

CSC373F - Dynamic Programming and Computing an Optimal Solution

As indicated in class and in the text, it is usually somewhat more efficient (in dynamic programming based algorithms for optimization problems) to first compute the value of an optimal solution and then using that value to compute an optimal solution. Following the text we can use a simple recursive procedure to obtain a optimal solution (e.g. see page 134). Alternatively we can use an iterative algorithm (which is basically just unwinding the recursion) or we can compute an optimal solution (with some possible loss of efficiency) as we compute the value.

Here follows these alternatives for the interval scheduling problem. First the iterative algorithm for computing an optimal solution given that the array $M[]$ of optimal values has already been computed. Then a DP algorithm that computes an optimal solution $S[j]$ corresponding to each optimal value $M[j]$.

Note: We assume the $p(j) = 0$ if there does not exist an i such that $f_i \leq s_j$.

```
S := ∅; j := n
While j > 0
  If  $w_j + M[p(j)] \geq M[j - 1]$ 
    then  $S := S \cup \{j\}$ ;  $j := p(j)$ 
    else  $j := j - 1$ 
EndWhile

M[0] := 0
For j = 1, 2, ..., n
  M[j] := max( $w_j + M[p(j)]$ ,  $M[j - 1]$ ) ;
  If  $w_j + M[p(j)] \geq M[j - 1]$ 
    then  $S[j] := S[p(j)] \cup \{j\}$ 
    else  $S[j] := S[j - 1]$ 
EndFor
```