Graphical user interface, text, application, email

Description automatically generated

1. Implement the solution of this problem using dynamic Programming. Name your function **max\_independent\_set(nums)**. Name your file **MaxSet.py**

def max\_independent\_set(nums):

n = len(nums)

if n == 0:

return []

cache = [0] \* (n + 1)

cache[0] = 0

cache[1] = max(nums[0], 0)

for i in range(2, n + 1):

cache[i] = max(cache[i - 1], cache[i - 2] + nums[i - 1])

if all(num < 0 for num in nums):

return []

solution = []

i = n

while i >= 1:

if cache[i] == cache[i - 1]:

i -= 1

else:

solution.append(nums[i - 1])

i -= 2

return list(reversed(solution))

1. What is the time complexity of your implementation?

**O(n)**

The time complexity of this implementation is O(n) where n is the length of the input list nums. This is because we are using a single loop to iterate through the input list, a single loop to fill the cache array with maximum sums, and a single loop to construct the solution by iterating through the cache array from the end. These operations have a linear time complexity, thus the overall time complexity is O(n).

Graphical user interface, text, application, email

Description automatically generated

def powerset\_helper(pointer, choices\_made, inputSet, result):

if pointer < 0:

result.append(choices\_made.copy())

return

choices\_made.append(inputSet[pointer])

powerset\_helper(pointer - 1, choices\_made, inputSet, result)

choices\_made.pop()

powerset\_helper(pointer - 1, choices\_made, inputSet, result)

def powerset(inputSet):

result = []

powerset\_helper(len(inputSet) - 1, [], inputSet, result)

return result

Text

Description automatically generated with low confidence

**O(n2)**

This is because for each element in the input set, the function either includes it or excludes it in the subset. For a set of n elements, this results in 2^n possible subsets. Therefore, the function needs to perform 2^n operations to generate all possible subsets.

The backtracking approach used in this implementation does not generate all 2^n subsets explicitly, but rather generates each subset by adding or removing elements as it backtracks. However, the worst-case time complexity is still O(2^n) because each element can be included or excluded in each subset, leading to a total of 2^n possible subsets.