Joohan Lee
joohanl@usc.edu

Coding Challenge Report
Dragonfruit AI

1 Introduction

- Python3.8.2
- OS: Windows 11
- Objective: Come up with data structures to store parasite images and a function to compute whether a parasite has cancer or not.
- Check the code here

2 Questions

2.1 Come up with efficient data structures to represent both types of images: those generated by the microscope, and those generated by the dye sensor. These need not have the same representation; the only requirement is that they be compact and take as little storage space as possible. Explain why you picked the representation you did for each image type, and if possible estimate how much storage would be taken by the images. What is the worst-case storage size in bytes for each image representation you chose?

We can represent both types of images using two matrices in which the black part of microscope and part where dye is lit are represented by 1 and the white part of microscope and part where dye is not lit are represented by 0. One matrix represents an image of microscope and the other represents an image of dye sensor. However, since parasite occupies only 25% or more of the total area of the image, images will be represented as a sparse matrix. It is inefficient to store and compute all elements including 0s' values in the matrix. Therefore, I want to store only 1s' coordinates using hash. For example, should we store the following matrix, [[1,0,1,0],[0,0,0,0],[0,0,1,0]], I will store as following, data = 0:[0,2],2:[2]. Each array in the data represents the row in the matrix and each element in the array represents the column in the array. Thus, the array with key 0 represents there are 1 values in the matrix on (0,0) and (0,2). In the second row of the matrix, there is no 1 value, so I did not store any coordinates from it. The array with key 2 represents third row's columns with 1 value in the matrix and means there is 1 value on (2,2) in the matrix. I don't need to even store values of matrix because the image matrix has only 1 or 0 values and I only care about 1s.

In the worst case, there can be all 1s in the matrix and it causes around 40MB for each image since one integer takes 4Bytes and I need 100,000 * 100,000 elements to store all coordinates. However, if there are many parasite occupying 50% or more of the total area of the image, I can add one variable to set status expressing which value has majority. For instance, if 1s are majority in the matrix, I can store 0s' coordinates in the data structure with the status variable that represents figure out which values are majority. This let the data structure use only about 20MB even in the worst case.

2.2 Before the researchers give you real images to work with, you would like to test out any code you write. To this end, you would like to create "fake" simulated images and pretend they were captured by the microscope and the dye sensor. Using the data structures you chose in (1) above, write code to create such simulated images. Try and be as realistic in the generated images as possible.

I created fake parasite bodies' images and dyed images of their vessels. I randomly chose a point on the frame and did BFS to draw parasite's body. Then I drew winding lines from the same point to four directions in order to represent dyed vessels. When I drew vessels, I made a function that can set thickness of lines and how many vessels are dyed. Since parasite with cancer has over 10% overlap between parasite image and dyed image, I made a fake cancer image by drawing a dyed image with from 50 to 100 lines. Finally I stored those images for the function to read them and check whether a parasite has cancer or not.

2.3 Using the simulated images generated by the code you wrote for (2) above as input, write a function to compute whether a parasite has cancer or not.

First of all, I read images of parasites and their dyed vessels. Next, I converted them to data structures that I came up with in (1) above. After that, I could easily get overlap between parasite's body and its vessel. If the overlap is over 10%, I printed it out as an image with cancer and saved that image with the name of 'cancer_vessel'.

2.4 You give your code from (3) to the researchers, who run it and find that it is running too slowly for their liking. What can you do to improve the execution speed? Write the code to implement the fastest possible version you can think of for the function in (3).

I stored all non 0 values using tuples like [(0,0),(0,2),(0,3),(1,1),(1,2)] for [[1,0,1,1],[0,1,1,0]] matrix at first. However, I realized saving rows causes repeated data as we can see repeating (0,) above. Therefore, I used hash table with keys as rows and values as columns. That let the function use less storage and show better performance.