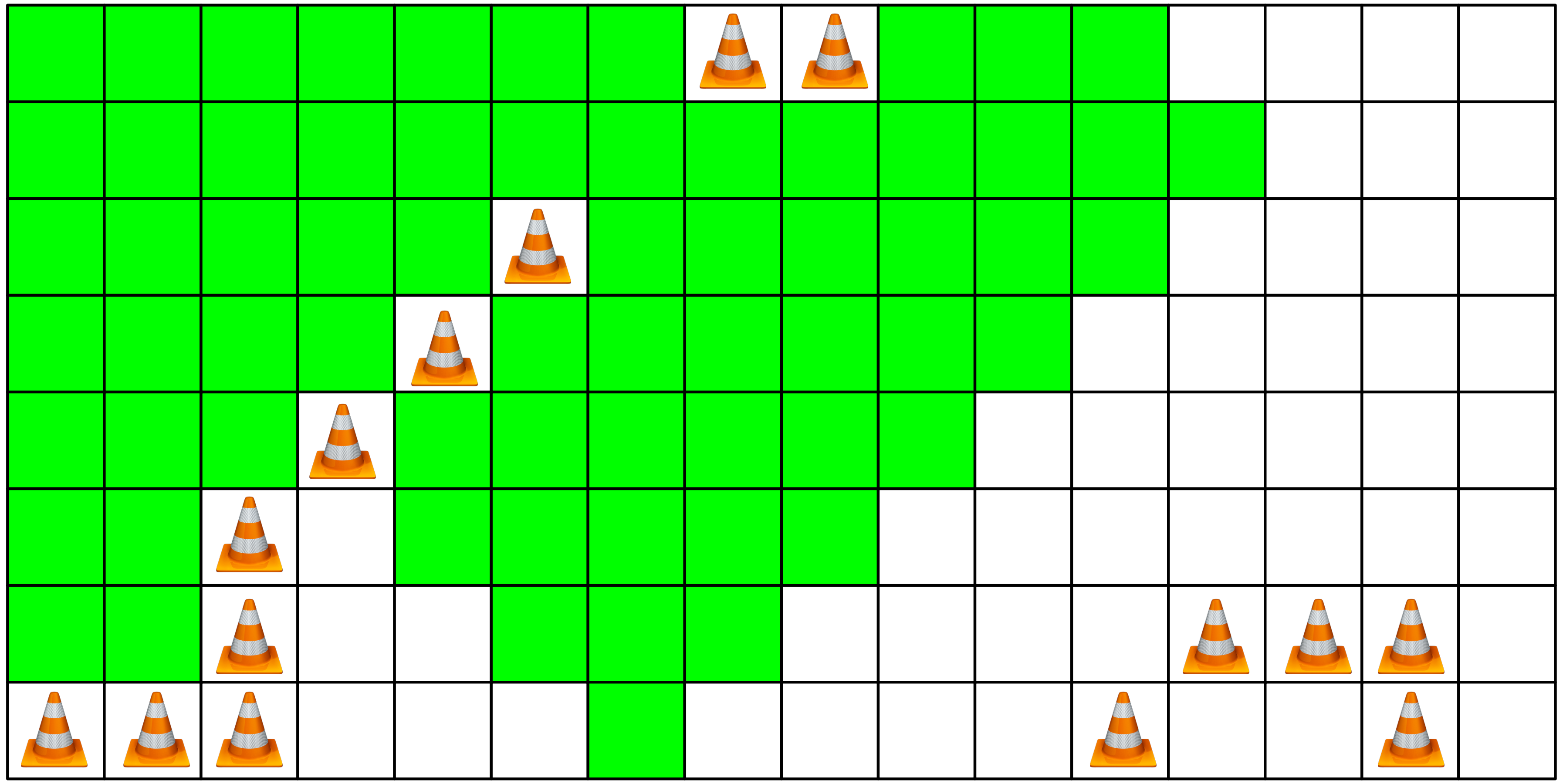
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1,1)$.
- Da ogni casella (i,j) è possibile visitare le caselle in $\{(i+1,j),(i,j+1),(i-1,j),(i,j-1)\}$, se esistono e **non ci sono ostacoli**.



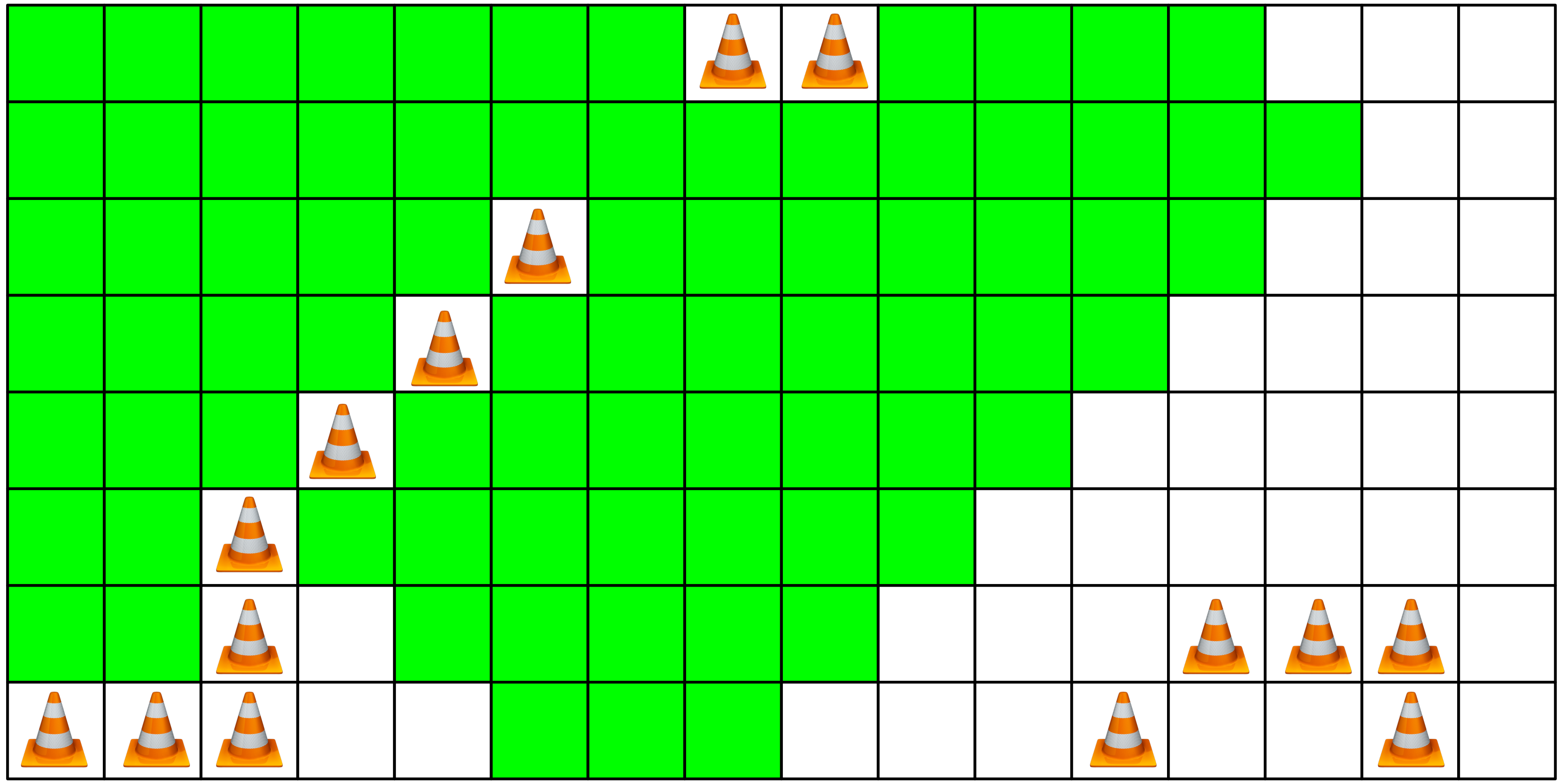
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



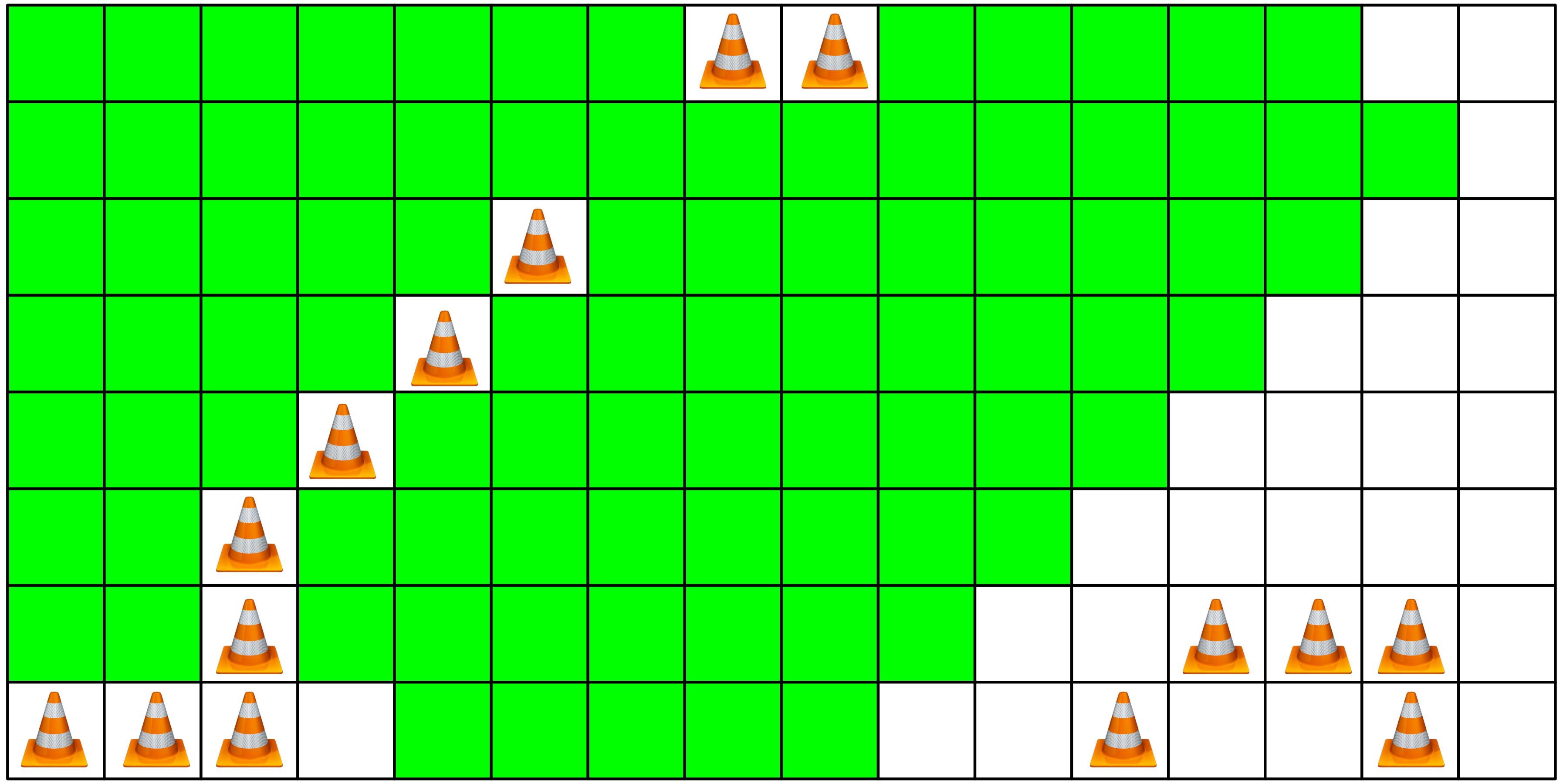
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1,1)$.
- Da ogni casella (i,j) è possibile visitare le caselle in $\{(i+1,j),(i,j+1),(i-1,j),(i,j-1)\}$, se esistono e **non ci sono ostacoli**.



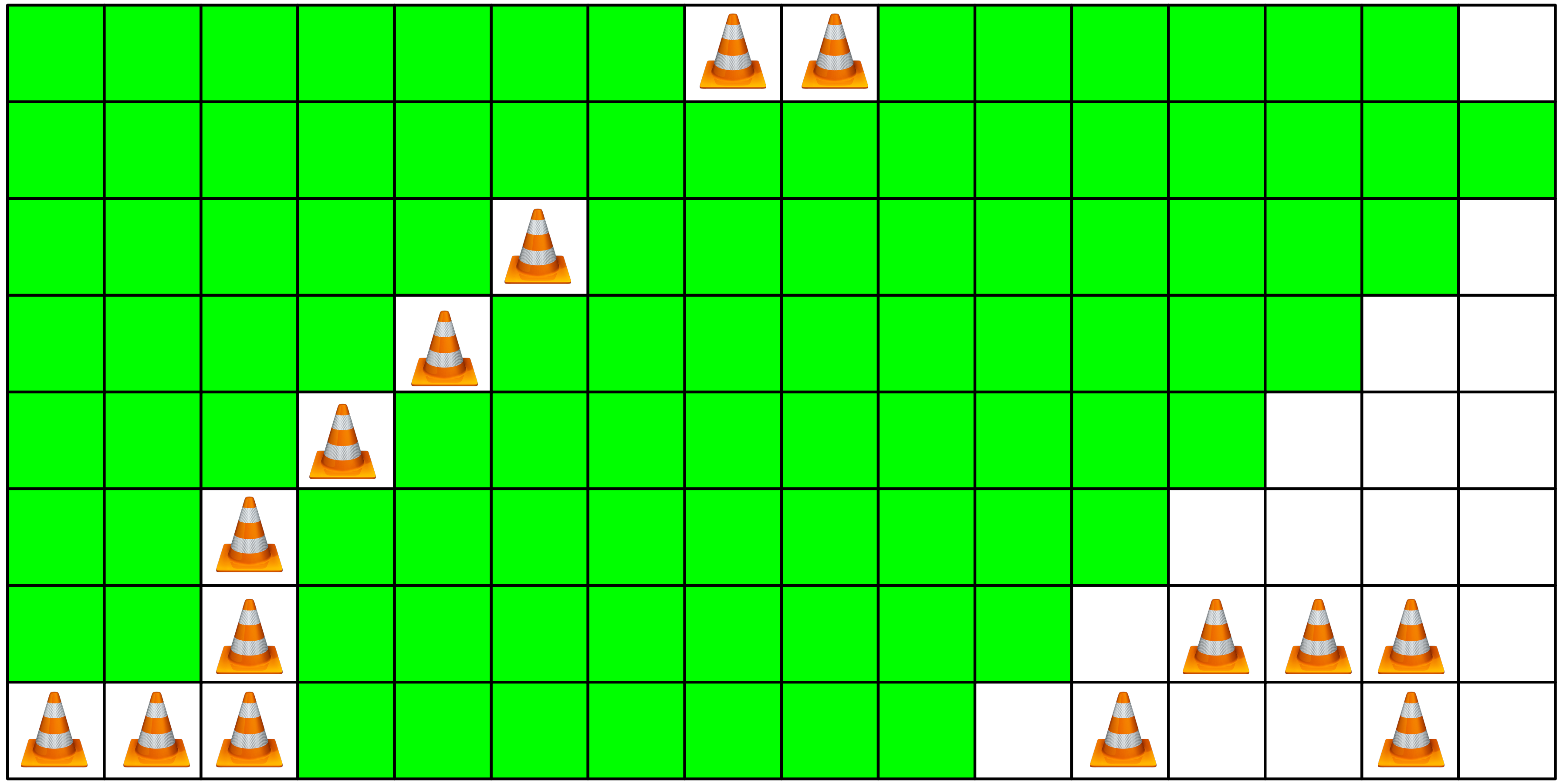
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



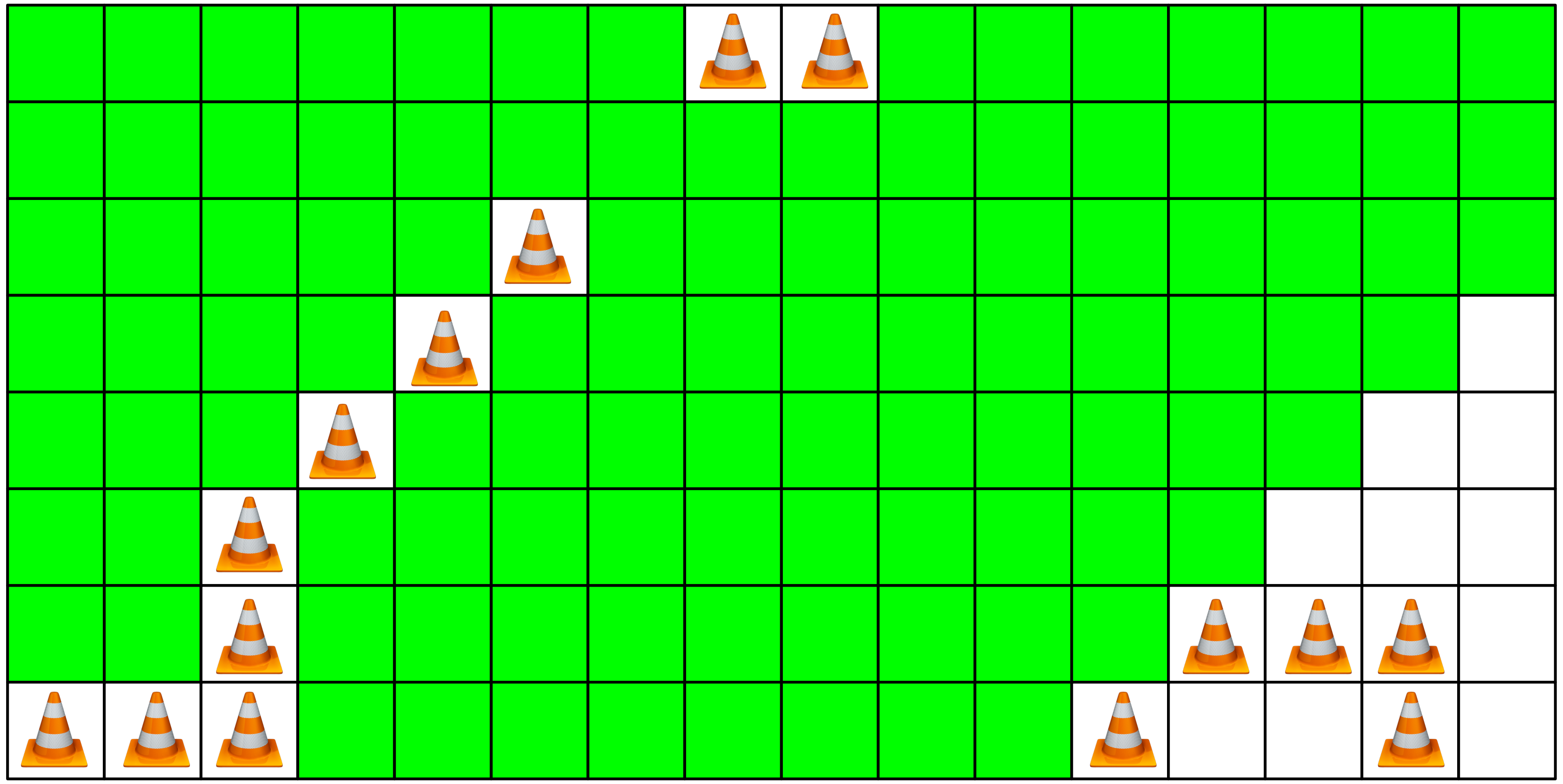
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



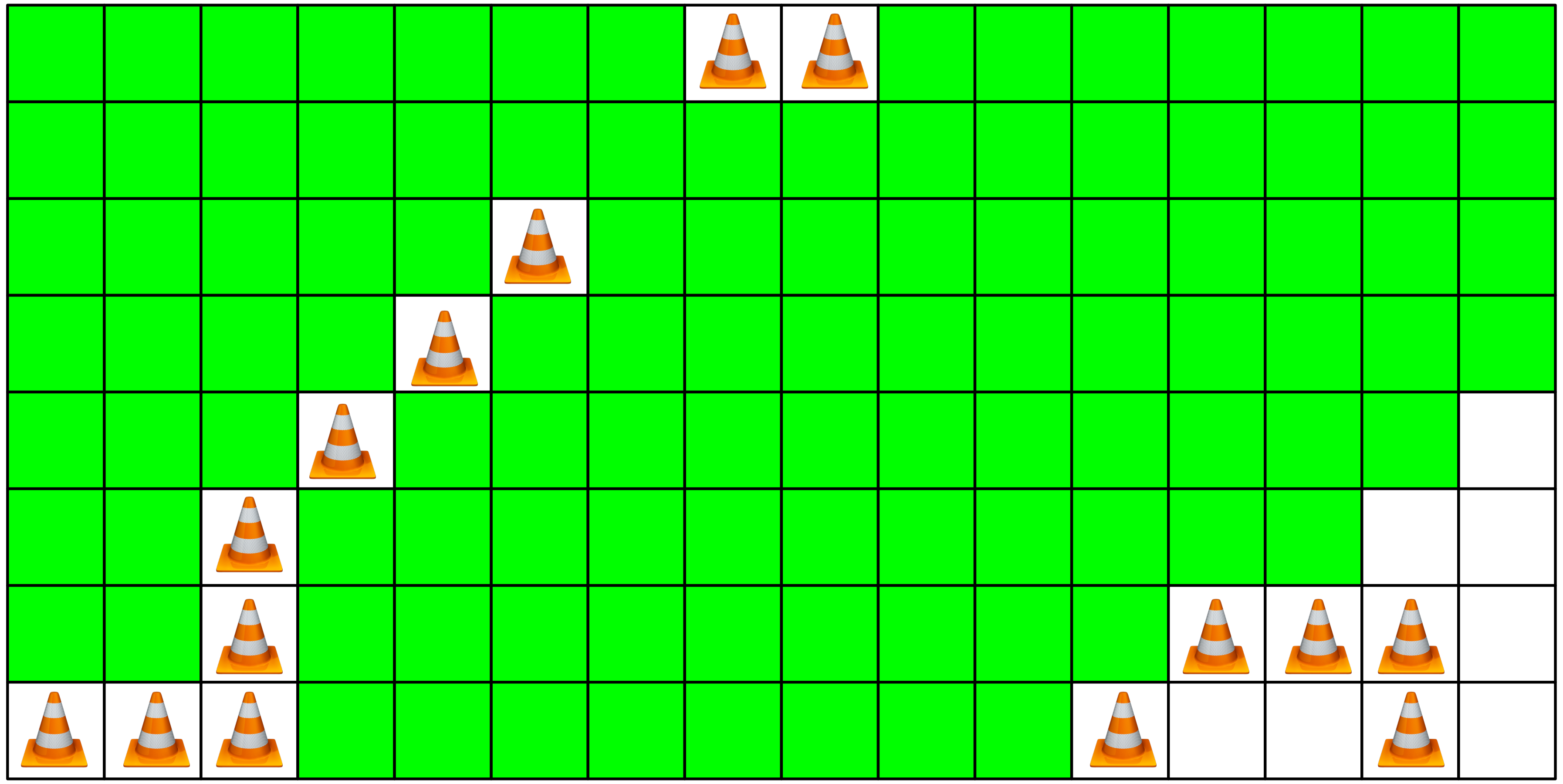
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



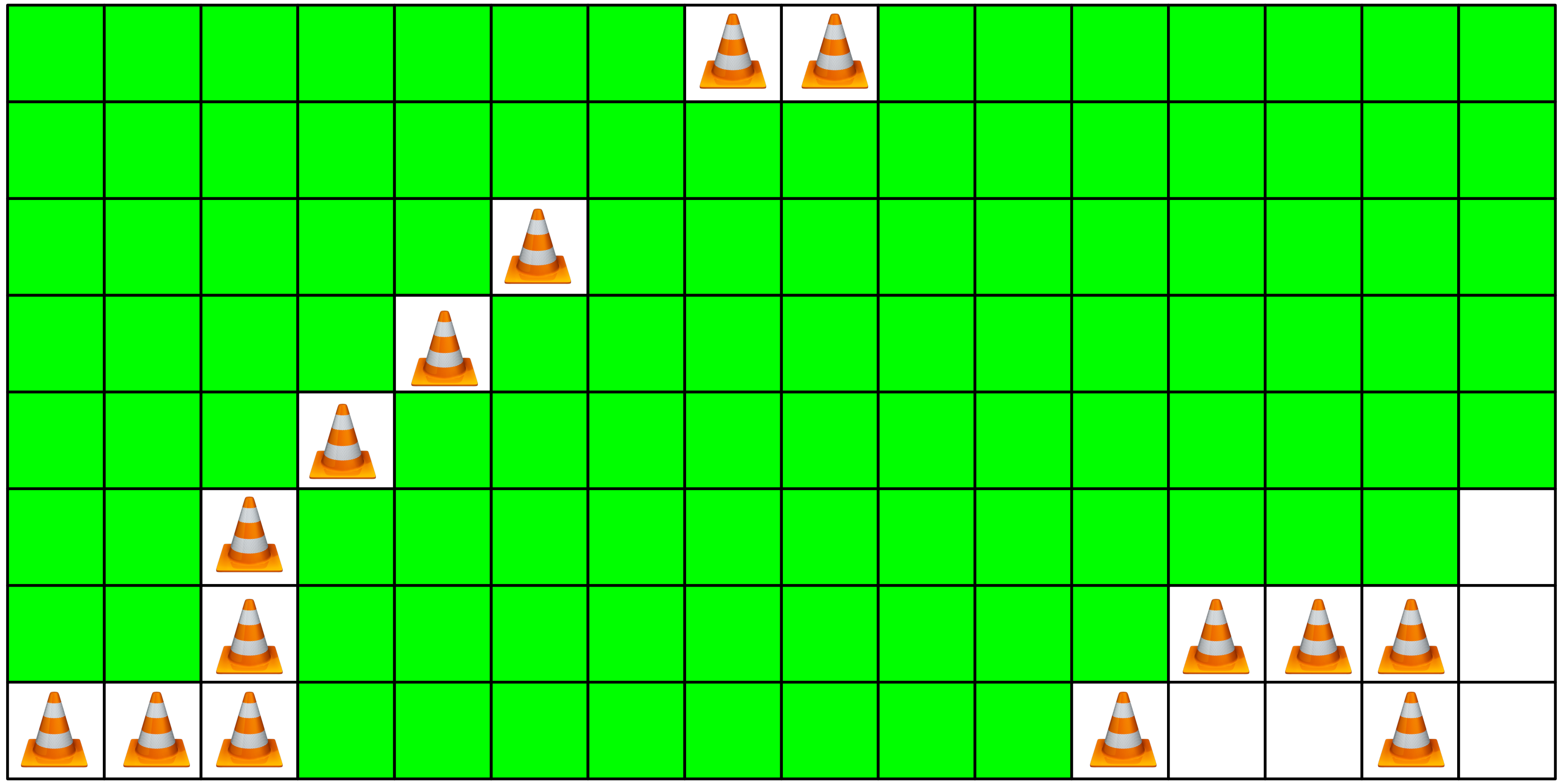
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



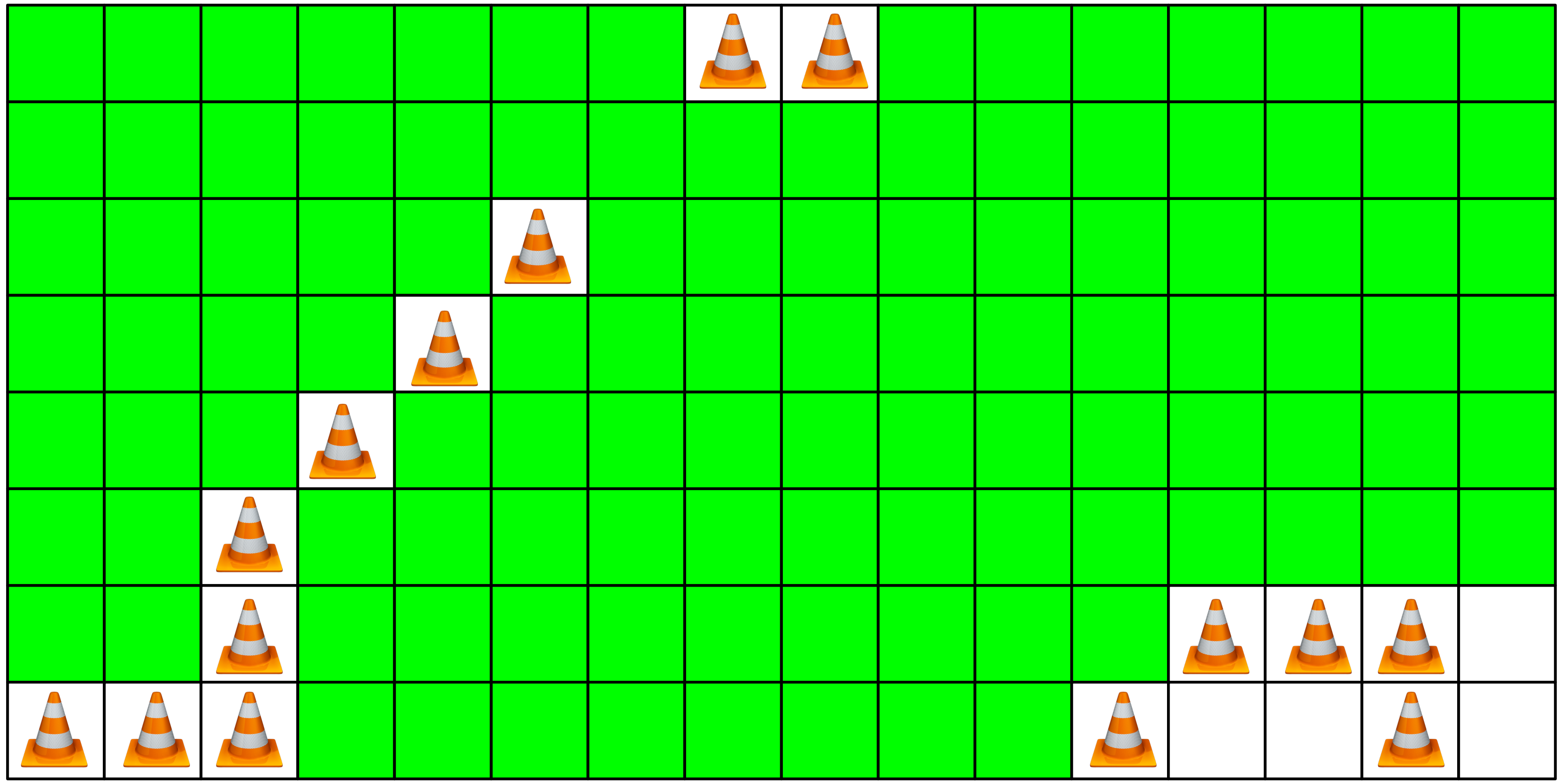
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



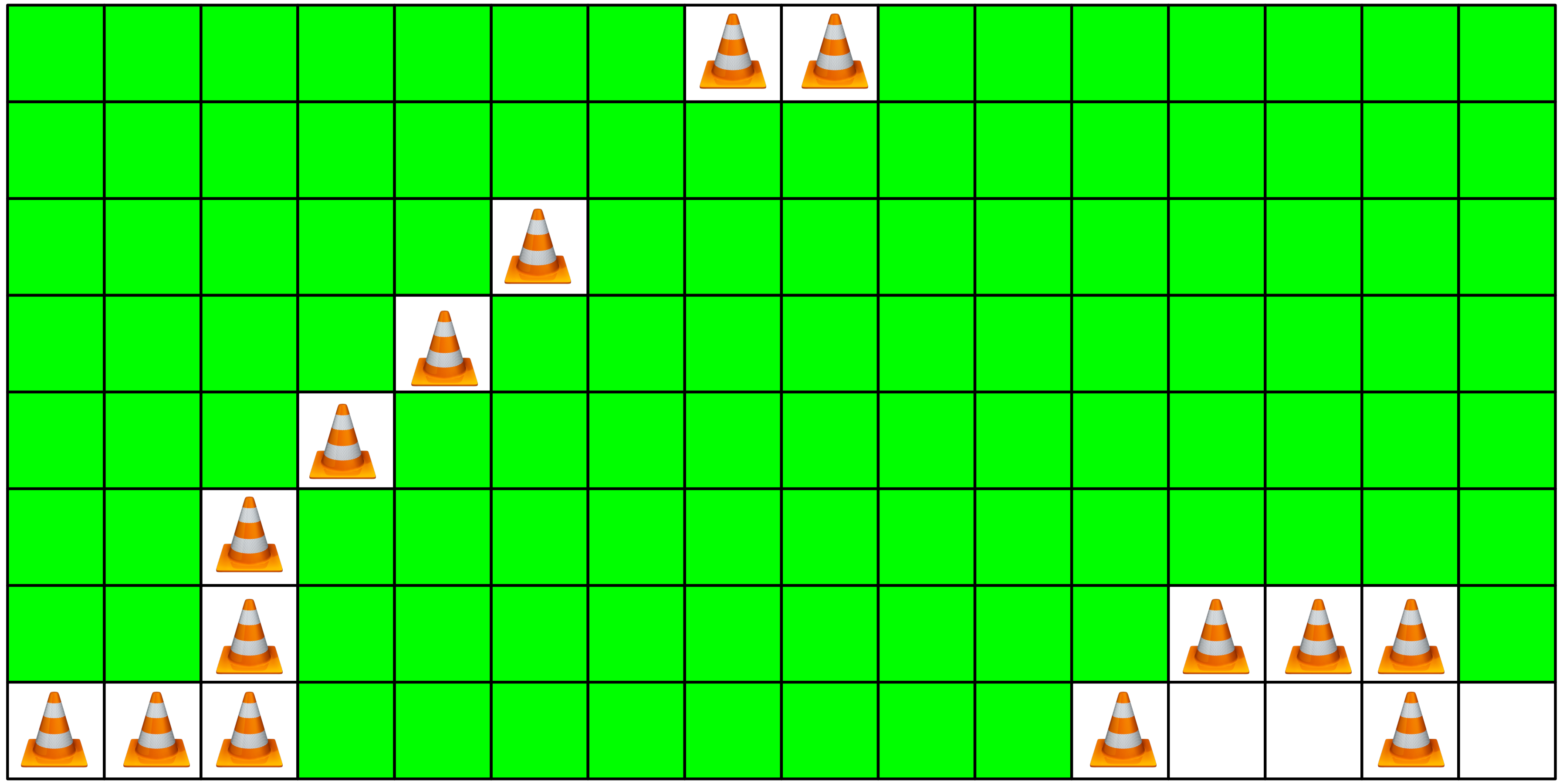
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



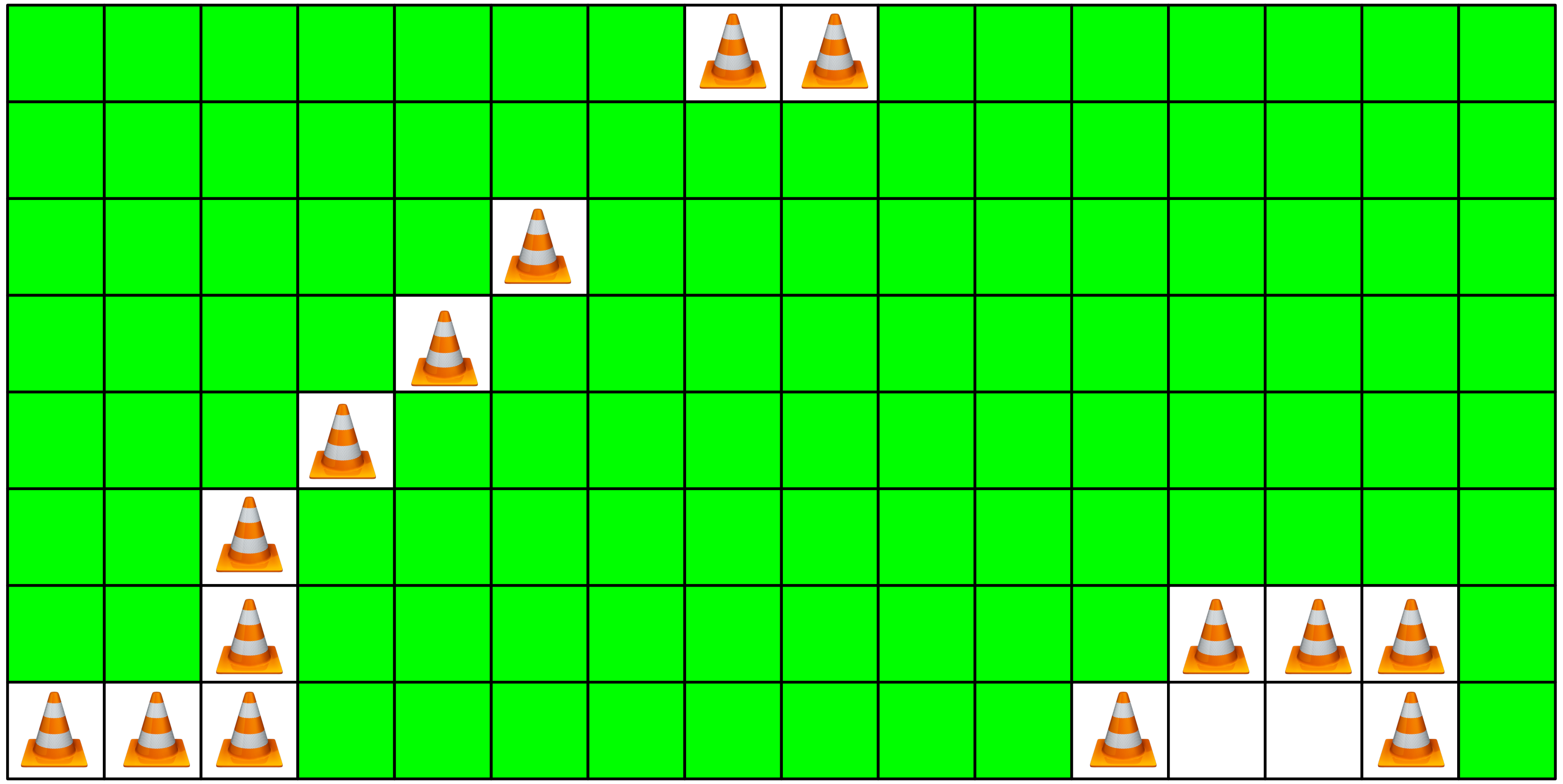
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



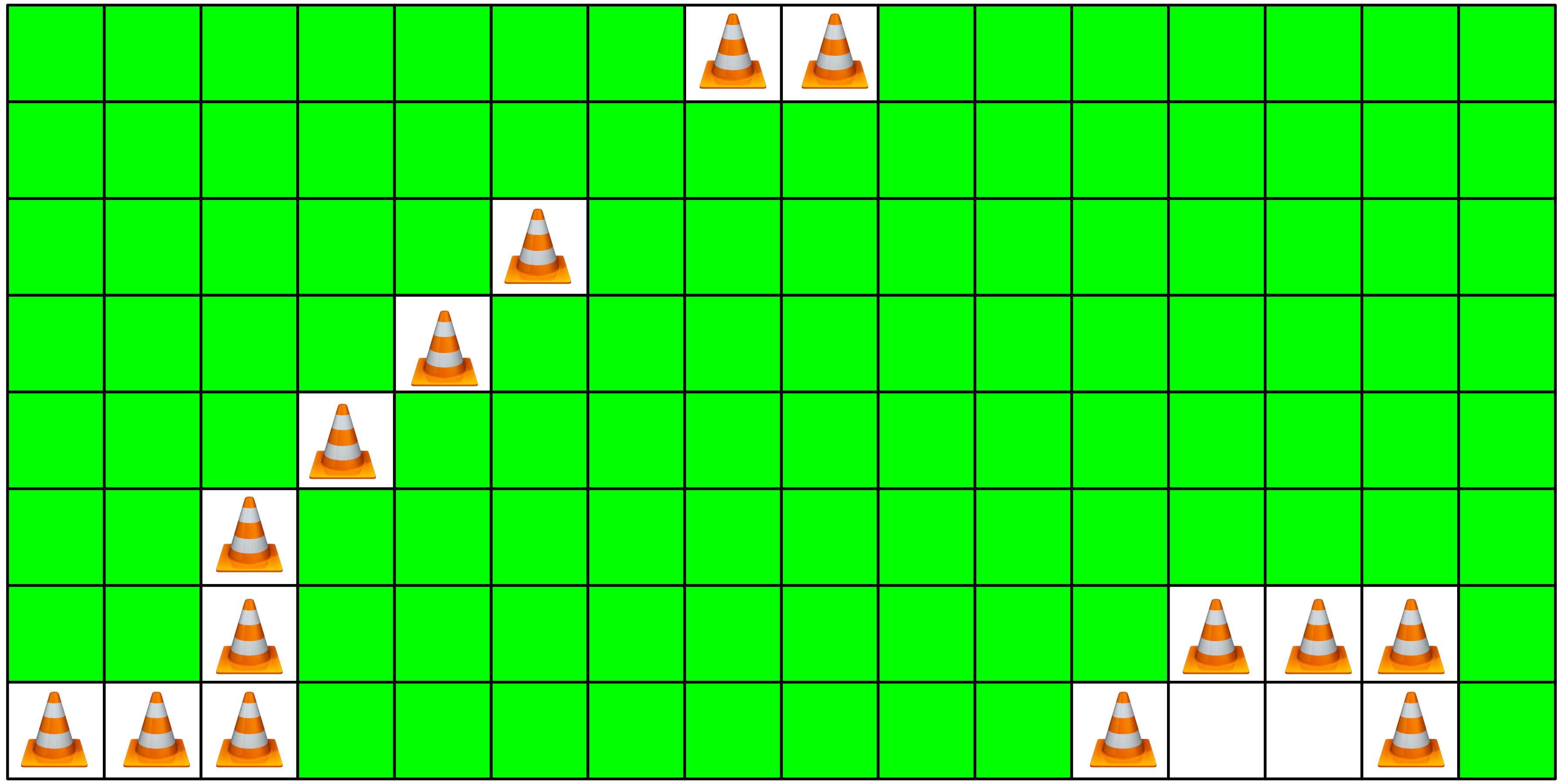
Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



Verificatore lineare

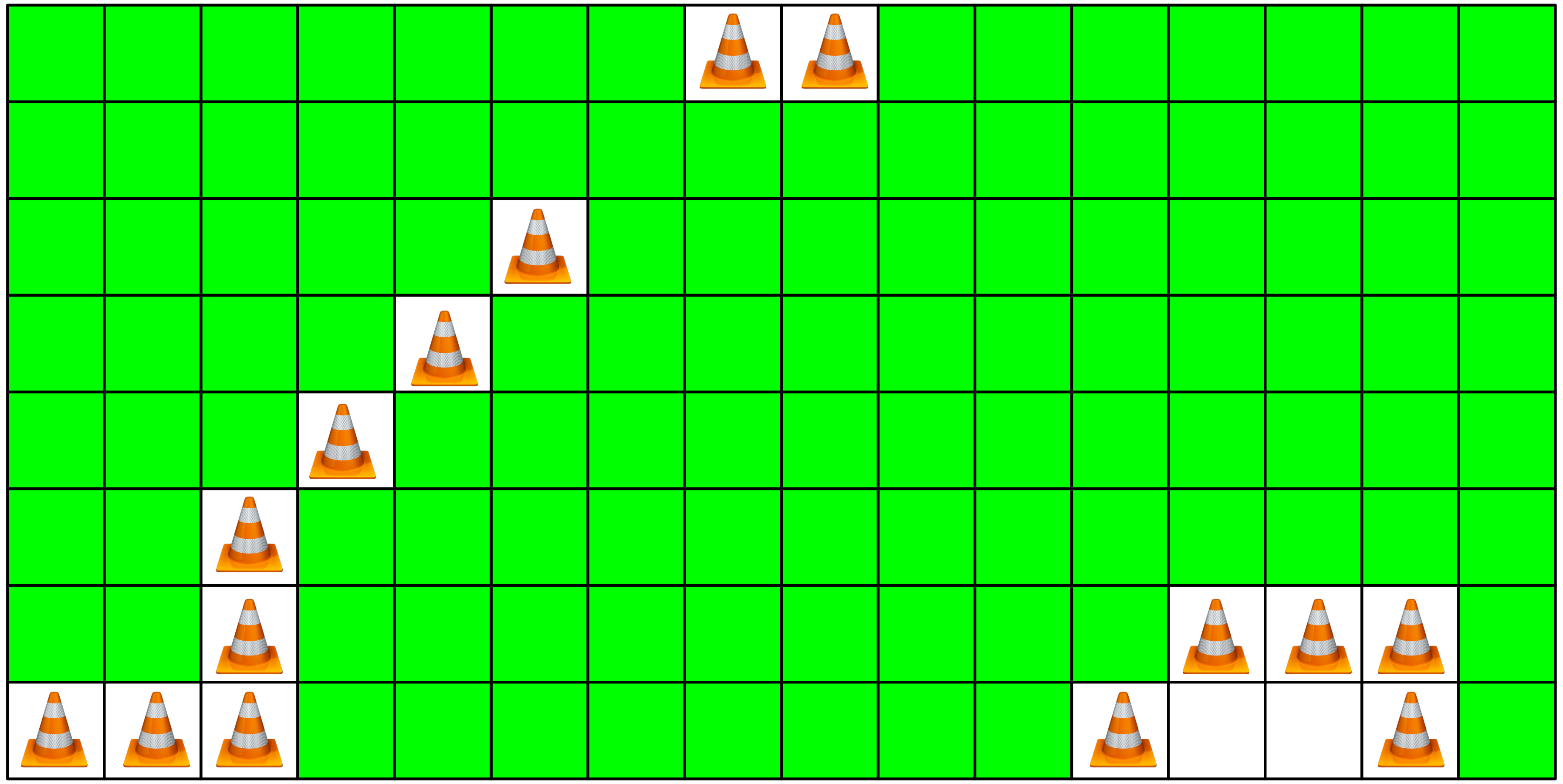
- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



È possibile raggiungere (n, m) da $(1, 1)$.

Verificatore lineare

- Lancia una visita (per esempio **BFS/DFS**) con casella sorgente $(1, 1)$.
- Da ogni casella (i, j) è possibile visitare le caselle in $\{(i+1, j), (i, j+1), (i-1, j), (i, j-1)\}$, se esistono e **non ci sono ostacoli**.



È possibile raggiungere (n, m) da $(1, 1)$.
Complessità temporale: $O(nm)$

Forza bruta

- Sia $O = \{(i,j) \in (\{1,2,\ldots,n\} \times \{1,2,\ldots,m\}) : A_{ij} = 1\}$ l'insieme degli ostacoli.

ALGORITMO: FORZABRUTA

- For $i \leftarrow 0$ to $n \cdot m$
  - For each set $O_i \in \binom{O}{i}$

$$A'_{ij} = \begin{cases} 0 & \text{if } (i,j) \in O_i, \\ A_{ij} & \text{otherwise} \end{cases}$$

se esiste un path **ammissibile** da $(1,1)$ a (n,m) rispetto ad A'_{ij} **return** i

- **Complessità temporale?**

$$T(n,m) \geq \sum_{i=0}^{n \cdot m} \binom{n \cdot m}{i} = 2^{n \cdot m}$$

Forza bruta

- Sia $O = \{(i,j) \in (\{1,2,\ldots,n\} \times \{1,2,\ldots,m\}) : A_{ij} = 1\}$ l'insieme degli ostacoli.

ALGORITMO: FORZABRUTA

- For $i \leftarrow 0$ to $n \cdot m$
  - For each set $O_i \in \binom{O}{i}$

$$A'_{ij} = \begin{cases} 0 & \text{if } (i,j) \in O_i, \\ A_{ij} & \text{otherwise} \end{cases}$$

se esiste un path **ammissibile** da $(1,1)$ a (n,m) rispetto ad A'_{ij} **return** i

È possibile maggiorare la soluzione ottima?

- **Complessità temporale?**

$$T(n,m) \geq \sum_{i=0}^{n \cdot m} \binom{n \cdot m}{i} = 2^{n \cdot m}$$

Forza bruta

- Sia $O = \{(i,j) \in (\{1,2,\ldots,n\} \times \{1,2,\ldots,m\}) : A_{ij} = 1\}$ l'insieme degli ostacoli.

ALGORITMO: FORZABRUTA

- For $i \leftarrow 0$ to $n \cdot m$
  - For each set $O_i \in \binom{O}{i}$

$$A'_{ij} = \begin{cases} 0 & \text{if } (i,j) \in O_i, \\ A_{ij} & \text{otherwise} \end{cases}$$

se esiste un path **ammissibile** da $(1,1)$ a (n,m) rispetto ad A'_{ij} **return** i

È possibile maggiorare la soluzione ottima?
Conseguentemente ridurre le i da considerare

- **Complessità temporale?**

$$T(n,m) \geq \sum_{i=0}^{n \cdot m} \binom{n \cdot m}{i} = 2^{n \cdot m}$$

Lemma

Sia O^* l'insieme di ostacoli ottimi da rimuovere rispetto a un'istanza di taglia $n \cdot m$.

$|O^*| \leq n + m$

Lemma

Sia O^* l'insieme di ostacoli ottimi da rimuovere rispetto a un'istanza di taglia $n \cdot m$.
$|O^*| \leq n + m$

Proof

Lemma

Sia O^* l'insieme di ostacoli ottimi da rimuovere rispetto a un'istanza di taglia $n \cdot m$.
$|O^*| \leq n + m$

Proof

Si consideri un qualunque cammino minimo da $(1, 1)$ a (n, m). Esso passa attraverso $n + m - 1$ caselle.

Lemma

Sia O^* l'insieme di ostacoli ottimi da rimuovere rispetto a un'istanza di taglia $n \cdot m$.
$|O^*| \leq n + m$

Proof

Si consideri un qualunque cammino minimo da $(1, 1)$ a (n, m). Esso passa attraverso $n + m - 1$ caselle. Nel caso peggiore tutte le caselle facenti parte del cammino minimo sono ostacoli.

Lemma

Sia O^* l'insieme di ostacoli ottimi da rimuovere rispetto a un'istanza di taglia $n \cdot m$.
$|O^*| \leq n + m$

Proof

Si consideri un qualunque cammino minimo da $(1, 1)$ a (n, m). Esso passa attraverso $n + m - 1$ caselle.
Nel caso peggiore tutte le caselle facenti parte del cammino minimo sono ostacoli.
Rimuovendo tutti gli O ostacoli da tale cammino minimo si connettono $(1, 1)$ e (n, m).

Lemma

Sia O^* l'insieme di ostacoli ottimi da rimuovere rispetto a un'istanza di taglia $n \cdot m$.
$|O^*| \leq n + m$

Proof

Si consideri un qualunque cammino minimo da $(1, 1)$ a (n, m). Esso passa attraverso $n + m - 1$ caselle.
Nel caso peggiore tutte le caselle facenti parte del cammino minimo sono ostacoli.

Rimuovendo tutti gli O ostacoli da tale cammino minimo si connettono $(1, 1)$ e (n, m).

Dal precedente argomento $|O^*| \leq O \leq n + m - 1 \leq n + m$.

Lemma

Sia O^* l'insieme di ostacoli ottimi da rimuovere rispetto a un'istanza di taglia $n \cdot m$.
$|O^*| \leq n + m$

Proof

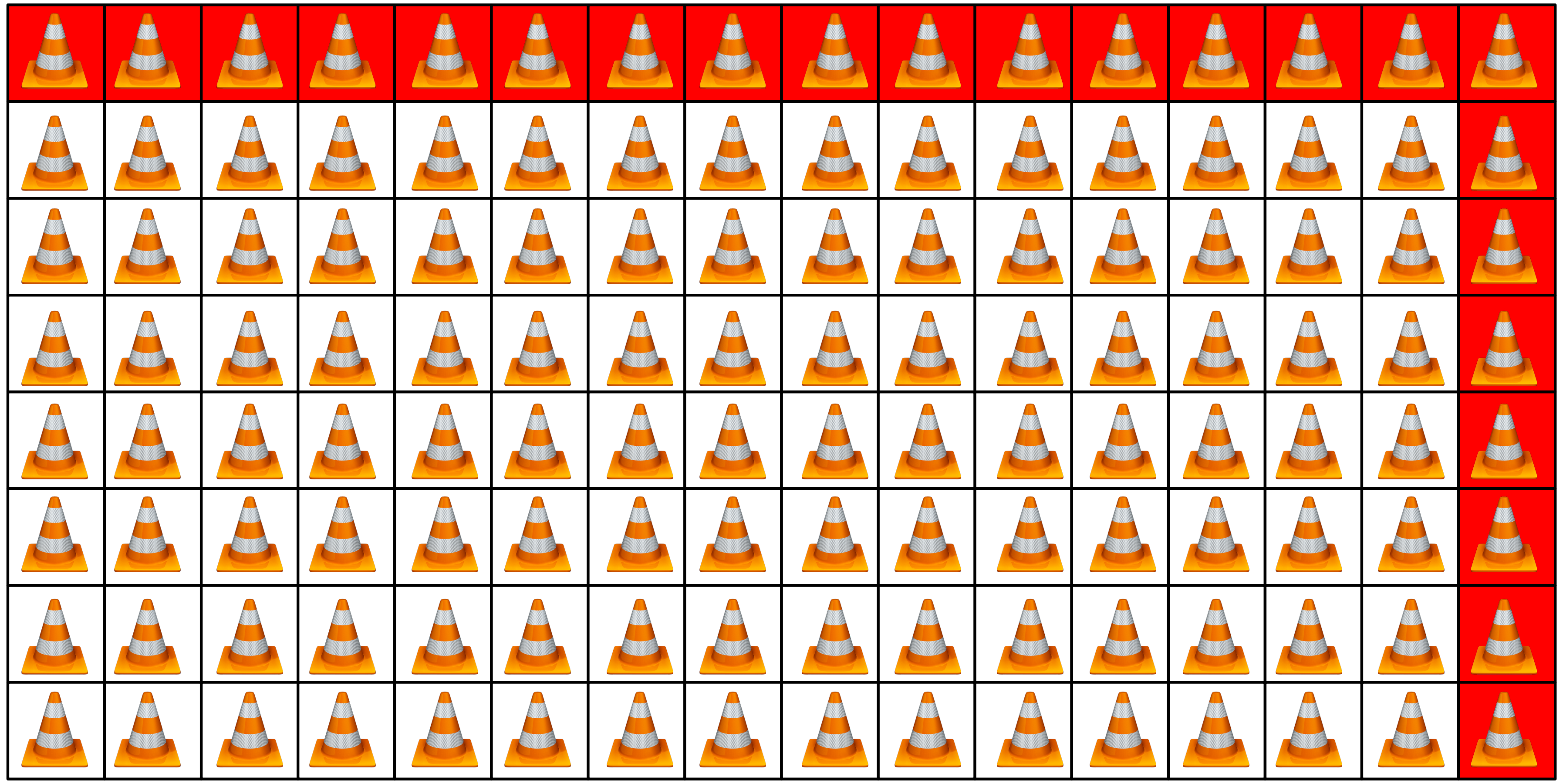
Si consideri un qualunque cammino minimo da $(1, 1)$ a (n, m). Esso passa attraverso $n + m - 1$ caselle.
Nel caso peggiore tutte le caselle facenti parte del cammino minimo sono ostacoli.

Rimuovendo tutti gli O ostacoli da tale cammino minimo si connettono $(1, 1)$ e (n, m).

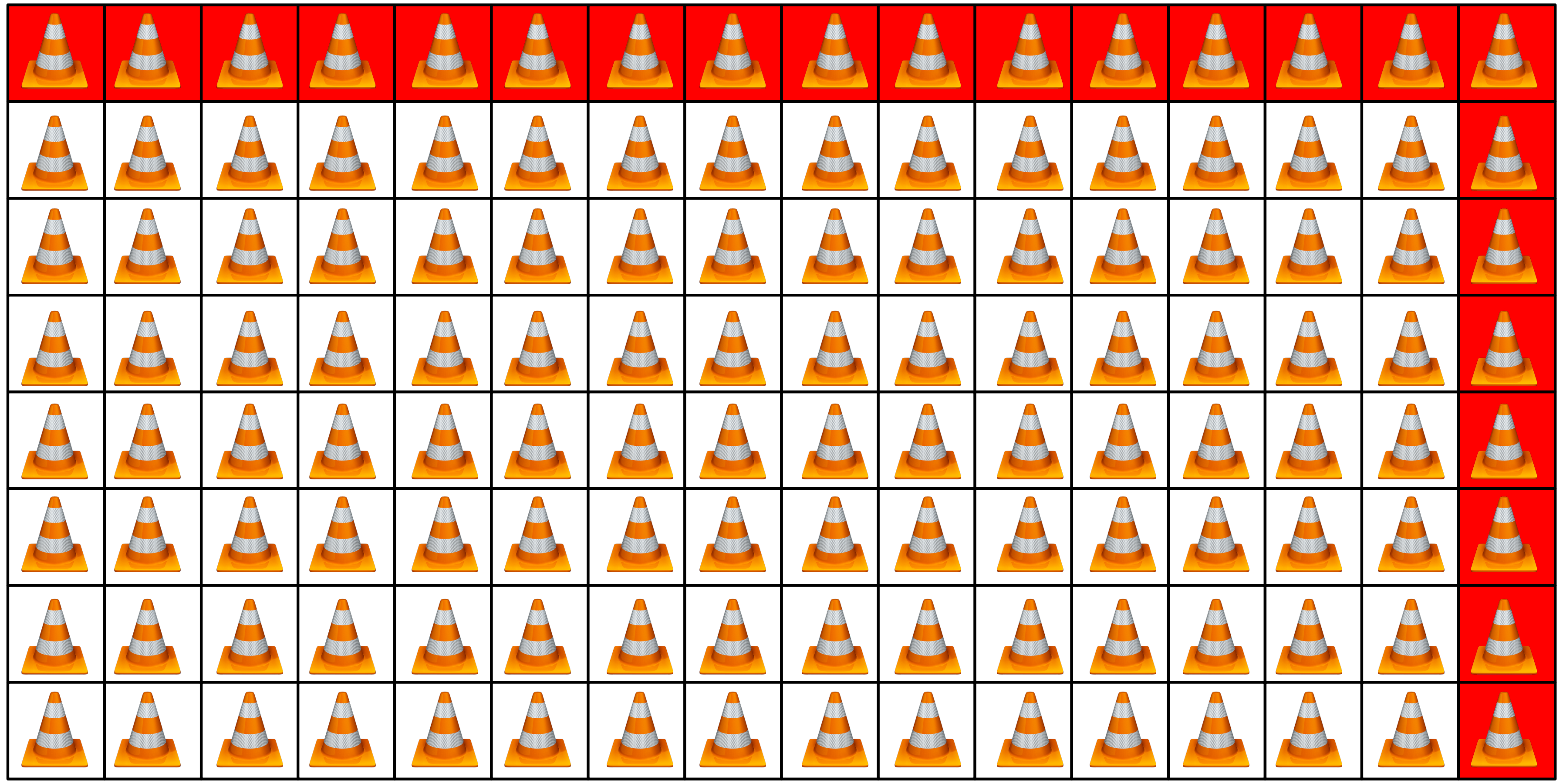
Dal precedente argomento $|O^*| \leq O \leq n + m - 1 \leq n + m$.

$\square$

Intuizione del lemma

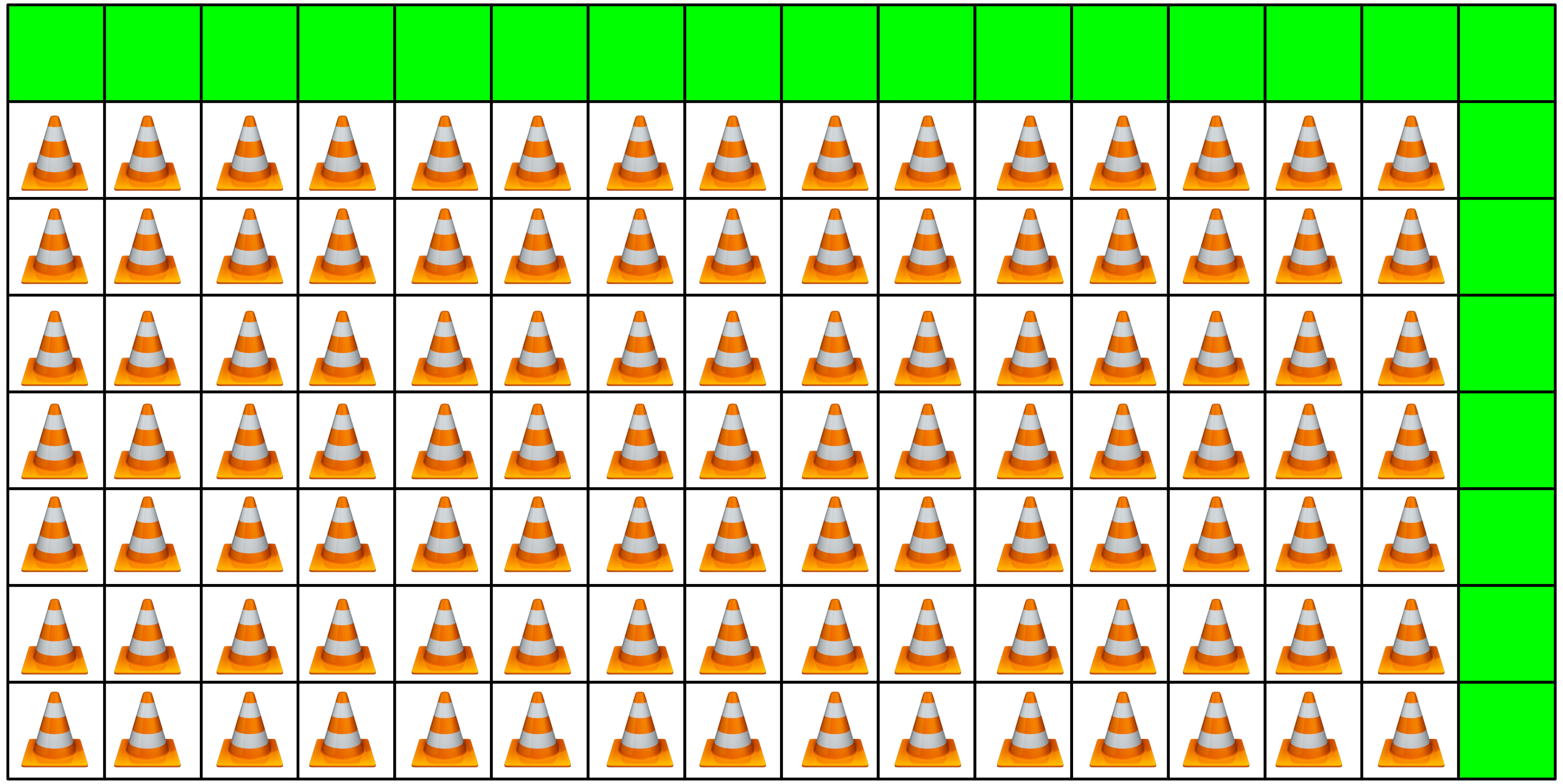


Intuizione del lemma



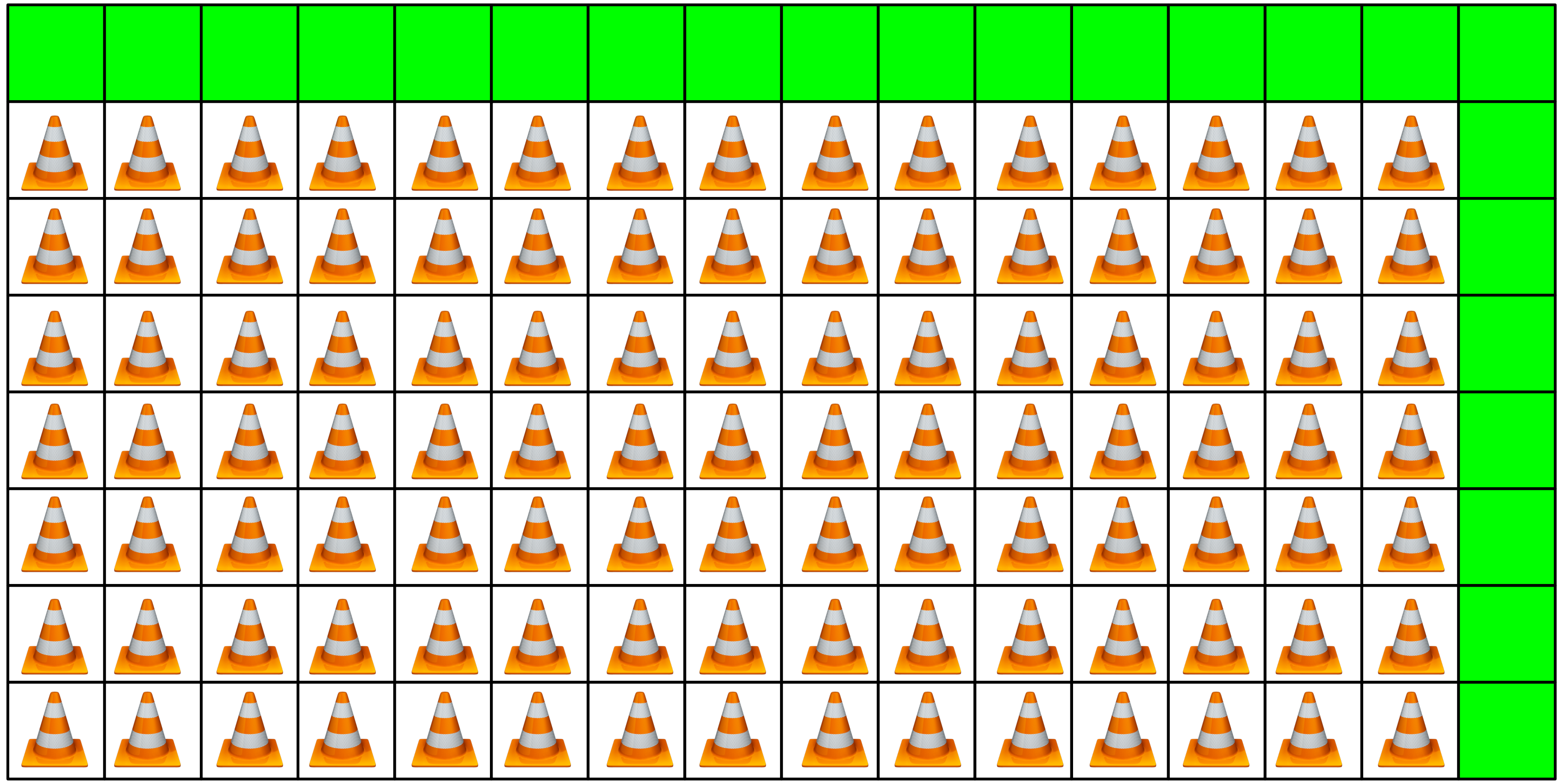
- Dopo la rimozione di tutti gli ostacoli nel path **evidenziato**, $(1,1)$ e (n,m) sono **connessi**.

Intuizione del lemma



- Dopo la rimozione di tutti gli ostacoli nel path **evidenziato**, $(1,1)$ e (n,m) sono **connessi**.

Intuizione del lemma



- Dopo la rimozione di tutti gli ostacoli nel path **evidenziato**, $(1,1)$ e (n,m) sono **connessi**.

Osservazione: È necessario esaminare solo insiemi di ostacoli da rimuovere di cardinalità $\leq n+m$.

Forza bruta Ottimizzata

- Sia $O = \{(i,j) \in (\{1,2,\dots,n\} \times \{1,2,\dots,m\}) : A_{ij} = 1\}$ l'insieme degli ostacoli.

  - For $i \leftarrow 0$ to $n+m$
    - For each set $O_i \in \binom{O}{i}$

$$A'_{ij} = \begin{cases} 0 & \text{if } (i,j) \in O_i, \\ A_{ij} & \text{otherwise} \end{cases}$$

se esiste un path **ammissibile** da $(1,1)$ a (n,m) rispetto ad A'_{ij} **return** i

Forza bruta Ottimizzata

- Sia $O = \{(i,j) \in (\{1,2,\ldots,n\} \times \{1,2,\ldots,m\}) : A_{ij} = 1\}$ l'insieme degli ostacoli.

  - For $i \leftarrow 0$ to $n+m$
    - For each set $O_i \in \binom{O}{i}$

$$A'_{ij} = \begin{cases} 0 & \text{if } (i,j) \in O_i, \\ A_{ij} & \text{otherwise} \end{cases}$$

I sottoinsiemi da analizzare sono stati ridotti in maniera considerevole!

se esiste un path **ammissibile** da $(1,1)$ a (n,m) rispetto ad A'_{ij} **return** i

Forza bruta Ottimizzata

- Sia $O = \{(i,j) \in (\{1,2,\ldots,n\} \times \{1,2,\ldots,m\}) : A_{ij} = 1\}$ l'insieme degli ostacoli.

  - For $i \leftarrow 0$ to $n+m$
    - For each set $O_i \in \binom{O}{i}$

$$A'_{ij} = \begin{cases} 0 & \text{if } (i,j) \in O_i, \\ A_{ij} & \text{otherwise} \end{cases}$$

I sottoinsiemi da analizzare sono stati ridotti in maniera considerevole!

se esiste un path **ammissibile** da $(1,1)$ a (n,m) rispetto ad A'_{ij} **return** i

Complessità temporale:

$$T(n,m) \geq \sum_{i=1}^{n+m} \binom{nm}{i} \geq \binom{nm}{n+m} \geq \frac{(nm)^{n+m}}{(n+m)^{n+m}} \geq \left(\frac{\sqrt{nm}}{2}\right)^{n+m} \geq 2^{\frac{n+m}{4} \cdot \log(nm)}$$

Forza bruta Ottimizzata

- Sia $O = \{(i,j) \in (\{1,2,\dots,n\} \times \{1,2,\dots,m\}) : A_{ij} = 1\}$ l'insieme degli ostacoli.

  - For $i \leftarrow 0$ to $n+m$
    - For each set $O_i \in \binom{O}{i}$

$$A'_{ij} = \begin{cases} 0 & \text{if } (i,j) \in O_i, \\ A_{ij} & \text{otherwise} \end{cases}$$

I sottoinsiemi da analizzare sono stati ridotti in maniera considerevole!

se esiste un path **ammissibile** da $(1,1)$ a (n,m) rispetto ad A'_{ij} **return** i

Complessità temporale:

$$T(n,m) \geq \sum_{i=1}^{n+m} \binom{nm}{i} \geq \binom{nm}{n+m} \geq \frac{(nm)^{n+m}}{(n+m)^{n+m}} \geq \left(\frac{\sqrt{nm}}{2}\right)^{n+m} \geq 2^{\frac{n+m}{4} \cdot \log(nm)}$$

L'algoritmo, se pur migliore rispetto al precedente, non è ancora polinomiale nella taglia dell'input.

Forza bruta Ottimizzata

- Sia $O = \{(i,j) \in (\{1,2,\ldots,n\} \times \{1,2,\ldots,m\}) : A_{ij} = 1\}$ l'insieme degli ostacoli.

  - For $i \leftarrow 0$ to $n+m$
    - For each set $O_i \in \binom{O}{i}$

$$A'_{ij} = \begin{cases} 0 & \text{if } (i,j) \in O_i, \\ A_{ij} & \text{otherwise} \end{cases}$$

se esiste un path **ammissibile** da $(1,1)$ a (n,m) rispetto ad A'_{ij} **return** i

I sottoinsiemi da analizzare sono stati ridotti in maniera considerevole!

Complessità temporale:

$$T(n,m) \geq \sum_{i=1}^{n+m} \binom{nm}{i} \geq \binom{nm}{n+m} \geq \frac{(nm)^{n+m}}{(n+m)^{n+m}} \geq \left(\frac{\sqrt{nm}}{2}\right)^{n+m} \geq 2^{\frac{n+m}{4} \cdot \log(nm)} \quad ☹$$

L'algoritmo, se pur migliore rispetto al precedente, non è ancora polinomiale nella taglia dell'input.

Possiamo ottenere un algoritmo con una complessità temporale polinomiale nella taglia dell'input?

Possiamo ottenere un algoritmo con una complessità temporale polinomiale nella taglia dell'input?

Provate a fornire la risposta entro il termine delle vacanze natalizie :)

Possiamo ottenere un algoritmo con una complessità temporale polinomiale nella taglia dell'input?

Provate a fornire la risposta entro il termine delle vacanze natalizie :)

$$y = \frac{\log_e\left(\frac{x}{m} - sa\right)}{r^2}$$

Possiamo ottenere un algoritmo con una complessità temporale polinomiale nella taglia dell'input?

Provate a fornire la risposta entro il termine delle vacanze natalizie :)

$$y = \frac{\log_e\left(\frac{x}{m} - sa\right)}{r^2}$$

$$yr^2 = \log_e\left(\frac{x}{m} - sa\right)$$

Possiamo ottenere un algoritmo con una complessità temporale polinomiale nella taglia dell'input?

Provate a fornire la risposta entro il termine delle vacanze natalizie :)

$$y = \frac{\log_e\left(\frac{x}{m} - sa\right)}{r^2}$$

$$yr^2 = \log_e\left(\frac{x}{m} - sa\right)$$

$$e^{yr^2} = \frac{x}{m} - sa$$

Possiamo ottenere un algoritmo con una complessità temporale polinomiale nella taglia dell'input?

Provate a fornire la risposta entro il termine delle vacanze natalizie :)

$$
\begin{gathered}
y = \frac{\log_e\left(\frac{x}{m} - sa\right)}{r^2} \\
yr^2 = \log_e\left(\frac{x}{m} - sa\right) \\
e^{yr^2} = \frac{x}{m} - sa \\
me^{yr^2} = x - msa
\end{gathered}
$$

Possiamo ottenere un algoritmo con una complessità temporale polinomiale nella taglia dell'input?

Provate a fornire la risposta entro il termine delle vacanze natalizie :)

$$y = \frac{\log_e\left(\frac{x}{m} - sa\right)}{r^2}$$

$$yr^2 = \log_e\left(\frac{x}{m} - sa\right)$$

$$e^{yr^2} = \frac{x}{m} - sa$$

$$me^{yr^2} = x - msa$$

$$me^{rry} = x - mas$$