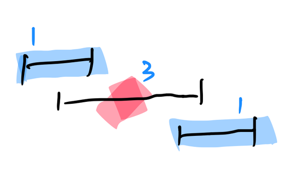
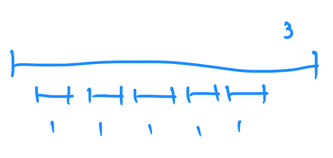
WEIGHTED INTERVAL SCHEDULING

WE ARE GIVEN A SET OF INTERVALS
$I = \{(s_1, f_1), (s_2, f_2), \dots, (s_n, f_n)\}$ (WHERE s_j IS THE STARTING TIME OF INTERVAL j, WHILE f_j IS ITS ENDING TIME, $\forall j$), AS WELL AS A WEIGHTING OF THE INTERVALS $\rightarrow$ THE WEIGHT OF INTERVAL IS $w_j \geq 0$.

GOAL: FIND A SUBSET $S \subseteq I$ THAT IS NON-OVERLAPPING, AND SUCH THAT $\sum_{(s_j, f_j) \in S} w_j$ IS MAXIMIZED.





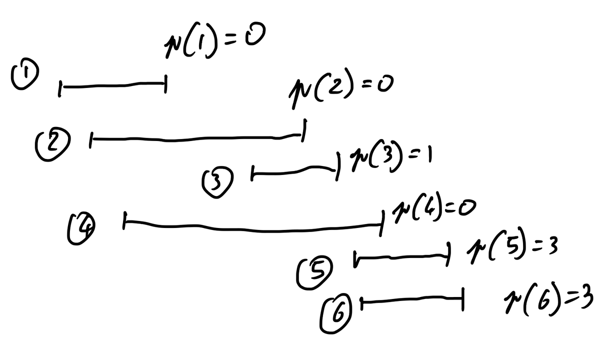
DYNAMIC PROGRAMMING

AN ALGORITHMIC TECHNIQUE THAT KEEPS TRACK OF ALL PARTIAL SOLUTIONS AT ONCE, IN AN EFFICIENT MANNER.
(GREEDY APPROACHES, ON THE OTHER HAND, KEEP TRACK OF A SINGLE PARTIAL SOLUTION AT ONCE).

FIRST OF ALL, LET US SORT THE INTERVALS BY FINISHING TIME: $f_1 \leq f_2 \leq f_3 \leq \dots \leq f_n$.

DEF: INTERVAL i "COMES BEFORE" INTERVAL j IFF $f_i \leq f_j$.

DEF: LET $p(j)$, FOR AN INTERVAL j, BE THE LARGEST INDEX $i < j$, S.T. INTERVALS i AND j ARE DISJOINT (COMPATIBLE); IF NO SUCH i EXISTS, LET $p(j) = 0$.



L: THE GENERIC INTERVAL i IS DISJOINT WITH EACH OF THE INTERVALS $1, 2, \dots, p(i)$, AND IS INCOMPATIBLE WITH INTERVAL $p(i)+1$.

P: EXERCISE.

SUPPOSE THAT O_i IS AN OPTIMAL SOLUTION TO THE PROBLEM RESTRICTED TO THE INTERVALS $\{(s_1, f_1), \dots, (s_i, f_i)\}$ (THE FIRST i INTERVALS).

LET OPT_i BE THE VALUE OF O_i:

$$OPT_i = \sum_{j \in O_i} w_j .$$

OBS: $\forall j \geq 1$:
(I) $w_j + OPT_{p(j)} \geq OPT_{j-1}$ IF AND ONLY IF THERE EXISTS AN OPTIMAL SOLUTION O_j SUCH THAT $j \in O_j$ (THE jTH INTERVAL IS IN O_j),
(II) $OPT_{j-1} \geq w_j + OPT_{p(j)}$ IF AND ONLY IF THERE EXISTS AN OPTIMAL SOLUTION O_j SUCH THAT $j \notin O_j$,
(III) $OPT_j = \max(w_j + OPT_{p(j)}, OPT_{j-1})$.

P: A SOLUTION O_{j-1} TO THE $(j-1)$-PROBLEM (THE PROBLEM WHOSE INPUT IS COMPOSED OF THE FIRST $j-1$ INTERVALS) ACTS AS A SOLUTION TO THE j-PROBLEM, AS WELL. THUS, $OPT_j \geq OPT_{j-1}$.

MOREOVER, A SOLUTION $O_{p(j)}$ TO THE $p(j)$-PROBLEM CAN BE TRANSFORMED INTO SOLUTION $O_{p(j)} \cup \{j\}$ FOR THE j-PROBLEM. THUS, $OPT_j \geq w_j + OPT_{p(j)}$.

THEN, $OPT_j \geq \max(OPT_{j-1}, w_j + OPT_{p(j)})$.

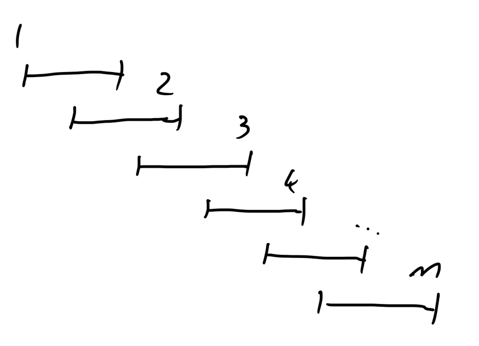
WE NOW PROVE THAT $OPT_j = \max(OPT_{j-1}, w_j + OPT_{p(j)})$.

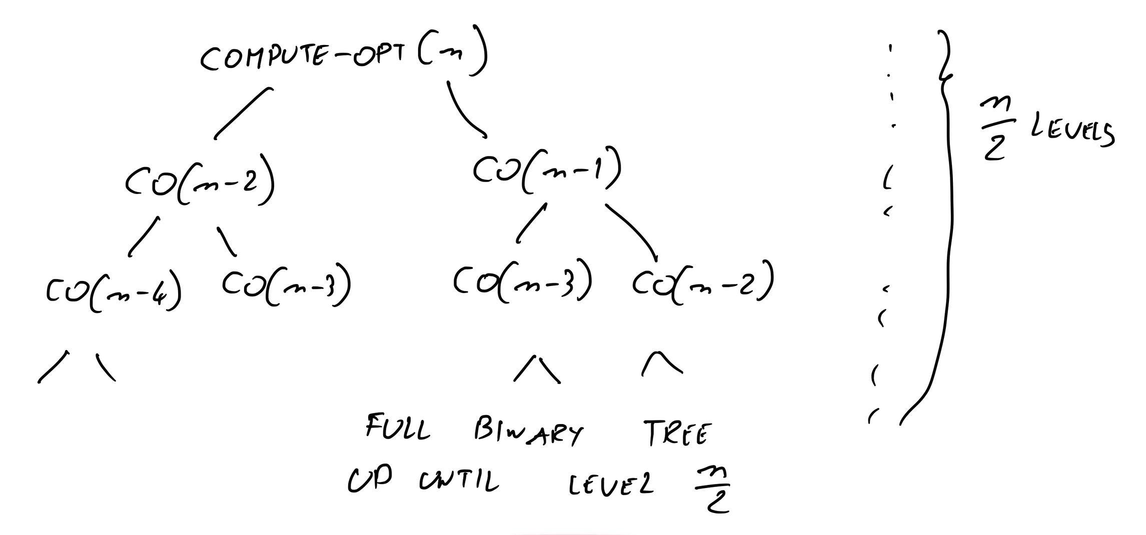
LET O_j BE AN OPTIMAL SOLUTION TO THE j-PROBLEM:
- IF $j \notin O_j$, THEN O_j IS ALSO AN OPTIMAL SOLUTION FOR THE $(j-1)$-PROBLEM. THUS, IF $j \notin O_j$, THEN $OPT_j = OPT_{j-1}$.
- IF $j \in O_j$, THEN $O_j - \{j\}$ IS AN OPTIMAL SOLUTION FOR THE $p(j)$-PROBLEM. THUS, IF $j \in O_j$, THEN $OPT_j = w_j + OPT_{p(j)}$.

THUS, (I), (II) AND (III) FOLLOW. □

```
DEF COMPUTE-OPT(j):
    IF j == 0:
        RETURN 0
    ELSE:
        RETURN max(w_j + COMPUTE-OPT(p(j)), COMPUTE-OPT(j-1))
```

CORRECT BUT EXPONENTIALLY SLOW!





TOT # OF CALLS $\geq 2^{n/2}$

"MEMOIZATION"

```
DEF M-COMPUTE-OPT(j):
    GLOBAL M              // M IS A DICTIONARY
    IF j IN M:
        RETURN M[j]
    ELSE:
        IF j == 0:
            M[j] = 0
        ELSE:
            M[j] = max(w_j + M-COMPUTE-OPT(p(j)), M-COMPUTE-OPT(j-1))
        RETURN M[j].
```

L: M-COMPUTE-OPT(j) RETURNS OPT_j

L: M-COMPUTE-OPT(j) RUNS FOR A TOTAL $O(j+1)$ TIME.

```
DEF WIS():
    OPT = [NONE] * (n+1)
    OPT[0] = 0
    FOR i = 1, ..., n
        OPT[i] = max(w_i + OPT[p[i]], OPT[i-1])
    RETURN OPT[n]
```

$\Theta(n^2)$ SPACE WORST CASE

```
DEF WIS():
    OPT = [NONE] * (n+1)
    O = [NONE] * (n+1)

    OPT[0] = 0
    O[0] = []

    FOR i = 1, ..., n
        IF w_i + OPT[p[i]] ≥ OPT[i-1]:
            OPT[i] = w_i + OPT[p[i]]
            O[i] = [i] + O[p[i]]
        ELSE:
            OPT[i] = OPT[i-1]
            O[i] = O[i-1]
    RETURN O[n]
```

RETURNS THE ACTUAL SOLUTION IN $O(n)$ TIME, $O(n)$ SPACE

```
DEF WIS():
    OPT = [NONE] * (n+1)
    OPT[0] = 0
    FOR i = 1, ..., n
        OPT[i] = max(w_i + OPT[p[i]], OPT[i-1])

    S = []
    i = n
    WHILE i ≥ 1:
        IF w_i + OPT[p[i]] > OPT[i-1]:
            S.APPEND(i)
            i = p[i]
        ELSE:
            i = i - 1
    RETURN S
```