REC_OREN(i): // Returns the max weight valid subset of $1, \dots, i$ along with its combined weight.

with "memoization"

if $i = 0$:

return $(0, \emptyset)$

$T[i]$ stores previously computed value of REC_OREN(i) if the function has been called before.

if $i > 0$:

check if $T[i] \neq NULL$. If so return $T[i]$.

Else

let (w_0, S_0) = REC_OREN($i-1$)

let (w_1, S_1) = REC_OREN($p(i)$)

if $w_i + w_1 > w_0$ $\quad T[i] = (w_i + w_1, \{i\} \cup S_1)$

else $\quad T[i] = (w_0, S_0)$.

return $T[i]$.

Running time?

Time spent on the first call to REC_OREN(i): $O(1)$.

Time spent on subsequent calls: $O(1)$ per subsequent call.

At most $n-i$ subsequent calls.

Total time on REC_OREN(i) = $O(n)$.

Summing over i: algorithm takes $O(n^2)$.

More accurate accounting: Every time REC_OREN(i) is called resulting in a "cache hit" ($T[i]$ is already stored) charge that $O(1)$ operation to the calling procedure.

Time spent on first call to REC_OREN(i): $O(1)$

subsequent calls: $\emptyset$

Sum over $i = 1, 2, \dots, n$: $O(n)$

Actual asymptotic running time of W.I.S. after sorting by finish time.