Artificial Intelligence

t

MANAS PRATIM BISWAS

Information Technology - 002011001025

1	2	3
8	0	4
7	6	5

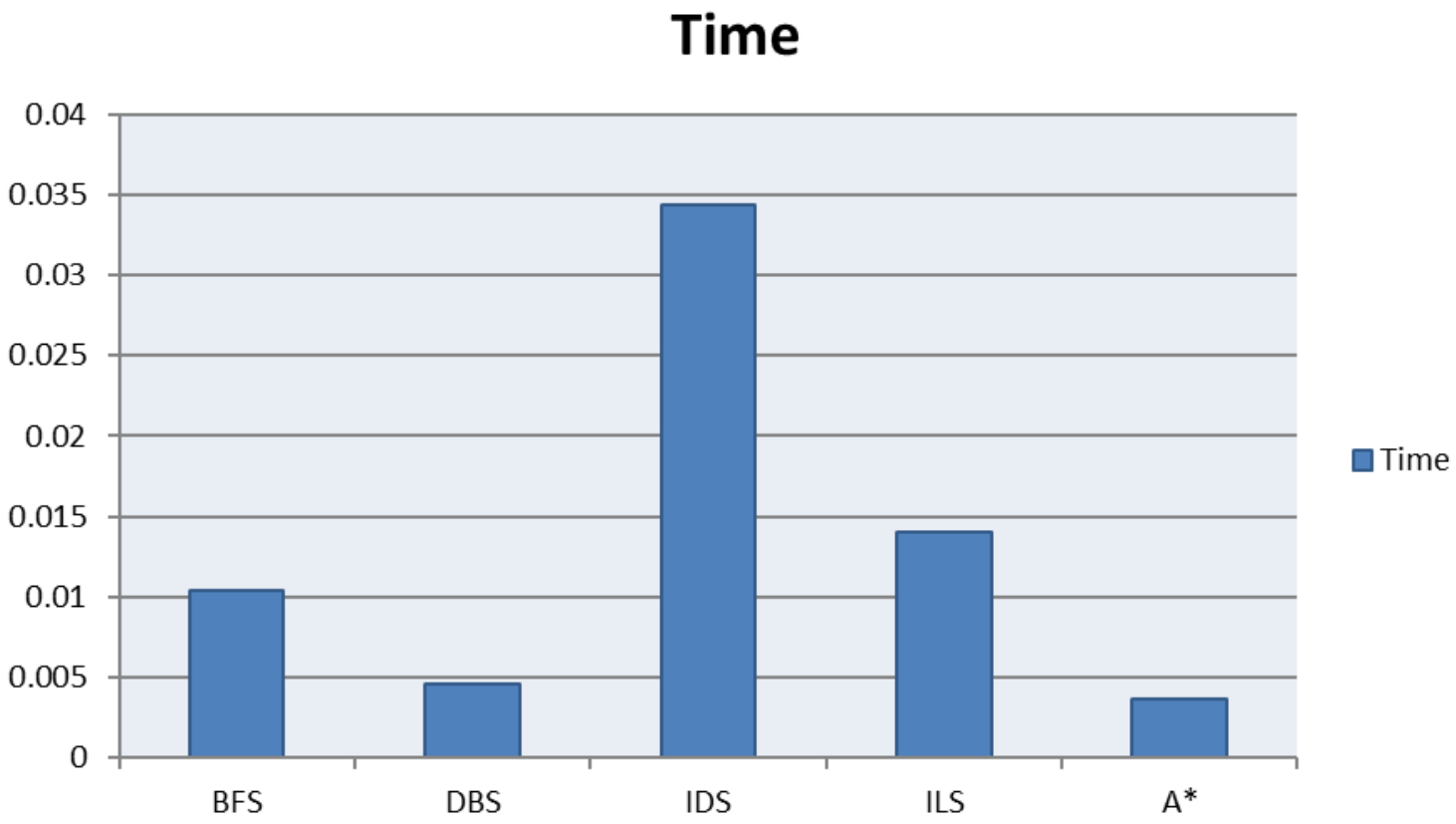
1	3	0
8	2	4
7	6	5

Source Puzzle **Destination Puzzle**

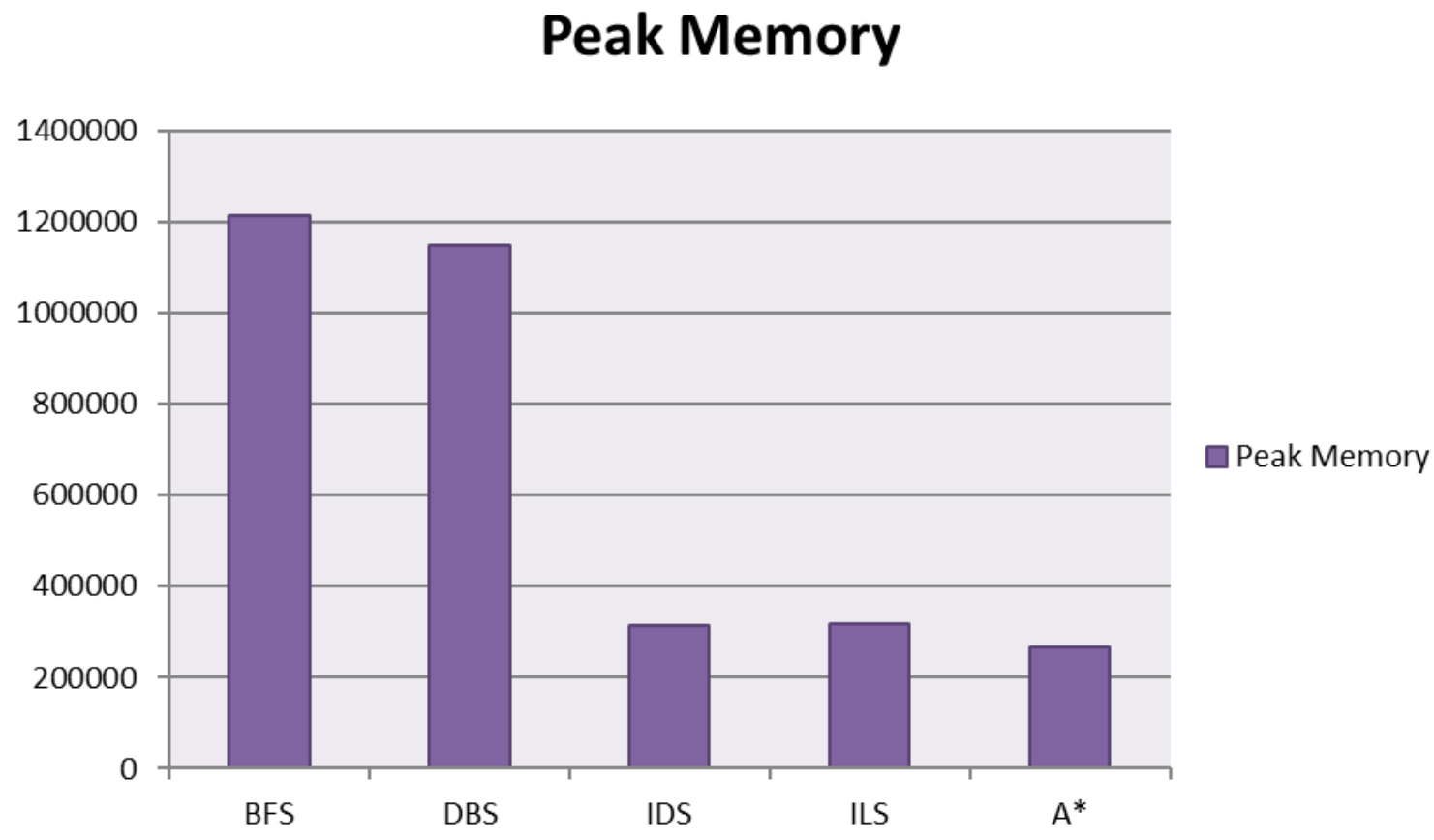
Time & Space value for each algorithm :

Algorithm	Time(s)	Space
BFS	0.01035	1216311
DBS	0.00455	1149299
IDS	0.03442	313911
ILS	0.01397	315867
A*	0.00359	266925

Comparing the time required for execution for each search algorithm :



Comparing the Peak Memory required for execution for each search algorithm:



Execution Sequence

- **BFS (Breadth-First Search):** In this algorithm, all the possible children of one level are stored before moving on to the next level until the goal state is found. Backtracking occurs from the goal state to find the smallest path.

  It saves all states in a level hence it requires huge amount of space. If the solution is not deep it doesn't take much time for search.

- **DBS (Depth-Bound Search):** In this algorithm, all children starting from one side (usually from left) in every level are checked before going back to the initial state and check the next list, this continues till the goal state is reached. A depth bound is set to limit the lowest level it will check and move back to the initial state without checking the lower levels.

  It saves a search list until a bound hence the space required depends on the bound. It doesn't have to search the whole list so the time required is lesser than DFS or IDS.

- **IDS (Iterative Deepening Search):** In this algorithm, a depth-bounded searcher is repeatedly called. It performs a DBS with increasing order of depth bound starting with 1, then 2 and so on until the goal state is reached.

  It can have to call a huge number of DBS until solution is reached hence takes more time. It removes one search list then searches again with increased depth bound hence doesn't take a lot of space.

- **ILS (Iterative Lengthening Search):** This algorithm is similar to BFS as it explores an entire level of states before moving on to the next level but uses limits on path costs instead of depth limits. A node is discarded if its path cost exceeds that of the limit and moved on to the next node.

It is a form of greedy search and uses paths that seems close to goal hence uses comparatively less space but can easily be misguided into wrong path hence can take a long time of execution.

- **A* (A-Star Search):** This algorithm uses the properties of both UCS and greedy search to reach a solution. It considers both the path cost and heurestic value to find the lowest combination by adding them and checking the lowest possible path until the goal state is reached. It requires less time as it is a **bidirectional search** and requires less space because it only considers path with lowest combination and deletes the previous path if it is blocked or have found a better path.