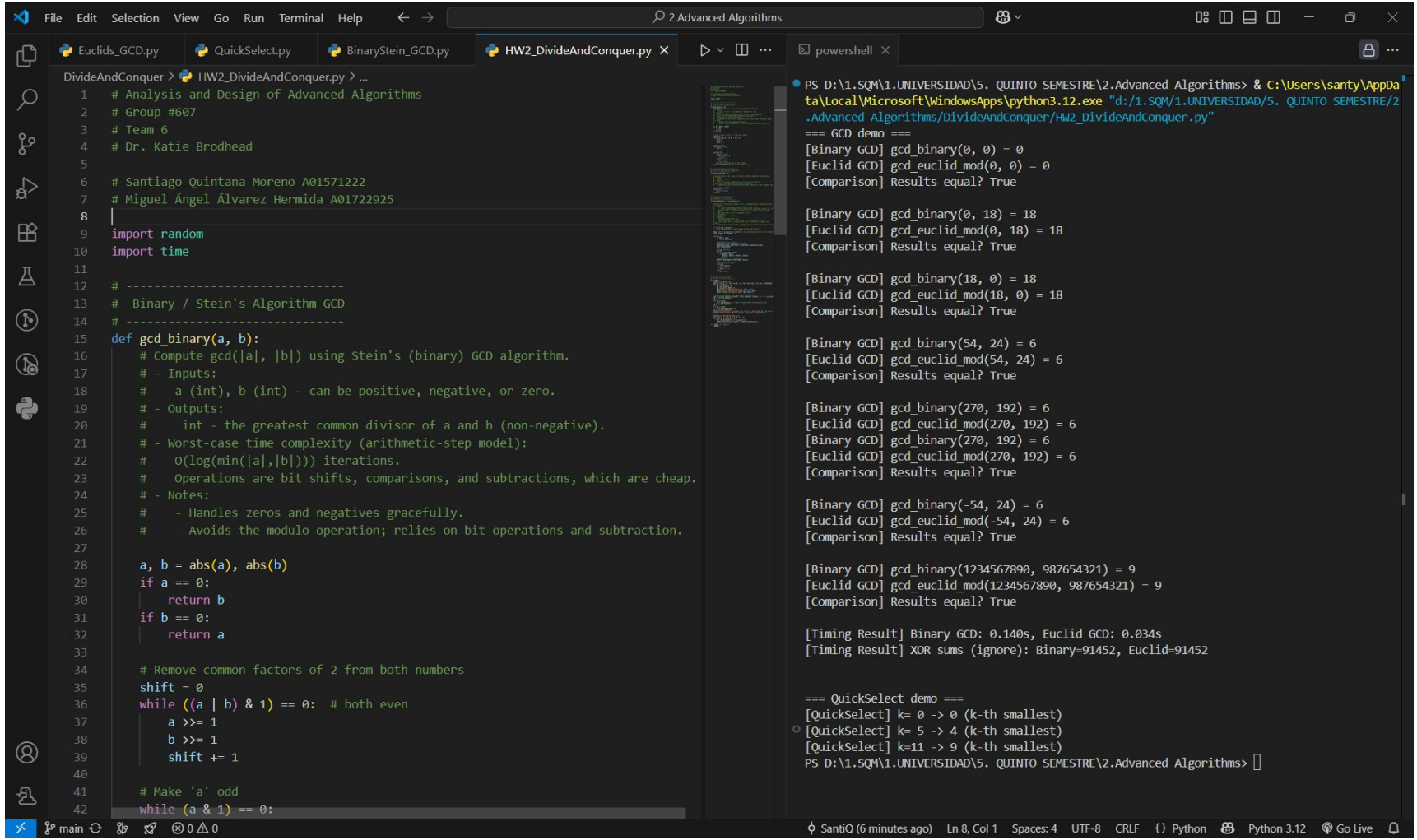
Tecnológico de Monterrey

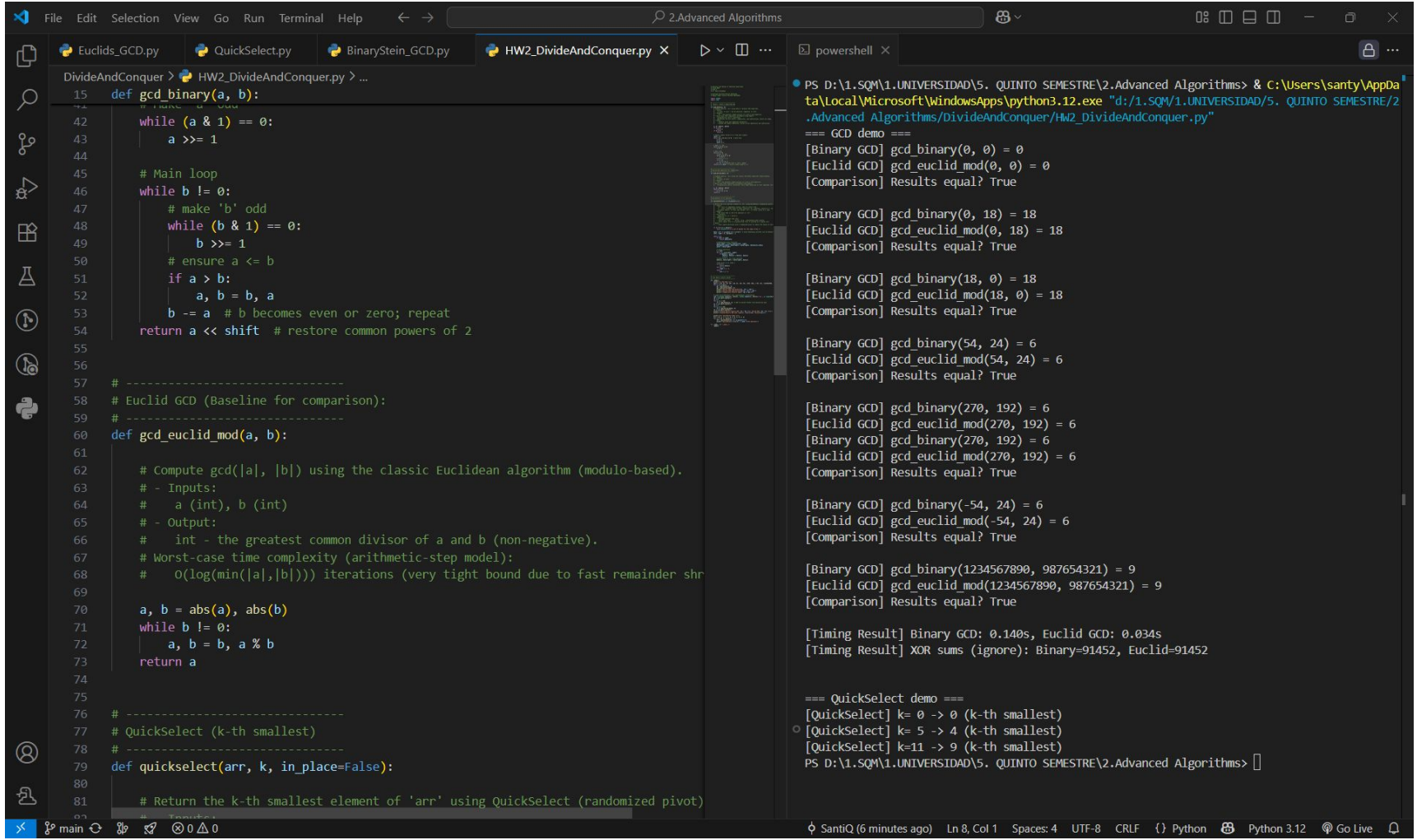
Tecnológico de Monterrey - Campus Monterrey
School of Engineering and Sciences
Engineering in Computational Technologies
Analysis and Design of Advanced Algorithms

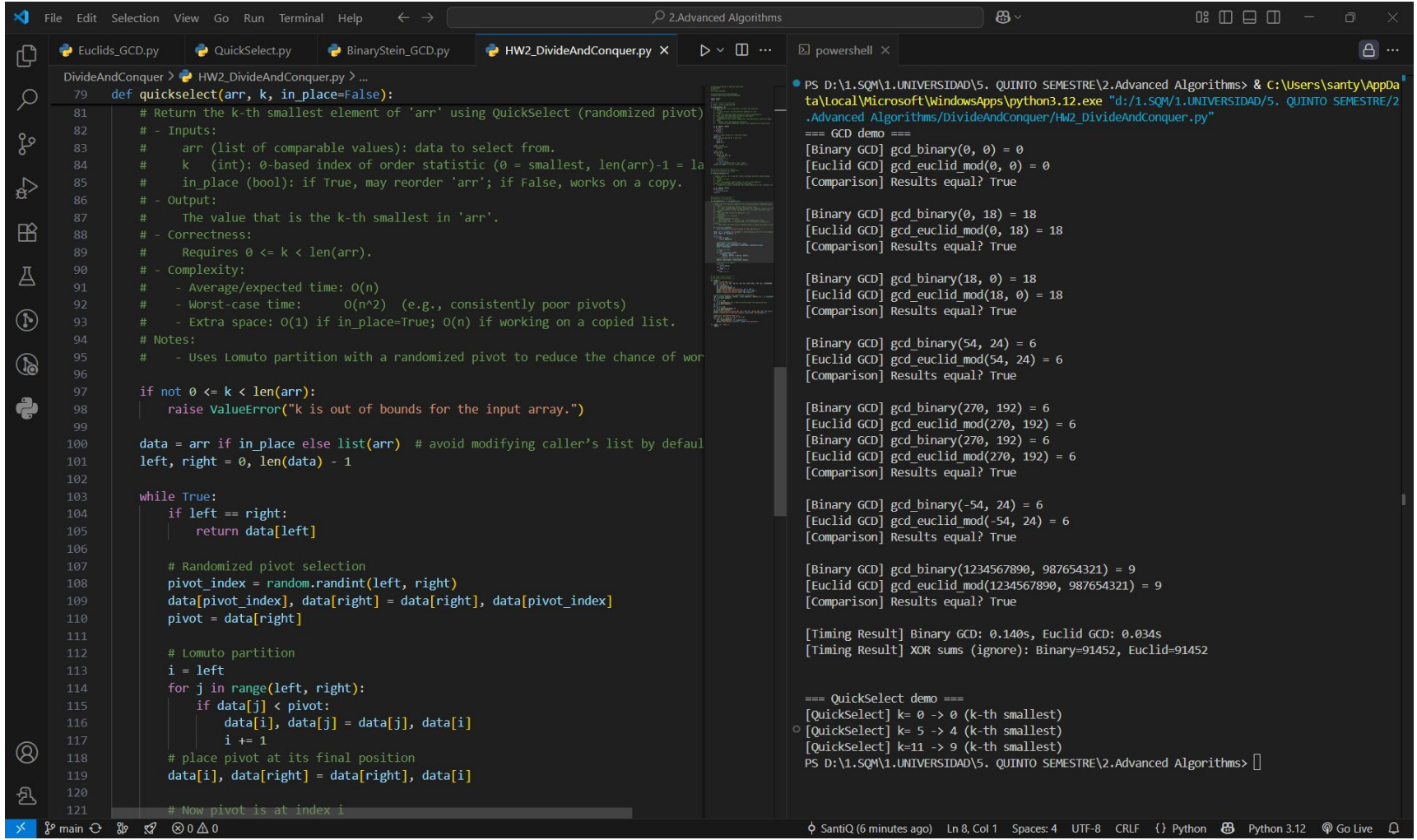
Homework 2:
Divide and Conquer

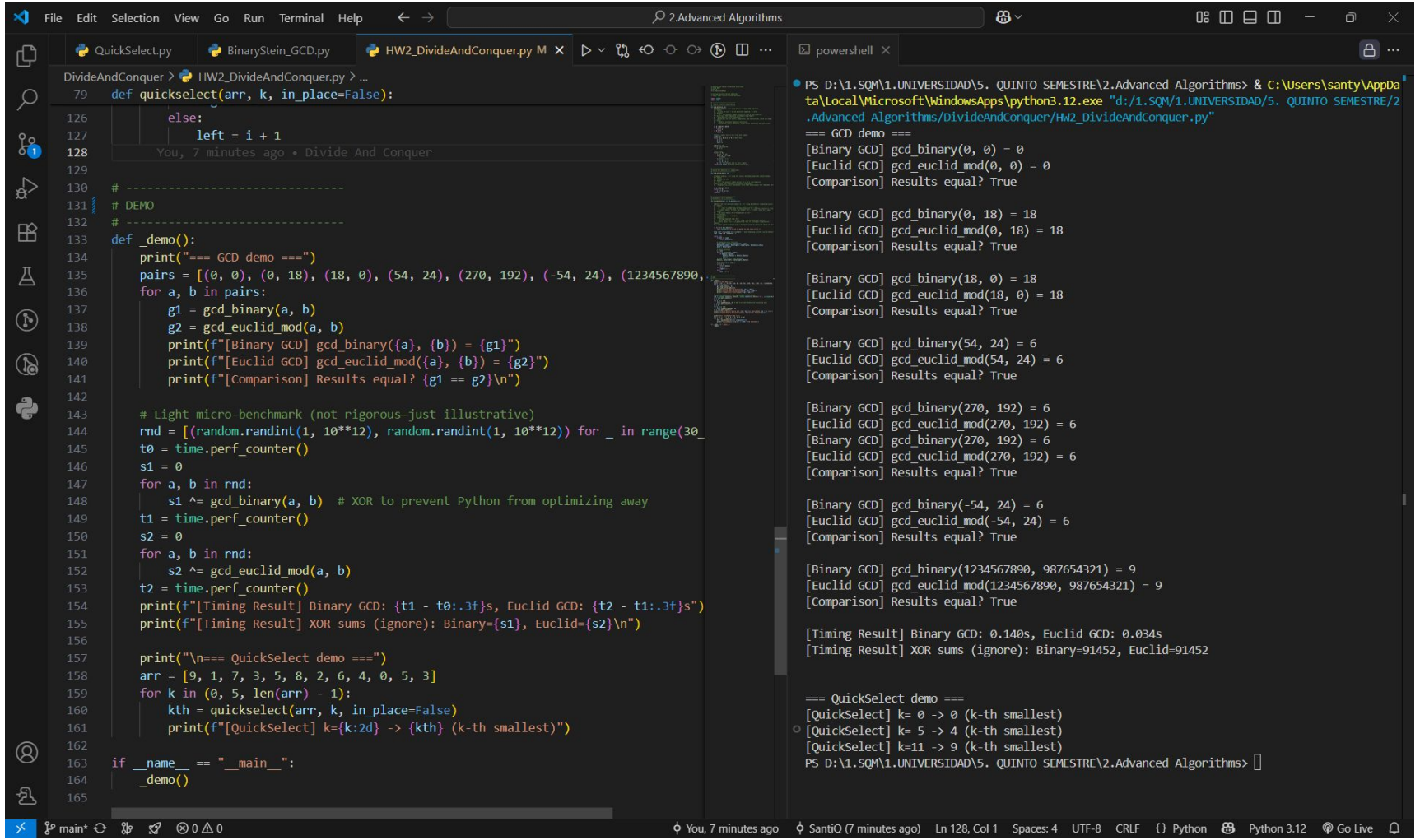
Group: 607
Team #6
Dr. Katie Brodhead

Santiago Quintana Moreno A01571222
Miguel Ángel Álvarez Hermida A01722925


File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
Euclids_GCD.py
QuickSelect.py
BinaryStein_GCD.py
HW2_DivideAndConquer.py
DivideAndConquer > HW2_DivideAndConquer.py > ...
# Analysis and Design of Advanced Algorithms
# Group #607
# Team 6
# Dr. Katie Brodhead
# Santiago Quintana Moreno A01571222
# Miguel Ángel Álvarez Hermida A01722925
import random
import time
# ------------------------------
# Binary / Stein's Algorithm GCD
# ------------------------------
def gcd_binary(a, b):
# Compute gcd(|a|, |b|) using Stein's (binary) GCD algorithm.
# - Inputs:
# a (int), b (int) - can be positive, negative, or zero.
# - Outputs:
# int - the greatest common divisor of a and b (non-negative).
# - Worst-case time complexity (arithmetic-step model):
# O(log(min(|a|,|b|))) iterations.
# Operations are bit shifts, comparisons, and subtractions, which are cheap.
# - Notes:
# - Handles zeros and negatives gracefully.
# - Avoids the modulo operation; relies on bit operations and subtraction.
a, b = abs(a), abs(b)
if a == 0:
return b
if b == 0:
return a
# Remove common factors of 2 from both numbers
shift = 0
while ((a | b) & 1) == 0: # both even
a >>= 1
b >>= 1
shift += 1
# Make 'a' odd
while (a & 1) == 0:
powershell
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/DivideAndConquer/HW2_DivideAndConquer.py"
=== GCD demo ===
[Binary GCD] gcd_binary(0, 0) = 0
[Euclid GCD] gcd_euclid_mod(0, 0) = 0
[Comparison] Results equal? True
[Binary GCD] gcd_binary(0, 18) = 18
[Euclid GCD] gcd_euclid_mod(0, 18) = 18
[Comparison] Results equal? True
[Binary GCD] gcd_binary(18, 0) = 18
[Euclid GCD] gcd_euclid_mod(18, 0) = 18
[Comparison] Results equal? True
[Binary GCD] gcd_binary(54, 24) = 6
[Euclid GCD] gcd_euclid_mod(54, 24) = 6
[Comparison] Results equal? True
[Binary GCD] gcd_binary(270, 192) = 6
[Euclid GCD] gcd_euclid_mod(270, 192) = 6
[Binary GCD] gcd_binary(270, 192) = 6
[Euclid GCD] gcd_euclid_mod(270, 192) = 6
[Comparison] Results equal? True
[Binary GCD] gcd_binary(-54, 24) = 6
[Euclid GCD] gcd_euclid_mod(-54, 24) = 6
[Comparison] Results equal? True
[Binary GCD] gcd_binary(1234567890, 987654321) = 9
[Euclid GCD] gcd_euclid_mod(1234567890, 987654321) = 9
[Comparison] Results equal? True
[Timing Result] Binary GCD: 0.140s, Euclid GCD: 0.034s
[Timing Result] XOR sums (ignore): Binary=91452, Euclid=91452
=== QuickSelect demo ===
[QuickSelect] k= 0 -> 0 (k-th smallest)
[QuickSelect] k= 5 -> 4 (k-th smallest)
[QuickSelect] k=11 -> 9 (k-th smallest)
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
main
SantiQ (6 minutes ago)
Ln 8, Col 1
Spaces: 4
UTF-8
CRLF
Python
Python 3.12
Go Live


File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
Euclids_GCD.py
QuickSelect.py
BinaryStein_GCD.py
HW2_DivideAndConquer.py
DivideAndConquer > HW2_DivideAndConquer.py > ...
def gcd_binary(a, b):
while (a & 1) == 0:
a >>= 1
# Main loop
while b != 0:
# make 'b' odd
while (b & 1) == 0:
b >>= 1
# ensure a <= b
if a > b:
a, b = b, a
b -= a # b becomes even or zero; repeat
return a << shift # restore common powers of 2
# ------------------------------
# Euclid GCD (Baseline for comparison):
# ------------------------------
def gcd_euclid_mod(a, b):
# Compute gcd(|a|, |b|) using the classic Euclidean algorithm (modulo-based).
# - Inputs:
# a (int), b (int)
# - Output:
# int - the greatest common divisor of a and b (non-negative).
# Worst-case time complexity (arithmetic-step model):
# O(log(min(|a|,|b|))) iterations (very tight bound due to fast remainder shr
a, b = abs(a), abs(b)
while b != 0:
a, b = b, a % b
return a
# ------------------------------
# QuickSelect (k-th smallest)
# ------------------------------
def quickselect(arr, k, in_place=False):
# Return the k-th smallest element of 'arr' using QuickSelect (randomized pivot)
powershell
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/DivideAndConquer/HW2_DivideAndConquer.py"
=== GCD demo ===
[Binary GCD] gcd_binary(0, 0) = 0
[Euclid GCD] gcd_euclid_mod(0, 0) = 0
[Comparison] Results equal? True
[Binary GCD] gcd_binary(0, 18) = 18
[Euclid GCD] gcd_euclid_mod(0, 18) = 18
[Comparison] Results equal? True
[Binary GCD] gcd_binary(18, 0) = 18
[Euclid GCD] gcd_euclid_mod(18, 0) = 18
[Comparison] Results equal? True
[Binary GCD] gcd_binary(54, 24) = 6
[Euclid GCD] gcd_euclid_mod(54, 24) = 6
[Comparison] Results equal? True
[Binary GCD] gcd_binary(270, 192) = 6
[Euclid GCD] gcd_euclid_mod(270, 192) = 6
[Binary GCD] gcd_binary(270, 192) = 6
[Euclid GCD] gcd_euclid_mod(270, 192) = 6
[Comparison] Results equal? True
[Binary GCD] gcd_binary(-54, 24) = 6
[Euclid GCD] gcd_euclid_mod(-54, 24) = 6
[Comparison] Results equal? True
[Binary GCD] gcd_binary(1234567890, 987654321) = 9
[Euclid GCD] gcd_euclid_mod(1234567890, 987654321) = 9
[Comparison] Results equal? True
[Timing Result] Binary GCD: 0.140s, Euclid GCD: 0.034s
[Timing Result] XOR sums (ignore): Binary=91452, Euclid=91452
=== QuickSelect demo ===
[QuickSelect] k= 0 -> 0 (k-th smallest)
[QuickSelect] k= 5 -> 4 (k-th smallest)
[QuickSelect] k=11 -> 9 (k-th smallest)
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
main
SantiQ (6 minutes ago)
Ln 8, Col 1
Spaces: 4
UTF-8
CRLF
Python
Python 3.12
Go Live


File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
Euclids_GCD.py
QuickSelect.py
BinaryStein_GCD.py
HW2_DivideAndConquer.py
DivideAndConquer > HW2_DivideAndConquer.py > ...
def quickselect(arr, k, in_place=False):
# Return the k-th smallest element of 'arr' using QuickSelect (randomized pivot)
# - Inputs:
# arr (list of comparable values): data to select from.
# k (int): 0-based index of order statistic (0 = smallest, len(arr)-1 = la
# in_place (bool): if True, may reorder 'arr'; if False, works on a copy.
# - Output:
# The value that is the k-th smallest in 'arr'.
# - Correctness:
# Requires 0 <= k < len(arr).
# - Complexity:
# - Average/expected time: O(n)
# - Worst-case time: O(n^2) (e.g., consistently poor pivots)
# - Extra space: O(1) if in_place=True; O(n) if working on a copied list.
# Notes:
# - Uses Lomuto partition with a randomized pivot to reduce the chance of wor
if not 0 <= k < len(arr):
raise ValueError("k is out of bounds for the input array.")
data = arr if in_place else list(arr) # avoid modifying caller's list by defaul
left, right = 0, len(data) - 1
while True:
if left == right:
return data[left]
# Randomized pivot selection
pivot_index = random.randint(left, right)
data[pivot_index], data[right] = data[right], data[pivot_index]
pivot = data[right]
# Lomuto partition
i = left
for j in range(left, right):
if data[j] < pivot:
data[i], data[j] = data[j], data[i]
i += 1
place pivot at its final position
data[i], data[right] = data[right], data[i]
Now pivot is at index i
powershell
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppDa
ta\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2
.Advanced Algorithms/DivideAndConquer/HW2_DivideAndConquer.py"
=== GCD demo ===
[Binary GCD] gcd_binary(0, 0) = 0
[Euclid GCD] gcd_euclid_mod(0, 0) = 0
[Comparison] Results equal? True
[Binary GCD] gcd_binary(0, 18) = 18
[Euclid GCD] gcd_euclid_mod(0, 18) = 18
[Comparison] Results equal? True
[Binary GCD] gcd_binary(18, 0) = 18
[Euclid GCD] gcd_euclid_mod(18, 0) = 18
[Comparison] Results equal? True
[Binary GCD] gcd_binary(54, 24) = 6
[Euclid GCD] gcd_euclid_mod(54, 24) = 6
[Comparison] Results equal? True
[Binary GCD] gcd_binary(270, 192) = 6
[Euclid GCD] gcd_euclid_mod(270, 192) = 6
[Binary GCD] gcd_binary(270, 192) = 6
[Euclid GCD] gcd_euclid_mod(270, 192) = 6
[Comparison] Results equal? True
[Binary GCD] gcd_binary(-54, 24) = 6
[Euclid GCD] gcd_euclid_mod(-54, 24) = 6
[Comparison] Results equal? True
[Binary GCD] gcd_binary(1234567890, 987654321) = 9
[Euclid GCD] gcd_euclid_mod(1234567890, 987654321) = 9
[Comparison] Results equal? True
[Timing Result] Binary GCD: 0.140s, Euclid GCD: 0.034s
[Timing Result] XOR sums (ignore): Binary=91452, Euclid=91452
=== QuickSelect demo ===
[QuickSelect] k= 0 -> 0 (k-th smallest)
[QuickSelect] k= 5 -> 4 (k-th smallest)
[QuickSelect] k=11 -> 9 (k-th smallest)
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
main
SantiQ (6 minutes ago)
Ln 8, Col 1
Spaces: 4
UTF-8
CRLF
Python
Python 3.12
Go Live


File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
QuickSelect.py
BinaryStein_GCD.py
HW2_DivideAndConquer.py M
DivideAndConquer > HW2_DivideAndConquer.py > ...
def quickselect(arr, k, in_place=False):
else:
left = i + 1
You, 7 minutes ago • Divide And Conquer
# ------------------------------
# DEMO
# ------------------------------
def _demo():
print("=== GCD demo ===")
pairs = [(0, 0), (0, 18), (18, 0), (54, 24), (270, 192), (-54, 24), (1234567890,
for a, b in pairs:
g1 = gcd_binary(a, b)
g2 = gcd_euclid_mod(a, b)
print(f"[Binary GCD] gcd_binary({a}, {b}) = {g1}")
print(f"[Euclid GCD] gcd_euclid_mod({a}, {b}) = {g2}")
print(f"[Comparison] Results equal? {g1 == g2}\n")
# Light micro-benchmark (not rigorous—just illustrative)
rnd = [(random.randint(1, 10**12), random.randint(1, 10**12)) for _ in range(30_
t0 = time.perf_counter()
s1 = 0
for a, b in rnd:
s1 ^= gcd_binary(a, b) # XOR to prevent Python from optimizing away
t1 = time.perf_counter()
s2 = 0
for a, b in rnd:
s2 ^= gcd_euclid_mod(a, b)
t2 = time.perf_counter()
print(f"[Timing Result] Binary GCD: {t1 - t0:.3f}s, Euclid GCD: {t2 - t1:.3f}s")
print(f"[Timing Result] XOR sums (ignore): Binary={s1}, Euclid={s2}\n")
print("\n=== QuickSelect demo ===")
arr = [9, 1, 7, 3, 5, 8, 2, 6, 4, 0, 5, 3]
for k in (0, 5, len(arr) - 1):
kth = quickselect(arr, k, in_place=False)
print(f"[QuickSelect] k={k:2d} -> {kth} (k-th smallest)")
if __name__ == "__main__":
_demo()
powershell
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/DivideAndConquer/HW2_DivideAndConquer.py"
=== GCD demo ===
[Binary GCD] gcd_binary(0, 0) = 0
[Euclid GCD] gcd_euclid_mod(0, 0) = 0
[Comparison] Results equal? True
[Binary GCD] gcd_binary(0, 18) = 18
[Euclid GCD] gcd_euclid_mod(0, 18) = 18
[Comparison] Results equal? True
[Binary GCD] gcd_binary(18, 0) = 18
[Euclid GCD] gcd_euclid_mod(18, 0) = 18
[Comparison] Results equal? True
[Binary GCD] gcd_binary(54, 24) = 6
[Euclid GCD] gcd_euclid_mod(54, 24) = 6
[Comparison] Results equal? True
[Binary GCD] gcd_binary(270, 192) = 6
[Euclid GCD] gcd_euclid_mod(270, 192) = 6
[Binary GCD] gcd_binary(270, 192) = 6
[Euclid GCD] gcd_euclid_mod(270, 192) = 6
[Comparison] Results equal? True
[Binary GCD] gcd_binary(-54, 24) = 6
[Euclid GCD] gcd_euclid_mod(-54, 24) = 6
[Comparison] Results equal? True
[Binary GCD] gcd_binary(1234567890, 987654321) = 9
[Euclid GCD] gcd_euclid_mod(1234567890, 987654321) = 9
[Comparison] Results equal? True
[Timing Result] Binary GCD: 0.140s, Euclid GCD: 0.034s
[Timing Result] XOR sums (ignore): Binary=91452, Euclid=91452
=== QuickSelect demo ===
[QuickSelect] k= 0 -> 0 (k-th smallest)
[QuickSelect] k= 5 -> 4 (k-th smallest)
[QuickSelect] k=11 -> 9 (k-th smallest)
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
main*
0 0
You, 7 minutes ago
SantiQ (7 minutes ago)
Ln 128, Col 1
Spaces: 4
UTF-8
CRLF
Python
Python 3.12
Go Live

```
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/DivideAndConquer/HW2_DivideAndConquer.py"
=== GCD demo ===
[Binary GCD] gcd_binary(0, 0) = 0
[Euclid GCD] gcd_euclid_mod(0, 0) = 0
[Comparison] Results equal? True

[Binary GCD] gcd_binary(0, 18) = 18
[Euclid GCD] gcd_euclid_mod(0, 18) = 18
[Comparison] Results equal? True

[Binary GCD] gcd_binary(18, 0) = 18
[Euclid GCD] gcd_euclid_mod(18, 0) = 18
[Comparison] Results equal? True

[Binary GCD] gcd_binary(54, 24) = 6
[Euclid GCD] gcd_euclid_mod(54, 24) = 6
[Comparison] Results equal? True

[Binary GCD] gcd_binary(270, 192) = 6
[Euclid GCD] gcd_euclid_mod(270, 192) = 6
[Binary GCD] gcd_binary(270, 192) = 6
[Euclid GCD] gcd_euclid_mod(270, 192) = 6
[Comparison] Results equal? True

[Binary GCD] gcd_binary(-54, 24) = 6
[Euclid GCD] gcd_euclid_mod(-54, 24) = 6
[Comparison] Results equal? True

[Binary GCD] gcd_binary(1234567890, 987654321) = 9
[Euclid GCD] gcd_euclid_mod(1234567890, 987654321) = 9
[Comparison] Results equal? True

[Timing Result] Binary GCD: 0.140s, Euclid GCD: 0.034s
[Timing Result] XOR sums (ignore): Binary=91452, Euclid=91452


=== QuickSelect demo ===
[QuickSelect] k= 0 -> 0 (k-th smallest)
[QuickSelect] k= 5 -> 4 (k-th smallest)
[QuickSelect] k=11 -> 9 (k-th smallest)
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
```

https://colab.research.google.com/drive/1fnWmrGHNdOmKIGpCEMMOsXr3_CwRIjdb?usp=sharing