Dynamic Programming

August 23, 2025

Fibonacci number

- $F[0] = 0$ and $F[1] = 1$;
- $F[i] = F[i-1] + F[i-2]$

Algorithm to Compute Fibonacci number

Fib(n)

- If $n = 0$ Then Return 0
- If $n = 1$ Then Return 1
- Return Fib$(n-1)$+ Fib$(n-2)$

Algorithm to Compute Fibonacci number

Fib(n)

- If $n = 0$ Then Return 0
- If $n = 1$ Then Return 1
- Return Fib$(n-1)$+ Fib$(n-2)$

- $F[0] \leftarrow 0$;
- $F[1] \leftarrow 1$;
- For $i = 2$ to n
  $F[i] \leftarrow F[i-1] + F[i-2]$
- Return $F[n]$

Algorithm to Compute Fibonacci number

Fib(n)

- If $n = 0$ Then Return 0
- If $n = 1$ Then Return 1
- Return Fib$(n-1)$+ Fib$(n-2)$

- $F[0] \leftarrow 0$;
- $F[1] \leftarrow 1$;
- For $i = 2$ to n
  $F[i] \leftarrow F[i-1] + F[i-2]$
- Return $F[n]$

How to reuse the previous computation results?

Interval Scheduling

Input

- n tasks;
- a start time s_i for each task i;
- a finish time f_i for each task i;

Desired output

A set S of non-overlapping tasks maximizing $|S|$.

Weighted Interval Scheduling

Input

- n tasks;
- a start time s_i for each task i;
- a finish time f_i for each task i;
- a weight w_i for each task i

Desired output

A set S of non-overlapping tasks maximizing the total weight $\sum_{i \in S} w_i$

Earliest Finish Time Fails

Brute Force Algorithm

Enumerate(i, S, u)

- If $i = n$ and $u \geq ans$ Then
  $ans \leftarrow u$;
  Return
- If $i = n$ and $u \leq ans$ Then
  Return
- Enumerate($i + 1, S, u$) Do not put i-th job in S
- If i does not overlap any $j \in S$
  Enumerate($i + 1, S \cup \{i\}, u + w_i$) Put i-th job in S

Analysis of Brute Force Algorithm

Dynamic Programming

Dynamic Programming

- $A[0] = 0$
- For $t = 1$ to t_{max}
  - $A[t] = A[t-1]$
  - For all i with $f_i = t$
    - If $w_i + A[s_i - 1] \geq A[t]$ Then $A[t] \leftarrow w_i + A[s_i - 1]$
- Output $A[t_{max}]$

Analysis of Dynamic Programming Algorithm

Analysis of Dynamic Programming Algorithm

Improved Dynamic Programming

Thanks!