Mon, Mar 4, 2024 ①

Scientific Computing

Announcements:

→ HW 3 due Fri, March 8

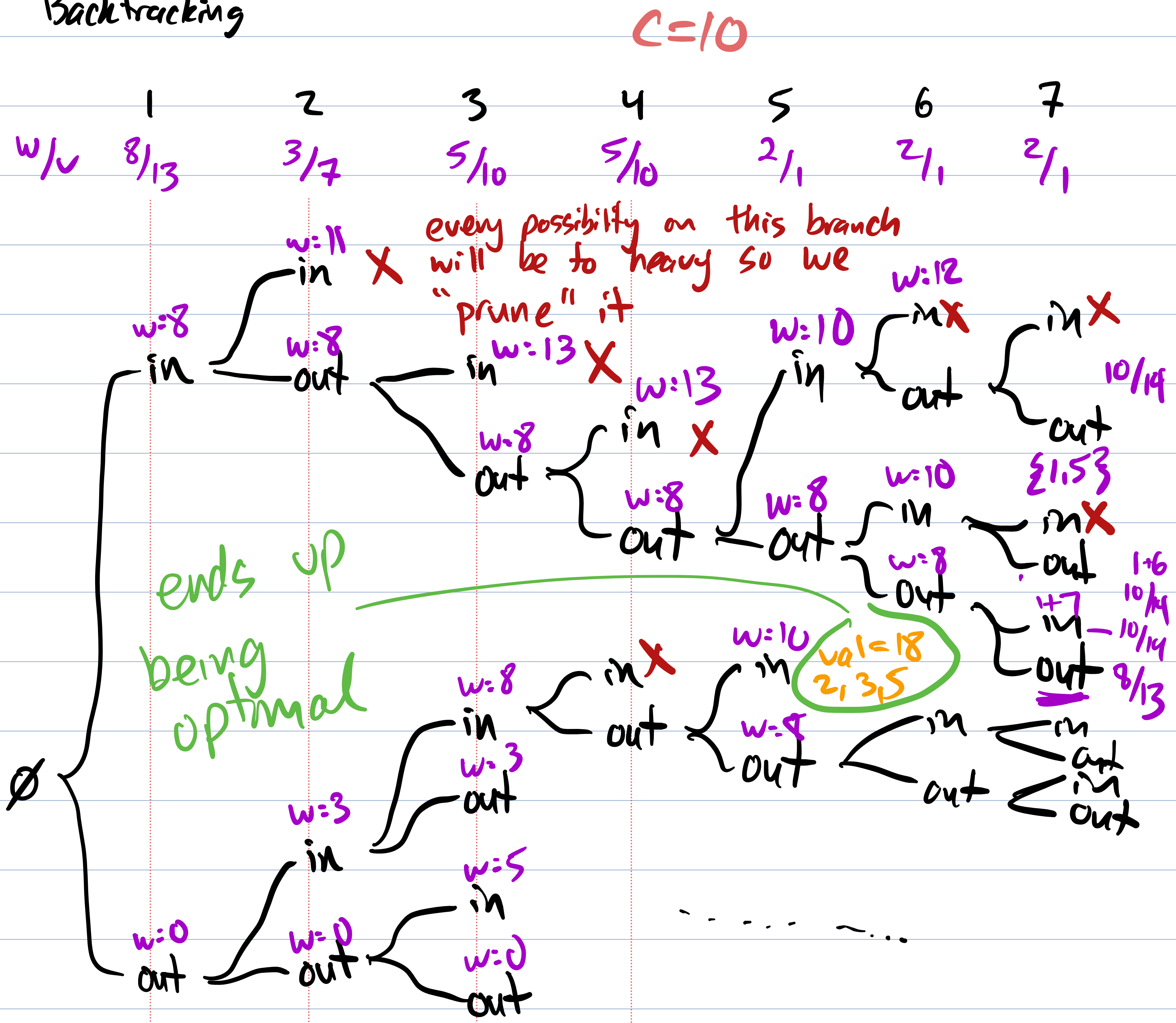
→ Wed March 6: In-class midterm

→ Format:

* Take the in-class part
* When done, take pictures of your answers, then turn in
* Keep Qs, and take take-home portion
* Fri: Office Hours in room during lecture
* Take-home due by the start of class on Wed after break.

Lecture 7 – Backtracking (continued)

Backtracking



Sudoku:

4	7	1	6	2	~~1~~ ~~2~~ 3	8	9 ~~6~~ ~~7~~ ~~8~~	5
6		8		5	4			
		5			8	7		4
8			4	3	2			
	3			1			4	
			9	8	7			1
1		3	8			4		
			3	4		5		9
				6	9		1	8

Backtracking

- Start filling in blank cells L-to-R then T-to-B.
- Start each cell at 1
- If that doesn't violate a rule, move to the next cell

- If it does violate, increase the value.
- If 1-9 are all bad, clear the cell, go back to the previous cell, and increase that one.

Ex: Weighted Interval Scheduling

Requests $R = \{r_1, r_2, \dots, r_n\}$

Every request has a start time s_i

finish time f_i

value v_i

Goal: To accept a set of requests with no conflicts that maximizes total value.

Build a solution bit-by-bit:

Look at each request one-by-one, accept or reject.

Once you accept a meeting you can then ignore all other meetings that conflict with it.

This set up is perfect for recursion because once we accept or reject a meeting we are left with solving two subproblems of the same form.

<u>Ex:</u> $R = \{r_1, \ldots, r_{10}\}$

$\text{solve}(\{r_1, \ldots, r_{10}\})$

accept r_1

reject r_1

R' = requests that don't conflict with r_1

return $\text{solve}(R')$

return $\text{solve}(\{r_2, \ldots, r_{10}\})$

recursion

Pseudocode

```
function solve(requests):
    #goal: return the subset of [requests]
           with no conflicts and highest total value

    if len(requests) = 0:
        return []
    new_request = requests[0]
    compatible = requests that do not 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
```

Ex: Job Scheduling Problem

	1	2	3	. . . . .
duration	-	-	-	- - -
deadline	-	-	-	- - -
profit	-	-	-	- - -

Search space: All ordered lists of a subset of the jobs.

n jobs, $j_1, j_2, \dots, j_n$

1   n

empty list

Search space $= \{ \varepsilon, [j_1], [j_2], \dots, [j_n],$

$[j_1, j_2], [j_2, j_1], [j_1, j_3], [j_3, j_1] \;\dots\dots\; [j_{n-1}, j_n], [j_n, j_{n-1}],$

$n \cdot (n-1)$

$\vdots$ 3 jobs: $n \cdot (n-1) \cdot (n-2)$

$[j_1, \dots, j_n]$ in any order $\}$

$n! = n(n-1)(n-2) \dots (3)(2)(1)$

$$1 + n + n(n-1) + n(n-1)(n-2) + \dots + n!$$

first second third ...

~~n - n!~~