COS 226 Programming Assignment Checklist: Autocomplete Me

Frequently Asked Questions

What is meant by an immutable data type? An object is *immutable* if its data-type value (state) never changes once it is constructed. A data type is *immutable* if every object of that type is immutable. For example, to make the `Autocomplete` data type immutable, you must make a defensive copy of the `terms[]` array in the constructor (because the client might mutate the original `terms[]` array after constructing the `Autocomplete` object).

Can my methods have side effects that are not described in the API? No. For example, the `Autocomplete` data type should not mutate the `terms[]` array (or it could have a serious side effect on the client). Similarly, the `firstIndexOf()` and `lastIndexOf()` should not mutate their argument arrays. We usually allow methods to have benign side effects, such as changing the state of a random number generator.

What is the meaning of the modifier `static` in the following function declarations?

```
public static Comparator<Term> byReverseWeightOrder()
```

It means that there is one (static) method for the entire class instance of one (instance) method per object. Since there is a single total order (by weight), the static modifier is appropriate.

What is the meaning of the type parameter `Key` in the following function declaration?

```
public static <Key> int firstIndexOf(Key[] a, Key key, Comparator<Key> comparator)
```

This is an example of a generic method, which introduces its own type parameter `Key`. The type parameter serves as a placeholder for the argument types that are passed to the generic method. For example, it enforces the constraint that the type of the elements in the array `a[]` must match the type of the search key `key` (so that the compiler would allow `Apple[]` and `Apple` but reject `Apple[]` and `Orange`). This bit of Java minutiae is necessary to enable the program to compile without warnings.

When I compile BinarySearchDeluxe.java I get the compiler warning "unchecked call to compare(T, T) as a member of the raw type java.util.Comparator." Is that OK? No, your program should compile without any errors or warnings.

Can I use Arrays.binarySearch()? Yes. However, note that `Arrays.binarySearch()` returns the index of *any* matching key (not necessarily the first or last such key), so it will not be of much help.

How can I compare two strings lexicographically? Lexicographic order is the *natural* order for strings, so you can use the `compareTo()` method to do so.

What exactly does it mean to compare two string by their first r characters? You should compare the two strings *lexicographically*, as in the natural order for Java strings, but you should never examine more than *r* characters of either string. For example, if *r* = 3, then `cat` is less than `dog`; `dog` is equal to `dogcatcher`; and `do` is less than `dogcatcher`. Note that in the last example, `do` has fewer than 3 characters, so we compare the whole string `do` to the first 3 characters of `dogcatcher`.

Can I assume that the weights are all distinct? No. However, if there are two terms with equal weights, the `allMatches()` method can return them in arbitrary order.

Can I assume that the Autocomplete constructor receives the terms in descending order by weight? No. While many of the sample input files are in descending order by weight, do not make any assumptions about the order.

What should `allMatches()` return if there are no matching terms? An array of length 0.

Should Autocomplete be case sensitive? For instance, if I enter in the prefix "prince" and if one of my terms has a query "Princeton", should "Princeton" be included among the matches? On this assignment, you should assume the case matters. In a real application, you might want some kind of case insensitive order. This can be a bit tricky with non-ASCII characters (but Java does have various libraries to deal with that).

Can a prefix or query string be the empty string? Yes and yes. For example, the empty-string prefix should return all queries, in descending order by weight. It might be harder to think of an application for an empty-string query, but the API does not exclude this possibility.

What do I need to do to handle Unicode characters and UTF-8 encoding? You shouldn't need to do anything special: The Java `String` data type handles Unicode naturally; The supplied test files are encoded using UTF-8. When reading input or writing output, `StdIn`, `In`, and `StdOut` assume UTF-8 encoding.

Why do I get a `java.util.InputMismatchException` from `readLong()` when I run the sample client? You probably have an old version of `stdlib.jar` in your classpath that doesn't assume UTF-8 encoding. On Mac OS X, be sure to check `/Library/Extensions/Java` and `~/Library/Extensions/Java` folders.

Why do I get a `java.util.NoSuchElementException` from `readInt()` when I run the sample client? Did you cut-and-paste the file instead of downloading it? Your text editor may have corrupted the file by changing the encoding from UTF-8 to something else.

How can I view the Unicode characters that my program prints out?

- Mac OS X Terminal: select *Terminal -> Preferences -> Advanced -> International -> Character encoding -> Unicode (UTF-8)*.
- Windows Command Prompt: type `chcp 65001`. Use the font *Lucida Console* (or some other font that supports Unicode).
- `AutocompleteGUI`: should work without adjustment.
- DrJava: unknown.
- Safari: *View -> Text Encodings -> Unicode (UTF-8)*.

Note that even if you can't view Unicode properly, it's likely that your program is still handling Unicode properly. You can use the program UnicodeTest.java to test whether your system is configured to display Unicode.

Input, Output, and Testing

Input files. Below are some input files for testing. Each input file consists of an integer *N* followed by *N* pairs of terms and integer weights, one pair per line, with the term and weight separated by a tab. The terms can be arbitrary strings consisting of Unicode characters, including spaces (but neither tabs nor newlines).

file	number	term	weight	source
pu-buildings.txt	166	Princeton buildings	age (in Y10K)	
pokemon.txt	729	Pokemon	popularity	Allen Qin
fortune1000.txt	1,000	company	revenue	Fortune 1000
nasdaq.txt	2,658	NASDAQ	market cap	nasdaq.com

metal-albums.txt	3,000	metal albums	votes	Metal Storm
pu-courses.txt	6,771	course	enrollment	Princeton Registrar
wiktionary.txt	10,000	word	frequency	Wiktionary
redditors.txt	10,000	redditor	subscribers	redditlist.com
baby-names.txt	31,109	first name	frequency	U.S. Social Security
trademarks.txt	92,254	company	U.S. trademark number	USPTO
cities.txt	93,827	city	population	geoba.se
mandarin.txt	94,339	Mandarin word	frequency	Invoke IT
bing.txt	250,000	word	Bing search frequency	Microsoft Web N-gram
2grams.txt	277,718	2-gram	frequency	Google Web Trillion Word Corpus
3grams.txt	1,020,009	3-gram	frequency	Corpus of Contemporary American English
4grams.txt	1,034,308	4-gram	frequency	Corpus of Contemporary American English
5grams.txt	1,044,269	5-gram	frequency	Corpus of Contemporary American English
words-333333.txt	333,333	word	frequency	Google Web Trillion Word Corpus
artists.txt	43,849	music artist	familiarity	Million Song Dataset
songs.txt	922,230	song	hotttnesss	Million Song Dataset
alexa.txt	1,000,000	web site	frequency	Alexa
imdb-votes.txt	82,455	movie	number of votes	Internet Movie Database
movies.txt	229,447	movies	revenue (USD)	Internet Movie Database
actors.txt	2,875,183	actors	revenue (USD)	Internet Movie Database

Possible progress steps

These are purely suggestions for how you might make progress. You do not have to follow these steps.

- Download the directory autocomplete. It contains sample input files.
- Implement `Term.java`.
  - Begin with the constructor and `toString()`.

    - Next, add `implements Comparable<Term>` to the class declaration and implement the `compareTo()` method.
    - Then, implement `byReverseWeightOrder()`. In support of this method, you will need to define a private static nested class, as in the lecture slides.
    - Finally, implement `byPrefixOrder()`. In support of this method, you will need to define another private static nested class. Since you will need to create a comparator that depends upon an integer parameter *r*, the nested class should have a constructor that takes *r* as an argument and saves it in an instance variable.
- Implement `BinarySearchDeluxe.java`. Binary search is a deceptively simple algorithm. Jon Bentley reports that in an experiment at Bell Labs and IBM, only about 10% of professional programmers got it right; so, test, debug, test.
    - To test `BinarySearchDeluxe.java`, you need a data type with an associated comparator. You could use `Term` and one of its comparators. Note that the `compare()` method associated with this comparator may return a value other than -1, +1, and 0.
    - A common error is for binary search to work correctly on a variety of inputs but go into an infinite loop on others. Be sure to test your program on many inputs. If you have an infinite loop, print out all of your loop-control variables, such as `lo`, `mid`, and `hi` to see where it stalls.
- Implement `Autocomplete`. You will need to use all three orders defined in `Term`: the natural order, decreasing order of weight, and lexicographically based on the first *r* characters.
    - You can access the compare-by-reverse-weight comparator via the Java code

      ```
      Comparator<Term>Term.byReverseWeightOrder();
      ```

    - You can uses `XXXX.sort(terms)` to sort the array `terms[]` according to the natural order (by query string) and `XXXX.sort(terms, comparator)` to sort the array terms[] according to the order defined by `comparator`. (XXXX will be the class for the sort you choose to use.)
    - To test, use either the sample client given in the assignment specification or `AutocompleteGUI.java`.

A video is provided for those wishing additional assistance. Be forewarned that video was made in early 2014 and is somewhat out of date. For example the API has changed.

Enrichment

Suppose that I want to find only the top-k matching terms (instead of all matching terms). Are there better algorithms? Yes. The problem of finding the top term matching a given prefix can be formulated as a range maximum query problem or equivalently as a *lowest common ancestor* problem. Here are some lecture notes by Erik Demaine on the equivalence.

- Build a Cartesian tree (in linearithmic time or linear time) and do the LCA computation directly in the tree (by following parent pointers until they meet). If the tree is reasonably balanced, this ought to work well in practice (but no performance guarantees).
- Use segment trees, as in this paper to solve the maximum query problem in log *N* time per query.
- Use an algorithm described by Michael A. Bender and Martin Farach-Colton in this paper to solve the maximum query problem in constant time per query.

Any way to find the range of words that start with a given prefix in time proportional to the length of the prefix (in the worst case)? Yes, use a suffix tree (or a suffix array with the lcp array), where the terms are separated by a word-separation symbol and concatenated together. To use space proportional to the number of terms consider a suffix array on words. Or consider a compact DAWG.