CS3000

5/29 - thurs !!

Admin

- HW3 due tmw 9pm
- HW4 out, due 6/5 9pm
- Fun algo recitation today 1:30

Agenda

1. scheduling problem
2. DP approach → greedy
3. greedy correctness

0. LCS Review

* what if characters are repeated? Yes, it's ok!

	y_j	E	N	O	E	N	H
x_i	0	0	0	0	0	0	0
N	0	0	1	1	1	1	1
E	0	1	1	1	2	2	2
N	0	1	2	2	2	3	3
T	0	1	2	2	2	3	3
H	0	1	2	2	2	3	4
O	0	1	2	3	3	3	4

c table - value of optimal solution

LCS(nenthe, enoenh) = 4

actual - n, e, n, h

→ solution to subproblem LCS(nen, enoen) = 3 (n, e, n)

soln to problem is contained in solns to subproblems

↳ Optimal Substructure

```
LCS-LENGTH(X, Y, m, n)
1  let b[1 : m, 1 : n] and c[0 : m, 0 : n] be new tables
2  for i = 1 to m                          } base case
3      c[i, 0] = 0
4  for j = 0 to n
5      c[0, j] = 0
6  for i = 1 to m                          - X
7      for j = 1 to n                      - Y
8          if x_i == y_j                   } chars match, LCS+1
9              c[i, j] = c[i − 1, j − 1] + 1
10             b[i, j] = " ↖ "
11         elseif c[i − 1, j] ≥ c[i, j − 1]   } LCS stays same
12             c[i, j] = c[i − 1, j]
13             b[i, j] = " ↑ "
14         else
15             c[i, j] = c[i, j − 1]
16             b[i, j] = " ← "
17 return c and b
```

- b table helps to reconstruct actual optimal solution

space: $\Theta(mn)$

time: $\Theta(mn)$

* Brute Force: exponential!

- X is length m
- Y is length n

} $mn \approx$ quadratic!

1. Scheduling Problem

= can we do better? =

1. D+C
2. change data structure
3. Dynamic Programming
4. Greedy

} optimization problems

recursion → recursive imp? → DP faster? → Greedy faster? easier!

optimal substructure

- overlapping subproblems
- does it work?

Scheduling Problem!

↳ activities, shared resource, only one activity at a time

$S = \{a_1, a_2, \ldots, a_n\}$

- select activities that are compatible with each other
- optimize for # activities

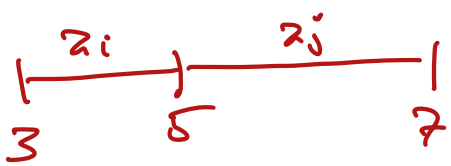
every a_i has:

- s_i start time
- f_i finish time

assume: a_i, a_j are compatible if $s_i \geq f_j$ or $s_j \geq f_i$

a_i has s_i 3, f_i 5

a_j has s_j 5, f_i 7



data stored in arrays s, f

(ex)

	1	2	3	4	5	6
s	2	1	3	3	6	12
f	3	6	6	8	10	14

→ position, activity #

S_{ij} = set of compatible activities that start after a_i finishes and ends before a_j starts

(ex) $S_{26} = \{a_5\}$

- start after a_2 (6)
- end before a_6 (12)

What is in these sets?

$S_{36} = \{2_5\}$ $S_{46} = \{\}$ $S_{13} = \{\}$ $S_{16} = \{2_3, 2_4, 2_5\}$

2_3 3→6
2_4 3→8 ✗ overlap
2_5 6→10

- S_{16} is *roughly* the whole solution!
- (sorted by finish times... how does that matter?)
- are $S_{36}, S_{46}, S_{13}, S_{16}$ optimal? yes!
- $S_{16} = \{2_3, 2_5\}$
  - ↳ remove 2_3 $S_{16} - \{2_3\} = \{2_6\}$
    ⇔ solution to S_{36}
  - $\{\}$
    ↳ solution to S_{13}

$S_{16} = S_{13} + S_{36} + \{2_3\}$ ⇝ optimal structure!
- soln to bigger problem is contained in soln to smaller subproblem

2. DP Approach to Greedy!

$S_{ij} = S_{ik} + S_{kj} + \{2_k\}$ ⇝ for any i, j cycle through all values of k to get smaller versions of the problem

↳ $S_{16} = S_{13} + S_{36} + \{2_3\}$

but also...

$= S_{12} + S_{26} + \{2_2\}$

$= S_{14} + S_{46} + \{2_4\}$

$= S_{15} + S_{56} + \{2_5\}$

Solution to S_{ij}
- try all values of k
- pick the max!

DP implementation:

```
for i = 1 to n
    for j = 1 to n
        - Sij uses all k's
```

$\Theta(n^3)$

=: can we do better? :=

Greedy Approach
- probably faster, easier
- is it correct??

} make locally best choice
- do not look at subproblems!

Often, start by sorting!
- locally best choice is just the next sorted thing

	1	2	3	4	5	6
s	2	1	3	3	6	12
f	3	6	6	8	10	14

ex ordered by finish time
- always select the next by finish time (compatible)

yuzi return a set of ints (activity #s)

$\Theta(n \lg n + n) = \Theta(n \lg n)$

```
SKED(s, f, n)
  sort s, f by finish time
  A = {1}
  k = 1
  for m = 2 to n
    // is m compatible with activities in A?
    if s[m] ≥ f[k]
        A = A ∪ {m}
        k = m
  return A
```

- choose next-earliest finish time
- add to set if compatible

→ leaves the most space open for more activities

3. Greedy Correctness

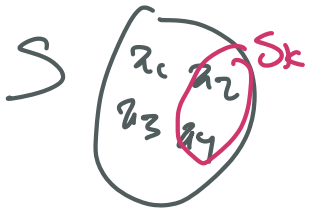
↳ does sorting by finish time work?

↳ optimal solution
- max # activities
- compatible

Approach ~ assume optimal soln does not contain locally optimal choice. then, add the locally optimal choice

Proof - scheduling - sort by finish time

- S is original problem (all activities)
- S_k is a subproblem
  - ↳ $a_m \in S_k$ has earliest finish time in S_k (would be locally optimal)



- Let A_k is optimal solution to S_k $\quad A_k \subseteq S_k$
  - ↳ activities in A_k are compatible
    $|A_k|$ is maximal for S_k

Case 1: $a_m \in A_k$? done!

Case 2: $a_m \notin A_k$? locally optimal choice not in optimal soln

↳ pick $a_j \in A_k$ with earliest finish time in A_k

by definition, $f_m \leq f_j$

swap a_j for a_m in solution $A_k' = A_k - \{a_j\} \cup \{a_m\}$

- a_m is compatible b/c a_j was compatible
- new finish time is earlier

$|A_k'| = |A_k|$ (cardinality same, still optimal solution

≈