# Lab05-DynamicProgramming

CS214-Algorithm and Complexity, Xiaofeng Gao, Spring 2020.

* If there is any problem, please contact TA Shuodian Yu.

* Name:Haotian Xue   Student ID:518021910506   Email: xavihart@sjtu.edu.cn

1. **Bookshelf:** Tim has $n$ books and he wants to make a bookshelf to them. The pages' width of the $i$-th book is $w_i$ and the thickness is $t_i$.

   Tim puts the books on the bookshelf in the following way. He selects some books and put them vertically. Then the rest of the books are put horizontally above the vertical books. Obviously, the total thickness of the books put vertically must be greater than the sum of widths of the horizontal books. As long as tim wants to make the bookshelf as small as possible, please help him to find the minimum total thickness of the vertical books.

   To simplify the problem, the thickness of each book is either 1 or 2. And all the numbers in this problem are positive integers.

   Design an algorithm based on dynamic programming and implement it in C/C++/Python. The file `Data-P1.txt` is a test case, where the first line contains an integer $n$. Each of the next $n$ lines contains two integers $t_i$ and $w_i$ denoting two attributes of the $i$-th book. Source code should be named as *Code-P1.** .You need to briefly describe your algorithm and find the result of `Data-P1.txt` by your program.

   **Example:**

   Given $n = 5$ books, and $\{(t_i, w_i)|1 \leq i \leq 5\} = \{(1, 12), (1, 3), (2, 15), (2, 5), (2, 1)\}$. The algorithm should return 5.

   **Solution.** We can define the $OPT(i, j)$ which is equal to the minimal horizontal length of the upper book we can get when we arrange the first $ith$ books and the vertical books has a total lthickness of $j$. If we cannot arrange the $ith$ books with the vertical thickness equal to $j$, $OPT(i, j) = +\infty$.

   So we can easily get the optimal structure:

   - We select the $ith$ book to be vertical: then $OPT(i, j) = OPT(i - 1, j - t[i])$.
   - We select the $ith$ book to be horizontal: the $OPT(i, j) = OPT(i - 1, j)$

   So the final recurssion will be:

$$OPT(i,j) = \begin{cases} +\infty & ij = 0 \ and \ i + j \neq 0 \\ 0 & i = j = 0 \\ min(OPT(i-1, j-t[i]), OPT(i-1, j)) & otherwise \end{cases}$$

□

   Also we should have the upper length no greater than the vertical thickness, so we can just tranverse $OPT(n, j)$ to find the minimal $j$ to make $OPT(n, j) <= j$.

   Answer: the output of Code-P1.py is 2542.

2. Recall the *String Similarity* problem in class, in which we calculate the edit distance between two strings in a sequence alignment manner.

   You are to find the lowest aligning cost between 2 DNA sequences, in which the cost matrix is defined as

|   | - | A | T | G | C |
|---|---|---|---|---|---|
| - | 0 | 1 | 2 | 1 | 3 |
| A | 1 | 0 | 1 | 5 | 1 |
| T | 2 | 1 | 0 | 9 | 1 |
| G | 1 | 5 | 9 | 0 | 1 |
| C | 3 | 1 | 1 | 1 | 0 |

where `(-, A)` means adding (or removing) one `A`, etc.

(a) Implement Hirschberg's algorithm with C/C++/Python. Please attach your source code named as *Code-P2.**. Your program will be tested against random inputs. Your program should be able to output two sequences after editing.

(b) Using your program, find the edit distance between the two DNA sequences found in attachments `Data-P2a.txt` and `Data-P2b.txt`.

**Solution**

(a) The outputs are in "./result/resulta.txt" and "./result/resultb.txt" respectively.

(b) Penalty in optimal condition : 7615

* To run Code-P2.cpp, you should put the "Data-P1a.txt" and "Data-P2b.txt" in the dir of code file. "./result2" is used to save the modified DNA. After compling Code-P2.cpp and run the execution file, you can get the minimal penalty, also you can use test.cpp to calculate the penalty of two DNA strings in "./result2".

**Remark:** You need to include your .pdf and .tex and 2 source code files in your uploaded .rar or .zip file.