1. What is the primary characteristic of a recursive algorithm?
   1. It uses loops extensively
   2. **It solves problems by dividing them into smaller, identical subproblems**
   3. It doesn't involve function calls
   4. It always requires a stack data structure
2. In recursion, what is the base case?
   1. The largest input value
   2. **A condition that terminates the recursion**
   3. The initial value of the variable
   4. The condition where recursion continues indefinitely
3. Which of the following is NOT a requirement for a recursive function?
   1. Base case
   2. Recursive case
   3. **Global variables**
   4. Progress towards base case
4. What is the drawback of using recursion in terms of memory usage?
   1. Recursion always uses less memory than iteration
   2. **Recursion can lead to excessive memory usage due to the call stack**
   3. Recursion has no impact on memory usage
   4. Recursion uses the heap instead of the stack
5. The process of a function calling itself directly or indirectly is known as:
   1. Looping
   2. Iteration
   3. **Recursion**
   4. Overloading
6. Backtracking is:
   1. A search algorithm that finds the shortest path
   2. An algorithm used to solve linear equations
   3. An optimization technique for recursion
   4. **An algorithmic technique to solve problems by trying multiple options and undoing if necessary**
7. Which data structure is commonly used to implement backtracking?
   1. Queue
   2. **Stack**
   3. Linked list
   4. Array
8. In backtracking, what is pruning?
   1. Removing dead code from the program
   2. **Cutting off branches of the search tree that cannot lead to a solution**
   3. Reversing the order of exploration
   4. Limiting the depth of recursion
9. The "N-Queens Problem" is a classic example often used to illustrate:
   1. Divide and conquer
   2. Greedy algorithms
   3. Dynamic programming
   4. **Backtracking**
10. What is the main idea behind the "subset sum problem" that makes it a backtracking candidate?
    1. **It involves finding all possible subsets of a set**
    2. It requires sorting the elements of the set
    3. It can be solved using a single loop
    4. It has an exponential number of possible solutions
11. When solving problems using backtracking, the process of undoing a move and trying a different one is known as:
    1. Pruning
    2. Branching
    3. Reverting
    4. **Back-jumping**
12. Which of the following problems is best suited for backtracking?
    1. Finding the maximum element in an array
    2. Calculating the factorial of a number
    3. **Solving a maze to find the shortest path**
    4. Counting the number of occurrences of a character in a string
13. A recursive function is called with an argument value of 4. How many times will the function be called if it contains only one recursive call and the base case is reached when the argument value becomes 1?
    1. 1
    2. **2**
    3. 3
    4. 4
14. Which of the following techniques can help improve the efficiency of recursive algorithms?
    1. Using global variables
    2. Using larger data structures
    3. **Memoization**
    4. Avoiding base cases
15. In a recursive function, what is the role of the call stack?
    1. It stores the values of global variables
    2. It prevents infinite recursion
    3. **It stores the return addresses of function calls**
    4. It is responsible for generating random numbers
16. Which of the following statements is true regarding tail recursion?
    1. It is a type of recursion that involves multiple recursive calls
    2. **It is a form of recursion where the recursive call is the last operation in the function**
    3. It can't be optimized by compilers
    4. It always leads to infinite recursion
17. The Fibonacci sequence is a classic example of a problem that can be solved efficiently using:
    1. Recursion
    2. Backtracking
    3. **Dynamic programming**
    4. Binary search
18. What is the time complexity of generating all permutations of a set using backtracking?
    1. O(n)
    2. O(n log n)
    3. **O(n!)**
    4. O(2^n)
19. What is the primary difference between iteration and recursion?
    1. Iteration uses function calls, while recursion uses loops
    2. **Iteration doesn't involve function calls, while recursion does**
    3. Iteration always requires a stack, while recursion doesn't
    4. Iteration is more memory-efficient than recursion
20. The "Traveling Salesman Problem" can be effectively solved using:
    1. Greedy algorithms
    2. Divide and conquer
    3. Dynamic programming
    4. **Backtracking**
21. In backtracking, what is the term for the process of moving back one step to a previous state?
    1. **Undoing**
    2. Pruning
    3. Reverting
    4. Back-jumping
22. The process of finding all possible permutations of a string is an example of:
    1. Dynamic programming
    2. Divide and conquer
    3. Recursion
    4. **Backtracking**
23. Which of the following problems cannot be solved using backtracking?
    1. Sudoku solving
    2. **Finding the greatest common divisor of two numbers**
    3. Solving the Tower of Hanoi puzzle
    4. Generating all valid parentheses combinations
24. The process of finding the "power set" of a given set refers to:
    1. Finding the set of prime numbers in the given set
    2. **Finding the set of all possible subsets of the given set**
    3. Finding the set of all odd numbers in the given set
    4. Finding the set of all even numbers in the given set
25. In backtracking, how do you know when a solution is valid?
    1. By checking if the solution satisfies the base case condition
    2. By comparing it to a predefined solution
    3. By ensuring it doesn't involve any recursive calls
    4. **By testing it against a set of constraints**
26. What is the purpose of the "visited" array in a backtracking solution?
    1. To keep track of the number of recursive calls
    2. **To mark which elements have been explored or used in the solution**
    3. To store the computed results of subproblems
    4. To control the flow of execution in the recursive function
27. The process of solving a problem by breaking it down into smaller subproblems of the same type is known as:
    1. Dynamic programming
    2. Memoization
    3. **Divide and conquer**
    4. Backtracking
28. The Fibonacci sequence can be implemented using recursion. What is the base case for the Fibonacci recursion?
    1. **fib(0) = 0**
    2. fib(1) = 1
    3. fib(2) = 1
    4. fib(n) = n
29. Backtracking is an algorithmic technique for solving problems that involves:
    1. **Trying all possible solutions**
    2. Iterating through a loop
    3. Using dynamic programming
    4. Using recursion only
30. What is the main characteristic of backtracking algorithms?
    1. They guarantee the optimal solution.
    2. They use divide and conquer approach.
    3. They use recursion for efficient solutions.
    4. **They explore multiple possibilities and backtrack when needed.**
31. The "Sudoku" puzzle can be solved using backtracking. What is the main goal of this problem?
    1. To find the highest number in the puzzle
    2. To rearrange the numbers in ascending order
    3. **To fill a 9x9 grid with digits so that each column, each row, and each of the nine 3x3 sub-grids that compose the grid contains all of the digits from 1 to 9**
    4. To rotate the puzzle to a specific angle
32. What is the primary challenge in solving the "Knapsack" problem using backtracking?
    1. Determining the maximum number of items
    2. Fitting the items into the knapsack
    3. **Maximizing the value of selected items while staying within the knapsack's capacity**
    4. Minimizing the weight of selected items
33. #include <iostream>

using namespace std;

void printSubsetSum(int arr[], int n, int sum, int idx, int currentSum) {

if (idx == n) {

if (currentSum == sum) {

cout << "Subset found with sum " << sum << endl;

}

return;

}

printSubsetSum(arr, n, sum, idx + 1, currentSum);

printSubsetSum(arr, n, sum, idx + 1, currentSum + arr[idx]);

}

int main() {

int arr[] = {2, 4, 6, 8};

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

int sum = 10;

printSubsetSum(arr, n, sum, 0, 0);

return 0;

}

a) Subset found with sum 10

**b) Subset found with sum 10**

**Subset found with sum 10**

c) Subset found with sum 18

d) No subset found with the given sum

1. What is the operation performed by given function?

void fun(string str, int l, int r) {

if (l == r) {

cout << str << endl;

return;

}

for (int i = l; i <= r; i++) {

swap(str[l], str[i]);

fun(str, l + 1, r);

swap(str[l], str[i]); // Backtrack

}

}

1. print string in reverse order
2. **print permutations of string**
3. print anagrams of string
4. swapping string character at first and last location
5. What is the role of below function?

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

if (board[i][col] == 1) {

return false;

}

}

1. check if there is 1 in upper left diagonal
2. check if there is 1 in upper right diagonal
3. **check if there is 1 in same column**
4. check if there is 1 in same row
5. #include <iostream>

using namespace std;

int countWays(int n) {

if (n <= 1) {

return 1;

}

return countWays(n - 1) + countWays(n - 2);

}

int main() {

int n = 4;

cout << "Ways: " << countWays(n) << endl;

return 0;

}

1. **Ways: 5**
2. Ways: 8
3. Ways: 4
4. Ways: 6
5. #include <iostream>

using namespace std;

void backtrack(int n) {

if (n <= 0) return;

cout << n << " ";

backtrack(n - 2);

cout << n << " ";

}

int main() {

backtrack(6);

return 0;

}

1. 6 4 2 1 3 5
2. 6 5 4 3 2 1
3. **6 4 2 2 4 6**
4. 6 4 2 1 3 5
5. #include <iostream>

using namespace std;

void generateSubsets(string str, int index, string current) {

if (index == str.length()) {

cout << "{" << current << "}" << " ";

return;

}

generateSubsets(str, index + 1, current);

generateSubsets(str, index + 1, current + str[index]);

}

int main() {

generateSubsets("abc", 0, "");

return 0;

}

1. {} {a} {b} {c} {ab} {ac} {bc} {abc}
2. **{} {c} {b} {bc} {a} {ac} {ab} {abc}**
3. {abc} {ab} {ac} {bc} {a} {b} {c} {}
4. {} {a} {b} {c} {ab} {ac} {bc} {abc}