Homework #5

*Please present a **printed** report with all your answers, explanations, and sample plots. Submit also a soft copy of the source code and binaries used to generate these results. Please note that copying of any results or source code will result in ZERO credit for the whole homework.*

Problem 1

Describe an efficient algorithm that returns the *length* (number of edges) of a minimum-length negative-weight cycle in a graph, and returns infinity if the graph does not have negative-weight cycles. Argue that it is correct, and explain its running time.

[Hint: consider the "slow" matrix multiplication algorithm without the logarithmic *trick*, and note that the values $d_{ij}^{(n)}$ is the weight of the shortest path from vertex *i* to *j* using only *n* edges].

Problem 2

Arbitrage is the use of discrepancies in currency exchange rates to transform one unit of a currency into more than one unit of the same currency. For example, suppose that 1 U.S. dollar buys 49 Indian rupees, 1 Indian rupee buys 2 Japanese yen, and 1 Japanese yen buys 0:0107 U.S. dollars. Then, by converting currencies, a trader can start with 1 U.S. dollar and buys 49 x 2 x 0.0107 = 1.0486 US dollar, thus turning a profit of 4.86 percent.
Suppose that we are given *n* currencies c_1, c_2, … c_n and an $n \times n$ table R of exchange rates, such that one unit of currency c_i buys $R[i, j]$ units of currency c_j.

1. Describe an $O(V^3)$ algorithm to find the ***best*** arbitrage cycle, that is find the sequence of currencies $c_{i1}, c_{i2}, \ldots, c_{ik}, c_{i1}$ such that the product across the cycle
$$R[i_1, i_2] \cdot R[i_2, i_3] \cdot \ldots \cdot R[i_{k-1}, i_k] \cdot R[i_k, i_1] > 1$$
is ***maximum***.
[Hint: to do this, you need to find the actual shortest paths, and not just the shortest-path distances. Look at the textbook for how to modify the algorithms discussed in the lecture to incorporate computations of shortest paths.]
2. Implement your algorithm using C++. The input is also given on stdin, with the first line giving the number of currencies *n*, followed by *n* lines of *n* exchange rates each. The output should be either the list of currencies with the best arbitrage cycle or No if no such cycle exists, and given on stdout. For example, the input

```
4
1      5      0.75   2.5
0.21   1      0.1    0.5
1.2    11     1      12
0.4    1.9    10     1
```

should have as output:

```
1 2 3 4 1
```

Problem 3

Let $G = (V, E)$ be a flow network with source s and sink t. Let f be a flow in G. Let G_f be the residual network of G induced by f, let p be an augmenting path in G_f and let $c_f(p)$ be the minimum capacity along the augmenting path p. Define the function f_p: V×V → R as

$$f_p(u,v) = \begin{cases} c_f(p) & \text{if } (u,v) \text{ on } p \\ -c_f(p) & \text{if } (v,u) \text{ on } p \\ 0 & \text{otherwise} \end{cases}$$

Prove that f_p is a flow with $|f_p| = c_f(p) > 0$, that is prove it satisfies the properties of flows: capacity upper bound, skew-symmetry, and conservation. Also prove that its value is as given.

Problem 4

Professor Adam has two children who, unfortunately, dislike each other. The problem is so severe that not only do they refuse to walk to school together, but in fact each one refuses to walk on any block that the other child has stepped on that day. The children have no problem with their paths crossing at a corner. Fortunately both the professor's house and the school are on corners, but beyond that he is not sure if it is going to be possible to send both of his children to the same school. The professor has a map of his town.

Show how to formulate the problem of determining if both his children can go to the same school as a maximum-flow problem.

Problem 5

You are given an image, and you are asked to "segment" the image i.e. convert the image into disjoint regions. The goal of segmentation is to extract *coherent* regions, that is regions whose pixels are similar in some sense. You will be segmenting the image into two regions, usually called *foreground* and *background*. To do this, you will use a max-flow algorithm.

Pixels will be represented as vertices in a graph $G = (V, E)$, and will be connected to its "neighboring" pixels by edges. You will consider only 4-connectivity i.e. each pixel is connected to pixels to the north, south, east, and west of it. Edges will have *weights* $w(u,v)$ representing the "*similarity*" from each pixel to its neighbors. Segmentation will be done by finding the best *cut* i.e. dividing the vertices into two disjoint sets S and T. The best cut should have *minimum weight* across edges connecting pixels in S and pixels in T that is $\sum_{u \in S} \sum_{v \in T} w(u,v)$ is minimum. Intuitively, we need to cut across edges that connect least similar pixels.

1. How many edges and vertices are there in an image of size n x n?
2. Explain how to represent and solve this problem as a max-flow problem. In particular, explain:
   a. how to modify the graph G to another graph G' on which to apply a max-flow algorithm.
   b. how to find the desired segmentation from the max-flow solution i.e. for each pixel, find out if it belongs to S or T.
3. Implement Edmonds-Karp algorithm in C++, and use it to find a segmentation for an image. The similarity function should be defined as:

$$w(u,v) = 3 \times 255 - d_c(u,v)$$

where $d_c(u,v)$ is the color distance between pixels u and v defined as

$$d_c(u,v)=|r_u-r_v|+|g_u-g_v|+|b_u-b_v|$$

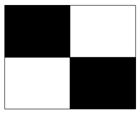
where the RGB color of pixel u is (r_u, g_u, b_u) and similarity for pixel v. Use the image in the attached archive to test your algorithm. It contains images in PPM format, which is an ASCII format. It has a header

```
P3
W H
255
R G B R G B …
```

where “P3” defines the version of the format, W is the width and H is the height, 255 is the maximum value for the color i.e. full red is 255. Pixels are then specified by their RGB values in a row-major order, starting from the top left pixel towards the bottom right pixel. For example, the following file:

```
P3
2 2
255
0 0 0 255 255 255 255 255 255 0 0 0
```

specifies the following 2x2 image:



Include the results of running your algorithm on the example images. You can do this by creating another binary image with pixel values of 1 for pixels of one region and pixel values of 0 for pixels of the other region.

4. What is the asymptotic running time of your algorithm?

Instructions

Please submit a soft copy of the solutions together with source code and binaries in one zip file. The file should be named as **CMP448.HW##.BN##.First.Last.zip** where HW## is the homework number e.g. HW01, BN## is your bench number, First and Last are your first and last name. So, if your bench number is 26, this is homework 3, and your name is Mohamed Aly, the file should be named **CMP448.HW03.BN26.Mohamed.Aly.zip.** Failing to follow these instructions will cost you points.