Hangman

# To run

Just make sure that sowpods.txt is in the bin directory and open a command line there. Type “Java hangman” to run.

# Algorighm Pseudocode (Short version)

Initialise dictionary

Initialise mask

Do

Input character

Initialise patterns with character

For word in dictionary

Generate pattern from word

Add word to PatternFamily

Endfor

Set dictionary to max|PatternFamily|

If pattern reveals letter

Add revealed letters to mask

Guesses—

While guesses are not 0 and mask is not full

# Algorithm Pseudocode (long version)

Initialise DataStructure

Do

Input wordLength

While DataStructure<wordLength> does not exist

Set workingDictionary to DataStructure<word length>

Initialise 2^wordLength regexGroups

Initialise wordMask of length wordLength

Do

Input guesses

While guesses is invalid

Initialise guessedChars array of length guesses

Do

Input showWords

While showWords is not ‘yes’ or ‘no’

While guessed is not 0 and game is not won do

If showWords

Print |workingDictionary|

If guessedChars is not empty

Print ‘you have guessed the letters:’

For char c in guessedChars

Print c

endIf

Print ‘You have ’ guesses ‘ guesses remaining, make a guess’

Do

Input letter

While letter is valid and not in guessedChars

Add letter to guessedChars

For x=0 to 2^wordLength

Set stringX to binary string of x

For I=0 to wordLength

If regexMask is empty at i

Initialise pattern with any char except guessedChars at slot i

Else

Initialise pattern with the guessed character at slot i

Endif

Endfor

Add pattern to regexPattern at slot x

Endfor

For word in workingDictionary

For character in word

Initialise binaryString

If character matches guessed character

Add 1 to binaryString

Else

Add 0 to binaryString

endIf

set I to parse integer from binaryString

add word to regexGroup(I)

endfor

endfor

Find max |group| in regexGroup

Set workingDictionary to group

For each pattern item in pattern

If item is not any character

Add item to regexMask

Initialise won to true

For each item in regexMask

If regexMask is any character

Set won to false

Guesses--

End While

# Data Structure

The main data structure used is a Hashmap of ArrayList of words. The first level of HashMap sorts the words into group based on length, where the HashMap key is the size so HashMap<1> has all the 1 letter words. The second level of ArrayLists simply contains all the words, there is no need for alphabetical order as the bucket sort used makes it a stable sort and the words will appear in each group the same order that they were retrieved out of the original dictionary.

Example:

Dictionary = a,b,c,aaa,aba,bac,cab

Key ‘1’, Value=[a,b,c]

Key’2’,Value=Null

Key’3’Value=[aaa,aba,bac,cab]

The complexity of initialising this data structure is O(n) as it takes constant time to insert the words into the structure.

## Initialising algorithm

For each word in dictionary

If HashMap<|word|> doesn’t exist

Make a new ArrayList

Add word to HashMap<|word|>

# Structure of regexGroup

Each group can be shown as a hit for the guessed letter such that if the word is ab and the letter is A, the first letter will hit followed by a miss making the string “10”, parsed to int,2. This will go in the group at slot 2. A character with no hits will have a string of 00 and so will go in the first slot. The structure itself is just an ArrayList of ArrayList of String to contain the words in each family but the fact that each group is directly related to the index is useful in my algorithm.