CS4800: Algorithms & Data
Jonathan Ullman

Lecture 8:
- Dynamic Programming: Lines of best fit, Knapsack

Feb 2, 2018

Logistics

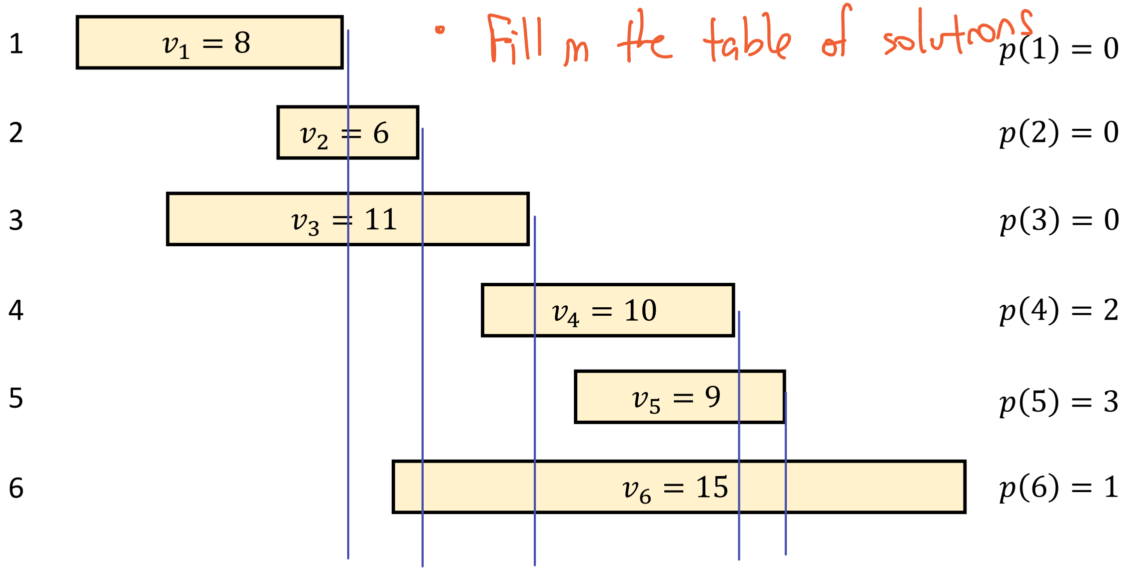
- Midterm I Feb 13^{th} (tuesday after next)
  - Divide-and-Conquer
  - Dynamic Programming

  - asymptotic notation
  - proofs by induction
  - recurrences / Master Theorem

  - One page of notes

Recap

- Define the subproblems
- Write the recurrence
- Fill in the table of solutions



- Find the opt. schedule by dynamic programming

$OPT = 23 \qquad S_{OPT} = \{1, 6\}$

Recap

Step 0: Sort the intervals by end time.

Step 1: Define the subproblems.

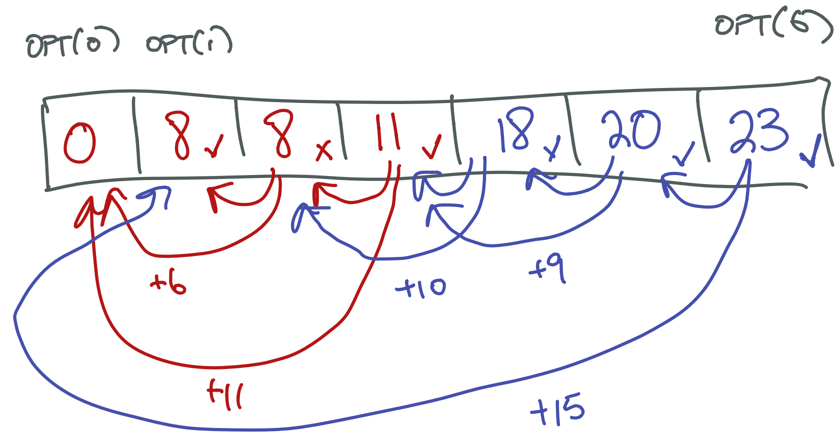
Step 2: Write recurrence $\boxed{OPT(i) = \max\{OPT(i-1),\ v_i + OPT(p(i))\}}$

Step 3: Build a table of solutions

1	$v_1 = 8$	$p(1) = 0$
2	$v_2 = 6$	$p(2) = 0$
3	$v_3 = 11$	$p(3) = 0$
4	$v_4 = 10$	$p(4) = 2$
5	$v_5 = 9$	$p(5) = 3$
6	$v_6 = 15$	$p(6) = 1$

Subproblems:

$OPT(i)$ = the value of the optimal sched for $\{1, \dots, i\}$ $\quad i = 0, 1, \dots, 6$



Today

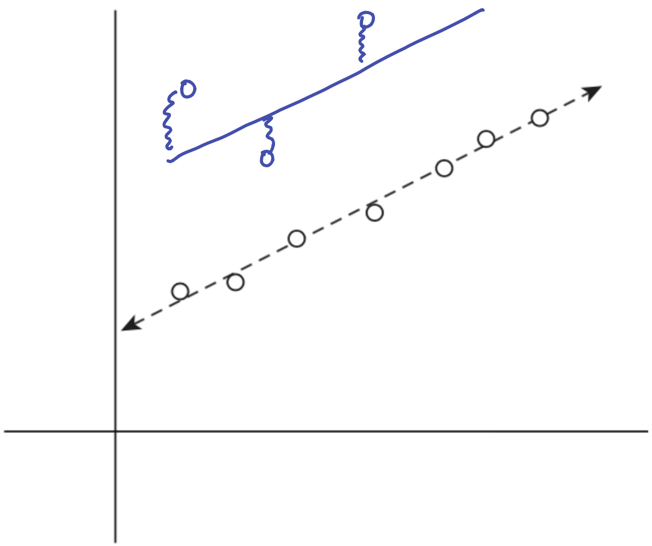
- Dynamic programming
  - More Practice: lines of best fit, knapsack
  - More Tricks: selecting a suffix, adding variables

Lines of best fit

Warmup: Line of Best Fit

- Input: n data points $P = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
- Output: the line L (i.e. $y = ax + b$) that fits "best"
  - "best" = minimizes $error(L, P) = \sum_i (y_i - ax_i - b)^2$

length of squiggly line for pt i.



Optimal Solution

$$a = \frac{n\sum x_i y_i - (\sum x_i)(\sum y_i)}{n\sum x_i^2 - (\sum x_i)^2}$$

$$b = \frac{\sum y_i - a\sum x_i}{n}$$

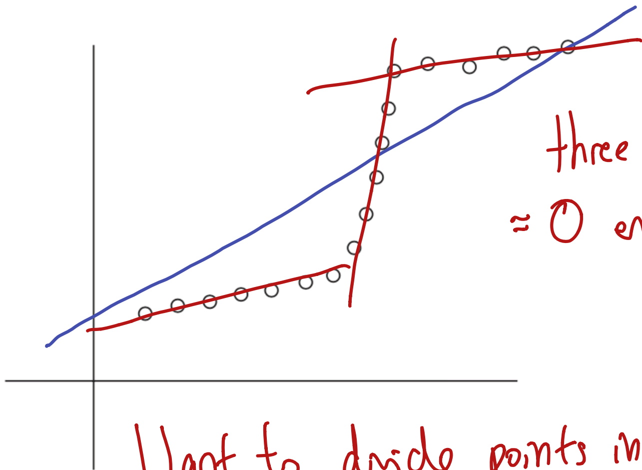
Can find optimal soln in $O(n)$ time

Lines of Best Fit

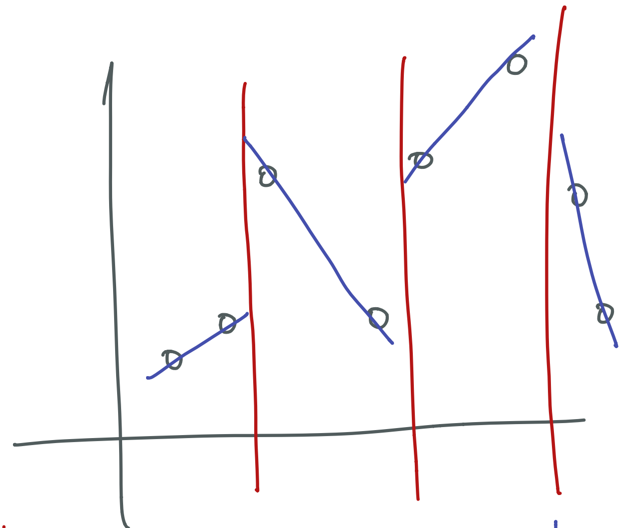
- Input: n data points $P = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
- What if the data does not look like a line?

one line has high error

three lines have ≈ 0 error



Want to divide points into intervals, find one line per interval.



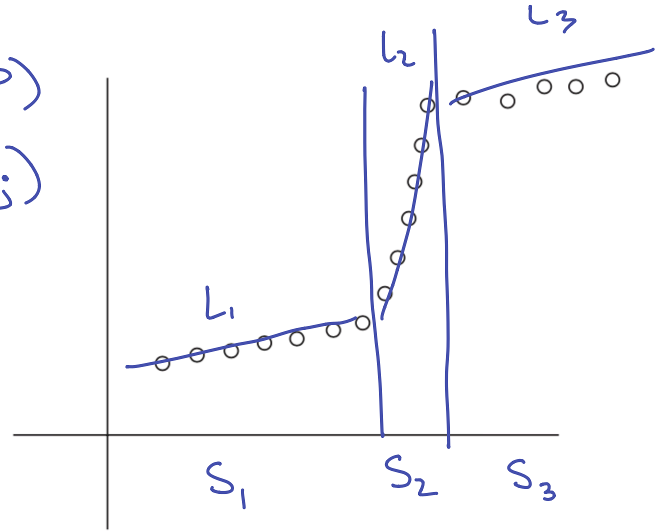
Problem does not make sense w/o some restriction on # of lines.

Lines of Best Fit

"cost of adding a new segment/line"

- Input: n data points $P = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, cost parameter $C > 0$
  - Assume $x_1 < x_2 < \cdots < x_n$; write $p_i = (x_i, y_i)$
- Output: a partition of P into contiguous segments $S_1, S_2, \ldots, S_m$, lines $L_1, L_2, \ldots, L_m$, minimizing "cost"

$$\mathrm{cost}(S_1, \ldots, S_m, L_1, \ldots, L_m, P) = Cm + \sum_{j=1}^{m} \mathrm{error}(L_j, S_j)$$



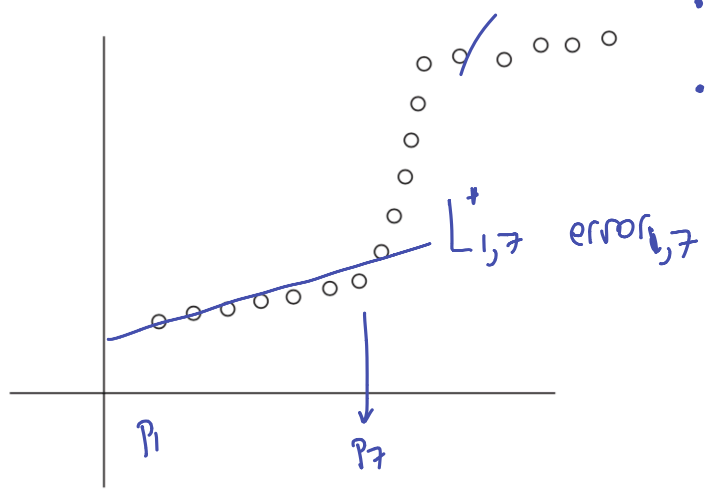
Lines of Best Fit

Our algorithm "only" needs to find the segments $S_1, \dots, S_m$

- First observation: for every segment S_j, L_j will be the (single) line of best fit for S_j
  - Let $L_{i,j}^*$ be the optimal line for $\{p_i, \dots, p_j\}$
  - Let $e_{i,j} = error(L_{i,j}^*, \{p_i, \dots, p_j\})$

For each i,j, can find $L^*_{i,j}, e_{i,j}$ in $O(n)$ time

- $O(n^3)$ time total
- Can improve to $O(n^2)$ time



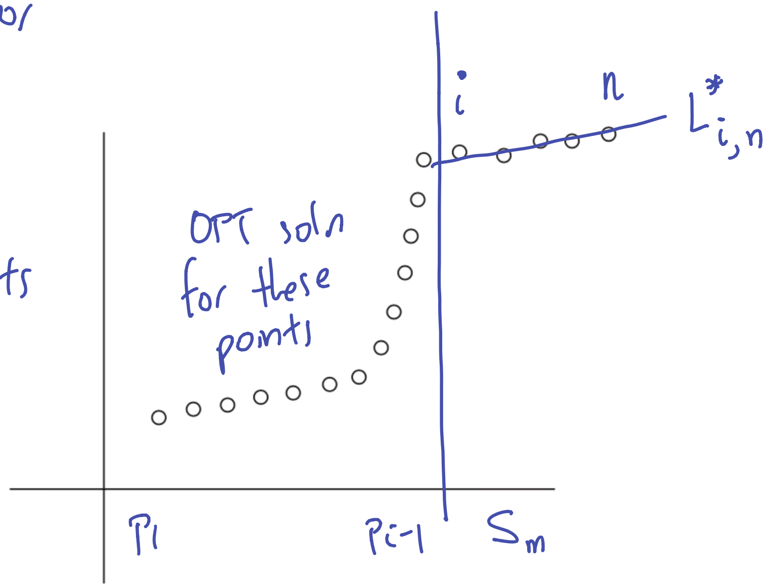
Lines of Best Fit

Let $L_{i,j}^*$ be the optimal line for $\{p_i, \ldots, p_j\}$
Let $e_{i,j} = error\left(L_{i,j}^*, \{p_i, \ldots, p_j\}\right)$

- Let O be the **optimal** solution
  - O is a set of segments $S_1, \ldots, S_m$
  - It has some final segment $S_m = \{p_i, \ldots, p_n\}$

- O must use $L_{i,n}^*$ for the last segment

- O must contain the optimal soln for points $\{p_1, \ldots, p_{i-1}\}$



cost of O $= e_{i,n} + C +$ cost of opt for $1, \ldots, i-1$

Lines of Best Fit

Let $L^*_{i,j}$ be the optimal line for $\{p_i, \ldots, p_j\}$

Let $e_{i,j} = error(L^*_{i,j}, \{p_i, \ldots, p_j\})$

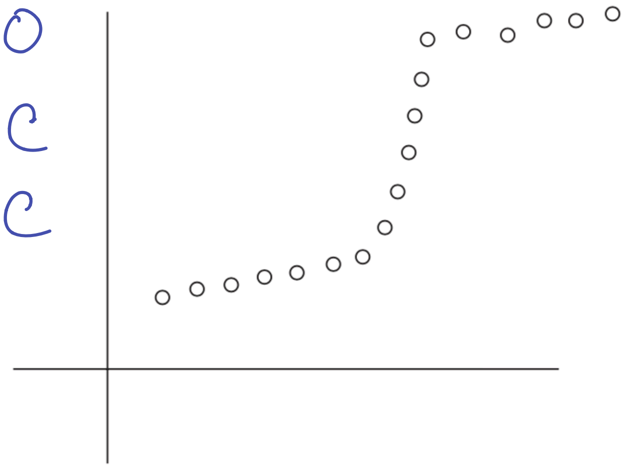
- Let O be the **optimal** solution
- Let $\cancel{OPT(i)}$ n $OPT(n)$ be the **cost** of the optimal solution for points $p_1, \ldots, \cancel{p_i}$ p_n

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

$OPT(0) = 0$

$OPT(1) = C$

$OPT(2) = C$



- Only $n+1$ subprobs ✓
- Recurrence for the subproblems ✓
- Can evaluate subproblems "in order" ✓

Lines of Best Fit

Let $L^*_{i,j}$ be the optimal line for $\{p_i, \dots, p_j\}$

Let $e_{i,j} = error\left(L^*_{i,j}, \{p_i, \dots, p_j\}\right)$

Let $P = \{p_1, \dots, p_n\}$ be the points

C is the cost

$\mathrm{LoBF}(n\cancel{, C})$:

  If $n = 0$: return 0   // Base Case

  Else: return $\min_{1 \le i \le n} e_{i,n} + C + \mathrm{LoBF}(i - 1)$

Lines of Best Fit

Let $L^*_{i,j}$ be the optimal line for $\{p_i, \dots, p_j\}$
Let $e_{i,j} = error\left(L^*_{i,j}, \{p_i, \dots, p_j\}\right)$

Let $P = \{p_1, \dots, p_n\}$ be the points   C is the cost
Let $M[1, \dots, n]$ be an array (initially empty)

$\mathrm{MLoBF}(i, \cancel{n})$:
  If $n = 0$: return 0
  Else If ($M[n]$ not empty): return $M[n]$
  Else:
    $M[n] \leftarrow \min_{1 \le i \le n} e_{i,n} + C + \mathrm{LoBF}(i-1)$
    return $M[n]$

- Every I make $\underline{\underline{n}}$ recursive calls I fill in $\underline{\underline{\text{one}}}$ subproblem
- There are n subproblems

$\Rightarrow$ Total # of calls is $\le n \times n = O(n^2)$ Total Time: $O(n^2)$

Lines of Best Fit

Let $L_{i,j}^*$ be the optimal line for $\{p_i, \dots, p_j\}$
Let $e_{i,j} = error(L_{i,j}^*, \{p_i, \dots, p_j\})$

- Let O be the **optimal** solution
- Let $OPT(i)$ be the **cost** of the optimal solution for points $p_1, \dots, p_i$

$$OPT(n) = \min_{1 \le i \le n} \{ e_{i,n} + C + OPT(i-1) \}$$

$$= \min \{ e_{1,n} + C + OPT(0), \text{-----}, e_{n,n} + C + OPT(n-1) \}$$

- How do we find the actual segments?
  - One of the n terms in the minimum is "best"
  - Call that i
  - Best final segment is $\{p_i, \dots, p_n\}$

Lines of Best Fit

Let $L^*_{i,j}$ be the optimal line for $\{p_i, \ldots, p_j\}$

Let $e_{i,j} = error\left(L^*_{i,j}, \{p_i, \ldots, p_j\}\right)$

Let $P = \{p_1, \ldots, p_n\}$ be the points

~~Let $M[1, \ldots, n]$ be an array (initially empty)~~

that minimizes $e_{i,n} + C + M[i-1]$

FindLoBF(n):

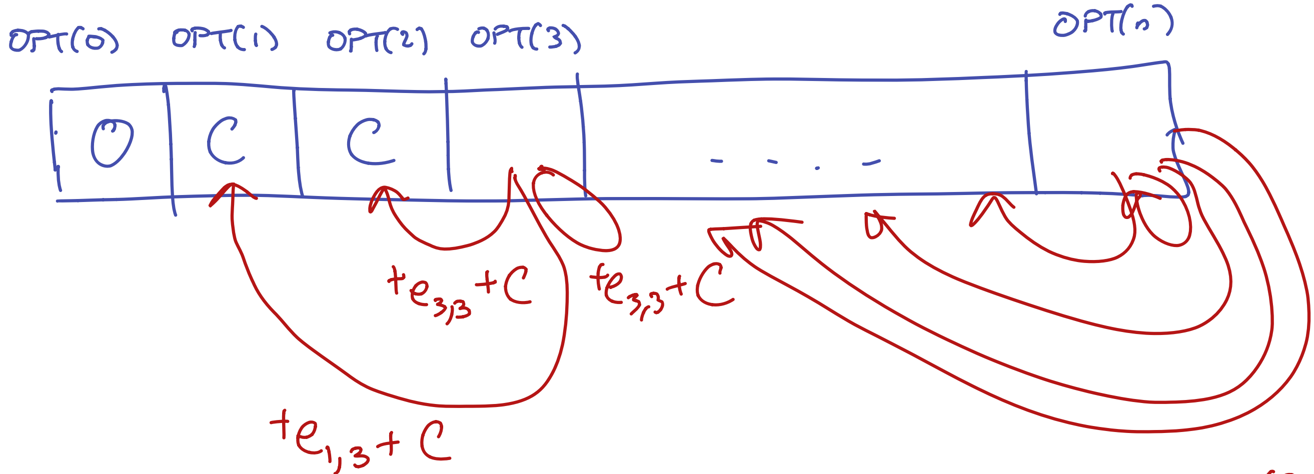
  find $i \in \{1, \ldots, n\}$ s.t $\cancel{M[n] = e_{i,n} + C + M[i-1]}$

  return $\left(\{p_i, \ldots, p_n\} + \text{FindLoBF}(i-1)\right)$

- $O(n^2)$ time to find the optimal set of segments once the table is filled.

Bottom Up Approach

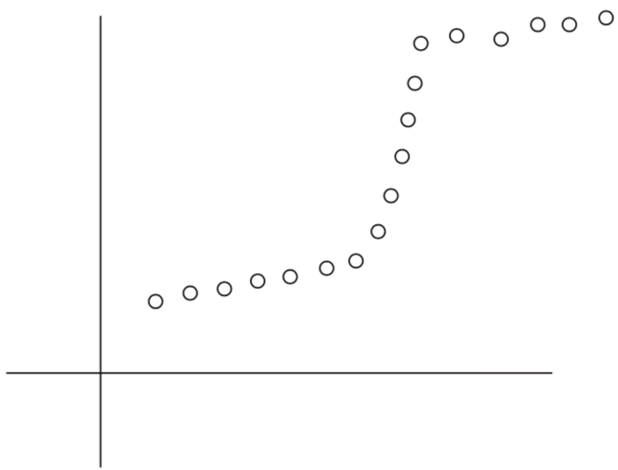
- $OPT(i)$ only depends on $OPT(0), \ldots, OPT(i-1)$



Arrow from i to j says OPT(i) depends on OPT(j)

Lines of Best Fit: Take II

- Input: n data points $P = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, a maximum number of segments k
  - Assume $x_1 < x_2 < \cdots < x_n$; write $p_i = (x_i, y_i)$
- Output: a partition of P into $\boxed{k}$ segments $S_1, \ldots, S_k$, lines $L_1, \ldots, L_k$, minimizing "cost"



Recap

penalty for adding new lines

- Can find the lines of best fit in time $O(n^2)$
  - Have to be careful about precomputing $e_{i,j}$

- New idea: find the best final segment
  - Compare to scheduling where we simply decided whether the final solution was in or out of the solution
  - Many problems have the flavor of splitting inputs into segments

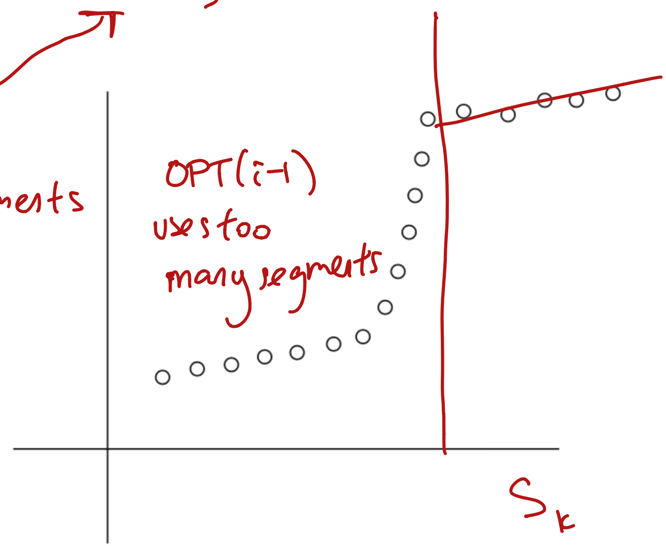
Lines of Best Fit: Take II

- Let O be the **optimal** solution (uses exactly k segments)
- Let $OPT(\overset{n}{\cancel{i}})$ be the **cost** of the optimal solution for points $p_1, \ldots, \cancel{p_i}$ p_n

$$OPT(n) = \min_{1 \le i \le n} \{ e_{i,n} + OPT(i-1) \}$$

- O contains some segments $S_1, \ldots, S_k$
- S_k is some segment $\{p_i, \ldots, p_n\}$

uses exactly k segments



best line for S_k

errors $e_{i,n}$

S_k

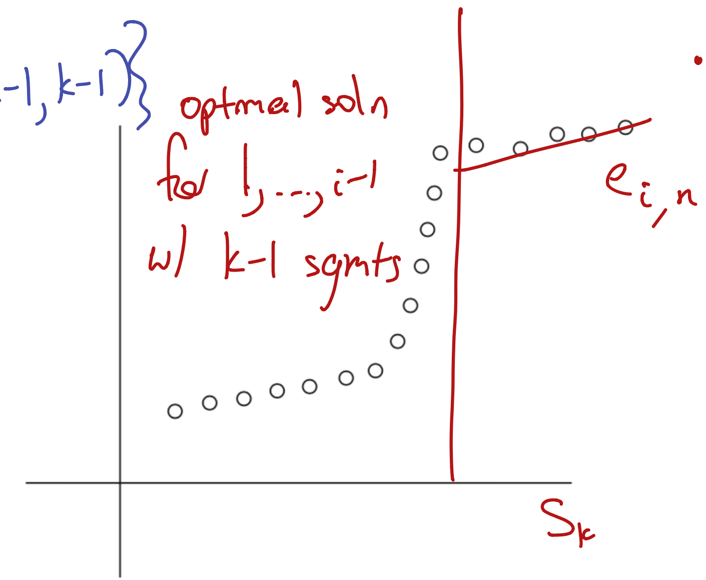
Key Idea: Adding Variables

(n+1)(k+1) subproblems

- Let O be the **optimal** solution
- Let $OPT(\cancel{i, s})$ ^{n,k} be the **cost** of the optimal solution for points $p_1, \dots, \cancel{p_i}$ p_n using $\cancel{s}$ k segments

$$OPT(n,k) = \min_{1 \le i \le n} \{ e_{i,n} + OPT(i-1, k-1) \}$$

- O uses $S_1, \dots, S_k$
- S_k is some $\{p_i, \dots, p_n\}$



Lines of Best Fit

Let $L^*_{i,j}$ be the optimal line for $\{p_i, \ldots, p_j\}$
Let $e_{i,j} = error(L^*_{i,j}, \{p_i, \ldots, p_j\})$

Let $P = \{p_1, \ldots, p_n\}$ be the points
Let $M[1:n, 1:k]$ be an array (initially empty)

$\mathrm{MLoBF}(n, k)$:
If $(n = 0)$: return 0; Else If $(n > 0, k = 0)$: return ∞
Else If ($M[n,k]$ not empty): return $M[n,k]$
Else:
n calls $\rightarrow$ $M[n,k] \leftarrow \min_{1 \le i \le n} e_{i,n} + \mathrm{LoBF}(i-1, k-1)$
fill in one entry $\rightarrow$ return $M[n,k]$

$\approx nk$ subproblems

$\approx n$ calls per problem

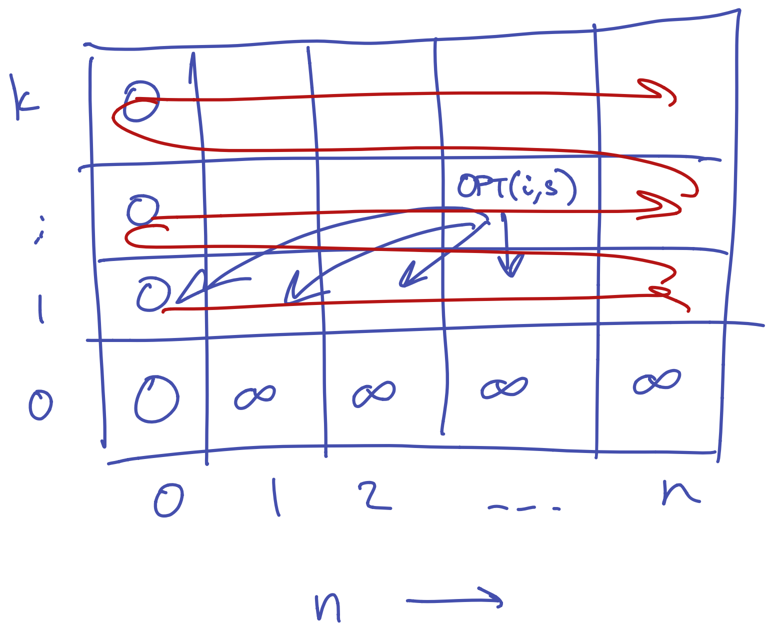
Total time is $O(n^2 k)$

Bottom Up Approach

$$OPT(i,s) = \min_{1 \le j \le n} \{e_{j,i} + OPT(j-1, s-1)\}$$

- $OPT(i, s)$ only depends on $OPT(j, s-1)$ for $j \le i$

$k =$



Recap

- Can find the k lines of best fit in time $O(n^2 k)$
  - Note: problem only makes sense for $1 \leq k < \frac{n}{2}$
- New idea: introduce a new variable
  - Use a larger set of subproblems
  - Gets easier with practice