...is also optimal

Algorithm: $(b_1, b_2, b_3 \ldots b_n)$

Set of n boxes b w/ weight w.

Set of trucks w/ max weight W

while ~~input set~~ station set has boxes

t = current truck

~~for~~ W = max weight of t

while W > w

load box onto truck

w += w_i

remove box from set

get next box

send off truck //remove truck

Analysis: outer while loop runs O(n) times

inner while loop runs O(n) times

Assume loading & removing box take O(1)

because it is at front of whatever data structure

(~~can take~~ O(n) ~~if reversed~~) for boxes

Assume sending off truck & moving to next truck take O(1) (could take O(t)) for trucks

best-worst case → O(1)·O(1) + O(n)·O(n)

$O(n^2)$

Proof: Greedy Stays Ahead

Say we have the greedy algorithm which uses m trucks in total, so truck one $= t_1, \ldots,$ truck $m = t_m$. If we have a sequence of packages $1, 2, \ldots, n$ and each package has a weight w_i and is placed in a truck t_m, due to the fact that packages have to be sent in order there is a constant sequence of assigning packages to trucks. And the algorithm will output $A = \{a_1, \ldots, a_k\}$

If this greedy solution A is not optimal, then there is another, optimal solution $O = \{o_1, \ldots, o_k\}$ where the # of trucks used $< m$. There must be some point j where the # of trucks used in the algorithms differ $(\exists j : (t_j = t_j' \wedge \forall i < j : t_i = t_i'))$

Since $t_j \neq t_j'$, either $t_j > t_j'$ or $t_j < t_j'$.

If $t_j > t_j'$ then the greedy algorithm could have switched trucks before the optimal algorithm but since the greedy algorithm only switches trucks when the current truck is full so the optimal solution could not fit another package this scenario cannot happen.

If $t_j < t_j'$ then the optimal algorithm switched trucks before the greedy algorithm.

If we have a solution where we have a

sequence of assignments to trucks t_i'' s.t. $t_i'' = t_i$ if $i \leq j$ and $t_j'' = t_i'$ if $i > j$. Then, package j has been moved from truck t_j' to truck $t_j' - 1$

t_j' cannot be overpacked since it has 1 less package than before and $t_j' - 1$ cannot be overpacked since it has the same amount of packages as the greedy solution. This means that we have a new solution which matches the greedy algorithm and does it more efficiently. This means that the original optimal solution is not optimal and we have a contradiction

So it must be the case that the greedy algorithm is the optimal solution