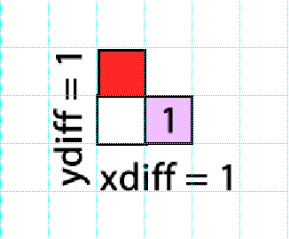
Casey Wilson, Anh Le

**Distance Formula**

We all know that the shortest distance between 2 points in 2D space can be calculate by the **distance formula:**   
 √( (x*goal* - x*start*)^2 + (y*goal* - y*start*)^2).   
However this formula overestimates the true cost by a factor of approximate √(1^2+1^2) = √2. It is an **overestimate**!

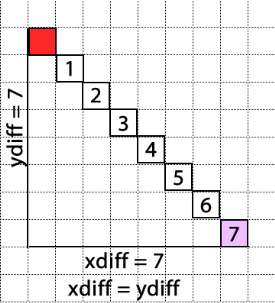
**Heuristic for 8-way Movement**

To **underestimate the distance** from a certain node to the goal we use our own method to calculate the **true shortest distance**

1. Get the horizontal difference abs(x*goal* – x*start*) = x*diff*

2. Get the vertical difference abs(y*goal* – y*start*) = y*diff*

3. If (x*diff* < y*diff*) then distance = ydiff.Else if (x*diff* >= y*diff*) then distance = xdiff

How it really works: 

* Moving diagonally until x*currrent* = x*goal* if y*diff*  > x*diff* then moving vertically from there to the goal.
* Else moving diagonally until y*currrent* = y*goal* if y*diff*  < x*diff* then moving horizontally from there to the goal. Else if y*diff*  = x*diff* then just move diagonally to the goal.

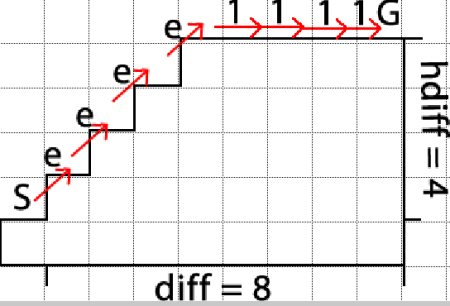
**Heuristic for (h2)/(h1+1)**

Since we already **underestimate the distance** by assume cost of moving is always equal to 1, we want to create a heuristic that always **underestimate the true cost**. To do this we multiply **distance \* c**, where c is a constant.

We came up with the value **c = 0.5** by considering one of the best possible scenarios for reaching the goal state. This is that you can drop to a height of 0 in your first step and then **alternate each next step going up to 1 and then back down to 0**. This gives you the following pattern for the cost function: 0 + 1 + 0 + 1 + 0 + … which you can average to 0.5 for each step.

**Heuristic for e^(h2-h1)**

1. Calculate the hdiff =hgoal – hcurrent.
2. If hdiff < 0, hgoal < hcurrent, moving downward, cost = hdiff \* e^-1 + distance – abs(hdiff).
3. Else if hdiff > 0, hgoal > hcurrent, moving upward, cost = hdiff \* e^1 + distance – abs(hdiff).
4. Else if hdiff = 0, hgoal = hcurrent, moving straight, cost = distance.



We already **underestimate** **the distance** using true shortest distance. To **underestimate the cost**, we want to **minimize** the **bigger cost** movement between **two possible best costs** which can be assigned to a scheme, one of which is:

going straight**, 1** (e^0).

moving upward, **e** (e^1). Proof: e^1 + e^1 = 5.4 < e^2 + 1 = 8.4.

moving downward, **1/e** (e^-1). Proof: e^-1 + e^-1 = 0.7<1 + e^-2 = 1.4.

Move up e by hdiff times, then go straight (1) \* the rest distance(distance - hdiff). **The bigger cost** **e** is minimized. Move down **e^-1** by hdiff times, then go straight (1) \* the rest distance(distance - hdiff). **The bigger cost** **1** is minimized. The two special cases where rest distance is 0 (distance = hdiff) are also covered in these scheme 1\*0 = 0. The last scheme is going straight, only return the distance times the cost of going straight (\*1). Since the heuristic always aim for the **least possible cost** in a moving scheme, it **always underestimate true cost** and thus **admissible**.