TDDI16 – Föreläsning 6

Grafer

Filip Strömbäck

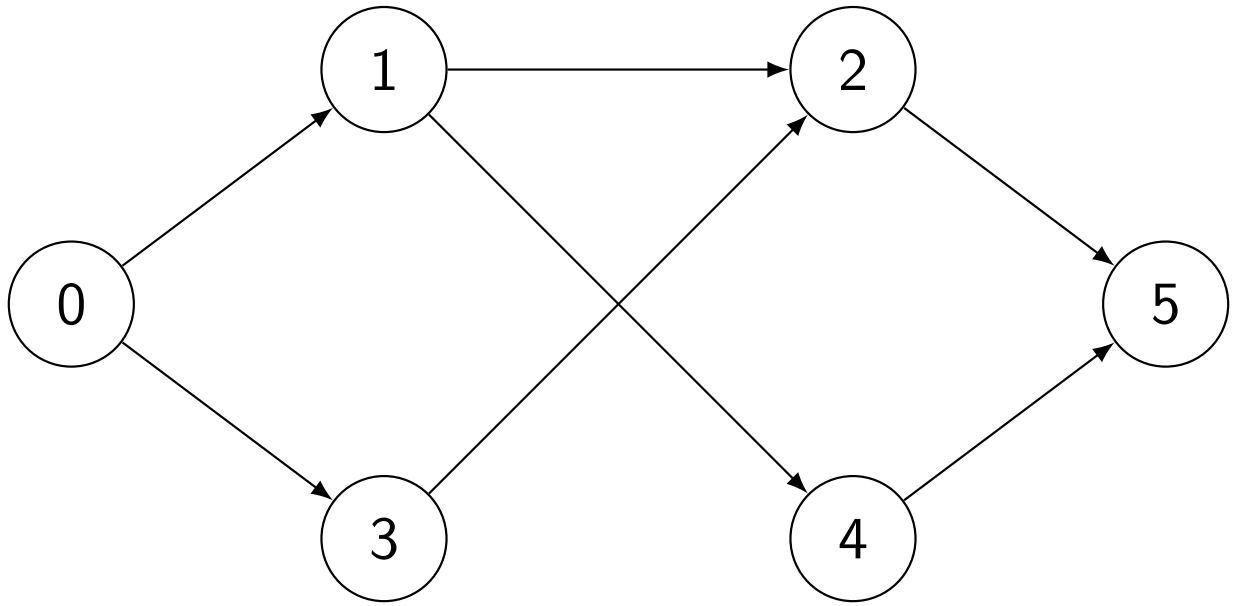
LINKÖPING UNIVERSITY

Planering

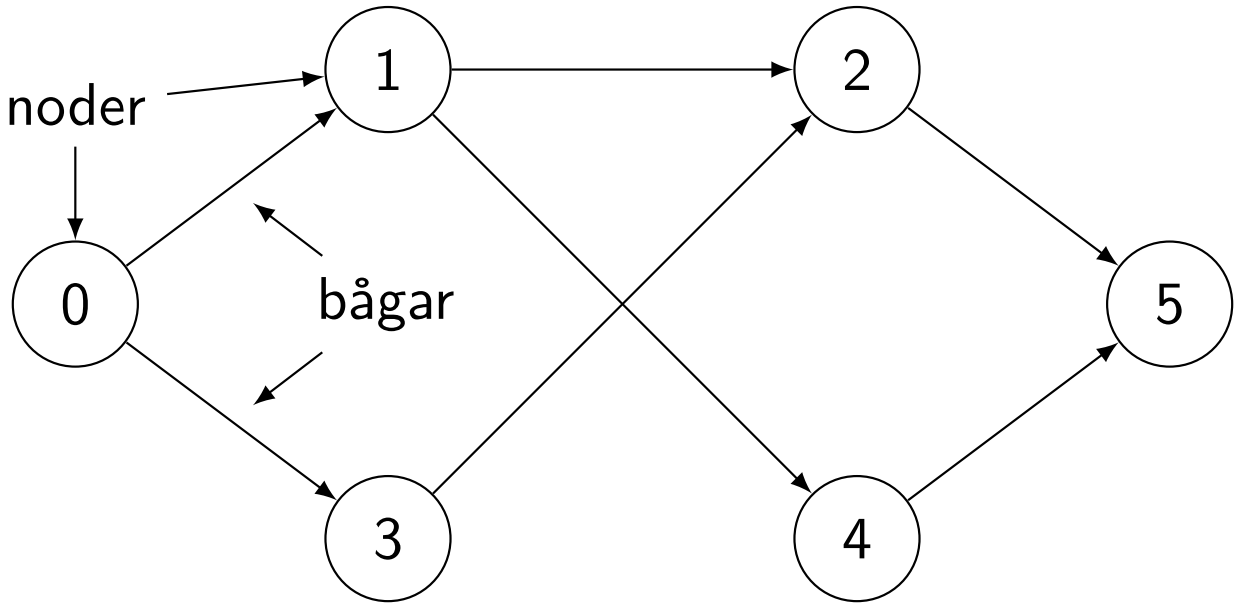
Vecka	Fö	Lab
36	Komplexitet, Linjära strukturer	----
37	Träd, AVL-träd	1---
38	Hashning, meet-in-the-middle	12--
39	Grafer, kortaste vägen	-23-
40	Grafer, andra grafalgoritmer	-23-
41	Sortering	--34
42	Repetition	---4

LINKÖPING UNIVERSITY

Vad är en graf?

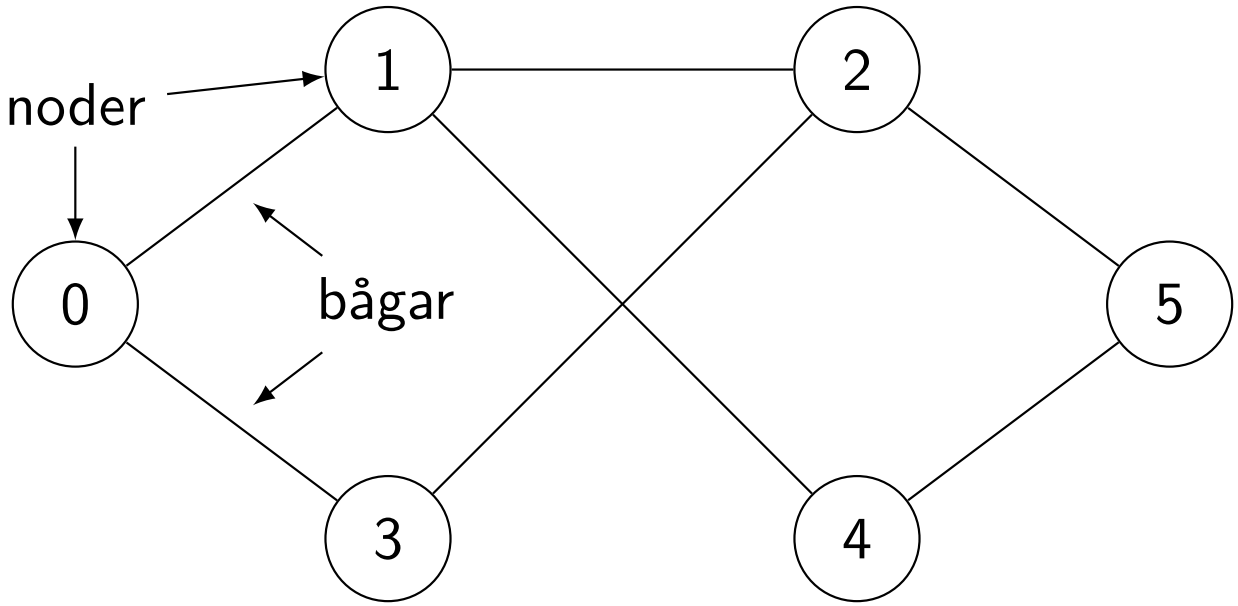


Vad är en graf?



Riktad graf

Vad är en graf?



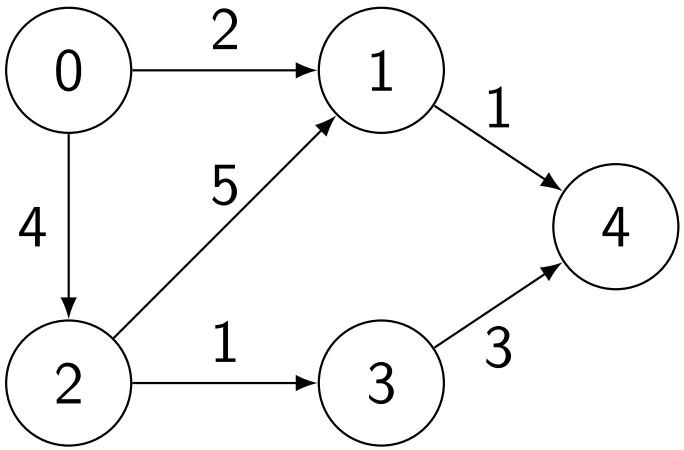
Oriktad graf

Hur representeras en graf?

- Formell definition:
  $G = (V, E)$
  V: en mängd av alla noder (*vertices*)
  E: en mängd av par motsvarande alla bågar (*edges*)
- Grannmatris
- Grannlistor
- Generera grafen ”on the fly”

Grannmatris (adjacency matrix)

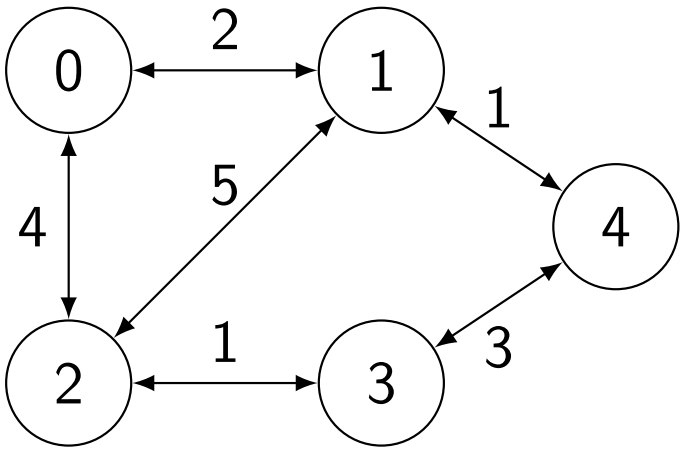
	0	1	2	3	4
0		2	4		
1					1
2		5		1	
3					3
4					



Minnesanvändning	$\mathcal{O}(\|V\|^2)$
Finns det en båge från x till y?	$\mathcal{O}(1)$
Vilka bågar finns från x?	$\mathcal{O}(\|V\|)$

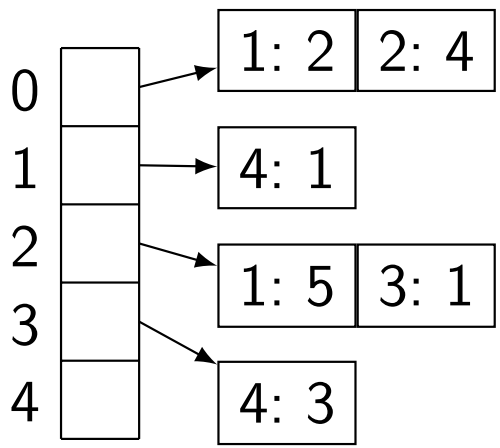
Grannmatris (adjacency matrix)

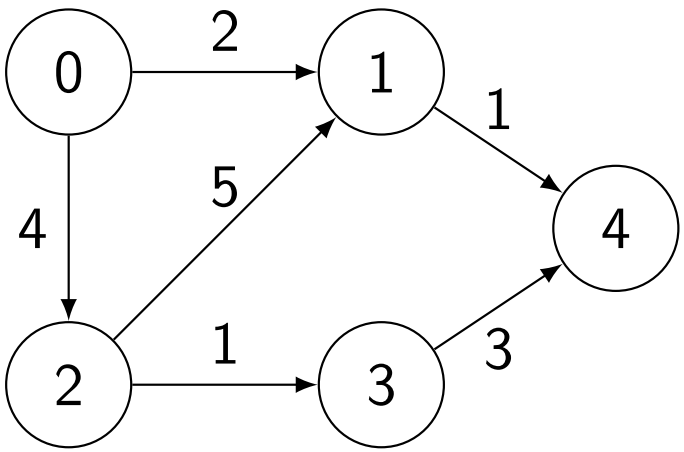
	0	1	2	3	4
0		2	4		
1	2		5		1
2	4	5		1	
3			1		3
4		1		3	



Minnesanvändning	$\mathcal{O}(\|V\|^2)$
Finns det en båge från x till y?	$\mathcal{O}(1)$
Vilka bågar finns från x?	$\mathcal{O}(\|V\|)$

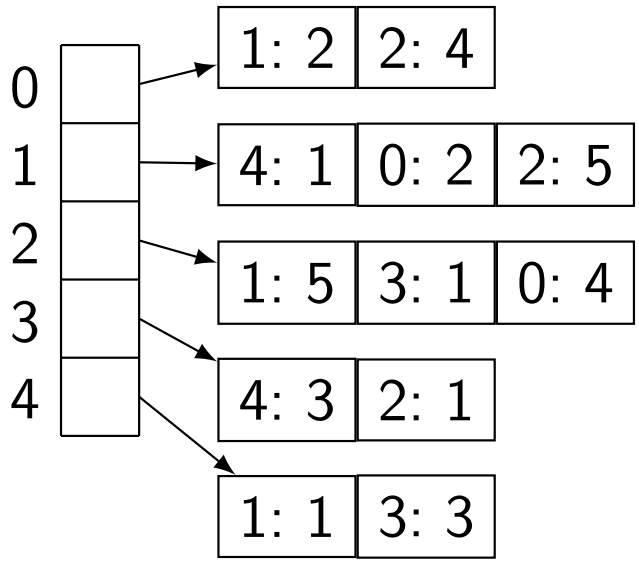
Grannlista (adjacency list)

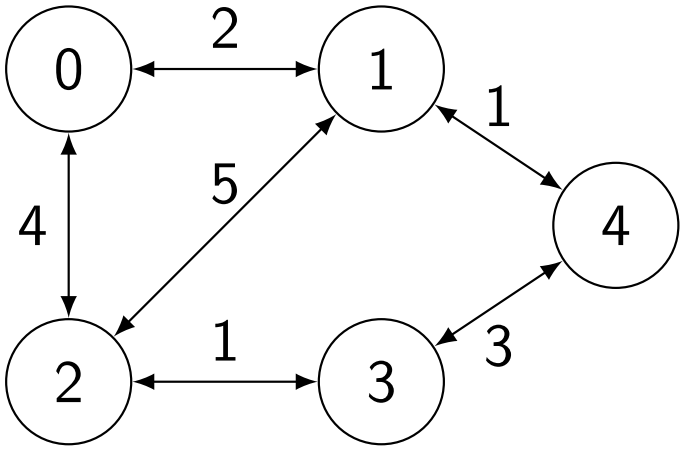




Minnesanvändning	$\mathcal{O}(\|V\| + \|E\|)$
Finns det en båge från x till y?	$\mathcal{O}(\|E\|)$
Vilka bågar finns från x?	$\mathcal{O}(\|E\|)$

Grannlista (adjacency list)



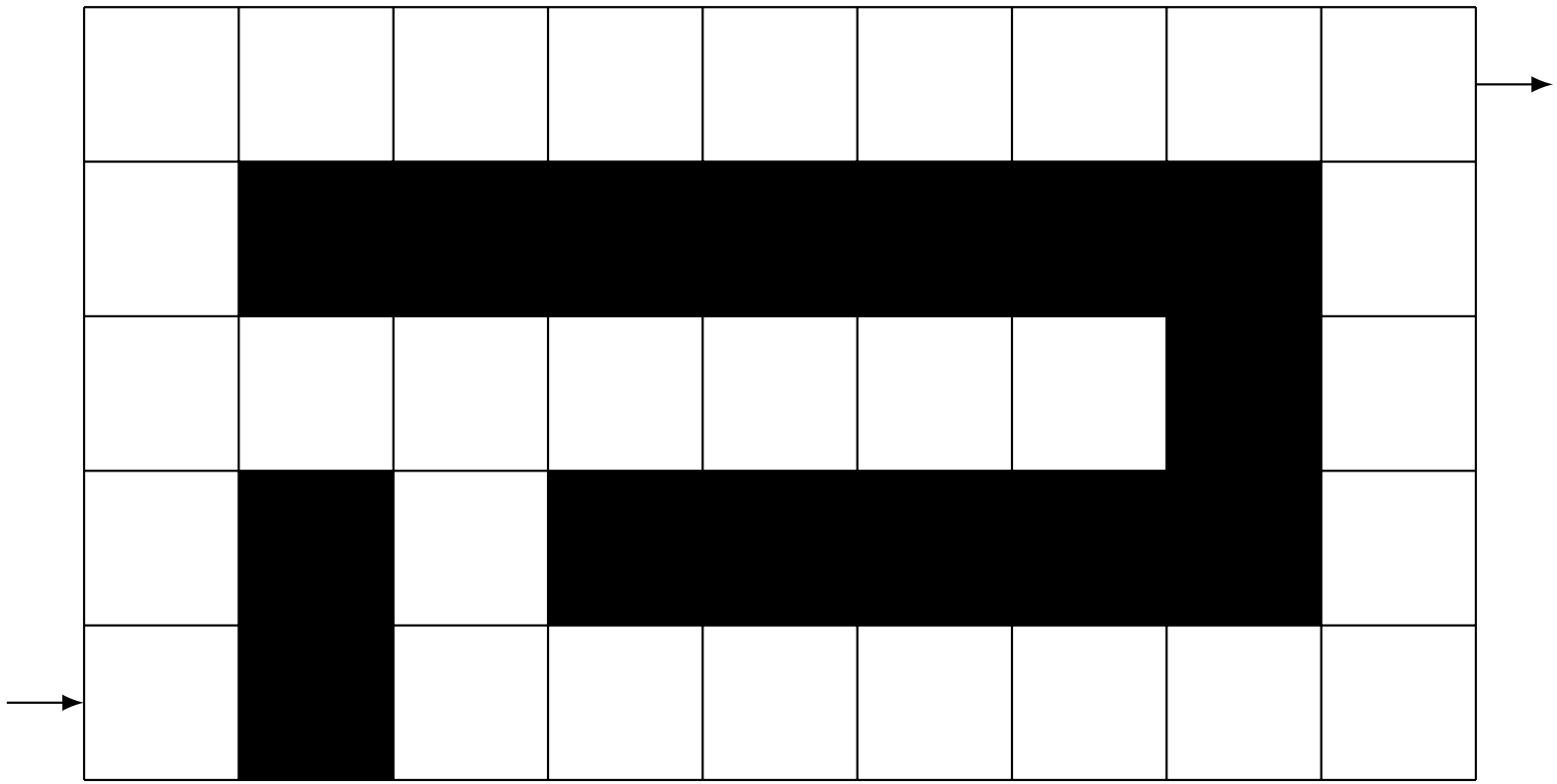


Minnesanvändning	$\mathcal{O}(\|V\| + \|E\|)$
Finns det en båge från x till y?	$\mathcal{O}(\|E\|)$
Vilka bågar finns från x?	$\mathcal{O}(\|E\|)$

LINKÖPING UNIVERSITY

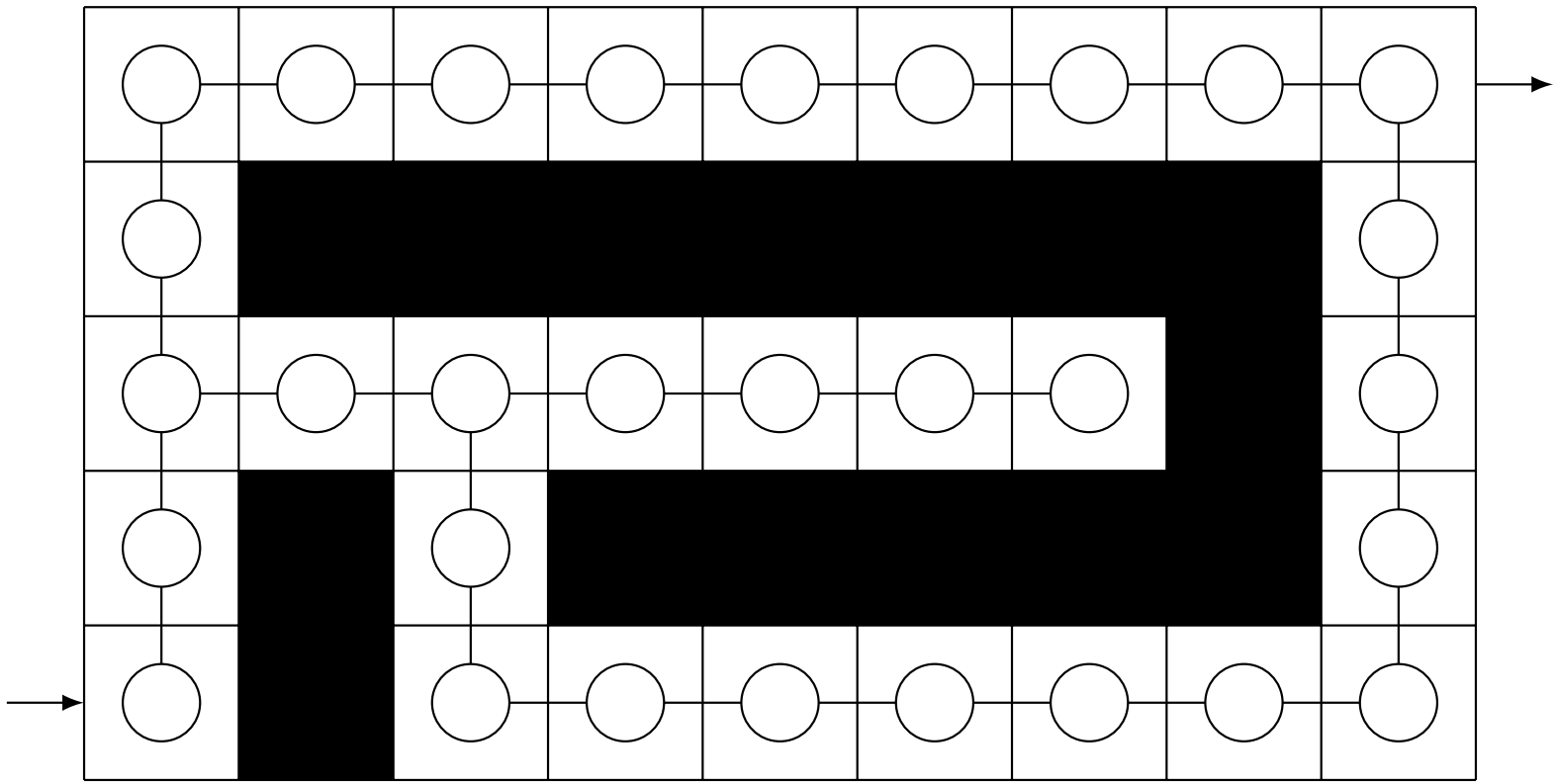
Problem

Hitta en väg genom labyrinten:



Problem

Hitta en väg genom labyrinten:



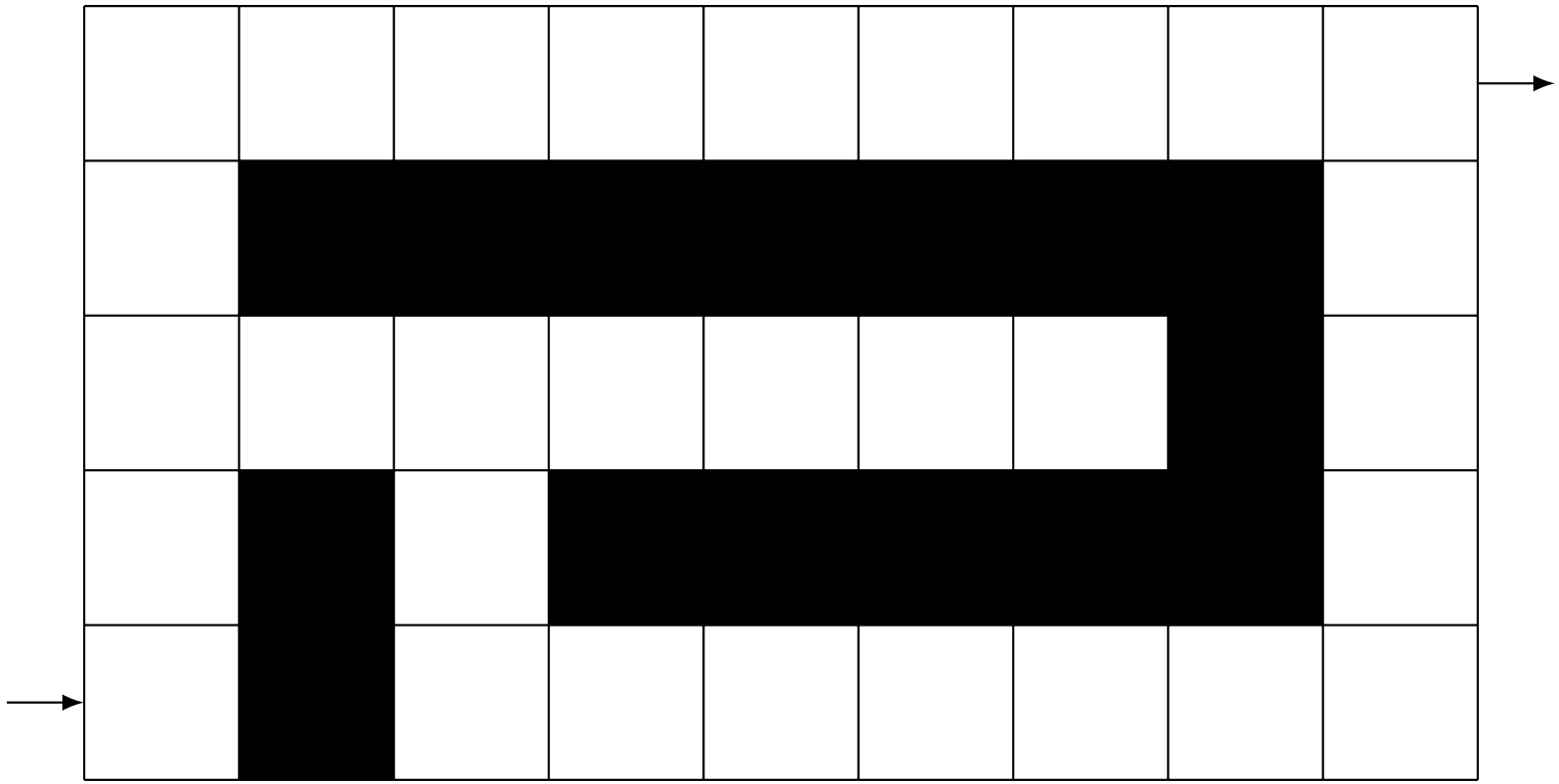
Grafdefinition

```
struct Node {
  vector<int> edges;
  bool visited = false;
  int previous;
};

vector<Node> graph;
```

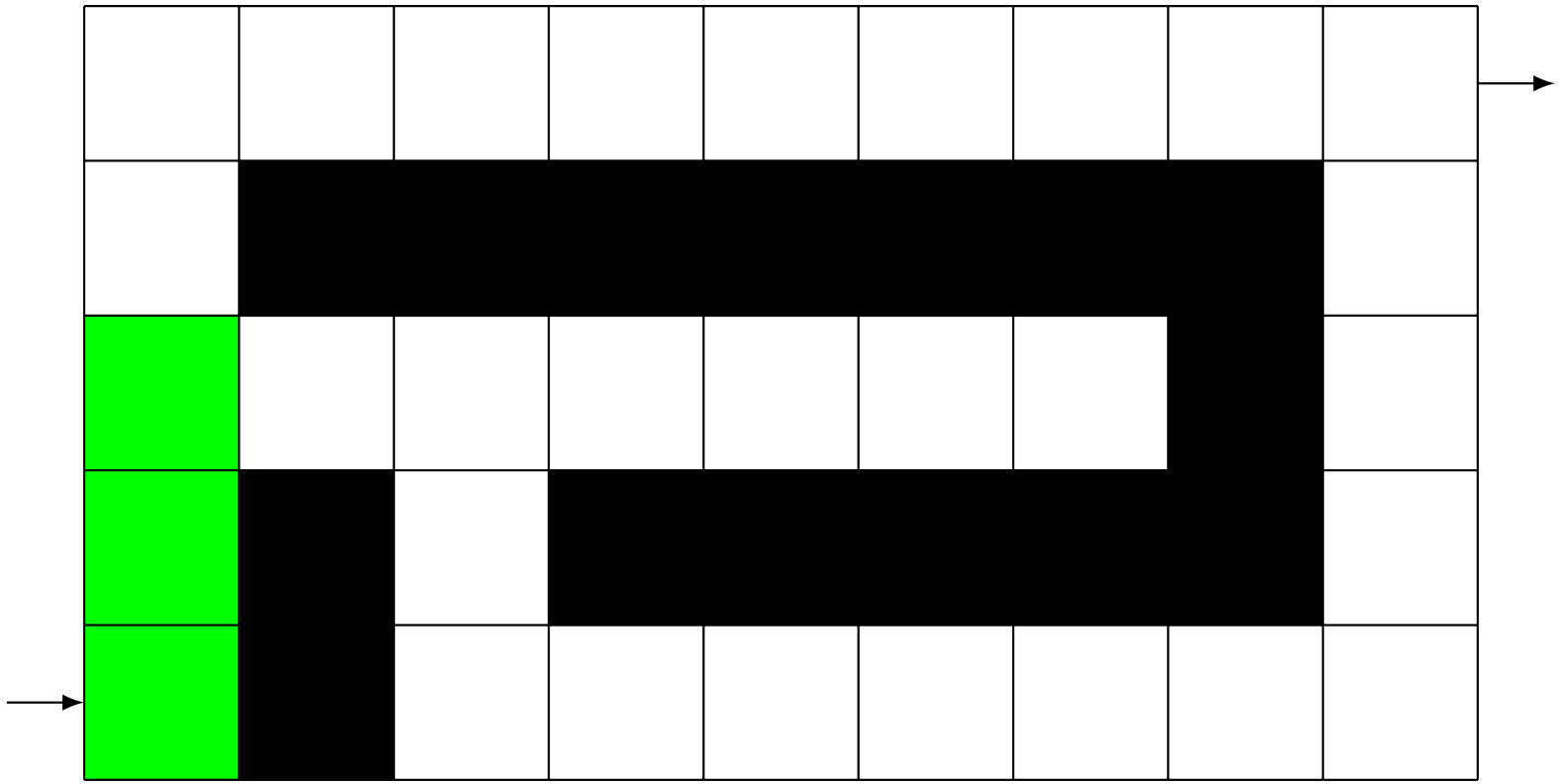
Djupet först (DFS)

```
bool dfs(int from, int to) {
  if (from == to)
    return true;

  for (int x : graph[from].edges) {
    if (!graph[x].visited) {
      graph[x].visited = true;
      if (dfs(x, to))
        return true;
    }
  }

  return false;
}
```

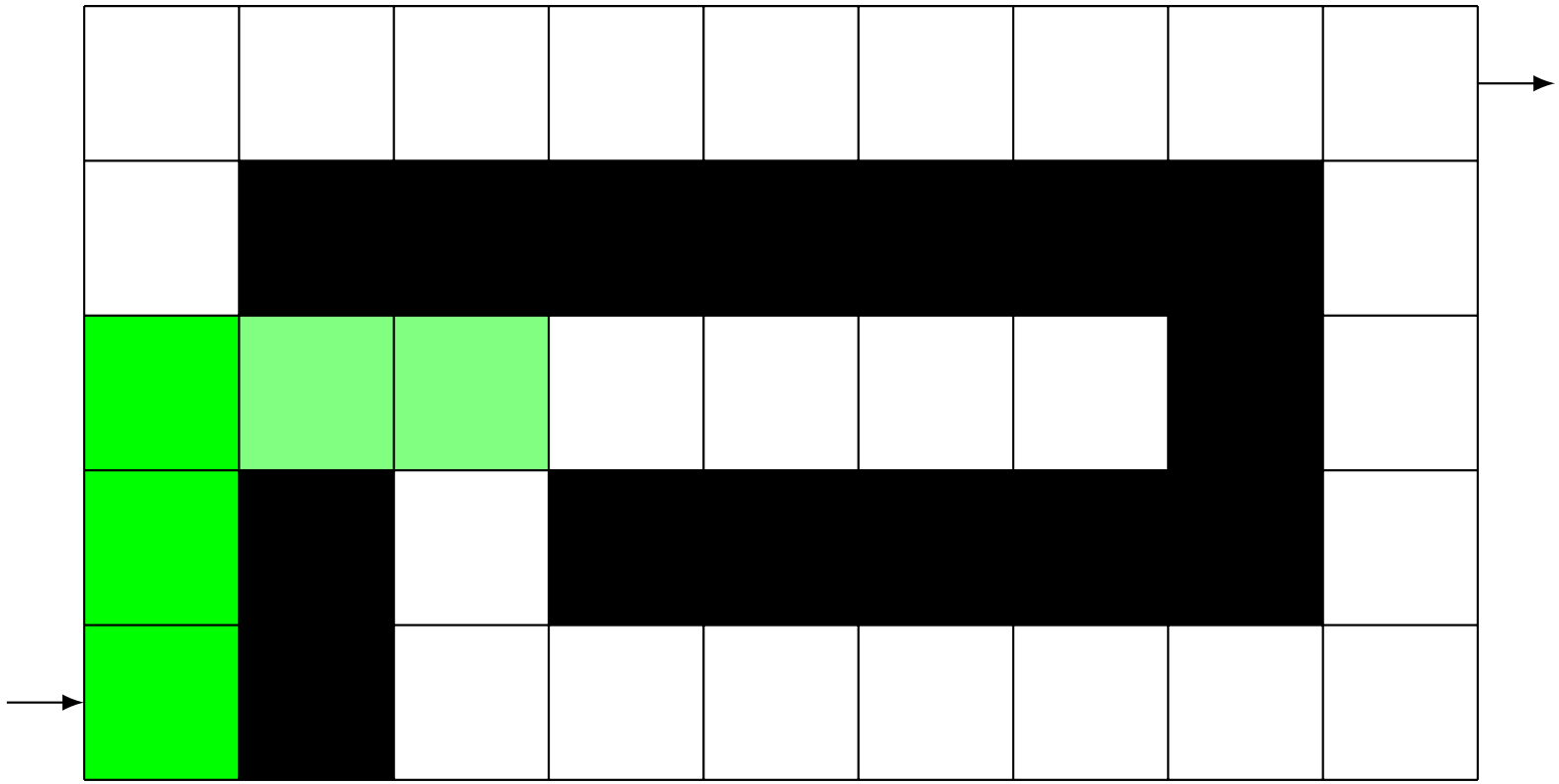
Djupet först (DFS)



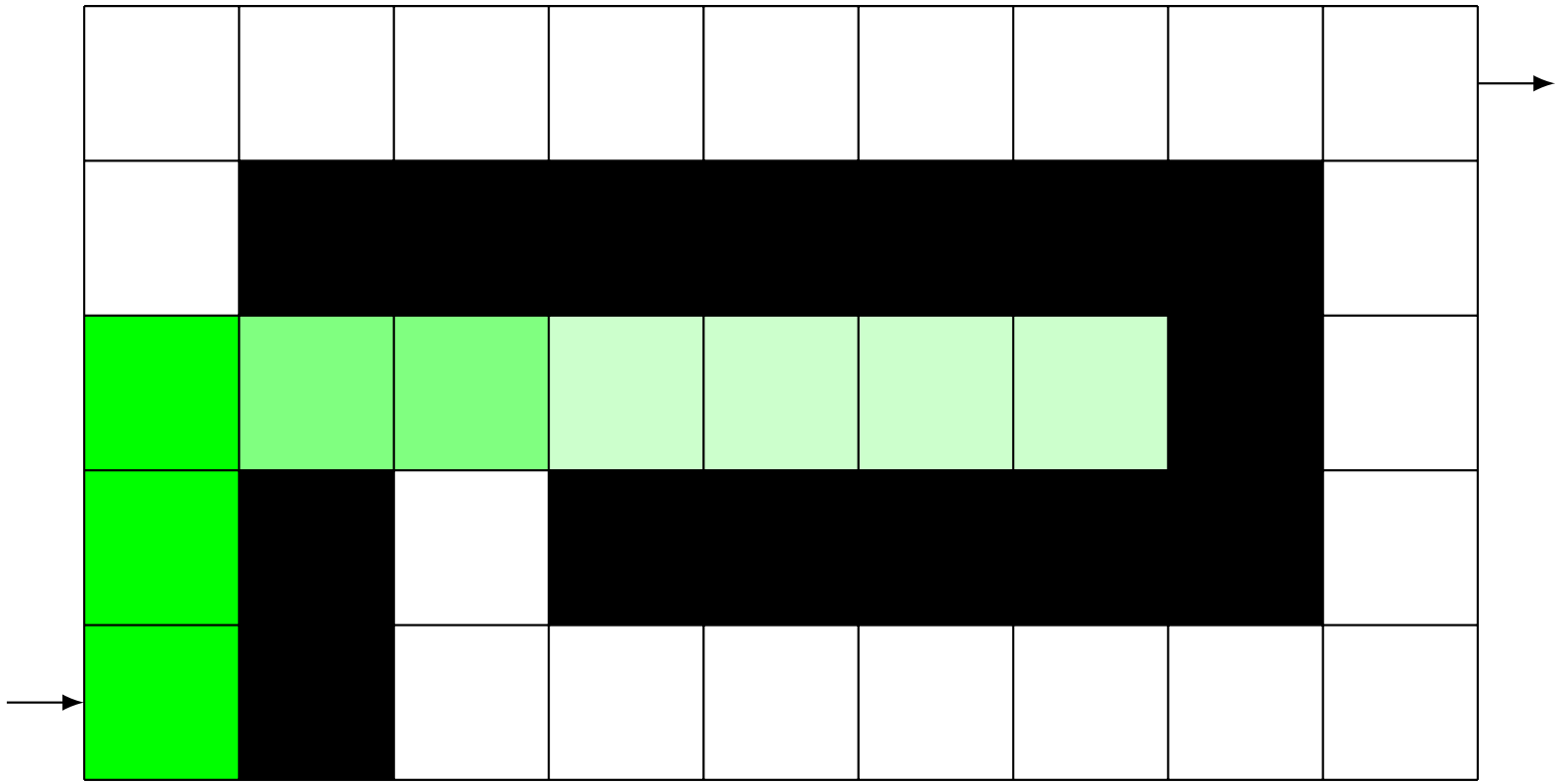
Djupet först (DFS)



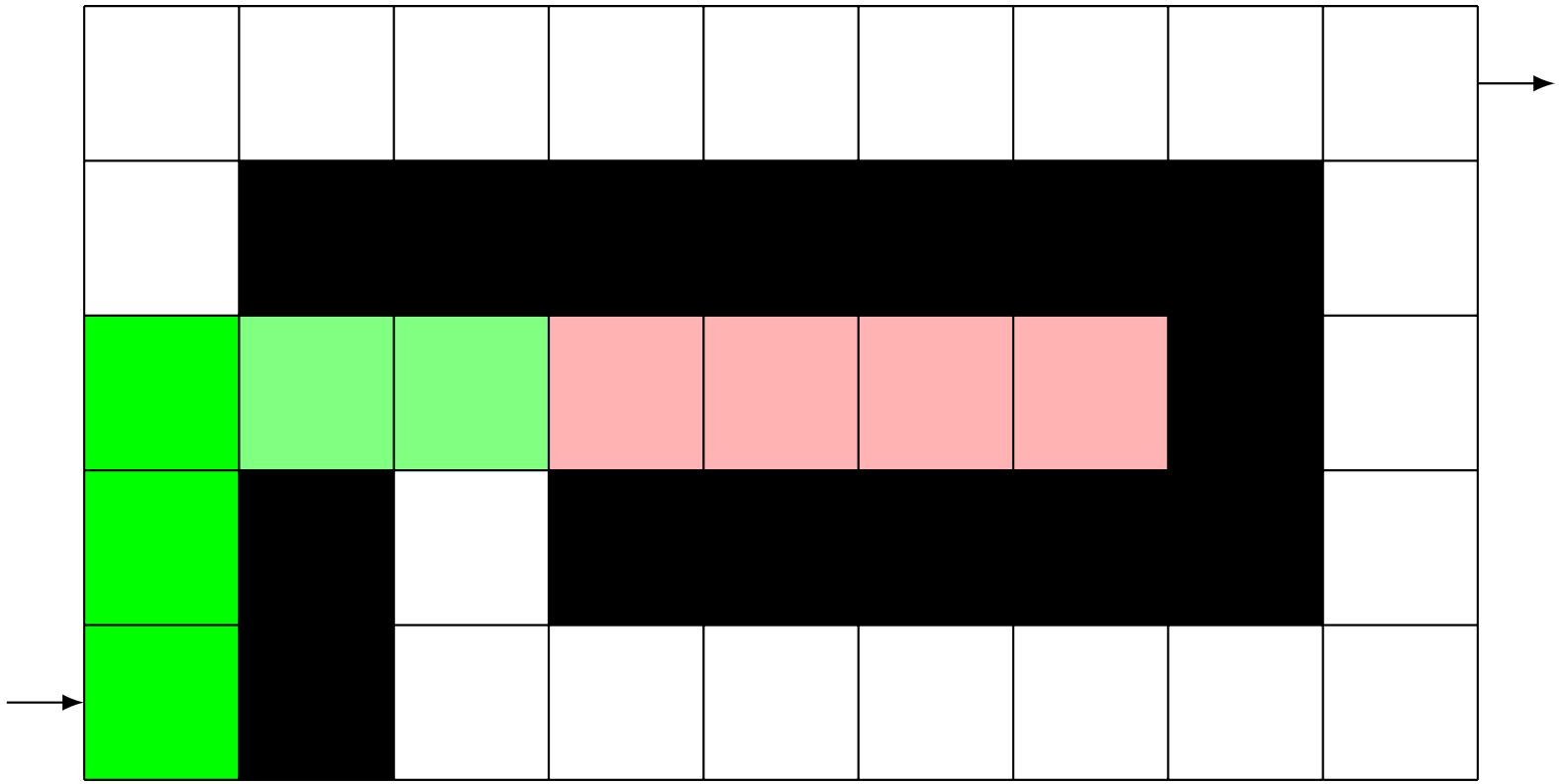
Djupet först (DFS)



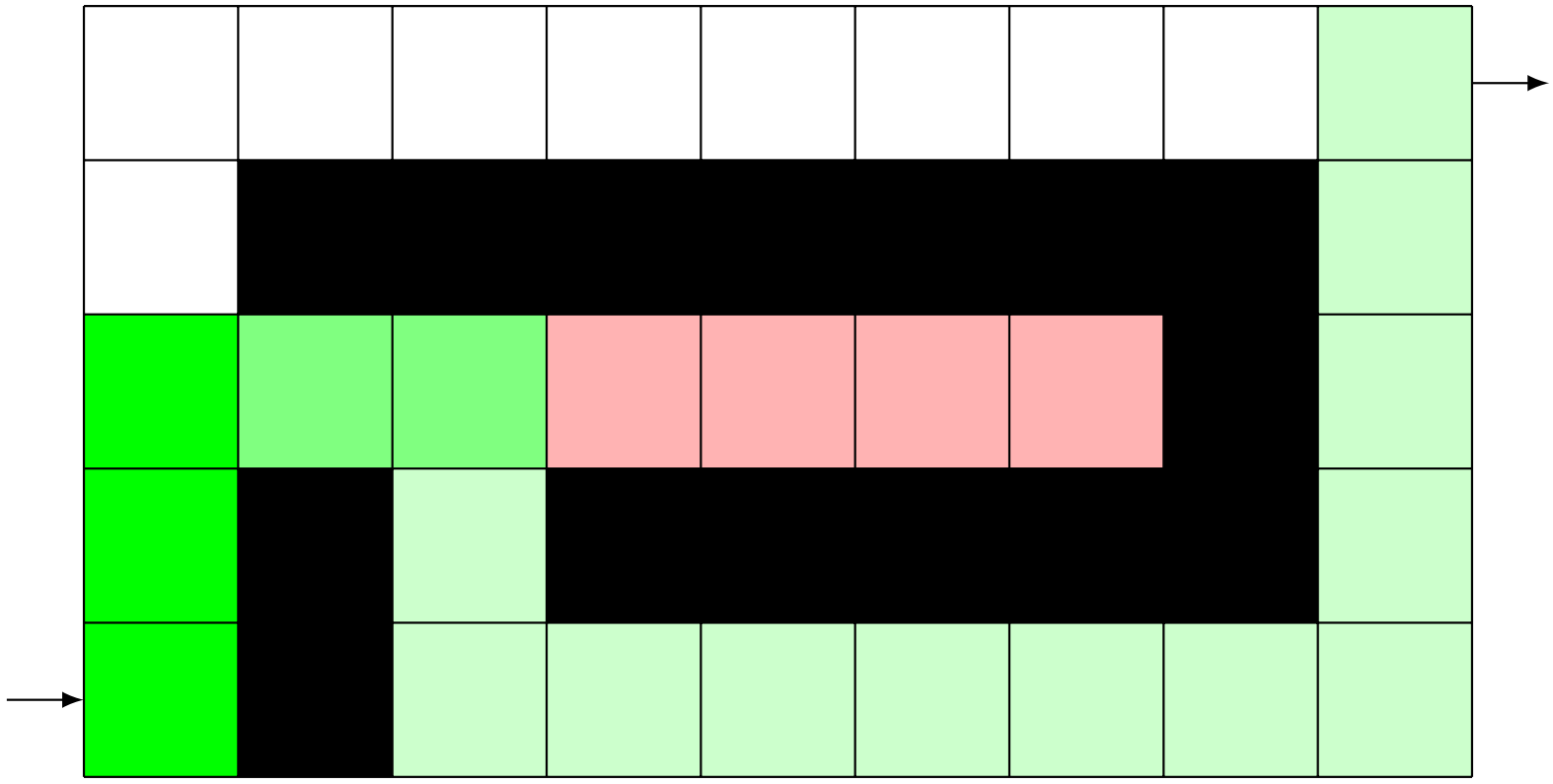
Djupet först (DFS)



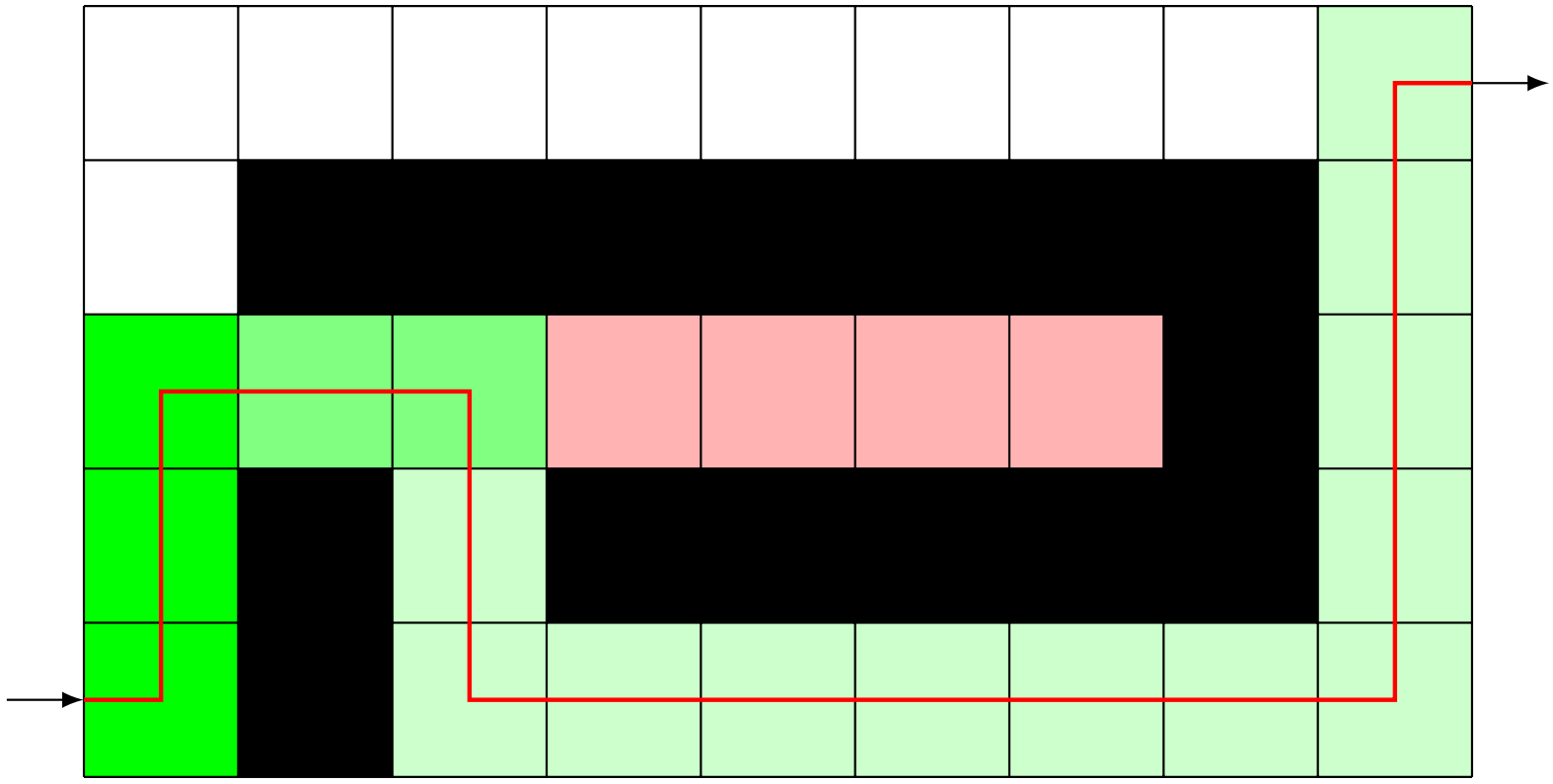
Djupet först (DFS)



Djupet först (DFS)



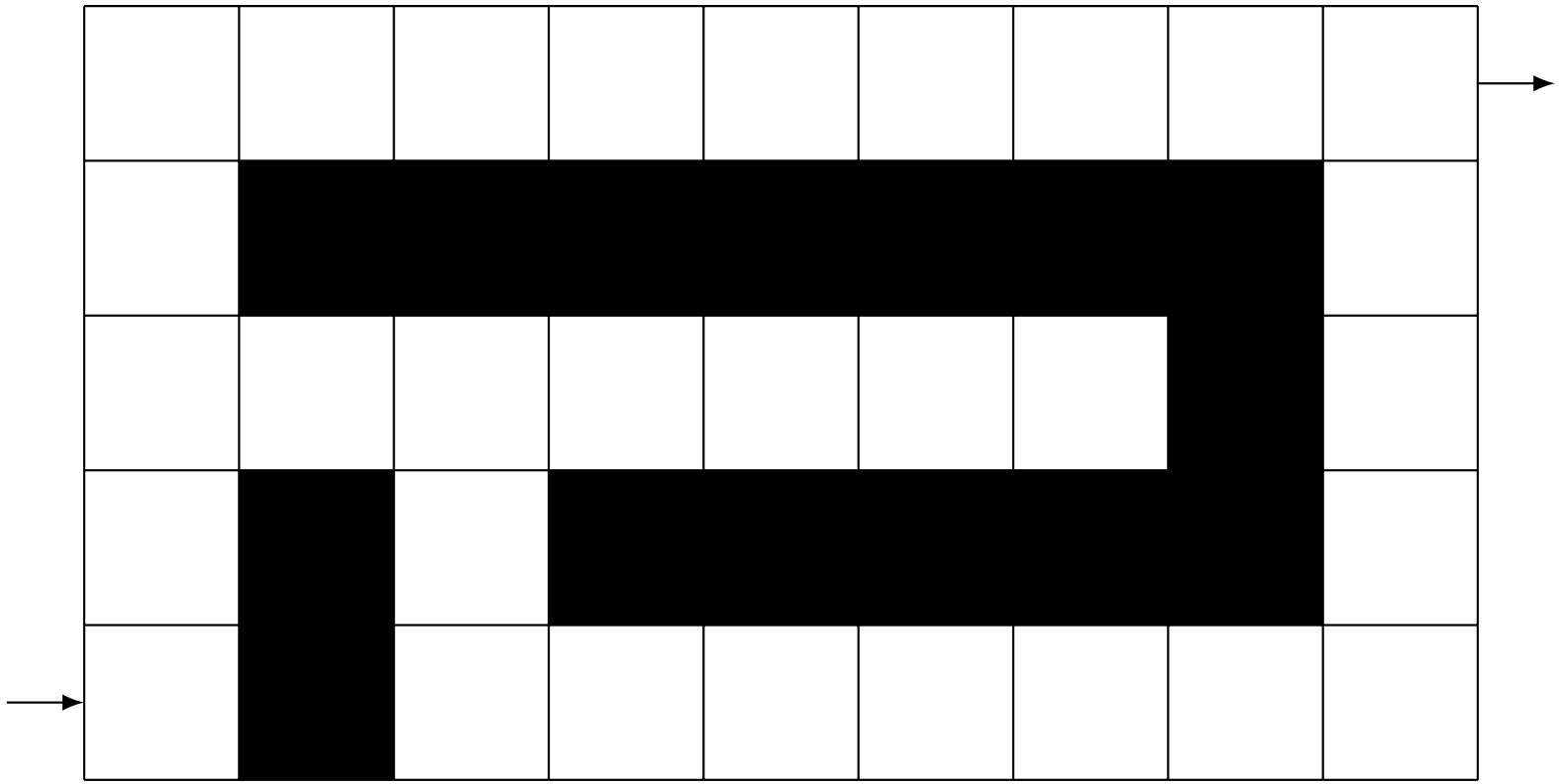
Djupet först (DFS)



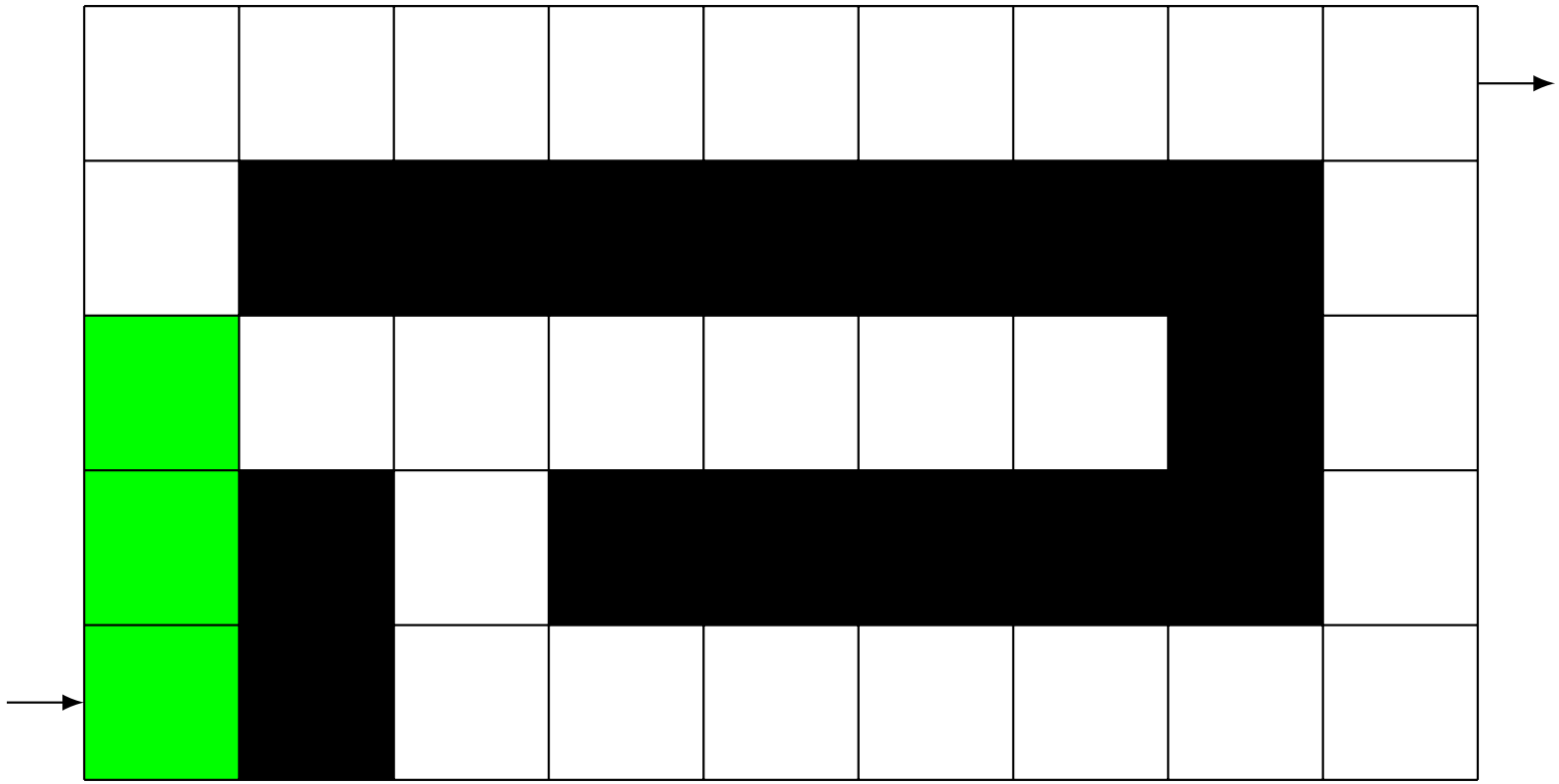
Bredden först (BFS)

```
bool bfs(int from, int to) {
  queue<int> q; q.push(from);
  graph[from].visited = true;

  while (!q.empty()) {
    int current = q.top(); q.pop();
    for (int x : graph[current].edges) {
      // om x är vårt mål är vi klara

      // om x inte redan är besökt, markera
      // den som besökt och lägg på kö
    }
  }
}
```

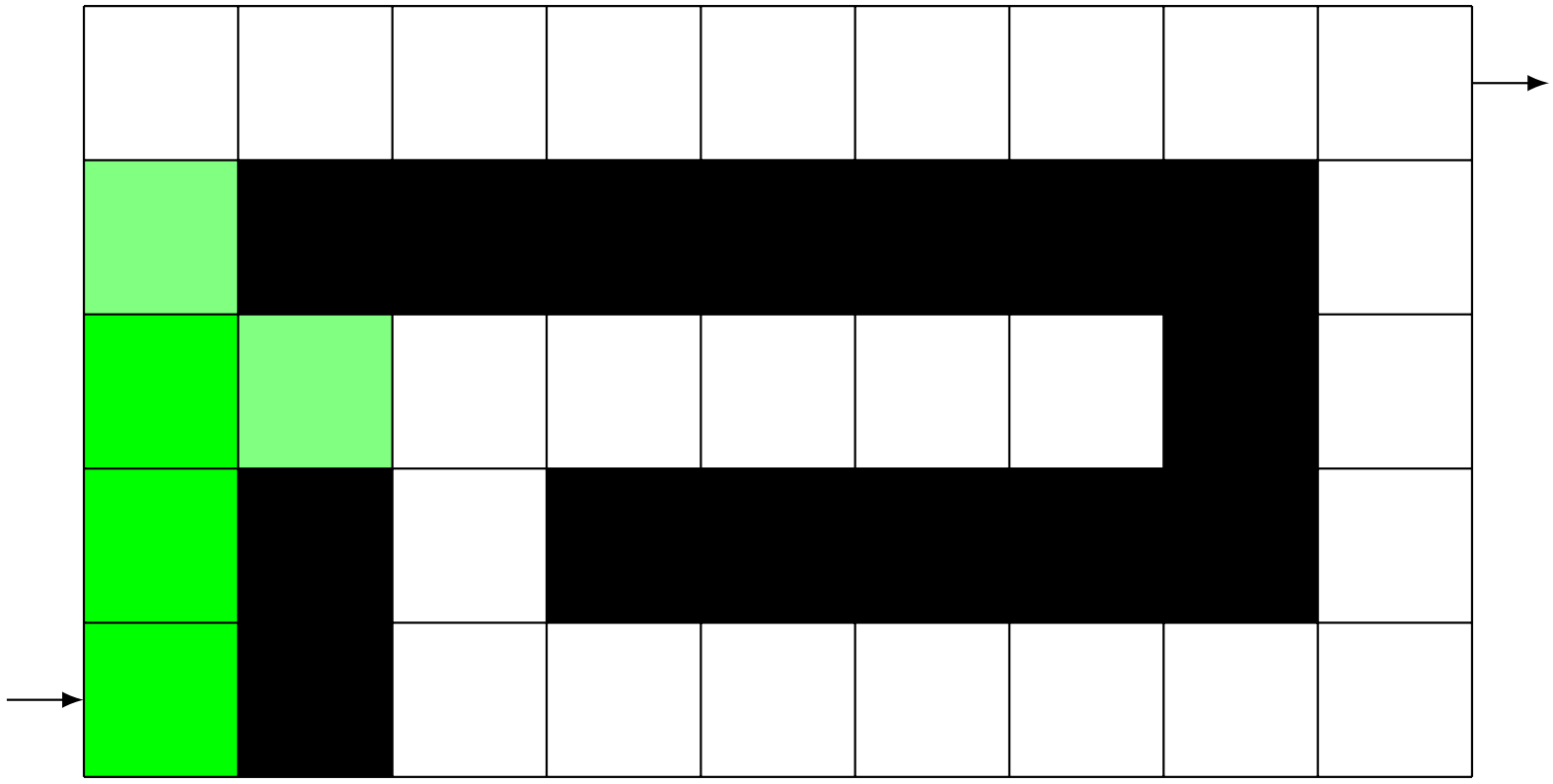
Bredden först (BFS)



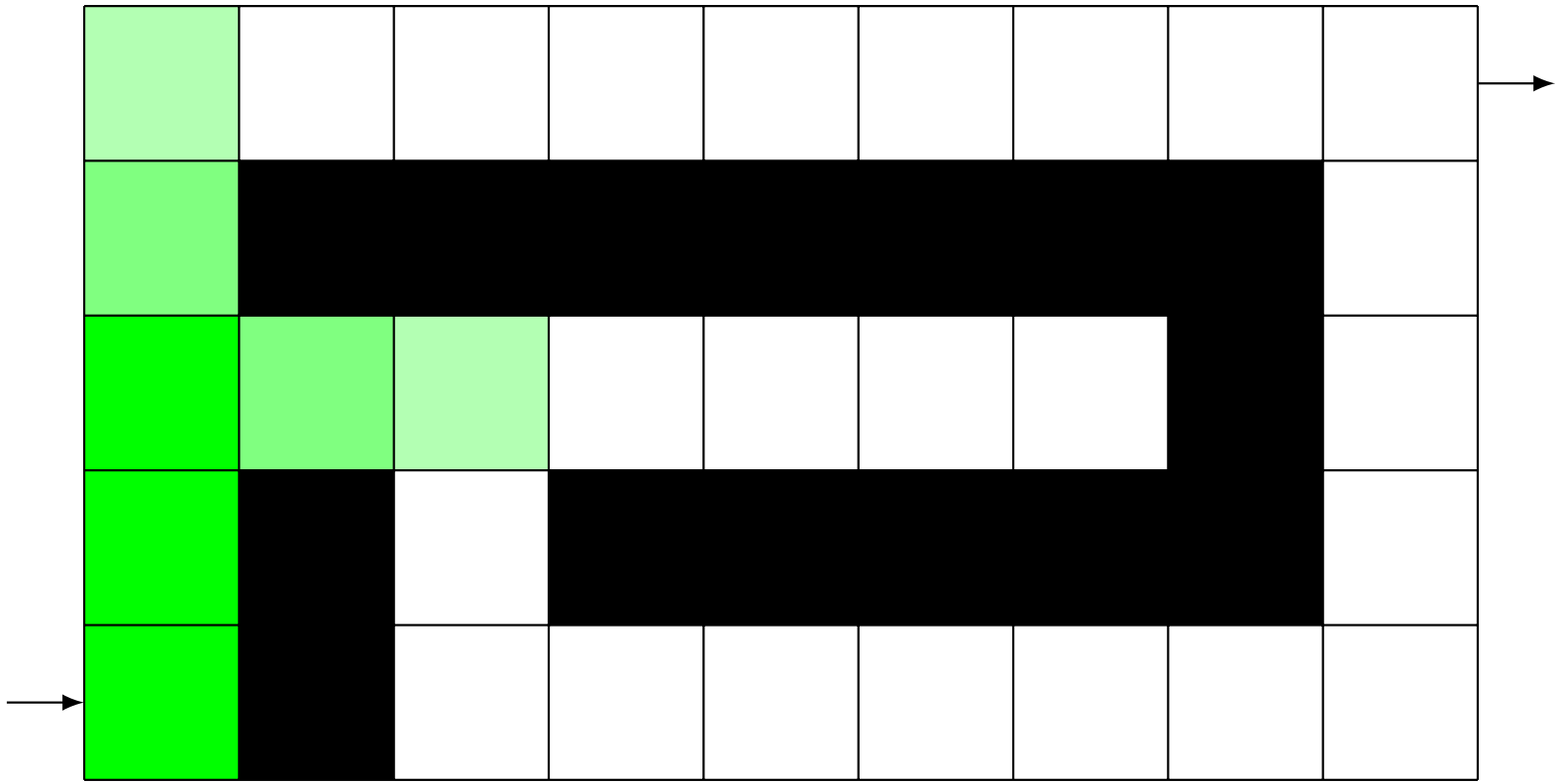
Bredden först (BFS)



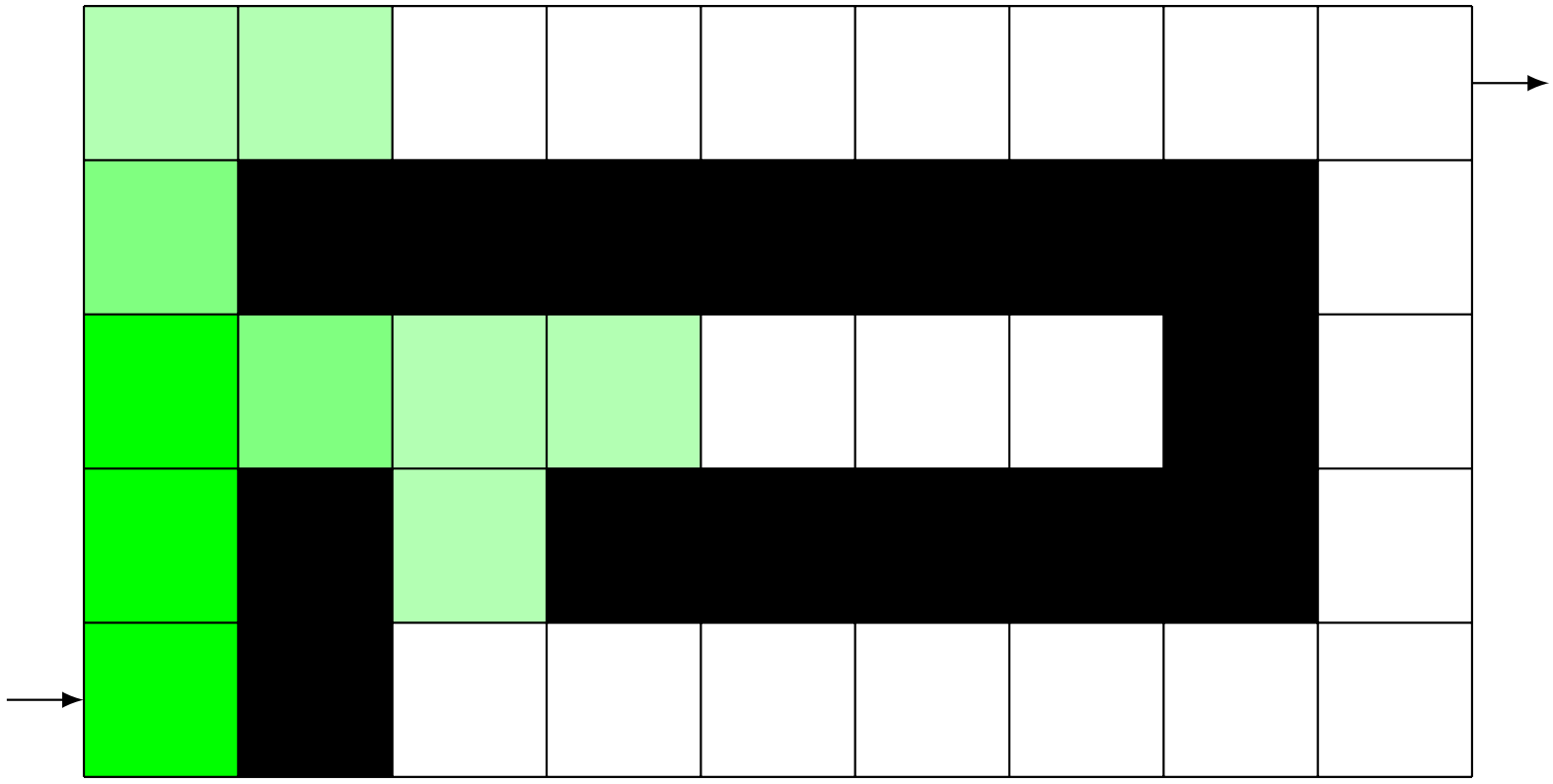
Bredden först (BFS)



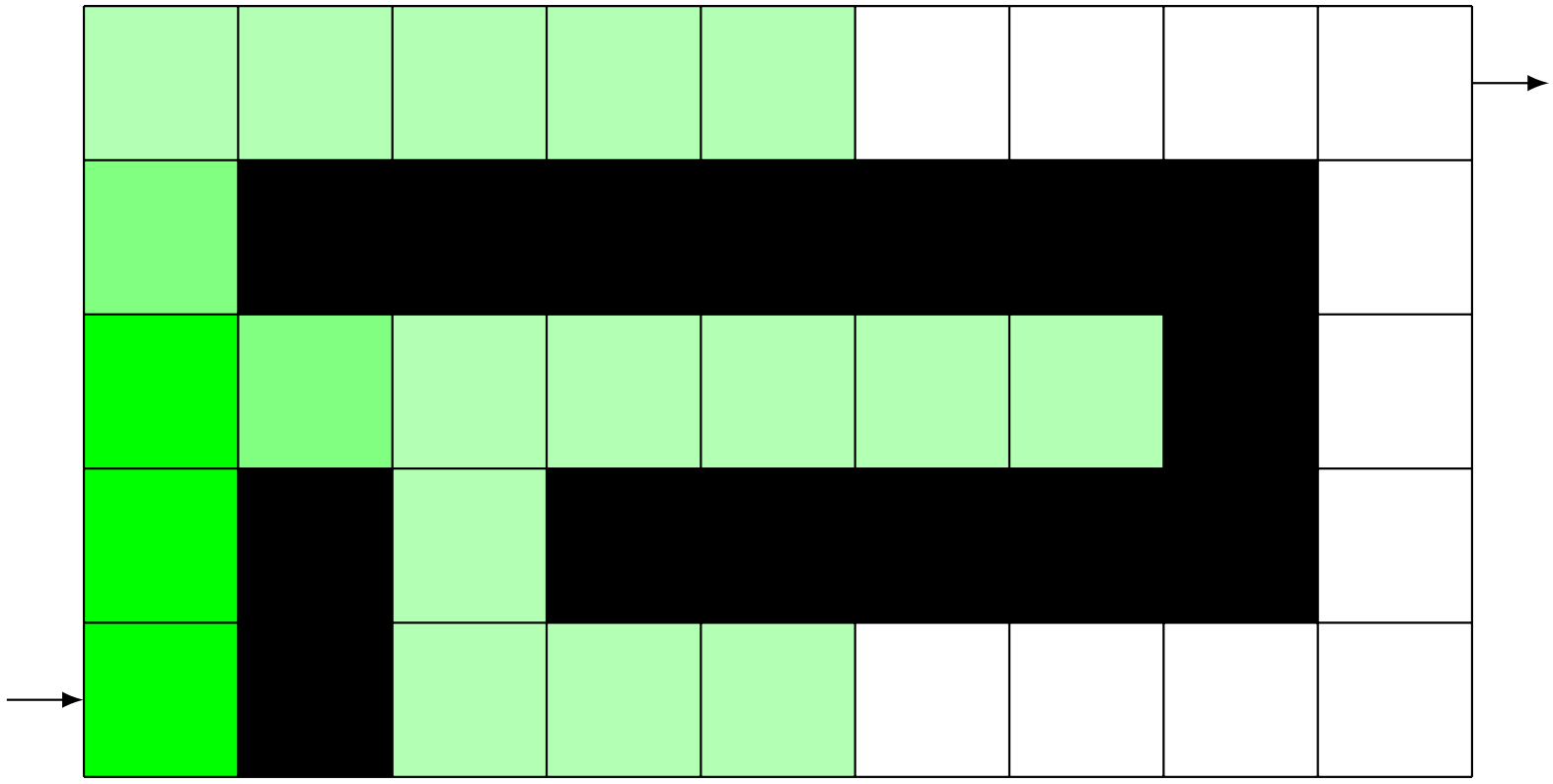
Bredden först (BFS)



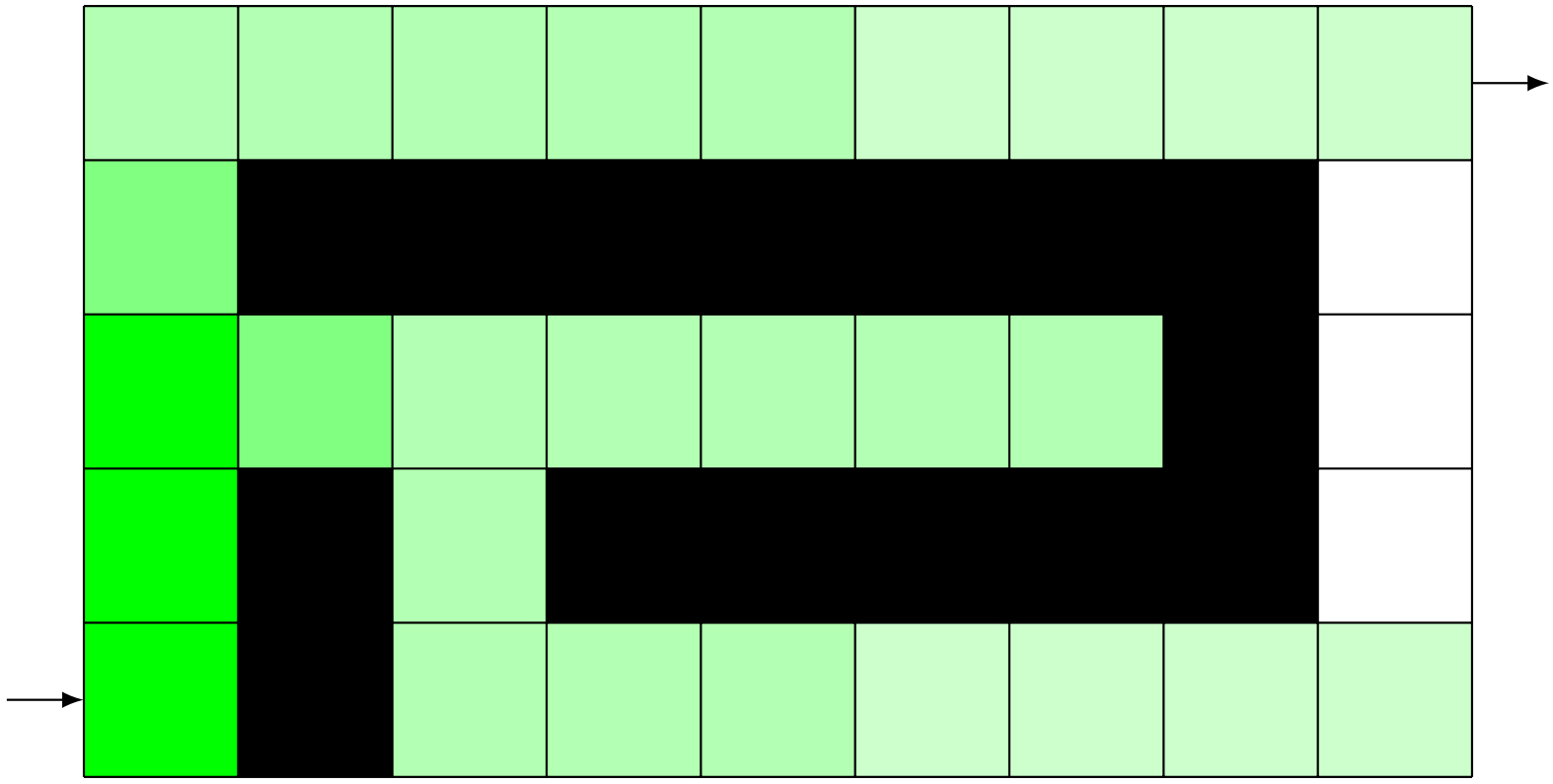
Bredden först (BFS)



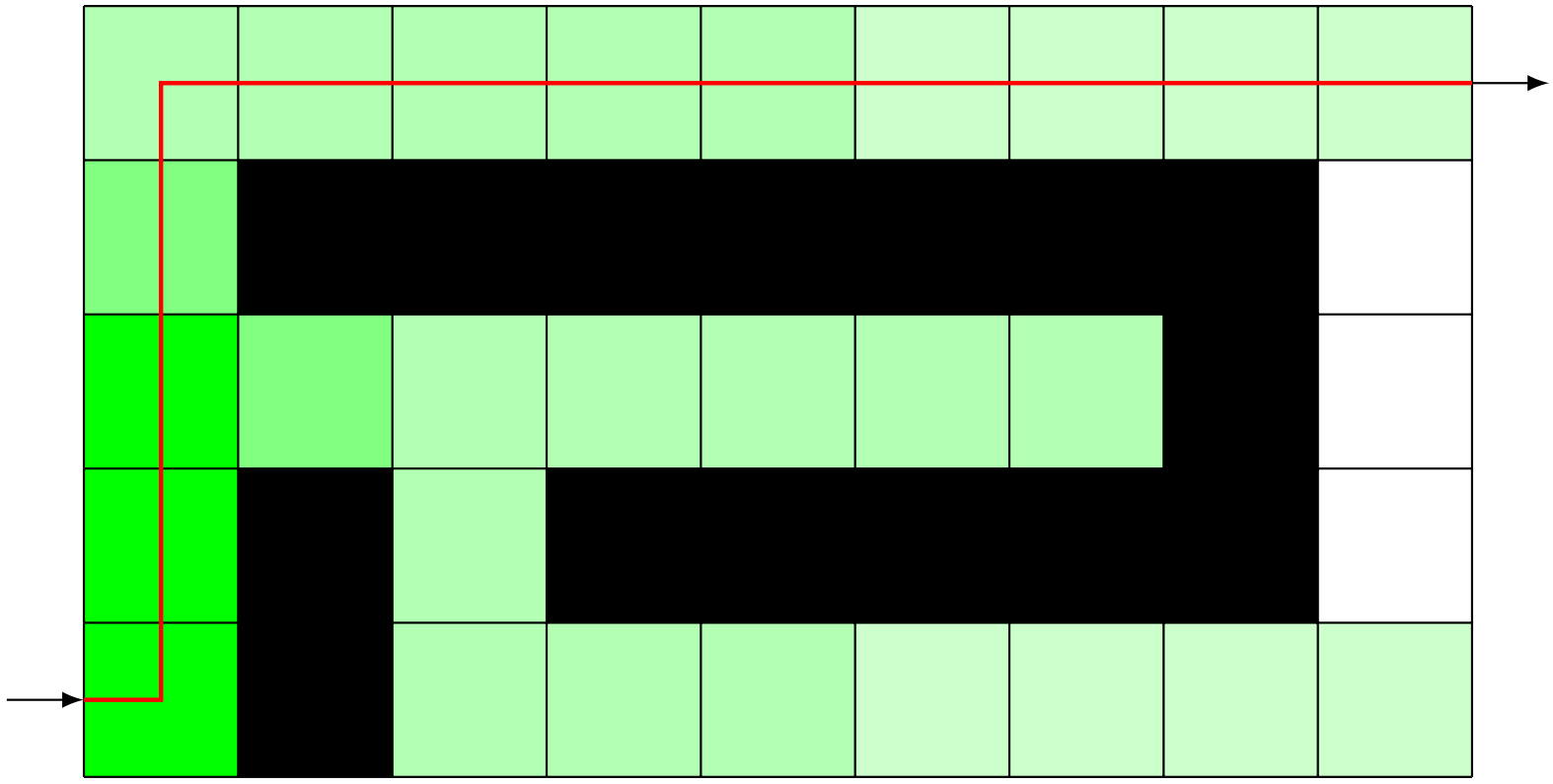
Bredden först (BFS)



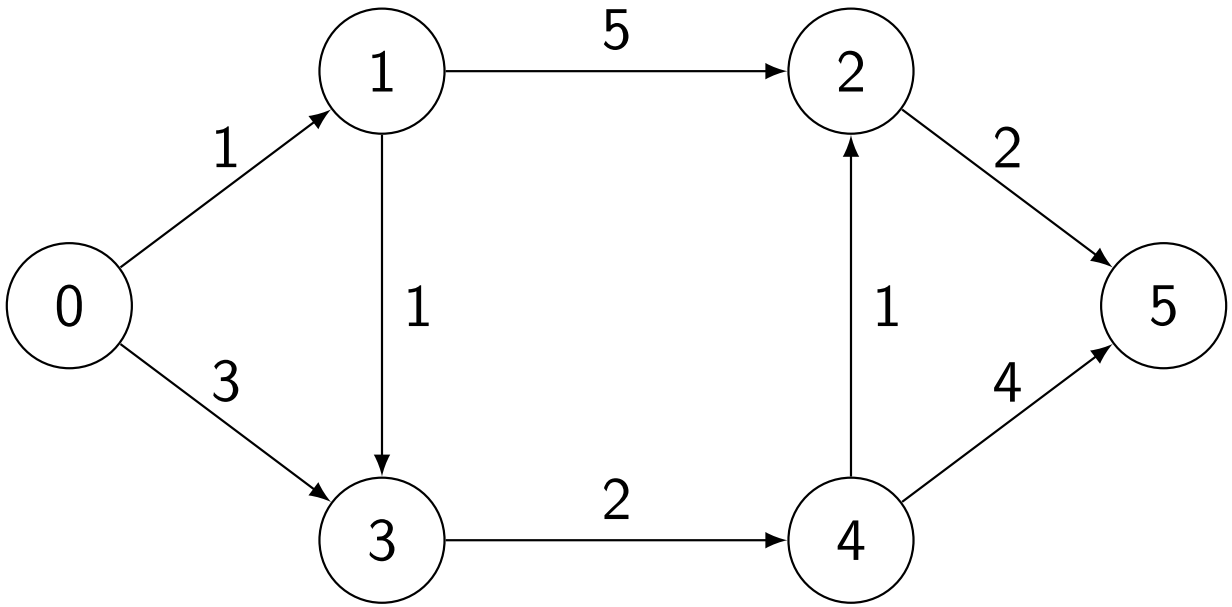
Bredden först (BFS)



Bredden först (BFS)



BFS i en viktad graf



Vilken väg väljs?

Inte så bra...

- Eftersom vi använder en kö antar BFS att alla bågar har samma vikt...
- Alltså ger den lösningen som traverserar minst antal noder

Idé:

- Vad händer om vi i stället hela tiden väljer den nod i kön som representerar kortast sträcka?
- Dijkstras algoritm

Dijkstra

```
bool bfs(int from, int to) {
  priority_queue<...> q; q.push(from);

  while (!q.empty()) {
    int current = q.top(); q.pop();
    graph[current].visited = true;
    // om current är vårt mål är vi klara

    for (int x : graph[current].edges) {
      // om x inte redan är besökt, markera
      // den som besökt och lägg på kö
    }
  }
}
```

Dijkstra – egenskaper

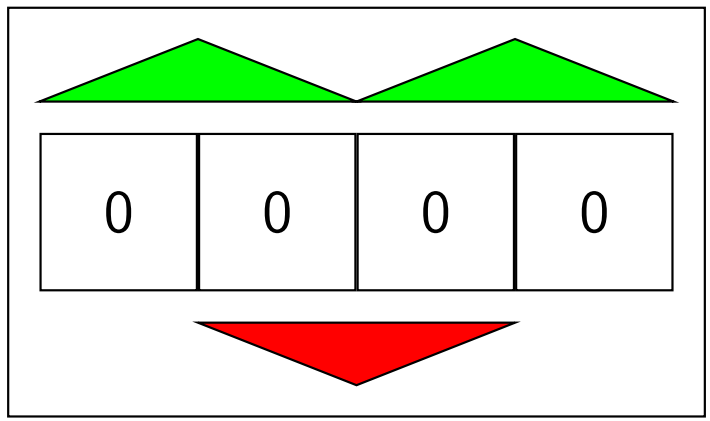
Hittar kortaste vägen i en *viktad graf*

- Tid: $\mathcal{O}(|E| + |V| \log(|V|))$
- Ofta långsammare och mer komplicerad att implementera än BFS
- Kan optimeras med hjälp av heurustik: A^*
- Förutsätter att det inte finns några **negativa cykler**
  Bellman-Ford klarar negativa cykler, men tar $\mathcal{O}(|E||V|)$
- Att hitta kortaste vägen mellan alla par av noder görs snabbare med
  Floyd-Warshall, vilken tar $\mathcal{O}(|V|^3)$ jämfört med
  $\mathcal{O}(|V|^3 \log(|V|))$

LINKÖPING UNIVERSITY

Problem

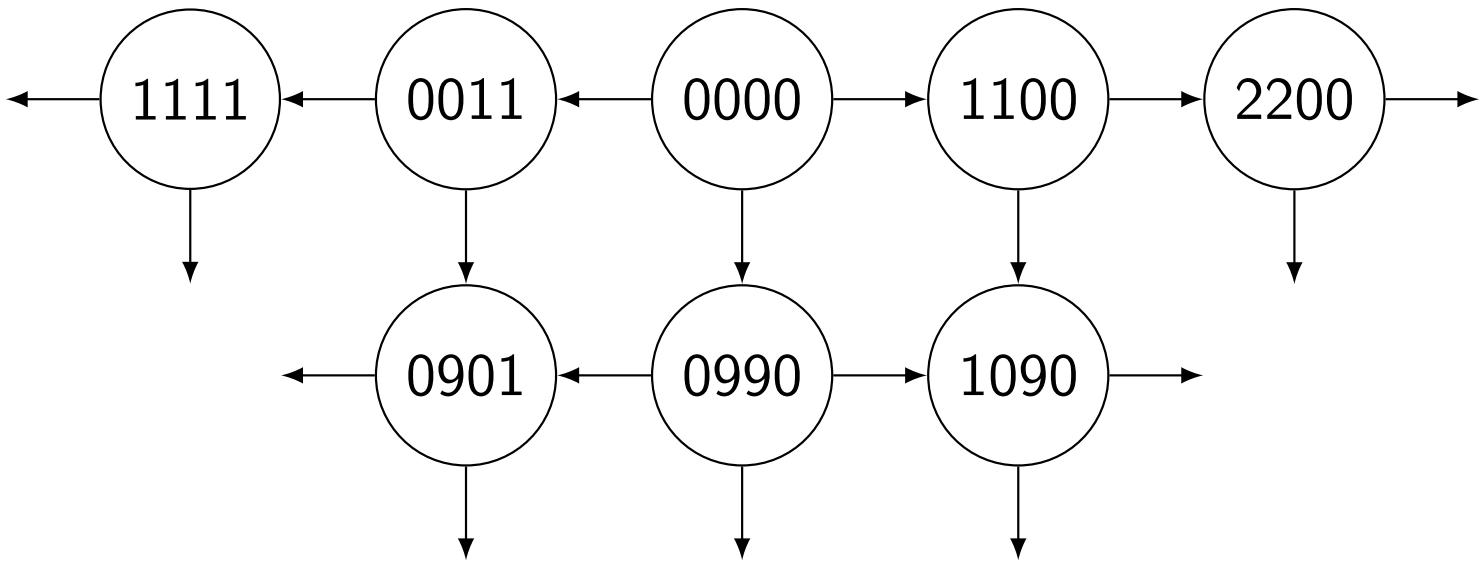
Du har hittat ett gammalt kodlås som du vill låsa upp. Du vet att koden är 3146, och just nu visar siffrorna på låset 0000. Låset har knappar för att öka eller minska de olika siffrorna, dock är många av knapparna trasiga. Hur skriver du in koden med så få knapptryck som möjligt? Låset ser ut som följer:



Lösningsidé

Representera problemet som en riktad graf:

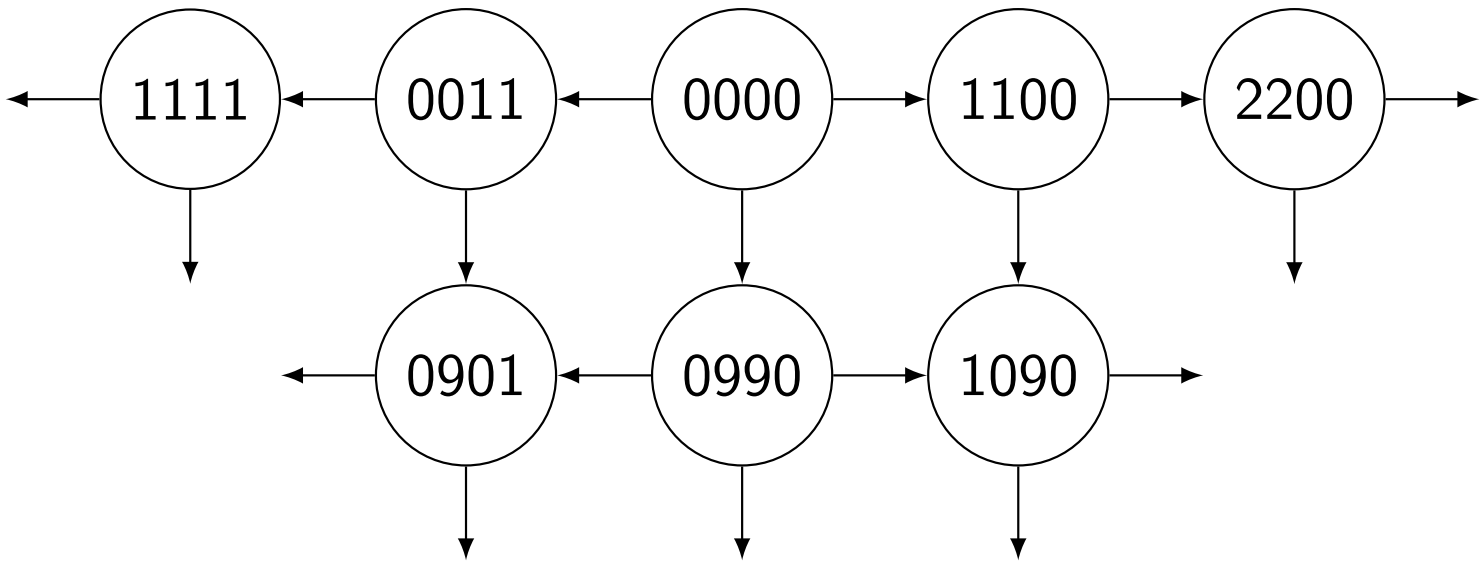
- En nod för varje kombination
- Varje båge motsvarar ett knapptryck



Lösningsidé

Representera problemet som en riktad graf:

- Oviktad graf $\Rightarrow$ BFS
- Varje väg motsvarar en lösning!



LINKÖPING UNIVERSITY

I kursen framöver

- Denna veckan
  - Se till att börja på lab 3 under veckan
- Nästa föreläsning
  - Fler grafalgoritmer, uppspännande träd etc.
- Uppgifter i Kattis
  - `hidingplaces` (enkel)
    Var är det bäst att gömma sig från en springare på ett schackbräde?
  - `coast` (svårare)
    Hur lång är kustlinjen på en given karta?

Filip Strömbäck

www.liu.se

LINKÖPING UNIVERSITY