Name: Deng Ziye

ZJU ID: 3210115444

Intl ID: ziye.21

NetID: ziyed2

Section a: analysis of memory allocation and running time in my original MP5 implementation

```
Drawing Mosaic: resizing tiles (213200/213200)
Saving Output Image... Done
==6410==
==6410== HEAP SUMMARY:
==6410==     in use at exit: 0 bytes in 0 blocks
==6410==   total heap usage: 955,499,830 allocs, 955,499,830 frees, 97,060,760,395 bytes allocated
==6410==
==6410== All heap blocks were freed -- no leaks are possible
==6410==
==6410== For lists of detected and suppressed errors, rerun with: -s
==6410== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

In my mp5's implementation, I use "valgrind –leak-check=full" command to check the memory allocation. I totally have 955,499,830 allocs, all of which are freed, and 97,060,760,395 bytes are allocated.

```
ziyed2@ziyed2-VirtualBox:~/cs225sp23/mp5$ time ./mp5 tests/source.png 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    2m27.454s
user    1m0.469s
sys     0m2.433s
```

About the running time, I use "time" command to check and the total running time is 2min27.454s for my mp5.

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

For my mp6, my mainly change is on the parameter. I changed the parameter

of **get_match_at_idx** function in **maptiles.{cpp, h}**. The parameter "**map<Point<3>, int> tile_avg_map**" becomes "**map<Point<3>, int>& tile_avg_map**", which is a reference.

```
Drawing Mosaic: resizing tiles (213200/213200)
Saving Output Image... Done
==5213==
==5213== HEAP SUMMARY:
==5213==     in use at exit: 0 bytes in 0 blocks
==5213==   total heap usage: 581,761 allocs, 581,761 frees, 5,388,318,256 bytes allocated
==5213==
==5213== All heap blocks were freed -- no leaks are possible
==5213==
==5213== For lists of detected and suppressed errors, rerun with: -s
==5213== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

For my mp6 implementation, I totally have 581,761 allocs and 5,388,318,256 bytes allocated, which is a lot smaller than mp5's.

I think the reason is that I used reference instead of value as parameter, which prevent the map **tile_avg_map** from copying each time I called the function **get_match_at_idx** . So, I reduce memory allocations in populating images .

```
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    0m26.017s
user    0m3.733s
sys     0m2.768s
```

As for the running time of my mp6 implementation, After I make this change, I used command "time ./mp6_tests/sources ../mp5/mp5_pngs 400 5 mosaic.png" and find that the time become much shorter than mp5. I successfully reduce the run time from 2min27s to 26s.

I think the reason might be that I prevent the copy of the map **time_avg_map** from everytime I call the function **get_match_at_idx**, which is an extremely large

work.

Section C: description of my change and the reason it reduce memory allocation and running time

For my analysis of the running time, I think the reason why original time complexity is **O(w*h*w'*h')** is that I didn't use reference of **tile_avg_map**, and it is copied at every loop when I called the function **get_match_at_idx**. After this change, I make the complexity become **O(w*h+n*w'*h')**, so the time reduces a lot.