傅其骞
3210115455
qiqianf2
qiqianf2@illinois.edu

1. Overall Analysis

a. analysis of memory allocation and running time in your original MP5 implementation

In mp5, it uses function to load all the tileimages at the same time. And return a vector containing all the tileimages instances. It not only has low efficiency in memory allocation (which causes a large amount of memory allocation), also wastes the precise png files it reads (which it needs to re-read while pasting).

b. analysis of memory allocation and running time in your new MP6 implementation

In MP6, I only store the address of the png while it is read for the first time in main.cpp. Since the Tileimage type of an image is used while looking for its average color (in main.cpp all the pngs are inserted to the vector after making sure there're no two same average color images existed in the vector), I store the average color as one of the values of the number of the png. So it saves the memory allocation since addresses is much smaller than Tileimage ADT. What's more, while pasting the tileimage to the canvas, I check whether the tileimage has been used or not (since for the first time the image will be cropped to a square). If it has been used, than just calling the address of its cropped image. In this way, it saves the time to translate tileimage to png and saves time to re-cropping for the same pngs.

c. description of changes made to reduce memory footprint and running time

In MP6, I only store the address of the png while it is read for the first time in main.cpp. I store the average color as one of the values of the number of the png. So it saves the memory allocation since addresses is much smaller than Tileimage ADT. What's more, while pasting the tileimage to the canvas, I check whether the tileimage has been used or not (since for the first time the image will be cropped to a square). If it has been used, than just calling the address of its cropped image. In this way, it saves the time to translate tileimage to png and saves time to re-cropping for the same pngs.

2. Specific Analysis

a. Analysis of how much additional memory is allocated when populating images in your original MP5 implementation

In mp5, it uses function to load all the tileimages at the same time in main.cpp. And in mosciacanvas.cpp, while pasting, all this tileimages are cropped to square, but many tileimages are cropped more than once. I think this are two main additional allocated memory.

b. Describe what you have done to reduce memory allocations in populating images (e.g., what functions/files you have modified, what you have adjusted, and so on) and how your changes theoretically reduce memory allocations in populating images.

In mainc.pp, I only store the addresses of png as one of the values of the map tiles. The other value of the tiles is average color, which is HSLAPixel ADT. In this way, we don't need to store Tileimage instances or the addresses of Tileimage. To satisfy the generateResizedimage function in mosaiccanvas.cpp, I rewrite a generateResizedimage for PNG ADT, so that I can no longer use Tileimage for resizing. And after each png is resized, I store the address of the resized png so that if the png was used for a second time, no memory will be need again.

c. Empirical analysis on your successful reduction of memory allocation: what command you have run, what additional images you have deployed (if any), and what the results are (can be in any form that can convince us, e.g., screenshots)

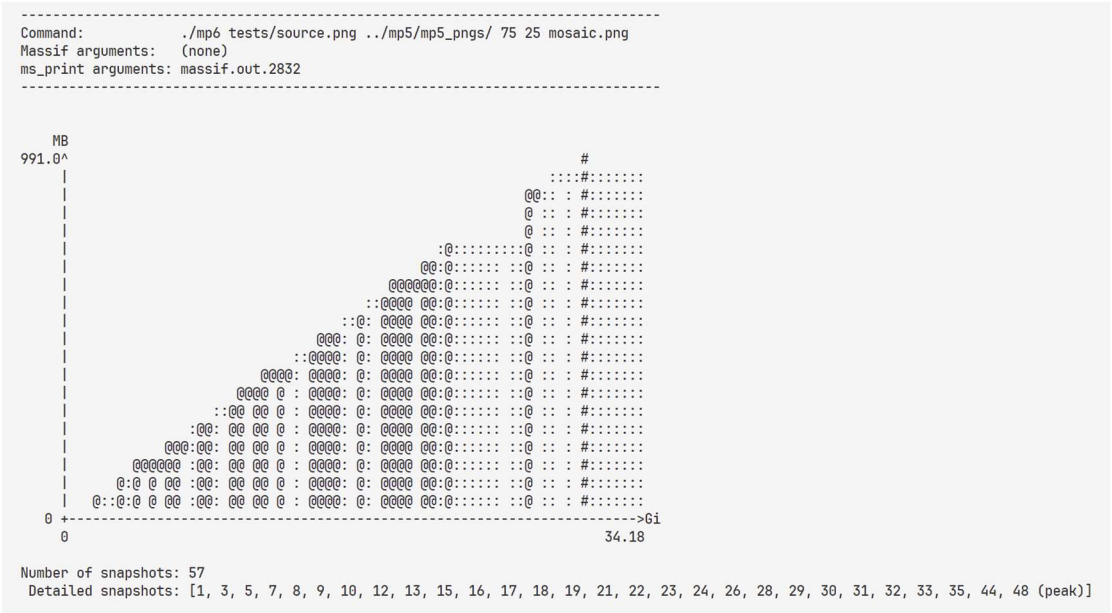
The command I run is this:

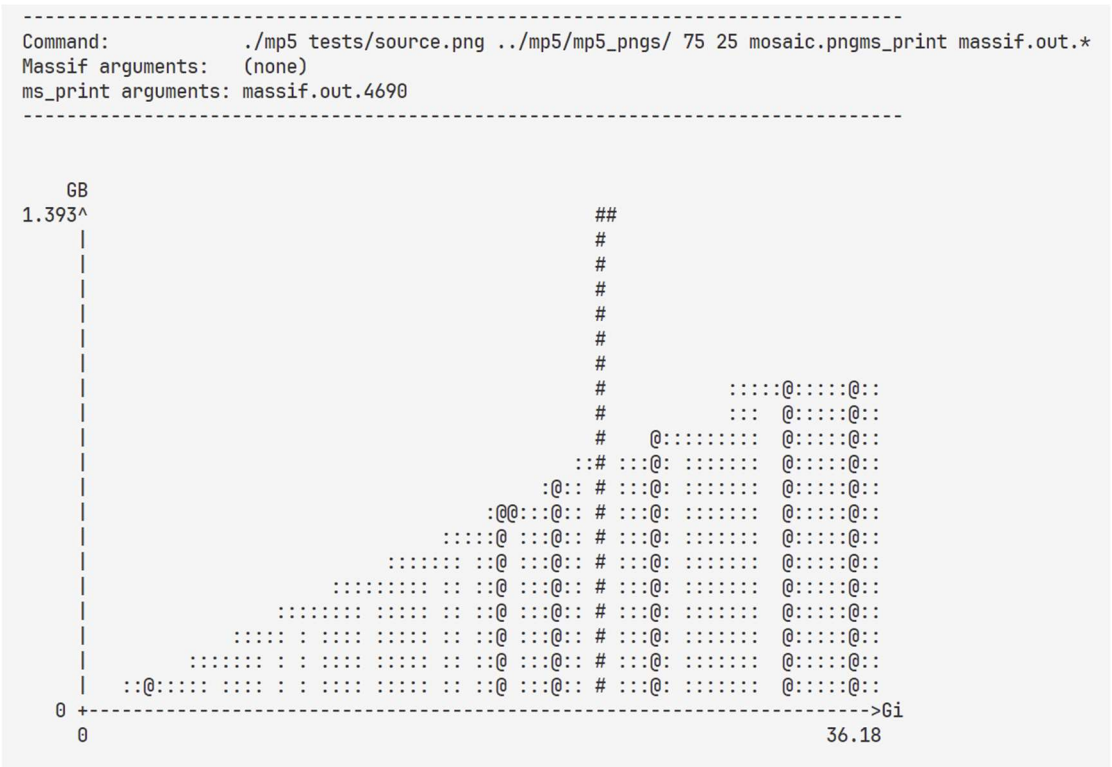
Valgrind --tool=massif ./mp5 tests/source.png ../mp5/mp5_pngs/ 75 25 mosaic.pngms_print massif.out.*
Valgrind --tool=massif ./mp6 tests/source.png ../mp5/mp5_pngs/ 75 25 mosaic.pngms_print massif.out.*
Valgrind --tool=massif ./mp6 tests/source.png ../mp5/mp5_pngs/ 400 5 mosaic.pngms_print massif.out.*

ms_print massif.out.*

It shows that in MP6, there's no such a peak of memory allocation at the middle of the program, because I substitute the vector of Tileimage to map of point of PNG.

That's how I improve the algorithm : use less space and delete the peak for transportation of a vector of tileimage.





```
--------------------------------------------------------------------------------
Command:            ./mp6 tests/source.png ../mp5/mp5_pngs/ 400 5 mosaic.pngms_print massif.out.2832
Massif arguments:   (none)
ms_print arguments: massif.out.410
--------------------------------------------------------------------------------


    GB
0.980^                                                                     #
     |                                                                    :#:
     |                                                                  @::#:
     |                                                                  @::#:
     |                                                                  @::#:
     |           @::::::::@:@::::@::@:::::::@::@::::@:::::@::::@::::@::@::#:
     |          @@: ::::::@:@::::@::@:::::::@::@: ::@:::::@::::@::::@::@::#:
     |          @@: ::::::@:@::::@::@:::::::@::@: ::@:::::@::::@::::@::@::#:
     |         ::@@: ::::::@:@::::@::@:::::::@::@: ::@:::::@::::@::::@::@::#:
     |         : @@: ::::::@:@::::@::@:::::::@::@: ::@:::::@::::@::::@::@::#:
     |        @: @@: ::::::@:@::::@::@:::::::@::@: ::@:::::@::::@::::@::@::#:
     |       :@: @@: ::::::@:@::::@::@:::::::@::@: ::@:::::@::::@::::@::@::#:
     |      @:@: @@: ::::::@:@::::@::@:::::::@::@: ::@:::::@::::@::::@::@::#:
     |     @@:@: @@: ::::::@:@::::@::@:::::::@::@: ::@:::::@::::@::::@::@::#:
     |    @@@:@: @@: ::::::@:@::::@::@:::::::@::@: ::@:::::@::::@::::@::@::#:
     |    @@@:@: @@: ::::::@:@::::@::@:::::::@::@: ::@:::::@::::@::::@::@::#:
     |   @@@@:@: @@: ::::::@:@::::@::@:::::::@::@: ::@:::::@::::@::::@::@::#:
     |  :@@@@:@: @@: ::::::@:@::::@::@:::::::@::@: ::@:::::@::::@::::@::@::#:
     | ::@@@@:@: @@: ::::::@:@::::@::@:::::::@::@: ::@:::::@::::@::::@::@::#:
     |:::@@@@:@: @@: ::::::@:@::::@::@:::::::@::@: ::@:::::@::::@::::@::@::#:
   0 +-------------------------------------------------------------------------->Gi
     0                                                                     135.1
```

d. Analysis of why drawing an image in your original MP5 implementation has an O(w*h*w'*h') running time

   Because no matter this Tileimage has been cropped or not, it will be cropped automatically, so for each pixel in the original image(w*h), all pixels in Tileimage(w'*h') are used for generating the cropping image.

e. Describe what you have done to reduce running time in drawing images (what functions/files you have modified, etc.) and how your changes theoretically reduce running time in drawing images.

   In MP6's main.cpp, I only store the address of the png while it is read for the first time in map ADT "titles". I store the average color as one of the values of the number of the png. So it saves the memory allocation since addresses is much smaller than Tileimage ADT. When average color is need for finding nearest pixel, just use the map to get it's HSLAPixel average color (this requires me to rewrite maptitles.cpp). What's more, in mosaiccanvas.cpp, while pasting the tileimage to the canvas, I check whether the tileimage has been used or not (since for the first time the image will be cropped to a square). If it has been used, than just calling the address of its cropped image. In this way, it saves the time to translate tileimage to png and saves time to re-cropping for the same pngs. Since generateResizedimage function is for Tileimage ADT, I rewrite a generateResizedimage function in PNG.cpp.

f. Empirical analysis on your successful reduction of running time: what command you have run, what additional images you have deployed (if any), and what the results are (can be in any form that can convince us, e.g., screenshots).

```
austin@Qiqian-Fu-ROG:~/cs225sp23/mp6$ time ./mp6 tests/source.png ../mp5/mp5_pngs/ 400 5 mosaic.png
Loading Tile Images... (4730/4730)... 4479 unique images loaded
Populating Mosaic: setting tile (399, 532)
Drawing Mosaic: resizing tiles (213200/213200)
Saving Output Image... Done

real    0m32.370s
user    0m26.982s
sys     0m3.630s
```

```
austin@Qiqian-Fu-ROG:~/cs225sp23/mp5$ time ./mp5 tests/source.png ../mp5/mp5_pngs/ 400 5 mosaic.png
Loading Tile Images... (4730/4730)... 4479 unique images loaded
Populating Mosaic: setting tile (399, 532)
Drawing Mosaic: resizing tiles (213200/213200)
Saving Output Image... Done

real    0m51.018s
user    0m26.022s
sys     0m3.273s
```

I ran the same command which is shown in the screenshot. It takes less time to generate the picture, which means I successfully reduce the running time.