CS4040/5040 Programming Project

Due on Apr 23 (Mon) by 9:00am

1 Optimal Self Matching

Let $X = \langle x_1, x_2, ..., x_n \rangle$ be a sequence of n symbols drawn from a finite alphabet Σ. We call a pair of indeces (i, j) a *matching pair* if $i < j$ and $x_i = x_j$. We say that a pair (i, j) *crosses* another pair (k, l) if $i \leq k \leq j \leq l$ or $k \leq i \leq l \leq j$. We call a set S of matching pairs over X a *self matching* of X if there are no two pairs in S that cross each other.

For example, the set $S = \{(1, 6), (2, 3)\}$ is a self matching of the sequence $X = \langle a\ b\ b\ c\ b\ a\ c \rangle$, where the alphabet is $\Sigma = \{a, b, c\}$.

Describe an efficient dynamic programming algorithm that finds the largest self matching of an input sequence X. Analyze its time and space complexity.

Bonus points: Same problem, but add the constraint that all matching pairs (i, j) satisfy $j > i + 1$ i.e., all matching pairs have a gap of one or more.

1.1 Implementation

Implement the algorithm in any of the following languages: C, C++, Java, Ada, R, Scheme, Python, Ruby. The input sequence should be read from an input text file (all symbols should be on the same line in the file, separated by white spaces). The input file should be given as a command line argument. The optimal self matching should be output on the screen.

2 Optimal Skip Chain

Let $X = \langle x_1, x_2, ..., x_n \rangle$ be a sequence of n positive integers. A sequence of indeces $S = \langle i_1, i_2, ..., i_k \rangle$ is called a *skip chain* if $i_{j+1} > i_j + 1$, for $1 \leq j < k$ i.e., there is a gap of one or more between any two consecutive positions in the subsequence. The sum of a skip chain $S = \langle i_1, i_2, ..., i_k \rangle$ is defined as $x_{i_1} + x_{i_2} + ... + x_{i_k}$.

For example, $S = \langle 2\ 4\ 7 \rangle$ is a skip chain of $X = \langle 1\ 9\ 5\ 6\ 6\ 1\ 8 \rangle$, with the sum $x_2 + x_4 + x_7 = 9 + 6 + 8 = 23$.

Describe an efficient dynamic programming algorithm that finds the skip chain of an input sequence X that has the maximum sum. Analyze its time and space complexity.

2.1 Implementation

Implement the algorithm in any of the following languages: C, C++, Java, Ada, R, Scheme, Python, Ruby, PHP. The input sequence should be read from an input text file (all numbers should be on the same line in the file, separated by white spaces). The input file should be given as a command line argument. The sequence of indeces in the optimal skip chain should be output on the screen on one line, followed by the sum on the next line.

3 Software Package and Project Report

Place all the files in one directory, and name it '$\langle lastname \rangle$_$\langle firstname \rangle$'. Use the standard software engineering principles to design your program. Make sure the program is well documented. Include a header comment for each file in C/C++ (class in Java) and each function in C/C++ (method in Java). Header comments should describe the input and output to the procedure, the preconditions and postconditions of the procedure and other pertinent information. For more information, see the guidelines at:
http://ace.cs.ohiou.edu/~chelberg/classes/361/style-guide.html

Your program should compile and run correctly on the departmental Unix prime machines. Include a README file in which you describe how to compile and execute your program.

Write a project report in which you describe in detail your dynamic programming solution (the four steps discussed in class). Describe the algorithm in pseudocode for steps 3 and 4. Analyze its time and space complexity. Include a PDF or PostScript version of your report in the submission and also submit a hardcopy of the report in class.

4 Submission

Create a gzipped, tar ball archive of your program, and upload it on Blackboard under the content Project section. Do not submit binary files. Late submissions will not be accepted.

For example, if the name is John Williams, creating the archive can be done using the following commands:

```
> tar cvf williams_john.tar williams_john

> gzip williams_john.tar
```

These two steps will create the file 'williams_john.tar.gz' that you can upload on Blackboard.**When submitting archived packages on Blackboard, make sure they contain the required files**. One way to verify your submission is to download the package that you uploaded on BB, unzip and untar the package on one of the prime machines and check that it contains the required files (especially the source files). Packages that do not contain the required files will receive zero points.