Homework: Implement bag-of-tasks paradigm

DUE: See exact time on blackboard

Bag-of-Tasks Paradigm

A parallel computing method in which tasks are placed in a bag shared by worker processes. Each worker repeatedly takes a task, executes it, and possibly generates new tasks that it puts into the bag. The computation has terminated when the bag is empty and the workers are idle.

Problem

Write a parallel program that uses the bag-of-tasks paradigm to solve the following problem. Use C and Pthreads. Run the timing tests described below, and turn in a table of results as well as your program listing.

There is an online dictionary of words posted on blackboard. Your task is to find all words in the dictionary that have unique letters—i.e., words in which no letter occurs more than once. I suggest that you first write a sequential program and then modify it to use the bag-of-tasks paradigm. Your sequential program should have the following phases:

- Read the file words into an array of strings. You may assume that each word is at most 25 characters long. Declare an integer array of the same length and initialize it to zeroes; this array will be used to record the lengths of words that contain unique letters.
- Examine each word, one at a time. If it has unique letters, then set the corresponding entry in the second array to the length of the word.
- After you have examined all words, go back through the array of word lengths. Count the total number of words with unique letters, and write those that have at least six unique letters to a file named *results*. At the end of the program, write the total count to standard out. (While debugging, you might want to write all words with unique letters to the *results* file.)

The first few words in the dictionary start with numbers (take a look!). Skip over these and start with "a", the first word that begins with "a". Some words are proper nouns and hence start with capital letters. Also skip over these. In short, you are to consider only those words that consist of lower-case letters. After you have a working sequential program, modify it to use the bag-of-tasks paradigm.

Your parallel program should use W worker processes, where W is a command-line argument. Use the workers just for the compute phase; do the input and output phases sequentially. Each worker should count the number of words with unique letters that it finds. Sum these W values during the output phase. (This avoids a critical section during the compute phase!) Use 26 tasks in your parallel program, one for each letter of the alphabet. In particular, the first task is to examine all words that begin with "a", the second task is to examine all words that begin with "b", and so on.

During the input phase you should build an efficient representation for the bag of tasks; I suggest using an array, where the value in task[0] is the index of the first "a" word, task[1] is the index of the first "b" word, and so on. Your parallel program should also time the compute phase. Do not time the sequential input and output phases. Write the elapsed time for the compute phase to the standard output. To summarize, your parallel program should have the following output: the total number of words with unique letters, the number found by each worker, the elapsed time for the compute phase, and the words that contain at least 6 unique letters. Write the first three items to standard out; write the words to a file named results in the directory that contains your program.

Results

Create a table with multiple runs of your serial and parallel program. You may switch the number of threads.

Serial execution time (microseconds)	Number of threads	Parallel execution time (microseconds)	Speedup
<add more rows as needed>			