Scientific Computing

Feb 28, 2025

Announcements

→ HW 3 due Wednesday, March 5 at 11:59pm

→ Wednesday, March 5 is also the in-person midterm exam

→ Friday, March 7, no lecture, extra office hours while you work on take-home (time TBD)

Office Hours:

Mon + Fri

9:30am - 10:30am

Cudahy 307

Today

→ Branch and Bound

General Procedure:

```
function bb(S, best_sol = None):
    if best_sol is None:
        best_score = -∞
    else:
        best_score = score(best_sol)
    if |S| = 1:
        candidate = the one thing in S
        value = score(candidate)
        if value > best_score:
            return candidate
        else:
            return best_sol
```

else

branch into two, but could be 3, 4, etc.

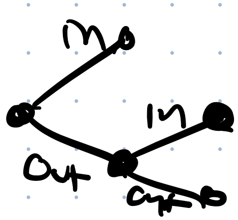
```
S_1, S_2 = branch(S)
if bound(S_1) > best_score:
    best_sol = bb(S_1, best_sol)
    best_score = score(best_sol)
if bound(S_2) > best_score
    best_sol = bb(S_2, best_sol)
    best_score = score(best_sol)
return best_sol
```

Relaxation

Let's try to figure this out with the knapsack problem.

Capacity: 14

item	weight	value
~~1~~	~~8~~	~~13~~
2	3	7
3	5	10
4	5	10
5	2	1
6	2	1
7	2	1



Branching: just like before

→ item 1 in or out

→ item 2 in or out

⋮

Bounding: Suppose we have put item 1 out and item 2 in. How can we find an <u>upper</u> bound on the best we could possibly do with the rest?

Remember:

* Greedy sol gives the wrong bound.

* "Add up the value of all remaining items" is technically an upper bound, but a useless one.

* Computing the UB can't be too slow.

Ideas?

Capacity: 14

item	weight	value
1	8	13
2	3	7
3	5	10
4	5	10
5	2	1
6	2	1
7	2	1

The trick is called relaxation: sometimes it's easier to find an UB if you adjust the problem to be more permissible.

Fractional Knapsack: You are allowed to take fractions of items

Capacity: 14

Example:

item	weight	value	fraction of item we're packing		weight / value
1	8	13	0.5	~~[0,1]~~	4 / 6.5
2	3	7	1	~~[0,1]~~	3 / 7
3	5	10	1		5 / 10
4	5	10	0.4		2 / 4
5	2	1			
6	2	1			
7	2	1			

14 / 27.5 ✓

$([0,1])^7$ $(\_,\_,\_,\_,\_,\_,\_)$

Not optimal, but it is a solution.

Theorem: A greedy and optimal solution can be found by: (1) ordering the items by value density $\frac{\text{value}}{\text{weight}}$

(2) taking items from the top in full until you can't anymore

(3) taking whatever fraction you can of the next item

We won't prove this, but you should think about it until you believe it.

Capacity: 14

item	weight	value	density		Take:	w/v
1	8	13	1.625	4	1/8	1/ $\frac{13}{8}$
2	3	7	2.333	1	1	3/7
3	5	10	2	2	1	5/10
4	5	10	2	3	1	5/10
5	2	1	0.5	5		
6	2	1	0.5	6		
7	2	1	0.5	7		

14/ 28.625

(If capacity = 10, we get value 21, a better solution than the regular knapsack.)

So, Fractional Greedy = Fractional Optimal

$\geq$ Regular Optimal.

$$\begin{pmatrix}\text{optimal score of}\\ \text{fractional knapsack}\end{pmatrix} \geq \begin{pmatrix}\text{optimal score of}\\ \text{regular knapsack}\end{pmatrix}$$

Thus, we can get an UB for B+B by computing the greedy fractional solution on remaining items.

Upper Bound for item 1 out, item 2 in: 

Capacity: 10* Back to 10 to be easier

item	weight	value	density			frac.	w/v
~~1~~	~~8~~	~~13~~	~~1.625~~	~~4~~			
2	3	7	2.333	1	✓		
3	5	10	2	2		1	5/10
4	5	10	2	3		0.4	2/4
5	2	1	0.5	5			
6	2	1	0.5	6			
7	2	1	0.5	7			

upper bound of
$7 + 10 + 4 = 21$

Let's work out the B+B tree.

Greedy start!

Capacity: 10

item	weight	value	density	
1	8	13	1.625	4
2	3	7	2.333	1
3	5	10	2	2
4	5	10	2	3
5	2	1	0.5	5
6	2	1	0.5	6
7	2	1	0.5	7

Three greedy solutions:

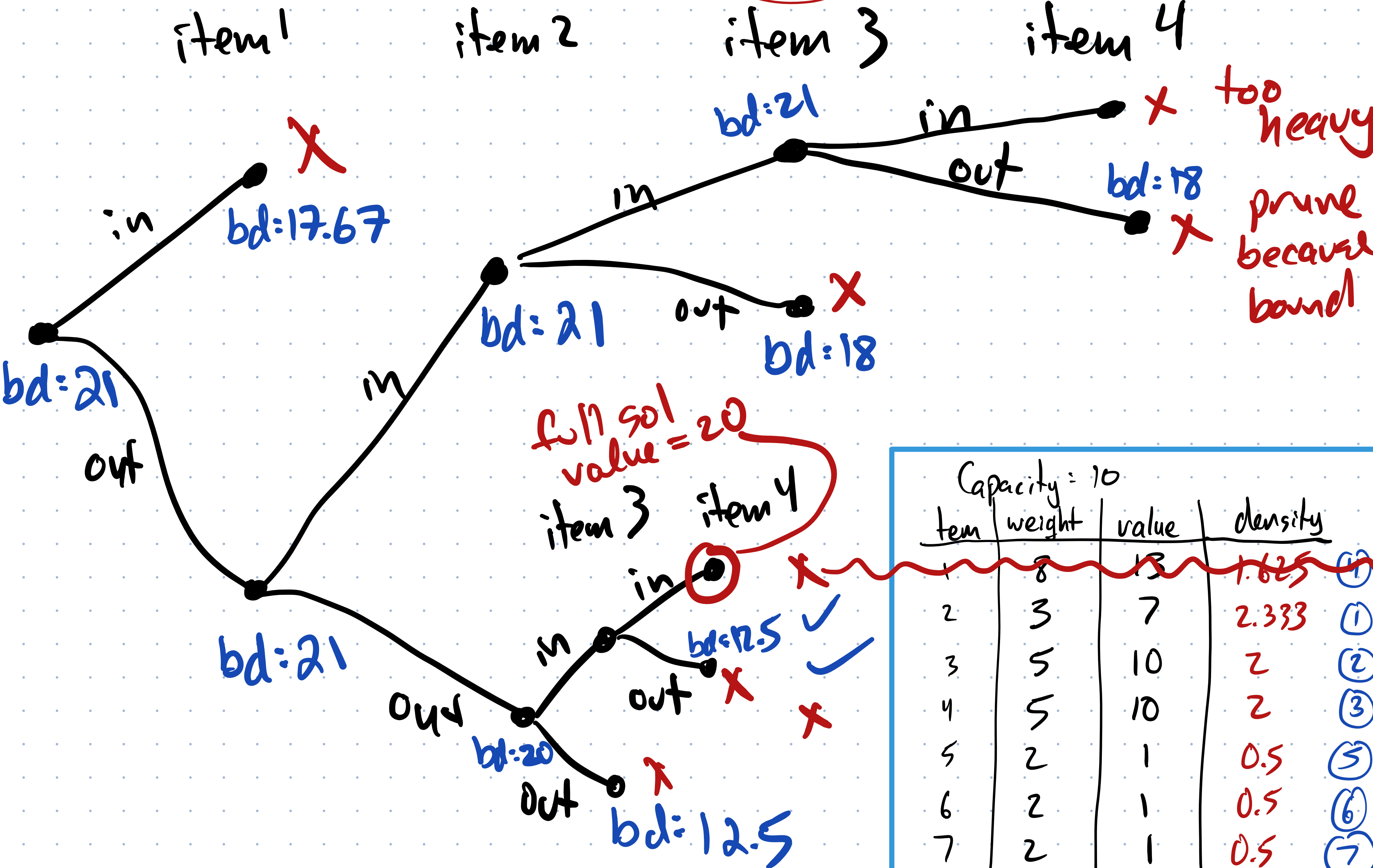
Most value:
$13+1=14$
(items 1,5)

Lightest:
items 5, 6, 7, 2
$1+1+1+7=10$

Most dense:
items 2, 3, 5
$7+10+1=18$

B+B tree

best so far: 18



Capacity: 10

item	weight	value	density	
~~1~~	~~8~~	~~13~~	~~1.625~~	~~4~~
2	3	7	2.333	1
3	5	10	2	2
4	5	10	2	3
5	2	1	0.5	5
6	2	1	0.5	6
7	2	1	0.5	7

Compare to backtracking alone!