Assign2

Assignment 2 – Heuristics and Search

z5340468 Ziyao Lu

Question 1: Search Strategies for the 15-Puzzle:

(a):

Start State	BFS	IDS	Greedy	A*
Start1	Expanded: 10978. Length: 12.	Expanded: 25121. Length: 12.	Expanded: 59182. Length: 12.	Expanded: 30. Length: 12.
Start2	Expanded: 344890. Length: 17.	Expanded: 349380. Length: 17.	Expanded: 19. Length: 17.	Expanded: 35. Length: 17.
Start3	Expanded: 641252. Length: 18.	Expanded: 1209934. Length: 18.	Expanded: 59196. Length: 22.	Expanded: 133. Length: 18.

(b):
BFS:
BFS guarantees the shortest path and can be seen in all instances to have achieved the goal with optimal length. However, it tends to expand a significant number of nodes, especially as the complexity of the start state increases. Of course, it is the way how BFS arithmetic works. It is evident from a mass of number of nodes it expanded for Start2 and Start3, indicating a high memory requirement and potentially longer run times in practice.

IDS:
Like BFS, IDS also guarantees the shortest path, which is evident from the lengths being optimal across all starts. However, it expands a vast number of nodes, even more than BFS in all cases here, which means it will achieve a best answer may be different with BFS but with more cost.

Greedy Search:
Greedy Search uses the Manhattan Distance heuristic to direct the search process, resulting in a significant reduction in the number of expanded nodes for Start2 and Start3. However, it may not be

able to guarantee the shortest path, such as in Start3, where the path length is 22 but the optimal length is 18. In general, use greedy search is a risk choice, it just has some possibility to reduce to the number of nodes expanded with a shortest path.

A* search:
A* Search is more balance than BFS, IDS or Greedy Search when finding the shortest path while also being more efficient in terms of nodes expanded, because it considers both the cost to reach the current node and the heuristic estimate. The table above shows that A* expanded far fewer nodes than BFS and IDS for all starts while maintaining the optimal path length, which proves its efficacy in both node expansion and path optimality.

Question 2: Heuristic Path Search for 15-Puzzle

(a):

We assume the A* Search is optimal, so $f_w(n) = g(n) + h(n)$, and $h(n)$ is admissible which means $h'(n) \leq h(n)$ for all n.
When $n = 1$, $f_w(n) = (2 - w)g(n) + wh(n) = f_w(n) = g(n) + h(n)$, it is A* Search which means this function is optimal when $w = 1$.
When $0 \leq w < 1$, $f_w(n) = (2 - w)g(n) + wh(n)$ is changing the weight for $g(n)$ and $h(n)$, the value of w reduces the dependence on $h(n)$, while increase the dependence on $g(n)$. As $g(n)$ is the path cost, and $h(n)$ is the heuristic guidance, the algorithm increasingly favors the actual path cost g(n) over the heuristic estimate h(n) when w is reduce, and we assume A* Search is optimal, so h(n) is admissible and never overestimates the true cost here, so the search remains optimal.

(b):

	Start4		Start5		Start6	
IDA* SEARCH	45	545120	50	4178819	56	169367641
HPS, w = 1.1	47	523052	54	857155	58	13770561
HPS, w = 1.2	47	29761	56	64522	60	265672
HPS, w = 1.3	55	968	62	5781	68	9066
HPS, w = 1.4	65	9876	70	561430	80	37869

(c):
When w = 1, the search strategy is a standard A* search. At this point the heuristic function h(n) provides an estimate of the cost to reach the goal and we assume it is not overestimate the cost. It balances the weight between the actual path cost g(n) and the heuristic estimate h(n).
As w increases from 1 to 1.3, the influence of h(n) becomes stronger, biasing the search more towards

the heuristic's guidance. This means the search would progress more directly towards the goal, theoretically reducing the number of nodes expanded because it avoids paths with higher estimated heuristic costs, thus enhancing search efficiency.
However, when w=1.4, the number of expanded nodes increases, which might be due to excessive optimism heuristic. Overweighting may lead to heuristics emphasizing estimated costs while neglecting actual path costs, causing the search to deviate from more expensive paths. So at this time the arithmetic does not care the cost too much and may lead to a wrong way and expand more nodes.

Question 3: Graph Paper Grand Prix

(a):
when $n = 1$: $[+, -]$
start : $v = 0, c = 0$
step1: + $v = 1, c = 1$
step2: - $v = 0, c = 1$
so $M(1, 0) = 2$

when $n = 2$: $[+, o, -]$
start : $v = 0, c = 0$
step1: + $v = 1, c = 1$
step2: o $v = 1, c = 2$
step3: - $v = 0, c = 2$
so $M(2, 0) = 3$

when $n = 3$: $[+, o, o, -]$
start : $v = 0, c = 0$
step1: + $v = 1, c = 1$
step2: o $v = 1, c = 2$
step3: o $v = 1, c = 3$
step4: - $v = 0, c = 3$
so $M(3, 0) = 4$

when $n = 4$: $[+, o, o, o, -]$
start : $v = 0, c = 0$
step1: + $v = 1, c = 1$
step2: o $v = 1, c = 2$
step3: o $v = 1, c = 3$
step4: o $v = 1, c = 4$
step5: - $v = 0, c = 4$
so $M(4, 0) = 5$

when $n = 5$: $[+, +, -, o, -]$
start : $v = 0, c = 0$
step1: + $v = 1, c = 1$
step2: + $v = 2, c = 3$
step3: - $v = 1, c = 4$
step4: o $v = 1, c = 5$
step5: - $v = 0, c = 5$
so $M(5, 0) = 5$

when $n = 6$: $[+, +, -, o, o, -]$
start : $v = 0, c = 0$
step1: + $v = 1, c = 1$
step2: + $v = 2, c = 3$
step3: - $v = 1, c = 4$
step4: o $v = 1, c = 5$
step5: o $v = 1, c = 6$
step6: - $v = 0, c = 6$
so $M(6, 0) = 6$

when $n = 7$: $[+, +, o, -, o, -]$
start : $v = 0, c = 0$
step1: + $v = 1, c = 1$
step2: + $v = 2, c = 3$
step3: o $v = 2, c = 5$
step4: - $v = 1, c = 6$
step5: o $v = 1, c = 7$
step6: - $v = 0, c = 7$
so $M(7, 0) = 6$

when $n = 8$: $[+, +, o, o, -, -]$
start : $v = 0, c = 0$
step1: + $v = 1, c = 1$
step2: + $v = 2, c = 3$
step3: o $v = 2, c = 5$
step4: o $v = 2, c = 7$
step5: - $v = 1, c = 8$
step6: - $v = 0, c = 8$
so $M(8, 0) = 6$

when $n = 9$: $[+, +, +, -, -, -]$
start : $v = 0, c = 0$
step1: + $v = 1, c = 1$
step2: + $v = 2, c = 3$
step3: + $v = 3, c = 6$

step4: - $v = 2, c = 8$
step5: - $v = 1, c = 9$
step6: - $v = 0, c = 9$
so $M(9, 0) = 6$

when $n = 10$: $[+, +, +, -, -, o, -]$
start : $v = 0, c = 0$
step1: + $v = 1, c = 1$
step2: + $v = 2, c = 3$
step3: + $v = 3, c = 6$
step4: - $v = 2, c = 8$
step5: - $v = 1, c = 9$
step6: o $v = 1, c = 10$
step7: - $v = 0, c = 10$
so $M(10, 0) = 7$

when $n = 11$: $[+, +, +, -, o, -, -]$
start : $v = 0, c = 0$
step1: + $v = 1, c = 1$
step2: + $v = 2, c = 3$
step3: + $v = 3, c = 6$
step4: - $v = 2, c = 8$
step5: o $v = 2, c = 10$
step6: - $v = 1, c = 11$
step7: - $v = 0, c = 11$
so $M(11, 0) = 7$

when $n = 12$: $[+, +, +, o, -, -, -]$
start : $v = 0, c = 0$
step1: + $v = 1, c = 1$
step2: + $v = 2, c = 3$
step3: + $v = 3, c = 6$
step4: o $v = 3, c = 9$
step5: - $v = 2, c = 11$
step6: - $v = 1, c = 12$
step7: - $v = 0, c = 12$
so $M(12, 0) = 7$

when $n = 13$: $[+, +, +, o, -, -, o, -]$
start : $v = 0, c = 0$
step1: + $v = 1, c = 1$
step2: + $v = 2, c = 3$
step3: + $v = 3, c = 6$
step4: o $v = 3, c = 9$

step5: - $v = 2, c = 11$
step6: - $v = 1, c = 12$
step7: o $v = 1, c = 13$
step8: - $v = 0, c = 13$
so $M(13, 0) = 8$

when $n = 14$: $[+, +, +, o, -, o, -, -]$
start : $v = 0, c = 0$
step1: + $v = 1, c = 1$
step2: + $v = 2, c = 3$
step3: + $v = 3, c = 6$
step4: o $v = 3, c = 9$
step5: - $v = 2, c = 11$
step6: o $v = 2, c = 13$
step7: - $v = 1, c = 14$
step8: - $v = 0, c = 14$
so $M(14, 0) = 8$

when $n = 15$: $[+, +, +, o, o, -, -, -]$
start : $v = 0, c = 0$
step1: + $v = 1, c = 1$
step2: + $v = 2, c = 3$
step3: + $v = 3, c = 6$
step4: o $v = 3, c = 9$
step5: o $v = 3, c = 12$
step6: - $v = 2, c = 14$
step7: - $v = 1, c = 15$
step8: - $v = 0, c = 15$
so $M(15, 0) = 8$

when $n = 16$: $[+, +, +, +, -, -, -, -]$
start : $v = 0, c = 0$
step1: + $v = 1, c = 1$
step2: + $v = 2, c = 3$
step3: + $v = 3, c = 6$
step4: + $v = 4, c = 10$
step5: - $v = 3, c = 13$
step6: - $v = 2, c = 15$
step7: - $v = 1, c = 16$
step8: - $v = 0, c = 16$
so $M(16, 0) = 8$

when $n = 17$: $[+, +, +, +, -, -, -, o, -]$
start : $v = 0, c = 0$

step1: + $v = 1, c = 1$
step2: + $v = 2, c = 3$
step3: + $v = 3, c = 6$
step4: + $v = 4, c = 10$
step5: - $v = 3, c = 13$
step6: - $v = 2, c = 15$
step7: - $v = 1, c = 16$
step8: o $v = 1, c = 17$
step9: - $v = 0, c = 17$
so $M(17, 0) = 9$

when $n = 18$: $[+, +, +, +, -, -, o, -, -]$
start : $v = 0, c = 0$
step1: + $v = 1, c = 1$
step2: + $v = 2, c = 3$
step3: + $v = 3, c = 6$
step4: + $v = 4, c = 10$
step5: - $v = 3, c = 13$
step6: - $v = 2, c = 15$
step7: o $v = 2, c = 17$
step8: - $v = 1, c = 18$
step9: - $v = 0, c = 18$
so $M(18, 0) = 9$

when $n = 19$: $[+, +, +, +, -, o, -, -, -]$
start : $v = 0, c = 0$
step1: + $v = 1, c = 1$
step2: + $v = 2, c = 3$
step3: + $v = 3, c = 6$
step4: + $v = 4, c = 10$
step5: - $v = 3, c = 13$
step6: o $v = 3, c = 16$
step7: - $v = 2, c = 18$
step8: - $v = 1, c = 19$
step9: - $v = 0, c = 19$
so $M(19, 0) = 9$

when $n = 20$: $[+, +, +, +, o, -, -, -, -]$
start : $v = 0, c = 0$
step1: + $v = 1, c = 1$
step2: + $v = 2, c = 3$
step3: + $v = 3, c = 6$
step4: + $v = 4, c = 10$
step5: o $v = 4, c = 14$

step6: - $v = 3, c = 17$
step7: - $v = 2, c = 19$
step8: - $v = 1, c = 20$
step9: - $v = 0, c = 20$
so $M(20, 0) = 9$

when $n = 21$: $[+, +, +, +, o, -, -, -, o, -]$
start : $v = 0, c = 0$
step1: + $v = 1, c = 1$
step2: + $v = 2, c = 3$
step3: + $v = 3, c = 6$
step4: + $v = 4, c = 10$
step5: o $v = 4, c = 14$
step6: - $v = 3, c = 17$
step7: - $v = 2, c = 19$
step8: - $v = 1, c = 20$
step9: o $v = 1, c = 21$
step10: - $v = 0, c = 21$
so $M(21, 0) = 10$

(b):
the value of `n` into three cases, which explain the number of steps needed based on the proximity of `n` to perfect squares:

1. $2s + 1$ steps for numbers just above a perfect square s^2 with a difference of `k (where $1 \leq k \leq s$), indicating one extra step 'o' is needed after acceleration.
2. $2s + 2$ steps for numbers that exceed s^2 by more than s, but not enough to be the next perfect square $(s + 1)^2$. This accounts that add an acceleration is too more, so we need to add anyother step 'o' after acceleration.
3. $2s + 2$ steps for the next perfect square $(s + 1)^2$, now we just have enough length to add another acceleration, so it needs $s + 1$ steps to accelerate to the target and $s + 1$ steps to decelerate to a stop.

so $M(n, 0) = \lceil 2\sqrt{n} \rceil$

(c):
In this case, it start with speed k, so we first need to know how far will it move when we add speed from 0 to k, this is obviously an arithmetic sequence, the first element is 1, and the the last one is k with k elements, so the sum is $\frac{1}{2}k(k + 1)$. Second if we keep reduce speed at speed k, the length will also be an arithmetic sequence, the first element is k-1, and the the last one is 0 with k elements, so the sum is $\frac{1}{2}k(k - 1)$, as $n \geq \frac{1}{2}k(k - 1)$, we can stop at n while speed is 0.
As we assume the result from part(b), so now if we start with speed is 0, the n' will be $n' = n + \frac{1}{2}k(k + 1)$.

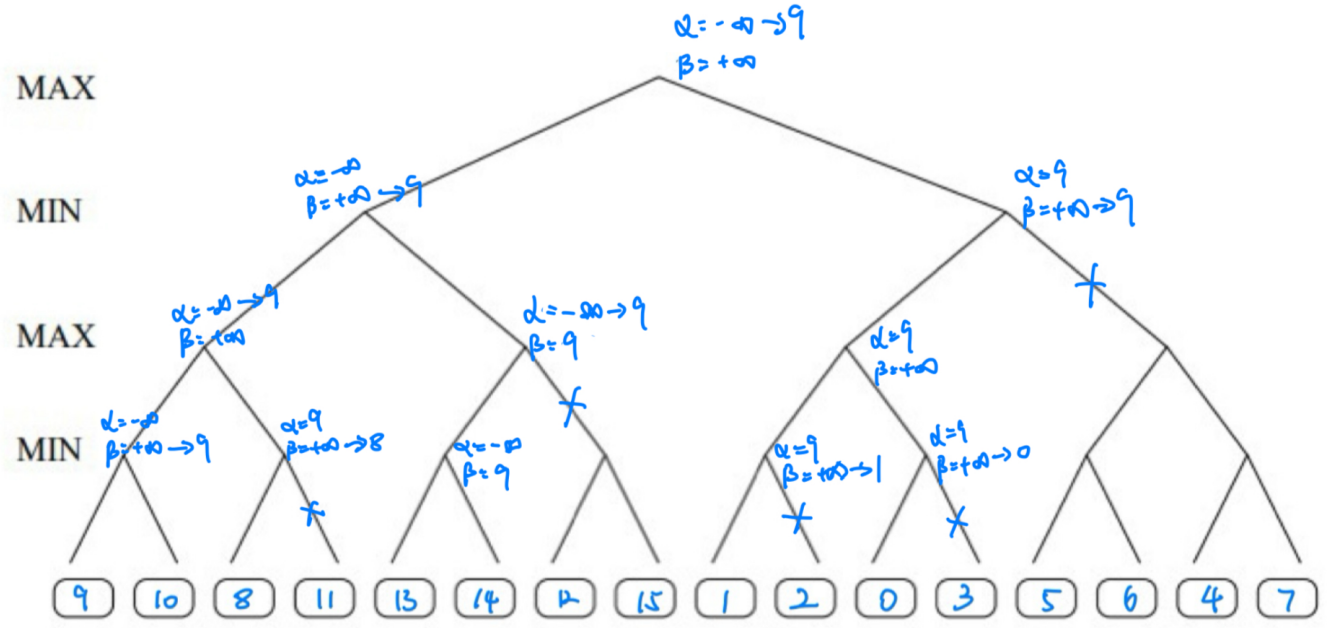
According to part(b), we can get $M(n,0) = \lceil 2\sqrt{n + \frac{1}{2}k(k+1)} \rceil$.
And we can know that, the speed goes from 0 all the way up to k is obviously going to take k steps, so $M(n,k) = \lceil 2\sqrt{n + \frac{1}{2}k(k+1)} \rceil - k$

(d):
As we calculate if we keep reduce speed, it will still need $\frac{1}{2}k(k-1)$ to stop and now n \le \frac{1}{2}k(k-1)$ means we can't stop at n with speed in 0.
So, we need to reverse back to n after we pass it.
First we consider the reverse part. when it stopped, the distance will be $\frac{1}{2}k(k-1)$, and we need to go back to n. Then we need a answer for M($\frac{1}{2}k(k-1) - n$, 0). According to part(b), $M(\frac{1}{2}k(k-1) - n, 0) = \lceil 2\sqrt{\frac{1}{2}k(k-1) - n} \rceil$. Then reduce speed from k to 0 need k steps, so the sum will be $M(n,k) = \lceil 2\sqrt{\frac{1}{2}k(k-1) - n} \rceil + k$

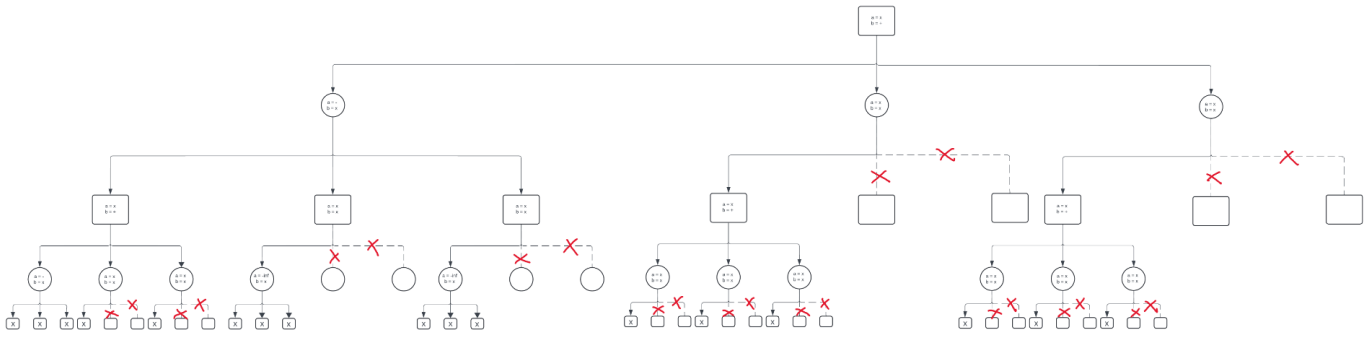
(e):
$h(r, c, u, r_G, c_G) = max(M(d_r, k_r), M(d_c, k_c))$
$d_r = \mid r_G - r \mid$ means the distance from the current position to the target position in the row direction (vertical direction)
$d_c = \mid c_G - c \mid$ means the distance from the current position to the target position in the column direction (horizontal direction)
k_r is the current velocity component in the row direction
k_c is the current velocity component in the column direction
$M(n,k)$ is the function we defined earlier to calculate the minimum number of time steps needed to reach and stop at position n in a one-dimensional case starting at position 0 with an initial velocity of k.

Question 4: Game Trees and Pruning

(a, b):



(c):



17 leaves will be evaluated.

(d):

In the best case, the pruning effect is maximized by always considering the best moves first. After checking the first (best) move, all subsequent moves at that level may be pruned based on the alpha and beta values established by the best move as $\alpha \geq \beta$. So at each depth level, the algorithm can prune half of the tree. The results in checking only a small fraction of the total nodes, roughly the square root of the number of nodes checked by the basic minimax algorithm, so the complexity is $O(b^{\frac{d}{2}})$.