Name: Lu Yanxin

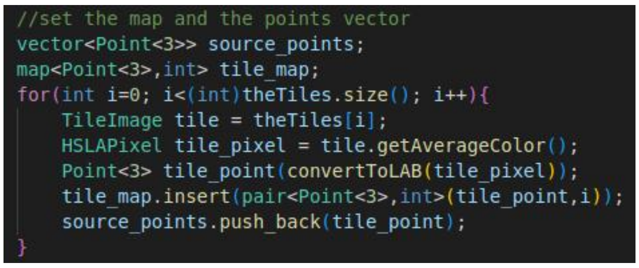
ZJU ID: 3210115435

intl ID: yanxin.21

UIUC NetID: yanxinl4

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

Observation 1: memory occupation

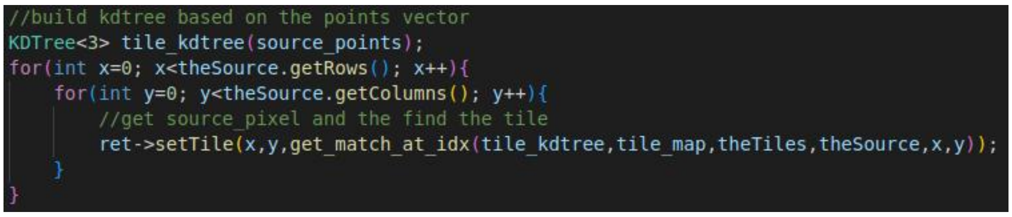


```
//set the map and the points vector
vector<Point<3>> source_points;
map<Point<3>,int> tile_map;
for(int i=0; i<(int)theTiles.size(); i++){
    TileImage tile = theTiles[i];
    HSLAPixel tile_pixel = tile.getAverageColor();
    Point<3> tile_point(convertToLAB(tile_pixel));
    tile_map.insert(pair<Point<3>,int>(tile_point,i));
    source_points.push_back(tile_point);
}
```

part of map_tiles in mp5

In my MP5, I have convert the PNG to point and store the pair in the map. In this situation, I have my MP allocate w*h*C memory with C=8 bytes instead of w*h*PNG so that the MP5 only require a small memory.

Observation 2: time complexity

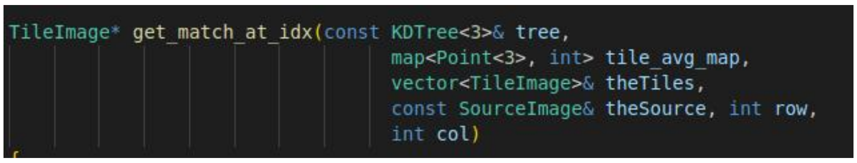


```
//build kdtree based on the points vector
KDTree<3> tile_kdtree(source_points);
for(int x=0; x<theSource.getRows(); x++){
    for(int y=0; y<theSource.getColumns(); y++){
        //get source pixel and the find the tile
        ret->setTile(x,y,get_match_at_idx(tile_kdtree,tile_map,theTiles,theSource,x,y));
    }
}
```

part of map_tiles in mp5

O(w*h+n*w′*h′) consists of two parts. O(w*h) is to set the tile of each pixel and O(n*w′*h′) is to draw the mosaic based on the set tiles. I have achieved this time complexity since I didn’t load the “tile images” in observation 1.

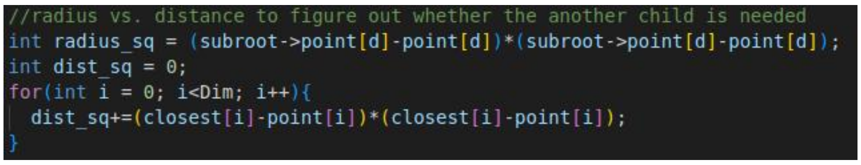
In the helper function “get_match_at_idx”, the parameter map<Point<3>, int> tile_avg_map copies the map when the function is called. Therefore, it requires a lot of memory to copy the map and time complexity for the copy.



```
TileImage* get_match_at_idx(const KDTree<3>& tree,
                            map<Point<3>, int> tile_avg_map,
                            vector<TileImage>& theTiles,
                            const SourceImage& theSource, int row,
                            int col)
```

get_match_at_idx in mp5

When checking the code, I find I make the mistake as below. When using integer to compare the distance of neighbors, the kdtree will be much more centralized since the distance is rounded. As a result, it takes more time to find the nearest neighbour in the kdtree and the result may be not correct.

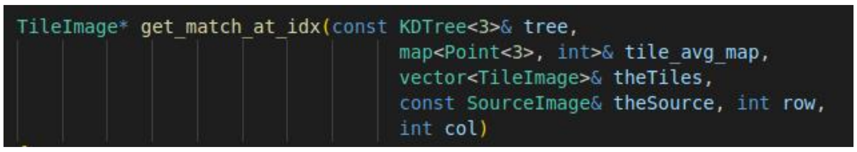


```
//radius vs. distance to figure out whether the another child is needed
int radius_sq = (subroot->point[d]-point[d])*(subroot->point[d]-point[d]);
int dist_sq = 0;
for(int i = 0; i<Dim; i++){
  dist_sq+=(closest[i]-point[i])*(closest[i]-point[i]);
}
```

get_match_at_idx in mp5

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

Since the function doesn't change the content of the map, so I add a "&" reference operator so that we don't need to copy the map but use it directly.

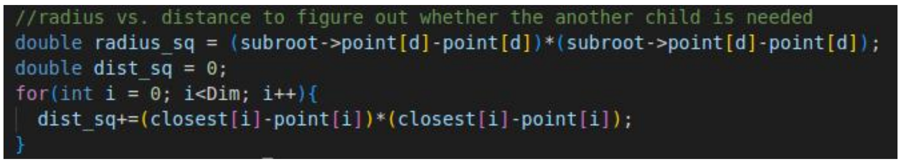


```
TileImage* get_match_at_idx(const KDTree<3>& tree,
                            map<Point<3>, int>& tile_avg_map,
                            vector<TileImage>& theTiles,
                            const SourceImage& theSource, int row,
                            int col)
```

get_match_at_idx in mp6

In addition, I also make some changes. In my mp5, I have all of the helper functions written in kdtree.h for convenience, but it takes more time to compile and run. So I write the declaration in kdtree.h and the body of the helper functions in kdtree.cpp to improve the time of running the program.

After changing the type of the variables, I find the code runs more smoothly and it takes less time to find the nearest neighbor.

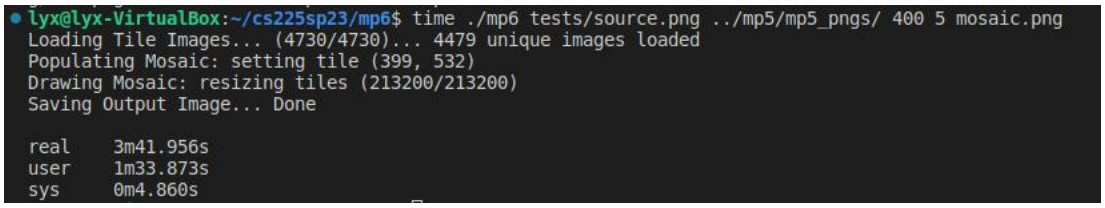


```
//radius vs. distance to figure out whether the another child is needed
double radius_sq = (subroot->point[d]-point[d])*(subroot->point[d]-point[d]);
double dist_sq = 0;
for(int i = 0; i<Dim; i++){
  dist_sq+=(closest[i]-point[i])*(closest[i]-point[i]);
}
```

get_match_at_idx in mp6

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

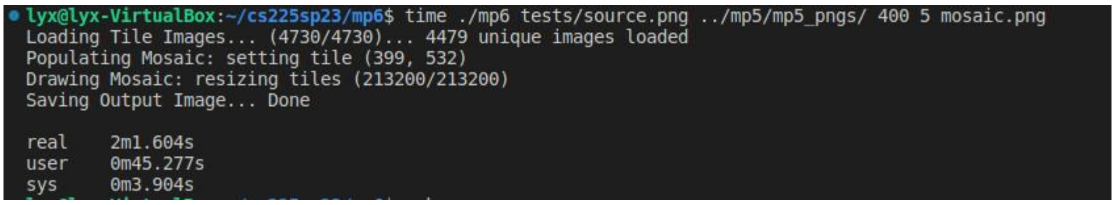
The initial version,



```
lyx@lyx-VirtualBox:~/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    3m41.956s
user    1m33.873s
sys     0m4.860s
```

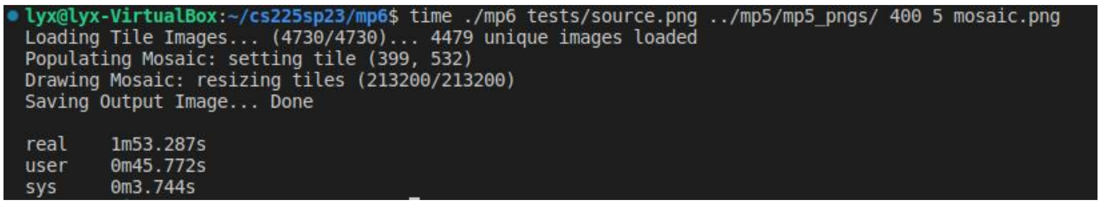
After changing the reference in get_match_at_idx,



```
lyx@lyx-VirtualBox:~/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    2m1.604s
user    0m45.277s
sys     0m3.904s
```

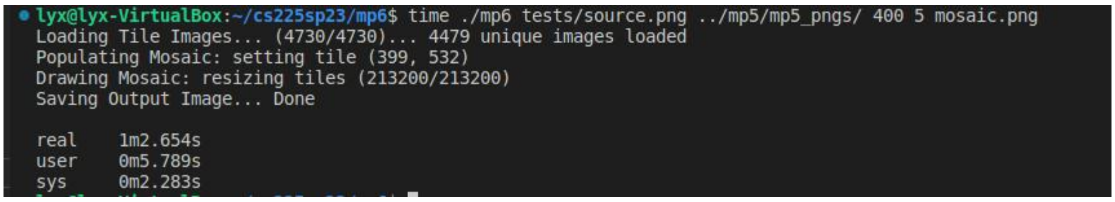
After changing the helper functions in .cpp or .h,



```
lyx@lyx-VirtualBox:~/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    1m53.287s
user    0m45.772s
sys     0m3.744s
```

After changing the type of variables,



```
lyx@lyx-VirtualBox:~/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    1m2.654s
user    0m5.789s
sys     0m2.283s
```