Part 4

Based on the Heuristic function I created for part 2, I tried multiple attempts to make my heuristic function better.

1. Because the map we want to climb is a mountain and the optimal position we want to have at specific point will proportion with the distances we climbed from the start point and the height of the mountain. So I tried to add this relationship into my heuristic function. If the current height is more close with the height we should be, we are more likely on the right track, but in this way the program cannot have the best path.
2. I think the search method is already the most optimal one. By using the Hashmap, the optimal time complexity for insert and delete will be O(1) and at most O(n).

In the end, I find that the access to start point and end point from class function from Terrainmap will save more time than create variables and give them the value of start point and end point. So I delete the original variable and get the value of start/end point from Terrainmap class every time I use it. Besides, I also create a index value to count the loop we have when execute the while loop. Because the shortest path will always larger or equal that the distance between the start and the end point. So the compare in the beginning of while loop will cost come time. So I add a if function and the compare of current point and end point will only happen after the index larger than the distance between the start and the end point. Besides, The most time consuming thing is to execute points in the priority queue, so I add a restriction that valid points will only emerge at the fourth quadrant to add less points into queue and increase the speed. Last but not least, the for loop inside the while loop which update the value for neighbor nodes can be processed by OpenMp, because every iterate is independent with each other. By using parallel calculation, the speed will be increased.

Comparing with part2, the speed is about 50% percent higher for the Div and 20%higher for the Exp on my local machine.