CS3000: Algorithms & Data
Drew van der Poel

Recitation 4:
- Dynamic Programming
- Review Questions

May 31, 2021

DP Practice

Longest Increasing Subsequence (LIS): Given a sequence of numbers, find the length of the longest subsequence such that the elements are in increasing order.

Ex. Input: 10, 22, 9, 33, 21, 50, 41, 60, 80, 47
Sol. LIS has length 6 (10, 22, 33, 50, 60, 80)

Let OPT(j) be the length of the LIS ending at the j-th number *(0 <= j <= n)*

Using DP…

    * give a recurrence expressing the solution to each subproblem in terms of the solution to smaller subproblems

    *sketch pseudocode of your algorithm & give the runtime

    *describe how you would recover the LIS if asked

$$OPT(j) = \max_{1 \le i < j} \left(OPT(i)+1 \text{ if } A_i < A_j ,\ 1 \right)$$

$OPT(0) = 0$

$OPT(1) = 1$

ex. 10, 22, 9, 33, 21

$j=2$ (22)

$\max(2, 1) = 2$

$j=3$ (9)

$\max(-, -, 1) = 1$

$j=4$ (33)

$\max(2, 3, 2, 1) = 3$

M

0	1	2	1	3	2	
0	1	2	3	4	5	

$j=5$

$\max(2, -, 2, -, 1) = 2$

FindOPT(A)

$M \leftarrow$ Array of length $n+1$, $M[0] \leftarrow 0$,

$M[1] \leftarrow 1$

for $j = 2, \ldots, n$:

temp = 1

for $i = 1, \ldots, j-1$:

if $A_i < A_j$:

if $M[i] + 1 > temp$:

temp = $M[i] + 1$

$M[j] = temp$

$\boxed{O(n^2)}$

$M \leftarrow$ array of length $n+1$, $M[0] \leftarrow 0$, $M[1] \leftarrow 1$

```
FindOPT(n)
    if M[n] is not empty:
        return M[n]
    else
        temp = 1
        for i = 1, ..., n-1:
            if A_i < A_n:
                if FindOPT(i) + 1 > temp:
                    temp = FindOPT(i) + 1
        M[n] = temp
        return M[n]
```

Value of opt. soln: $\max_{1 \le j \le n} (M[j])$

(iv) let $j = \operatorname{argmax}_{1 \le i \le n} (M[i])$

- A_j is the last value in the LIS
- Find K s.t. $A_K < A_j$, $K < j$, $M[K] = M[j] - 1$
- Add A_K to front of LIS
- Repeat w/ $j = K$ until $M[K] = 1$

Return the LIS

You are given the following ordered set of three points P = { (1,4), (2,5), (3,9) }. The least squares line-of-best-fit for these points is y = 2.5x + 1 and it has error 1.5. Suppose we want to solve the segmented least squares problem on this set of points P, with segment cost C = 1.

10

In the next set of questions, you will fill in values of OPT(i), representing the total error of the segmented least squares solution for the first i points. Your responses should be single numeric values.

OPT(1) = 1 | 10

OPT(3) = 2 | 11.5

[a,b,c]

all 3 together

1.5 1.5

+1 10

2.5 11.5

[a] [b,c]

[a,b] [c]

0

0

1 10

+1 10

2 20

$$OPT(0) = 0$$

$$OPT(1) = C$$

$$OPT(2) = C$$

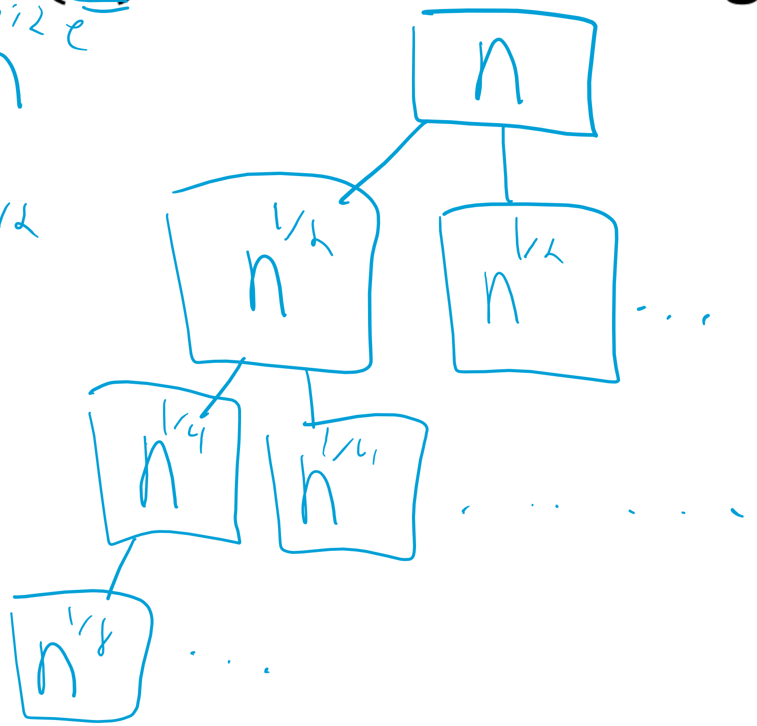
$$OPT(3) = \min_{1 \le i \le 3} \left(C + \varepsilon_{i,3} + OPT(i-1) \right)$$

$$i=1 \longrightarrow C + 1.5 + 0 = C + 1.5$$

$$i=2 \longrightarrow C + 0 + OPT(1) = 2C$$

$$i=3 \longrightarrow C + 0 + OPT(2) = 2C$$

- Consder the recurrence $T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n$ with $T(2) = 1$. Solve using a recursion tree.



	size	#	work @ level
0	n	1	n
1	$n^{1/2}$	$n^{1/2}$	$n^{1/2} n^{1/2} = n$
2	$n^{1/4}$	$n^{1/2} \cdot n^{1/4} = n^{3/4}$	$n^{3/4} n^{1/4} = n$
3	$n^{1/8}$	$n^{3/4} \cdot n^{1/8} = n^{7/8}$	$n^{7/8} n^{1/8} = n$
i	$n^{1/2^i}$	(# @ level $i-1$)($\sqrt{\text{piece size @ level } i-1}$)	

$$n^{1/2^i} = 2$$

$$\frac{1}{2^i} \log_2 n = 1$$

$$\log_2 n = 2^i \longrightarrow \log_2 \log_2 n = i$$

last level

$$\sum_{i=0}^{\text{last level}} \text{work @ level } i$$

$$\sum_{i=0}^{\log\log n} n = (\log\log n + 1)\, n$$

$$= O(n \log\log(n))$$