Birla Institute of Technology & Science, Pilani
Data Structures & Algorithms (CS F211)
Lab Assignment - 6 (Radix Sort)

Instructions

- All input expressions should be read from stdin and output should be printed on stdout.

- For 1 hour 45 min, only a subset of test cases will be visible to students after submitting the code on the portal. After 1hour 45 min, all test cases will be made visible and they will have last 15 min to correct their code and resubmit.

- At the end of 2 hour period, the online system will stop accepting the submissions.

- Only the last submission by the student before end of lab will be considered for evaluation.

- Following messages by online portal will **tentatively** fetch these marks:

  - Correct $\rightarrow$ 4 marks
  - Wrong-answer (correct for more than half test cases) $\rightarrow$ 3 marks
  - Run-error/Compiler-error/Timelimit-error $\rightarrow$ 2 marks

- All submitted source code will be later checked manually by the instructor and final marks will be awarded, which will be posted on Nalanda after the lab assignment has been done by all lab sections.

- Solution must be implemented using the algorithm and data structures mentioned in the lab sheet only.

Problem Statement

Creating an index of a file using Radix Sort.

Input

You will be given a file (on stdin) with page numbers as input. First line of the file will have "1" in a single line. From second line onwards it will have English sentences. These English sentences will have page number 1. Page 2 will start with "2" written in a single line. Next lines will have English sentences having page number 2. Similarly page n will start with "n" written in a single line followed by English sentences from the next line.

Output

You will have to output the index of the file created as follows:

1. Collect all English words with their page numbers from the file.

2. Sort the words in lexicographical (dictionary) order using Radix Sort.

3. For each unique word, create an entry in one line in the index as:

$$word \quad p_1, p_2, ..., p_m$$

where $word$ is the unique English word. $word$ is followed by a blank space. Then we have unique page numbers $p_1, p_2, ..., p_m$ in which $word$ appears. The page numbers are separated by comma and blank. The $word$ will be in all small letters even if the original word has capital letters.

Procedure

We use Radix-27 sort. The letters will have their usual values: $a = 1$, $b = 2$, ..., $y = 25$, $z = 26$. Null characters will have value 0. For example, if we have three words: a, abc, and ab. Then we write each letter as a 3-letter word by appending null character with value 0 so that all letters are of the same length. The letters will now have the values (in Radix-27):

- $a = 100$

- $abc = 123$

- $ab = 120$

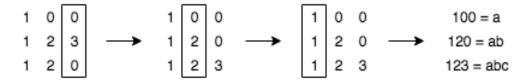Now starting from rightmost digit (in Radix 27) apply counting sort:

Figure 1: Radix Sort

Now the words are sorted in dictionary order. Counting sort is a linear time stable sorting algorithm which is used as a subroutine in Radix Sort to sort the individual digits. In the code for counting sort, we assume that the input is an array $A[1 \ldots n]$, and thus $A.length = n$. We require two other arrays: the array $B[1 \ldots n]$ holds the sorted output, and the array $C[1 \ldots n]$ provides temporary working storage. ($0 \le A[i] \le k$ for $i \le i \le n$).

Algorithm 1 Counting-Sort(A, B, k)

1: $C[0 \ldots k] \leftarrow$ new array
2: **for** $i = 0$ to k **do**
3:    $C[i] \leftarrow 0$
4: **end for**
5: **for** $j = 1$ to $A.length$ **do**
6:    $C[A[j]] \leftarrow C[A[j]] + 1$
7: **end for**{C[i] now contains the number of elements equal to i.}
8: **for** $i = 1$ to k **do**
9:    $C[i] \leftarrow C[i] + C[i-1]$
10: **end for**{C[i] now contains the number of elements less than or equal to i.}
11: **for** $j = A.length$ to 1 **do**
12:    $B[C[A[j]]] \leftarrow A[j]$
13:    $C[A[j]] = C[A[j]] - 1$
14: **end for**

Test Cases

Case 1: Simple Text

Input

```
1  1
2  This is page 1. It is having one paragraph.
3  2
4  This is page 2. It is having one sentence.
5  3
6  This is page 3. Now the index will start.
```

Output

```
1   having 1, 2
2   index 3
3   is 1, 2, 3
4   it 1, 2
5   now 3
6   one 1, 2
7   page 1, 2, 3
8   paragraph 1
9   sentence 2
10  start 3
11  the 3
12  this 1, 2, 3
13  will 3
```

Case 2: Peter-Piper Tongue Twister

Input

```
1  1
2  Peter Piper picked a peck of pickled peppers.
3  2
4  A peck of pickled peppers Peter Piper picked.
5  3
6  If Peter Piper picked a peck of pickled peppers
7  4
8  Wheres the peck of pickled peppers Peter Piper picked.
```

Output

```
1   a 1, 2, 3
2   if 3
3   of 1, 2, 3, 4
4   peck 1, 2, 3, 4
5   peppers 1, 2, 3, 4
6   peter 1, 2, 3, 4
7   picked 1, 2, 3, 4
8   pickled 1, 2, 3, 4
9   piper 1, 2, 3, 4
10  the 4
11  wheres 4
```

Case 3: Chuck-Woods Tongue Twister

```
1  1
2  How much wood could Chuck Woods woodchuck chuck
3  2
4  if Chuck Woods woodchuck could and would chuck wood.
5  3
6  If Chuck Woods woodchuck could and would chuck wood
7  4
8  how much wood could and would Chuck Woods woodchuck chuck.
9  5
10 Chuck Woods woodchuck would chuck he would as much as he could
11 6
12 and chuck as much wood as any woodchuck would
13 7
14 if a woodchuck could and would chuck wood.
```

Case 4: The Mikado Tongue Twister

```
1  1
2  To sit in solemn silence in a dull dark dock
3  In a pestilential prison with a lifelong lock
4  2
5  Awaiting the sensation of a short sharp shock
6  From a cheap and chippy chopper on a big black block
7  3
8  To sit in solemn silence in a dull dark dock
9  In a pestilential prison with a lifelong lock
10 4
11 Awaiting the sensation of a short sharp shock
12 From a cheap and chippy chopper on a big black block
13 5
14 A dull dark dock a lifelong lock
15 A short sharp shock a big black block
16 To sit in solemn silence in a pestilential prison
17 12
18 And awaiting the sensation
19 From a cheap and chippy chopper on a big black block
20 by W S Gilbert of Gilbert and Sullivan from The Mikado
```

Case 5: Plagiarized Text

<div align="center">Input</div>

```
1  1
2  Notice the matching text in Page 2 with respect to all other pages. This code can
      be used as a Plagiarism detector.
3  2
4  A type B is a subtype of A if every function that can be invoked on an object of
      type A can also be invoked on an object of type B. Subtyping does not only
      mean that operations on the supertype can be performed on the subtype. Subtype
      is also a type which is related other to another type called parent type or
      supertype. The subtype must satisfy the features values methods and properties
      of the parent type.
5  3
6  A type B is a subtype of A if every function that can be invoked on an object of
      type A can also be invoked on an object of type B.
7  4
8  Subtyping merely means that operations on the supertype can be performed on the
      subtype.
9  5
10 Note that subclassing is a special case of subtyping.
11 6
12 Subtype is also a type which has a relation other to another type called parent
      type or supertype.
13 7
14 The subtype must satisfy the features values methods and properties of the parent
       type.
```

Case 6: Long DNA Sequence Matching

<div align="center">Input</div>

```
1  1
2  aaaaactttt   aggaaaaacg   tacgtcaagg   gatgtcacta
3  2
4  aaatcgttct   aaaaactttt   tgctttttcc   gatgtcacta
5  3
6  tttcaatttc   aaaaactttt   gatgtcacta   tgctttttcc
7  4
8  tgctttttcc   gatgtcacta   aaaaactttt   aggaaaaacg
9  5
10 aagaccatga   gatgtcacta   ttccaatcat   gaagtgccca
11 6
12 tgctttttcc   gtaggtatgg   gaagtgccca   catctccaca
13 7
14 aattggataa   gatgtcacta   aaaaactttt   tgctttttcc
15 8
16 acaattactt   gatgtcacta   aaaaactttt   tttcctgctt
17 9
18 acactgcaaa   aaaaactttt   gatgtcacta   acaagctaca
19 10
20 aggaaaaacg   aaaaactttt   acataccaat   gatgtcacta
21 11
22 aaaaactttt   acaagctaca   agtataccct   gatgtcacta
23 12
24 acgaactgcg   aggaaaaacg   tgactggtaa   gaagtgccca
25 13
26 acgctctgaa   acaagctaca   aatgaagtct   tcaacgtgac
27 14
28 actcggcgat   tgctccagaa   acaagctaca   gaagtgccca
```