13 Feb 2019 The Knapsack Problem (§6.4)

Items $i = 1, 2, \ldots, n$

w_i = "weight of item i" ≥ 0

v_i = "value of item i" ≥ 0

} assume integers

Given W.

Question. Select a subset of items whose combined weight is $\leq W$ and whose combined value is maximum.

{Valid solutions} = {subsets of items whose combined weight is at most W}

Lemma. Every non-empty valid solution has one element, i, combined with subset of the other elements $1, 2, \ldots, i-1, i+1, i+2, \ldots, n$.

☑ TRUE

☒ LESS USEFUL THAN IT COULD BE.

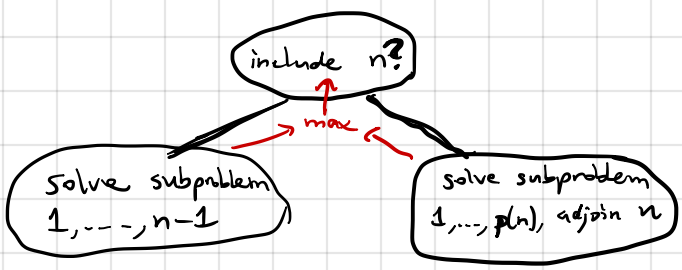
... also some invalid solutions can be formed by combining i with a subset of $1, 2, \ldots, i-1, i+1, \ldots, n$.

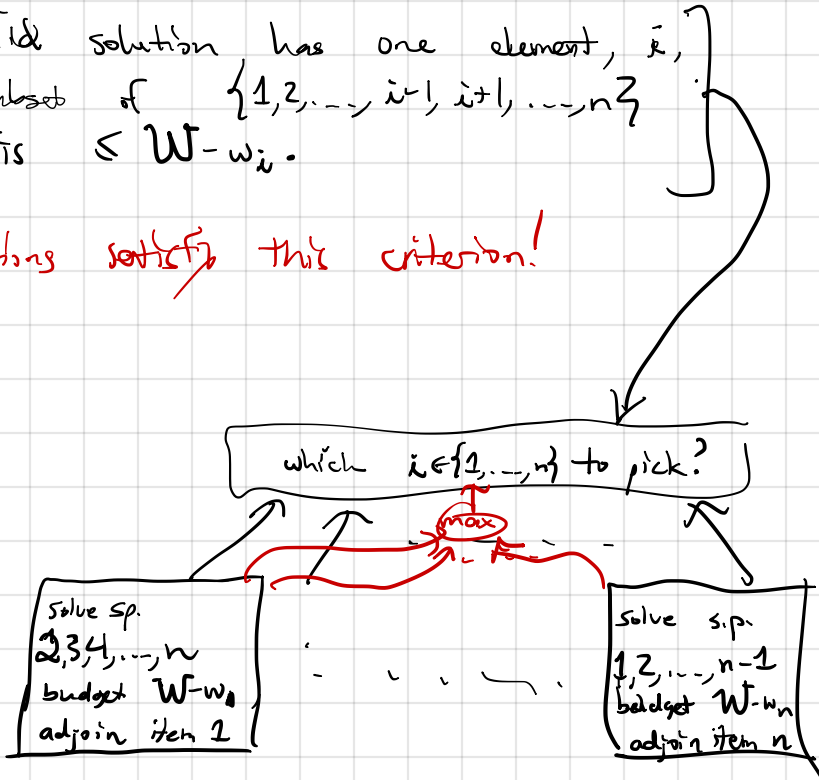
Lemma. Every non-empty valid solution has one element, i, combined with a subset of $\{1, 2, \ldots, i-1, i+1, \ldots, n\}$ whose total weight is $\leq W - w_i$.

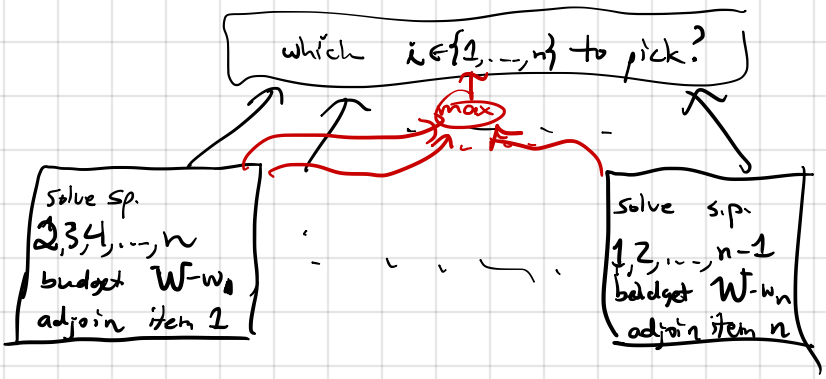
☑ TRUE.

☑ No invalid solutions satisfy this criterion!

Cartoon of Weighted Interval Sched Alg.







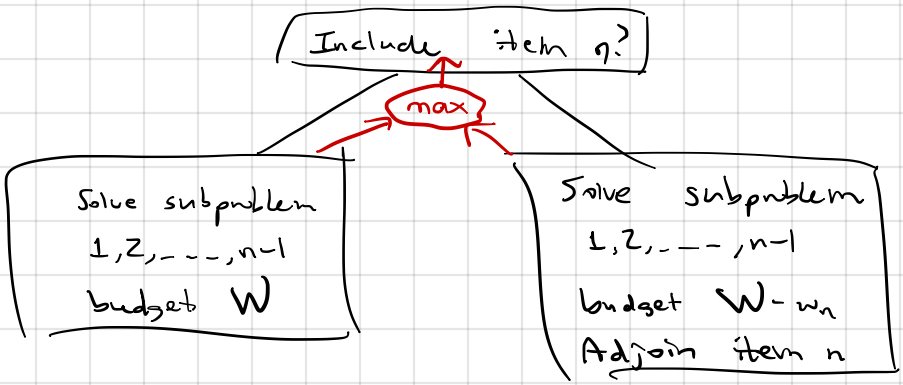
Running time? Solving one problem with n items reduces to solving n subproblems each with $n-1$ items.

$$T(n) \leq n \cdot T(n-1) + O(n)$$

↳ solves to $T(n) = 2^{O(n \log n)}$.

Lemma. Every non-empty valid solution has one element, i, combined with a subset of $\{1,2,\ldots,i-1,i+1,\ldots,n\}$ whose total weight is $\leq W - w_i$.

Alternative Lemma. Every valid solution either contains item n or it doesn't. In the latter case, it is a knapsack solution with budget W and item set $1,\ldots,n-1$. In the former case it is formed by combining item n with ... a knapsack solution with budget $W-w_n$ and item set $1,2,\ldots,n-1$.



PLAN: Dynamic programming table should store, for each $i \in \{0, 1, \ldots, n\}$ and $B \in \{0, \ldots, W\}$, the max value of a knapsack solution with item set $1, 2, \ldots, i$ and budget B.

function REC_KNAPSACK(i, B):

if $T[i, B] \neq$ NULL return $T[i, B]$.

else

if $i = 0$ or $B = 0$ $T[i, B] = (0, \emptyset)$.

else

Let (v_{no}, S_0) = REC_KNAPSACK$(i-1, B)$

Let (v_{yes}, S_1) = REC-KNAPSACK$(i-1, \max\{0, B - w_i\})$

if $v_{no} > v_{yes} + v_i$

$T[i, B] = T[i-1, B]$

else

$T[i, B] = (v_{yes} + v_i, S_1 \cup \{i\})$

endif

endif

return $T[i, B]$.

Running Time: Table size $O(n \cdot W)$. Work $O(1)$ per table entry.

$\Rightarrow O(nW)$ running time.

"pseudopolynomial": potentially exponential in the # of bits in the input since writing W requires only $O(\log W)$ bits.

But polynomial in the magnitude of the input numbers.