Lecture 13.1

- Anurag Sharma & Shymal Chandra

Branch & Bound Algorithms

Branch-and-bound algorithms

- Branch-and-bound algorithms are very similar to backtracking algorithms in that a state space tree is used to solve a problem
- The differences are:
  - branch-and-bound method does not limit us to any particular way of traversing the tree (backtracking uses depth-first search)
  - branch-and-bound is used only for optimization problems

Cont.

- A branch-and-bound algorithm computes a number (bound) at a node to determine whether the node is promising. This number is a bound on the value of the solution that could be obtained by expanding beyond the node. If that bound is no better than the value of the best solution found so far, the node is non-promising so there is no need to expand beyond that node
- In fact, the last example we saw in backtracking (0-1 Knapsack problem) was an example of branch-and-bound (remember, a bound was calculated at each node…)
- Branch-and-bound however does not visit the nodes in some predetermined order (i.e. no depth-first search)

Breadth-first Search

- In branch-and-bound we use best-first search, which is a modification of another approach, breadth-first search
- Breadth-first search involves visiting the root first, then all nodes at level 1 of the tree, then all nodes at level 2 and so on
- Each node is checked and if it is non-promising, it is eliminated using the same rules as in a backtracking algorithm
- This process of pruning is same as in backtracking which reduces the portion of the state space tree

The 0-1 Knapsack Problem

Same example

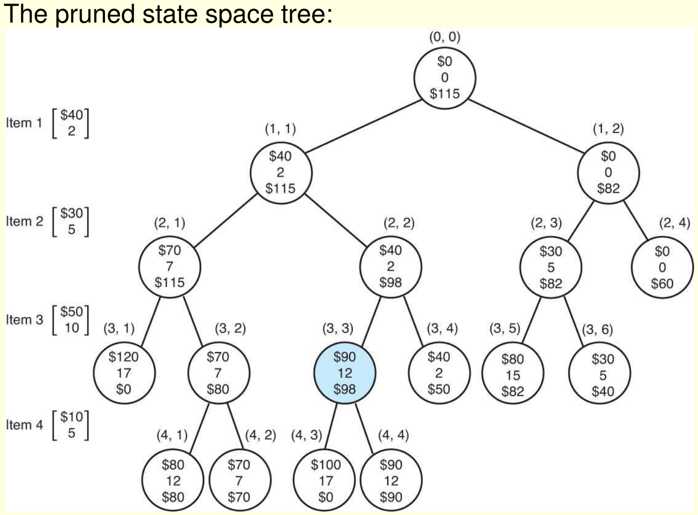
Example 6.1

Suppose we have the instance of the 0-1 Knapsack problem presented in Exercise 5.6. That is, $n = 4$, $W = 16$, and we have the following:

i	p_i	w_i	$\frac{p_i}{w_i}$
1	\$40	2	\$20
2	\$30	5	\$6
3	\$50	10	\$5
4	\$10	5	\$2

The 0-1 Knapsack Problem using Breadth-first Search

- The pruned state space tree:



Best-first Search

- In general, the breadth-first search strategy has no significant advantage over a depth-first search (backtracking)
- However, this can be improved by using our bound to do more than just determine whether a node is promising
- This is where best-first search comes in…
- After visiting all children of a given node, we can look at all the promising, unexpanded nodes and expand beyond the node with the best bound. In this way an optimal solution is arrived more quickly than if some predetermined order is used
- In the 0-1 Knapsack problem example, we pick only that child node to expand which has the highest maximum possible profit (bound)

The 0-1 Knapsack Problem using Best-first Search

- The pruned state space tree with branch-and-bound pruning (note – this is a smaller tree than the previous one which is the result of best-first search)