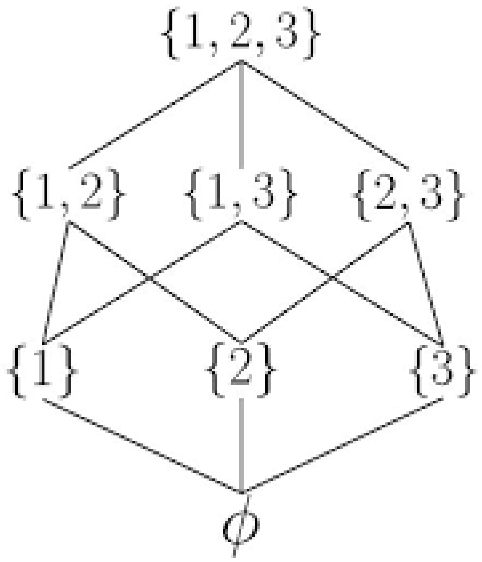
DP for Optimal Subset Selection Problem
(Abstract View)

Optimal Subset Selection Problem

- Let S be a set of n item: $x_1, x_2, \ldots, x_n$
- Consider any subset T from S, define
  - Eligibility: valid?(T) = T/F
  - Reward:
    - $R(T) \rightarrow \mathbb{R}_{\geq 0}$
    - Monotone: $R(T \cup \{x_j\}) = f(R(T), R(x_j)), \quad f$ is monotone
- Problem: find valid T* with the minimum reward

Solution space



- Since we only need the optimal solution, can we partition the powerset of S into exhaustive and disjoint partitions, and only keep the best solution in each partition?
  - A natural way is the prefix partitioning
  - S_k := set of first k items of S

$x_{k+1} \circ S_k$

exhaustive

$$\forall T, T \in S_k \vee T \in S_{k+1} \setminus S_k$$

disjoint

Recurrence

- Let T_k be the optimal subset in S_k, T^c_k be the optimal subset in $S_{k+1} \setminus S_k$
- Can we grow T_k to T_{k+1}? ➔ Greedy
- Can we grow T_k or T^c_k to T_{k+1} ➔ Dynamic Programming

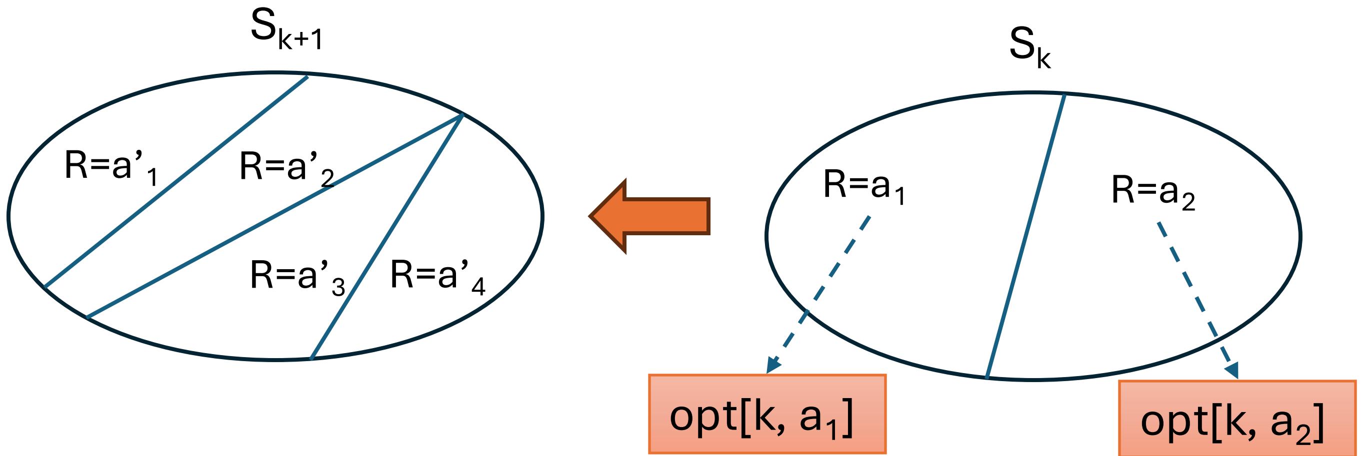
Exercise: Draw and Trace these ‘compressed solution’ spaces for a simple (knapsack) problem.

Knapsack Problem

- Try 1:
  - Grow T_k with x_{k+1}
    - Only choice: take x_{k+1} if the result is valid (i.e., not violating the weight constraint)
  - Observe that this may not be T_{k+1}
    - T_k cannot take x_{k+1} (weight limit)
    - $T_k^{(2)}$ can take x_{k+1}, and result in better solution
      - Pushing it further, all $\{\tau \mid \tau \in S_k, R(\tau) > R(T_k) - R(x_{k+1})\}$ need to be considered!

$T_k^{(2)}$: 2nd-best solution in S_k

If we do not have a bound on $R(x_{k+1})$, then $R(\tau)$ can be as small as 0
➔ **Any** subset in S_k may be grown into T_{k+1} !!!



Nevertheless, among subsets with the same reward value, only need to “keep” one subset
➔ (solution space) Compression achieved !

Exercise: Balanced Partition Problem

- (1) Do the similar reasoning on BP
- (2) Find a (smallest) example where T_{k+1} is not grown from T_k

Exercise: Balanced Partition Problem

- (1) Do the similar reasoning on BP
- (2) Find a (smallest) example where T_{k+1} is not grown from T_k

$S = \{2, 10, 8, 5\}$

$T_1 = \{2\}$

$T_2 = \{2\}$

$T_3 = \{2,8\}$

$T_4 = \{2,8\}$

$T_3^{(2)} = \{2, 10\}$

$T_4 = \{2,10\}$

Knapsack Problem

- Try 2:
  - What about $opt[W] = \min_{x \in S} \{opt[W - w(x)] + R(x)\}$

Knapsack Problem

- Try 2:
  - What about $opt[W] = \min_{x \in S} \{opt[W - w(x)] + R(x)\}$
  - Bug
    - opt[W] and opt[W-w(x)] may both require using the same x
    - But each x has only one supply!
    - ➔ Must track available items
  - Fix: $$opt[U, W] = \min_{x \in U} \{opt[U \setminus \{x\}, W - w(x)] + R(x)\}$$
    - Correct but useless
      - Subproblems in U alone require enumerating ALL subsets
      - ➔ Use the compression idea

Always valid as long as opt solution contains at least one x

We consider U \ {x} because we need to consider set of solutions that contains x

Knapsack Problem

We consider U \ {x} because we need to consider set of solutions that contains x

- Try 2: $opt[U,W] = \min_{x \in U} \{opt[U \setminus \{x\}, W - w(x)] + R(x)\}$
  - Partition all subsets into S_k, k in [n], based on the largest element it contains
  - The recurrence becomes: $opt[U,W] = \min_{i \in [n]} \{opt[S_i, W]\}$
  - Overlapping substructure emerges!

Knapsack Problem

- Try 2: $opt[U, W] = \min_{x \in U} \{opt[U \setminus \{x\}, W - w(x)] + R(x)\}$
  - Partition all subsets into S_k, k in [n], based on the largest element it contains
  - The recurrence becomes: $opt[U, W] = \min_{i \in [n]} \{opt[S_i, W]\}$
  - Overlapping substructure emerges!

$$opt[S_{i+1}, W] = \min\left(opt[S_i, W], opt[S_i, W - w(x_{i+1})] + R(x_{i+1})\right)$$

Opt solution do not uses x_{i+1}

Opt solution uses x_{i+1}