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

**ST-4 (SET-V)**

**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 classic example of a "Counting" type 1-Dimensional DP problem?
2. Fibonacci Sequence
3. Longest Common Subsequence
4. **Coin Change**
5. Knapsack Problem
6. In a grid DP problem, what does the "overlapping subproblems" property imply?
   1. Subproblems are entirely unrelated.
   2. **The same subproblems are solved multiple times.**
   3. Subproblems have no common elements.
   4. Subproblems always have unique solutions.
7. Which algorithm is used to solve the Fractional Knapsack problem optimally?
   1. **Greedy algorithm**
   2. Dynamic programming
   3. Backtracking
   4. Branch and bound
8. What is the primary advantage of using dynamic programming over a naive recursive approach?
   1. **Faster execution**
   2. Smaller code
   3. Easier implementation
   4. Higher memory efficiency
9. In Dynamic Programming on Trees, what does the term "bottom-up" approach refer to?
   1. **Starting from the leaves and moving up to the root**
   2. Starting from the root and moving down to the leaves
   3. Moving from one branch to another within the tree
   4. Traversing the tree level by level
10. Which greedy algorithm is used for task scheduling to minimize waiting time?
    1. Huffman coding
    2. Dijkstra's algorithm
    3. **Shortest Job First (SJF)**
    4. Kruskal's algorithm
11. How do you set the i-th bit of an integer x to 1?
    1. **x | (1 << i)**
    2. x & (1 << i)
    3. x ^ (1 << i)
    4. x >> i
12. If φ(10) is the Euler's Totient Function of 10, what is its value?
    1. 1
    2. 2
    3. **4**
    4. 5
13. How many onto (or surjective) functions are there from an n-element (n >= 2) set to a 2-element set?
    1. 2(2^n - 2)
    2. **2^n - 2**
    3. 2^n - 1
    4. 2^n
14. What is the possible number of reflexive relations on a set of 5 elements?
    1. 2^25
    2. **2^20**
    3. 2^15
    4. 2^10

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

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

11) What is the output of the below code: for n=5 and arr[] = {3, 10, 2, 1, 20};

int function(int arr[], int n)

{

int lis[n];

lis[0] = 1;

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

lis[i] = 1;

for (int j = 0; j < i; j++)

if (arr[i] > arr[j] && lis[i] < lis[j] + 1)

lis[i] = lis[j] + 1;

}

return \*max\_element(lis, lis + n);

}

a) 2

b) 10

c) 1

**d) 3**

12) The Partition Problem can be reduced to which well-known problem?

a) Knapsack Problem

b) Traveling Salesman Problem

**c) Subset Sum Problem**

d) Graph Coloring Problem

13) Define Rn to be the maximum amount earned by cutting a rod of length n meters into one or more pieces of integer length and selling them. For i>0, let p[i] denote the selling price of a rod whose length is i meters. Consider the array of prices:

p[1]=1, p[2]=5, p[3]=8, p[4]=9, p[5]=10, p[6]=17, p[7]=18

Which of the following statements is/are correct about R7?

**a) R7=18**

b) R7=19

**c) R7 is achieved by three different solutions**

d) R7 cannot be achieved by a solution consisting of three pieces

14) What is the return value of following function for arr[] = {9, 12, 2, 11, 2, 2, 10, 9, 12, 10, 9, 11, 2} and n is size of this array.

int fun(int arr[], int n)

{

int x = arr[0];

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

x = x ^ arr[i];

return x;

}

a) 0

**b) 9**

c) 12

d) 2

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

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

int arr[] = {5, 2, 6, 1, 3};

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

sort(arr, arr + n, greater<int>());

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

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

}

return 0;

}

a) 1 2 3 5 6

**b) 6 5 3 2 1**

c) 1 3 2 5 6

d) 6 5 2 3 1

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

Q16) Write a C++ code to rotate bits of number n by specified number d.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | n=16 d=2 | n=90 d=4 | n=4 d=1 |
| **Output** | Left Rotation of 16 by 2 is 64  Right Rotation of 16 by 2 is 4 | Left Rotation of 90 by 4 is 1440  Right Rotation of 90 by 4 is -1610612731 | Left Rotation of 4 by 1 is 8  Right Rotation of 4 by 1 is 2 |

Solution :

**#include<iostream>**

**using namespace std;**

**#define INT\_BITS 32**

**class test**

**{**

**/\*Function to left rotate n by d bits\*/**

**public:**

**int leftRotate(int n, unsigned int d)**

**{**

**/\* In n<<d, last d bits are 0. To put first 3 bits of n at**

**last, do bitwise or of n<<d with n >>(INT\_BITS - d) \*/**

**return (n << d)|(n >> (INT\_BITS - d));**

**}**

**/\*Function to right rotate n by d bits\*/**

**int rightRotate(int n, unsigned int d)**

**{**

**/\* In n>>d, first d bits are 0.To put last 3 bits of at first, do bitwise or of n>>d with n <<(INT\_BITS - d) \*/**

**return (n >> d)|(n << (INT\_BITS - d));**

**}**

**};**

**int main()**

**{**

**test g;**

**int n = 16;**

**int d = 2;**

**cout << "Left Rotation of " << n <<" by " << d << " is ";**

**cout << g.leftRotate(n, d);**

**cout << "\nRight Rotation of " << n <<" by " << d << " is ";**

**cout << g.rightRotate(n, d);**

**getchar();**

**}**

Q17) Given a two strings S and T, find the count of distinct occurrences of T in S as a subsequence.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | S = banana, T = ban | S = pizzapizza, T = za | S = rapido, T = ra |
| **Output** | 3 | 6 | 1 |

Solution :

**/\* C/C++ program to count number of times S appears as a subsequence in T \*/**

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

**using namespace std;**

**int f(int i, int j, string s, string t,**

**vector<vector<int> >& dp)**

**{**

**if (t.size() - j > s.size() - i)**

**return 0;**

**if (j == t.size()) {**

**return 1;**

**}**

**if (i == s.size()) {**

**return 0;**

**}**

**if (dp[i][j] != -1) {**

**return dp[i][j];**

**}**

**if (s[i] == t[j]) {**

**return dp[i][j] = f(i + 1, j + 1, s, t, dp)**

**+ f(i + 1, j, s, t, dp);**

**}**

**return dp[i][j] = f(i + 1, j, s, t, dp);**

**}**

**int findSubsequenceCount(string s, string t)**

**{**

**vector<vector<int> > dp(s.size(),vector<int>(t.size(), -1));**

**return f(0, 0, s, t, dp);**

**}**

**int main()**

**{**

**string T = "za";**

**string S = "pizzapizza";**

**cout << findSubsequenceCount(S, T) << endl;**

**return 0;**

**}**

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

Q18) Given a text and a wildcard pattern, implement wildcard pattern matching algorithm that finds if wildcard pattern is matched with text. The matching should cover the entire text (not partial text). The wildcard pattern can include the characters ‘?’ and ‘\*’

‘?’ – matches any single character

‘\*’ – Matches any sequence of characters (including the empty sequence)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | str = "baaabab"  pattern = "\*\*\*\*\*ba\*\*\*\*\*ab" | str = "baaabab"  pattern = "aa?ab" | str = "baaabab"  pattern = "\*baaaba\*" |
| **Output** | Yes | No | Yes |

Solution :

**// C++ program to implement wildcard pattern matching algorithm**

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

**using namespace std;**

**// Function that matches input str with given wildcard pattern**

**bool strmatch(char str[], char pattern[], int n, int m)**

**{**

**// empty pattern can only match with empty string**

**if (m == 0)**

**return (n == 0);**

**// lookup table for storing results of subproblems**

**bool lookup[n + 1][m + 1];**

**// initialize lookup table to false**

**memset(lookup, false, sizeof(lookup));**

**// empty pattern can match with empty string**

**lookup[0][0] = true;**

**// Only '\*' can match with empty string**

**for (int j = 1; j <= m; j++)**

**if (pattern[j - 1] == '\*')**

**lookup[0][j] = lookup[0][j - 1];**

**// fill the table in bottom-up fashion**

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

**for (int j = 1; j <= m; j++) {**

**// Two cases if we see a '\*'**

**// a) We ignore ‘\*’ character and move to next character in the pattern, i.e., ‘\*’ indicates an empty sequence.**

**// b) '\*' character matches with ith character in input**

**if (pattern[j - 1] == '\*')**

**lookup[i][j]**

**= lookup[i][j - 1] || lookup[i - 1][j];**

**// Current characters are considered as matching in two cases**

**// (a) current character of pattern is '?'**

**// (b) characters actually match**

**else if (pattern[j - 1] == '?'**

**|| str[i - 1] == pattern[j - 1])**

**lookup[i][j] = lookup[i - 1][j - 1];**

**// If characters don't match**

**else**

**lookup[i][j] = false;**

**}**

**}**

**return lookup[n][m];**

**}**

**int main()**

**{**

**char str[] = "baaabab";**

**char pattern[] = "\*\*\*\*\*ba\*\*\*\*\*ab";**

**// char pattern[] = "ba\*\*\*\*\*ab";**

**// char pattern[] = "ba\*ab";**

**// char pattern[] = "a\*ab";**

**// char pattern[] = "a\*\*\*\*\*ab";**

**// char pattern[] = "\*a\*\*\*\*\*ab";**

**// char pattern[] = "ba\*ab\*\*\*\*";**

**// char pattern[] = "\*\*\*\*";**

**// char pattern[] = "\*";**

**// char pattern[] = "aa?ab";**

**// char pattern[] = "b\*b";**

**// char pattern[] = "a\*a";**

**// char pattern[] = "baaabab";**

**// char pattern[] = "?baaabab";**

**// char pattern[] = "\*baaaba\*";**

**if (strmatch(str, pattern, strlen(str),**

**strlen(pattern)))**

**cout << "Yes" << endl;**

**else**

**cout << "No" << endl;**

**return 0;**

**}**