Thank you for taking the Week 10: Assignment 10.

Week 10: Assignment 10

Your last recorded submission was on 2023-04-05, 23:31 IST Due date: 2023-04-05, 23:59 IST.

1) Consider the Problem **Frog 1** discussed in the lecture. What would be the minimum possible total cost for a given sequence of the height of stones?

15, 25, 12, 35, 26, 14, 28, 35, 30, 45, 37, 48

39

1 point

2) Consider the Problem **Dice with combinations** discussed in the lecture. The number of ways to construct sum **10** by throwing a dice one or more times is __.

492

1 point

3) Which of the following statement is true about **memoization** in dynamic programming? ***1 point***

- (●) It is a way to store the solutions of sub-problems to reuse
- ( ) It is a way to divide a problem into smaller sub-problems
- ( ) It is a way to make locally optimal choices
- ( ) It is a way to backtrack from a wrong decision

4) Consider the following 5 items (one unit for each) with their weights and value.

Item No.	Weight(W)	Value(V)
1	7	28
2	11	55
3	2	34
4	4	32
5	9	63

The task is to pick a subset of these items such that their total weight should be lesser or equal to 13 (maximum weight capacity C <= 13) and their total value is maximized. Consider that each item has only one unit and it can not be split.

V_{opt} = The total value of items picked by an optimal algorithm.

V_{greedy} = The total value of items picked by one greedy approach that sorts the item by `Value(V)` to `Weight(W)` ratio in descending order and picks them greedily starting from the first item in the ordered list (pick the item if the total weight of that item is less than or equal to the remaining capacity, otherwise, skip that item).

The value of $V_{opt} - V_{greedy}$ is ___.

3

1 point

5) Consider the following algorithm using a dynamic programming approach to find the optimal solution to **Question 4**: ***1 point***

```
#Weight = [w1,w2,...,wn] start from index 0
#Value = [v1,v2,...,vn] start from index 0
#C = Maximum capacity

Algoritm:-
1. Define the 2D array dp[n+1][C+1], where n is the number of items and C is the maximum capacity.
2. Initialize the base cases: dp[0][w] = 0 for all w and dp[i][0] = 0 for all i.
3. For i = 1 to n, and w = 1 to C, compute dp[i][w] using the following recurrences condition:

   ##recurrences condition##

4. return dp[n][C].
```

Week 10 Programming Assignment Q2 (/noc23_cs30/progassignment?name=227)

Week 11 ()

Download Videos ()

Live Sessions ()

Transcripts ()

Which of the following is correct **recurrences condition** to complete the given algorithm, so it can return the correct solution?

```
if Weight[i-1] > w:
    dp[i][w] = max(dp[i-1][w], dp[i-1][w-Weight[i-1]] + Value[i-1])
else:
    dp[i][w] = dp[i-1][w]
```

```
if Weight[i-1] > w:
    dp[i][w] = dp[i-1][w]
else:
    dp[i][w] = max(dp[i-1][w], dp[i-1][w-Weight[i-1]] + Value[i-1])
```

```
dp[i][w] = max(dp[i-1][w], dp[i-1][w-Weight[i-1]] + Value[i-1])
```

```
if Weight[i-1] > w:
    dp[i][w] = dp[i][w-1]
else:
    dp[i][w] = max(dp[i][w-1], dp[i-1][w-Weight[i-1]] + Value[i-1])
```

Common data for question 6 & 7

The subset sum problem is defined as follows. Given a list `L` of `n` non-negative integers and a value `k`, determine if there is a subset of elements from the list `L` whose sum is equal to `k`. Consider the following solution code, where `T` is a 2-dimensional Boolean array, with `n + 1` rows and `k + 1` columns. `T[i][j], 1 ≤ i ≤ n, 1 ≤ j ≤ k` is `True`, if and only if there is a subset of `{a1, a2, ..., ai}` whose sum is equal to `j`.

```
def subsetSum(L, k):
    n = len(L)
    T = [[False for x in range(k + 1)] for y in range(n + 1)]
    for i in range(n + 1):
        T[i][0] = True
    for i in range(1, n + 1):
        for j in range(1, k + 1):
            if L[i - 1] > j:
                T[i][j] = T[i - 1][j]
            else:
                T[i][j] = T[i - 1][j] or T[i - 1][j - L[i - 1]]
    return ___
```

Assessment submitted.
X

6) In the given code, which entry of `T` should be returned? If `True`, it implies that there is a subset of elements whose sum is equal to `k`. ***1 point***

- `T[1][k]`
- (selected) `T[n][k]`
- `T[k][1]`
- `T[0][k]`

7) What is the time complexity of function `subsetSum()`? ***1 point***

- $O(n^2)$
- $O(n\ log\ n)$
- $O(n+k)$
- (selected) $O(nk)$

8) The number `n` is to be created by adding the elements (one element can be used more than one time) of a list of positive integers `L` of length `m`. ***1 point***

```
def nsum(n, L):
    if n == 0:
        return []
    if n < 0:
        return None
    for i in L:
        x = nsum(n-i, L)
        if x != None:
            return x+[i]
    return None
```

What will be the asymptotic complexity of the above recursive function without and with memoization respectively?

- (selected) $O(m^n), O(mn)$
- $O(2^n), O(n^2)$
- $O(n^m), O(n)$
- $O(2^{m+n}), O(m+n)$

Assessment submitted.
X

You may submit any number of times before the due date. The final submission will be considered for grading.

Submit Answers