Fri, March 12 Lecture #20

Dynamic Programming

Suppose we have requests $R = \{r_1, \ldots, r_n\}$ with start s_i, finish f_i, value v_i. Assume these are sorted by finish time:

$$f_1 \le f_2 \le \ldots \le f_n$$

Given any set of requests S, define $\mathcal{O}(S)$ to be the score of the optimal solution using the requests in S. (We want $\mathcal{O}(R)$.)

$R_k = \{r_1, r_2, \ldots, r_k\}$.

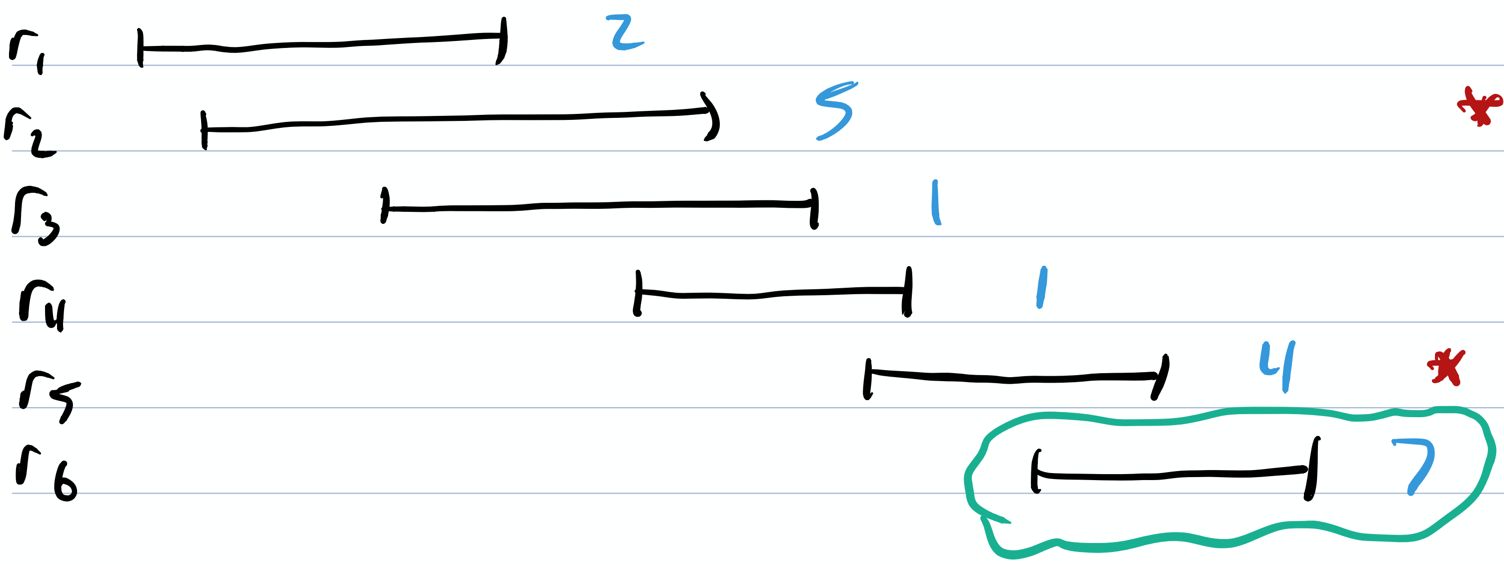
Question: If we know $\mathcal{O}(R_1), \mathcal{O}(R_2), \mathcal{O}(R_3), \ldots, \mathcal{O}(R_{\ell-1})$, can we use this to compute $\mathcal{O}(R_\ell)$?

If so:
- → $\mathcal{O}(R_1)$ is easy
- → Use $\mathcal{O}(R_1)$ to compute $\mathcal{O}(R_2)$.
- → Use $\mathcal{O}(R_1)$ and $\mathcal{O}(R_2)$ .... $\mathcal{O}(R_3)$

  ⋮

- → Use $\mathcal{O}(R_1), \ldots, \mathcal{O}(R_{n-1})$ to compute $\mathcal{O}(R_n)$

  $R_n = R$.



$O(R_6)$: r_6 is either in an optimal solution or it's not

If not: $O(R_6) = O(R_5)$

If it is: $O(R_6) = 7 + O(\cancel{R_5})$

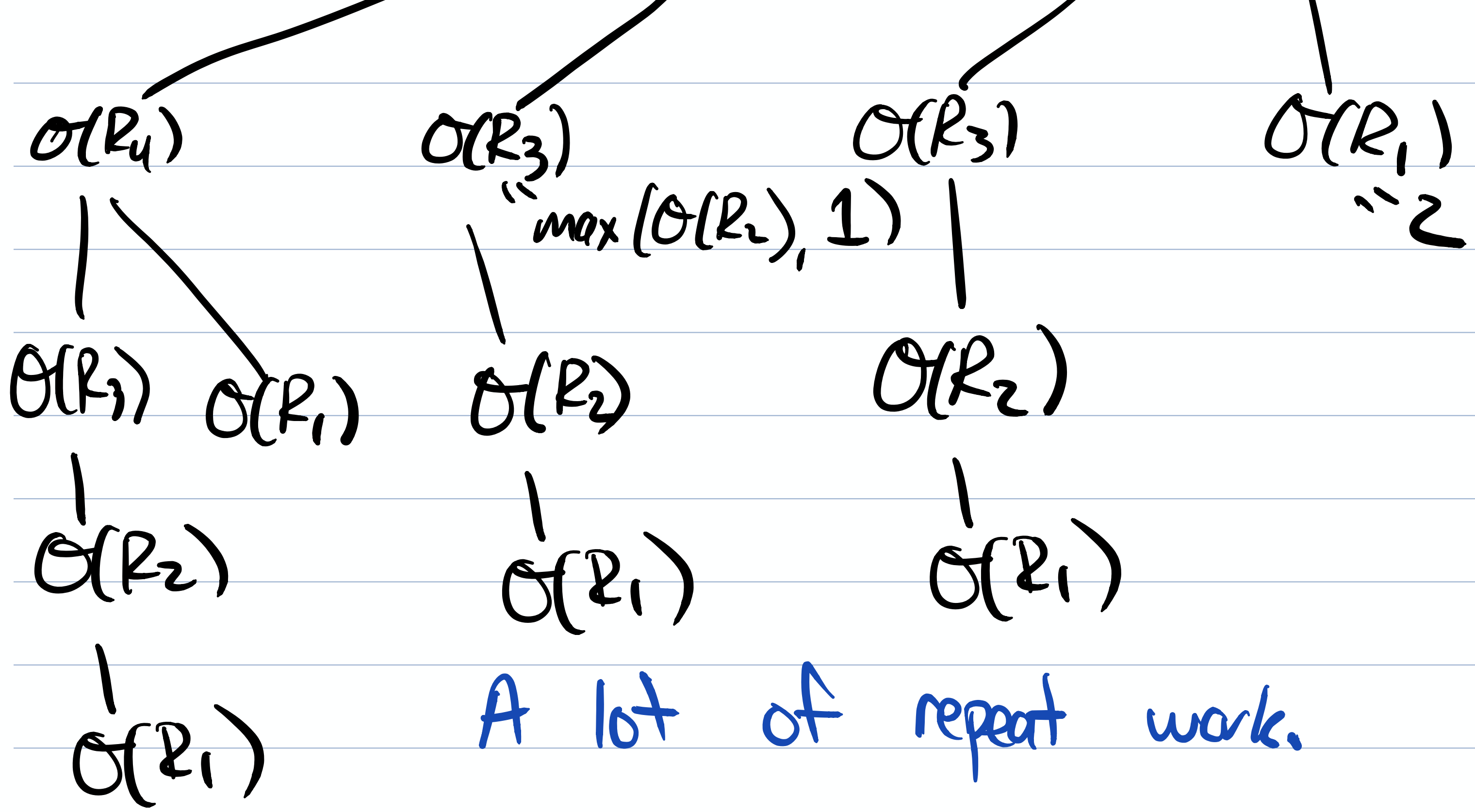
$$= 7 + O(R_4)$$

$$O(R_6) = \max\big(\underbrace{O(R_5)}_{\text{don't take } r_6},\ 7 + \underbrace{O(R_4)}_{\text{do take } r_6}\big)$$

Function call tree:

$$O(R_6) = \max(O(R_5),\ 7 + O(R_4))$$

$O(R_5) = \max(O(R_4), 4 + O(R_3))$ $\quad$ $O(R_4) = \max(O(R_3), 1 + O(R_1))$



A lot of repeat work.

More precise:

Let $p(S)$ be the intervals in S that don't conflict with the last element of S.

Ex: $p(R_6) = \{r_1, r_2, r_3, r_4\}$

$p(R_4) = \{r_1\}$

Then: $$O(R_k) = \begin{cases} 0, & \text{if } R_k = \{\} \\ \max\left(O(R_{k-1}),\ v_k + O(p(R_k))\right), & \text{otherwise} \end{cases}$$

Repeat work: just store a value the first time you compute it

Memoization

Pseudocode:

```
memo = empty dictionary          (global variable)
function wi(S):   list of requests, sorted by end time
    if S is a key in memo:
        return memo[S]
    if S = {}:
        memo[S] = 0
        return 0
    r = last request in S
    v = value of r
    score = max(wi(S-{r}), v + wi(p(S)))
    memo[S] = score
    return score
```

Linear time: $O(n)$

* This returns the score because

recursively keeping track of the sets would take ~~exponential~~ more time.

To get the solution itself, trace back through and build it up.

At the end of our example, the memo dict is:

$\{\}: 0$ $R_4: 5$
$R_1: 2$ $R_5: 9$
$R_2: 5$ $R_6: 12$
$R_3: 5$

$$wi(R_6) = \max(wi(R_5), 7 + wi(R_4))$$

$$12 = \max(9, \underbrace{7+5}_{\text{took } r_6 \text{ and called on } R_4})$$

r_6 ✓

r_5 X

$$wi(R_4) = \max(wi(R_3), 1 + wi(R_1))$$

$$5 = \max(\underbrace{5}_{\text{not taking } r_4}, 1+2)$$

r_4 X

$wi(R_3) \rightarrow$ don't take r_3 X

$wi(R_2) \rightarrow$ do take r_2 ✓

$\{r_2, r_6\}$ = optimal