Recursion

Frank Burkholder

Objectives

After this lecture you will be able to:

- Define recursion (in computer science)
- List the two cases of a recursive algorithm
  - base case, recursive case
- Step through `factorial`, a classic example of recursion
- List advantages/disadvantages of using recursion in your code
- Describe an application of recursion used in a capstone
  - Route finding between two points

Recursion (in computer science)

- Recursion is a method of solving a problem where the solution depends on solutions to smaller instances of the same problem.
- Recursion solves problems by using functions that call themselves from within their own code.
- Often iteration can be used to solve the same problem instead, but you need to determine the number of iterations beforehand to solve the problem.

--Wikipedia

General structure of a recursive algorithm

```
def recursive_function(input):
    check if input is some value
        if so, return the base case (non-recursive value)
    otherwise
        return the recursive case (recursive_function(modified input))
```

Classic recursion example: factorial

3! = 3 * 2 * 1 = 6
5! = 5 * 4 * 3 * 2 * 1 = 120
0! = 1

Classic recursion example: factorial

3! = 3 * 2 * 1 = 6
5! = 5 * 4 * 3 * 2 * 1 = 120
0! = 1

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
```

Classic recursion example: factorial

3! = 3 * 2 * 1 = 6
5! = 5 * 4 * 3 * 2 * 1 = 120
0! = 1

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
```

base case

recursive case

Code investigation

`factorial.py`

Advantages / disadvantages of recursion

Advantages

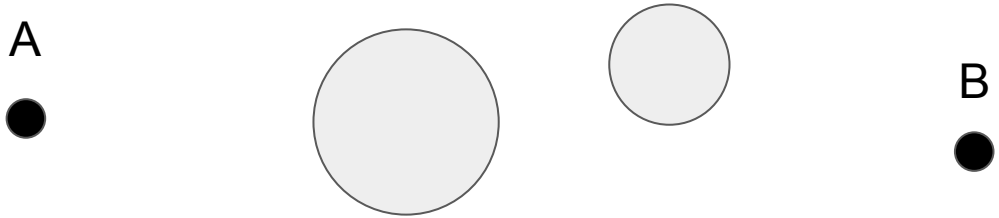
- Often results in easier-to-read code with less to debug
- Better than iteration if it’s not known beforehand how many iterations are required (tree-traversal)

Disadvantages

- Uses more memory (function add to the call stack with each recursive call)
- Can be slower than iteration (though saving results of past recursive calls - memoization - can mitigate this)

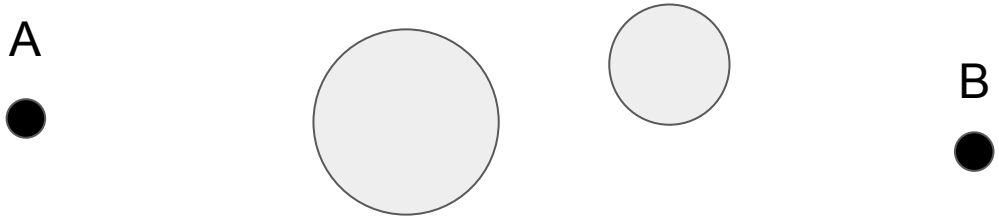
source

Example used in a capstone: route-finding



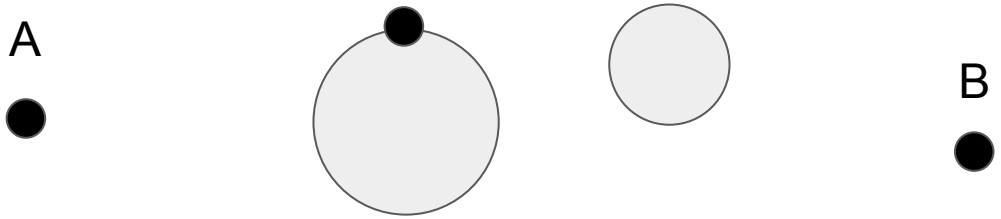
Find the shortest route between points A and B that doesn’t go through a circle obstruction.

Example used in a capstone: route-finding



get_route(start_pt, end_pt, obstructions)
    check if there are obstructions between the start_pt and end_pt
    if there are no obstructions:
        find the straight-line path *(base case)*
    if there are obstructions:
        determine a new_start_pt on the nearest obstruction
        remove that obstruction from possible obstruction list
        travel there
        get_route(new_start_pt, end_pt, obstructions) *(recursive case)*

Example used in a capstone: route-finding

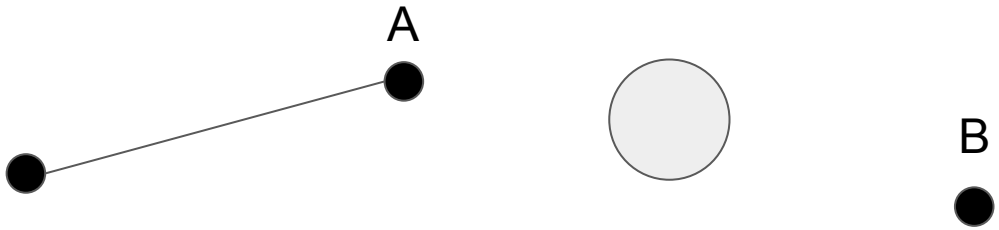



```
get_route(start_pt, end_pt, obstructions)
    check if there are obstructions between the start_pt and end_pt
    if there are no obstructions:
        find the straight-line path (base case)
    if there are obstructions:
        determine a new_start_pt on the nearest obstruction
        remove that obstruction from possible obstruction list
        travel there
        get_route(new_start_pt, end_pt, obstructions)
```

Example used in a capstone: route-finding

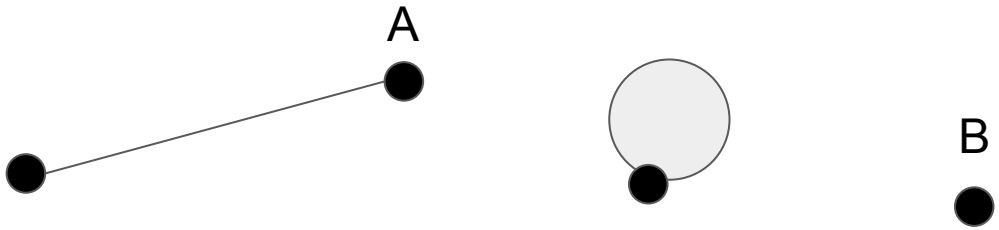

```
get_route(start_pt, end_pt, obstructions)
    check if there are obstructions between the start_pt and end_pt
    if there are no obstructions:
        find the straight-line path (base case)
    if there are obstructions:
        determine a new_start_pt on the nearest obstruction
        remove that obstruction from possible obstruction list
        travel there
        get_route(new_start_pt, end_pt, obstructions)
```

Example used in a capstone: route-finding



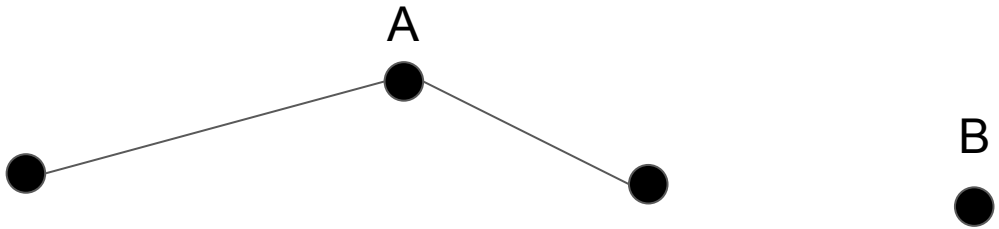

```
get_route(start_pt, end_pt, obstructions)
    check if there are obstructions between the start_pt and end_pt
    if there are no obstructions:
        find the straight-line path (base case)
    if there are obstructions:
        determine a new_start_pt on the nearest obstruction
        remove that obstruction from possible obstruction list
        travel there
        get_route(new_start_pt, end_pt, obstructions)
```

Example used in a capstone: route-finding



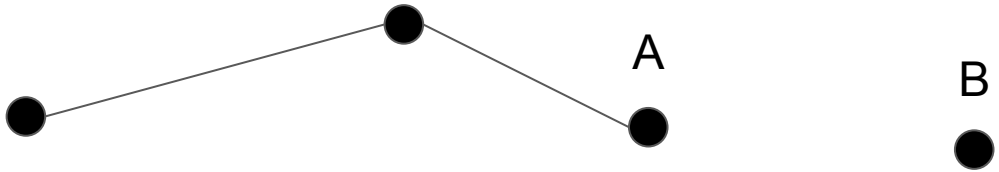

```
get_route(start_pt, end_pt, obstructions)
    check if there are obstructions between the start_pt and end_pt
    if there are no obstructions:
        find the straight-line path (base case)
    if there are obstructions:
        determine a new_start_pt on the nearest obstruction
        remove that obstruction from possible obstruction list
        travel there
        get_route(new_start_pt, end_pt, obstructions)
```

Example used in a capstone: route-finding



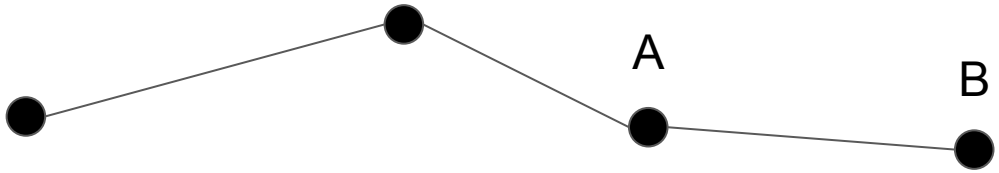

```
get_route(start_pt, end_pt, obstructions)
    check if there are obstructions between the start_pt and end_pt
    if there are no obstructions:
        find the straight-line path (base case)
    if there are obstructions:
        determine a new_start_pt on the nearest obstruction
        remove that obstruction from possible obstruction list
        travel there
        get_route(new_start_pt, end_pt, obstructions)
```

Example used in a capstone: route-finding




```
get_route(start_pt, end_pt, obstructions)
    check if there are obstructions between the start_pt and end_pt
    if there are no obstructions:
        find the straight-line path (base case)
    if there are obstructions:
        determine a new_start_pt on the nearest obstruction
        remove that obstruction from possible obstruction list
        travel there
        get_route(new_start_pt, end_pt, obstructions)
```

Example used in a capstone: route-finding




```
get_route(start_pt, end_pt, obstructions)
    check if there are obstructions between the start_pt and end_pt
    if there are no obstructions:
        find the straight-line path (base case)
    if there are obstructions:
        determine a new_start_pt on the nearest obstruction
        remove that obstruction from possible obstruction list
        travel there
        get_route(new_start_pt, end_pt, obstructions)
```

See it in a Capstone: UAV package delivery

https://github.com/Frank-W-B/UAV_delivery_project

Code example

`randomized_test_routing_not_smoothed_occasional_failures.py`

Objectives

After this lecture you will be able to:

- Define recursion (in computer science)
- List the two cases of a recursive algorithm
  - base case, recursive case
- Step through `factorial`, a classic example of recursion
- List advantages/disadvantages of using recursion in your code
- Describe an application of recursion used in a capstone
  - Route finding between two points