Rebecca Lashua
CS 325
lashuar@oregonstate.edu
HW 4

1.

a. The recurrence formula:

f(x,i) = { } if n= 0
{x_0} if n = 1
max(x_i + f(i-2), f(i-1))** if n >= 2

**Here max(setA, setB) returns the set whose elements sum to the larger value

b. Pseudocode:

```
max_independent_set(nums):
    output = []
    if len(nums) = 0: # no sum
        return output
    if len(nums) = 1: # the 1 val is sum
        output.append(nums[0])
        return output
    dp = [0] * len(nums)
    dp[0] = nums[0]
    dp[1] = max(nums[0], nums[1])
    n = len(nums)
    for i in range(2, n):
        val_1 = (nums[i] + dp[i-2])
        val_2 = dp[i-1]
        dp[i] = max(val_1, val_2)

    max_sum = dp[-1]
    for j in reversed(nums):
        if nums[j] > 0:
            new_sum = max_sum - nums[j]
            if new_sum = 0:
                output.append(nums[j])
                return output
            if new_sum in dp:
                output.append(nums[j])
                max_sum = new_sum
    return output
```

c. See MaxSet.py

```python
def max_independent_set(nums):
    output = []
    if len(nums) == 0: # no sum
        return output
    if len(nums) == 1: # the 1 val is sum
        output.append(nums[0])
        return output
    dp = [0] * len(nums)
    dp[0] = nums[0]
    dp[1] = max(nums[0], nums[1])
    n = len(nums)
    for i in range(2, n):
        val_1 = (nums[i] + dp[i-2])
        val_2 = dp[i-1]
        dp[i] = max(val_1, val_2)

    max_sum = dp[-1]
    for j in range(len(nums)-1,-1,-1):
        if nums[j] > 0:
            new_sum = max_sum - nums[j]
            if new_sum == 0:
                output.append(nums[j])
                return output
            if new_sum in dp:
                output.append(nums[j])
                max_sum = new_sum
    return output
```

d. The time complexity is O(n)

2.

a.

```python
from copy import deepcopy

def powerset_helper(pointer, choices_made,input, result):
    if pointer < 0:
        result.append(deepcopy(choices_made))
        return

    choices_made.append(input[pointer])
    powerset_helper(pointer-1, choices_made, input, result)

    choices_made.pop()
    powerset_helper(pointer - 1, choices_made, input, result)


def powerset(input):
    result = []
    powerset_helper(len(input)-1, [], input, result)
    return result
```

b. The time complexity is O(2^n), because each call makes two calls and the recursion tree has a height of n. (This assumes that the deepcopy function is a constant time operation.

If the deepcopy is not considered constant time, but has a runtime of O(n), then the overall time complexity would be O(n*2^n))

Debriefing (required!): ----------

Report:

1. Approximately how many hours did you spend on this assignment?
2. Would you rate it as easy, moderate, or difficult?
3. How deeply do you feel you understand the material it covers (0%–100%)?
4. Any other comments?

1. 13 hrs
2. Medium
3. 60%
4. N/A