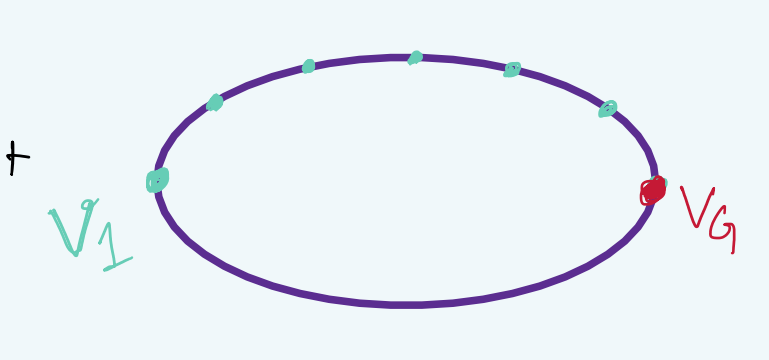
Assignment 2

Thursday, February 18, 2021 6:46 PM

Anushree Bajpai ab2167
Pavan Kumar Velaga pv217

Ans 1:

BFS for cycle with n nodes (tree search)



→ Since there are n nodes the node $n/2$ will be farthest point from v_1. For each point in the cycle there are two branches i.e. one on right and left.

$\therefore b = 2, \ d = n/2$

worst case of number of elements added on Queue is $O(b^d) \Rightarrow O(2^{n/2})$

BFS graph search for cycle with n nodes:

→ Since there are n nodes in the cycle and the farthest point from v_1 is $v_{n/2}$. In a graph search each point is visited only once as visited list is maintained. So, as the search starts from v_1 there are two branches i.e. v_2 and v_{-1}, where both v_2, v_{-1} traverse through all the points without repetition and finally reaches $n/2$. So in the process we have visited $n/2$ nodes from either end.

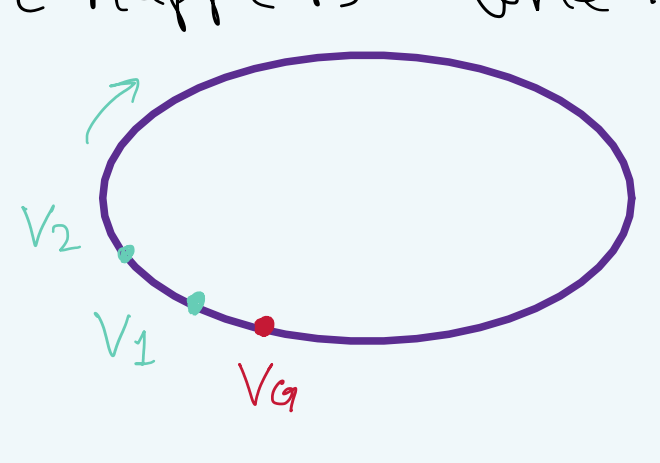
$\therefore$ total number nodes added to Queue is maximum of $n/2 + n/2 = n$

$\therefore$ total nodes added is order $O(n)$

DFS tree search for cycle with n nodes:

Given there are n nodes in cycle DFS tree seach starts searching from v_1 (goes left from v_1) and then the tree expands in the same direction until it reches the goal state. So, the maximum number of nodes the tree would visit would be $n-1$ (happens when goal is v_G)

$\therefore O(n)$



DFS graph search for cycle with n nodes:

Given there are n nodes in cycle DFS graph seach works similar to tree search starts from v_1 (goes left from v_1) and then the tree expands in the same direction until it reches the goal state. So, the maximum number of nodes the tree would visit would be $n-1$ (happens when goal is v_G)

$\therefore O(n)$

DFS graph search for IDS:

In infinite dimond strip given in the question, nodes on the left expand prior to nodes on the right. As we are using DFS graph search, the graph continues to expand to the left of starting node. Given the goal state is on right side of intial state the graph will never reach the goal state. therefore DFS graph expands infinitly to its left.

$\therefore$ total nodes added to queue $= O(\infty)$

DFS tree search for IDS:

In infinite dimond strip given in the question, nodes on the left expand prior to nodes on the right. The tree continues to expand to the left of starting node. Given the goal state is on right side of intial state the graph will never reach the goal state. therefore DFS graph expands infinitely to its left.

$\therefore$ total nodes added to queue $= O(\infty)$

BFS tree search for IDS:

In BFS tree search we add all the elements into FIFO queue. In tree search we visit nodes multiple times and they are added to queue multiple times.

$\therefore$ total nodes visited : $O(b^d)$

$\Rightarrow O(4^d)$

we know $6d = 2m$,

$\therefore d = 2m/6$

$\therefore$ total nodes added to queue $= O(4^{m/3})$

BFS graph search for IDS:

In BFS graph search we donot visit any node multiple times as we maintain visited list. i.e. the graph is expected to travese on both the directions at the same rate. However, the graph expand first on left but it simentaneusly expand even on right. If goal is at depth d from initial point then we need to travese in the order of $O(n)$

$\therefore$ total nodes added to queue $= O(n)$

Ans 2) $f(n) = (2-w)g(n) + wh(n), \quad 0 \leq w \leq 2$

The values of w for which $f(n)$ represents an algorithm that will be optimal is –

(i) When $w = 1$, $f(n) = g(n) + h(n)$ which is the evaluation function of A^* algorithm

A^* has the properties –

- the tree search version of A^* is optimal if $h(n)$ is admissible
- the graph search version of A^* is optimal if $h(n)$ is consistent.

Hence, when $w = 1$ and h is admissible, for the case of TREE-SEARCH, A^* is guaranteed to be optimal.

(ii) For $w = 0$, $f(n) = 2g(n)$
This is Uniform Cost Search as the evaluation function only depends on the cost-to-come $g(n)$.

Hence, when $w = 0$, it is Uniform Cost Search which is always guaranteed to find the optimal path.

(iii) The algorithm is guaranteed to be optimal in the range $0 \leq w \leq 1$. Multiplying $g(n)$ with a value between 1 and 2 will not impact the order of traversed path. Keeping w between 0 and 1 will not lead the heuristic to overestimate the cost.

The search performed by the algorithm following the evaluation function $f(n) = (2-w)g(n) + wh(n)$ is –:

- $w = 0$ → Uniform Cost Search $f(n) = 2g(n)$
- $w = 1$ → A^* search $f(n) = g(n) + h(n)$
- $w = 2$ → Greedy Best-First Search $f(n) = h(n)$

Greedy Best First Search tries to expand the node closest to the goal state. It evaluates the nodes just on the basis of heuristic function i.e. $f(n) = h(n)$.

Ans 3) Heuristic function is an estimated value of the cost-to-go function.

- Admissible heuristic : A heuristic is admissible if it never overestimates the cost to reach the goal. A heuristic $h(n)$ is admissible if for every node n, $h(n) \leq h^*(n)$ where $h^*(n)$ is the true cost of reaching the goal state from n.
- Consistent heuristic : A heuristic is consistent if for every node n and every successor n' of n generated by any action a, the estimated cost of reaching the goal from n is no greater than the step cost of getting to n' plus the estimated cost of reaching the goal from n': $h(n) \leq c(n, a, n') + h(n')$

(i) $h_{min}(n) = \min\{h_1(n), h_2(n)\}$

- Given that $h_1(n)$ and $h_2(n)$ are admissible, we know that $h_1(n) \leq h^*(n)$ and $h_2(n) \leq h^*(n)$.
The minimum out of h_1 and h_2 is also admissible as it is either h_1 or h_2 and we already know they are both less than or equal to the true cost $h^*(n)$.
$\therefore h_{min}(n)$ is admissible.

- Given that $h_1(n)$ and $h_2(n)$ are consistent, we know that $h_1(n) \leq c(n, n') + h_1(n')$ and $h_2(n) \leq c(n, n') + h_2(n')$.

$$\begin{aligned} h_{min}(n) &= \min(h_1(n), h_2(n)) \\ &\leq \min(h_1(n') + c(n,n'), h_2(n') + c(n,n')) \\ &\leq \min(h_1(n'), h_2(n')) + c(n,n') \\ &\leq h_{min}(n') + c(n,n') \end{aligned}$$

$\therefore h_{min}(n)$ is consistent

(ii) $h_{max} = \max\{h_1(n), h_2(n)\}$

- Given that $h_1(n)$ and $h_2(n)$ are admissible, we know that $h_1(n) \leq h^*(n)$ and $h_2(n) \leq h^*(n)$.
The maximum out of h_1 and h_2 is also admissible as it is either h_1 or h_2 and we already know they are both less than or equal to the true cost $h^*(n)$.
$\therefore h_{max}(n)$ is admissible.

- Given that $h_1(n)$ and $h_2(n)$ are consistent, we know that $h_1(n) \leq c(n, n') + h_1(n')$ and $h_2(n) \leq c(n, n') + h_2(n')$.

$$\begin{aligned} h_{max}(n) &= \max(h_1(n), h_2(n)) \\ &\leq \max(h_1(n') + c(n,n'), h_2(n') + c(n,n')) \\ &\leq \max(h_1(n'), h_2(n')) + c(n,n') \\ &\leq h_{max}(n') + c(n,n') \end{aligned}$$

$\therefore h_{max}(n)$ is consistent

iii) $h_{lin}(n) = wh_1(n) + (1-w)h_2(n), \quad 0 \leq w \leq 1$

When $w = 0$, $h_{lin} = h_2(n)$
$w = 1$, $h_{lin} = h_1(n)$

The range of $h_{lin}(n)$ is $\min(h_1(n), h(n))$ to $\max(h_1(n), h_2(n))$.

Given that $h_1(n)$ and $h_2(n)$ are admissible we have already proved that $\min(h_1(n), h_2(n))$ and $\max(h_1(n), h_2(n))$ are admissible.
$\therefore h_{lin}$ is admissible.

We know that $\max(h_1(n), h_2(n))$ is consistent.
The maximum value of h_{lin} is $\max(h_1(n), h_2(n))$.

$$\begin{aligned} h_{lin}(n) &\leq h_{max}(n) \\ &\leq h_{max}(n') + c(n,n') \quad \text{— (1)} \end{aligned}$$

$$h_{lin}(n') \leq h_{max}(n') \quad \text{— (2)}$$

From (1) and (2) $h_{lin}(n) \leq h_{lin}(n') + c(n, n')$

$\therefore h_{lin}(n)$ is consistent.

Ans 4) h_1 : number of misplaced tiles
h_2 : Manhattan distance

8 Puzzle Problem :

- Using h_1 heuristic –
When we move any tile to change state, the heuristic can change in the following ways :

- If the tile reached its correct position, h_1 will decrease by 1.
- If the tile went from its correct to an incorrect position, it will increase by 1.
- If the tile goes to another incorrect position, h remains unchanged.

For consistency,
$h(n) \leq C(n, n') + h(n')$
In our case
$h_1(n) \ ? \ 1 + h_1(n')$

$\because$ At most the value of $h_1(n')$ will be +1 or -1 the value of $h_1(n)$. So we can write
$h_1(n) \leq 1 + (h_1(n) - 1)$

The left side value will be less than or equal to the right side value $\therefore h_1$ is consistent.

- Using h_2 heuristic -

When we move any tile to change state, the heuristic can change in the following ways :

- If the tile moved in a correct direction, h_1 will decrease by 1.
- If the tile moved in an incorrect direction, h_2 will increase by 1.

So cost will only change (increase / decrease) by 1.

$h_2(n) \ ? \ c(n, n') + h_2(n')$

$h_2(n) \leq 1 + (h_2(n') - 1)$

The left side value will be less than or equal to the right side value $\therefore h_2$ is consistent.

Ans 5)

State 1:

3	Q	3	3
Q	3	5	3
3	4	3	Q
3	3	Q	3

4 attacking pairs

↓

State 2

Q	Q	4	3
4	2	4	1
3	3	4	Q
3	2	Q	3

Move Q1 from (2, 1) to (1, 1)

3 attacking pairs

↓

State 3

Q	Q	4	3
4	2	3	Q
0	1	2	3
2	1	Q	3

Move Q_4 from (3,4) to (2,4)

1 attacking pair

↓

State 4

0	Q	0	0
0	0	0	Q
Q	0	0	0
0	0	Q	0

Move Q_1 from (1,1) to (3,1)

0 attacking pairs

Goal state