CS217: Artificial Intelligence and Machine Learning
(associated lab: CS240)

Pushpak Bhattacharyya
CSE Dept.,
IIT Bombay

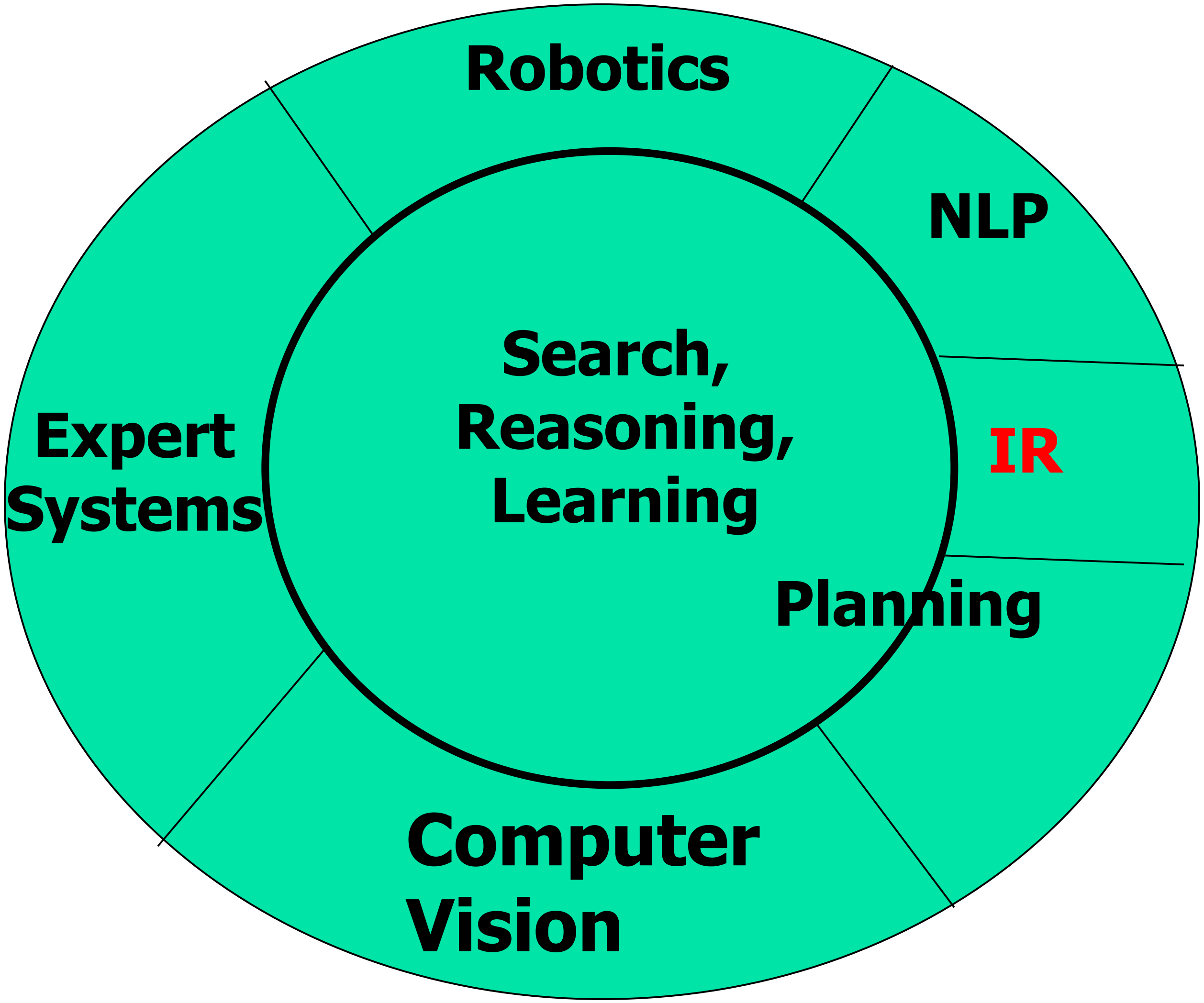
*Week2 of 13jan25, A**

Main points covered: week of 6jan25

Course website: very important

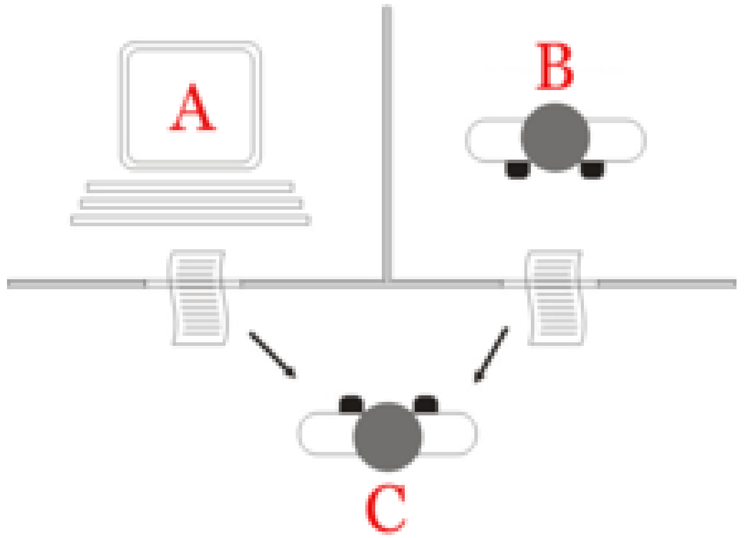
- https://www.cse.iitb.ac.in/~cs217/2025/

AI Perspective (post-web)



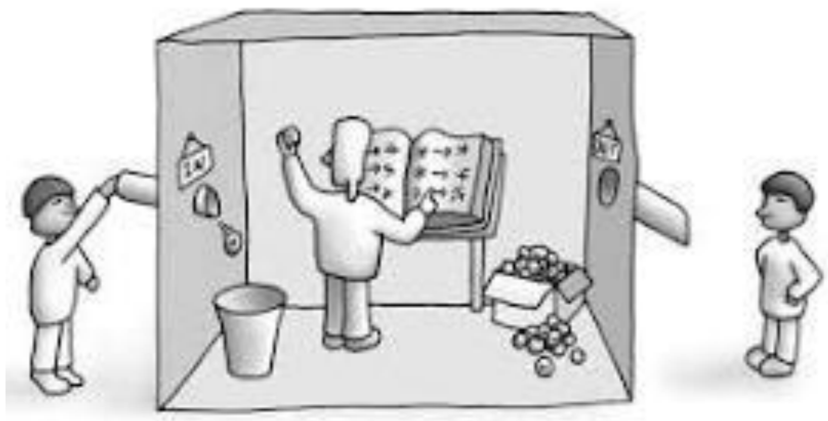
Turing Test *(wikipedia)*

- The **Turing test**, originally called the **imitation game** by Alan Turing in 1950
- Test of a machine's ability to exhibit intelligent behavior
- Equivalent to, or **indistinguishable** from, that of a human



The "standard interpretation" of the Turing test, in which player C, the interrogator, is given the task of trying to determine which player – A or B – is a computer and which is a human. The interrogator is limited to using the responses to written questions to make the determination

Searl's Chinese Room Experiment

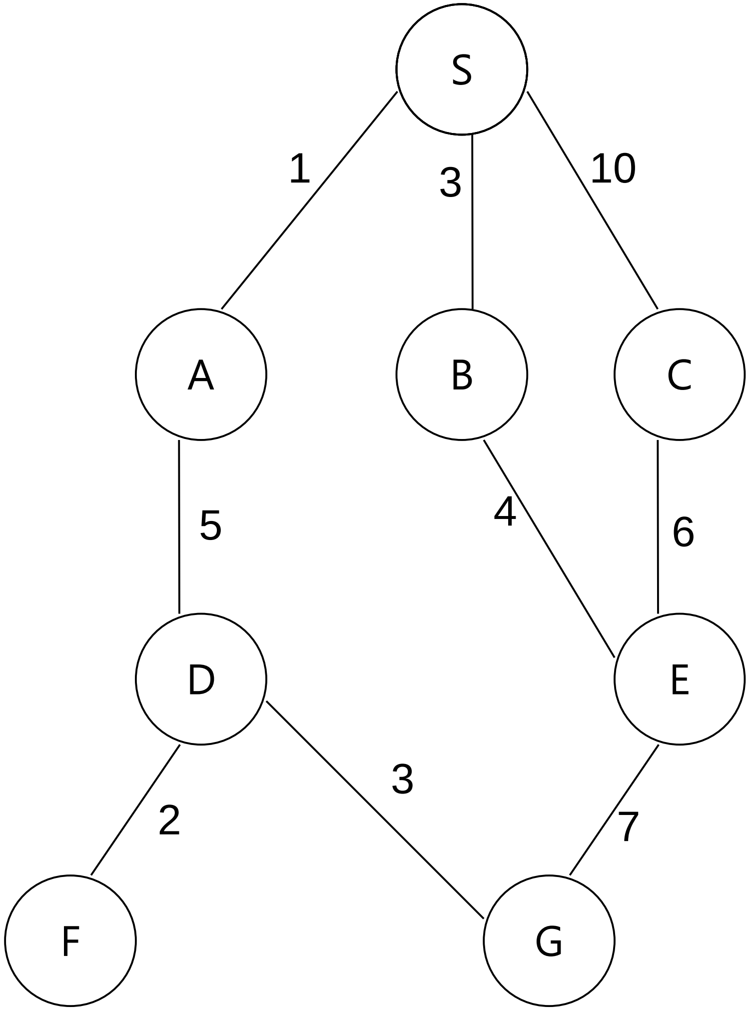


- A computer program cannot have a "mind", "understanding", or "consciousness", regardless of how intelligently or human-like the program may make the computer behave. Philosopher John Searle presented the argument in his paper "Minds, Brains, and Programs", published in *Behavioral and Brain Sciences* in 1980.
- **A human being sits in the room and does exactly as the program does, gives an impression of "knowing" Chinese, but in actuality does not understand Chinese.**

Grading (Tentative)

- CS217
  - Midsem: 30%
  - Endsem: 50%
  - Quizzes (2): 20%
- CS240
  - Weekly lab: each 10%
  - Midsem exam: 10%
  - Final Viva: 20%

General Graph search Algorithm



Graph G = (V,E)

1) Open List : S $^{(Ø, 0)}$

Closed list : Ø

2) OL : $A^{(S,1)}$, $B^{(S,3)}$, $C^{(S,10)}$

CL : S

3) OL : $B^{(S,3)}$, $C^{(S,10)}$, $D^{(A,6)}$

CL : S, A

4) OL : $C^{(S,10)}$, $D^{(A,6)}$, $E^{(B,7)}$

CL: S, A, B

5) OL : $D^{(A,6)}$, $E^{(B,7)}$

CL : S, A, B , C

6) OL : $E^{(B,7)}$, $F^{(D,8)}$, $G^{(D, 9)}$

CL : S, A, B, C, D

7) OL : $F^{(D,8)}$, $G^{(D,9)}$

CL : S, A, B, C, D, E

8) OL : $G^{(D,9)}$

CL : S, A, B, C, D, E, F

9) OL : Ø

CL : S, A, B, C, D, E,
F, G

Steps of GGS
(*principles of AI, Nilsson,)*

- 1\. Create a search graph *G*, consisting solely of the start node *S*; put *S* on a list called *OPEN.*
- *2.* Create a list called *CLOSED* that is initially empty.
- 3\. Loop: if *OPEN* is empty, exit with failure.
- 4\. Select the first node on *OPEN*, remove from *OPEN* and put on *CLOSED*, call this node *n*.
- 5\. if *n* is the goal node, exit with the solution obtained by tracing a path along the pointers from *n* to *s* in *G*. (ointers are established in step 7).
- 6\. Expand node *n*, generating the set *M* of its successors that are not ancestors of *n*. Install these memes of *M* as successors of *n* in *G*.

GGS steps (contd.)

- 7. Establish a pointer to *n* from those members of *M* that were not already in *G* (*i.e.*, not already on either *OPEN* or *CLOSED*). Add these members of *M* to *OPEN*. For each member of *M* that was already on *OPEN* or *CLOSED*, decide whether or not to redirect its pointer to *n*. For each member of M already on *CLOSED*, decide for each of its descendents in *G* whether or not to redirect its pointer.
- 8. Reorder the list *OPEN* using some strategy.
- 9. Go *LOOP.*

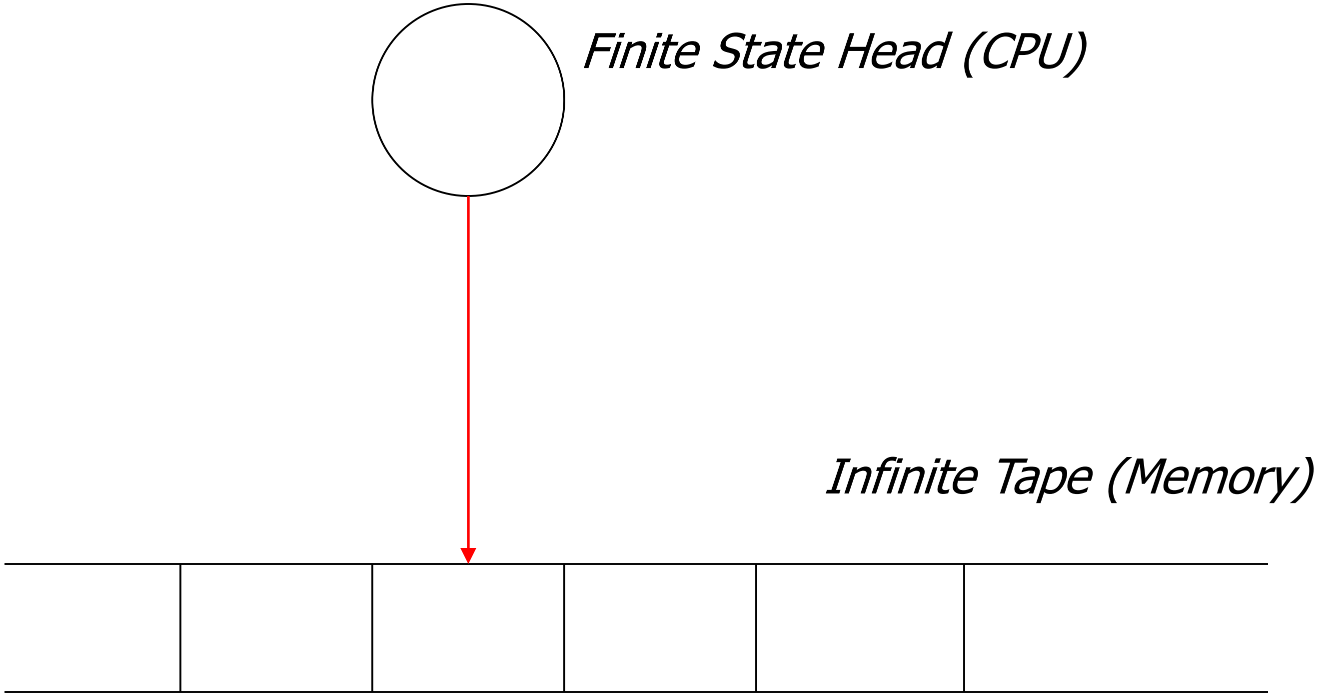
End of main points

Foundational Points

Foundational Points

- Church Turing Hypothesis
  - Anything that is computable is computable by a Turing Machine
  - Conversely, the set of functions computed by a Turing Machine is the set of ALL and ONLY computable functions

Turing Machine



Foundational Points *(contd)*

- Physical Symbol System Hypothesis (Newel and Simon)
  - *For Intelligence to emerge it is enough to manipulate symbols*

Foundational Points *(contd)*

- Society of Mind (Marvin Minsky)
  - *Intelligence emerges from the interaction of very simple information processing units*
  - *Whole is larger than the sum of parts!*

Foundational Points *(contd)*

- Limits to computability
  - *Halting problem: It is impossible to construct a Universal Turing Machine that given any given pair <M, I> of Turing Machine M and input I, will decide if M halts on I*
  - What this has to do with intelligent computation? *Think!*

Foundational Points *(contd)*

- Limits to Automation
  - *Godel Theorem: A "sufficiently powerful" formal system cannot be BOTH complete and consistent*
  - "Sufficiently powerful": at least as powerful as to be able to capture Peano's Arithmetic
  - Sets limits to automation of reasoning

Foundational Points *(contd)*

- Limits in terms of time and Space
  - *NP-complete and NP-hard problems: Time for computation becomes extremely large as the length of input increases*
  - *PSPACE complete*: Space requirement becomes extremely large
  - Sets limits in terms of resources

Two broad divisions of Theoretical CS

- Theory A
  - Algorithms and Complexity
- Theory B
  - Formal Systems and Logic

AI as the forcing function

- Time sharing system in OS
  - Machine giving the illusion of attending simultaneously with several people
- Compilers
  - Raising the level of the machine for better man machine interface
  - Arose from Natural Language Processing (NLP)
    - NLP in turn called the forcing function for AI

Allied Disciplines

Philosophy	Knowledge Rep., Logic, Foundation of AI (is AI possible?)
Maths	Search, Analysis of search algos, logic
Economics	Expert Systems, Decision Theory, Principles of Rational Behavior
Psychology	Behavioristic insights into AI programs
Brain Science	Learning, Neural Nets
Physics	Learning, Information Theory & AI, Entropy, Robotics
Computer Sc. & Engg.	Systems for AI

Symbolic AI

Connectionist AI is contrasted with Symbolic AI

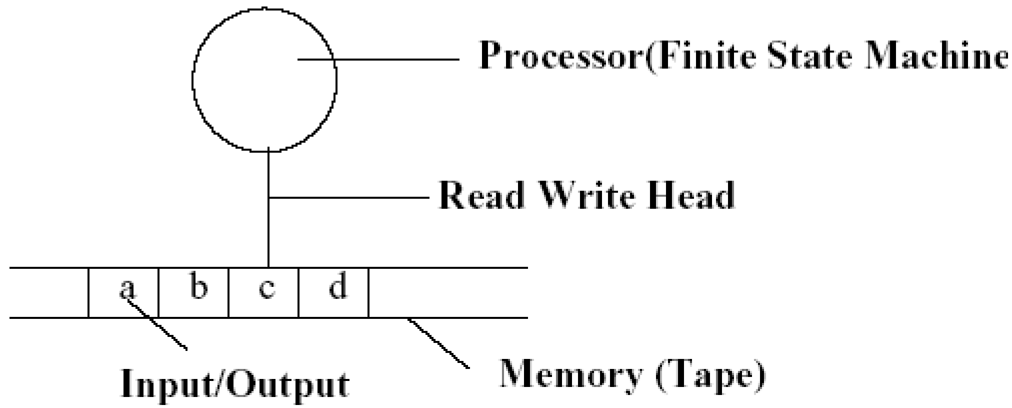
Symbolic AI - Physical Symbol System Hypothesis

> **Every intelligent system can be constructed by storing and processing symbols and nothing more is necessary.**

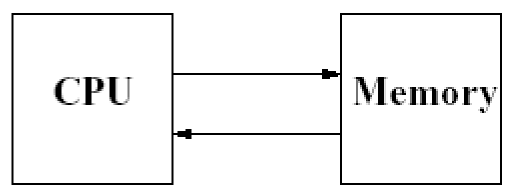
Symbolic AI has a bearing on models of computation such as

- **Turing Machine**
- **Von Neumann Machine**
- **Lambda calculus**

Turing Machine & Von Neumann Machine



Turing machine



VonNeumann Machine

Challenges to Symbolic AI

Motivation for challenging Symbolic AI

A large number of computations and information process tasks that living beings are comfortable with, are not performed well by computers!

The Differences

Brain computation in living beings	TM computation in computers
Pattern Recognition	Numerical Processing
Learning oriented	Programming oriented
Distributed & parallel processing	Centralized & serial processing
Content addressable	Location addressable

A* Search: one of the pillars of AI

Search building blocks

- State Space : Graph of states (Express constraints and parameters of the problem)
- Operators : Transformations applied to the states.
- Start state : S_0 (Search starts from here)
- Goal state : $\{G\}$ - Search terminates here.
- Cost : Effort involved in using an operator.
- Optimal path : Least cost path

Examples

Problem 1 : 8 – puzzle

4	3	6
2	1	8
7		5

S

1	2	3
4	5	6
7	8	

G

Tile movement represented as the movement of the blank space.

Operators:

L : Blank moves left

R : Blank moves right

U : Blank moves up

D : Blank moves down

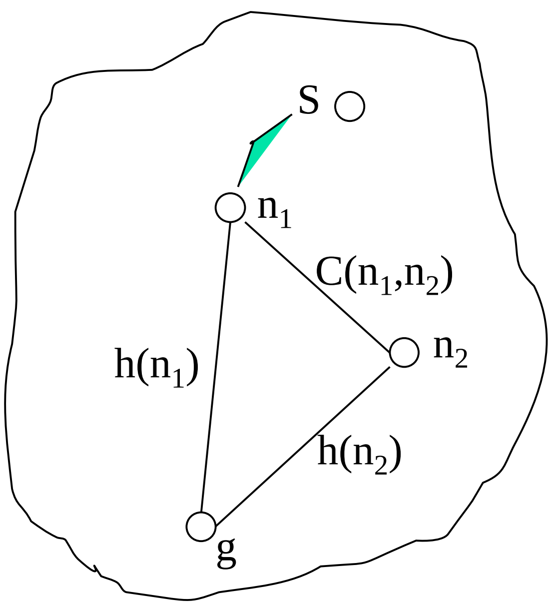
$$C(L) = C(R) = C(U) = C(D) = 1$$

GGS is a general umbrella

OL is a queue (BFS)

OL is stack (DFS)

OL is accessed by using a functions $f = g+h$ (Algorithm A)



Algorithm A

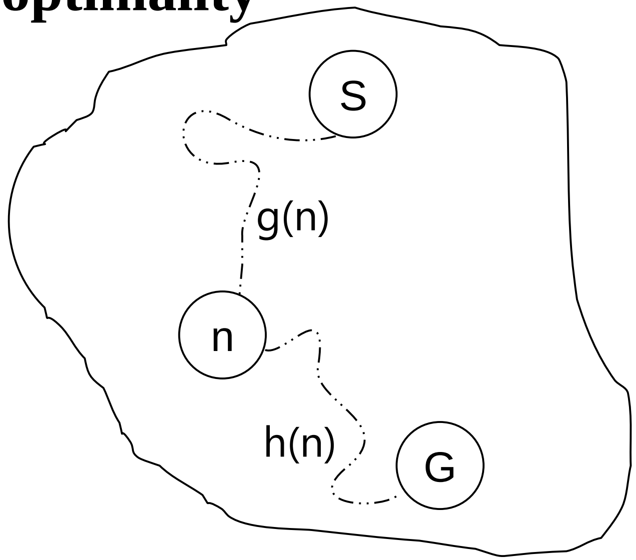
- A function *f* is maintained with each node

  $f(n) = g(n) + h(n)$, n is the node in the open list
- Node chosen for expansion is the one with least *f* value
- For BFS: $h = 0$, g = number of edges in the path to S
- For DFS: $h = 0$, $g = \frac{1}{\text{No of edges in the path to S}}$

Algorithm A*

- One of the most important advances in AI
- *g(n)* = least cost path to n from S found so far
- *h(n)* <= *h*(n)* where *h*(n)* is the actual cost of optimal path to G(node to be found) from *n*

"Optimism leads to optimality"



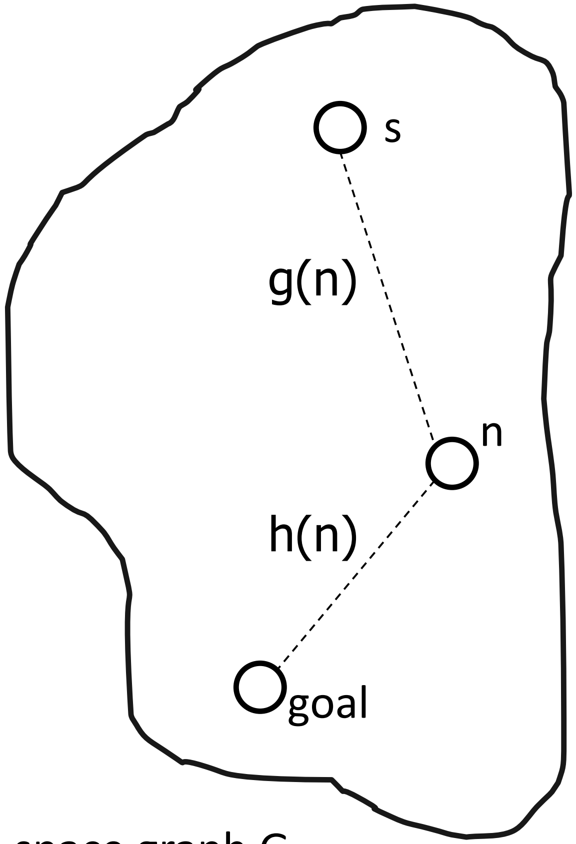
A* Algorithm – Definition and Properties

- $f(n) = g(n) + h(n)$
- The node with the least value of f is chosen from the *OL*.

- $f^*(n) = g^*(n) + h^*(n)$, where,
  $g^*(n)$ = actual cost of the optimal path (s, n)
  $h^*(n)$ = actual cost of optimal path (n, g)

- $g(n) \geq g^*(n)$

- By definition, $h(n) \leq h^*(n)$



State space graph G

8-puzzle: heuristics

Example: 8 puzzle

2	1	4
7	8	3
5	6	

s

1	6	7
4	3	2
5		8

n

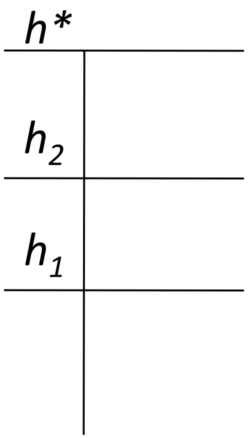
1	2	3
4	5	6
7	8	

g

$h^*(n)$ = actual no. of moves to transform *n* to *g*

1. $h_1(n)$ = no. of tiles displaced from their destined position.
2. $h_2(n)$ = sum of Manhattan distances of tiles from their destined position.

$h_1(n) \le h^*(n)$ and $h_1(n) \le h^*(n)$



Comparison

A* critical points

- **Goal**
  1. Do we know the goal?
  2. Is the distance to the goal known?
  3. Is there a path (known?) to the goal?

A* critical points

- **About the path**

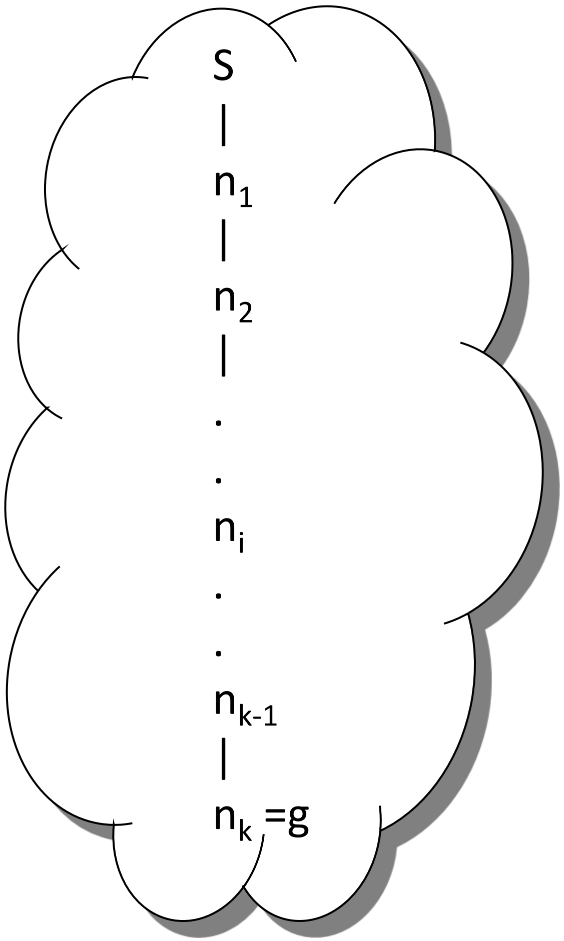
  Any time before A* terminates there exists on the OL, a node from the optimal path all whose ancestors in the optimal path are in the CL.

This means,

$\exists$ in the OL always a node ‘n’ s.t.

$$g(n) = g^*(n)$$

Key point about A* search



<u>Statement</u>:

Let S $-n_1-n_2-n_3 \ldots n_i \ldots -n_{k-1}-$ n_k(=G) be an optimal path.

At any time during the search:

1. There is a node n_i from the optimal path in the OL
2. For n_i all its ancestors $S, n_1, n_2, \ldots, n_{i-1}$ are in CL
3. $g(n_i) = g^*(n_i)$

Proof of the statement

Proof by induction on iteration no. j

Basis : j = 0, S is on the OL, S satisfies the statement

Hypothesis : Let the statement be true for j = p (p^{th} iteration)

Let n_i be the node satisfying the statement

Proof (continued)

Induction : Iteration no. $j = p+1$

Case 1 : n_i is expanded and moved to the closed list

Then, n_{i+1} from the optimal path comes to the OL

Node n_{i+1} satisfies the statement (note: if n_{i+1} is in CL, then n_{i+2} satisfies the property)

Case 2 : Node $x \neq n_i$ is expanded

Here, n_i satisfies the statement

A* Algorithm- Properties

- **Admissibility**: An algorithm is called admissible if it always terminates and terminates in optimal path
- **Theorem**: A* is admissible.
- **Lemma**: Any time before A* terminates there exists on *OL* a node *n* such that $f(n) <= f^*(s)$
- **Observation**: For optimal path $s \rightarrow n_1 \rightarrow n_2 \rightarrow \ldots \rightarrow g$,
  1. $h^*(g) = 0$, $g^*(s)=0$ and
  2. $f^*(s) = f^*(n_1) = f^*(n_2) = f^*(n_3) \ldots = f^*(g)$

A* Properties *(contd.)*

$f^*(n_i) = f^*(s)$, $n_i \neq s$ and $n_i \neq g$

Following set of equations show the above equality:

$$f^*(n_i) = g^*(n_i) + h^*(n_i)$$

$$f^*(n_{i+1}) = g^*(n_{i+1}) + h^*(n_{i+1})$$

$$g^*(n_{i+1}) = g^*(n_i) + c(n_i, n_{i+1})$$

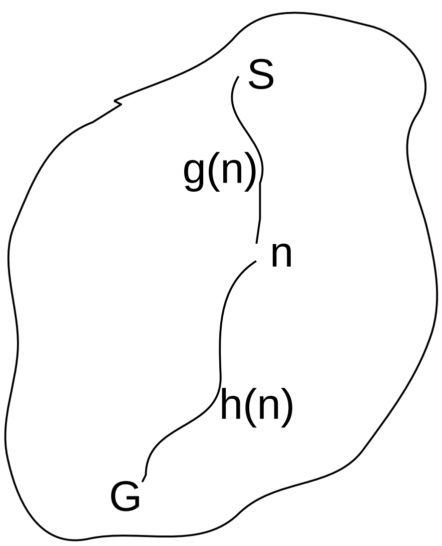
$$h^*(n_{i+1}) = h^*(n_i) - c(n_i, n_{i+1})$$

Above equations hold since the path is optimal.

Admissibility of A*

A* always terminates finding an optimal path to the goal if such a path exists.

Intuition



(1) In the open list there always exists a node *n* such that *f(n)* <= *f*(S)* .

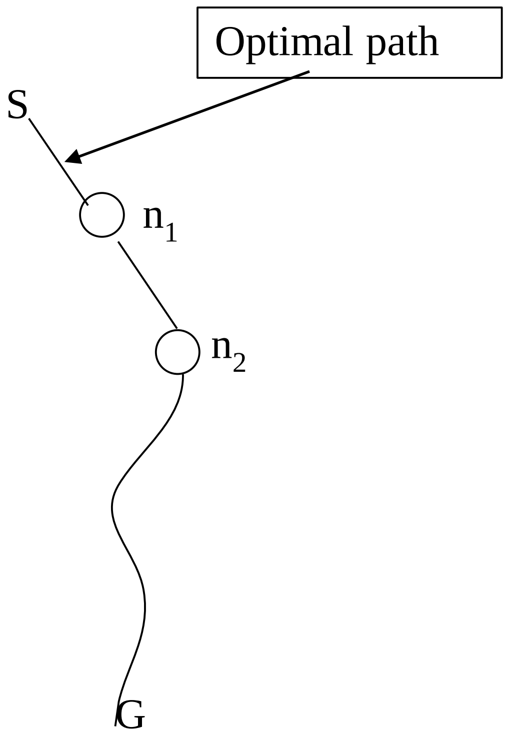
(2) If A* does not terminate, the *f* value of the nodes expanded become unbounded.

1) and 2) are together inconsistent

Hence A* must terminate

Lemma

Any time before A* terminates there exists in the open list a node *n'* such that $f(n') <= f^*(S)$



For any node n_i on optimal path,

$f(n_i) = g(n_i) + h(n_i)$

$<= g^*(n_i) + h^*(n_i)$

Also $f^*(n_i) = f^*(S)$

Let *n'* be the first node in the optimal path that is in OL. Since <u>all</u> parents of *n'* in the optimal have gone to CL,

$g(n') = g^*(n')$ *and* $h(n') <= h^*(n')$

$=> f(n') <= f^*(S)$

If A* does not terminate

Let e be the least cost of all arcs in the search graph.

Then $g(n) >= e.l(n)$ where $l(n)$ = # of arcs in the path from S to n found so far. If A* does not terminate, $g(n)$ and hence $f(n) = g(n) + h(n)$ $[h(n) >= 0]$ will become unbounded.

This is not consistent with the lemma. So A* has to terminate.

2^{nd} part of admissibility of A*

The path formed by A* is optimal when it has terminated

Proof

Suppose the path formed is not optimal

Let *G* be expanded in a non-optimal path.

At the point of expansion of *G*,

$f(G) = g(G) + h(G)$

$= g(G) + 0$

$> g^*(G) = g^*(S) + h^*(S)$

$= f^*(S)$ [$f^*(S)$ = cost of optimal path]

This is a contradiction

So path should be optimal

Summary on Admissibility

- 1. A* algorithm halts
- *2.* A* algorithm finds optimal path
- 3. If $f(n) < f^*(S)$ then node n has to be expanded before termination
- 4. If A* does not expand a node n before termination then $f(n) >= f^*(S)$

Exercise-1

Prove that if the distance of every node from the goal node is "known", then no "search:" is necessary

Ans:

- For every node n, $h(n)=h^*(n)$. The algo is A*.
- Lemma proved: any time before A* terminates, there is a node m in the OL that has $f(m) <= f^*(S)$, S= start node (m is the node on the optimal path all whose ancestors in the optimal path are in the closed list).
- For m, $g(m)=g^*(m)$ and hence $f(m)=f^*(S)$.
- Thus at every step, the node with $f=f^*$ will be picked up, and the journey to the goal will be completely directed and definite, with no "search" at all.
- Note: when $h=h^*$, f value of any node on the OL can never be less than $f^*(S)$.

Exercise-2

If the *h* value for every node over-estimates the *h** value of the corresponding node by a constant, then the path found need not be costlier than the optimal path by that constant. Prove this.

Ans:

- Under the condition of the problem, $h(n) <= h^*(n) + c$.
- Now, any time before the algo terminates, there exists on the OL a node *m* such that $f(m) <= f^*(S)+c$.
- The reason is as follows: let *m* be the node on the optimal path all whose ancestors are in the CL (there *has to be* such a node).
- Now, $f(m)= g(m)+h(m)=g^*(m)+h(m) <= g^*(m)+h^*(m)+c = f^*(S)+c$
- When the goal *G* is picked up for expansion, it must be the case that
- $f(G)<= f^*(S)+c=f^*(G)+c$
- *i.e.*, $g(G)<= g^*(G)+c$, since $h(G)=h^*(G)=0$.

Better Heuristic Performs Better

Theorem

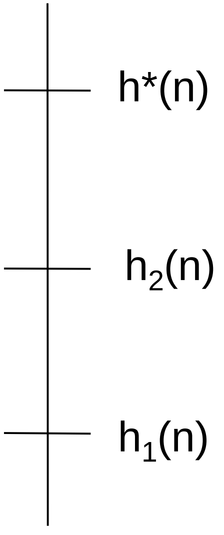
A version A_2* of A* that has a "better" heuristic than another version A_1* of A* performs at least "as well as" A_1*

Meaning of "better"

$h_2(n) > h_1(n)$ for all n

Meaning of "as well as"

A_1* expands at least all the nodes of A_2*



For all nodes n, except the goal node

Proof by induction on the search tree of A_2*.

A* on termination carves out a tree out of *G*

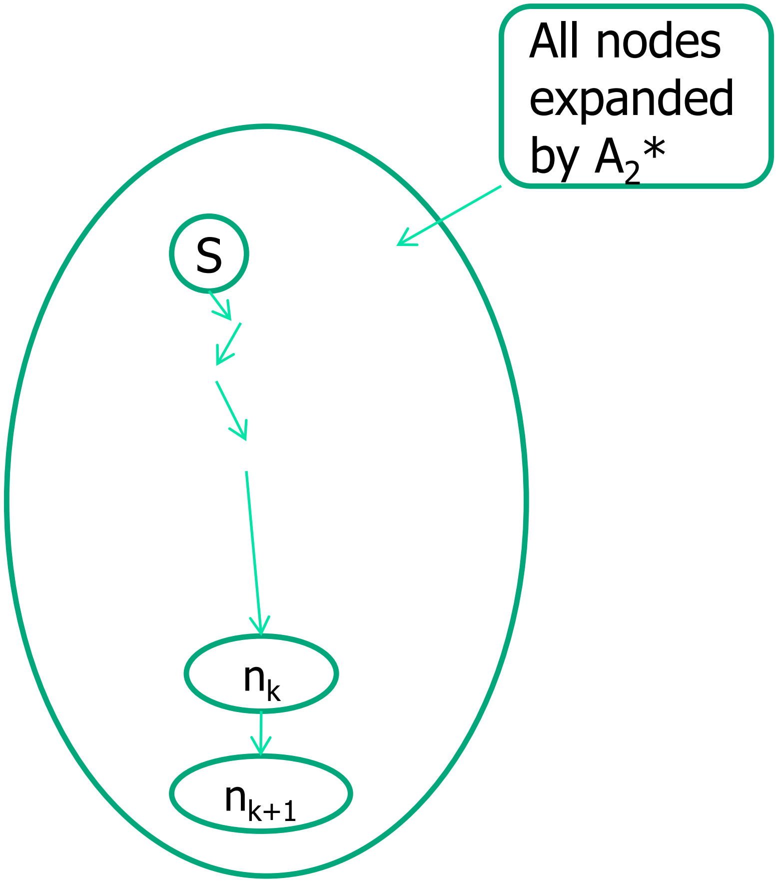
Induction
on the depth *k* of the search tree of A_2*. A_1* before termination expands all the nodes of depth *k* in the search tree of A_2*.

k=0. True since start node *S* is expanded by both

Suppose A_1* terminates without expanding a node *n* at depth *(k+1)* of A_2* search tree.

Since A_1* has seen all the parents of *n* seen by A_2*
$g_1(n) <= g_2(n)$ *(1)*

Proof for $g_1(n_{k+1}) <= g_2(n_{k+1})$ (1/3)



Proof for $g_1(n_{k+1}) <= g_2(n_{k+1})$ (2/3)

Case 1: n_k is the parent of n_{k+1} even in A_1*

$\mathbf{g_1(n_{k+1})} = g_1(n_k) + cost\ (n_k, n_{k+1})$

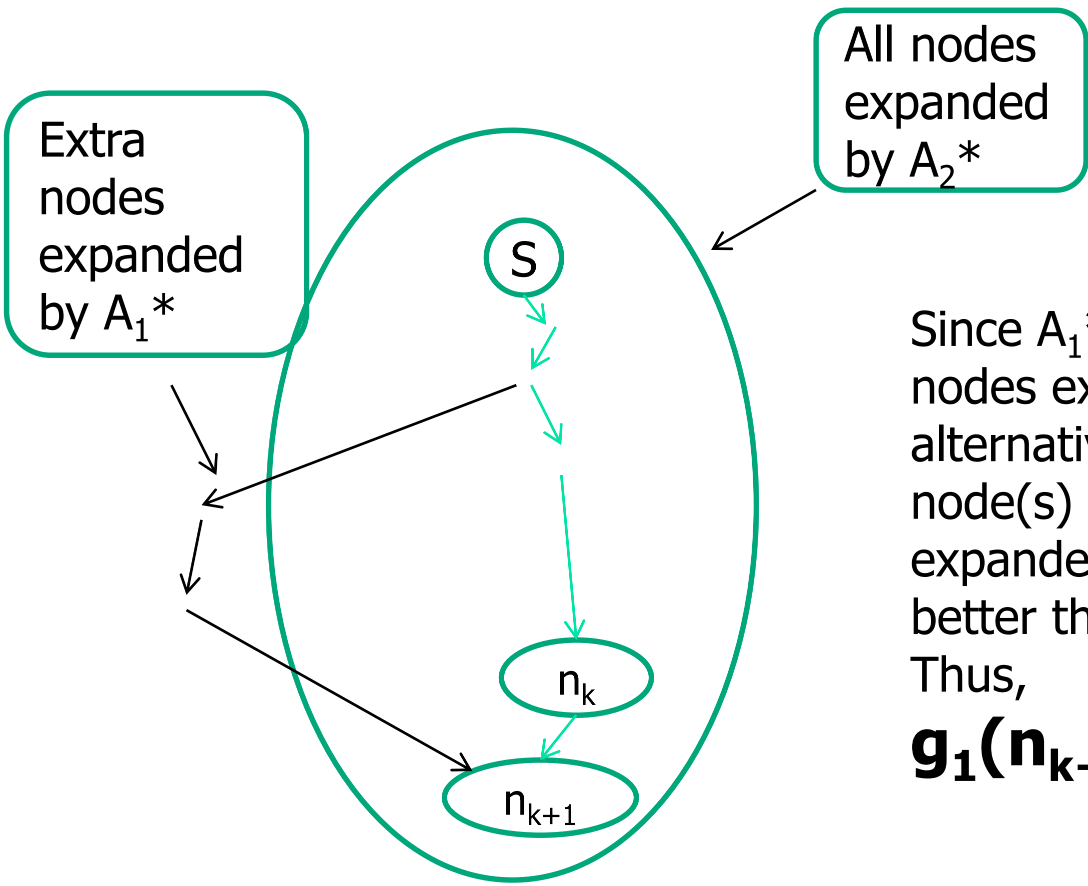
$<= g_2(n_k) + cost\ (n_k, n_{k+1})$ ……………….(1)

$\mathbf{<= g_2(n_{k+1})}$

(1) -> $g_1(n_k)$ has to be less than $g_2(n_k)$ as all nodes expanded by A_2* have been expanded by A_1* too. Therefore, it can only find a path better than A_1* to n_k.

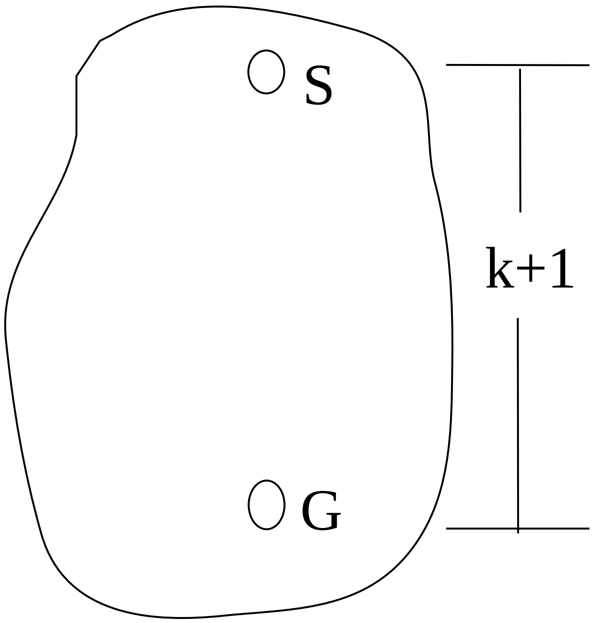
Proof for $g_1(n_{k+1}) <= g_2(n_{k+1})$ (3/3)

Case 2: n_k is not the parent of n_{k+1} in A_1*



Since A_1* has already expanded all nodes expanded by A_2*, if it finds an alternative path through some node(s) other than the ones expanded by A_2*, then it has to be better than the path as per A_2*. Thus,

$$\mathbf{g_1(n_{k+1}) <= g_2(n_{k+1})}$$



Since A_1* has terminated without expanding *n*,
$f_1(n) >= f^*(S)$ (2)

Any node whose *f* value is strictly less than $f^*(S)$ has to be expanded.
Since A_2* has expanded *n*
$f_2(n) <= f^*(S)$ (3)

From (1), (2), and (3)
$h_1(n) >= h_2(n)$ which is a contradiction. Therefore, A_1* has to expand all nodes that A_2* has expanded.

<u>Exercise</u>

If better means $h_2(n) > h_1(n)$ for some *n* and $h_2(n) = h_1(n)$ for others, then Can you prove the result ?