Graph theory toolkit

SOI 2019, Sarnen, Gary Ye

February 11, 2019

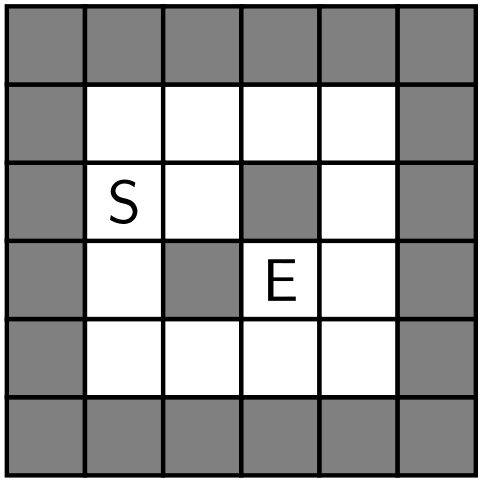
Introduction

- Previous lecture: Graph theory basics
- This lecture: Graph theory toolkit to solve many problems:
  - Implicit graphs
  - Reduction to graph problem
  - Modification of a graph problem
  - State space graph

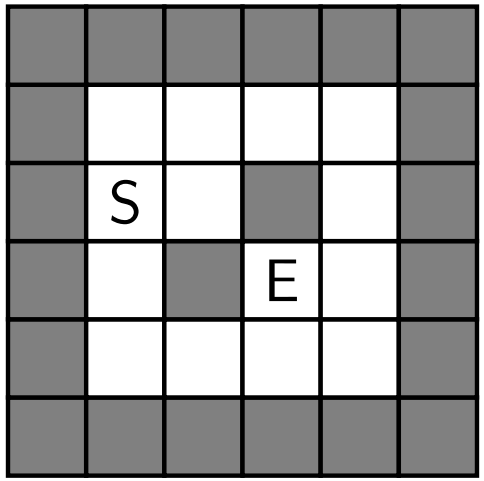
Outline

Problem: Maze

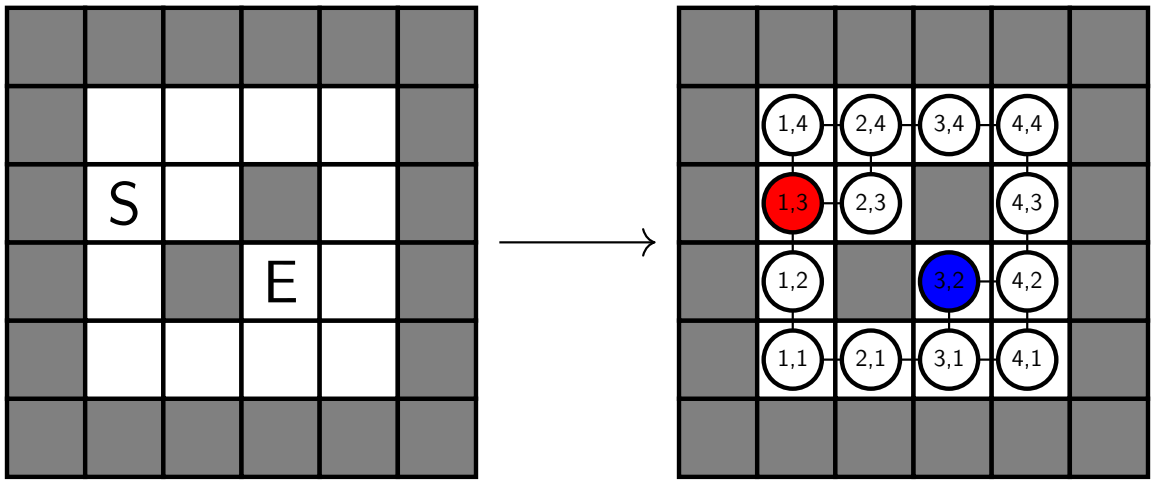
- Grey Cell ... Wall (unpassable)
- S ... Starting position
- E ... End position
- Each step: up, down, left or right.
- Goal: find path with minimum number of steps.



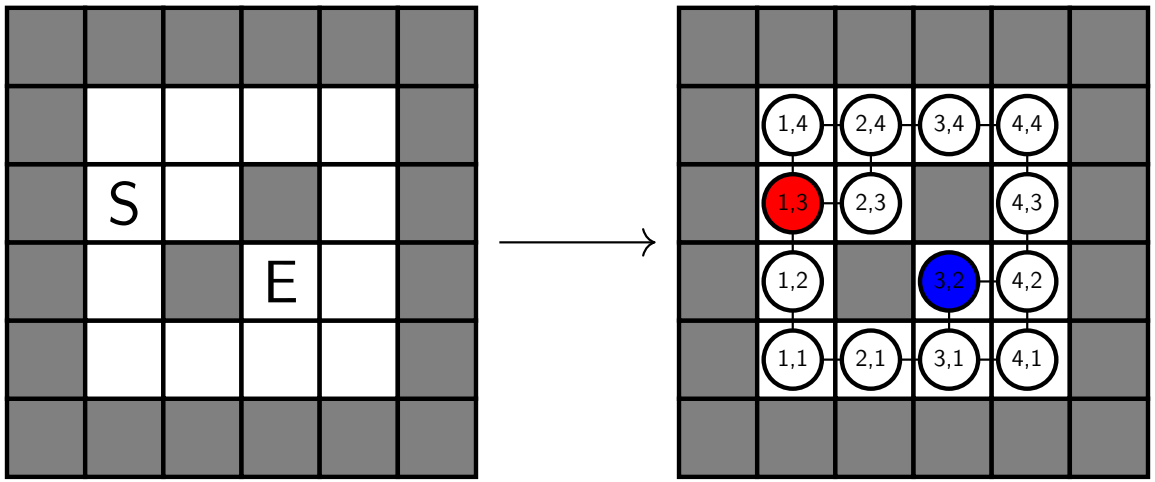
Idea: Convert to Graph



Idea: Convert to Graph

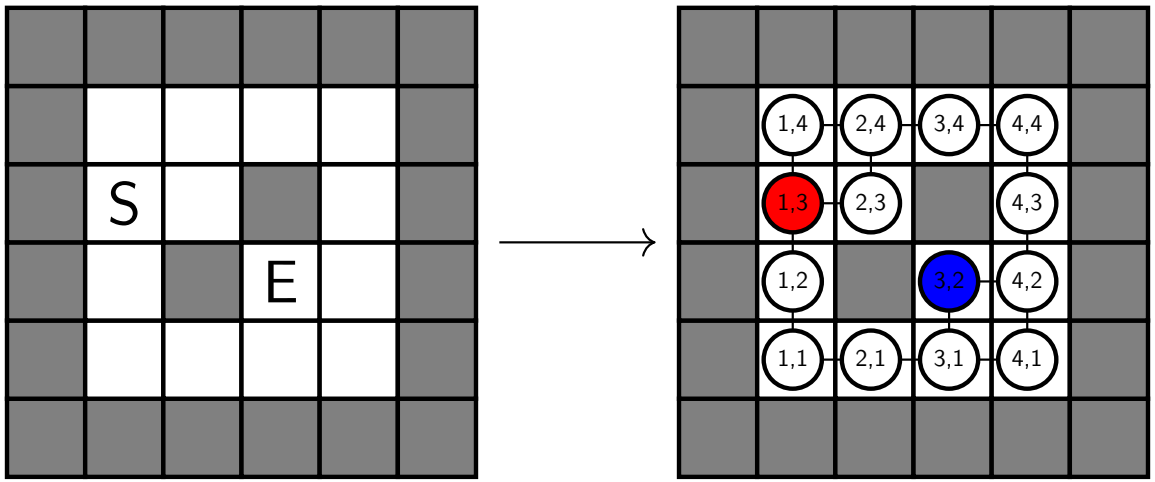


Idea: Convert to Graph



→ Then apply shortest path algorithm on unweighted graph!

Idea: Convert to Graph



→ Then apply shortest path algorithm on unweighted graph!

→ BFS!

Graph representation: Explicit

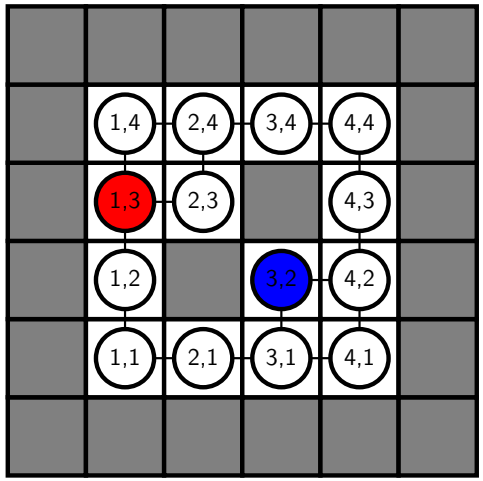
Store the graph explicitly in an adjacency list.

Graph representation: Explicit

Store the graph explicitly in an adjacency list.

```
vector<vector<char>> maze = read(); // input
vector<vector<vector<pair<int,int>>>> adj = init();
for(int i = 0; i < maze.size(); i++) {
  for(int j = 0; j < maze[i].size(); j++)  {
    // code for adding edges to adj[i][j]
  }
}
```

Graph representation: Explicit



Example:

```
adj[1][3] = {(1, 4),
             (2, 3),
             (1, 2)};
adj[3][2] = {(4, 2),
             (3, 1)};
```

Graph representation: Explicit

Downsides ☹:

- Additional memory required (up to $\mathcal{O}(n^2)$ though)
- Additional implementation time required (only relevant for contests)

Graph representation: Implicit

Instead of storing the edges explicitly, compute them using a simple rule:

Graph representation: Implicit

Instead of storing the edges explicitly, compute them using a simple rule:

Rule

There is an edge between cell c_1 and cell c_2 iff c_2 is immediately adjacent to c_1 and both cells do not represent a wall.

Graph representation: Implicit

Instead of storing the edges explicitly, compute them using a simple rule:

Rule

There is an edge between cell c_1 and cell c_2 iff c_2 is immediately adjacent to c_1 and both cells do not represent a wall.

Explicit

```
// adj is precomputed
for(pair<int, int> q: adj[i][j]) {
  // do something with neighbor q
}
```

Graph representation: Implicit

Instead of storing the edges explicitly, compute them using a simple rule:

Rule

There is an edge between cell c_1 and cell c_2 iff c_2 is immediately adjacent to c_1 and both cells do not represent a wall.

Explicit

```
// adj is precomputed
for(pair<int, int> q: adj[i][j]) {
  // do something with neighbor q
}
```

Implicit

```
if(maze[i][j] != '#') {
  if(maze[i+1][j] != '#') {
    // do something with neighbor (i+1, j)
  }
  if(maze[i-1][j] != '#') {
    // do something with neighbor (i-1, j)
  }
  if(maze[i][j+1] != '#') {
    // do something with neighbor (i, j+1)
  }
  if(maze[i][j-1] != '#') {
    // do something with neighbor (i, j-1)
  }
}
```

Maze: Lessons Learned

- We converted an ASCII grid problem into a graph problem
- We used an implicit graph representation to save memory and implementation time

Outline

Problem: Easteregg (SOI 2017)

Formal statement

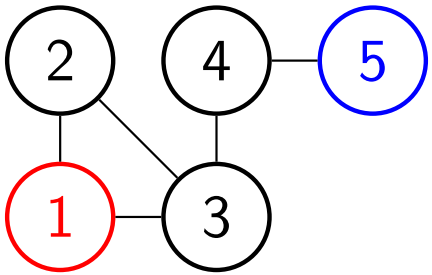
Given an undirected, unweighted graph with two starting vertices a and b. Choose a vertex c and two paths $p = [a, ..., c]$, $q = [b, ..., c]$ of equal length ($|p| = |q|$) such that $|p|$ (or $|q|$) is minimized.

Problem: Easteregg (SOI 2017)

Formal statement

Given an undirected, unweighted graph with two starting vertices a and b. Choose a vertex c and two paths $p = [a, ..., c]$, $q = [b, ..., c]$ of equal length ($|p| = |q|$) such that $|p|$ (or $|q|$) is minimized.

Example



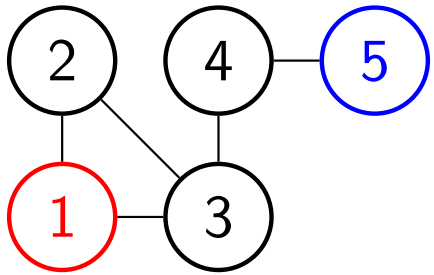
$a = 1, b = 5$

Problem: Easteregg (SOI 2017)

Formal statement

Given an undirected, unweighted graph with two starting vertices a and b. Choose a vertex c and two paths $p = [a, ..., c]$, $q = [b, ..., c]$ of equal length ($|p| = |q|$) such that $|p|$ (or $|q|$) is minimized.

Example



$a = 1, b = 5$

Solution: Choose $c = 3$ and $p = 1, 2, 3$ and $q = 5, 4, 3$.

Problem: Easteregg (SOI 2017)

Solution Idea

If P is the shortest path between a and b:

Problem: Easteregg (SOI 2017)

Solution Idea

If P is the shortest path between a and b:
$\rightarrow$ choose the first half and the second half as the two paths and the vertex in the center vertex as the meeting point.

Problem: Easteregg (SOI 2017)

Solution Idea

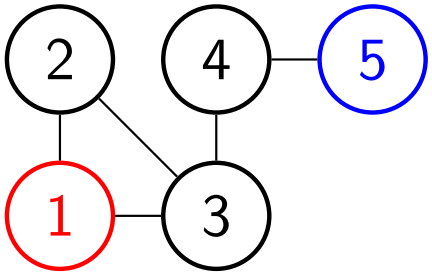
If P is the shortest path between a and b:
→ choose the first half and the second half as the two paths and the vertex in the center vertex as the meeting point.
→ but only works if the length of the path P is even!

Problem: Easteregg (SOI 2017)

Solution Idea

If P is the shortest path between a and b:
→ choose the first half and the second half as the two paths and the vertex in the center vertex as the meeting point.
→ but only works if the length of the path P is even!

Illustration

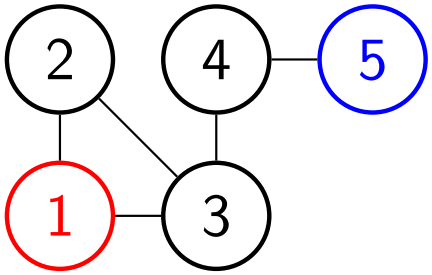


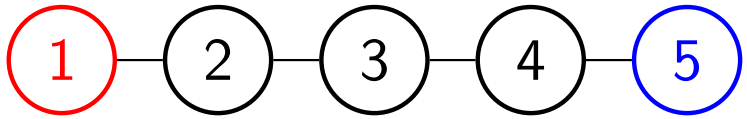
Problem: Easteregg (SOI 2017)

Solution Idea

If P is the shortest path between a and b:
→ choose the first half and the second half as the two paths and the vertex in the center vertex as the meeting point.
→ but only works if the length of the path P is even!

Illustration





The path above represents the shortest path with an even length, therefore we can choose $c = 3$.

Problem: Easteregg (SOI 2017)

We have reduced our problem to an easier problem ☺:

Easier problem

Find the shortest path P between a and b of *even* length.

Problem: Easteregg (SOI 2017)

We have reduced our problem to an easier problem ☺:

Easier problem

Find the shortest path P between a and b of *even* length.

However ...

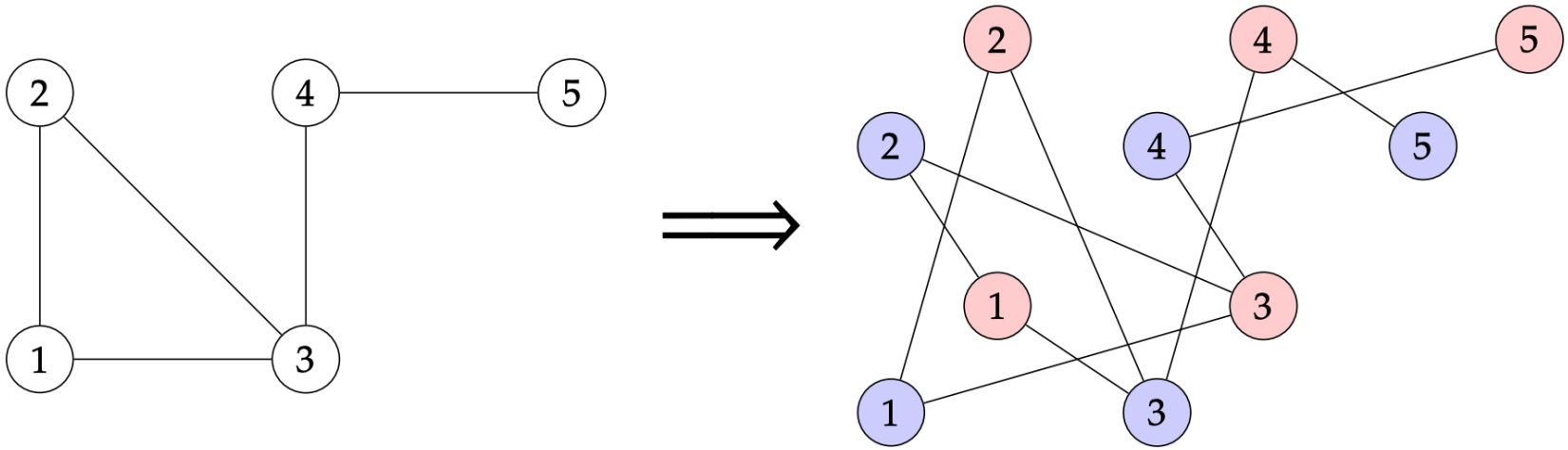
- Using BFS does not guarantee us that the path is even!
- If the shortest path P between a and b has an odd length, it does not mean that there exists no answer!

Trick: Graph Duplication

We modify our graph by duplicating every node and every edge:

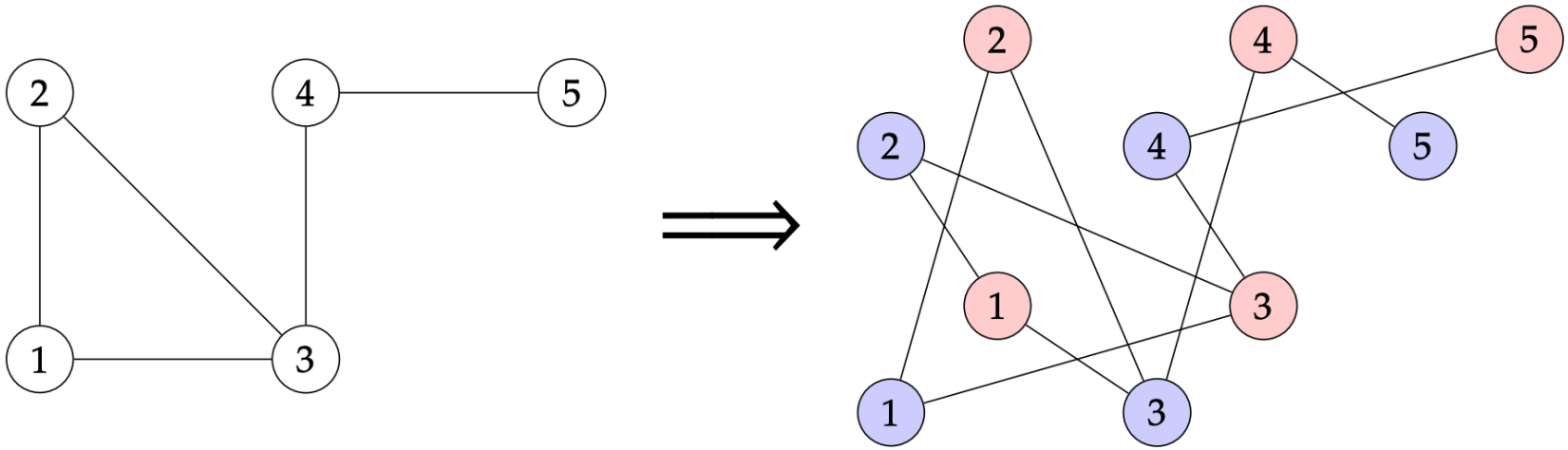
Trick: Graph Duplication

We modify our graph by duplicating every node and every edge:



Trick: Graph Duplication

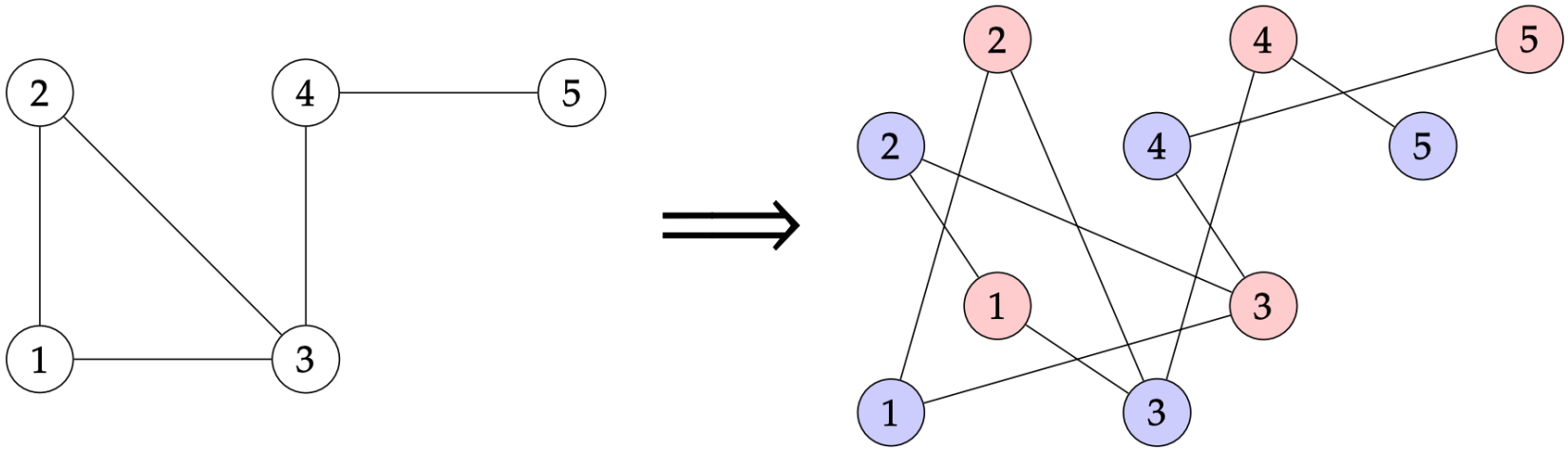
We modify our graph by duplicating every node and every edge:



→ whether the length of a path P is even can be checked by comparing the colors of the start and end point

Trick: Graph Duplication

We modify our graph by duplicating every node and every edge:

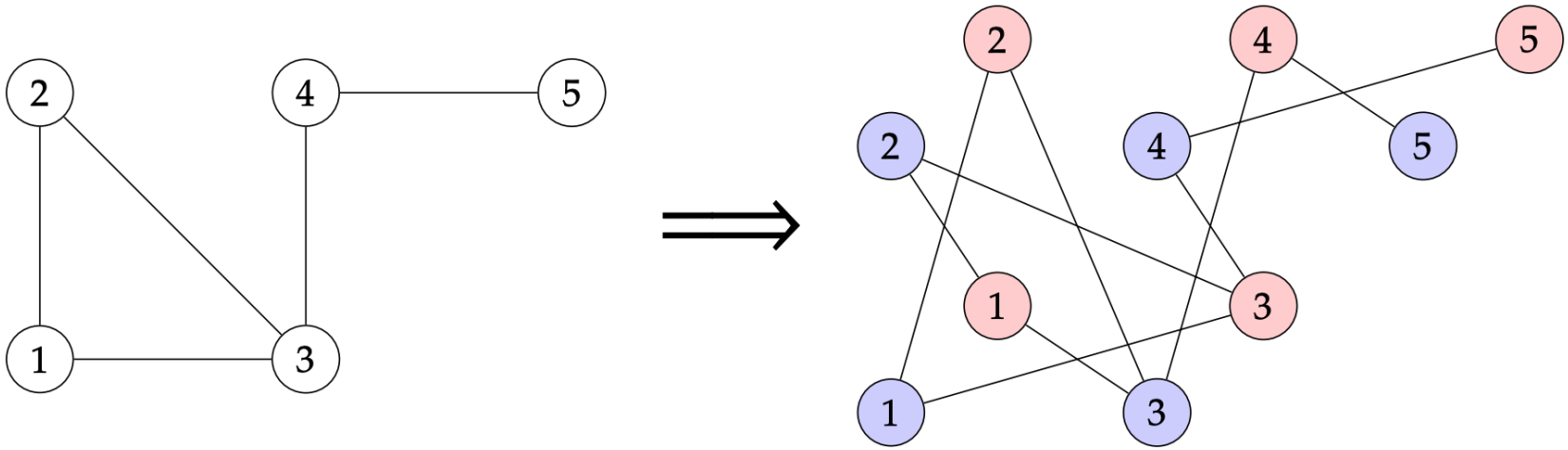


→ whether the length of a path P is even can be checked by comparing the colors of the start and end point

→ Example: $P = 1, 2, 3, 1, 2$ has an even length because $P_{\text{first}} = 1$ and $P_{\text{last}} = 2$ have the same color.

Trick: Graph Duplication

We modify our graph by duplicating every node and every edge:



→ whether the length of a path P is even can be checked by comparing the colors of the start and end point

→ Example: $P = 1, 2, 3, 1, 2$ has an even length because $P_{\text{first}} = 1$ and $P_{\text{last}} = 2$ have the same color.

Conclusion for the new graph

A path between two nodes of the same color will always have even length

Problem: Easteregg (SOI 2017)

Easier problem

Find the shortest path P between a and b of *even* length.

Problem: Easteregg (SOI 2017)

Easier problem

Find the shortest path P between a and b of *even* length.

Solution

1. Perform a graph duplication to get a new graph G'.

Problem: Easteregg (SOI 2017)

Easier problem

Find the shortest path P between a and b of *even* length.

Solution

1. Perform a graph duplication to get a new graph G'.
2. Find the shortest path between a and b in the new graph G' (or between a and b) using BFS.

Easteregg: Lessons Learned

- Reducing a complex problem to an easier problem.
- Even if we are given a graph, we should not be afraid to modify it!
- Here we have seen one technique called "Graph Duplication".
- Other techniques:
  - duplicate graph more than two times
  - add dummy nodes
  - add dummy edges
  - ... be creative!

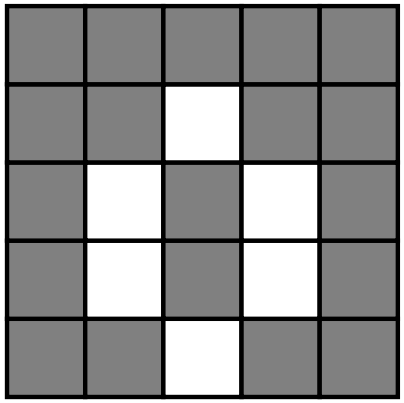
Outline

Problem: Lights Out!

Description

Given a 5x5 grid with some lights on or off. Clicking a cell toggles that cell and each of its immediate neighbors.
Goal: Turn off the lights with minimum number of clicks.

Example

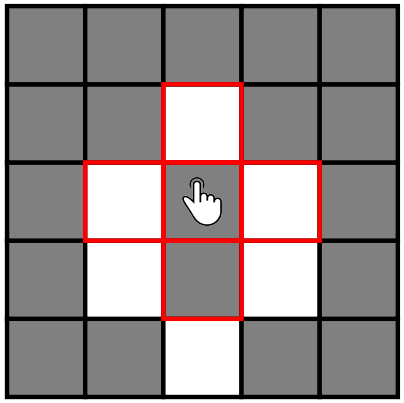


Problem: Lights Out!

Description

Given a 5x5 grid with some lights on or off. Clicking a cell toggles that cell and each of its immediate neighbors.
Goal: Turn off the lights with minimum number of clicks.

Example

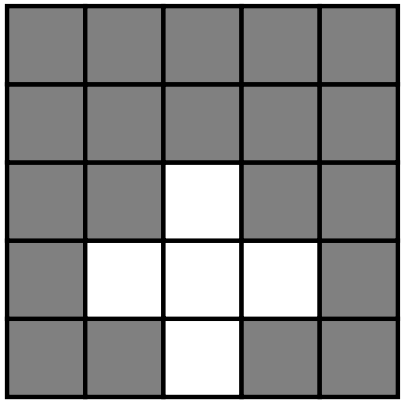


Problem: Lights Out!

Description

Given a 5x5 grid with some lights on or off. Clicking a cell toggles that cell and each of its immediate neighbors.
Goal: Turn off the lights with minimum number of clicks.

Example

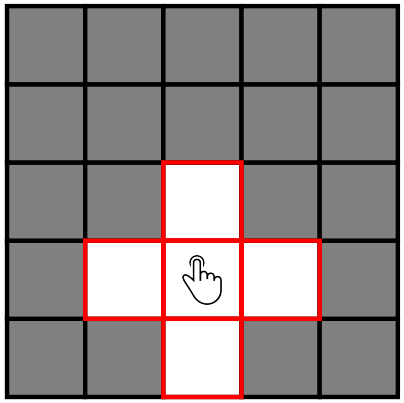


Problem: Lights Out!

Description

Given a 5x5 grid with some lights on or off. Clicking a cell toggles that cell and each of its immediate neighbors.
Goal: Turn off the lights with minimum number of clicks.

Example

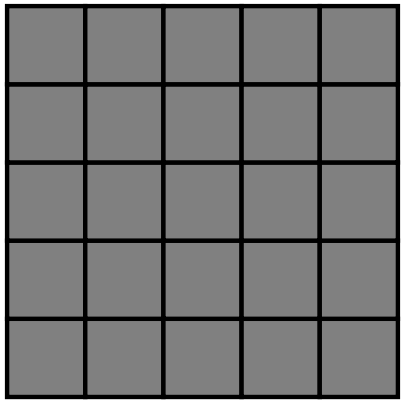


Problem: Lights Out!

Description

Given a 5x5 grid with some lights on or off. Clicking a cell toggles that cell and each of its immediate neighbors.
Goal: Turn off the lights with minimum number of clicks.

Example



State space

Definition

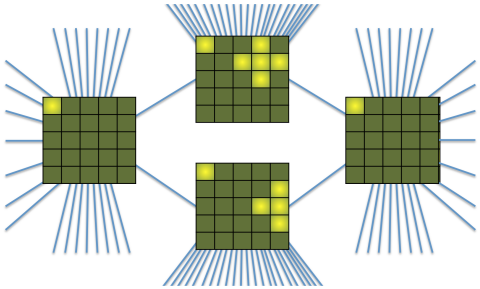
Set of states that a problem can be in. The set of states forms a graph where two states are connected if there is an operation that can be performed to transform the first state into the second.

State space

Definition

Set of states that a problem can be in. The set of states forms a graph where two states are connected if there is an operation that can be performed to transform the first state into the second.

Example: Lights Out!



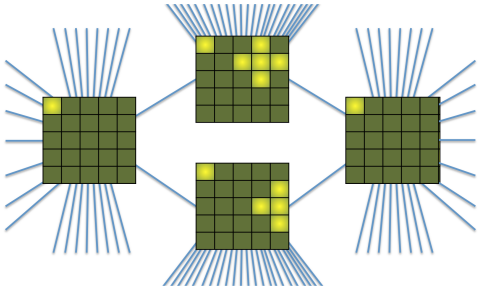
(Source: CMU)

State space

Definition

Set of states that a problem can be in. The set of states forms a graph where two states are connected if there is an operation that can be performed to transform the first state into the second.

Example: Lights Out!



(Source: CMU)

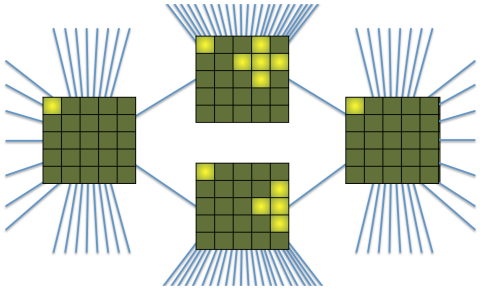
Q: How many states?

State space

Definition

Set of states that a problem can be in. The set of states forms a graph where two states are connected if there is an operation that can be performed to transform the first state into the second.

Example: Lights Out!



(Source: CMU)

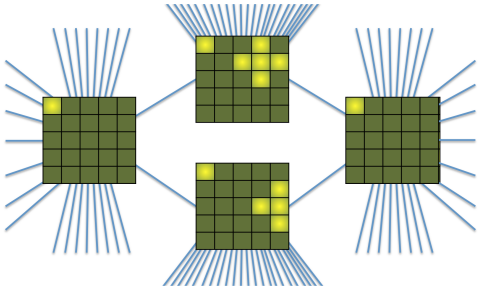
Q: How many states? A: $2^{25} \approx 32$mio.

State space

Definition

Set of states that a problem can be in. The set of states forms a graph where two states are connected if there is an operation that can be performed to transform the first state into the second.

Example: Lights Out!



(Source: CMU)

Q: How many states? A: $2^{25} \approx 32$mio.

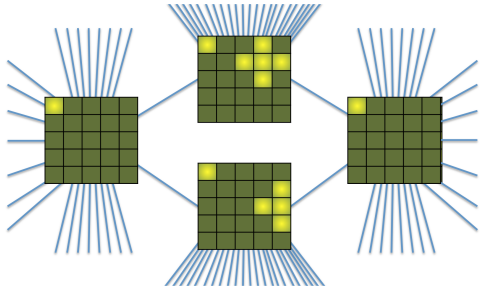
Q: How many edges?

State space

Definition

Set of states that a problem can be in. The set of states forms a graph where two states are connected if there is an operation that can be performed to transform the first state into the second.

Example: Lights Out!



(Source: CMU)

Q: How many states? A: $2^{25} \approx 32$mio.

Q: How many edges? A: $25 \cdot 2^{24} \approx 400$mio.

Problem: Lights Out!

In order to avoid storing 32 mio. states and 400 mio. edges we will use an implicit graph!

Problem: Lights Out!

In order to avoid storing 32 mio. states and 400 mio. edges we will use an implicit graph!

Implicitly retrieve the neighbor states of current

```
for(int i = 0; i < 5; i++) {
  for(int j = 0; j < 5; j++) {
    // current.click(i, j) simulates a click on (i, j)
    Board neighbor = current.click(i, j);
    // do something with neighbor
  }
}
```

Problem: Lights Out!

In order to avoid storing 32 mio. states and 400 mio. edges we will use an implicit graph!

Implicitly retrieve the neighbor states of current

```
for(int i = 0; i < 5; i++) {
  for(int j = 0; j < 5; j++) {
    // current.click(i, j) simulates a click on (i, j)
    Board neighbor = current.click(i, j);
    // do something with neighbor
  }
}
```

Solution

Perform a BFS on the state space graph (implicit graph)

Lights Out: Lessons Learned

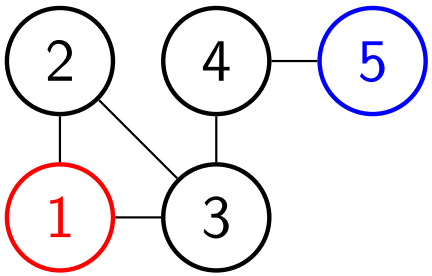
- Concepts of state space graph
- A very simple reduction between a problem into a space state graph problem

Easter Egg (Revisited)

Instead of graph duplication we define a state space for the problem.
We define the state space as (CurrentNode, CurrentLengthParity).

Easter Egg (Revisited)

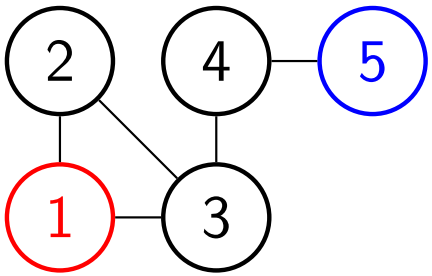
Instead of graph duplication we define a state space for the problem.
We define the state space as (CurrentNode, CurrentLengthParity).



$$\text{StateSpace} = \{$$
$$(1,0),(1,1),$$
$$(2,0),(2,1),$$
$$(3,0),(3,1),$$
$$(4,0),(4,1),$$
$$(5,0),(5,1)\}$$

Easter Egg (Revisited)

Instead of graph duplication we define a state space for the problem.
We define the state space as (CurrentNode, CurrentLengthParity).



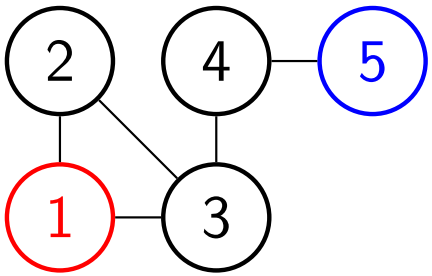
$$\text{StateSpace} = \{(1,0),(1,1),(2,0),(2,1),(3,0),(3,1),(4,0),(4,1),(5,0),(5,1)\}$$

Neighbor states of (u, p)

If (u, v) is an edge in the given graph, then $(v, 1-p)$ is a neighbor state of (u, p).

Easter Egg (Revisited)

Instead of graph duplication we define a state space for the problem.
We define the state space as (CurrentNode, CurrentLengthParity).



$$\text{StateSpace} = \{(1,0),(1,1),(2,0),(2,1),(3,0),(3,1),(4,0),(4,1),(5,0),(5,1)\}$$

Neighbor states of (u, p)

If (u, v) is an edge in the given graph, then $(v, 1-p)$ is a neighbor state of (u, p).

Solution

Find shortest path from $(a, 0)$ to $(b, 0)$ in state space graph.

Shortest Path of Even Length with even Sum

Formal Statement

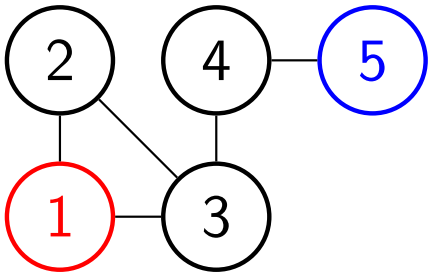
Find the shortest path P between a and b, such that the length of P is even and the sum of vertex numbers in P is even as well (i.e. ΣP_i is even).

Shortest Path of Even Length with even Sum

Formal Statement

Find the shortest path P between a and b, such that the length of P is even and the sum of vertex numbers in P is even as well (i.e. ΣP_i is even).

Use state: (CurrentNode, CurrentLengthParity, **CurrentSumParity**).



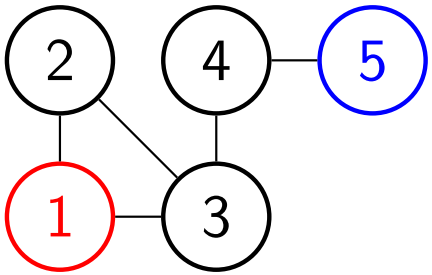
$$\begin{aligned}&\text{StateSpace} = \{\\&(1,0,0),(1,0,1),(1,1,0),(1,0,0),\\&(2,0,0),(2,0,1),(2,1,0),(2,0,0),\\&\ldots,\\&(5,0,0),(5,0,1),(5,1,0),(5,0,0)\}\end{aligned}$$

Shortest Path of Even Length with even Sum

Formal Statement

Find the shortest path P between a and b, such that the length of P is even and the sum of vertex numbers in P is even as well (i.e. ΣP_i is even).

Use state: (CurrentNode, CurrentLengthParity, **CurrentSumParity**).



$$\begin{aligned}&\text{StateSpace} = \{\\&(1,0,0),(1,0,1),(1,1,0),(1,0,0),\\&(2,0,0),(2,0,1),(2,1,0),(2,0,0),\\&\ldots,\\&(5,0,0),(5,0,1),(5,1,0),(5,0,0)\}\end{aligned}$$

Neighbor states of (u, p, q)

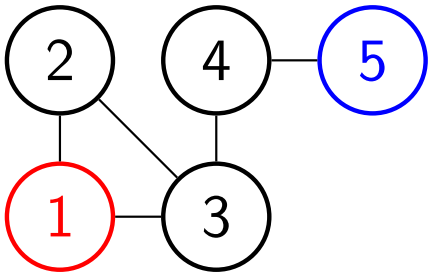
If (u, v) is an edge in the given graph, then $(v, 1-p, (v+q) \bmod 2)$ is a neighbor state of (u, p, q).

Shortest Path of Even Length with even Sum

Formal Statement

Find the shortest path P between a and b, such that the length of P is even and the sum of vertex numbers in P is even as well (i.e. ΣP_i is even).

Use state: (CurrentNode, CurrentLengthParity, **CurrentSumParity**).



StateSpace = $\{$
$(1,0,0),(1,0,1),(1,1,0),(1,0,0),$
$(2,0,0),(2,0,1),(2,1,0),(2,0,0),$
$\ldots,$
$(5,0,0),(5,0,1),(5,1,0),(5,0,0)\}$

Neighbor states of (u,p,q)

If (u,v) is an edge in the given graph, then $(v,1-p,(v+q) \bmod 2)$ is a neighbor state of (u,p,q).

Solution: Find shortest path from $(a,0,a \bmod 2)$ to $(b,0,0)$.

Lessons Learned

- Using a state space graph to represent a graph problem.
- We can use multiple dimensions to represent our state.

Coding Tasks

Task 1: Farmland ($\approx$ Maze)

Try to improve the code that implicitly retrieves the neighbors of (i, j).
Hint: Use loops

Task 2: Easteregg

Solve it two times:

- Duplicate the graph explicitly and perform a BFS.
- Use the state $(\text{node}, \text{parity})$ and use implicit graphs.

Task 3: Lights Out! (Optional)

Hint: Use a 32 bit integer to represent the state.