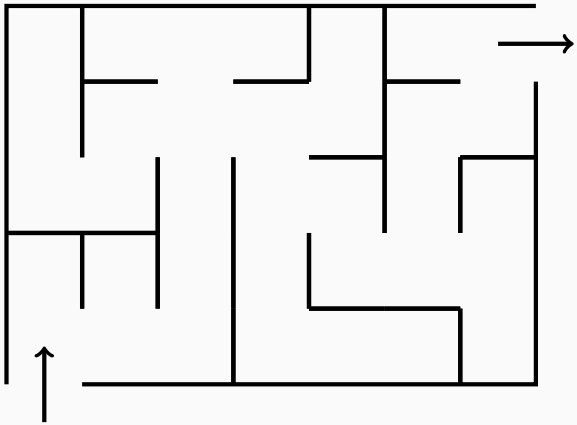
Parcours de graphes

Christoph Dürr Jill-Jênn Vie

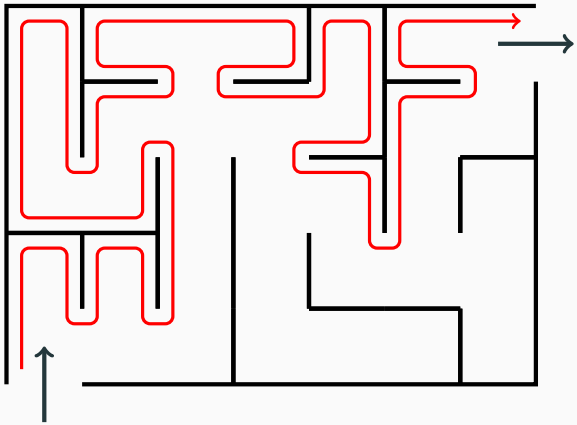
SU November Camp

2 novembre 2021

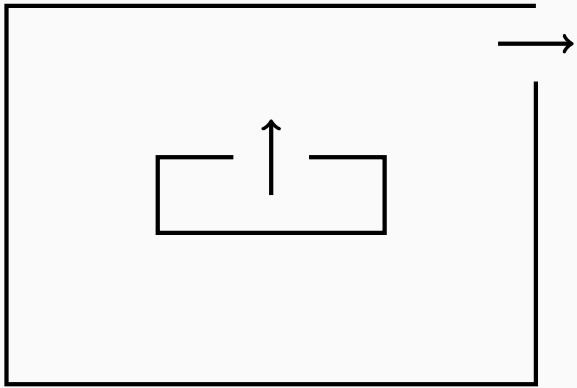
Un labyrinthe



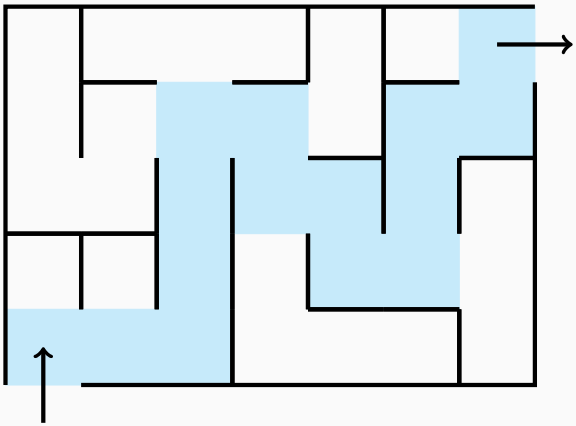
Parcours main gauche



Parcours main gauche ?



Parcours en profondeur (DFS) avec une pile



```
from tryalgo.dfs import dfs  # pip install tryalgo
prec = dfs(laby_graph)
```

Depth-first search

```
process(node)
    mark this node
    for each neighbor of node
        if neighbor is not marked
            process(neighbor)
```

Depth-first search

```
process(node)
    mark this node
    for each neighbor of node
        if neighbor is not marked
            process(neighbor)
```

```
def dfs_recursive(graph, node, seen):
    seen[node] = True
    for neighbor in graph[node]:
        if not seen[neighbor]:
            dfs_recursive(graph, neighbor, seen)
```

Depth-first search, iterative

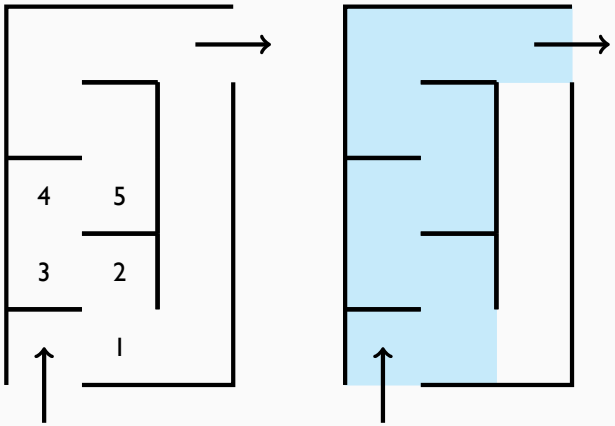
```
dfs(node)
    todo ← empty stack, then push node to todo
    while todo is not empty
        node ← pop from todo
        mark this node
        for each neighbor of node
            if neighbor is not marked
                push neighbor to todo
```

Depth-first search, iterative

```
dfs(node)
    todo ← empty stack, then push node to todo
    while todo is not empty
        node ← pop from todo
        mark this node
        for each neighbor of node
            if neighbor is not marked
                push neighbor to todo
```

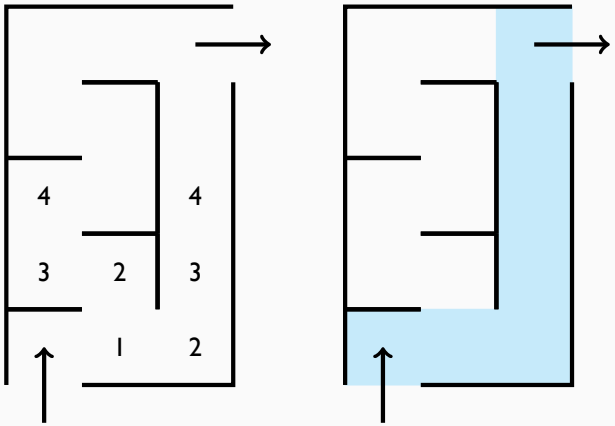
```python
def dfs_iterative(graph, start, seen):
    seen[start] = True
    to_visit = [start]
    while to_visit:
        node = to_visit.pop()
        for neighbor in graph[node]:
            if not seen[neighbor]:
                seen[neighbor] = True
                to_visit.append(neighbor)
```

Parcours en profondeur (DFS) optimal ?




```
from tryalgo.dfs import dfs
prec = dfs(laby_graph)
```

Parcours en largeur (BFS) avec une file




```
from tryalgo.bfs import bfs
dist, prec = bfs(laby_graph)
```

Breadth-first search

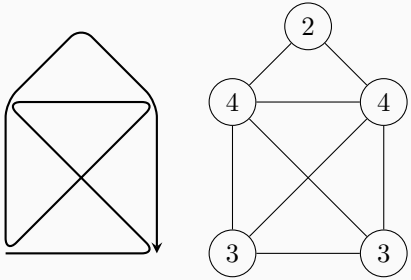
```
bfs(node)
    todo ← empty queue
    mark node
    push node to todo
    while todo is not empty
        node ← pop from todo
        for each neighbor of node
            if neighbor is not marked
                mark neighbor
                push neighbor to todo
```

Le graphe de Paris

Énoncé original

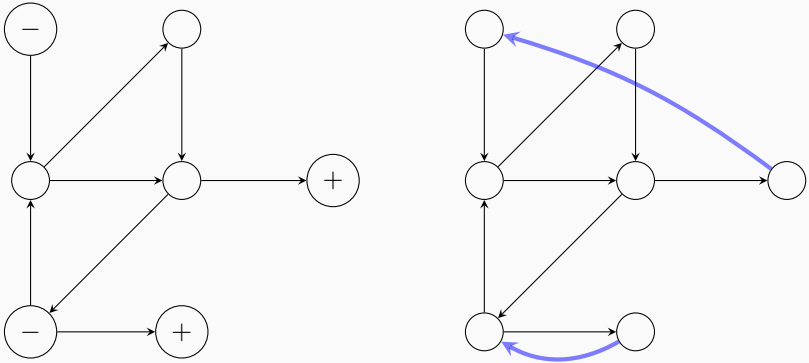
- On vous donne 8 voitures partant de Google Paris sur le graphe de Paris à 11348 intersections, 17958 rues.
- Chaque rue est étiquetée par une distance en mètres et un temps de parcours en secondes.
- Certaines rues sont à double sens, d'autres ne le sont pas.
- Comment explorer un maximum de km de Paris en 15 heures ?

La meilleure solution : graphes eulériens



Condition : 0 ou 2 nœuds ayant un nombre impair de voisins.

Eulérianiser Paris par des plus courts chemins



Certains nœuds ont trop d'arcs entrants, d'autres en manquent.

Idée : les coupler par des plus courts chemins.

Références

- TryAlgo Maps in Paris, un notebook Jupyter pour jouer vos algorithmes sur le graphe de Paris
- Pour en savoir plus sur l'épreuve Google Hash Code 2014
  par Antoine Amarilli