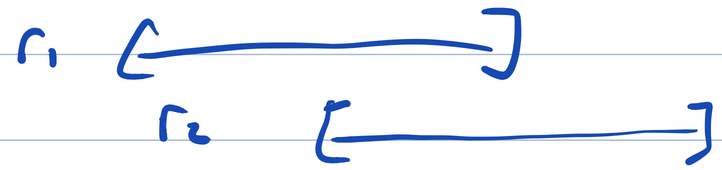
Friday, Feb 26 Lecture #15

Ex 3: Weighted Interval Scheduling

Requests $R = \{r_1, r_2, r_3, \dots\}$.

→ You either accept or reject each request. If you accept r_i, then in the future you can ignore all requests that conflict with it.



$\{r_1, r_2, \dots\}$

This is perfect for recursion.

$R = \{r_1, r_2, \dots, r_{10}\}$

solve($\{r_1, r_2, \dots, r_{10}\}$)

accept r_1:

R' = requests that don't conflict with r_1

return r_1 + solve(R')

reject r_1:

return solve($\{r_2, r_3, \dots, r_{10}\}$)

solve(R')

Pseudocode

```
function solve(requests):
    # goal: return best solution that can
            be made from [requests]
    if len(requests) = 0:
        return []
    new_request = requests[0]
    compatible = requests that don't conflict with
                                        new_request
    accept_solution = [new_request] + solve(compatible)
    reject_solution = solve(requests[1:])
    return whichever of accept_solution and
        reject_solution has the highest value
```

Topic 8 - Branch and Bound (B+B)

Our problems usually involve 2 considerations:

(1) Constraints that <u>must</u> be satisfied

<u>ex</u>: capacity of the knapsack

choosing requests that don't conflict

row/col/square conditions of Sudoku

(2) A value/score that we want to minimize/maximize.

Some problems are only about constraints.
Some problems don't really have constraints.
Ex: Minimum Spanning Tree

Backtracking use constraints to save time.
Never used score.

Branch + Bound is just backtracking with an extra way to rule out a partial solution.

* Assume maximizing from now on.

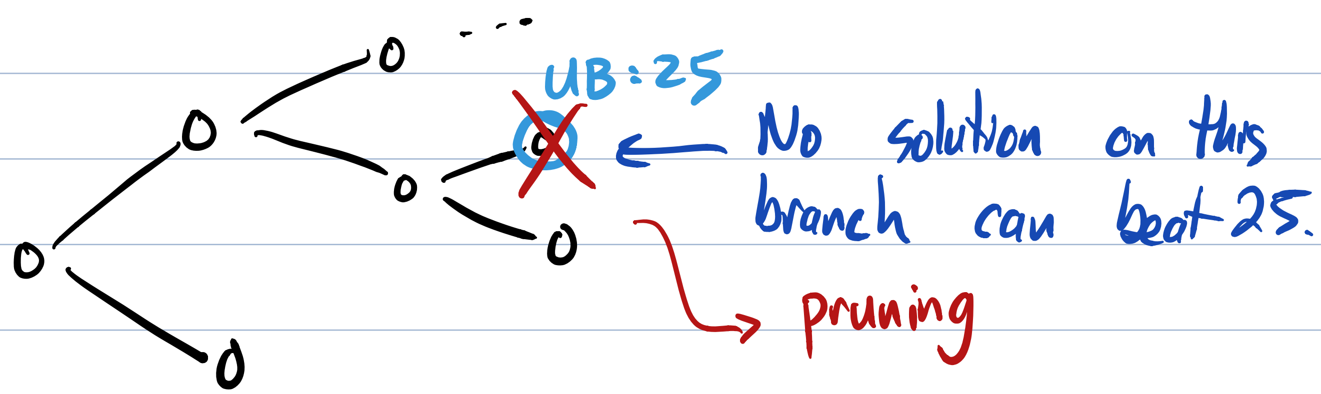
* If I've already seen a complete solution with a score of X, and I know for sure that there is no way to complete some partial solution that beats X, then abandon it (stop expanding).

There's no way to know exactly the best you can do to complete a partial solution.

Need: A way to get an upper bound on the best you could do when completing a partial solution.

"I don't know how good I can do, but I know for sure I can't do better than Y."

Have a sol. with a score of 30.



Hard part: How to compute these kinds of bounds.