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

**ST-4 (SET-IV)**

**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 Bottom-Up DP for 1-Dimensional problems with n states?

1. O(1)
2. O(log n)
3. **O(n)**
4. O(n^2)

2) In grid DP, what is typically represented along the horizontal axis of the DP table?

1. Subproblem size
2. Subproblem index
3. Grid row
4. **Grid column**

3) In dynamic programming for the 0/1 Knapsack problem, what does the state "dp[i][w]" represent?

1. **The maximum value achievable by selecting items from the first "i" items with a maximum weight limit of "w."**
2. The total weight of items selected from the first "i" items.
3. The number of items selected from the first "i" items.
4. The maximum weight limit achievable by selecting items from the first "i" items.

4) What does the "Optimal Substructure" property mean in dynamic programming?

1. It describes problems that have a single solution.
2. **It states that the global optimal solution can be constructed from optimal solutions of subproblems.**
3. It indicates problems with no overlapping subproblems.
4. It refers to the order in which subproblems are solved.

5) In multidimensional dynamic programming, what is the role of the transition table?

1. **Storing subproblem solutions**
2. Defining the problem statement
3. Creating recursive functions
4. Identifying base cases

6) What is the primary disadvantage of using a recursive (top-down) approach in multidimensional dynamic programming?

1. It is more challenging to implement.
2. **It may result in exponential time complexity.**
3. It requires less memory.
4. It is less accurate.

7) Which of the following problems can be efficiently solved using Dynamic Programming on Trees?

1. Determining if a graph is connected
2. Finding the shortest path between two nodes in a graph
3. Finding the minimum spanning tree of a graph
4. **Calculating the diameter of a tree**

8) Which of the following problems is NOT solved using a greedy algorithm?

1. Fractional Knapsack Problem
2. Huffman coding
3. Shortest Path Problem
4. **Sorting an array**

9) How do you check if the rightmost (least significant) bit is set to 1 in an integer x?

1. **(x & 1) == 1**
2. (x & 1) == 0
3. (x | 1) == 1
4. (x ^ 1) == 1

10) Which theorem is used to calculate modular inverses when 'p' is a prime number?

1. Euler's Totient Theorem
2. **Fermat's Little Theorem**
3. Chinese Remainder Theorem
4. Extended Euclidean Theorem

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

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

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

#include <iostream>

using namespace std;

int main() {

int dp[7] = {0};

dp[1] = 1;

for (int i = 2; i <= 6; i++) {

dp[i] = i;

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

dp[i] = min(dp[i], dp[j] + dp[i - j]);

}

}

cout << "Minimum number of coins needed: " << dp[6] << endl;

return 0;

}

a) Minimum number of coins needed: 1

b) Minimum number of coins needed: 2

c) Minimum number of coins needed: 3

**d) Minimum number of coins needed: 6**

12) The subset-sum problem is defined as follows. Given a set of n positive integers, S = {a1 ,a2 ,a3 ,…,an} and positive integer W, is there a subset of S whose elements sum to W? A dynamic program for solving this problem uses a 2-dimensional Boolean array X, with n rows and W+1 columns. X[i, j],1 <= i <= n, 0 <= j <= W, is TRUE if and only if there is a subset of {a1 ,a2 ,...,ai} whose elements sum to j. Which of the following is valid for 2 <= i <= n and ai <= j <= W?

a) X[i, j] = X[i - 1, j] ∨ X[i, j -ai]

**b) X[i, j] = X[i - 1, j] ∨ X[i - 1, j - ai]**

c) X[i, j] = X[i - 1, j] ∧ X[i, j - ai]

d) X[i, j] = X[i - 1, j] ∧ X[i -1, j - ai]

13) Consider a sequence F00 defined as : F00(0) = 1, F00(1) = 1 F00(n) = 10 ∗ F00(n – 1) + 100 F00(n – 2) for n ≥ 2 Then what shall be the set of values of the sequence F00 ?

**a) (1, 110, 1200)**

b) (1, 110, 600, 1200)

c) (1, 2, 55, 110, 600, 1200)

d) (1, 55, 110, 600, 1200)

14) Consider a graph G=(V, E), where V = { v1,v2,…,v100 }, E={ (vi, vj) ∣ 1≤ i < j ≤ 100} and weight of the edge (vi, vj) is ∣i–j∣. The weight of minimum spanning tree of G is \_\_\_\_\_\_\_\_. Note - This question was Numerical Type.

**a) 99**

b) 100

c) 98

d) 101

15) What does the following C expression do?

x = x & (x-1)

a) Sets all bits as 1

b) Makes x equals to 0

**c) Turns of the rightmost set bit**

d) Turns of the leftmost set bit

**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;**

**}**