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

**ST-4 (SET-III)**

**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 key characteristic of a "Counting" type 1-Dimensional DP problem?
2. Minimizing a value
3. Maximizing a value
4. **Counting the number of valid solutions**
5. Finding a specific solution

2) What is the primary goal of dynamic programming in grid-based problems?

1. To create complex algorithms
2. To find the optimal solution to any problem
3. To reduce the time complexity of solving problems
4. **To solve problems by breaking them into smaller, overlapping subproblems**

3) In the Unbounded Knapsack problem, what does the term "unbounded" signify?

1. There are no constraints on item selection.
2. Each item can be selected at most once.
3. **Each item can be selected multiple times.**
4. The knapsack capacity is unlimited.

4) The 0/1 Knapsack problem is an example of which dynamic programming pattern?

1. Top-down
2. **Bottom-up**
3. Memoization
4. Greedy

5) What is the primary advantage of using the bottom-up approach in multidimensional dynamic programming?

1. It is easier to implement.
2. **It reduces space complexity.**
3. It avoids overlapping subproblems.
4. It ensures optimal solutions.

6) In Dynamic Programming on Trees, what is the main benefit of solving subproblems efficiently?

1. It reduces the number of nodes in the tree.
2. **It speeds up the execution of the program.**
3. It avoids using memoization.
4. It allows for more complex problems to be solved.

7) In Huffman coding, what is the main idea behind assigning shorter codes to frequent characters?

1. To improve encryption
2. **To save memory space**
3. To speed up decoding
4. To improve sorting

8) What is the binary representation of 10 in two's complement?

1. **1111 0110**
2. 1010 0101
3. 0011 1010
4. 1101 1011

9) What is the primary purpose of the extended Euclidean algorithm?

1. Finding prime numbers
2. **Solving Diophantine equations**
3. Calculating modular exponentiation
4. Computing Euler's totient function

10) Mala has a colouring book in which each English letter is drawn two times. She wants to paint each of these 52 prints with one of k colours, such that the colour-pairs used to colour any two letters are different. Both prints of a letter can also be coloured with the same colour. What is the minimum value of k that satisfies this requirement ?

1. 9
2. 8
3. **7**
4. 6

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

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

11) A sub-sequence of a given sequence is just the given sequence with some elements (possibly none or all) left out. We are given two sequences X[m] and Y[n] of lengths m and n respectively, with indexes of X and Y starting from 0. We wish to find the length of the longest common sub-sequence(LCS) of X[m] and Y[n] as l(m,n), where an incomplete recursive definition for the function l(i,j) to compute the length of The LCS of X[m] and Y[n] is given below:

l(i,j) = 0, if either i=0 or j=0

= expr1, if i,j > 0 and X[i-1] = Y[j-1]

= expr2, if i,j > 0 and X[i-1] != Y[j-1]

a) expr1 ≡ l(i-1, j) + 1

b) expr1 ≡ l(i, j-1)

**c) expr2 ≡ max(l(i-1, j), l(i, j-1))**

d) expr2 ≡ max(l(i-1,j-1),l(i,j))

12) Consider two strings A = "qpqrr" and B = "pqprqrp". Let x be the length of the longest common subsequence (not necessarily contiguous) between A and B and let y be the number of such longest common subsequences between A and B. Then x + 10y = \_\_\_.

a) 33

b) 23

c) 43

**d) 34**

13) Six files F1, F2, F3, F4, F5 and F6 have 100, 200, 50, 80, 120, 150 records respectively. In what order should they be stored so as to optimize act. Assume each file is accessed with the same frequency

**a) F3, F4, F1, F5, F6, F2**

b) F2, F6, F5, F1, F4, F3

c) F1, F2, F3, F4, F5, F6

d) Ordering is immaterial as all files are accessed with the same frequency

14) Consider the following code snippet for checking whether a number is power of 2 or not.

/\* Incorrect function to check if x is power of 2\*/

bool isPowerOfTwo (unsigned int x)

{

return (!(x&(x-1)));

}

What is wrong with above function?

a) It does reverse of what is required

b) It works perfectly fine for all values of x.

**c) It does not work for x = 0**

d) It does not work for x = 1

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

#include <iostream>

using namespace std;

int main() {

int arr[] = {1, 3, 2, 4, 6};

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

int target = 8;

int dp[9] = {0};

dp[0] = 1;

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

for (int j = arr[i]; j <= target; j++) {

dp[j] += dp[j - arr[i]];

}

}

cout << "Number of ways to reach the target: " << dp[target] << endl;

return 0;

}

a) Number of ways to reach the target: 18

b) Number of ways to reach the target: 21

**c) Number of ways to reach the target: 17**

d) Number of ways to reach the target: 4

**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;**

**}**