CS140 - Assignment 7

Due: Sunday, Oct. 30 at 11:59pm

THE IMPORTANT THING IN COMPUTER SCIENCE IS BEING GOOD WITH *DETAILS*. *DETAILS* SEPARATE THE DONALD KNUTHS FROM THE NOBODIES.
YOU'RE NOT WEARING PANTS.
I SAID DETAILS! NOT BIG PICTURE STUFF!

http://www.smbc-comics.com/index.php?db=comics&id=2217

For this assignment, you may (and are encouraged to) work with a partner.

1. **[12 points]** You are given a string of characters $S = s_1, s_2, ..., s_n$ where all non-alphabetic characters have been removed (e.g. "thisisasentencewithoutanyspacesorpunctuation") and a function DICT(w, i, j), which takes as input a string w and two indices i and j and returns $true$ if the string $w_{i...j}$ is a dictionary word and $false$ otherwise.

   (a) **[10 points]** Give a dynamic programming solution that determines whether the string S consists of a sequence of valid dictionary words. Make sure to explicitly state the size of the table, which elements you fill in first, how you fill in the table, and where the solution is found.

   (b) **[2 points]** State the running time of your algorithm assuming calls to DICT are $O(1)$.

Greedy or DP

One of the problems below can be solved more efficiently using a greedy approach and the other cannot (i.e. you must use dynamic programming). For each problem clearly describe your algorithm and state the run-time. For the dynamic programming problem, make sure to describe the table and how it is filled in. For the greedy problem, prove that your solution is optimal.

2. **[10 points]** You're going on a road trip with friends. Unfortunately, your headlights are broken, so you can only drive in the daytime. Therefore, on any given day you can drive no more than d miles. You have a map with n different hotels and the distances from your start point to each hotel $x_1 < x_2 < ... < x_n$. Your final destination is the last hotel. Describe an algorithm that determines which hotels you should stay in if you want to minimize the number of days it takes you to get to your destination.

3. **[10 points]** Same setup as above, however, you also want to do some sightseeing along the way. To make sure you don't spend too little or too much time in any one place, you decide to add a penalty for having too much free time. If you travel x miles in a day, then the penalty for that day is $(d - x)^2$. Describe an algorithm that determines the hotel sequence that minimizes the total penalty, that is the sum of the daily penalties over all travel days.

DP in code

You decide that you looove latex so much that you're going to try and write your own version of the program. As a first start, you decide to write a program that will take a string of text as input and formats the text so that it is left-justified (all lines begin at the same left column) and the right margin is as even as possible (where "even as possible" is defined below).

For example, given the text:

```
Call me Ishmael.  Some
years ago, never mind how long precisely,
having little
or
no money in my purse, and nothing
particular to interest me on shore, I thought I
would sail
about a little and see the watery part of the
world.
```

Your system might output something like:

```
Call me Ishmael.  Some years ago, never
mind how long precisely, having little
or no money in my purse, and nothing
particular to interest me on shore, I
thought I would sail about a little
and see the watery part of the world.
```

You formulate this problem as follows. You are given a sequence of words $S = (w_1, \ldots, w_n)$ where w_i consists of $len(w_i)$ characters where each character is of the same width (a so-called "fixed-width" font such as Courier). (Punctuation symbols are considered to be regular characters.)

The words must be placed in the order in which they appear in the sequence, S. The maximum length of a line is L. That is, a line can accommodate up to L symbols, including the space that goes after each word on a line. (Of course, there does not necessarily need to be a space added after the last word placed on a line.)

If words $w_i, \ldots, w_j$ are placed on a line then the total length placed on that line, length(i, j), is:

$$\text{length}(i,j) = \left(\sum_{k=i}^{j-1} (\text{len}(w_k) + 1) \right) + \text{len}(w_j)$$

Notice that this accounts for the length of each of the words and the space immediately after the word. The last word is treated separately since it does not need a space afterwards.

Recall that length(i, j) must be at most L. The *slack* of that line is defined as $(L - \text{length}(i, j))$.

Your goal is to determine how to break the words in S sequentially on lines so as to *minimize the sum of the cubes of the slacks*, known as the *penalty* of the packing.

4. **[20 points]** Write a program that given a string, determines and outputs the best line breaking scheme given the specification above.

   Specification:

   - Your program should accept L and the name of a file as input.

- Your program should print out the overall penalty followed by the actually formatted text.
- To determine the "words", just split up the string based on spaces.
- You may assume that no word is longer than L.
- You may use any programming language that you'd like.

Submission:

- Submit the output of your approach on the following four test cases below. Make sure to include the overall penalty and be carful when you hand-in your results to make sure that latex (or your word-processor) isn't splitting any lines that you didn't intend to have split.
- In addition, as an appendix, include your code in your writeup. You can either include it in latex (verbatim mode made be helpful), of you can also just make a pdf of your code and add it to the end of your writeup. Your submission should still just be a single pdf file.

(a) $L = 10$

```
This is a sentence.
```

(b) $L = 50$

```
Call me Ishmael.  Some years ago, never mind how long precisely,
having little or no money in my purse, and nothing
particular to interest me on shore, I thought I
would sail about a little and see the watery part of the world.
```

(Note, your program should treat this as all one string.)

(c) $L = 20$

Same text as previous (i.e. the Ishmael text).

(d) $L = 80$

```
It was the best of times, it was the worst of times, it was the age of
wisdom, it was the age of foolishness, it was the epoch of belief,
it was the epoch of incredulity, it was the season of Light, it
was the season of Darkness, it was the spring of hope, it was the
winter of despair, we had everything before us, we
had nothing before us, we were all going direct to
heaven, we were all going direct the other way - in short, the
period was so far like the present period, that some of its noisiest
authorities insisted on its being received, for good or for evil,
in the superlative degree of comparison only.
```

I highly recommend writing some test cases and checking them by hand before submitting your results!