https://www.codeproject.com/Tips/5299463/How-to-Unscramble-Any-Word?fid=1969546&df=90&mpp=25&sort=Position&spc=Relaxed&prof=True&view=Normal&fr=26#xx0xx

How to Unscramble Any Word

This is an unscramble class that can be used to decypher any word.
I could never find a good way to unscramble a word on the interwebs. Every algorithm was either brute-force or permutations...

Introduction
I could never find a good way to unscramble a word on the interwebs. Every algorithm was either brute-force or permutations.
The problem with brute-force is that it's a guessing game... very slow, or if you're lucky, very fast.
The problem with permutations is that once you go over 7 characters, you're using a bunch of memory.
e.g., a 12 character scrambled word has **479,001,600** configurations!
It finally dawned on me that if you sort the scrambled word and then sort the dictionary entries, then if we equate any sorted dictionary entry to our sorted scramble, then they must be a match!
There is probably some fancy machine learning algorithm that could do this, but my method works perfectly and instantly.

Using the Code
You'll want to embed your favorite dictionary into your project (for speed and portability).
There are a lot of free dictionary files out there; here's the one I used... https://github.com/dwyl/english-words.

Direct link... https://raw.githubusercontent.com/dwyl/english-words/master/words.txt.
The work-horse is the **UnscrambleWord** method; this will take care of loading the dictionary, filtering and then sorting the results and storing them in a List<string> object that will be returned to you from the call.

```
class Unscramble
{
     private static bool _dictionaryLoaded = false;
     private static string _wordToUnscramble = "";
     private static int _totalEntries = 0;
     private static Dictionary<string, string> _sortedDictionary =
                                        new Dictionary<string, string>();
     private static List<string> _results = new List<string>();
     private static Stopwatch _stopwatch;

     //=======================================================================
     /** We don't really need a constructor **/
     //public Unscramble(string wordToUnscramble)
     //{
     //    _WordToUnscramble = wordToUnscramble;
     //}

     //=======================================================================
     public List<string> UnscrambleWord(string wordToUnscramble, bool useFiltering = true)
     {
         _stopwatch = Stopwatch.StartNew();

         if (string.IsNullOrEmpty(_wordToUnscramble))
         {
```

```
        _wordToUnscramble = wordToUnscramble;
    }
    else if (!_wordToUnscramble.Equals
         (wordToUnscramble, StringComparison.OrdinalIgnoreCase) && useFiltering)
    {   //If re-using the object and the word is different,
        //we'll need to reload the dictionary
        _dictionaryLoaded = false;
        _wordToUnscramble = wordToUnscramble;
        _results.Clear();
    }
    else if (_wordToUnscramble.Equals
            (wordToUnscramble, StringComparison.OrdinalIgnoreCase))
    {
        _results.Clear(); //we should clear the results array so they don't stack
    }

    if (!_dictionaryLoaded) //the first call will be slightly slower
        LoadEmbeddedDictionary(wordToUnscramble.ToUpper(), useFiltering);

    string scrambleSorted = SortWord(wordToUnscramble);

    //var kvp = SortedDictionary.FirstOrDefault
    //(p => SortedDictionary.Comparer.Equals(p.Value, scrambledSort));
    var matchList = _sortedDictionary.Where
        (kvp => kvp.Value == scrambleSorted).Select(kvp => kvp.Key).ToList();

    if (matchList.Count > 0)
    {
        foreach (string result in matchList)
        {
            System.Diagnostics.Debug.WriteLine($"> Match: {result}");
            _results.Add(result);
        }

        _stopwatch.Stop();
        System.Diagnostics.Debug.WriteLine($"> Elapsed time: {_stopwatch.Elapsed}");
        return _results;
    }
    else //no matches
    {
        _stopwatch.Stop();
        _results.Clear();
        System.Diagnostics.Debug.WriteLine($"> Elapsed time: {_stopwatch.Elapsed}");
        return _results;
    }
}

//=====================================================================
private static void LoadEmbeddedDictionary(string wordText, bool filter = false)
{
    char[] delims = new char[1] { '\n' };
    string[] chunks;
    int chunkCount = 0;
    if (filter)
```

```
            chunks = global::Utility.Properties.Resources.
                                    DictionaryNums.ToUpper().Split(delims);
        else
            chunks = global::Utility.Properties.Resources.
                                    DictionaryNums.ToUpper().Split(delims);

        System.Diagnostics.Debug.WriteLine($"> Length filter: {wordText.Length}");
        _sortedDictionary.Clear();
        foreach (string str in chunks)
        {
            chunkCount++;
            if (filter)
            {
                //we're assuming the word will have at least 3 characters...
                //I mean would you really need this program if it was only two?
                if ((str.Length == wordText.Length) &&
                     str.Contains(wordText.Substring(0, 1)) &&
                     str.Contains(wordText.Substring(1, 1)) &&
                     str.Contains(wordText.Substring(2, 1))) //just checking the 1st,
                                 //2nd & 3rd letter will trim our search considerably
                {
                    try
                    {
                        _sortedDictionary.Add(str, SortWord(str));
                    }
                    catch
                    {
                        //probably a key collision, just ignore
                    }
                }
            }
            else
            {
                try
                {
                    _sortedDictionary.Add(str, SortWord(str));
                }
                catch
                {
                    //probably a key collision, just ignore
                }
            }
        }
        System.Diagnostics.Debug.WriteLine($">
        Loaded {_sortedDictionary.Count} possible matches out of
{chunkCount.ToString()}");
        _totalEntries = chunkCount;
        _dictionaryLoaded = true;
    }

    //=====================================================================
    private static string SortWord(string str)
    {
        return String.Concat(str.OrderBy(c => c));
```

```
        /*** Character Array Method ***
        return String.Concat(str.OrderBy(c => c).ToArray());
        ******************************/

        /*** Traditional Method ***
        char[] chars = input.ToArray();
        Array.Sort(chars);
        return new string(chars);
        **************************/
    }

    #region [Helper Methods]
    //=========================================================================
    public TimeSpan GetMatchTime()
    {
       return _stopwatch.Elapsed;
    }

    //=========================================================================
    public List<string> GetMatchResults()
    {
       return _results;
    }

    //=========================================================================
    public int GetMatchCount()
    {
       return _results.Count;
    }

    //=========================================================================
    public int GetFilterCount()
    {
       return _sortedDictionary.Count;
    }

    //=========================================================================
    public int GetDictionaryCount()
    {
       return _totalEntries;
    }
    #endregion
}
```

Testing/Implementation

To drive the code, you would do this...

```
string scrambled = "mctmouicnaino";
Unscramble obj1 = new Unscramble();
List<string> results = obj1.UnscrambleWord(scrambled);
if (results.Count > 0)
{
    Console.WriteLine($"> Total matches: {obj1.GetMatchCount()}");
    foreach (string str in results)
```

```
    {
        Console.WriteLine($">> {str}");
    }
    Console.WriteLine($"> Total time: {obj1.GetMatchTime()}");
    Console.WriteLine($"> Filtered set: {obj1.GetFilterCount()}
                        out of {obj1.GetDictionaryCount()}");
}
else
{
    Console.WriteLine("> No matches available:
            Check your spelling, or the dictionary may be missing this word.");
}
```

In the class, we could add some more LINQ methods to change the order, take the top result, etc., but this should be a good remedy for any unscramble engine base.

History

- 11th April, 2021: Initial version