Link to the PyNotebook:
https://colab.research.google.com/drive/1FcSPYQntN5AhQ1C57yexBeXSPO3NnE7D?usp=sharing

Designing Admissible Heuristics (12 Points)

Assuming the cost function for chess movement (all eight neighbors), create an admissible heuristic, define it as a mathematical equation, and prove/show it is admissible. This is one heuristic for each cost function, 6 points each.

Steps=Chebyshev distance between two points, calculated by max(dx, dy)

Exponential:

H(n)=

0 if steps is zero, reaching the goal

Steps*1 when start height equals to the end height

Steps*e when start height is smaller than the end height and steps is smaller than or equals to the difference between heights

(end_height - start_height)*e + (steps-(end_height-start_height))*1

when start height is smaller than the end height and steps is larger than the difference between heights

large_steps*e^(-large_difference)+small_steps*e^(-small_difference)

where,

large_steps= (start_height-end_height)%steps;

small_steps=steps=large_steps;

small_difference=(start_height-end_height)/steps;

large_difference=small_difference+1;

when start height is larger than the end height

The reason why it is admissible is it is calculated based on the optimal solution. If the end height equals to the start height, we want to go as equal height as possible. If the end height is larger than the start height, we want to go one height each step and moves end height minus start height steps. If the end height is larger than the start height, we want to decrease the height as average as possible. That is because:

If h2 is the middle point of h0 and h1, then we want to minimize e^(h2-h0)+e^(h0-h2). By calculating the derivative of the equation, we get h'(h2)=e^(h2-h0)-e^(h0-h2)=0. Then h2-h0=h0-h1, and h2=(h0+h1)/2.

Div Functions:

Start is the height of the start point; end is the height of the end point.

H(n)=0 if steps=0;

start/(end+1) if steps=1

h(h2) when h2=closest_integer(math.sqrt(start*(end+1))-1) if steps=2

By calculating the derivative of h function, h(h2)=h0/(h2+1)+h2/(h1+1), we get h'(h2)=-h0/(h2+1)^2+1/(h1+1)=0. Then h(h2) is smallest when h2=closest_integer(math.sqrt(h0*(h1+1))-1).

Then we can calculate h(h2) by substitute the value of h2 into h(h2).

If the steps is larger than 3, we can first calculate the multiple of each step.

Multiple=(start/(end+1))^(1/steps)

Then we can get our list of number by appending 1/steps of the previous number.

For example, if start=128, end =1, and steps=3. Then multiple=(2/128)^1/3=1/4.

We can generate the list: [128, 32, 8, 1].

We calculate the heuristic by sigma(list[i]/(list[i+1]+1), where i is from 0 to len(list)-1.

In this example, h(n)=128/33+32/9+8/2=11.43.

The reason why I do like that is because each number's square almost equals to its previous number times its next number, which is similar with h2^2=h0*(h1+1), where h0->h2->h1.

Climbing Mt. St. Helens - A-Star Variants (8 Points)

As part of our coming demonstration, we will have a hiker use the Skamania II to climb to the top of Mt. St. Helens, located here in Skamania County using the "exp" cost function. It is crucial that during this demonstration, hikers can get the optimal route quickly, so we want you to modify your A-Star approach from above to use an A-Star variant of your choosing. As long as you can demonstrate that this variant poses some benefit (eg more memory efficiency, faster) compared to the vanilla A* algorithm you implemented previously, that's fine. You are free to use any variant you chose, and recommend the following resource. The optimal path cost for this route is ~515.81.

Implement your variant and discuss the costs and benefits of using this over vanilla A*.

The cost of A Star Variants is that the result will not be as correct as A*. From the result of the code, It can be found that if using A*, the result is 578.89. However, the result is 515.29. That is because by using a general weights multiplying the heuristic, it will be bias for some special situations.

The benefits is that it will take less time than A*. Because of adding the weight to the heuristic, which is larger than 1, h(n) will plays a more crucial role and will definitely choose less node and save time.

Time of A* exp: 8.30s;

Time of MSH: 2.57s.

Nodes of A* exp: 119128.

Nodes of MSH: 8934