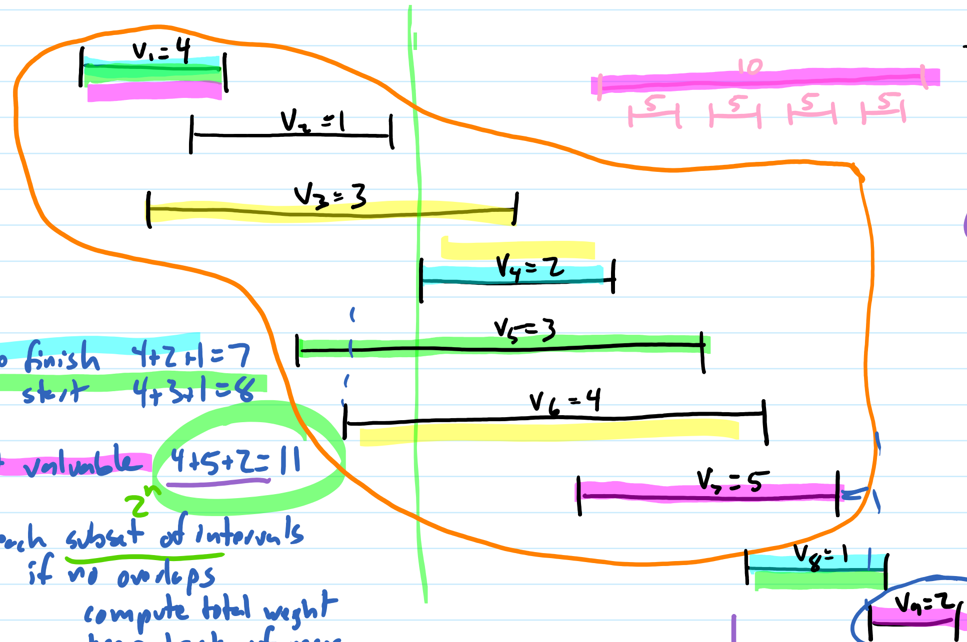
Weighted Interval Selection



1st to finish $4+2+1=7$

1st to start $4+3+1=8$

most valuable $4+5+2=11$

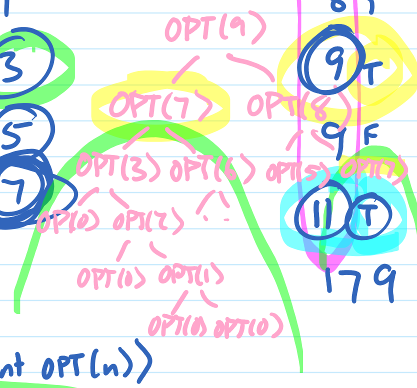
for each subset of intervals (2^n)
 if no overlaps
 compute total weight
 keep track of max

interval 2, interval 7 | interval 9

optimal soln for 1st 7 intervals

P(i) → index of last interval finishes before i starts		memo
		0
0	OPT(0)+4 / OPT(0)	4 T
0	OPT(0)+1 / OPT(1)	4 F
0	OPT(0)+3 / OPT(2)	4 F
2		6 T
1		7 T
1		8 T
3		9 T
5		9 F
7		11 T

179



(we want OPT(n))

OPT(i) = value of optimal selection using activities $\leq i$

$$OPT(i) = \begin{cases} 0 & \text{if } i = 0 \\ \max\left(OPT(P(i)) + v_i,\ OPT(i-1)\right) \end{cases}$$

optimal solution includes i

P(i): intervals finish before interval i starts

doesn't include i

```
COMPUTE-OPT-REC (n, v, P)      <- initialize memo
    return comp_OPT-HELP (n, v, P)
```

(n: # intervals, v: weights)

```
COMP-OPT-HELP (i, v, P, memo)          memoized
    if i = 0
        return 0
    else if i is in memo
        return memo[i]
    else
        memo[i] = max (C-O-H(P(i), memo) + v_i , C-O-H(i-1, v, P, memo))
        return memo[i]
```

$\Theta(1)$ work per function call

2 function calls per value of i

n possible values of i

$\Theta(n)$ work total

```
COMPUTE-OPT-ITER (n, v, P)
    memo <- (n+1) element array
    memo[0] <- 0
    for i=1 to n                                  Θ(n) iter
        memo[i] <- max( memo[P(i)] + v_i , memo[i-1] )   Θ(1)
```

base case

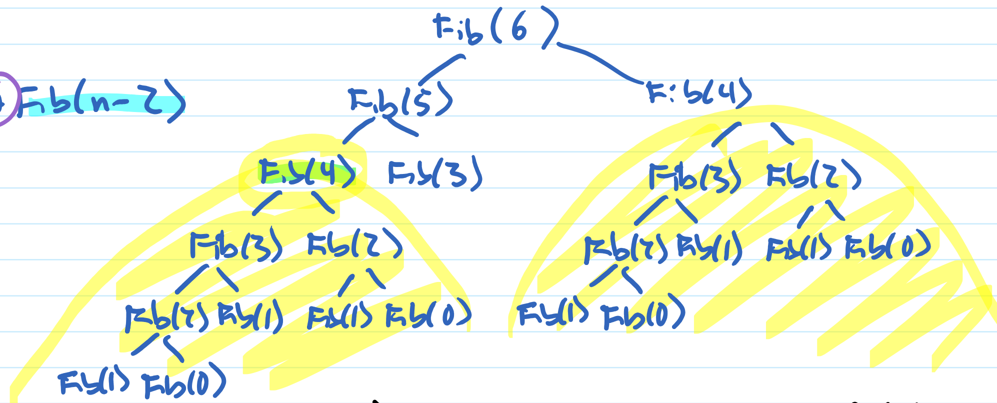
$\Theta(n)$ iter $\longrightarrow$ for $i = 1$ to n

memo$[i] \leftarrow \max(\text{memo}[P(i)] + v_i, \text{memo}[i-1])$ $\Theta(1)$

$\longrightarrow$ choice$[i] \leftarrow$ T if ~~↗ is max~~ / F otherwise $\quad \Theta(n)$ total

SELECT(OPT, j)

Fib(0) = 0
Fib(1) = 1
Fib(n) = Fib(n-1) + Fib(n-2)



T(n) = # additions to compute fib(n)

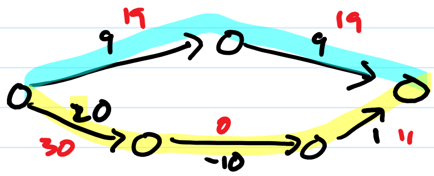
$T(0) = T(1) = 0$

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

$T(n) > Fib(n) = \dfrac{\left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{1-\sqrt{5}}{2}\right)^n}{2}$

Negative Weights

Negative weights



If $s, v_1, v_2, \ldots, v_k$ is a shortest path $s \leadsto v_k$ then

for each v_i, v_j $\quad v_i, v_{i+1}, \ldots, v_j$ is shortest path $v_i \leadsto v_j$