Jiashu Liao Ziru Liu
of Rutgers

Project 3

CS 440

Intro to Artificial Intelligence

Submitted to:
Pro. Kostas Bekris
Rahul Shome
Chaitanya Mitash

Contents

1 Project Report

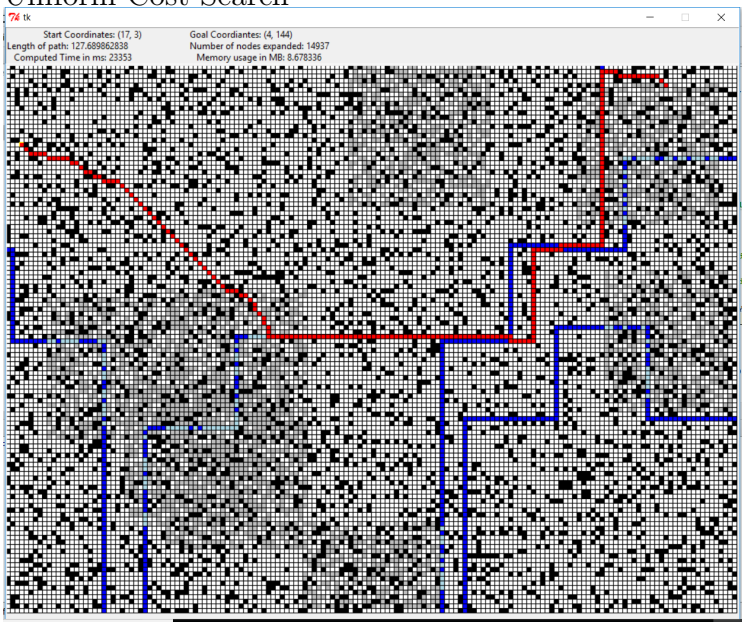
Question a:
In this assignment we create a grid world, with nodes that unblocked, partially blocked, completely blocked, or serve as highways or rivers for unblocked or partially blocked nodes. We used A* and Uniform Cost algorithms to find the shortest path from the start node to the goal node.
By coding

Question b:
We made a grid as described in the assignment description. In order to make this grid we used the TKinter library, a UI library in the Python language. We display each of the cells with a different color corresponding to its type. The colors are as follows:

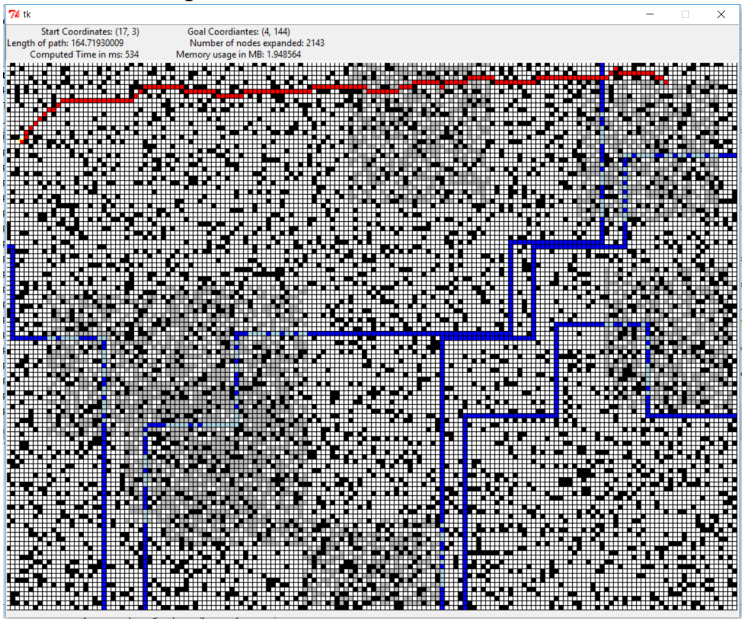
- White corresponds to unblocked nodes
- Black corresponds to blocked nodes
- Gray corresponds to partially blocked nodes
- Light Blue corresponds to highways on partially blocked nodes
- Blue corresponds to highways on unblocked nodes
- Red corresponds to path nodes

By Coding
Next Page for several example of final maps.

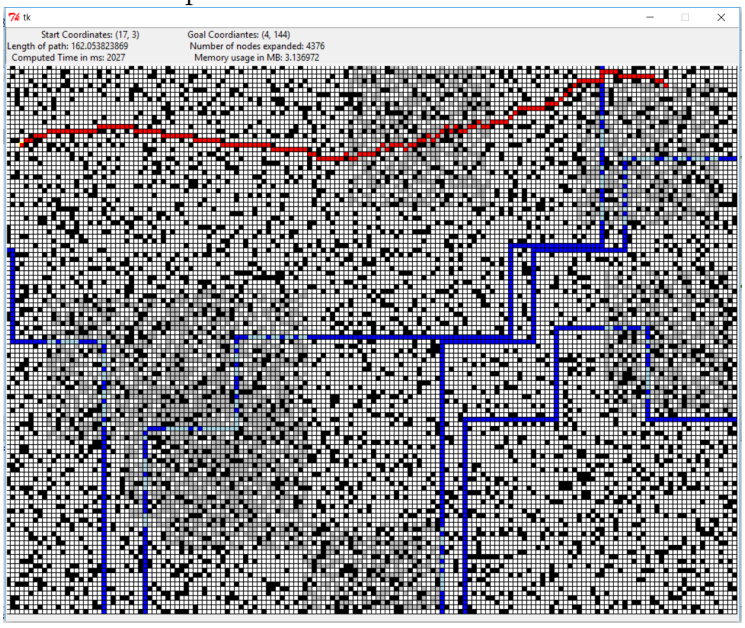
Uniform Cost Search



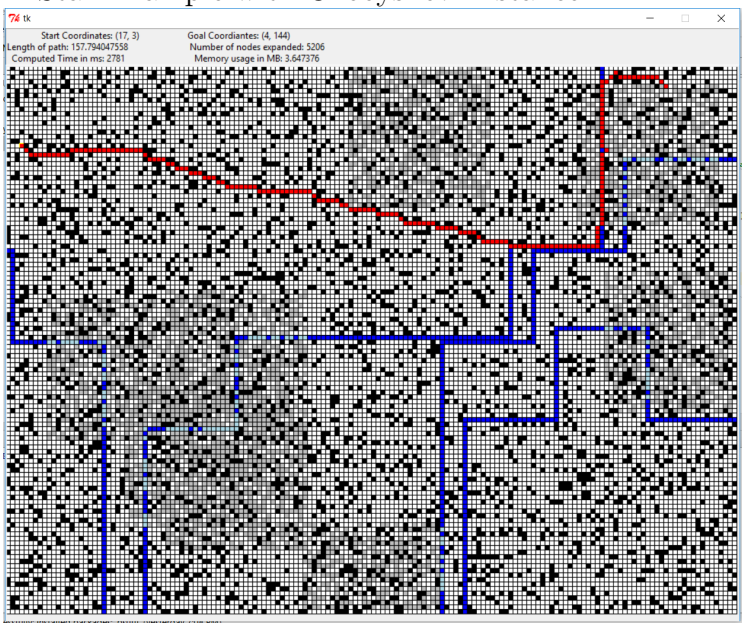
A Star Example with Manhattan Distance



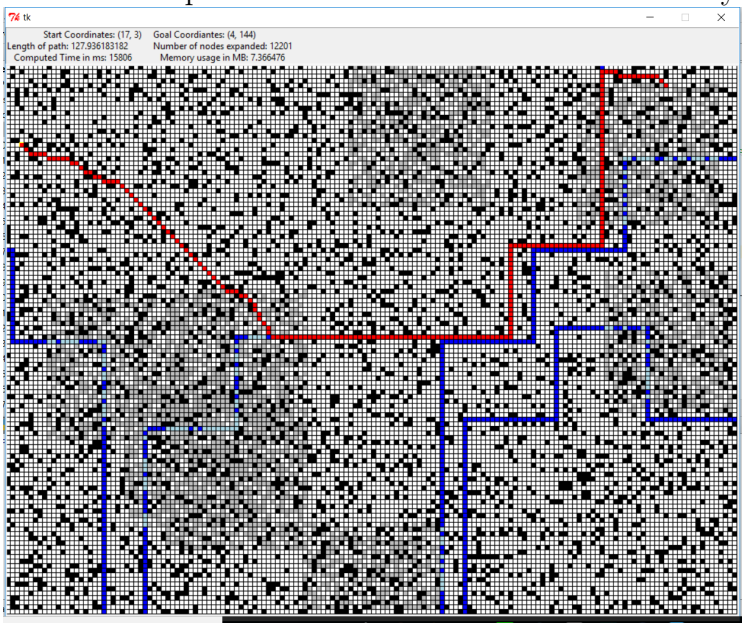
A Star Example with Eucilidian Distance



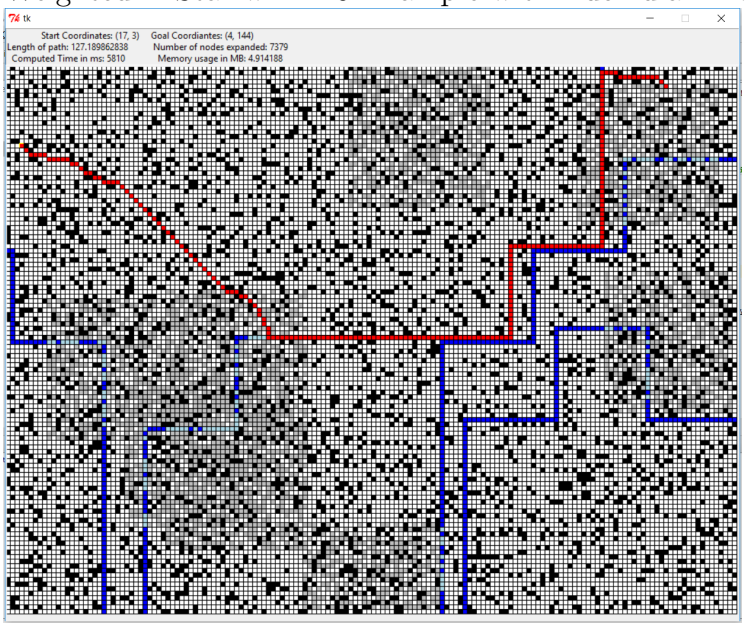
A Star Example with Chebyshev Distance



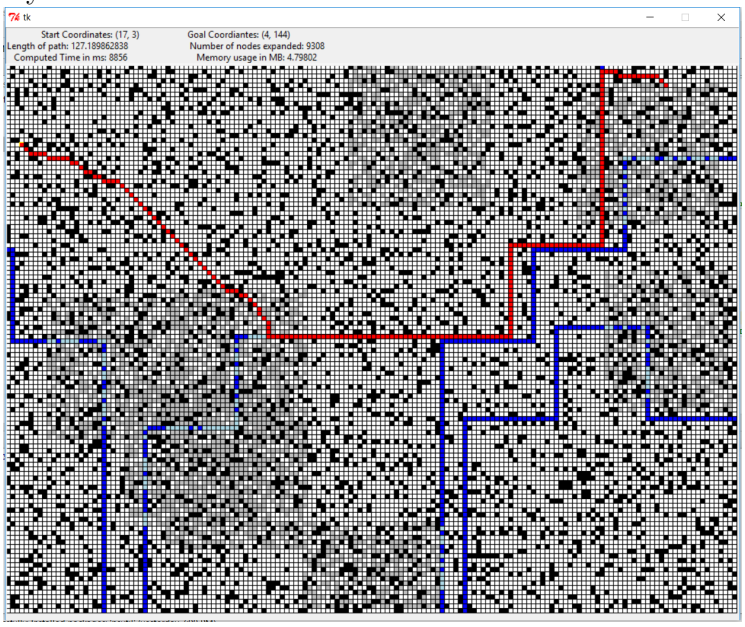
A Star Example with Eucilidian Distance Divided By 4



Weighted A Star w = 2.0 Example with Eucilidian Distance



Sequential A Star w1 = 2.0, w2 = 1.2 Example with Eucilidian Distance Divided By 4



Question c:
We code in python. The python priority queue is not efficient for coding. Therefore we used heap queue of list for Fringe. Then we can always expand the best node (greedy algorithm) and also check whether one node is in the heap queue. We used List for Closed List. For path we used node of list. We always will keep updating a Matrix of Nodes to keep the map information.

Question d:
We proposed the best admissible heuristic is Eucilidian Distance Divided By 4, because in this world we have 8 directions in every node (suppose we are not in the fringe, it can differ in different special cases). The cost of highways on unblocked or partially blocked nodes is 1/4 for the original. If we divided the Eucilidian Distance by 4, it will guide the agent to traverse the highway more potentially than the Eucilidian Distance.
We proposed five heuristic:
Manhatten Distance

$$|x_1 - x_2| + |y_1 - y_2| \tag{1}$$

Eucilidian Distance

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{2}$$

Chebyshev Distance

$$\max\left(|x_1 - x_2|, |y_1 - y_2|\right) \tag{3}$$

Octile Distance

$$|x_1 - x_2| + |y_1 - y_2| + \sqrt{min(|x_1 - x_2|, |y_1 - y_2|)} - 2 * (min(|x_1 - x_2|, |y_1 - y_2|)) \tag{4}$$

Euclidean Distance, squared

$$(\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2})^2 \tag{5}$$

Eucilidian Distance Divided By 4

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}/4 \tag{6}$$

where (x1, y1) are the coordinates of the current node and (x2, y2) are the coordinates of the goal node. In order to analyze the quality of algorithm's results, we take the time it took to compute a path from the start node to the end node, the number of nodes that were expanded and path cost for each algorithm by all proposed heuristic.

Question e:

Algorithm	Uniform Cost
Avg(LEN OF PATH)	111.8718
Avg(NODES EXPANDED)	13250.9333
Avg(TIME)	17691.5333
Avg(Memory) MB	7.7957

Algorithm	A* by Manhattan Distance	A* by Euclidean Distance	A* by Chebyshev Distance	A* by Octile Distance	A* by Euclidean Distance, squared	A* by Euclidean Distance Divided By 4
Avg(LEN OF PATH)	138.4312	122.8357	127.2734	124.135	195.4383	111.4611
Avg(NODES EXPANDED)	826.2667	1443.667	3027.4667	1003.7333	138.2667	10469.5333
Avg(TIME)	170.8	414.4	1251.6667	266.3333	50.4667	12397.8
Avg(Memory) MB	1.3619	1.7214	2.5051	1.4937	0.96	6.4137

Algorithm Weighted =1.25	Weighted A* by Manhattan Distance	Weighted A* by Euclidean Distance	Weighted A* by Chebyshev Distance	Weighted A* by Octile Distance	Weighted A* by Euclidean Distance, squared	Weighted A* by Euclidean Distance Divided By 4
Avg(LEN OF PATH)	180.523	137.3766	138.1424	144.9315	196.0983	111.5589
Avg(NODES EXPANDED)	380	311.8667	1207.5333	278.0667	138.2	9528.8667
Avg(TIME)	76.9333	71.9333	349.6	58.8667	46.9333	10831.0667
Avg(Memory) MB	1.1018	1.6865	2.4247	1.4656	0.9603	5.9288

Algorithm Weighted =2.0	Weighted A* by Manhattan Distance	Weighted A* by Euclidean Distance	Weighted A* by Chebyshev Distance	Weighted A* by Octile Distance	Weighted A* by Euclidean Distance, squared	Weighted A* by Euclidean Distance Divided By 4
Avg(LEN OF PATH)	194.8426	161.7319	165.5098	162.9606	196.1649	112.5172
Avg(NODES EXPANDED)	151.2	142	166.9333	143.3333	138.1333	6156.2
Avg(TIME)	56.2	50.8667	47.4667	52.2	47.3333	5004.2
Avg(Memory) MB	0.9764	1.7189	2.3717	1.482	0.9558	4.2071

By data above, we can find out that:
For (LEN OF PATH) part, Uniform Cost Search is the best one. For (NODES EXPANDED) part, Weighted A * (weight = 2.0) in by Heuristic Euclidean Distance, squared is the best one. For (TIME) part, Weighted A * (weight = 1.25) in by Heuristic Euclidean Distance, squared is the best one.

Question f:
Algorithms Comparison(Execution Time, Nodes Expanded):

- Uniform Cost Search: It will make sure giving an optimal cost (length of path) but it is not feasible. Since it need to expand a lot of nodes (almost the whole map if the start and goal nodes are far from each other) which means it also will take a lot time to find the path.

- A*: The execution time of A* is shorter than the execution time of uniform cost search and also the number of nodes expanded is much lower than uniform cost search. At the same time, A* with feasible heuristic will also have a feasible cost of path (length of path)

- Weighted A*: This is a greedy first algorithm (greedy for heuristic). Therefore it is the fastest algorithm because it expand more in direction towards the goal nodes, but the cost of the path might (usually) be higher than A* and Uniform Cost Search.

- With A*, Euclidean Distance, squared Distance is computed the fastest on average time. Euclidean Distance, Divided by Four Distance is the slowest on average time. Euclidean Distance has least number of node expanded on average and memory space. Euclidean Distance, Divided by Four Distance has largest number of node expanded on average and requires the most memory space. Euclidean Distance, Divided by Four Distance is the lowest average path cost. Euclidean Distance,

squared Distance is the highest average path cost.

- With Weighted A* (W=1.25), Euclidean Distance, squared Distance is computed the fastest on average time. Euclidean Distance, Divided by Four Distance is the slowest on average time. Euclidean Distance, squared Distance has least number of node expanded on average and memory space. Euclidean Distance, Divided by Four Distance has largest number of node expanded on average and memory space. Euclidean Distance, Divided by Four Distance is the lowest average path cost. Euclidean Distance, squared Distance is the highest average path cost.

- With Weighted A* (W=2.0), Euclidean Distance, squared Distance and Chebyshev Distance is computed the fastest on average time. Euclidean Distance, Divided by Four Distance is the slowest on average time. Euclidean Distance, squared Distance has least number of node expanded on average. Euclidean Distance, Divided by Four Distance has largest number of node expanded on average and memory space. Euclidean Distance, Divided by Four Distance is the lowest average path cost. Euclidean Distance, squared Distance is the highest average path cost.

- With A*, the Euclidean Distance, squared Distance heuristic took the longest,followed by Manhattan distance, Chebyshev distance, Octile Distance, the Euclidean Distance, finally Divided by Four Distance.

- We used 6 different heuristics in our project. The best admissible/consistent heuristic we chose is the Euclidean Distance, Divided by Four Distance because in order to be admissible/consistent the function can never overestimate the cost. The reason is that agent run on highways the g cost is lowered by a factor of 4, we just divided the heuristic function to match that when the agent goes on the highway.

- In A* and weighted A*, we need to get the shortest path, i.e the path with the total lowest cost. This ensured that we chose highways whenever possible, and tried not to go for partially blocked cells. The the Euclidean Distance, squared Distance, Mahatton, Octile, Chebyshev and Euclidean distance heuristics can be inadmissible. The reason for this is because they assume there are no highways. Therefore, it is possible to overestimate the cost. Consider using the x distance or y distance; if the start and goal nodes are also on the same coordinates, and there are highways, then it is possible that the “estimated” distance is greater than the actual cost of traveling along the path.

Question g:

Sequential A *	Sequential A* w1 = 2, w2 = 1.2	Sequential A* w1 = 1.8, w2 = 1.5
Avg(LEN OF PATH)	111.545868	109.8061466
Avg(NODES EXPANDED)	8479.153846	9406.692308
Avg(TIME)	9179.461538	11066.61538
Avg(Memory) MB	4.288029538	4.677442154

The admissible heuristic is Eucilidian Distance Divided By 4.
For (LEN OF PATH) part, w = 2, w2 = 1.2 is better.For (NODES EXPANDED) and (TIME) and (MEMORY) part w1 = 1.8, w2 = 1.5 is better. This is a trade off between time/expanded node and path length.

Question h:
Sequential A* search is efficient because it utilizes information from not only the admissible heuristic but also other heuristic. There is a good amount of information we have at each node by five different heuristics. Checking five heuristic and combining these results allows us to get a more accurate measure of the distance from the goal, which leads a more optimal solution.

Question i:
For any state s with $key(s,0) \leq key(u,0) \forall u \in OPEN0$, it holds that $key(s,0) \leq w1 * g * (sgoal)$.
To prove this by contradiction. Assuming $key(s,0) = g0(s) + w1 * h0(s) > w1 * g * (sgoal)$. A least cost path from start to sgoal given as $P = (s0 = start, ..., sk = sgoal)$. If we pick the first state si on the path. The state is not expanded yet by the anchor search and is part of $OPEN0(si \in OPEN0)$. The algorithm will always find such a state $si \in OPEN0, s0 = start$ is put in $OPEN0$ at the initialization. whenever any state $sj \in P$ is expanded in the anchor search $sj + 1 \in P$ is always inserted in OPEN0, $sk = sgoal$ is never expanded in the anchor search, otherwise, whenever sgoal has the least key in OPEN0 the search terminates
Examining $g0(si)$. If i = 0, we have $g0(si) = g0(sstart) = 0 \leq w1 * g * (si)$. If $i \neq 0$, by the choice of si we know that si−1 has already been expanded in the anchor search. When si−1 was chosen for expansion, we had $g0(si-1) \leq w1 * g * (si-1)$ from Theorem 1. Now, as si is a successor of si−1, we have:
$g0(si) \leq g0(si-1) + c(si-1, si)$
$\leq w1 * g * (si-1) + c(si-1, si)$
$\leq w1 * (g * (si-1) + c(si-1, si))$
$= w1 * g * (si)$
Thus, we have $g0(si) \leq w1 * g * (si)$. Using this we obtain:
$key(si, 0) = g0(si) + w1 * h0(si)$
$\leq w1 * g * (si) + w1 * h0(si)$
$\leq w1 * g * (si) + w1 * c * (si, sgoal)$
$= w1 * g * (sgoal)$
Now, as $si \in OPEN0 and key(si, 0) \leq w1 * g * (sgoal) < key(s, 0)$, we have a contra-

diction to our assumption that $key(s, 0) \leq key(u, 0), \forall u \in OPEN0$. Therefore, for any state s for which it is true that $key(s, 0) \leq key(u, 0), \forall u \in OPEN0$, , it holds that $key(si, 0) \leq w1 * g * (sgoal)$
For Sequential A * exits (in the i th search), $gi(sgoal) \leq w1 * w2 * g * (sgoal)$. Or say the solution cost obtained is bounded by w1 $*$ w2 sub-optimality factor.
We have:
$g0(sgoal) \leq w1 * g * (sgoal)$
$\leq w1 * w2 * g * (sgoal)$, As w2 ≥ 1.0
$gi(sgoal) \leq w2 * OPEN0.Minkey()$
$\leq w2 * w1 * g * (sgoal)$, proved above.
Therefore, in both the above mentioned cases, i.e., if either the anchor terminates or an inadmissible search terminates, we have the solution cost to be within w1 $*$ w2 factor of the optimal solution cost.
On the other hand, if the search terminates unsuccessfully at $OPEN0.Minkey() > \infty$, from $OPEN0.Minkey() \leq w1 * g * (sgoal)$ as condition check for other heuristic besides the default or admissible heuristic. We know $OPEN0.Minkey() \leq w1 * g * (sgoal)$ Now, $OPEN0.Minkey() \geq \infty \Longrightarrow g * (sgoal) \geq \infty$. , there is no finitecost solution