COMP 10280
Programming I (Conversion)

John Dunnion

School of Computer Science
University College Dublin

COMP 10280 Programming I (Conversion)/Lecture 12

Outline

Exhaustive Enumeration

- Exhaustive Enumeration is a technique of searching for a solution to a problem
- Navigating the search space
- We go through all possibilities until we get the right answer or exhaust the space of possibilities
- Very simple, even stupid, way to solve a problem
- Often the most practical approach
- Easy to implement and easy to understand
- Often fast enough for practical purposes

Python program 1 to find the cube root

```
# Finding the integer cube root of a number
# Program prompts the user for the number

# Prompt the user for a number
number = int(input('Enter the number for which you wish
                        to calculate the cube root:  '))
# If the number entered is negative, look for the cube root
                        of its negation
if number < 0:
    new_number = -number
else:
    new_number = number
cube_root = 0
while cube_root ** 3 < new_number:
    cube_root += 1

if cube_root ** 3 == new_number:
    if number < 0:
        cube_root = -cube_root
    print('Cube root of', number, 'is', cube_root)
else:
    print(number, 'is not a perfect cube.')

print('Finished!')
```

Python program 2 to find the cube root

```
# Finding the integer cube root of a number
# Program prompts the user for the number
# Uses abs function

# Prompt the user for a number
number = int(input('Enter the number for which you wish
                        to calculate the cube root:  '))

# Look for the cube root of the absolute value of the number

cube_root = 0
while cube_root ** 3 < abs(number):
    cube_root += 1

if cube_root ** 3 == abs(number):
    if number < 0:
        cube_root = -cube_root
    print('Cube root of', number, 'is', cube_root)
else:
    print(number, 'is not a perfect cube.')

print('Finished!')
```

Python program 3 to find the cube root

```
# Finding the integer cube root of a number
# Program prompts the user for the number
# Uses abs function
# Uses a for loop and a break statement

# Prompt the user for a number
number = int(input('Enter the number for which you wish
                        to calculate the cube root:  '))

# Look for the cube root of the absolute value of the number

for cube_root in range(abs(number) + 1):
    if cube_root ** 3 >= abs(number):
        break

if cube_root ** 3 == abs(number):
    if number < 0:
        cube_root = -cube_root
    print('Cube root of', number, 'is', cube_root)
else:
    print(number, 'is not a perfect cube.')

print('Finished!')
```

Finding an approximate solution (1)

- It is not always possible to find an exact solution to a problem
- For example, finding the square root of 4, finding the cube root of 1000
- For example, finding the square root of 1000, finding the cube root of 4
- These are not rational numbers
- They cannot be represented in a finite string of digits (either as an `int` or as a `float`)
- The problem as initially stated cannot be solved

Finding an approximate solution (2)

- Instead, we have to find an approximate solution to the problem
- The approximate solution should give us an answer that is close enough to the actual answer to be useful
- What is “close enough”?
- We usually define “close enough” in terms of some tolerance
- Often referred to ϵ (“epsilon”)
- The answer should lie within this tolerance or ϵ

Python program 1 to approximate the square root

```
# Finding the square root of a number
# Program prompts the user for the number
# Uses exhaustive enumeration to find an approximate solution

epsilon = 0.01
step = epsilon ** 2

numGuesses = 0
# Prompt the user for a number
number = float(input('Enter the number for which you wish
                     to calculate the square root:  '))

root = 0.0
while abs(number - root ** 2) >= epsilon and root <= number:
    root += step
    numGuesses += 1
    if numGuesses % 100000 == 0:
        print('Still running.  Number of guesses:', numGuesses)
print('Number of guesses:', numGuesses)
if abs(number - root ** 2) < epsilon:
    print('The approximate square root of', number, 'is', root)
else:
    print('Failed to find a square root of', number)
print('Finished!')
```

Python program 2 to approximate the square root

```
# Finding the square root of a number
# Program prompts the user for the number
# Uses exhaustive enumeration to find an approximate solution

epsilon = 0.01
step = epsilon ** 2

numGuesses = 0
# Prompt the user for a number
number = float(input('Enter the number for which you wish
                     to calculate the square root:  '))

root = 0.0
while abs(number - root ** 2) >= epsilon and root ** 2 <= number:
    root += step
    numGuesses += 1
    if numGuesses % 100000 == 0:
        print('Still running.  Number of guesses:', numGuesses)
print('Number of guesses:', numGuesses)
if abs(number - root ** 2) < epsilon:
    print('The approximate square root of', number, 'is', root)
else:
    print('Failed to find a square root of', number)
print('Finished!')
```