

An upper bound on the computing time of Algorithm 7.7 may be arrived at by noticing that the number of internal nodes in the state space tree is  $\sum_{i=0}^{n-1} m^i$ . At each internal node,  $O(mn)$  time is spent by NEXTVALUE to determine the children corresponding to legal colorings. Hence, the total time is bounded by  $\sum_{i=0}^{n-1} m^i n = n(m^{n+1} - 1) / (m - 1) = O(nm^n)$ .

Figure 7.12 shows a simple graph containing four nodes. Below that is the tree which is generated by procedure MCOLORING. Each path to a leaf represents a coloring using at most 3 colors. Note that only twelve solutions exist with exactly 3 colors.

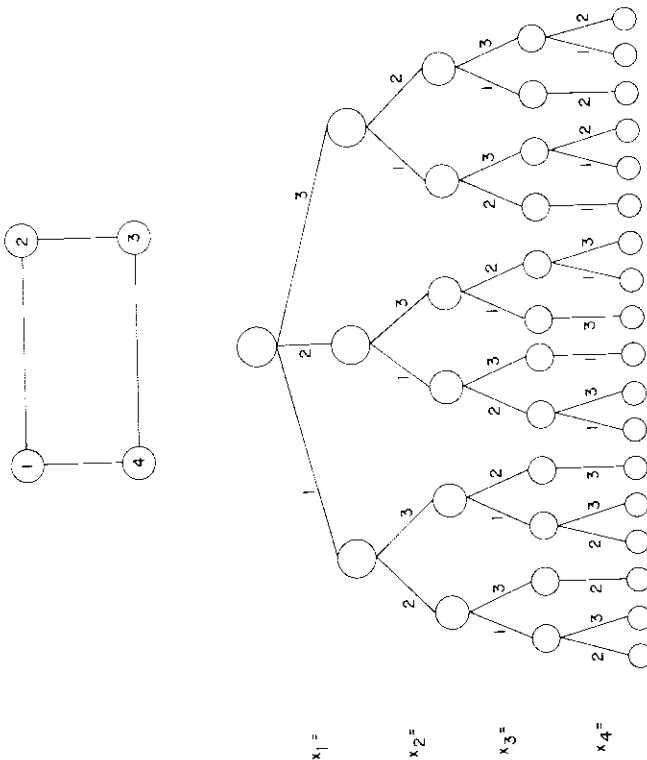


Figure 7.12 A 4 node graph and all possible 3 colorings

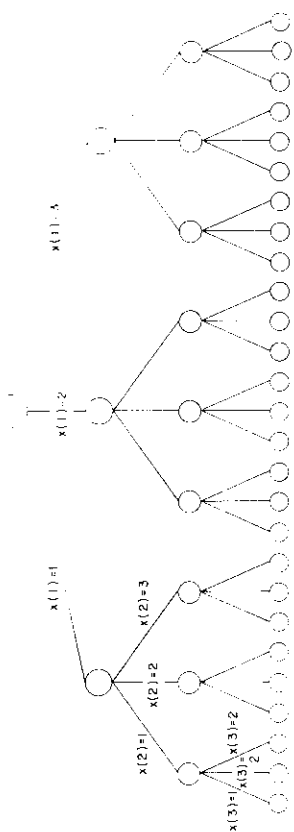


Figure 7.11 State space tree for MCOLORING when  $n = 3$  and  $m = 3$

have been defined. The main loop of MCOLORING repeatedly picks an element from the set of possibilities, assigns it to  $X(k)$ , and then calls MCOLORING recursively.

**procedure** NEXTVALUE( $k$ )

```
//X(1), ..., X(k - 1) have been assigned integer values in the range //
//[1, m] such that adjacent vertices have distinct integers. A value for //
//X(k) is determined in the range [0, m]. X(k) is assigned the next //
//highest numbered color while maintaining distinctness from the //
//adjacent vertices of vertex k. If no such color exists then X(k) ← 0. //
global integer m, n, X(1:n) boolean GRAPH(1:n, 1:n)
```

**integer** j, k

**loop**

```
X(k) ← (X(k) + 1) mod (m + 1) //next highest color //
```

```
if X(k) = 0 then return endif //all colors have been exhausted //
```

```
for j ← 1 to n do //check if this color is distinct from adjacent colors //
```

```
if GRAPH(k, j) and //if (k, j) is an edge //
```

```
X(k) = X(j) //and if adjacent vertices have identical colors //
```

```
then exit endif
```

**repeat**

```
if j = n + 1 then return endif //new color found //
```

```
repeat //otherwise try to find another color //
```

**end** NEXTVALUE

Algorithm 7.8 Generating a next color