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

**ST-4 (SET-I)**

**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. Dynamic Programming is primarily used to solve problems that exhibit which property?
   1. **Overlapping Subproblems**
   2. Non-deterministic behavior
   3. Regular structure
   4. Static nature
2. What is the main disadvantage of using a recursive (top-down) approach in grid DP?
   1. Slower execution time
   2. **Greater memory usage**
   3. Difficulty in handling base cases
   4. Inability to handle grid-based problems
3. Which Knapsack problem involves selecting each item at most once and has a binary decision for item selection?
   1. **0/1 Knapsack**
   2. Fractional Knapsack
   3. Bounded Knapsack
   4. Unbounded Knapsack
4. Dynamic programming is applicable to problems that exhibit which of the following properties?
   1. **Overlapping subproblems**
   2. Large input size
   3. Linear time complexity
   4. Simple recursive structure
5. In multidimensional dynamic programming, what is typically represented in the form of a table?
   1. Recursive function calls
   2. Variables
   3. **Subproblem solutions**
   4. Loops
6. In Dynamic Programming on Trees, what is the primary purpose of the "visited" flag for nodes?
   1. **To track nodes that have not been processed**
   2. To keep track of the tree's structure
   3. To store intermediate results
   4. To represent the tree's root node
7. In the greedy algorithm for Huffman coding, how are characters represented in the tree?
   1. As nodes
   2. As edges
   3. **As leaves**
   4. As roots
8. What is the result of 9 ^ 9 in binary?
   1. **0**
   2. 1
   3. 9
   4. 18
9. What is the time complexity of the Sieve of Eratosthenes for finding prime numbers up to 'n'?
   1. O(n)
   2. **O(n logn log n)**
   3. O(n^2)
   4. O(sqrt(n))
10. Two girls have picked 10 roses, 15 sunflowers and 14 daffodils. What is the number of ways they can divide the flowers amongst themselves?
    1. 1638
    2. 2100
    3. **2640**
    4. None of the above

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

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

1. Which of the following standard algorithms is not Dynamic Programming based?
   1. Bellman–Ford Algorithm for single source shortest path
   2. Floyd Warshall Algorithm for all pairs shortest paths
   3. 0-1 Knapsack problem
   4. **Prim's Minimum Spanning Tree**
2. We use dynamic programming approach when
   1. We need an optimal solution
   2. **The solution has optimal substructure**
   3. The given problem can be reduced to the 3-SAT problem
   4. It's faster than Greedy
3. A networking company uses a compression technique to encode the message before transmitting over the network. Suppose the message contains the following characters with their frequency:

character Frequency

a 5

b 9

c 12

d 13

e 16

f 45

Note : Each character in input message takes 1 byte. If the compression technique used is Huffman Coding, how many bits will be saved in the message?

1. 224
2. 800
3. **576**
4. 324
5. what will do the following code in bit manipulation?

int function(int n)

{

if (n % 4 == 0)

return n;

if (n % 4 == 1)

return 1;

if (n % 4 == 2)

return n + 1;

else

return 0;

}

1. It will return the last set bit in a number.
2. It will return the first set bit in a number.
3. It will xor of two numbers.
4. **It will give the xor of numbers from 1 to N**
5. What is the pseudo code to compute the shortest path in Dijkstra’s algorithm?

**a)**

**if(!T[w].Known)**

**if(T[v].Dist + C(v,w) < T[w].Dist) {**

**Decrease(T[w].Dist to T[v].Dist +C(v,w));**

**T[w].path=v; }**

b)

if(T[w].Known)

if(T[v].Dist + C(v,w) < T[w].Dist) {

Increase (T[w].Dist to T[v].Dist +C(v,w));

T[w].path=v; }

c)

if(!T[w].Known)

if(T[v].Dist + C(v,w) > T[w].Dist) {

Decrease(T[w].Dist to T[v].Dist +C(v,w);

T[w].path=v; }

d)

if(T[w].Known)

if(T[v].Dist + C(v,w) < T[w].Dist) {

Increase(T[w].Dist to T[v].Dist);

T[w].path=v; }

**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;**

**}**