Generic description of a backtracking algorithm

```
Initialize Q = {(ε, D_init)}.
while Q is not empty {
    Take a node (C,D) from Q, and delete that node from Q.
    Determine all the children (C_1,D_1),(C_2,D_2),...,(C_k,D_k) of (C,D).
    for i = 1,2,...,k {
        If (C_i,D_i) is a leaf node marked Yes, return Yes.
        If (C_i,D_i) is a non-leaf node {
            If the search has not reached a dead end at (C_i,D_i), add (C_i,D_i) to Q.
        }
    }
}
Return No.
```

Figure 128: A non-deterministic computation tree for the TSP

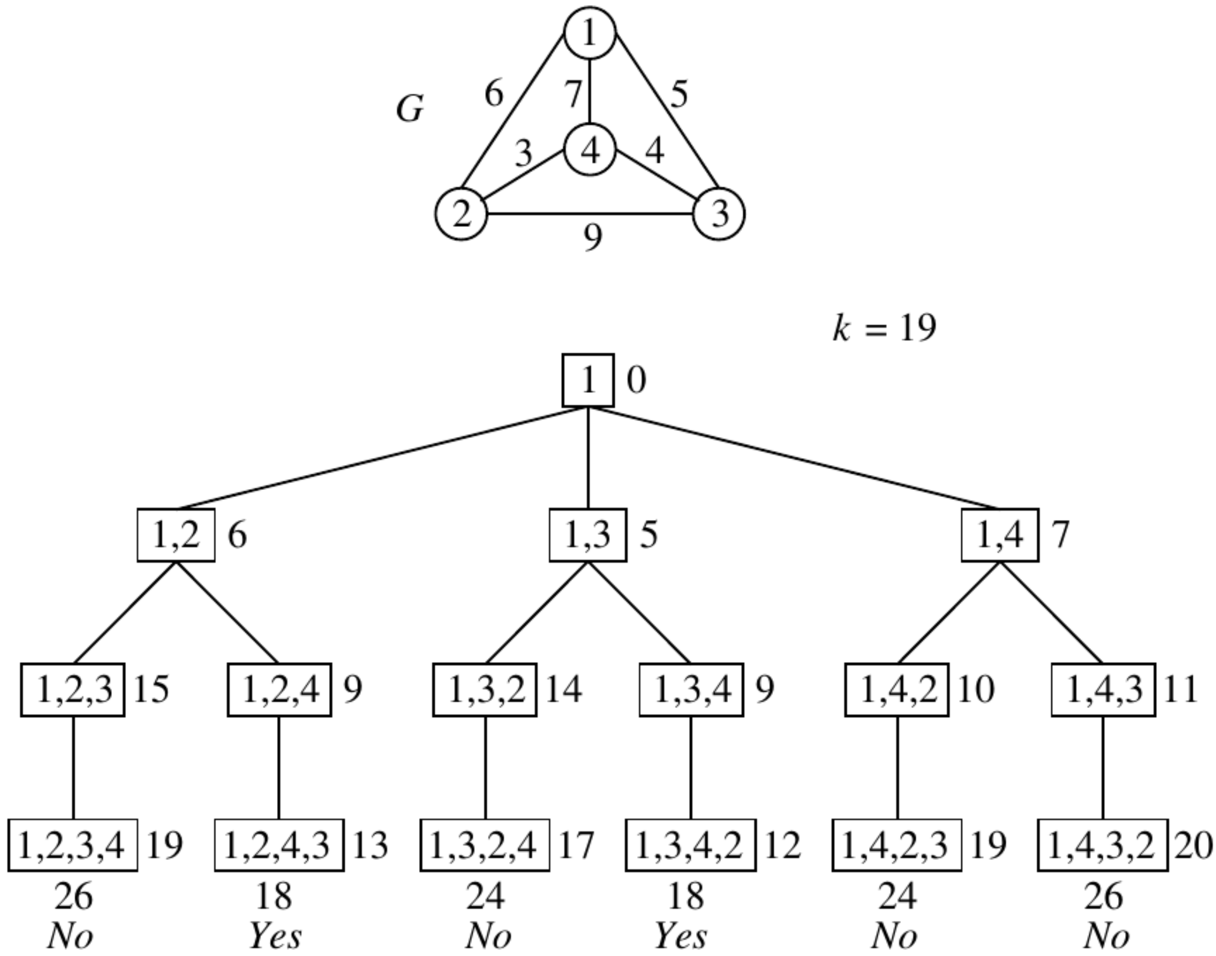


Figure 129: BFS traversal of the tree of Figure 128

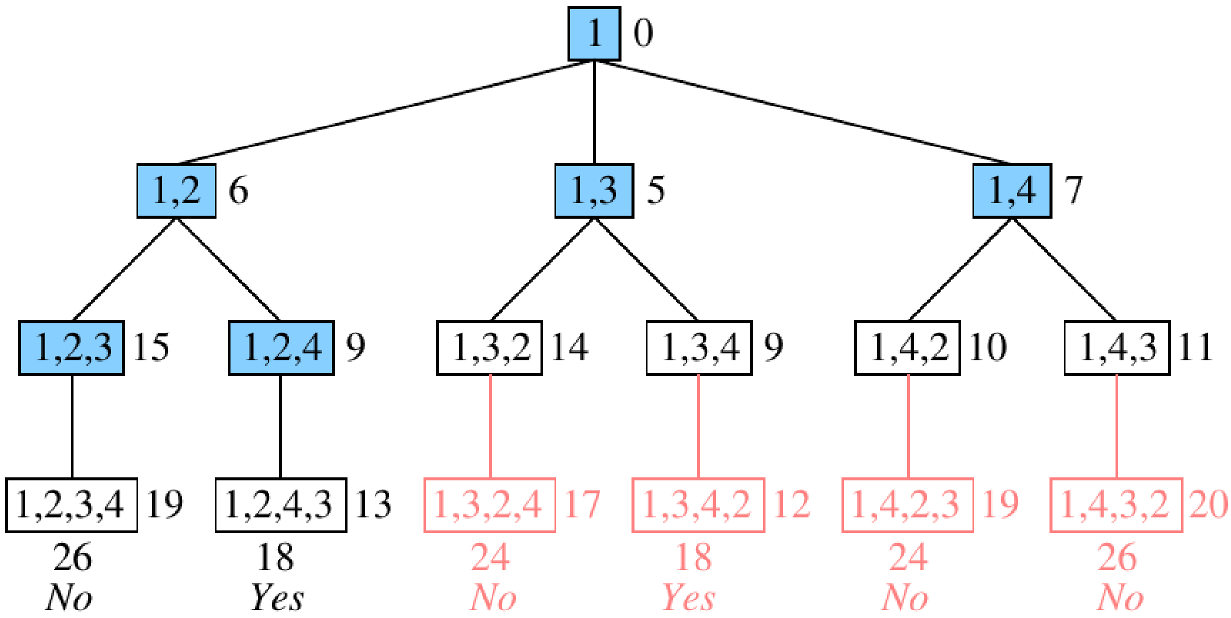


Figure 130: DFS traversal of the tree of Figure 128

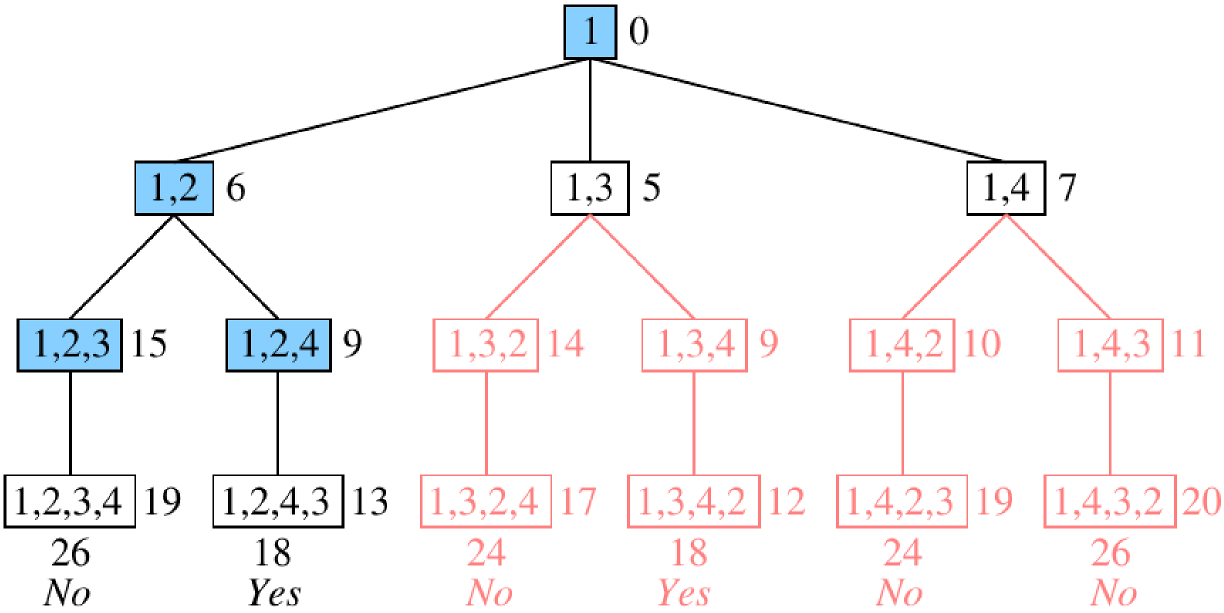


Figure 131: Heap traversal of the tree of Figure 128

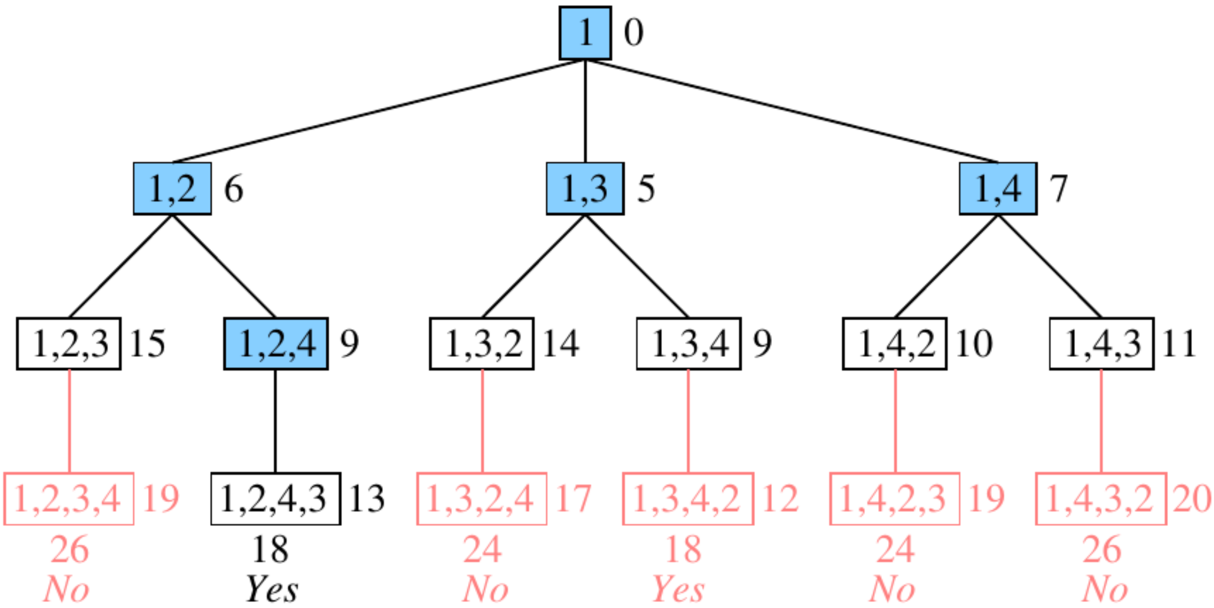
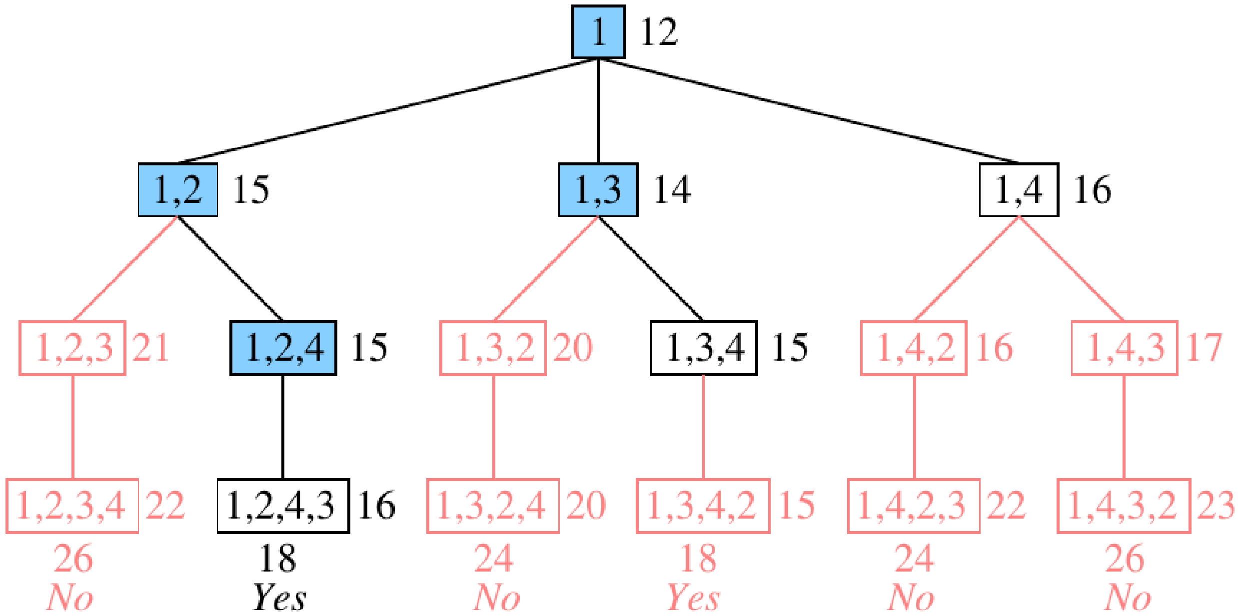


Figure 132: Heap traversal of the tree of Figure 128 with pruning



Generic description of a branch-and-bound algorithm

```
Initialize B = +∞, and Q = {(ε, D_init)}.
while Q is not empty {
    Take a node (C,D) from Q, and delete that node from Q.
    Determine all the children (C_1,D_1),(C_2,D_2),...,(C_k,D_k) of (C,D).
    for i = 1,2,...,k {
        If (C_i,D_i) is a leaf node {
            Compute the value F of the objective function at (C_i,D_i).
            If (F < B), replace B by F, and remember the solution (C_i,D_i).
        } else { /* (C_i,D_i) is a non-leaf node */
            Compute the lower bound L for the node (C_i,D_i).
            If (L < B), add (C_i,D_i) to Q.
        }
    }
}
Return B along with the stored best solution.
```

Figure 133: Branch-and-bound algorithm for TSP on the graph of Figure 128