**Roll No…………….. Total No. of Pages:……**

**ST-4 (SET-II)**

**6th SEMESTER 2023-24**

**CS192- Advanced Data Structures**

**Time allowed: 90 Minutes Max. Marks: 40**

**General Instructions:**

* **Follow the instructions given in each section.**
* **Make sure that you attempt the questions in order.**

**SECTION-A (10\*1 mark=10 marks)**

***(All questions are compulsory)***

1. What is the time complexity of a typical 1-Dimensional DP problem with n states and each state taking O(1) time to compute?
   1. **O(n)**
   2. O(log n)
   3. O(n^2)
   4. O(2^n)
2. Which step is typically performed after populating the DP table in a bottom-up approach?
   1. Initializing the base cases
   2. **Calculating the final result**
   3. Using recursion to fill in missing values
   4. Resetting the table to its initial state
3. Which Knapsack problem has a time complexity of O(n log n) when solved optimally?
   1. 0/1 Knapsack
   2. **Fractional Knapsack**
   3. Bounded Knapsack
   4. Unbounded Knapsack
4. What is the time complexity of the Fibonacci sequence calculation using dynamic programming?
   1. O(1)
   2. **O(n)**
   3. O(log n)
   4. O(n^2)
5. In multidimensional dynamic programming, what does the transition function define?
   1. The initial state
   2. The final state
   3. **How to move from one state to another**
   4. The number of dimensions
6. Which of the following is NOT a typical application of Dynamic Programming on Trees?
   1. Finding the maximum independent set of nodes in a tree
   2. Calculating the total number of nodes in a tree
   3. Finding the minimum spanning tree of a graph
   4. **Sorting the nodes of a tree in ascending order**
7. In the greedy algorithm for the coin change problem, which coin is selected at each step?
   1. **The largest available coin**
   2. The smallest available coin
   3. A random coin
   4. The rarest coin
8. What is the result of 10 & 3 in binary?
   1. **2**
   2. 3
   3. 0
   4. 1
9. What is the primary purpose of the segmented sieve algorithm?
   1. **Finding prime numbers in a given range**
   2. Finding prime factors of a number
   3. Generating random prime numbers
   4. Finding prime numbers up to 'n'
10. The number of 4 digit numbers having their digits in non-decreasing order (from left to right) constructed by using the digits belonging to the set {1, 2, 3} is \_\_\_\_.
    1. 12
    2. 13
    3. 14
    4. **15**

**SECTION-B (5\*2 mark=10 marks)**

***(All questions are compulsory)***

11) Four matrices M1, M2, M3 and M4 of dimensions pxq, qxr, rxs and sxt respectively can be multiplied is several ways with different number of total scalar multiplications. For example, when multiplied as ((M1 X M2) X (M3 X M4)), the total number of multiplications is pqr + rst + prt. When multiplied as (((M1 X M2) X M3) X M4), the total number of scalar multiplications is pqr + prs + pst. If p = 10, q = 100, r = 20, s = 5 and t = 80, then the number of scalar multiplications needed is

a) 248000

b) 44000

**c) 19000**

d) 25000

12) In the 0-1 Knapsack Problem, if an item's weight is greater than the remaining capacity of the knapsack, what action is typically taken?

**a) Ignore the item and move to the next one**

b) Remove the least valuable item from the knapsack to make space

c) Add the fractional part of the item that can fit into the knapsack

d) Remove the most valuable item from the knapsack to make space

13) Suppose the letters a, b, c, d, e, f have probabilities 1/2, 1/4, 1/8, 1/16, 1/32, 1/32 respectively. Which of the following is the Huffman code for the letter a, b, c, d, e, f?

**a) 0, 10, 110, 1110, 11110, 11111**

b) 11, 10, 011, 010, 001, 000

c) 11, 10, 01, 001, 0001, 0000

d) 110, 100, 010, 000, 001, 111

14) What is the output of the following code snippet?

#include <iostream>

using namespace std;

void fun(int& num, int k) { num &= (~(1 << k)); }

int main()

{

int num = 7;

int k = 1;

fun(num, k);

cout << num << endl;

return 0;

}

a) It will unset the all bits of num

b) It will clear all the bits of bits

**c) It will unset the kth bit of num**

d) None

15) What is the output of the following C++ code?

#include <iostream>

#include <vector>

using namespace std;

int main() {

vector<int> weights = {1, 2, 3};

vector<int> values = {6, 10, 12};

int capacity = 5;

int n = weights.size();

vector<vector<int>> dp(n + 1, vector<int>(capacity + 1, 0));

for (int i = 1; i <= n; i++) {

for (int w = 1; w <= capacity; w++) {

if (weights[i - 1] <= w) {

dp[i][w] = max(values[i - 1] + dp[i - 1][w - weights[i - 1]], dp[i - 1][w]);

} else {

dp[i][w] = dp[i - 1][w];

}

}

}

cout << "Maximum value: " << dp[n][capacity] << endl;

return 0;

}

a) Maximum value: 6

b) Maximum value: 10

c) Maximum value: 12

**d) Maximum value: 22**

**SECTION-C(Coding Question) (2x5 marks=5 marks)**

Q16) Write a C++ Program to Check if binary representation of a number is palindrome

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | x=0 | x=10 | x=19 |
| **Output** | 1 | 0 | 0 |

Solution :

**#include<iostream>**

**using namespace std;**

**// This function returns true if k'th bit in x is set (or 1). For example if x (0010) is 2 and k is 2, then it returns true**

**bool isKthBitSet(unsigned int x, unsigned int k)**

**{**

**return (x & (1 << (k - 1))) ? true : false;**

**}**

**// This function returns true if binary representation of x is palindrome.**

**// For example (1000...001) is palindrome**

**bool isPalindrome(unsigned int x)**

**{**

**int l = 1; // Initialize left position**

**int r = sizeof(unsigned int) \* 8; // initialize right position**

**// One by one compare bits**

**while (l < r)**

**{**

**if (isKthBitSet(x, l) != isKthBitSet(x, r))**

**return false;**

**l++; r--;**

**}**

**return true;**

**}**

**int main()**

**{**

**unsigned int x = 0;**

**cout << isPalindrome(x) << endl;**

**x = 10;**

**cout << isPalindrome(x) << endl;**

**return 0;**

**}**

Q17) A Derangement is a permutation of n elements, such that no element appears in its original position. For example, a derangement of {0, 1, 2, 3} is {2, 3, 1, 0}.

Given a number n, find the total number of Derangements of a set of n elements.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | n = 2 | n = 3 | n = 4 |
| **Output** | 1 | 2 | 9 |

Solution :

**// A Dynamic programming based C++ program to count derangements**

**#include <bits/stdc++.h>**

**using namespace std;**

**int countDer(int n)**

**{**

**// Create an array to store counts for subproblems**

**int der[n + 1] = {0};**

**// Base cases**

**der[1] = 0;**

**der[2] = 1;**

**// Fill der[0..n] in bottom up manner using above recursive formula**

**for (int i = 3; i <= n; ++i)**

**der[i] = (i - 1) \* (der[i - 1] +**

**der[i - 2]);**

**// Return result for n**

**return der[n];**

**}**

**int main()**

**{**

**int n = 4;**

**cout << "Count of Derangements is "**

**<< countDer(n);**

**return 0;**

**}**

**SECTION-D (Coding Question)(1x10 mark=10 mark)**

Q18) In a candy store, there are N different types of candies available and the prices of all the N different types of candies are provided. There is also an attractive offer by the candy store. We can buy a single candy from the store and get at most K other candies (all are different types) for free.

Find the minimum amount of money we have to spend to buy all the N different candies.

Find the maximum amount of money we have to spend to buy all the N different candies.

In both cases, we must utilize the offer and get the maximum possible candies back. If k or more candies are available, we must take k candies for every candy purchase. If less than k candies are available, we must take all candies for a candy purchase.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | price[] = {3, 2, 1, 4}  k = 2 | price[] = {9, 8, 5, 4}  k = 3 | price[] = {8, 7, 1, 4}  k = 2 |
| **Output** | Min = 3, Max = 7 | Min = 4, Max = 9 | Min = 5, Max = 15 |

Solution :

**// C++ implementation to find the minimum and maximum amount**

**#include <bits/stdc++.h>**

**using namespace std;**

**// Function to find the minimum amount to buy all candies**

**int findMinimum(int arr[], int n, int k)**

**{**

**int res = 0;**

**for (int i = 0; i < n; i++) {**

**// Buy current candy**

**res += arr[i];**

**// And take k candies for free from the last**

**n = n - k;**

**}**

**return res;**

**}**

**// Function to find the maximum amount to buy all candies**

**int findMaximum(int arr[], int n, int k)**

**{**

**int res = 0, index = 0;**

**for (int i = n - 1; i >= index; i--)**

**{**

**// Buy candy with maximum amount**

**res += arr[i];**

**// And get k candies for free from**

**// the starting**

**index += k;**

**}**

**return res;**

**}**

**int main()**

**{**

**int arr[] = { 3, 2, 1, 4 };**

**int n = sizeof(arr) / sizeof(arr[0]);**

**int k = 2;**

**sort(arr, arr + n);**

**// Function call**

**cout << findMinimum(arr, n, k) << " "**

**<< findMaximum(arr, n, k) << endl;**

**return 0;**

**}**