summing the arc costs encountered while tracing the pointers from n to s. (This path is the lowest cost path from s to n found so far by the search algorithm. The value of $g(n)$ for certain nodes may decrease if the search tree is altered in step 7.) Notice that this definition implies $g(n) \geq g^*(n)$. For the estimate $h(n)$, of $h^*(n)$, we rely on heuristic information from the problem domain. Such information might be similar to that used in the function $W(n)$ in the 8-puzzle example. We call h the *heuristic function* and will discuss it in more detail later.

Suppose we now use as an evaluation function

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

We call the **GRAPHSEARCH** algorithm using this evaluation function for ordering nodes, *algorithm* **A**. Note that when $h \equiv 0$ and $g \equiv d$ (the depth of a node in the search tree), algorithm **A** is identical to breadth-first search. We claimed earlier that the breadth-first algorithm is guaranteed to find a minimal length path to a goal. We now show that if h is a lower bound on h^* (that is, if $h(n) \leq h^*(n)$ for all nodes n), then algorithm **A** will find an optimal path to a goal. When algorithm **A** uses an h function that is a lower bound on h^*, we call it *algorithm* **A*** (read "A-star"). Since $h \equiv 0$ is certainly a lower bound on h^*, the fact that the breadth-first algorithm finds minimal length paths follows directly as a special case of this more general result for algorithm **A***.

2.4.3. THE ADMISSIBILITY OF A*.

Let us say that a search algorithm is *admissible* if, for any graph, it always terminates in an optimal path from s to a goal node whenever a path from s to a goal node exists. In this section we show informally that **A*** is admissible.

To show that an algorithm is admissible, it is necessary to show, at least, that it terminates whenever a goal node is accessible. The **GRAPHSEARCH** algorithm terminates (if at all) either in step 3 or in step 5. Notice that in every cycle through the loop of the algorithm, a node is removed from *OPEN* and that only a finite number of new successors are added to *OPEN*. For finite graphs, we ultimately run out of new successors, and thus, unless the algorithm terminates successfully in step 5 by finding a goal node, it will terminate in step 3 after eventually depleting *OPEN*. Therefore,

RESULT 1: **GRAPHSEARCH** always terminates for finite graphs.

Next we would like to show that if a path from s to a goal node exists, **A*** will terminate even for infinite graphs. To do so, let us suppose the opposite, that **A*** does not terminate. Termination is prevented only if new nodes are forever added to *OPEN*. But in this case we can show that even the smallest of the f values of the nodes on *OPEN* will grow impossibly large.

Let $d^*(n)$ be the length of the shortest path in the implicit graph being searched from s to any node n in the search tree produced by **A***. Then since the cost of each arc in the graph is at least some small positive number e, $g^*(n) \geq d^*(n)e$. (Recall that $g^*(n)$ is the cost of the optimal path from s to n, and that $g(n)$ is the cost of the path in the search tree from s to node n.) Clearly, $g(n) \geq g^*(n)$, and thus $g(n) \geq d^*(n)e$. If $h(n) \geq 0$ (which we henceforth assume), $f(n) \geq g(n)$, and thus $f(n) \geq d^*(n)e$. In particular, for every node n on *OPEN*, the value of $f(n)$ is at least as large as $d^*(n)e$. Even though **A*** selects for expansion that node on *OPEN* whose f value is smallest, the node selected will ultimately have an arbitrarily large value of d^* and therefore also of f if **A*** does not terminate.

Now, to show that **A*** must eventually terminate, we show that before termination of **A***, there is always a node n on *OPEN* such that $f(n) \leq f^*(s)$. Let the ordered sequence $(s = n_0, n_1, \ldots, n_k)$, where n_k is a goal node, be an optimal path from s to a goal node. Then, for any time before **A*** terminates, let n' be the first node in this sequence that is on *OPEN*. (There must be at least one such node, because s is on *OPEN* at the beginning and if n_k is on *CLOSED*, **A*** has terminated.) By the definition of f for **A***, we have

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

We know that **A*** has already found an optimal path to n' since n' is on an optimal path to a goal and all of the ancestors on this path are on *CLOSED*. Therefore, $g(n') = g^*(n')$ and

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

Since we are assuming $h(n') \leq h^*(n')$, we can write

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