**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) Given a number N having only one ‘1’ and all other ’0’s in its binary representation, find position of the only set bit. If there are 0 or more than 1 set bit the answer should be -1. Position of set bit ‘1’ should be counted starting with 1 from the LSB side in the binary representation of the number.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | N = 2 | N = 5 | N= 128 |
| **Output** | 2 | -1 | 8 |

Solution :

**// C++ program to find position of only set bit in a given number**

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

**using namespace std;**

**// A utility function to check whether n is power of 2 or not**

**int isPowerOfTwo(unsigned n)**

**{**

**return n && (!(n & (n - 1)));**

**}**

**// Returns position of the only set bit in 'n'**

**int findPosition(unsigned n)**

**{**

**if (!isPowerOfTwo(n))**

**return -1;**

**unsigned count = 0;**

**// One by one move the only set bit to right till it reaches end**

**while (n)**

**{**

**n = n >> 1;**

**// increment count of shifts**

**++count;**

**}**

**return count;**

**}**

**int main(void)**

**{**

**int n = 0;**

**int pos = findPosition(n);**

**(pos == -1) ? cout<<"n = "<<n<<", Invalid number\n" :**

**cout<<"n = "<<n<<", Position "<< pos<<endl;**

**n = 12;**

**pos = findPosition(n);**

**(pos == -1) ? cout<<"n = "<<n<<", Invalid number\n" :**

**cout<<"n = "<<n<<", Position "<< pos<<endl;**

**n = 128;**

**pos = findPosition(n);**

**(pos == -1) ? cout<<"n = "<<n<<", Invalid number\n" :**

**cout<<"n = "<<n<<", Position "<< pos<<endl;**

**return 0;**

**}**

Q17) Given a paper of size, A x B. Task is to cut the paper into squares of any size. Find the minimum number of squares that can be cut from the paper.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | 13 x 29 | 4 x 5 | 10 x 10 |
| **Output** | 9 | 5 | 1 |

Solution :

**// C++ program to find minimum number of squares to cut a paper.**

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

**using namespace std;**

**// Returns min number of squares needed**

**int minimumSquare(int a, int b)**

**{**

**long long result = 0, rem = 0;**

**// swap if a is small size side .**

**if (a < b)**

**swap(a, b);**

**// Iterate until small size side is**

**// greater than 0**

**while (b > 0)**

**{**

**// Update result**

**result += a/b;**

**long long rem = a % b;**

**a = b;**

**b = rem;**

**}**

**return result;**

**}**

**int main()**

**{**

**int n = 13, m = 29;**

**cout << minimumSquare(n, m);**

**return 0;**

**}**

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

Q18) Write a dynamic programming based function to find nth Catalan number.

Catalan numbers are defined as a mathematical sequence that consists of positive integers, which can be used to find the number of possibilities of various combinations.

The nth term in the sequence denoted Cn, is found in the following formula: \frac{(2n)!}{(n + 1)! n!)}

The first few Catalan numbers for n = 0, 1, 2, 3, … are : 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, …

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | n = 6 | n = 8 | n=1 |
| **Output** | 132 | 1430 | 1 |

Solution :

**#include <iostream>**

**using namespace std;**

**// A dynamic programming based function to find nth Catalan number**

**unsigned long int catalanDP(unsigned int n)**

**{**

**// Table to store results of subproblems**

**unsigned long int catalan[n + 1];**

**// Initialize first two values in table**

**catalan[0] = catalan[1] = 1;**

**// Fill entries in catalan[] using recursive formula**

**for (int i = 2; i <= n; i++) {**

**catalan[i] = 0;**

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

**catalan[i] += catalan[j] \* catalan[i - j - 1];**

**}**

**// Return last entry**

**return catalan[n];**

**}**

**int main()**

**{**

**for (int i = 0; i < 10; i++)**

**cout << catalanDP(i) << " ";**

**return 0;**

**}**