9101 Assignment 2
Haojin Guo
z5216214

Q3.

3. A city is attacked by N monsters and is defended by a single hero with initial strength of S units. To kill a monster i the hero must dissipate

a_i units of strength; if monster i is killed successfully the hero gains g_i units of strength. Thus, if the hero had strength $c \geq a_i$ before tackling monster i he can kill the monster and he will end up with $c - a_i + g_i$ units of strength. Note that for some monsters i you might have $g_i \geq a_i$ but for some other j you might have $a_j > g_j$. You are given S and for each i you are given a_i and g_i. Design an algorithm which determines in which order the hero can fight the monsters if he is to kill them all (if there is such an ordering). In case there is no such ordering the algorithm should output "no such ordering". Assume all values are positive integers.

Solution,

1) Traversing the list of Monsters, which including the parameters a_i, g_i. Then divide the N monsters into two lists, CASE1 and CASE2, where,
   CASE1: a_i <= g_i
   (e.g. CASE1 = [(a_0, g_0), (a_1, g_1), …])
   CASE2: a_i > g_i

The complexity in step 1) is O(n).

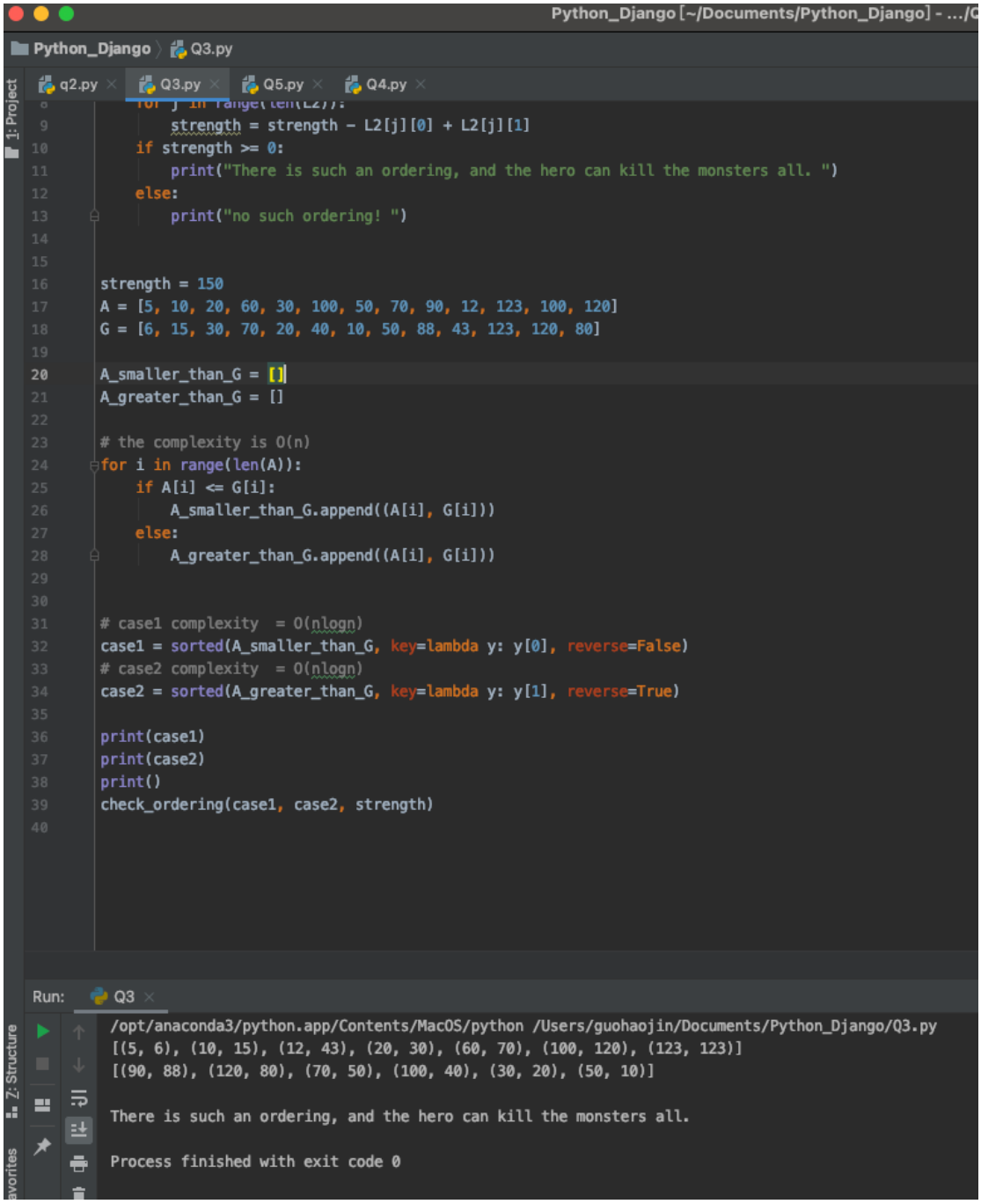
2) Merge sorting the CASE1 in ascending order based on the value of a_i, which means when a_i <= g_i, the hero starts with the monster with the smallest a_i value to fight the monsters.

The complexity is O(nlogn)

3) Merge sorting the CASE2 in descending order based on the value of g_i, which means when a_i > g_i, the hero starts with the monster with the largest g_i value to fight the monsters.

The complexity is O(nlogn)

Therefore, the total cost is O(n) + O(nlogn) + + O(nlogn) = O(n) + O(nlogn).


Python_Django [~/Documents/Python_Django] - .../Q
Python_Django
Q3.py
q2.py
Q3.py
Q5.py
Q4.py
1: Project
strength = strength - L2[j][0] + L2[j][1]
if strength >= 0:
print("There is such an ordering, and the hero can kill the monsters all. ")
else:
print("no such ordering! ")
strength = 150
A = [5, 10, 20, 60, 30, 100, 50, 70, 90, 12, 123, 100, 120]
G = [6, 15, 30, 70, 20, 40, 10, 50, 88, 43, 123, 120, 80]
A_smaller_than_G = []
A_greater_than_G = []
# the complexity is O(n)
for i in range(len(A)):
if A[i] <= G[i]:
A_smaller_than_G.append((A[i], G[i]))
else:
A_greater_than_G.append((A[i], G[i]))
# case1 complexity = O(nlogn)
case1 = sorted(A_smaller_than_G, key=lambda y: y[0], reverse=False)
# case2 complexity = O(nlogn)
case2 = sorted(A_greater_than_G, key=lambda y: y[1], reverse=True)
print(case1)
print(case2)
print()
check_ordering(case1, case2, strength)
Run:
Q3
/opt/anaconda3/python.app/Contents/MacOS/python /Users/guohaojin/Documents/Python_Django/Q3.py
[(5, 6), (10, 15), (12, 43), (20, 30), (60, 70), (100, 120), (123, 123)]
[(90, 88), (120, 80), (70, 50), (100, 40), (30, 20), (50, 10)]
There is such an ordering, and the hero can kill the monsters all.
Process finished with exit code 0
Z: Structure
avorites