CSCE 321, Dr. Goneid
Exercises (8) Backtracking

1. Consider 5 objects with weights \((w_1,w_2,\ldots,w_5) = (10, 3, 5, 7, 2)\) and a knapsack with size \(m = 15\). Use a backtracking method to find all combinations of objects that will exactly fill the knapsack. Compute the efficiency of that backtracking method.

**Solution:**
Let \(X = \{x_1,x_2,\ldots,x_n\}\) be a solution such that \(x_i \in \{0,1\}\), 
\(x_i = 1\) if \(w_i\) is selected and \(x_i = 0\) otherwise. Hence, we obtain fixed-size tuples.
Draw a permutation tree with 5 layers. The possible answers are:
\(\{1, 1, 0, 0, 1\}, \{1, 0, 1, 0, 0\}, \{0, 1, 1, 1, 0\}\)
Brute force generates \(M = 63\) nodes. Compute \(N\) the number of nodes generated in the backtracking permutation tree and then the gain is \(1 - \frac{N}{M}\)

2. Consider the set of integers \(S = \{20, 15, 10, 7, 5\}\) and \(m = 30\). Draw the portion of the state space that is generated in the backtracking algorithm to find all possible subsets of \(S\) that sum exactly to \(m\). Compute the efficiency of that backtracking method relative to the brute force method for that problem.

3. Implement a backtracking algorithm to find all permutations of the four digits \((1, 2, 3, 4)\) such that no digit is repeated and the absolute difference between a digit and the previous digit is always greater than or equal to 2. Test your algorithm by tracing it using part of the permutation tree to find at least one permutation satisfying the above constraints.

**Solution:**
Let \(X = \{x_0, x_1, x_2, x_3, x_4, \ldots, N\}\) be a solution such that \(x_0 = -2\) and \(x_i = \) digit at position \(i\). The bound function is that digit \((i)\) can take position \((k)\) given that all previous positions have been assigned digits. Hence a bound would be:

```cpp
bool Can-Take(i, k)
{
    for all previous positions 
        if digit is already there, return false
        if \(\text{abs}(x_{k-1} - i) < 2\) return false; // difference violates condition
        return true;
}
```

Build the permutation backtracking algorithm **Permute** \((k,N)\) for \(N\) digits using the above function. 
To test:
Start with \(x = \{-2, 0, 0, 0\}\) and call Permute \((1,4)\).
One possible answer is \(\{2, 4, 1, 3\}\) and another is \(\{3, 1, 4, 2\}\)
4. Consider a graph with 5 vertices (A…E) and the following edges: (AB), (AE), (CD), (DE). Draw the portion of the permutation tree representing the backtracking method to find a possible coloring of this graph using only two colors (e.g. Black and White).

Answer: a possible coloring is A: Black, B:white, C:White, D: Black, E:White

5. Given the following backtracking function:

```c++
void Backtrack ( int b, int c, int d)
{
    if (b <= c)
    {
        cout << d;
        for ( int k = b; k <= c; k++)
            Backtrack (b+1 , c , k);
    }
}
```

Trace the function to find the output of the call Backtrack (1 , 3 , 9).

6. Write a backtracking function with a bound condition to find all permutations of the four characters (A, B, C, D) satisfying the constraints:

Constraint (1): No character is repeated and
Constraint (2): The absolute ASCII difference between a character and the previous one is always greater than 2.

Test your algorithm by tracing it using part of the permutation tree to find all permutations satisfying the above constraints. How many answers are obtained? If only constraint (1) is imposed, what will be the number of possible answers?

Solution: Same as problem (3). The possible answers are only 2: {B, D, A, C} and {C, A, D, B}
If constraint (2) is removed, we obtain 4! = 24 answers.

7. Given n = 4 Boolean variables x_j = 0 or 1 (j = 1, 2, 3, 4) subject to the implicit constraints that x_1 \neq x_2 and the number of variables with value (1) must be odd:
(a) Draw the permutation tree of this problem for the backtracking algorithm.
(b) Find the answer vectors for the problem.
(c) How many nodes are generated, have survived or have been killed?
(d) What is the efficiency of the method compared to a brute force technique?

Answer:
{0100}, {0111}, {1000}, {1011}, 19 nodes have been generated so that the gain is 1 – 19/31 = 38.7%
8. Given \( n = 4 \) binary pixels (black ‘0’ or white ‘1’) arranged in a square. A backtracking algorithm is to flag (1) iff all pixels are black or just one pixel is white:
   - Draw the permutation tree of this problem for the backtracking algorithm.
   - Find the answer vectors for the problem.
   - How many nodes are generated, have survived or have been killed?
   - What is the bound function for this problem? Write a C++ backtracking function to find all possible answer vectors of the problem.

   **Answers:** Possible answers are \{000\}, \{0001\}, \{0010\}, \{0100\} and \{1000\}

9. Draw the permutation tree for the 3-Queens problem and find the answer vector(s) if any.

   **Solution:** After drawing the permutation tree, you will find that there is no possible answer to this problem.

10. Consider the shown arrangement of objects. Given only two colors (Black and White), we would like to color the objects such that no two touching objects have the same color.
    - Draw the permutation tree representing the backtracking method to find a possible coloring of this arrangement.
    - What is the efficiency of the method compared to a brute force technique?

    **Answer:** Model the arrangement as a graph and color it as in problem (3). A possible coloring is: A:Black, B:White, C:White, D:Black, E:White

11. Apply backtracking to the problem of finding a Hamiltonian circuit in the following graph.