CS 341: Algorithms @ Waterloo **Last Updated:** June 26, 2017
Homework 4 **Due Date: July 10, 2017, 11:00pm**

You are allowed to discuss with others but are not allowed to use any references other than the course notes. Please list your collaborators for each question. You must write your own solutions.

There are totally 60 marks. The full mark is 50. This homework is counted 5% of the course grade.

---

1. **Programming Problem: Text Formatting (20 marks)**

   Given a sequence of n words (possibly with punctuations), you are asked to put line breaks between the words such that the text is left-justified and there are at most w characters per line. We want the lines to be tidy, i.e. not having lines with lots of trailing spaces. We define the untidiness of a line as the square of the number of trailing spaces at the end of the line, i.e. if there are s trailing spaces, then the untidiness of that line is s^2. The untidiness of the whole text is the sum of untidiness of all lines except the last. Your task is to format the text to achieve minimum untidiness. Note that you can put only one space between the words and the text have to be left-justified.

   INPUT: The first line contains two integers n, w. Then the next n lines contain the sequence of the words. The words are strings consist of at most 20 characters from [A-Z], [a-z], [0-9], and punctuations with no space.

   It is guaranteed that $1 \leq n \leq 1,000,000$, $1 \leq w \leq 80$ and w is greater than the maximum length of all the words.

   OUTPUT: The first line of the output should contain two integers U and m, the minimum untidiness you can achieve and the number of lines in your format. Then you should print your text in the next m lines. If there are multiple formatting that achieve the minimum untidiness, you can output any one of them.

   SAMPLE INPUT:

```
12 20
Dynamic
programming
is
fun.
It
can
be
used
to
solve
practical
problems.
```

Sample Output:

```
53 4
Dynamic programming
is fun. It can
be used to solve
practical problems.
```

Explanation: The first line has $7 + 1 + 11 = 19$ characters, so the number of trailing spaces at the end of the line is $20 - 19 = 1$. The number of extra space at the end of each line is $1, 6, 4, 1$, and the total untidiness if $1^2 + 6^2 + 4^2 = 53$ as the last line does not count. (If dynamic programming is very fun, then the text will be even tidier.)

2. **Written Problem: Envelopes** (10 marks)

   Given n rectangular envelopes $E_1, \ldots, E_n$ with width $w_1, \ldots, w_n$ and height $h_1, \ldots, h_n$, an envelope E_i encloses another envelope E_j if $w_i \geq w_j$ and $h_i \geq h_j$ or $w_i \geq h_j$ and $h_i \geq w_j$ (i.e. we can rotate an envelope). A subsequence of envelops $1 \leq i_1 < \ldots < i_k \leq n$ is called an enclosing subsequence if E_{i_l} contains $E_{i_{l+1}}$ for all $1 \leq l < k$. Design an efficient algorithm to find a longest enclosing subsequence of the envelopes. Prove the correctness and analyze the time complexity.

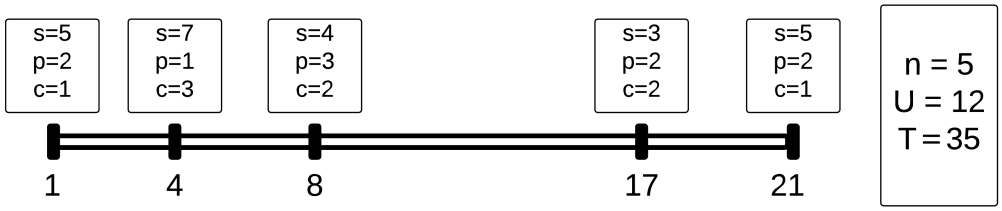
3. **Written Problem: Fuel The Tank, Take 2** (10 marks)

   Let's consider a more realistic version of the famous "Fuel The Tank" problem. There are n cities named with number $1, \ldots, n$. These cities are located from left to right on the x-axis. City i locates at coordinates x_i and it is guaranteed that $x_i < x_{i+1}$ for $1 \leq i < n$. One day you want to travel from city 1 to city n. There are $n - 1$ one way roads, where the i-th road goes from city i to city $i + 1$, and the length of the road is $(x_{i+1} - x_i)$ kilometers long.

   Your car travels 1 kilometer and consumes 1 litre of fuel per hour. Your plan is to arrive city n within T hours.

   There is a fuel station in each city i. With the price of p_i dollars, you can transfer s_i litres of fuel to your fuel tank in c_i hours. Initially your fuel tank is empty with total capacity U. You can fuel multiple times in a fuel station, but if it exceeds the total capacity, you still need to pay p_i dollars for the last time.

   Design an efficient algorithm to find the minimum cost for you to travel from city 1 to city n within T hours. Your algorithm should return the full schedule of where to fuel the tank and by how much. Prove the correctness and analyze the time complexity. The time complexity could depend on n, T, and U.



Explanation:

Time	Currently city	Current cost	Remaining fuel	Action
$t = 0$	City 1	\$0	0 L	None
$t = 1$	City 1	\$2	5 L	Fill the tank in city 1 one time
$t = 4$	City 2	\$2	2 L	Move to city 2
$t = 10$	City 2	\$4	12 L	Fill the tank in city 2 two times
$t = 14$	City 3	\$4	8 L	Move to city 3
$t = 16$	City 3	\$7	12 L	Fill the tank in city 3 one time
$t = 27$	City 4	\$7	1 L	Move to city 4
$t = 29$	City 4	\$9	4 L	Fill the tank in city 4 one time
$t = 31$	City 5	\$9	0 L	Move to city 5

4. **Written Problem: Covering Set** (10 marks)

   Given an undirected graph $G = (V, E)$, a subset $S \subseteq V$ is a covering set if for every vertex $v \in V - S$ there exists $u \in S$ such that $uv \in E$ (i.e. every vertex $v \in V - S$ has a neighbor in S). We are interested in finding a covering set of minimum total weight, but this problem is NP-hard in general graphs. Show that this problem is easy in trees.

   Input: A tree $T = (V, E)$ in the adjacency list representation, a weight w_v for each vertex $v \in V$.

   Output: A covering set $S \subseteq V$ that minimizes the total weight $\sum_{v \in S} w_v$.

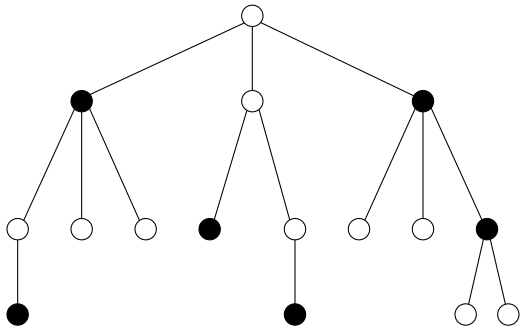


Figure 1: The black vertices form a minimum covering set assuming all weights are one.

   Design an efficient algorithm to solve this problem. Prove the correctness and analyze the time complexity.

5. **Written Problem: Trading** (10 marks)

   Suppose somehow we could use 1 unit of metal to trade for 2 units of wood, and 1 unit of wood to trade for 0.4 unit of glass, and 1 unit of glass to trade for 1.5 units of metal. Then we can start with 1 unit of metal, and end up with $1 \times 2 \times 0.4 \times 1.5 = 1.2$ units of metal, and make profit by just trading. Help your company to solve the following problem.

   Input: n items $a_1, \ldots, a_n$ and an $n \times n$ table T of trading rates, such that one unit of item a_i can trade for $T[i, j]$ units of item a_j, where $T[i, j] \geq 0$ for $1 \leq i, j \leq n$.

   Output: Whether or not there exists a sequence of items $a_{i_1}, \ldots, a_{i_k}$ such that $T[i_1, i_2] \cdot T[i_2, i_3] \cdot \cdots \cdot T[i_{k-1}, i_k] \cdot T[i_k, i_1] > 1$.

   Design an efficient algorithm to solve this problem. Prove the correctness and analyze the time complexity.