Dynamic Programming

1 Maximum Weight Interval Selection

Input: A collection of intervals where each interval has an associated value.

Output: A collection of non-overlapping intervals that maximizes the total value.

Algorithm. Sort the input by end time. Let $v_1, \ldots, v_n$ be the values of the intervals in this order. Let $\text{opt}(i)$ be the best value I can get over the first i intervals. Then

$$\begin{aligned} \text{opt}(0) &= 0, \\ \text{opt}(1) &= v_1, \end{aligned}$$

$$\text{opt}(i) = \max \begin{cases} \text{opt}(i-1) & \text{if we don't include the } i\text{th interval,} \\ v_i + \text{opt}(P(i)) & \text{if we do,} \end{cases}$$

where $P(i)$ is the index of the last interval that doesn't overlap with the ith.

Naive recursion

function OPT(n)
  if $n = 0$ **then**
    return 0
  if $n = 1$ **then**
    return v_1
  return $\max\big(\text{OPT}(n-1),\ v_n + \text{OPT}(P(n))\big)$

$$T(n) = 2T(n-1) + \Theta(1)$$

Bottom-up

$W \leftarrow$ an array of length $n+1$
$W[0] \leftarrow 0$; $W[1] \leftarrow v_1$

$p_idx \leftarrow 0$; let P be an array of length $n+1$
$P[0] \leftarrow 0$; $i \leftarrow 0$
while $i \leq n$ **do**
  if $p_idx = 0$ **or** the p_idxth interval does not overlap with the ith interval **then**
    increase p_idx until it overlaps with the ith interval
  $P[i] \leftarrow p_idx - 1$

$i \leftarrow i + 1$

for $i = 2, \ldots, n$ **do**
$W[i] \leftarrow \max\big(W[i-1],\ v_i + W[P[i]]\big)$

Recovering the solution

output $\leftarrow [\,]$
$i \leftarrow n$
while $i > 0$ **do**
// deciding if the ith interval contributed to $W[i]$ *or not*
if $W[i] = W[i-1]$ **then** *// skip i*
$i \leftarrow i - 1$
else *//* $W[i] = v_i + W[P[i]]$
add i to *output*
return *output*

2 Subset Sum

Given: $x_1, \ldots, x_n$ (positive integers) and a target T.

Output: The collection of numbers that sums to T, if it exists.

Example. Given $1, 2, 3, 4$ and $T = 5$: $2 + 3 = 5$.

First attempt (one parameter)

Let $\text{opt}(i) = \textsc{true}$ if we can sum to T from the first i numbers, otherwise FALSE.

$$\text{opt}(0) = \begin{cases} \textsc{true} & \text{if } T = 0, \\ \textsc{false} & \text{otherwise,} \end{cases} \qquad \text{opt}(i) = \begin{cases} \text{opt}(i-1) & \text{// if I don't include } x_i \\ \ ? & \text{// if I do} \end{cases}$$

Two parameters

Let $\text{opt}(i, j) = \textsc{true}$ if we can sum to j from the first i numbers, otherwise FALSE.

$$\text{opt}(0, 0) = \textsc{true},$$
$$\text{opt}(0, j > 0) = \textsc{false},$$

$$\text{opt}(i, j) = \begin{cases} \text{opt}(i-1, j) & \text{// if I don't include } x_i \\ \text{opt}(i-1, j - x_i) & \text{// if I do} \end{cases}$$

The answer is $\text{opt}(n, T)$.

Bottom-up

```
let W be an (n+1) × (T+1) table
W[0,0] ← TRUE
for j = 1, ..., T do
    W[0,j] ← FALSE
for i = 1, ..., n do
    for j = 0, ..., T do
        W[i,j] ← W[i−1,j] ∨ W[i−1,j−x_i]

if W[n,T] = FALSE then
    give up
else
    output ← [ ]
    i ← n
    j ← T
    while i > 0 do
        if W[i,j] = W[i−1,j] then                                   // skip i
            i ← i − 1
        else
            add i to output
            j ← j − x_i
            i ← i − 1
    return output
```

Running time: $\Theta(nT)$.