R18 Subset Sum Variants

Knapsack

- Input: Knapsack with size S, want to fill with items each item i has size s_i and value v_i
- Output: A subset of items with sum s_i <= S maximizing value sum v_i
- Example: Items{(s_i,v_i} = {(6,6),(9,9),(10,12)}, S=15
  -Subproblems
  Idea: Is last item in an optimal knapsack? (Guess!)
  if yes, get value v_i and pack remaining space S-s_i
  if no, then try to sum to S using remaining items
  x(i,j): maximum value by packing knapsack of size j using items 1 to i

-Relate
$x(i,j) = max\{v_i + x(i-1, j-s_i)\ if j >= s_i\ or\ x(i-1,j)\ always\}$
for i in {0,...,n}, j in {0,...,S}

-Topo
subproblems x(i,j) only depend on stricly smaller i, so acyclic
-Base
x(i,0) = 0 for i in {0,...,n}
x(0,j) = 0 for j in {1,....,S}
-Original
Solve subproblems via recursive top down or iterative bo bottom up
Maximum evaluated expression is given by x(n,S)
Store parent pointers to reconstruct items to put in knapsack
-Time
number subproblems O(nS)
work per subproblem O(1)