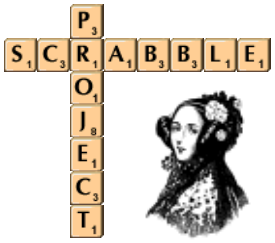
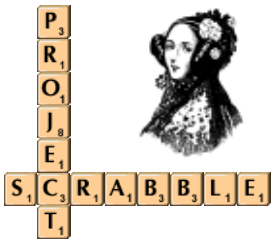


Assignment of the course

Programming I & II

SCRABBLE PROJECT

VERSION 1.0.0, 29.2.2001



Introduction

Searching is a very common, but tricky task in computer science. The algorithms used for searching can have a major impact on performance once the number of elements to be handled gets bigger. The number of elements, the kind of elements, the nature of the collection of elements that must be searched, the internal data structures used for searching, the available memory and the underlying hardware are factors that must be taken into account when writing a search algorithm. In this project, our intention is to show you that finding good searching algorithms (and related sorting algorithms) is of fundamental importance in many cases.

Goal

The goal of the project is to search in a dictionary for words that can be written with a set of given letters. You will have to implement two different search strategies:

1. Search for all words with the maximal length.

2. Search for all words with the maximal weight. The weight of a word is the sum of the weights of all its letters. Each letter has a weight, according to the rules of the English Scrabble game (i.e. ‘z’ has the weight 10, ‘k’ has the weight 5, etc.)

Grading

Grading will be based on correctness, style, and performance. The program you hand in will be tested thoroughly.

A program that correctly finds all longest words (situation 1) and all words with the highest weight (situation 2) for all test cases in a finite amount of time (i.e. 15 minutes maximum) will give you 5 points. If only one situation is handled correctly, then you’ll get 4 points. 1 point will be given based on programming style and presentation of the code (proper indentation, not more than 80 characters per line, understandable variable / constant / operation names, appropriate comments[1]).

The last point will be given based on performance. The performance of the program will be determined by measuring the time spent inside the `Search_Words_Max_Length` / `Search_Words_Max_Weight` procedures (see below). The programs will be tested on the SUN stations in the room CO 021. Measurements will be done using *gprof*. A ranking will be established for each test case.

The number of points (a total of 7 points can be obtained) will determine your grade for this assignment. Each member of the group will receive the same grade.

---

1. Using the option `-gnatg` during compilation will enforce good presentation and correct indentation of the source code.

Groups

4 students together form a group. Groups must be formed and announced until the 30th of March by sending an email to `Xavier.Caron@epfl.ch` with the subject: `Group for scrabble project` or by telling me directly during the course on Friday.

Dictionary

You have to search for the words in the famous English Webster's dictionary. The dictionary is provided in form of a text file that can be found at `~lgl/scrabble/webster.txt`. It contains one single word on each line. No word is composed, and no other characters than 'a' to 'z' or 'A' to 'Z' are used (no accentued letters). Beware that the file is formatted for Unix systems: it means that the characters used for end of lines or end of files may be different from other systems (such as Windows for example).

All test cases will be executed with this dictionary.

Example Test Cases

This section shows some example test cases that you can use to test your implementation. Additional test cases will be used during testing, of course.

1. A search for "firstyear" provides the following results:
   - 11 longest words (7 letters): *terrify - tarrify - strayer - retiary - seriary - astrier - tarsier - fratery - strafer - frasier - fraiser*
   - `3 words with maximal weight (13 points):` *`fratery - tarrify - terrify`*
2. A search for "windowsninetyfive" provides the following results:
   - 1 longest word (13 letters): *nondefinitive* (!!!)
   - `1 word with maximal weight (20 points) :` *`nondefinitive`*
3. A search for "windowsmilleniumedition" provides the following results:
   - 1 longest word (13 letters): *undimensioned* (!!)
   - `1 word with maximal weight (19 points) :` *`disendowment`*

Programming Details

The scrabble program must be written in Ada 95, compilable and executable on the SUN stations of the rooms CO 020,021,023.

The complete program is split into three packages, `Scrabble_Types`, `Scrabble_Utilities` and the main program `Scrabble`. You have to implement the package body of `Scrabble_Utilities`. You are not allowed to modify the other packages, or the specification of the package `Scrabble_Utilities`.

Here is the description of the 6 procedures defined in the specification of the package `Scrabble_Utilities`:

- **`procedure`** `Load;`
  This procedure is responsible for loading the dictionary into the data structures you are using. You are allowed to pre-process the dictionary in this procedure, e.g. sort

the words. The time your algorithm takes to load and process the dictionary will not be measured. Nevertheless, if the pre-processing takes too much time, you are also allowed to store a pre-processed version of the dictionary, and load this pre-processed version in order to increase performance. Of course, you must then also hand-in the pre-processed version of the dictionary.

- `procedure Search_Words_Max_Length (Letters : in String);`
  This procedure must search for the longest words that can be written with the characters provided in the string `Letters`. The results you find must be stored in some data structure. The time spent in this procedure will be measured.
- `procedure Search_Words_Max_Weight (Letters : in String);`
  This procedure searches for all words with maximum weight that can be written with the characters provided in the string `Letters`. The words you find must be stored in some data structure. The time spent in this procedure will be measured.
- `procedure Print_Results;`
  This procedure prints the words found during the last search to the screen. The time spent in this procedure will not be measured.
- `procedure Save_And_CleanUp_Results (File_Name : in String);`
  This procedure must save the results found during the last search into the file called `File_Name`, one word per line, words sorted in alphabetical order. Then, the procedure should clean up all data structures used by the result. The time spent in this procedure will not be measured.
- `procedure Clean_Up;`
  This procedure will be called before the application quits. It allows you to clean up all global data structures (for example free the memory used to store the dictionary). The time spent in this procedure will not be measured.

Here is the complete specification of the package `Scrabble_Utilities`:

```
package Scrabble_Utilities is
    procedure Load;
    procedure Search_Words_Max_Length (Letters : in String);
    procedure Search_Words_Max_Weight (Letters : in String);
    procedure Print_Results;
    procedure Save_And_CleanUp_Results (File_Name : in String);
    procedure Clean_Up;
end Scrabble_Utilities;
```

Here is the specification of the package `Scrabble_Types` that defines the global types and the weights of the characters:

```
with Ada.Unchecked_Deallocation;

package Scrabble_Types is

    --  Definition of the weights (English scrabble) given to each letter.
    subtype Small_Letters is Character range 'a' .. 'z';
    Weights : array (Small_Letters'Range) of Positive :=
       ('q' => 10, 'z' => 10,
       'j' => 8, 'x' => 8,
       'k' => 5,
       'f' => 4, 'h' => 4, 'v' => 4, 'w' => 4, 'y' => 4,
```

```
      'b' => 3, 'c' => 3, 'm' => 3, 'p' => 3,
      'd' => 2, 'g' => 2,
      others => 1);  --  "esarintulo" : 10 letters with 1 point each.

   --  Definition of a reference type for the Ada standard String type.
   type String_Ref is access all String;

   --  This procedure allows to free the memory space pointed by the ref.
   --  Example: if you have My_String of type String_Ref,
   --  call Free (My_String) to free the memory (then My_String is null).
   procedure Free is new Ada.Unchecked_Deallocation (String, String_Ref);

end Scrabble_Types;
```

The file `scrabble.adb` contains the main procedure of the program. You can use it to test your implementation of the searching algorithms. We will probably use a different main program for testing.

What you know for sure is the following:

1. The main program will first call `Load`.
2. Then `Search_Words_Max_Length` or `Search_Words_Max_Weight` is called. The time spent in the procedure is measured.
3. The procedure `Save_And_CleanUp_Results` is called.
4. The procedure `Clean_Up` is called.

Steps 2 and 3 might be repeated multiple times.

Available Files

The files:

- `scrabble_types.ads`
- `scrabble_utilities.ads`
- `scrabble.adb`

can be found in the folder `~lgl/scrabble_project` starting from Friday, 16th of March. Inside the same folder you will also find a file named `webster.txt`. It contains the webster's dictionary used for the project.

Hand-in

You must hand-in the source file `scrabble_utilities.adb`, that contains the implementation of the body of the `Scrabble_Utilities` package. Please don't modify the names of the predefined procedures, nor the provided source files. We will use the original source files when compiling your program for performance measuring.

If you modify the dictionary according to your needs, please don't send the file (or the files) by e-mail, but mention in your e-mail the full path of the directory where we can find the files on the cosuns server. Make sure that the access rights for reading are set correctly for all files and the directory.

Send the file `scrabble_utilities.adb` and any auxiliary files you need for your program to run `Xavier.Caron@epfl.ch` by Friday, June 1st. Submissions after the 1st of

June will receive a penalty of 0.5 points per day. The email should have the subject: `Scrabble project group x`, where x stands for your group number. The body should also contain the compiler flags that we must use to compile your program. If no compiler flags are specified, the default command will be used:

```
gnatmake -pg scrabble -largs /soft/lib/gcc-lib/sparc-sun-solaris2.6/2.7.2.1/gmon.o
```

Please state also who has implemented what part of the project, so that we can check that the work has been distributed among the members of the group.

Prize

The winning group will receive a gift check with a value of 400 SFr. The winning group is determined by comparing the ranks obtained for each test case using the official rules of the International Skating Union.

Note

Small modifications to this project description or the provided source files are possible during the first weeks of the project.