**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) Evaluate the value of an arithmetic expression in Reverse Polish Notation (postfix notation).

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | ["2", "1", "+", "3", "\*"] | ["6", "3", "2", "4", "+","-","\*"] | ["6", "3","/"] |
| **Output** | 9 | -18 | 2 |

Solution :

**#include <iostream>**

**#include <stack>**

**#include <vector>**

**using namespace std;**

**// Function to evaluate Reverse Polish Notation (RPN) expression**

**int evalRPN(vector<string>& tokens) {**

**stack<int> st; // Initialize a stack to hold operands**

**// Iterate through each token in the expression**

**for (string token : tokens) {**

**if (token == "+" || token == "-" || token == "\*" || token == "/") {**

**// If the token is an operator, pop the top two elements from the stack**

**int num2 = st.top(); st.pop();**

**int num1 = st.top(); st.pop();**

**// Perform the operation based on the operator**

**if (token == "+") st.push(num1 + num2);**

**else if (token == "-") st.push(num1 - num2);**

**else if (token == "\*") st.push(num1 \* num2);**

**else if (token == "/") st.push(num1 / num2);**

**} else {**

**// If the token is an operand, convert it to an integer and push it onto the stack**

**st.push(stoi(token));**

**}**

**}**

**// The final result will be left on the top of the stack**

**return st.top();**

**}**

**int main() {**

**vector<string> tokens = {"6", "3", "2", "4", "+","-","\*"};**

**cout << "Result: " << evalRPN(tokens) << endl; // Evaluate the RPN expression and display the result**

**return 0;**

**}**

Q17) Write a CPP program to print the top view of binary tree. Top view of a binary tree is the set of nodes visible when the tree is viewed from the top.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | 10  / \  20 30  / \ / \  40 60 90 100 | 1  / \  2 3  \  4  \  5  \  6 | 10  / \  20 30 |
| **Output** | 40 20 10 30 100 | 2 1 3 6 | 20 10 30 |

Solution :

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

**using namespace std;**

**// Structure to represent a tree node**

**struct Node {**

**int data;**

**Node\* left;**

**Node\* right;**

**Node(int val) {**

**data = val;**

**left = right = NULL;**

**}**

**};**

**// Function to print the top view of a binary tree**

**void topView(Node\* root) {**

**if (!root)**

**return;**

**map<int, int> verticalMap; // Map to store vertical level and node data**

**queue<pair<Node\*, int>> q; // Queue for BFS traversal**

**q.push({root, 0});**

**while (!q.empty()) {**

**Node\* node = q.front().first;**

**int verticalLevel = q.front().second;**

**q.pop();**

**// Insert the node's data if not present in the map**

**if (verticalMap.find(verticalLevel) == verticalMap.end()) {**

**verticalMap[verticalLevel] = node->data;**

**}**

**// Push left child with decreased vertical level**

**if (node->left) {**

**q.push({node->left, verticalLevel - 1});**

**}**

**// Push right child with increased vertical level**

**if (node->right) {**

**q.push({node->right, verticalLevel + 1});**

**}**

**}**

**// Print the nodes in the top view**

**for (const auto& entry : verticalMap) {**

**cout << entry.second << " ";**

**}**

**}**

**int main() {**

**Node\* root = new Node(10);**

**root->left = new Node(20);**

**root->right = new Node(30);**

**root->left->right = new Node(60);**

**root->left->left = new Node(40);**

**root->right->left = new Node(90);**

**root->right->right = new Node(100);**

**cout << "Top view of the binary tree: ";**

**topView(root);**

**return 0;**

**}**

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

Q18) Write a C++ program to insert new element to MAX-Heap.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | heap={ 10, 5, 3, 2, 4 }  key=15 | heap={ 12,6,4,3,5 }  key=21 | heap={ 30,15,19 }  key=2 |
| **Output** | 15 5 10 2 4 3 | 21 6 12 3 5 4 | 30 15 19 2 |

Solution :

**#include <iostream>**

**using namespace std;**

**#define MAX 1000 // Max size of Heap**

**// Function to heapify ith node in a Heap**

**// of size n following a Bottom-up approach**

**void heapify(int arr[], int n, int i)**

**{**

**// Find parent**

**int parent = (i - 1) / 2;**

**if (arr[parent] > 0) {**

**// For Max-Heap**

**// If current node is greater than its parent**

**// Swap both of them and call heapify again**

**// for the parent**

**if (arr[i] > arr[parent]) {**

**swap(arr[i], arr[parent]);**

**// Recursively heapify the parent node**

**heapify(arr, n, parent);**

**}**

**}**

**}**

**// Function to insert a new node to the Heap**

**void insertNode(int arr[], int& n, int Key)**

**{**

**// Increase the size of Heap by 1**

**n = n + 1;**

**// Insert the element at end of Heap**

**arr[n - 1] = Key;**

**// Heapify the new node following a**

**// Bottom-up approach**

**heapify(arr, n, n - 1);**

**}**

**// A utility function to print array of size n**

**void printArray(int arr[], int n)**

**{**

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

**cout << arr[i] << " ";**

**cout << "\n";**

**}**

**int main()**

**{**

**// Array representation of Max-Heap**

**// 10**

**// / \**

**// 5 3**

**// / \**

**// 2 4**

**int arr[MAX] = { 10, 5, 3, 2, 4 };**

**int n = 5;**

**int key = 15;**

**insertNode(arr, n, key);**

**printArray(arr, n);**

**// Final Heap will be:**

**// 15**

**// / \**

**// 5 10**

**// / \ /**

**// 2 4 3**

**return 0;**

**}**