Algorithm A

$$f(n)$$

The cost of a minimal path

constrained to go through node n

The node of OPEN are arranged according to f

Notation

$k(n_i, n_j)$
Actual cost of going from n_i to n_j
(Undefined if no path between them)

$k(n, t_i)$
Minimal Cost Path from n to some goal t_i

$h^*(n)$
Minimum of all the $k(n, t_i)$

Costs

Any path which achieves $h^*(n)$is an *optimum* path

$g^*(n)= k(s,n)$
Minimum cost of getting to node n (from the start)

$f^*(n)= g^*(n)+ h^*(n)$
The optimal path to a solution
constrained to go through node n

$f(n) = g(n) + h(n)$
will be an approximation of $f^*(n)$

Costs

$g(n)$ An extimate of $g^*(n)$

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

One choice
$g(n) =$ The sum of costs from the start to the node n

$h(n)$
Rely on information about the domain

Admissible

Breadth First Search

$$h(n) = 0$$
$$g(n) = d = \text{The depth of the node}$$

Breadth First Search guarenteed to find the minimum path to goal

$$h(n) = 0 \leq h^*(n)$$

General Result:

If

$$h(n) \leq h^*(n)$$

Then

Then the algorithm will find the optimal path to goal

Breadth First is a special Case of this

A search is

Admissible

If it terminates with the optimal path

Best First Algorithm

1. Put Start node on OPEN list
2. If OPEN is empty, exit with failure; No solution exists
3. Remove *best* node,n, from OPEN determined by f(n).

4. Place n on CLOSED list and expand, generating all its successors; Each successor has a pointer back to n.
5. If any successors are goal nodes:
   (a) Trace Solution back through parents
6. For Every Successor, n':
   (a) Calculate $f(n')$
   (b) If n' not on the OPEN or CLOSED
      i. Add to OPEN list
      ii. Assign $f(n')$ to n'
   (c) If n' is on OPEN or CLOSED
      i. Compare $f(n')$ of each
      ii. If worse, discard otherwise replace
      iii. If on CLOSED, move to OPEN
7. Repeat

Termination for Finite Graphs

Two Places to terminate

If OPEN is empty, exit with failure; No solution exists

If any successors are goal nodes

For every Cycle
One node is removed from OPEN and a finite number is put on

For finite graphs, eventually run out of new successors

Thus will terminate either when a goal is found
or
When runs out of nodes on OPEN

Termination for Infinite Graphs

If

a path from the start node, s, to a goal node exists

Then

the algorithm will terminate

Thought of Proof

Termination prevented only if
new nodes are forever added to OPEN

Will show:
Even the smallest of the cost values,
$f(n)$ will grow impossibly large

f(n) Arbitrarily Large

$$d^*(n)$$

length of shortest path to node n

$$g^*(n) \geq d^*(n)e$$

where e is some value

$$g(n) \geq g^*(n) \geq d^*(n)e$$

If

$$h(n) \geq 0$$

$$f(n) \geq g(n) \geq \geq d^*(n)e$$

Thus if it does not terminate,

$f(n)$ will become arbitrarily large

f(n') Bounded

Before Termination

There is always a node, such that

$f(n) \leq f^*(n)$

Ordered Sequence: Optimal Path

$(s = n_0, n_1, ..., n_k)$ where n_k is a goal node

let n' be the first node of this sequence which is on OPEN

There must be one because s is on OPEN

$f(n') = g(n') + h(n')$

Since n' is on optimal path

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

$f(n') = g^*(n) + h(n')$

Assumption:

$h(n') \leq h^*(n)$

$f(n') \leq g^*(n) + h^*(n) = f^*(n)$

But $f^*(n)$on optimal path is equal to $f^*(s)$

$f(n') \leq f^*(s)$

This puts an upper limit to $f(n')$

It is bounded

Termination if a Solution

If a solution exists

There is a node, n' on the optimal path

For this node
$f(n')$ is bounded

If the algorithm is non-bounding
the f(n) gets arbitrarily large

Thus termination for even infinite graphs
(If there is a solution)

Admissible

Proof by Contradiction

Suppose termination is found to some goal t
t is not optimal
$f(t) = g(t) > f^*(s)$

By the previous proof
There was an n' on the optimal path

$$f(n') \leq f^*(s) < f(t)$$

But if this were true
n' would have been selected from OPEN
not t
Thus didn't terminate

Contradiction

Corollary

The value of f will never exceed $f^*(s)$ of the optimal path

Let n be any node selected for expansion

Case 1: If n is a goal node
Then
$f(n) = f^*(s)$

Corollary

Case 2: If n is not a goal node

n selected before termination

There existed on OPEN some node n' on the optimal path
$$f(n') \leq f^*(s)$$

If n=n'
$$f(n) = f(n') \leq f^*(s)$$

if n does not equal n'

The algorithm chose n before n'

$$f(n) \leq f(n') \leq f^*(s)$$

thus for any node selected for expansion

$$f(n) \leq f^*(s)$$