Knapsack

Items i_1 i_2 ... i_n

Value v_1 v_2 ... v_n

Weight w_1 w_2 ... w_n

Bag of fixed weight capacity W

Choose a set of items that fits in your bag

to maximize value
- unlimited copies of each item
- one copy of each item

Unlimited copies

Suggestion

Order items, say by $\frac{v_j}{w_j}$

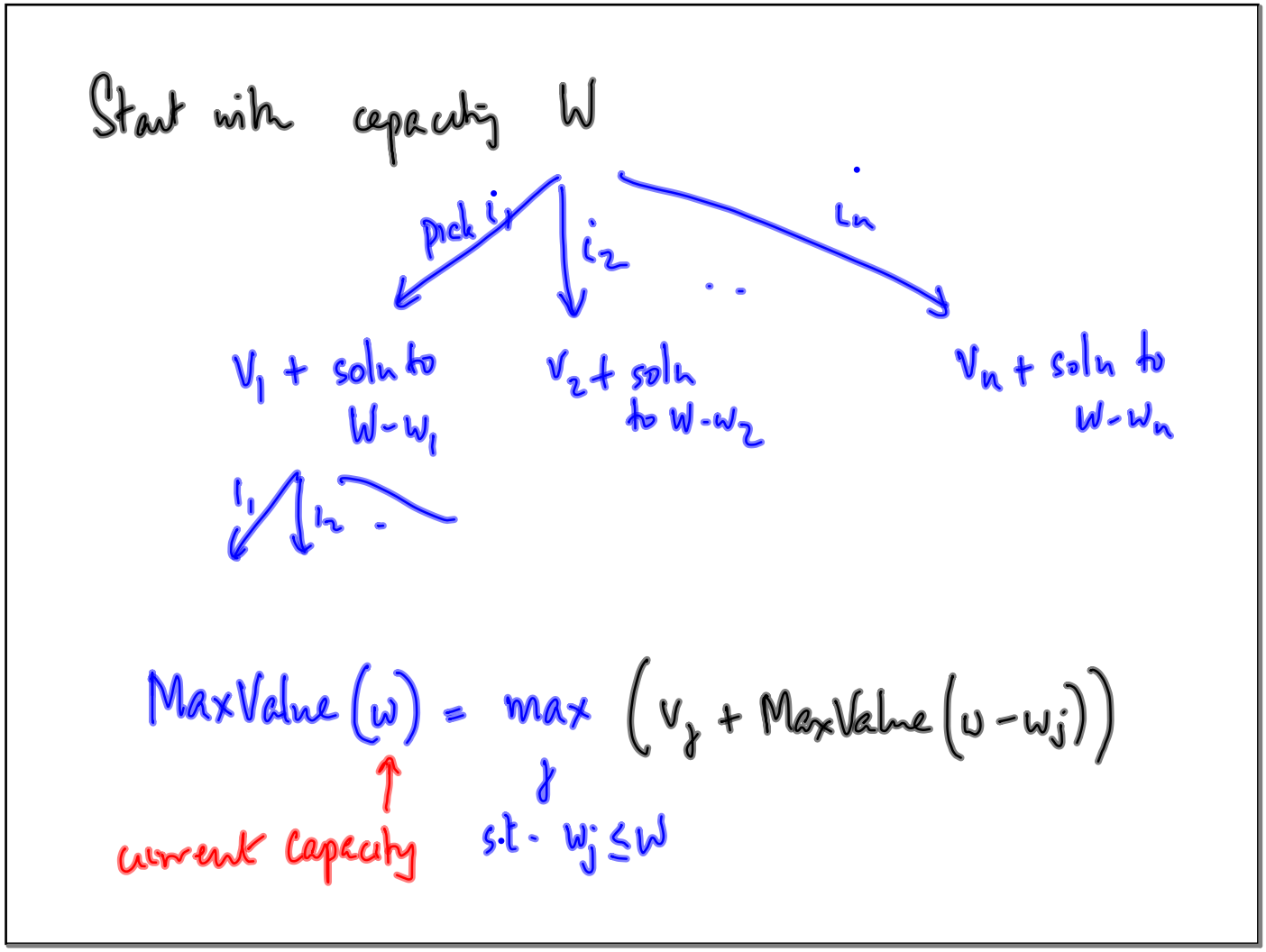
Optimal solution has n_j copies of item i_j

⋮

OR

Just pick items one at a time

Picking i_j reduces capacity by w_j, adds value v_j


Start with capacity W
pick i1
i2
in
v1 + soln to W-w1
v2 + soln to W-w2
vn + soln to W-wn
i1
i2
MaxValue(w) = max_j (vj + MaxValue(w-wj))
current capacity
s.t. wj ≤ w

Final solution is MaxValue(W)

– need MaxValue(0), MaxValue(1), ..., MaxValue(W-1)

DP of order $O(n \cdot W)$

Depends on W, and the "fractions" of W generated by the w_j's

Input size of value W is $\log W$ (no. of bits to write W)

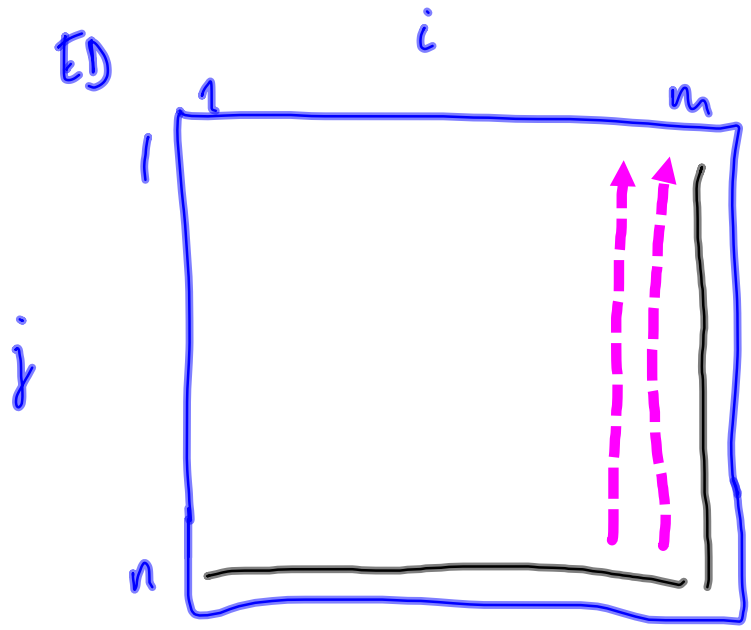
Not poly time

Space: Do we need space proportional to W to compute MaxValue(W)?

Naively we need such an array to store subproblem values

Subproblems reachable from MaxValue(w) are bounded by max w_j

What about edit distance



O(min(m,n)) space suffices — row wise or columnwise

only need current & previous row/column

Knapsack, limited (=1) copy of each item

MaxValue(w) choose i_j MaxValue($w - w_j$)

cannot choose i_j again

Need to incorporate list of available items as a parameter

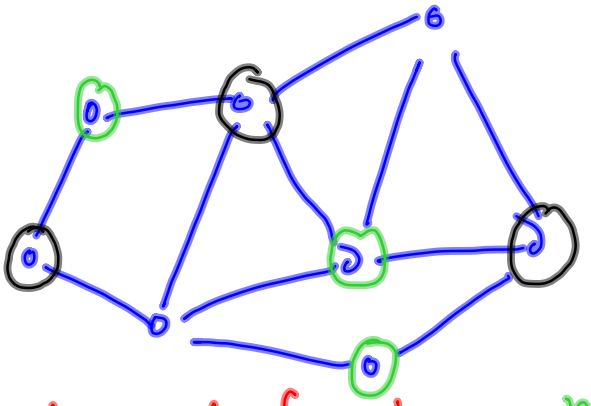
Fix some order of items $i_1, i_2, \ldots, i_n$

MaxValue(w, k) Optimum solution for capacity w and items available are $i_1, i_2 \ldots i_k$ (or $i_{k,k+1} \ldots i_n$)

MaxValue(w, k) ↑ $i_1, i_2, \ldots, i_k$

- Choose i_k → MaxValue$(w - w_k, k-1) + v_k$
- skip i_k → MaxValue$(w, k-1)$

GRAPHS



Independent set of vertices – no edges between them

Aim: Find an independent set of max size

MIS

Interval scheduling problem

requests $r_1, \ldots, r_n$ each $r_i = (s_i, f_i)$

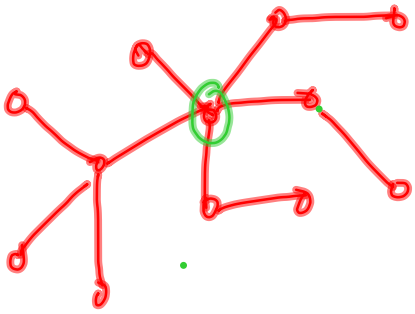
graph $V = \{r_1, \ldots, r_n\}$

$E = \{(r_i, r_j) \mid (s_i, f_i)$ & (s_j, f_j) overlap$\}$

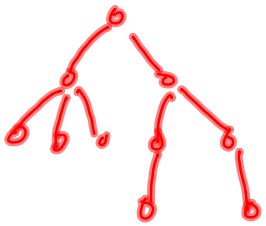
Solution is an MIS in this graph

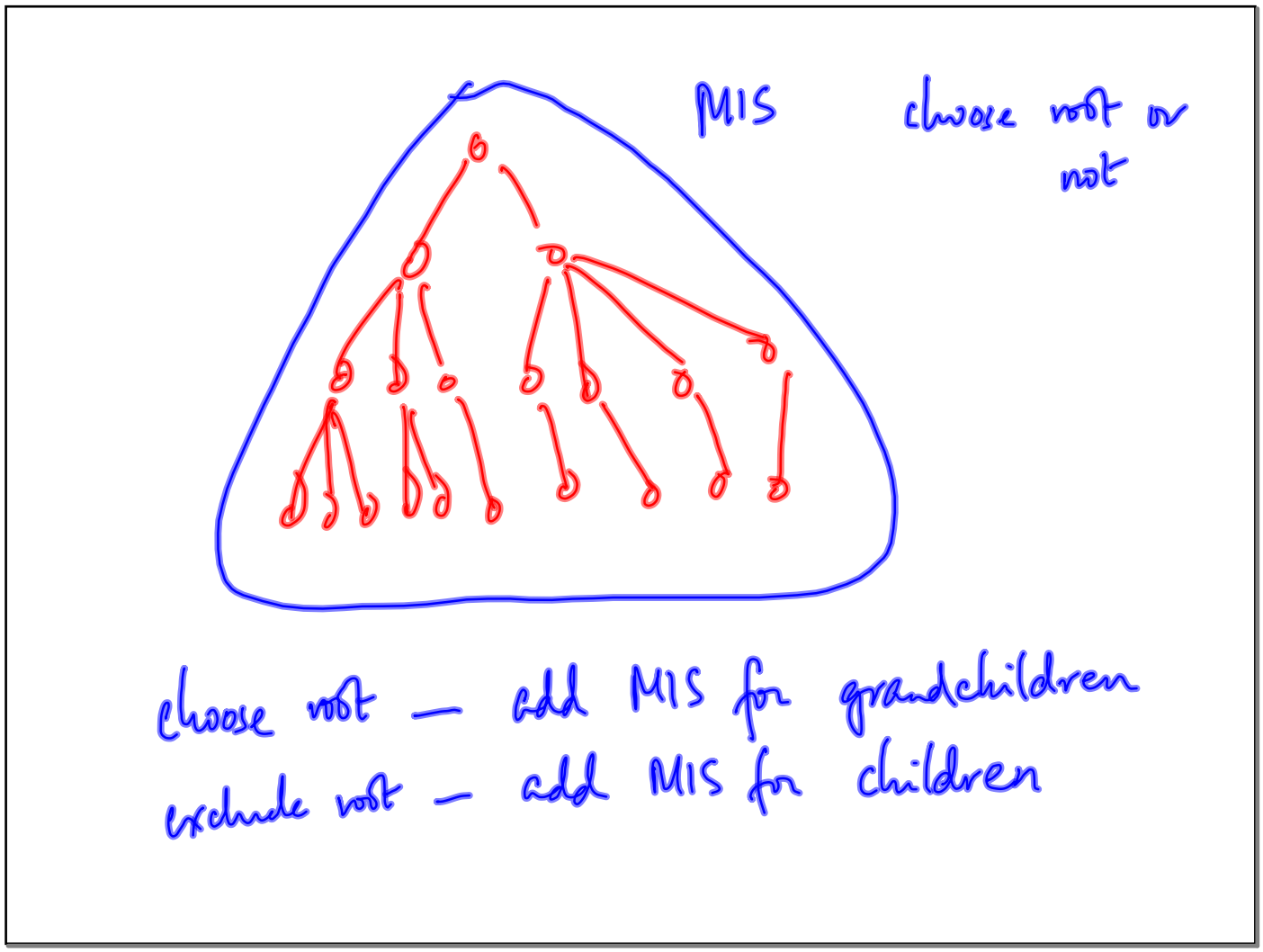
In general MIS is hard

But, it has a nice solution over trees

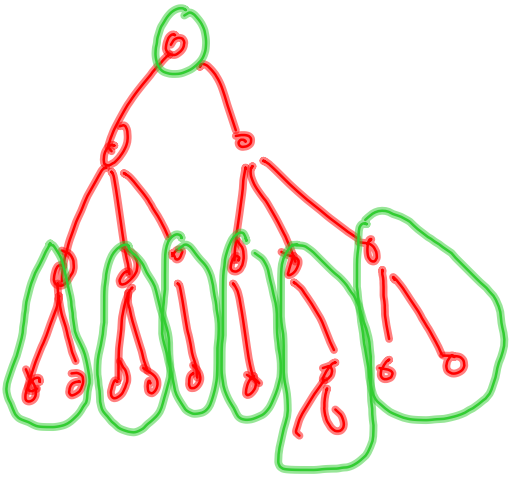


Pick a root and "hang" up the tree

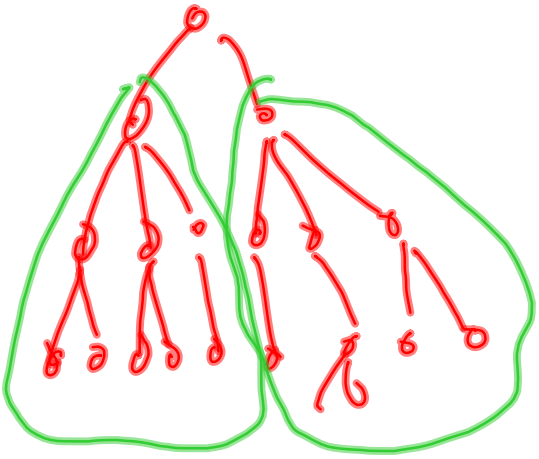



MIS
choose root or not
choose root – add MIS for grandchildren
exclude root – add MIS for children

Choose not



Exclude not



Also bottom up?

Tree structure controls/limits no. of subproblems