Lab05-DynamicProgramming

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

∗ If there is any problem, please contact TA Shuodian Yu.
∗ Name:Hongjie Fang Student ID:518030910150 Email: galaxies@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 $\{(t_i, w_i)|1 \le i \le 5\} = \{(1, 12), (1, 3), (2, 15), (2, 5), (2, 1)\}$. The algorithm should return 5.

   Solution. Let $f(i, j)$ denote the minimum total thickness of the vertical books when we only consider about the first i books (book 1 to book i) and we can still put some horizontal books of total width j (When $j < 0$ it means we need to add horizontal books of total width j in the future). We assume that if the feasible solution does not exist under a certain previous restriction i and j, then $f(i, j) = +\infty$. Notice that the thickness of each book does not exceed 2, so the total thickness cannot exceed $2n$ in any time. Therefore, we have $-2n \le j \le 2n$. Hence, the **optimal substructure** is:

   - If OPT puts book i in the vertical position, then
     - Book i contributes t_i to the total thickness;
     - Argument j must satisfy $j - t_i \ge -2n$ owing to the previous restriction to j.
     - OPT must includes the optimal solution when we only consider about the first $(i-1)$ books and we can still put some horizontal books of total width $(j - t_i)$.
   - If OPT puts book i in the vertical position, then:
     - Book i does not contribute to the total thickness;
     - Argument j must satisfy $j + w_i \le 2n$ owing to the previous restriction to j;
     - OPT must includes the optimal solution when we only consider about the first $(i-1)$ books and we can still put some horizontal books of total width $(j + w_i)$.

   Therefore, the **recurrence** is as follows:

$$f(i,j) = \begin{cases} 0 & (i = 0, j = 0) \\ f(i-1, j+w_i) & (i \ge 1, j - t_i < -2n, j + w_i \le 2n) \\ f(i-1, j-t_i) + t_i & (i \ge 1, j - t_i \ge -2n, j + w_i > 2n) \\ \min(f(i-1, j-t_i) + t_i, f(i-1, j+w_i)) & (i \ge 1, j - t_i \ge -2n, j + w_i \le 2n) \\ +\infty & (otherwise) \end{cases}$$

Thus, the minimum total thickness of the vertical books under the restriction above is

$$\min_{j\in[0,2n]}\{f(n,j)\}$$

Here is the pseudo-code of the algorithm (Alg. 1).

Algorithm 1: Bookshelf Problem - Dynamic Programming Solution

Input: Total number of books n; two attributes t_i, w_i of the i-th book.
Output: Minimum total thickness of the vertical books under the restriction above.

```
for i = 0 to n do
    for j = −2n to 2n do
        f[0, j] ← +∞;
f[0, 0] ← 0;
for i = 1 to n do
    for j = −2n to 2n do
        if j − t_i ≥ −2n then
            f[i, j] ← min(f[i, j], f[i − 1, j − t_i] + t_i);
        if j + w_i ≤ 2n then
            f[i, j] ← min(f[i, j], f[i − 1, j + w_i]);
ans ← +∞;
for j = 0 to 2n do
    ans ← min(ans, f[n, j]);
return ans;
```

The C++ implementation of the algorithm is in the Code-P1.cpp in the code directory.

Run the program and we can get the result of Data-P1.txt is 2542, that is, the minimum total thickness of the vertical books in Data-P1.txt 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. The C++ implementation of the Hirschberg's algorithm is in the Code-P2.cpp in the code directory. Here are some explanations.

   - **(Small Data)** If you want to test program on small data using command prompt and manual input, then you can uncomment `#define TEST_SMALL_DATA` in line 101, compile and execute the program to test manually.
   - **(Big Data)** If you want to test program on big data using file input, then you can comment `#define TEST_SMALL_DATA` in line 101, change line 109 and line 112 to the correct file directory, compile and execute the program to test automatically. **The output of the program will be saved as a file named `result.res` in the current directory. You can open it and check the answer.**
   - **(Output Cost)** If you want to output the minimum cost, then you can uncomment `#define OUTPUT_COST` in line 157, compile and execute the program to get the minimum cost in output.
   - **(Output Answer)** If you want to output the answer (How to modify the sequences), you can uncomment `#define OUTPUT_ANSWER` in line 158, compile and execute the program to get the modification sequences. **The program will output two aligned sequences using "_" (the underline notation) to represent insertion of the sequences. Matching these two sequences in this way can minimize the total cost.**

   We use the **Big Data** mode mentioned above and output both cost and answer to get the edit distance between `Data-P2a.txt` and `Data-P2b.txt`. The result is in the result.res in the code directory. The minimum cost to aligning these two sequences is 7615, and the aligning method is in the output file of the program. □

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