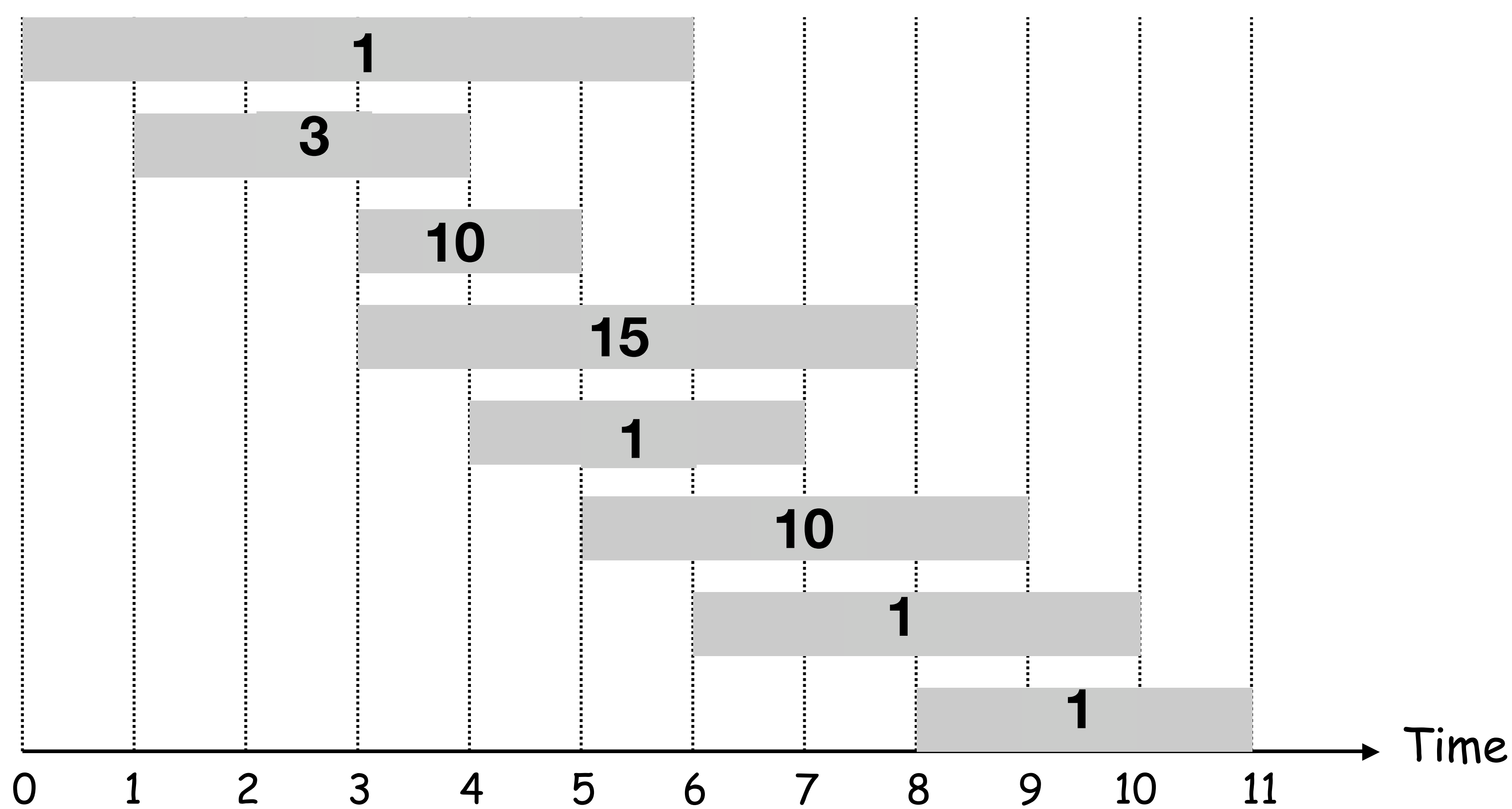
WEIGHTED INTERVAL SCHEDULING

The weighted interval scheduling problem:

- Job j starts at s_j, finishes at f_j, and has weight or value v_j
- Two jobs are *compatible* if they do not overlap
- Goal: find maximum value subset of mutually compatible jobs



Dynamic Programming

method for solving optimization problems:

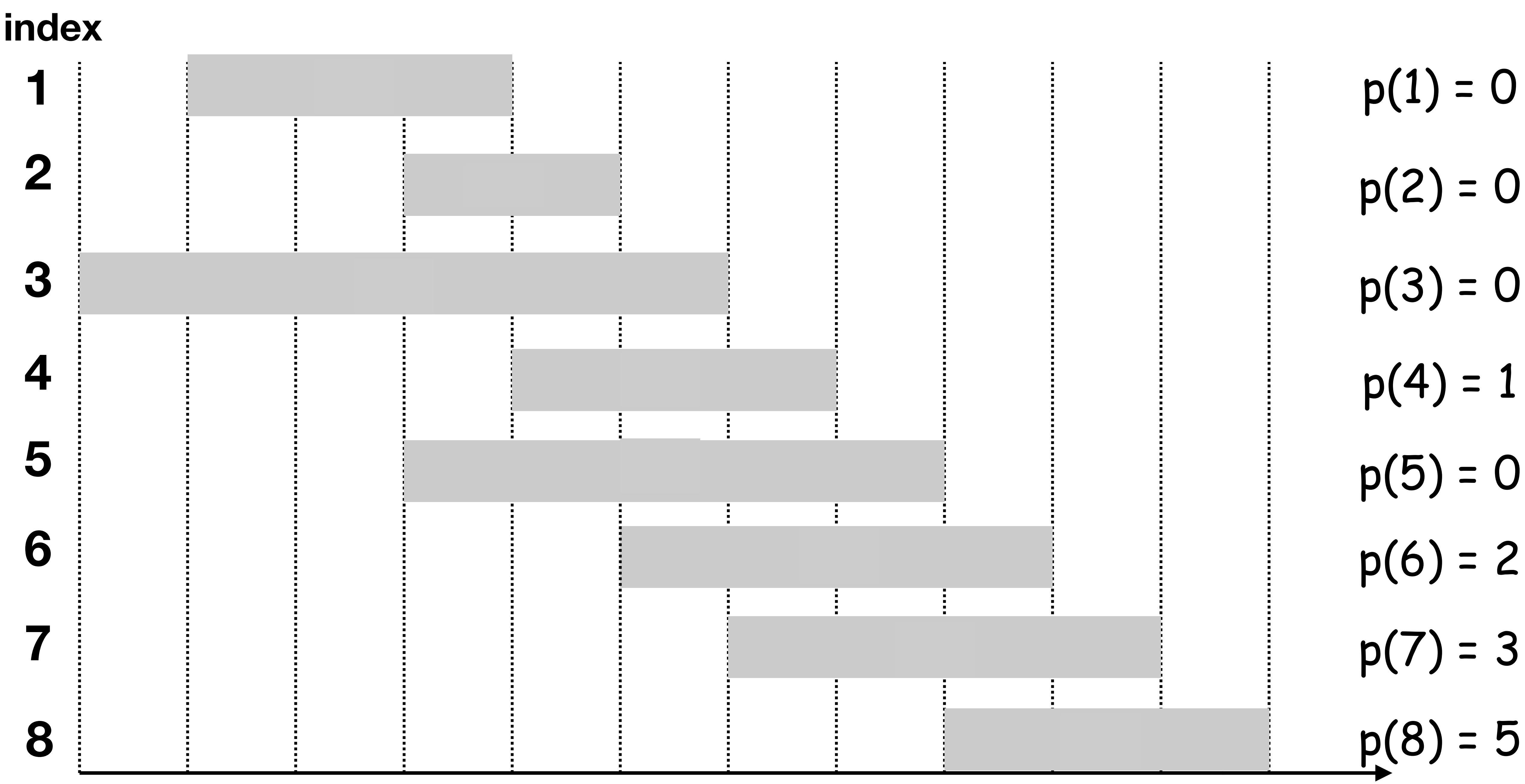
1. Use optimal substructure to find recurrence for OPT (the value of the optimal solution)

2. Use memoization or iteratively compute OPT

3. If need solution rather than just OPT, extract it from the table computed in step 2

Efficiently Generating Subproblem

Sort jobs by finish time, and let p(j) be the largest index i < j such that job i is compatible with job j.



(Note: Nothing special about finish time. Could have alternative version with start time)

Compute OPT: Iterative Algorithm

```
Input: n, s_1,…,s_n , f_1,…,f_n , v_1,…,v_n

Sort jobs by finish times so that f_1 ≤ f_2 ≤ ... ≤ f_n.

Compute p(1), p(2), …, p(n)

Iterative-Compute-Opt {
   M[0] = 0
   for j = 1 to n
      M[j] = max(v_j + M[p(j)], M[j-1])
}
```

Compute Solution

This algorithm has only computed the value of the optimal solution (i.e., the value of the optimal set of jobs). What if we want the solution itself (i.e., which jobs we should choose)?

```
Run Iterative-Compute-Opt(n)
Run Find-Solution(n)

Find-Solution(j) {
   if (j = 0)
      output nothing
   else if (v_j + M[p(j)] > M[j-1])
      print j
      Find-Solution(p(j))
   else
      Find-Solution(j-1)
}
```

Knapsack

```
Input: n, w_1,…,w_N, v_1,…,v_N

for w = 0 to W
   M[0, w] = 0

for i = 1 to n
   for w = 1 to W
      if (w_i > w)
         M[i, w] = M[i-1, w]
      else
         M[i, w] = max {M[i-1, w], v_i + M[i-1, w-w_i ]}

return M[n, W]
```