

---

# PROJECT REPORT

A\* Algorithm

**SUBMITTED TO:**

Muhammad Jafer

21-11-2012

**SUBMITTED BY:**

MUHAMMAD TAQI HASSAN (10-ARID-128)

BSCS 8<sup>th</sup> A (Morning)

---

## WHAT & HOW

The A-Star algorithm is a best-first search algorithm that finds the least costly path from an initial configuration to a final configuration. It uses an exact + estimate cost heuristic function. The exact cost is known for any previous steps, while the estimated cost to reach the final configuration from the current can be estimated. This Algorithm is similar to Dijkstra, the only difference is that Dijkstra finds the minimum costs from the starting node to all others. A\* finds the minimum cost from the start node to the goal node.

The A-Star algorithm maintains two lists, an open list and a closed list. The open list is a priority queue of states, where we can pick out the next least costly state to evaluate. Initially, the open list contains the starting state. When we iterate once, we take the top of the priority queue, and then initially, check whether it is the goal state. If so, we are done. Otherwise, we calculate all adjacent states and their associated costs, and add them into the open queue..

A-Star is complete. It will find a solution if a solution exists. If it doesn't find a solution, then we can guarantee that no such solution exists.

A-Star will find a path with the lowest possible cost. This will depend heavily upon the quality of the cost function, and estimates provided.

---

## HISTORY & EXAMPLE

In 1968 Nils Nilsson suggested a heuristic approach for Shakey the Robot to navigate through a room containing obstacles. This path-finding algorithm, called A1, was a faster version of the then best known formal approach, Dijkstra's algorithm, for finding shortest paths in graphs. Bertram Raphael suggested some significant improvements upon this algorithm, calling the revised version A2. Then Peter E. Hart introduced an argument that established A2, with only minor changes, to be the best possible algorithm for finding shortest paths. Hart, Nilsson and Raphael then jointly developed a proof that the revised A2 algorithm was optimal for finding shortest paths under certain well-defined conditions. They thus named the new algorithm in Kleene star syntax to be the algorithm that starts with A and includes all possible version numbers or A\*

---

## APPLICATION

- The A-Star algorithm is adapted for the first time to search through 3-space to solve the Inverse Kinematics problem, and can produce a decidable, sound and complete search, which can perform on par or better than other numeric methods when dealing with complicated constrained systems.
- Nokia Implement's A-Star Algorithm in their Mobile Phone Games (Snake Games).
- A\* can be used for planning moves computer-controlled player (e.g., chess).

---

## WHY

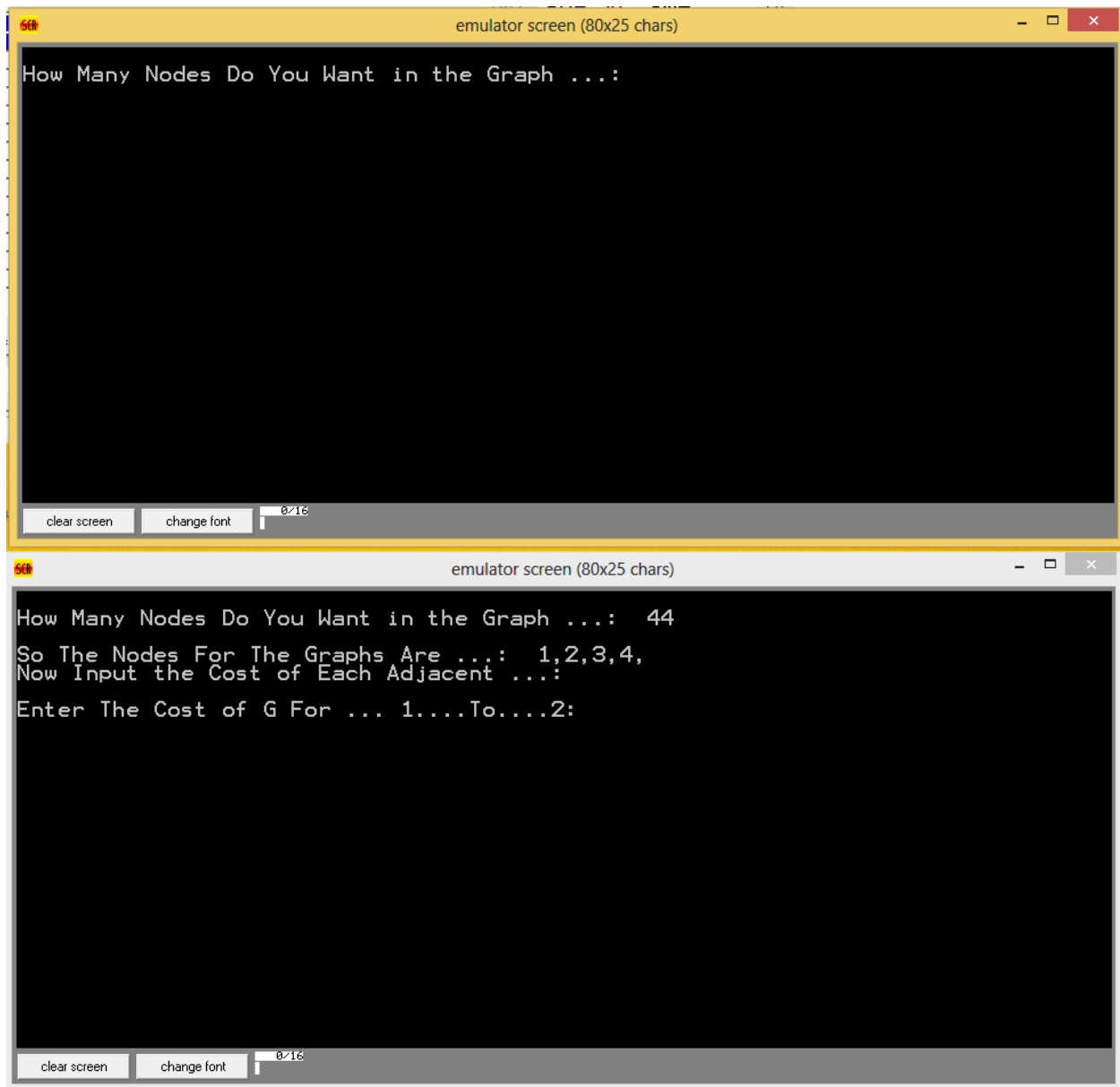
I have chosen this Algorithm to because this algorithm is complete. It will find a solution if a solution exists. If it doesn't find a solution, then we can guarantee that no such solution exists.

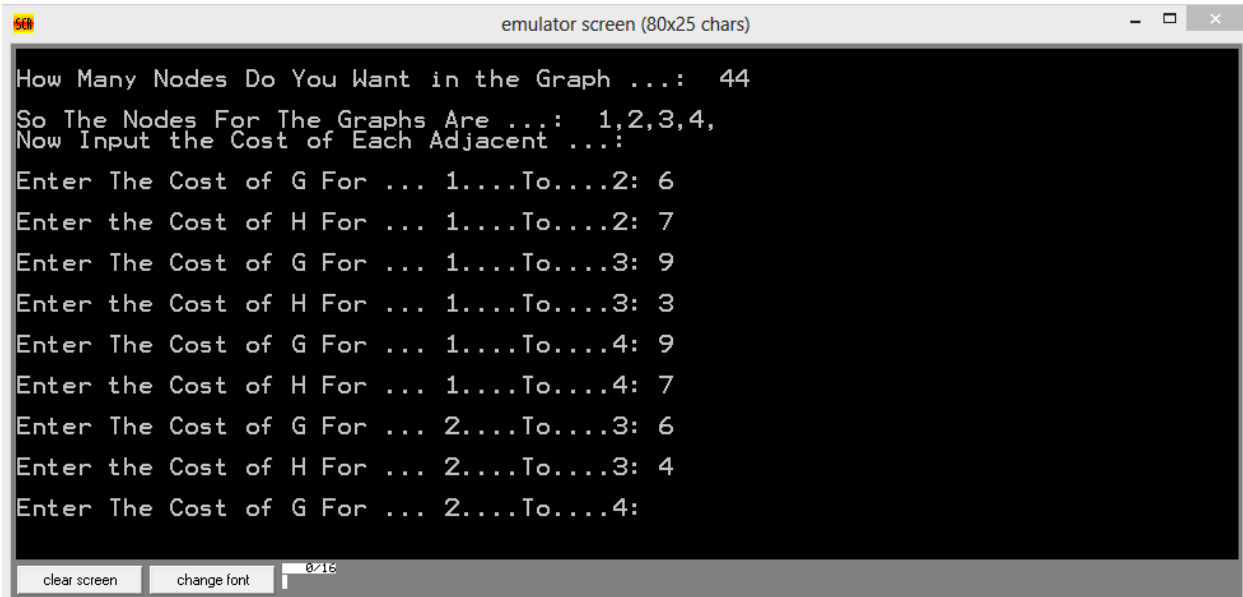
---

## IMPLEMENTATION TASK OF ALGORITHM IN X86 ASSEMBLY PROGRAMMING

1. Input the No. of nodes for the graph from the user.
2. Create adjacent of Each Node. Each node can have maximum 3 adjacent nodes to it.
3. Input the Value of G and H For each adjacent From the User and Store these values in 2 separate arrays.
4. During these also maintain an address array which contain mapping of node array to its adjacent array.
5. Calculate Each  $F = g + h$  value for each adjacent..
6. Calculate the minimum value of each 3 adjacent of a single node.
7. Save the adjacent with minimum value of f in the stack or array.

# SCREEN SHOTS





The image shows a screenshot of a terminal window titled "emulator screen (80x25 chars)". The terminal has a black background with white text. The text displays a sequence of prompts and user inputs for a graph program. The prompts ask for the number of nodes, the nodes themselves, and the costs of edges between them. The user has entered '44' for the number of nodes, '1,2,3,4' for the nodes, and provided costs for several edges. The bottom of the window features a control bar with buttons for "clear screen" and "change font", and a font size indicator showing "0/16".

```
How Many Nodes Do You Want in the Graph ...: 44
So The Nodes For The Graphs Are ...: 1,2,3,4,
Now Input the Cost of Each Adjacent ...:
Enter The Cost of G For ... 1....To....2: 6
Enter the Cost of H For ... 1....To....2: 7
Enter The Cost of G For ... 1....To....3: 9
Enter the Cost of H For ... 1....To....3: 3
Enter The Cost of G For ... 1....To....4: 9
Enter the Cost of H For ... 1....To....4: 7
Enter The Cost of G For ... 2....To....3: 6
Enter the Cost of H For ... 2....To....3: 4
Enter The Cost of G For ... 2....To....4:
```

clear screen   change font   0/16