Cost Function of Power world: Math.pow(2.0, (getTile(p2) - getTile(p1))) = $2^{h_2-h_1}$

In this part of the heuristic, we are trying to locate the best case of overall cost function. The overall cost function is:

Overall Cost = # of steps $*$ power per step

Firstly we give this function a limit so that the # of steps equals the least amount of steps from starting(current) point to end point. Because we can do chess move, this LeastDistance is the larger number of the difference between Start and End in x-coordinate and the one in y-coordinate.

```
double DistanceX = Math.abs(pt2.x - pt1.x);
double DistanceY = Math.abs(pt2.y - pt1.y);
double LeastDistance = Math.max(DistanceX,DistanceY)+20;
```

With a fixed # of steps, the minima of the Overall Cost happens when all steps get the least difference in height between Start and End. In such case the height difference is separated averagely to each step.

Overall Cost = LeastDistance $*$ $2^{\frac{Difference\ of\ Height}{LeastDistance}}$

Then we realize that the minima of Overall Cost might not happen in the case of LeastDistance. We set up a function

Overall Cost = x $*$ $2^{\frac{Difference\ of\ Height}{x}}$ where x $\geq$ LeastDistance

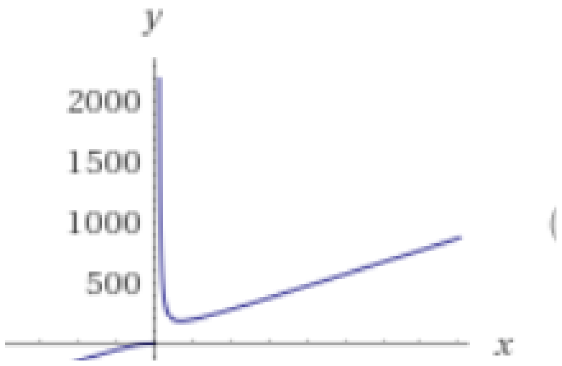


Figure 1: Example of Difference in Height = 100

Mathematical estimation gives that the minima of this function is

When Difference in Height $\geq$ 0:

Overall Cost = Difference in Height $*$ e $*$ $\log_{10} 2$

at x = Difference in Height $*$ $\log_{10} 2$

When Difference in Height $<$ 0:

Overall Cost = LeastDistance $*$ $2^{\frac{Difference\ of\ Height}{LeastDistance}}$

at x = LeastDistance

This leads to the huristic in which each estimate is further scaled down:

```
        double DifferenceHeight = map.getTile(pt2) - map.getTile(pt1);
        if (DifferenceHeight >= 0){
            return 1.8*DifferenceHeight;
        }else if(DifferenceHeight < 0){
            double DistanceX = Math.abs(pt2.x - pt1.x);
            double DistanceY = Math.abs(pt2.y - pt1.y);
            double LeastDistance = Math.max(DistanceX,DistanceY);
            return 0.8 * LeastDistance *
Math.pow(2,DifferenceHeight/LeastDistance);
        }
```

This heuristic is guaranteed consistent as we can already prove the best case cost. However, we then realize that this heuristic underestimates too much. It includes too many cases that are impossible to happen, leading to a bad performance in reducing blocks visited and time used. In part 4 we try to be more greedy and get heuristic of

$$\textbf{Overall Cost} = \textbf{0.8} * \textbf{LeastDistance} * 2^{\frac{\textit{Difference of Height}}{\textit{LeastDistance}}}$$

In which 0.8 is a scale down, thus an underestimate of the best case. This new heuristic then replace our old part 1 heuristic. **Consistency** examples are given as followed:

Consider moving from T1 to T2 toward end tile Te.

If Te≥T1 and Te≥T2, taking example of T1 = 1; T2 = 3; Te = 10; LeastDistance = 10 for T1;

$f(1) = 10 * 2^{\frac{9}{10}} \approx 18.661$

$f(2) = 2^2 + 9 * 2^{\frac{7}{9}} \approx 19.43$

$f(2) \geq f(1)$ and this holds for all $f(n+1) \geq f(n)$

If Te≥T1 and Te<T2, taking example of T1 = 1; T2 = 13; Te = 10; LeastDistance = 10 for T1;

$f(1) = 10 * 2^{\frac{9}{10}} \approx 18.661$

f(2) = 2^{12} + 9 * $2^{\frac{-3}{9}}$ ≈ 4103.14

f(2) ≥ f(1) and this holds for all f(n+1) ≥ f(n)

If Te<T1 and Te<T2, taking example of T1 = 10; T2 = 3; Te = 1; LeastDistance = 10 for T1;

f(1) = 10 * $2^{\frac{-9}{10}}$ ≈ 5.359

f(2) = 2^{-7} + 9 * $2^{\frac{-2}{9}}$ ≈ 7.723

f(2) ≥ f(1) and this holds for all f(n+1) ≥ f(n)

If Te<T1 and Te≥T2, taking example of T1 = 10; T2 = 1; Te = 2; LeastDistance = 10 for T1;

f(1) = 10 * $2^{\frac{-8}{10}}$ ≈ 5.743

f(2) = 2^{-9} + 9 * $2^{\frac{1}{9}}$ ≈ 9.722

f(2) ≥ f(1) and this holds for all f(n+1) ≥ f(n)

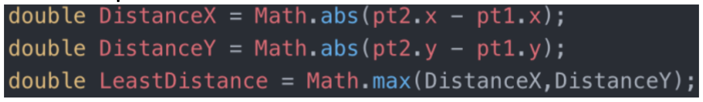
Thus in all cases this heuristic is consistent.

Admissable

Cost Function of Bizzaro World : **1.0*(getTile(p1) / (getTile(p2) + 1)) =** $\frac{h1}{h2+1}$ where h1 is the height of current point and h2 is the height of successor point given action a.

The overall cost function is : **= # of steps *** $\frac{h1}{h2+1}$ **per step**

In the proof of power cost heuristic, we have showed that the least amount of steps from current point to end point is equal to LeastDistance.



```
double DistanceX = Math.abs(pt2.x - pt1.x);
double DistanceY = Math.abs(pt2.y - pt1.y);
double LeastDistance = Math.max(DistanceX,DistanceY);
```

In the definition of Terrain map problem, the height of any point is ranged from 0 to 255. Therefore, the most height of successor of current point is 255 and the least height of successor of current point is 0. Our heuristic depends on the height of current point and the height of end point.

1. The following example is the case that the current point is under the end point.

			E(58)
		3	9
	S(2)	4	6

We can find that in order to get to end point which is higher than start point, Each step cost from current point to successor is always at least larger than or equal to $\frac{h1}{255+1}$ for any action. Given the numerator in fraction as constant, the value of fraction becomes larger as the denominator has smaller value. And we can assure that the denominator or the height of successor + 1 is always bounded below by 256. Therefore, given the least amounts of steps from current point to end point, LeastDistance, we can show that " heuristic = h1/256 * LeastDistance" is always underestimated by true cost get from current point to end point, when the current point is below the end point.

The following examples show that the heuristic is admissible when the start point is below or at the same height of end point.

12	1	2	E(58)
23	3	57	9
22	S(56)	4	6
12	45	22	13

The heuristic of starting point S = 56/256 * 2 =0.4375

The true cost from S to E = $\frac{56}{57+1} + \frac{57}{58+1} = 1.9316$

So, heuristic underestimates by true cost

The best case of starting point below end point is when the starting point is at the height 1 and end point is at the height of 255. And all the successor in path has the height of 255

12	1	2	10
23	3	E(255)	9
22	S(1)	4	6
12	45	22	13

The heuristic of starting point S = 1/256 = $\frac{1}{256}$

The true cost from S to E = $\frac{1}{255+1} == \frac{1}{256}$. We can see that the best step cost along best path is equal to $\frac{1}{256}$. And heuristic is equal to true cost of best cases.

Therefore, heuristic is admissible.

2. The following example is the case that the current point is above the end point.

S(155)	1	2	2
23	89	57	9
22	11	E(10)	6
12	45	22	13

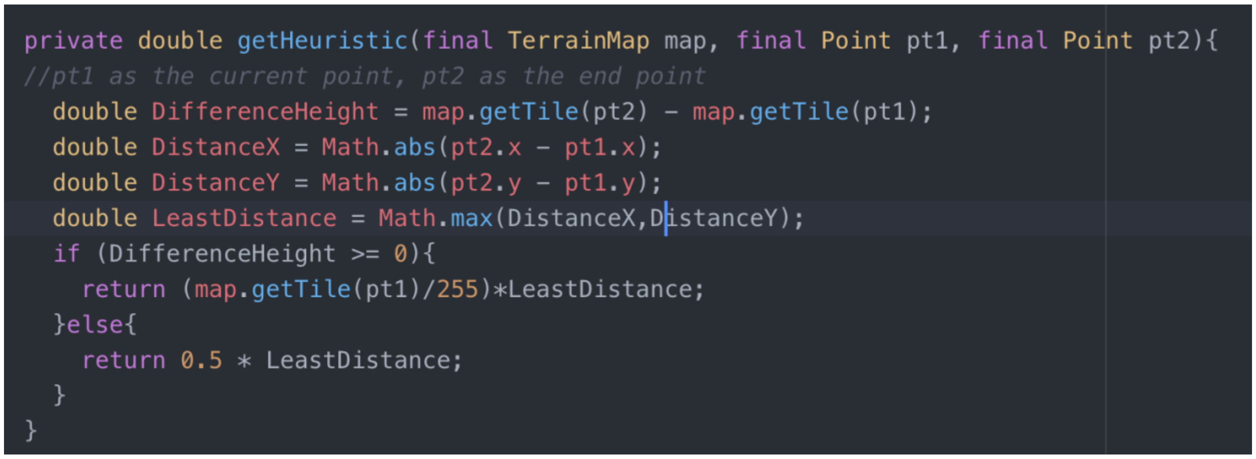
Consider the step cost is equal the fraction $\frac{h1}{h2+1}$. Then for each step, the step cost is always larger than or equal to 1 if the successor is below the current point. Although there is some cases that the successor is above the current point, the path cost of this cases is always larger than the path cost of the case that successors are always below current point because the more difference height of current point to end, the larger the value of fraction $\frac{h1}{height\ of\ end\ point\ +1}$.

Consider the best cases that the starting point above end point is 256 and its successors along the path are at the same height of end point which is 1.

S(155)	1	2	2
1	89	57	9
1	11	0	6
E(1)	45	22	13

We can see the along the path of the best case, the best path cost is $\frac{1}{2}$. Therefore, by setting the **heuristic = 0.5 * LeastDistance.** We have the heuristic is always underestimates true cost.

Overall, we have two heuristics for points depending on the differences of its height and height of end point and we have showed that they are all underestimated and admissible.



```
private double getHeuristic(final TerrainMap map, final Point pt1, final Point pt2){
//pt1 as the current point, pt2 as the end point
  double DifferenceHeight = map.getTile(pt2) - map.getTile(pt1);
  double DistanceX = Math.abs(pt2.x - pt1.x);
  double DistanceY = Math.abs(pt2.y - pt1.y);
  double LeastDistance = Math.max(DistanceX,DistanceY);
  if (DifferenceHeight >= 0){
    return (map.getTile(pt1)/255)*LeastDistance;
  }else{
    return 0.5 * LeastDistance;
  }
}
```

Consistency

In order to prove a heuristic equation is consistent, it must obey the triangle inequality at all points, since it is almost impossible to calculate every single combination of 3 points. We have selected 3 extreme cases (when goal state is above current state) to demonstrate. As the result shows, all of these extreme cases obey the triangle inequality, thus it is logical to assume that this hold for all points on the map.

ex1. current point height: 1, intermediate point: 1, end point: 255, max distance: 2

$$h(n) \leq h(n') + c(n,a,n') \text{—>} \frac{2}{256} \leq \frac{1}{256} + \frac{1}{256}$$

ex2. current point height: 1, intermediate point: 255, end point: 255, max distance: 2

$$h(n) \leq h(n') + c(n,a,n') \text{—>} \frac{2}{256} \leq \frac{255}{256} + \frac{1}{256}$$

ex3. current point height: 255, intermediate point: 255, end point: 255, max distance: 2

$$h(n) \leq h(n') + c(n,a,n') \text{—>} \frac{2}{256} \leq \frac{255}{256} + \frac{255}{256}$$

ex4.

S(255)	Next(100)				E(0)

$$h(n) \leq h(n') + c(n,a,n') \text{—>} 0.5 * 5 \leq \frac{255}{101} + 0.5 * 4$$

ex5.

S(1)			
N(1)	E(1)		

$$h(n) \leq h(n') + c(n,a,n') \text{—>} 0.5 * 1 \leq 0.5 + 0.5 * 1$$

ex6.

S(100)			
N(255)	E(10)		

$$h(n) \leq h(n') + c(n,a,n') \text{—>} 0.5 * 1 \leq \frac{100}{256} + 0.5 * 1$$