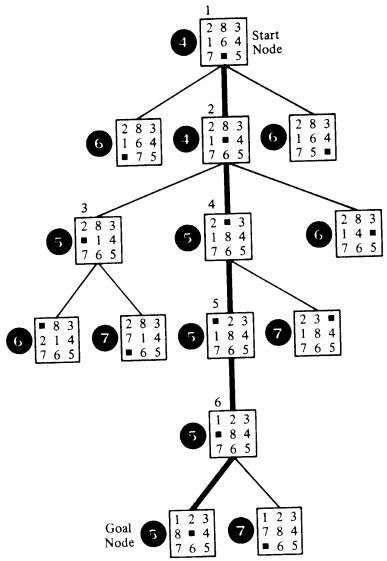

Fig. 2.8 A search tree using an evaluation function.

expanded. We see that the same solution path is found here as was found by the other search methods, although the use of the evaluation function has resulted in substantially fewer nodes being expanded. (If we simply use the evaluation function $f(n) = d(n)$, we get the breadth-first search process.)

The choice of evaluation function critically determines search results. The use of an evaluation function that fails to recognize the true promise of some nodes can result in nonminimal cost paths; whereas, the use of an evaluation function that overestimates the promise of all nodes (such as the evaluation function yielding breadth-first search) results in expansion of too many nodes. In the next few sections, we develop some theoretical results about the performance of **GRAPHSEARCH** when it uses a particular kind of evaluation function.

2.4.2. ALGORITHM A

Let us define the evaluation function f so that its value, $f(n)$, at any node n estimates the sum of the cost of the minimal cost path from the start node s to node n plus the cost of a minimal cost path from node n to a

goal node. That is, $f(n)$ is an estimate of the cost of a minimal cost path *constrained to go through node n*. That node on *OPEN* having the smallest value of f is then the node estimated to impose the least severe constraint; hence it is appropriate that it be expanded next.

Before demonstrating some of the properties of this evaluation function, we first introduce some helpful notation. Let the function $k(n_i, n_j)$ give the *actual* cost of a minimal cost path between two arbitrary nodes n_i and n_j. (The function k is undefined for nodes having no path between them.) The cost of a minimal cost path from node n to some particular goal node, t_i, is then given by $k(n, t_i)$. We let $h^*(n)$ be the minimum of all of the $k(n, t_i)$ over the entire set of goal nodes $\{t_i\}$. Thus, $h^*(n)$ is the cost of the minimal cost path from n to a goal node, and any path from node n to a goal node that achieves $h^*(n)$ is an *optimal* path from n to a goal. (The function h^* is undefined for any node n that has no accessible goal node.)

Often we are interested in knowing the cost $k(s,n)$ of an optimal path from a given start node, s, to some arbitrary node n. It will simplify our notation somewhat to introduce a new function g^* for this purpose. The function g^* is defined as

$$g^*(n) = k(s,n)\,,$$

for all n accessible from s.

We next define the function f^* so that its value $f^*(n)$ at any node n is the actual cost of an optimal path from node s to node n plus the cost of an optimal path from node n to a goal node, that is,

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

The value of $f^*(n)$ is then the cost of an optimal path from s constrained to go through node n. (Note that $f^*(s) = h^*(s)$ is the actual cost of an unconstrained optimal path from s to a goal.)

We desire our evaluation function f to be an estimate of f^*. Our estimate can be given by

$$f(n) = g(n) + h(n)\,,$$

where g is an estimate of g^* and h is an estimate of h^*. An obvious choice for $g(n)$ is the cost of the path in the search tree from s to n given by