ECE368 Programming Assignment #5

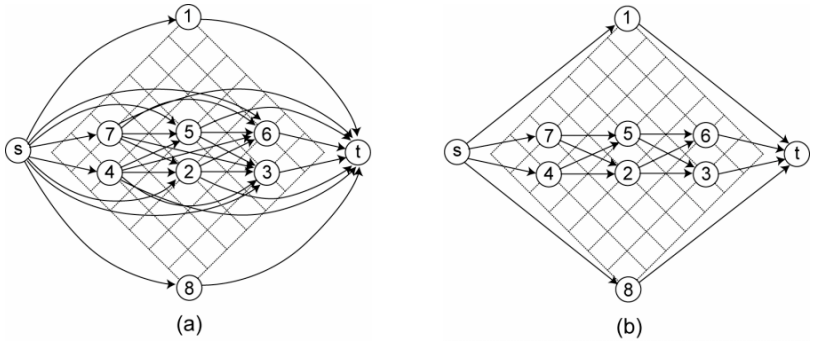
Due Tuesday, April 10, 2018, 11:59pm

Description

This programming assignment is to be completed on your own. You will implement a program involving perhaps "graph traverals" to compute the "packing" of rectangles, represented by a sequence pair $SP(S_1, S_2)$. Suppose the n rectangles are labeled 1 through n, a sequence is a permutation of the n integers. Consider a problem instance of 8 rectangles, $(1,7,4,5,2,6,3,8)$ is a sequence, and $(8,4,7,2,5,3,6,1)$ is also a sequence. Together, they form a sequence pair $SP((1,7,4,5,2,6,3,8),(8,4,7,2,5,3,6,1))$.

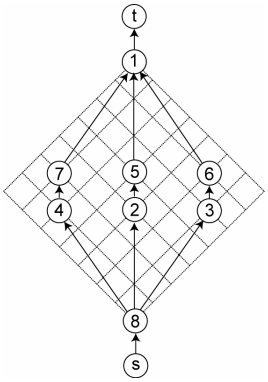
The order in which a rectangle appear in the two sequences determines its relative positions with respect to other rectangles. Given a rectangle x in a sequence pair $SP(S_1, S_2)$, the list of rectangles that appear before x in both S_1 and S_2 are located to the left of x in the packing. All rectangles that appear after x in both S_1 and S_2 are located to the right of x in the packing. Rectangles that appear after x in S_1 but before x in S_2 are located below x in the packing. Rectangles that appear before x in S_1 but after x in S_2 are located above x in the packing.

To compute the coordinates of the rectangles in the packing, we can use a directed graph called Horizontal Constraint Graph (HCG) to represent the "right-of" and "left-of" relation, where a directed edge $\langle a, b\rangle$ means that rectangle a is to the left of b. The weight of the edge $\langle a, b\rangle$ is the width of rectangle a. We add a source node s and connect it to all nodes in HCG with zero-weight edges. We also add a sink node t and connect all nodes in HCG to it. The weights of these edges are the widths of the originating nodes. The x coordinate of a rectangle in the packing is given by the longest path from the source to the corresponding node in HCG. The width of the bounding rectangle containing the packing is the longest path from the source to the sink. The left figure in the following shows the HCG of the sequence pair $SP((1,7,4,5,2,6,3,8),(8,4,7,2,5,3,6,1))$, whereas the right figure shows the same HCG except that transitive edges are removed for clarity. An edge $\langle a, c\rangle$ is transitive if there exists a node b such that a can reach c through b, i.e., edges $\langle a, b\rangle$ and $\langle b, c\rangle$ exist.

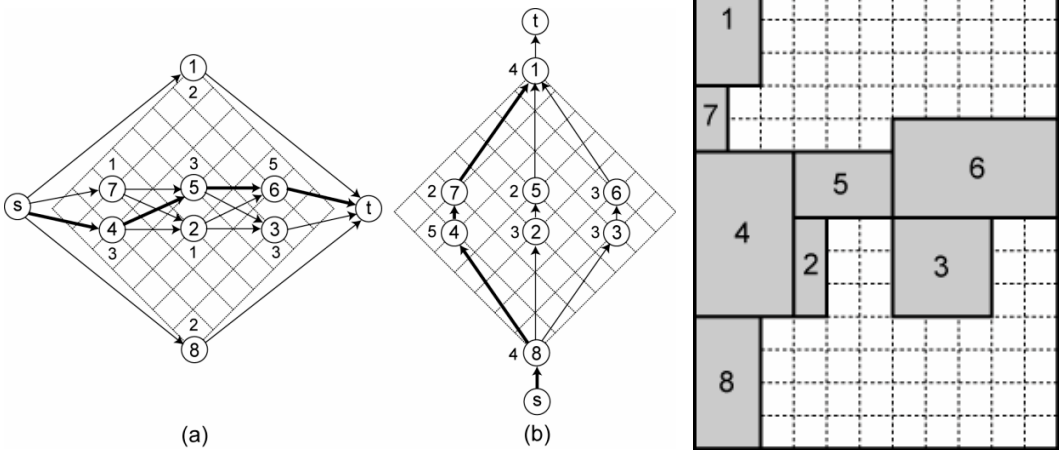


Likewise, we can use a Vertical Constraint Graph (VCG) to represent the "above" and "below" relation, where a directed edge $\langle a, b\rangle$ means that rectangle a is below b. The weight of the edge

$\langle a, b \rangle$ is the height of a. We add a source node and coonect it to all nodes in VCG with zero-weight edges, and add a sink node and connect all nodes in VCG to it with edges whose weights are the heights of the originating rectangles. The y coordinate of a rectanlge in the packing is given by the longest path from the source to the corresponding node in VCG. The height of the bounding rectangle containing the packing is the longest path from the source to the sink. The following figure shows the VCG of the sequence pair $SP((1,7,4,5,2,6,3,8),(8,4,7,2,5,3,6,1))$, with the transitive edges are removed for clarity.



Assuming that the (width, height) of the rectangles 1 through 8 are $\{(2,4), (1,3), (3,3), (3,5), (3,2), (5,3), (1,2), (2,4)\}$. The following figure shows the longest paths from the source node to the sink node in (a) HCG and (b) VCG. The numbers next to each node denotes its width (in HCG) or height (in VCG). The longest paths from the source to each node in HCG and VCG, respectively, give the x and y coordinates of each rectangle. The figure to the right of (a) and (b) shows the packing based on the sequence pair $SP((1,7,4,5,2,6,3,8),(8,4,7,2,5,3,6,1))$.



Deliverables:

You are to develop your own include file `seqpair.h` and source files `seqpair.c` and `seqpair_main.c`, which can be compiled with the following command:

```
gcc -std=c99 -Wall -Wshadow -Wvla -pedantic -O3 seqpair.c seqpair_main.c -o
proj5
```

(While developing your program, you should use the flag `-g` instead of the flag `-O3` so that you can debug your program more effectively.)

To run the program,

```
./proj5 input_file output_file
```

where the `input_file` is the filename of an input file and `output_file` is the filename of the output file. The formats of the input file and the output file are explained below.

Format of Input File

The input file is divided into three sections. The first section is a line that contains the number of rectanlges (in format `"%d\n"`) you have to "pack."

The second section specifies the dimensions of the rectangles, with each line specifiying the label, width, and height of a rectangle (in format `"%d(%le,%le)\n"`). This is similar how the label and dimensions of a rectangle are specified in PA3.

If n is the number of rectangles specified in the first line of the input file, the next n lines of the file specifies the label and dimensions of the rectangles. The labels are from 1 through n in ascending order.

The third section contains 2 lines, with each line containing a permutation of $1 through n$. The first line corresponds to the first sequence in the sequence pair, and the second line corresponds to the second sequence in the sequence pair. In each line, an integer is in the format `"%d"`, and every two adjacent integers are separated by a white space.

The following corresponds to the input file for the example shown earlier.

```
8
1(2.000000e+00,4.000000e+00)
2(1.000000e+00,3.000000e+00)
3(3.000000e+00,3.000000e+00)
4(3.000000e+00,5.000000e+00)
5(3.000000e+00,2.000000e+00)
6(5.000000e+00,3.000000e+00)
7(1.000000e+00,2.000000e+00)
8(2.000000e+00,4.000000e+00)
1 7 4 5 2 6 3 8
8 4 7 2 5 3 6 1
```

Format of Output File

The format of the output file is similar to the last output file in PA3. The file should contain a line for each rectanlge, with the label, x-coordinate and y-coordinate of the rectangle specified in the `"%d(%le,%le)\n"`. The rectangles are ordered from 1 through n, with n being the total number of rectangles.

The output file for the example given above should have the following format:

```
1(0.000000e+00,1.100000e+01)
2(3.000000e+00,4.000000e+00)
3(6.000000e+00,4.000000e+00)
4(0.000000e+00,4.000000e+00)
5(3.000000e+00,7.000000e+00)
6(6.000000e+00,7.000000e+00)
7(0.000000e+00,9.000000e+00)
8(0.000000e+00,0.000000e+00)
```

Grading:

The project requires the submission (through Blackboard) of the source and include files and a 1-page report. Your report should comment on the run-time and space complexity of your algorithm. It is best that your run-time is supported by run-times tabulated by executing your program on a set of examples of different numbers of rectangles. The report accounts for 10% of the grade and the program accounts for 90% of the grade. You should create a zip file called proj5.zip that contains all of the above and submit the zip file. For example, you can create a zip file with the following command:

```
zip proj5.zip seqpair.c seqpair.h seqpair_main.c report.pdf
```

Getting started:

We provide sample input files for you to evaluate the run-times of your packing program. Four sample input files `r8.sp`, `r100h.sp`, `r100v.sp`, and `r100.sp` are provided with respective solutions: `r8.pck`, `r100h.pck`, `r100v.pck`, `r100.pck`. In particular, `r8.sp` and `r8.pck` are the input and output files for the example used in the description.

Two other sample input files are larger but no solutions are provided: `r1K.sp` and `r2K.sp`.

Copy over the files from the Blackboard website. Check out the Blackboard website for any updates to these instructions. *Start packing!*