sort & searching algos
file I/O
matrix class

EECE 2560: Fundamentals of Engineering Algorithms

Department of Electrical and Computer Engineering

Project Word Search

Write a program that solves a word search puzzle. The program reads an n x n grid of letters from a file and prints out all the words that can be found in the grid. Words can be found in the array by starting from any letter and reading left, right, up, down, or along any of the four diagonals. Words can also wrap around the edges of the array. Words must be at least 5 characters long.

The list of k possible words is included in the file `dictionary`. Several sample word search puzzles are also provided.

The goal is to find an algorithm that solves this problem that runs as quickly as possible for large n and k.

(2) print: … ?? wtf **********

Part a

(2) print: pos_x(x) pos+y(y)

1. Implement a class called `dictionary` that reads the words from the dictionary file and stores them in a vector, and which includes:
   (a) a function to read the words from the dictionary file,
   (b) an overloaded output operator to print the word list, << file i/o // external file
   (c) a function that sorts the words using selectionsort,
   (d) a function to handle word lookups using binary search. → index-dict
2. Implement a class called `grid` that reads the letters in the grid from a file and stores them in a matrix.
3. Implement a global function `findMatches()` that is passed the dictionary and the grid as parameters and prints out all candidate words that can be found in the dictionary.
4. Implement a global function `search()` which reads the name of the grid file from the keyboard and prints out all words from the candidate word list that is found in the grid.

findMatches (d, g)

(1) scan 'g' to get all candidate words (brute force). skip len<5
(2) search the 'd' for each candidate word (binary search)
if 'd' is found -> print()

read 2 ints to init matrix size

```
15 15
n y d m k u a s l m o q y r c
u o t e u i t n m o o t w w p
e m r w t u h i d t n r m p h
g s b t d a t q k i r a a y o
d f q e h r c h f v i m u v i
d g n e m e i u b a v s p c l
q t j r q q a w d t p s b a j
s b h y s f u s o e a r e r e
o o r e n m j f t d d n s a p
y e j n a c w j o e k n b w p
v n f a k m k n c c r v r p c
d e t n e l a t r c u n k i q
z s c k q c d c n y l o t g n
s p a q n a w c g s f c i l h
h x p p i z u t w x b g m r a
```

search()
(1) get the input file name and the grid
(2) read grid "input15"
-> read dictionary
-> g
-> d
(3) sort d (set selection sort)
(4) findMatches( )

word dimensions: (5, matrix.size()) -> actually wrap it all