Chapter 5

Branch-and-Bound

The idea

- The ***branch-and-bound algorithm*** is an improvement on the backtracking algorithm.
- The Similarity to backtracking
  - The tree state space
- The differences
  - Not any blind predetermined order to traverse the tree
  - Used only for optimization problems.

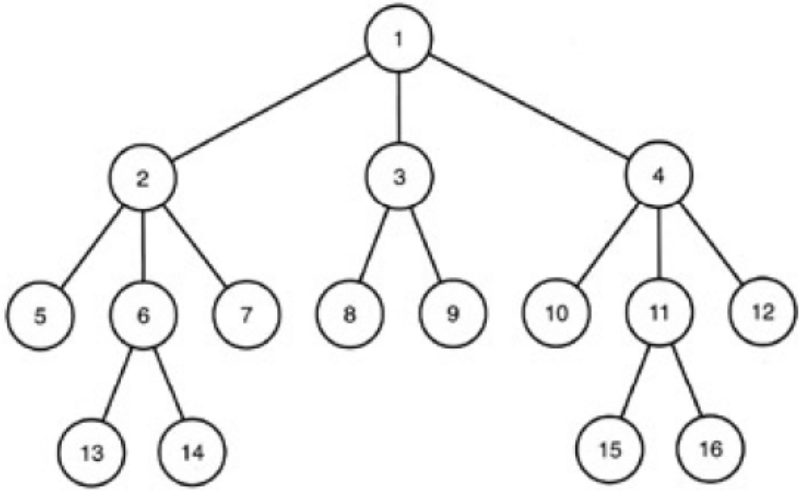
The idea

- A branch-and-bound algorithm computes a *bound* at a node to determine whether the node is promising.
- If that bound is no better than the value of the best solution found so far, the node is ***nonpromising***. Otherwise, it is ***promising.***
- It compares the bounds of promising nodes and visit the children of the one with the best bound.

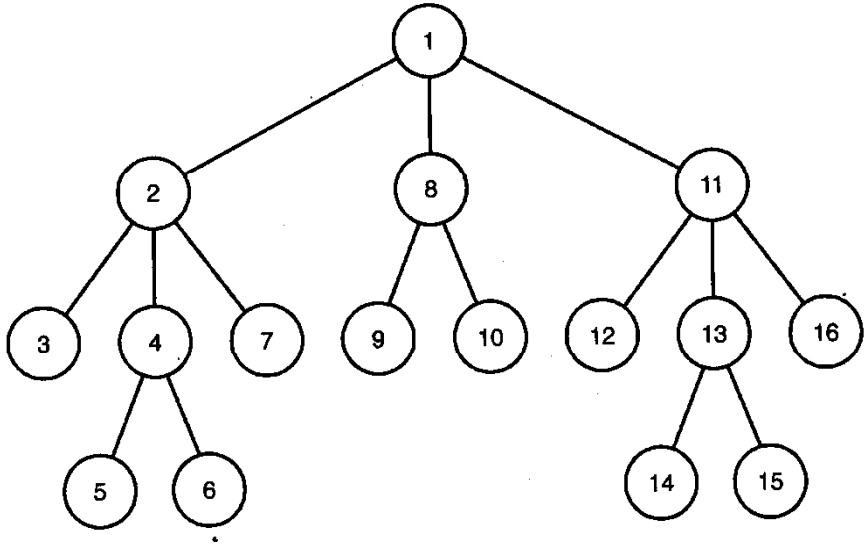
As is the case for backtracking algorithms, branch-and-bound algorithms are ordinarily exponential-time (or worse) in the worst case. However, they can be very efficient for many large instances.

Search strategies

Breadth-first search



Depth first search

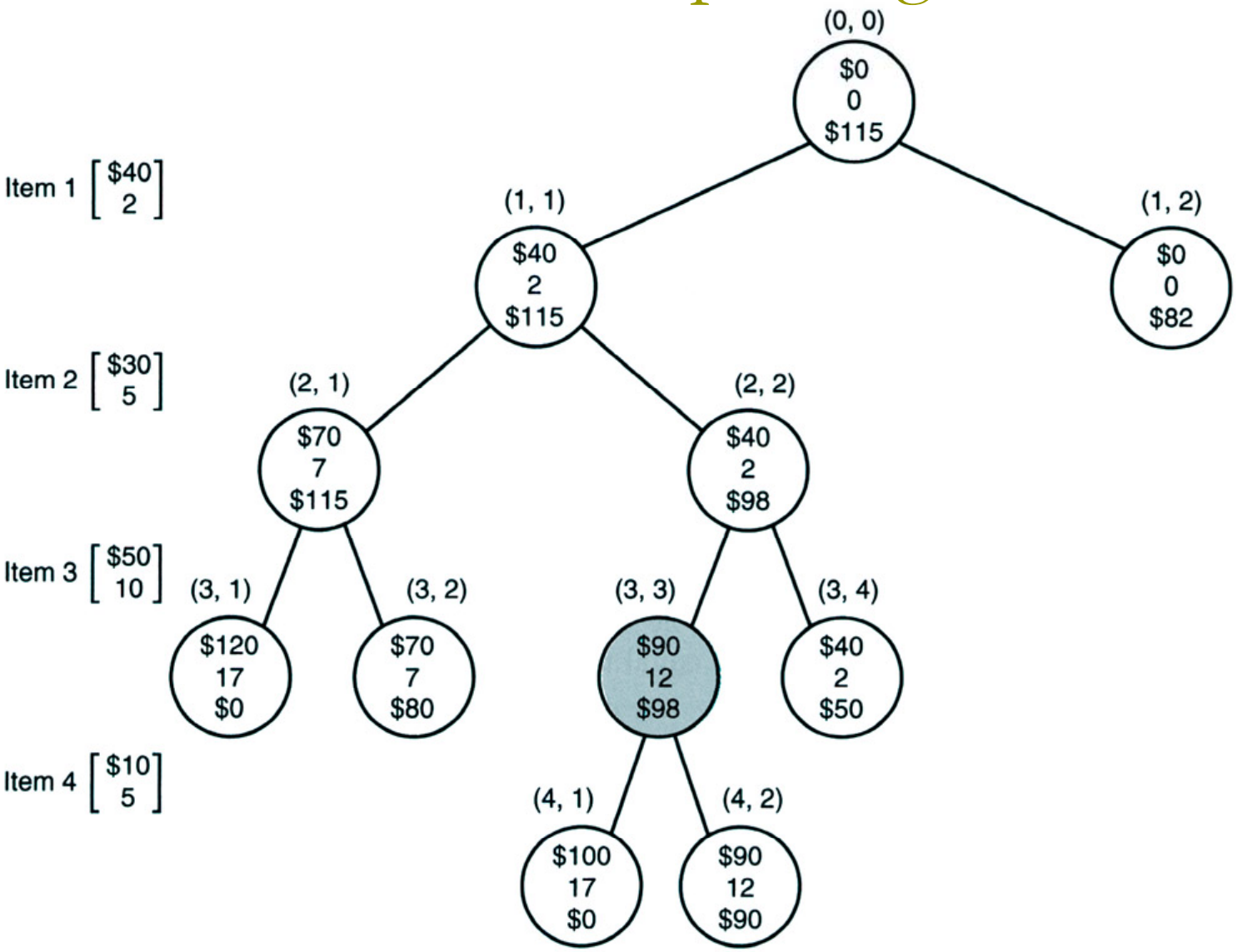


The algorithm

```
void breadth_first_tree_search (tree T) {
queue_of_node Q;
node u, v, initialize (Q); // Intialize Q to be empty.
v = root of T;
visit v;
enqueue (Q, v);
while (! empty (Q)) {
        dequeue (Q, v);
        for (each child u of v){
                visit u;
                enqueue (Q, u); }
}
}
```

The implementation of branch and bound consists of a simple modification to breadth-first search. Instead of using a queue, we use a priority queue.

The pruned state space tree produced using branch-and-bound pruning



The pruned state space tree produced using backtracking