Sorting strings

This week's material will focus on grids and their use in facilitating 2D spatial search. However, grids can also be used in other applications that are not spatial. In this practice activity, we will take a grid-like approach to sorting lists of strings.

Ordering a list of strings by a single letter

This activity will describe a method for sorting a list of strings into alphabetical order using a grid whose cells correspond to characters in the alphabet. Examine this program that sorts a list of strings. The function `order_by_letter` takes a list of strings and a letter position in those strings and returns the strings ordered alphabetically using the letters in the specified position. For example, if the list was `["cat", "dog", "pig"]` and the position was 1, this function would return the list `["cat", "pig", "dog"]` since the letters at position one (`"a"`, `"i"`, and `"o"`) are now in alphabetical order.

Observe that `order_by_letter` does not rely on comparisons between strings or letters. Instead, the function orders the strings by creating a list called `buckets` that is indexed by the letters from `"a"` to `"z"`. (The letters are converted to numbers using the `ord` function.) Each entry in this list is itself a list of strings that contain the specified character in the specified position. To generate a list of strings in the desired order, each string in the input list is appended to the appropriate entry in the list `buckets` (lines 18-20). The 26 lists in `buckets` are then appended together to form the final list of strings in the desired order.

One important fact to note is that the number of statements executed during the evaluation of `order_by_letter` is linear in the number of strings in the input list. Finding and appending a string from the input list to the appropriate list in buckets requires executing only two statements.

Sorting a list of strings with a fixed number of letters

The function `order_by_letter` can be used to sort a list of fixed length strings. If we called `order_by_letter` using position zero, the resulting strings would be in alphabetical order based on their first letter.   For example, the list `["ape", "bat", "ant"]` would be reordered to form the list `["ape", "ant", "bat"]`. However, note that `"ape"` and `"ant"` are not in alphabetical order.   The trick in this case would be to first order the strings by their second letters to form `["bat", "ant", "ape"]` and then order this list by the first letter to form `["ant", "ape", "bat"]`.

More generally, the function `string_sort` sorts a list of strings by repeatedly applying `order_by_character` to the list of strings working from the last letter to the first. This order method relies on the fact that if two string share the same k initial letters, the call to `order_by_letter` at position k will order these two strings correctly and that relative order is preserved by all subsequent calls to `order_by_letter`.

Created Wed4 Jun 2014 11:26 PM BST

Last Modified Wed 2 Jul 2014 5:38 PM BST