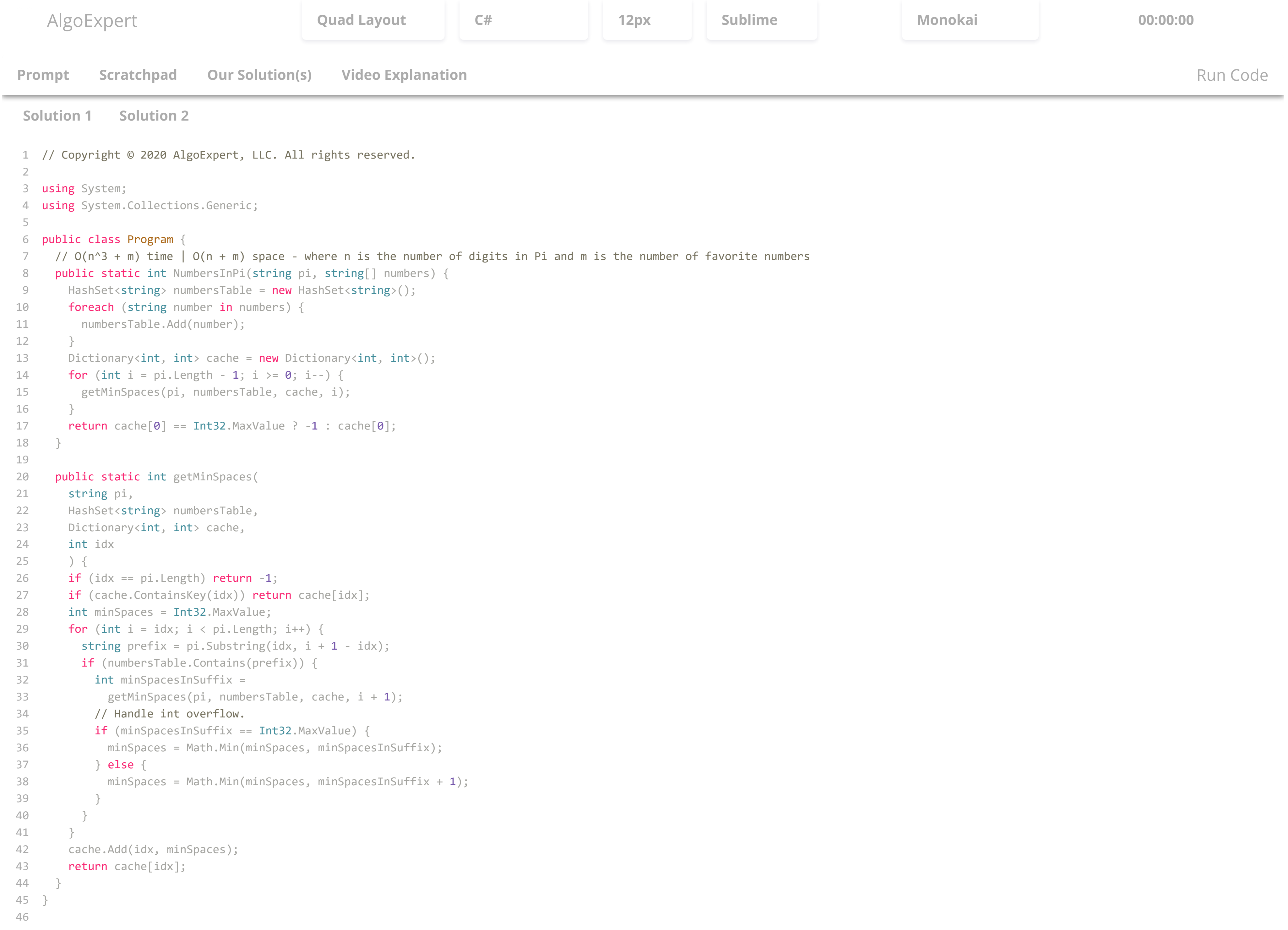

```
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

using System;
using System.Collections.Generic;

public class Program {
  // O(n^3 + m) time | O(n + m) space - where n is the number of digits in Pi and m is the number of favorite numbers
  public static int NumbersInPi(string pi, string[] numbers) {
    HashSet<string> numbersTable = new HashSet<string>();
    foreach (string number in numbers) {
      numbersTable.Add(number);
    }
    Dictionary<int, int> cache = new Dictionary<int, int>();
    for (int i = pi.Length - 1; i >= 0; i--) {
      getMinSpaces(pi, numbersTable, cache, i);
    }
    return cache[0] == Int32.MaxValue ? -1 : cache[0];
  }

  public static int getMinSpaces(
    string pi,
    HashSet<string> numbersTable,
    Dictionary<int, int> cache,
    int idx
    ) {
    if (idx == pi.Length) return -1;
    if (cache.ContainsKey(idx)) return cache[idx];
    int minSpaces = Int32.MaxValue;
    for (int i = idx; i < pi.Length; i++) {
      string prefix = pi.Substring(idx, i + 1 - idx);
      if (numbersTable.Contains(prefix)) {
        int minSpacesInSuffix =
          getMinSpaces(pi, numbersTable, cache, i + 1);
        // Handle int overflow.
        if (minSpacesInSuffix == Int32.MaxValue) {
          minSpaces = Math.Min(minSpaces, minSpacesInSuffix);
        } else {
          minSpaces = Math.Min(minSpaces, minSpacesInSuffix + 1);
        }
      }
    }
    cache.Add(idx, minSpaces);
    return cache[idx];
  }
}
```