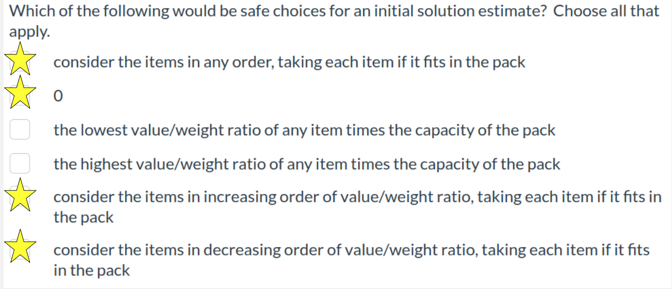
# 0-1 Knapsack

Initial solution value – upper bound or lower bound?

- maximization problem, so bigger is better
- update if solution is better → “safe” is an estimate which is not good enough → “not good enough” is smaller → looking for lower bound on the solution value



Which of the following would be safe choices for an initial solution estimate? Choose all that apply.

★ consider the items in any order, taking each item if it fits in the pack

★ 0

□ the lowest value/weight ratio of any item times the capacity of the pack

□ the highest value/weight ratio of any item times the capacity of the pack

★ consider the items in increasing order of value/weight ratio, taking each item if it fits in the pack

★ consider the items in decreasing order of value/weight ratio, taking each item if it fits in the pack

fill the pack with as much of each item as will fit (fractional amounts allowed), considering items in order of decreasing value/weight ratio

# Strategies

A good bound function depends on the specific nature of the problem and what you can exploit about its structure.

But we've seen a few general tactics that might serve as starting points –

- value so far + best single choice ✕ number of choices left
- value so far + best single next choice ✕ number of choices left
  - only safe if all choices are available at each stage (e.g. knapsack but not TSP)
- value so far + greedy solution from that point
  - only safe if greedy can do better than the actual solution (true for knapsack, not for TSP and max independent set)
- consider trivial bound and what is over/undercounted
  - e.g. max independent set - |S| overcounts because a vertex and its neighbor can't both be in the set; |S|-mindeg(S) addresses that for one vertex picked

# Initial Solution Estimate

Upper or lower bound?

- safe = conservative = worse than the optimal
  - if your estimate is better than the optimal, you'll prune away the branch containing the optimal as not good enough

Note: bound is on the value of the optimal solution, not the value of any legal solution

- e.g. “upper bound” does not mean that it needs to be worse than all possible legal solutions – and that wouldn't help you prune anything at all

# Initial Solution Estimate

Any legal solution is a safe estimate – it will be no better than the optimal.

- greedy can be a good strategy
  - e.g. greedy TSP – take cheapest edge to not-yet-included vertex
  - e.g. maximal independent set – take any legal vertex until there are no more

But you may be able to get a tighter estimate without having an actual solution in mind.

(Then safety is important to establish.)

  - e.g. 2*MST ≥ optimal TSP solution

# Additional Strategies

How much can be pruned depends on two things:

- the tightness of the bound function
- the value of the best solution so far, which depends on:
  - how well we can estimate its value at the beginning, and/or
  - how quickly a good solution is found

How to search the best branches first?

- modified depth-first search: at each step, order alternatives to explore most promising first
- best-first search: choose most promising subproblem first
  - priority queue stores discovered nodes of the search tree with priority corresponding to cost of partial solution
    - A* heuristic: use bound function (cost of any complete solution stemming from the partial solution)
  - downside of BFS is potentially exponential size of the queue