Word Search

EECE 3326 Optimization Methods
Instructor: Ningfang Mi

Word Search Problem

- Given a file containing
  - an n x n grid of letters
  - a dictionary containing a list of K possible words

s	e	z	y	w
k	c	o	d	e
u	z	a	e	a
a	b	z	c	d
t	j	m	k	p

5x5

code
bad
cake
dab
deck
…

- Print out all words found in the grid to the screen "correctness"
- Find an algorithm to solve this problem which runs quickly for large n and K "efficiency"

4. Given starting letter, stop search when we've back to this letter.

Word Searching Rules

Sub prob. get a candidate string from the grid

1. Start from any letter in the grid (n x n starting points)
2. Search in eight directions
   - up, down, left, or right, or in any of the 4 diagonals
3. Wrap around the edges of the grid

- For example:

i/j : 4(n)   i/j = 0

i/j : -1   i/j = 4(n-1)

	0	1	2	3	4
0	s	e	z	y	w
1	k	c	o	d	e
2	u	z	a	e	a
3	a	b	z	c	d
4	t	j	m	k	p

up right

jzees

(i, j)   (4, 1) → (3, 2)

↓

(2, 3)

↓

(1, 4)

↓

← (0, $\cancel{5}$ 0)

i: up i--

down i++

j: left j--

right j++

($\cancel{-1}$ 4, 1)

Brute-force Solution

- Scan the grid looking for all candidate words at ① each possible start positions ② all directions (8) ③ all possible lengths =
- For example:

5x5	R	D	L	U	UR	DR	UL	DL	
①	a ✓								
2	ae	az	az	ao					(8)
3	aea	..	azu						(8)
4									
5	aeanz		azuae						(8)

(n-1) rows

Running Time for Brute Force

- How many starting locations are there?

$n \times n = 25$

- How many candidate strings start at each location?

$(n-1) * 8 + 1 = 4 \times 8 + 1 = 33$

- Total number of candidate strings:

$25 * 33 = 825$

Subprob2: Search a key string in the dict.

Running Time for Brute Force

- Running time for searching each candidate word in the dictionary

  $K = 90{,}000$

  - Worst case: not found

    K comparisons

  - Avg case: $K/2 = 45{,}000$

    - Avg time to compare two strings: 100ns √
    - Avg time to look up one candidate string in dictionary:

      $45{,}000 * 100ns = \underline{4.5 * 10^{-3}\ s}$

    - Time to look up all candidate strings in dictionary:

      $825 \times$ ← $= 3.7\ s$

Analysis of Brute Force Algorithm

- Input size
  - n x n letters in grid
  - K words in **unsorted** dictionary
- Basic operation:

  string comparison

- Total candidate words:

  $$C(n) = n^2(8(n\overset{-}{\cancel{a}}1)+1)$$

- Total comparisons:

  $$(8n^3 - 7n^2) * \frac{k}{2} = O(n^3 k)$$

Analysis of Brute Force Algorithm

- Input size
  - n x n letters in grid
  - K words in sorted dictionary
- Sorting Algorithms
  - ✓ Selection Sorting: $O(k \log k)$
  - Binary Search: $O(\log k) \leftarrow O(k)$ (seq. search)
- Total candidate words: $8n^3 - 7n^2$
- Total comparisons: $(8n^3 - 7n^2) \cdot \log k = O(n^3 \log k)$
- Total run time: $O(n^3 \log k + k \log k)$

Analysis of Brute Force Algorithm

- Input size
  - n x n letters in grid
  - K words in sorted dictionary
- Sorting Algorithms
  - QuickSort :
  - Binary Search:
- Total candidate words:
- Total comparisons:
- Total run time: