Dynamic programming: log cutter, matrix chains, typesetting
What are the inputs and outputs of the FFT algorithm?

Describe the algorithm in a few sentences.

Do you remember any applications of the FFT?
how many ways are there to climb n steps if you take either 1 or 2 steps at a time??

\[ T(n) = \# \text{ways to climb n steps} \]

\[ T(n) = T(n-1) + T(n-2) \]

Recurrence:
Fibonacci:

\[
\begin{array}{cccc}
2 & 1 & 1 & 1 \\
2 & 2 & 1 & 1 \\
1 & 2 & 1 & 1 \\
1 & 2 & 1 & 1 \\
\end{array}
\]

\[ \sim \phi^n / \phi \]

golden ratio
Stairs(n)
if n<=1 return 1
return Stairs(n-1) + Stairs(n-2)

$S(n) \sim S(n-1) + S(n-2) \rightarrow \text{Fibonacci number} \sim O(\phi^n)$
Stairs(n) if n<=1 return 1
ret Stairs(n-1) + Stairs(n-2)

Stairs(5)

Stairs(4) Stairs(3)
Stairs(n) if n<=1 return 1
ret Stairs(n-1) + Stairs(n-2)

Stairs(5)

Stairs(4) Stairs(3)

Stairs(3) Stairs(2) Stairs(2) Stairs(1)
Stairs(n) if n<=1 return 1
ret Stairs(n-1) + Stairs(n-2)

Stairs(5)

Stairs(4)  Stairs(3)

Stairs(3)  Stairs(2)  Stairs(2)  Stairs(1)

Stairs(2)  Stairs(1)  Stairs(1)  Stairs(0)  Stairs(1)  Stairs(0)
initialize memory M

Stairs(n)

Base case as before

if M contains n, return M[n]
else
    answer = Stairs(n-1) + Stairs(n-2)
    M[n] = answer

return answer
initialize memory M

Stairs(n)
    if n<=1 then return n
    if n is in M, return M[n]
    answer = Stairs(i-1)+ Stairs(i-2)
    M[n] = answer
    return answer
Stairs(n)

```cpp
if n <= 1 then return 1
if n is in M, return M[n]
answer = Stairs(i-1) + Stairs(i-2)
M[n] = answer
return answer
```
Stairs(n)

\[
\begin{align*}
\text{stair}[0] &= 1 \\
\text{stair}[1] &= 1 \\
\text{for } (i = 2, \text{ to } n) & \\
\text{Stairs}[i] &= \text{Stairs}[i-1] + \text{Stairs}[i-2] \\
\text{return Stairs}[n]
\end{align*}
\]
Stairs(n)

    stair[0]=1
    stair[1]=1
    for i=2 to n
        stair[i] = stair[i-1]+stair[i-2]
    return stair[i]
Dynamic Programming
two big ideas

1) recursive substructure.

\[ T(n) = T(n-1) + T(n-2) \]

2) memoization

Keep track of intermediate results, solve the intermediate problems in a specific order to maximize efficiency.
two big ideas

recursive structure + memoizing
Spot price for lumber

\[
P_1 \quad p_2 \quad p_3 \quad p_4 \quad \ldots \quad p_8
\]

\[P_i \rightarrow \text{spot price for an } i\text{-" wide slab of lumber}\]

\[
\begin{align*}
\text{if } & n=5 \quad \frac{p_1}{p_2} \quad \frac{p_4}{p_3} \\
\text{if } & n=200\text{"}
\end{align*}
\]
Log cutter dilemma

input to the problem: \( n, (p_1, \ldots, p_n) \)

\( n \) wide log \( \Rightarrow \) spot prices for slabs of width \( i \)

goal: \[ \max \text{ profits} \]

find a set of cuts \( i_1, i_2, i_3, \ldots, i_k \)

\[ \forall j \neq 0 \quad i_j < n \]

\[ \max \sum_{j=0}^{k} p_{i_j} \]
Observation

\[ \text{best move in the optimal solution} \]

\[ \text{Best}_n = P_i + \text{Best}_{n-1} \]

\[ n - i_k \]

\[ i_k \]
Solution equation

\[ B_n = \frac{P_i + B_{n-i}}{n} \]

\[ B_n = \max \{ P_i + B_{n-i} \} \quad \text{for } i = 1^n \]

\[ \text{Best}_{200} = \max \left\{ \frac{1 + B_{199}}{2}, \frac{P_2 + B_{198}}{3}, \frac{P_3 + B_{197}}{4}, \ldots, \frac{P_{200} + B_0}{200} \right\} \]
Approach

Start here
Approach

\[ B(i) = p_i + B_0 \]

\[ B_2 = \max \begin{cases} p_2 + B_0, \\ p_1 + B_1 \end{cases} \]

\[ B_3 = \max \begin{cases} p_3 + B_0, \\ p_2 + B_1, \\ p_1 + B_2 \end{cases} \]
BestLogs\( (n, (p_1, \ldots, p_n)) \)

if \( n \leq 0 \) return 0

\[ \text{for } i = 1 \text{ to } n \]

\[
B[i] = \max_{j=1}^{i} \{ p_j + B[i-j] \}
\]

\[
B[i] = -\infty \quad \text{for } j=1 \text{ to } i
\]

\[
t = p_j + B[i-j]
\]

if \( t > B[i] \)

\[
B[i] = t
\]

Running time: \( 1 + 2 + 3 + \ldots + (n-1) \sim \Theta(n^2) \)
BestLogs\( (n, (p_1, \ldots, p_n)) \)

\[
\begin{align*}
\text{if } n \leq 0 & \text{ return } 0 \\
\text{for } i = 1 \text{ to } n & \\
\quad \text{Best}[i] = \max_{k = 1 \ldots i} \{ p_k + \text{Best}[i - k] \} \\
\quad \\text{Choice}[i] = k^* \\
\text{return } \text{Best}[n] 
\end{align*}
\]

(Work on an example)
The actual cuts?
BestLogs($n, (p_1, \ldots, p_n)$)

if $n \leq 0$ return 0
for $i = 1$ to $n$
    $\text{Best}[i] = \max_{k = 1 \ldots i} \{ p_k + \text{Best}[i - k] \}$

return $\text{Best}[n]"
Matrix
\[ c_1 = r_2 \]

\[ A_1 \]

\[ A_2 \]

\[ B \]

\[ c_1 \cdot r_1 \cdot c_2 = \# \text{ of operations} \]
$R_1 \hspace{2cm} \hspace{1cm} R_2 \hspace{2cm} A_1 \hspace{2cm} A_2 \hspace{1cm} C_1 \hspace{2cm} C_2 \hspace{1cm} B \hspace{1cm} \hspace{1cm} = \hspace{1cm} \hspace{1cm}$
\[(A_1 \cdot A_2) \cdot A_3\]
$A_1 \cdot A_2 \cdot A_3$

$(A_1 \cdot A_2) \cdot A_3 \quad A_1 \cdot (A_2 \cdot A_3)$
\((A_1 \cdot A_2) \cdot A_3\)

\[
\begin{align*}
10 \cdot 100 \cdot 5 + 10 \cdot 5 \cdot 50 \\
\text{operations}
\end{align*}
\]
\[ A_1 \cdot (A_2 \cdot A_3) \]

\[ 100 \cdot 5 \cdot 50 = 25000 \text{ N} \]

\[ \Rightarrow 75\text{ kN}! \]
\[ A_1 \cdot A_2 \cdot A_3 \]

100 \cdot 5 \cdot 50 + 10 \cdot 100 \cdot 50

operations
order matters
(for efficiency)
how many ways to compute?

\[ A_1 A_2 A_3 \ldots A_n \]
how many ways to compute?

\[ A_1 A_2 A_3 \ldots A_n \]
how do we solve it?

identify smaller instances of the problem
device method to combine solutions
small # of different subproblems
  solved them in the right order
optimal way to compute $A_1 A_2 A_3 A_4 \ldots A_n$
optimal way to compute

\[ A_1 A_2 A_3 A_4 \ldots A_n \]

\[ B_{1,n} = B_{1,\ell} + B_{\ell+1,n} + r_1 c_\ell c_n \]
optimal way to compute

\[A_1 A_2 A_3 A_4 \ldots A_n\]

B[1,n]
optimal way to compute

\[
\begin{array}{cccc}
A_1 & A_2 & A_3 & A_4 \ldots A_n \\
R_1C_1C_n & R_1C_2C_n & \ldots & R_1C_{n-2}C_n & R_1C_{n-1}C_n \\
\end{array}
\]
\[ B(i, i) = 1 \]

\[ B(1, n) = \min \]
\( B(i, i) = 1 \)

\[
B(1, n) = \min \begin{cases} 
B(1, 1) + B(2, n) + r_1 c_1 c_n \\
B(1, 2) + B(3, n) + r_1 c_2 c_n \\
\vdots \\
B(1, n - 1) + B(n, n) + r_1 c_{n-1} c_n 
\end{cases}
\]
\[ B(i, j) = \begin{cases} 
0 & \text{if } i = j \\
\min_k \{B(i, k) + B(k + 1, j) + r_i c_k c_j \} & \text{otherwise}
\end{cases} \]
how did we solve it?

identified smaller instances of the problem
devised method to combine solutions
small # of different subproblems
solved them in the right order
\[ B(i, j) = \begin{cases} 
0 & \text{if } i = j \\
\min_k \{ B(i, k) + B(k + 1, j) + r_i c_k c_j \} & \text{otherwise}
\end{cases} \]

which order to solve?
\( B(1, 2) = \)
\[ B(i, j) = \begin{cases} 
0 & \text{if } i = j \\
\min_k \{ B(i, k) + B(k + 1, j) + r_i c_k c_j \} & \text{otherwise} 
\end{cases} \]
\[
B(i, j) = \begin{cases} 
0 & \text{if } i = j \\
\min_k \{B(i, k) + B(k + 1, j) + r_i c_k c_j\} & \text{otherwise}
\end{cases}
\]
$$B(i, j) = \begin{cases} 0 & \text{if } i = j \\ \min_k \{B(i, k) + B(k + 1, j) + r_i c_k c_j \} & \text{otherwise} \end{cases}$$
\[ C(1, 6) = \]
\[
C(1, 6) = \min \left\{ \begin{array}{l}
k = 1 \quad C(1, 1) + C(2, 6) + r_{1c_1c_6} \\
k = 2 \quad C(1, 2) + C(3, 6) + r_{1c_2c_6} \\
k = 3 \quad C(1, 3) + C(4, 6) + r_{1c_3c_6} \\
k = 4 \quad C(1, 4) + C(5, 6) + r_{1c_4c_6} \\
k = 5 \quad C(1, 5) + C(6, 6) + r_{1c_5c_6}
\end{array} \right. 
\]
\[ B(i, j) = \begin{cases} 
0 & \text{if } i = j \\
\min_k \{ B(i, k) + B(k + 1, j) + r_i c_k c_j \} & \text{otherwise} 
\end{cases} \]
\[ C(1, 6) = \min \begin{cases} 
  k = 1 & C'(1, 1) + C'(2, 6) + r_1 c_1 c_6 \\
  k = 2 & C'(1, 2) + C'(3, 6) + r_1 c_2 c_6 \\
  k = 3 & C'(1, 3) + C'(4, 6) + r_1 c_3 c_6 \\
  k = 4 & C'(1, 4) + C'(5, 6) + r_1 c_4 c_6 \\
  k = 5 & C'(1, 5) + C'(6, 6) + r_1 c_5 c_6 
\end{cases} \]
\[ C(1, 6) = \min \begin{cases} 
  k = 1 & 0 + 10500 + 30 \cdot 35 \cdot 25 \\
  k = 2 & 15750 + 5375 + 30 \cdot 15 \cdot 25 \\
  k = 3 & 7875 + 3500 + 30 \cdot 5 \cdot 25 \\
  k = 4 & 9375 + 5000 + 30 \cdot 10 \cdot 25 \\
  k = 5 & 11875 + 0 + 30 \cdot 20 \cdot 25 
\end{cases} \]
$C(1, 6) = \min \begin{cases} 
  k = 1 & 0 + 10500 + 26250 \\
  k = 2 & 15750 + 5375 + 11250 \\
  k = 3 & 7875 + 3500 + 3750 \\
  k = 4 & 9375 + 5000 + 7500 \\
  k = 5 & 11875 + 0 + 15000 
\end{cases}$
<table>
<thead>
<tr>
<th>30</th>
<th>35</th>
<th>15</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>30<em>35</em>15 = 15750</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35<em>15</em>5 = 2625</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15<em>5</em>10 = 750</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15<em>15</em>10 = 750</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35<em>15</em>15 = 3500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10<em>20</em>25 = 5000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6</th>
<th>15125</th>
<th>10500</th>
<th>5375</th>
<th>3500</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>11875</td>
<td>7125</td>
<td>2500</td>
<td>1000</td>
</tr>
<tr>
<td>4</td>
<td>9375</td>
<td>4375</td>
<td>750</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>7875</td>
<td>2625</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>15750</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
\[
\begin{align*}
30 \times 35 \times 15 &= 15750 \\
35 \times 15 \times 5 &= 2625 \\
15 \times 5 \times 10 &= 750 \\
5 \times 10 \times 20 &= 1000 \\
10 \times 20 \times 25 &= 5000
\end{align*}
\]
**matrix-chain-mult(p)**

initialize array \( m[x,y] \) to zero
matrix-chain-mult(p)

initialize array m[x,y] to zero
starting at diagonal, working towards upper-left

compute m[i,j] according to

\[
\begin{cases}
0 & \text{if } i = j \\
\min_k \{ B(i, k) + B(k + 1, j) + r_i c_k c_j \}
\end{cases}
\]
running time?

initialize array $m[x,y]$ to zero

starting at diagonal, working towards upper-left

compute $m[i,j]$ according to

$$
\begin{cases}
0 & \text{if } i = j \\
\min_k \{ B(i, k) + B(k + 1, j) + r_i c_k c_j \} & \text{otherwise}
\end{cases}
$$
It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to heaven, we were all going direct the other way - in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.
It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to heaven, we were all going direct the other way - in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.

never print in the margin!

are simply not allowed
It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to heaven, we were all going direct the other way - in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.
It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to heaven, we were all going direct the other way - in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.
It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to heaven, we were all going direct the other way - in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.
It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to heaven, we were all going direct the other way - in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.
Typesetting problem

input:

output:
  such that
Typesetting problem

input: \[ W = \{ w_1, w_2, w_3, \ldots, w_n \} \]

output: \[ L = (w_1, \ldots, w_{\ell_1}), (w_{\ell_1+1}, \ldots, w_{\ell_2}), \ldots, (w_{\ell_x+1}, \ldots, w_n) \]

such that
Typesetting problem

input: \[ W = \{ w_1, w_2, w_3, \ldots, w_n \} \]

output: \[ L = (w_1, \ldots, w_{\ell_1}), (w_{\ell_1+1}, \ldots, w_{\ell_2}), \ldots, (w_{\ell_x+1}, \ldots, w_n) \]

\[ c_i = \left( \sum_{j=\ell_i+1}^{\ell_i+1} |w_j| \right) + (\ell_{i+1} - \ell_i - 1) \]

such that \[ c_i \leq M \quad \forall i \]

\[ \min \sum (M - c_i)^2 \]
how to solve

define the right variable:
imagine optimal solution
imagine optimal solution

last line
some word has to be the first-word-of-last-line (fwoll)
imagine optimal solution
last line

slack when line starts with
we
FWOLL is slack when line starts with $w_\ell$.

\[ \text{BEST}_n = \text{BEST}_{\ell-1} + S_{\ell,n} \]
how many candidates are there for the foothill?
is $w_1$ flat?

there is no slack (no solution even) because words go beyond edge!

define $S_{1,n} = \infty$ if this happens
is \( w_2 \) fwoll?

\[ S_{2,n} = \infty \]
is $w_j$ fwoll?
imagine optimal solution

w_{\ell} S_{\ell,n} last line

fwoll is \ w_{\ell} slack when line starts with \ w_{\ell}
which word is fwoll?

\[
\text{BEST}_n = \min \}
\]
which word is fwwwll?

\[ \text{BEST}_n = \min \left\{ \text{BEST}_0 + S_{1,n}^2, \text{BEST}_1 + S_{2,n}^2, \text{BEST}_2 + S_{3,n}^2, \ldots, \text{BEST}_{\ell-1} + S_{\ell,n}^2, \ldots, \text{BEST}_{n-1} + S_{n,n}^2 \right\} \]
how to compute $S_{i,j}$

slack when line starts with $w_i$ and ends $w_j$
Simplest case

$S_{1,1}$

slack when line starts with $w_i$ and ends $w_i$
Simplest case

slack when line starts with $w_i$ and ends $w_2$

$S_{1,2}$
how to compute $S_{i,j}$

slack when line starts with $w_i$
and ends $w_j$
typesetting algorithm

make table for $S_{i,j}$
typesetting algorithm

make table for $S_{i,j}$

for $i=1$ to $n$

$$best[i] = \min \{ best[j] + s[j+1][i]^2 \}$$

```java
// compute best_0,...,best_n
    int best[] = new int[n+1];
    int choice[] = new int[n+1];
    best[0] = 0;
    for(int i=1;i<=n;i++) {
        int min = infty;
        int ch  = 0;
        for(int j=0;j<i;j++) {
            int t = best[j] + S[j+1][i]*S[j+1][i];
            if (t<min) { min = t; ch = j;}
        }
        best[i] = min;
        choice[i] = ch;
    }
```
It was the best of times, it was the worst of times; it was the age of wisdom, it was the age of foolishness; it was the epoch of belief, it was the epoch of incredulity; it was the season of
first step: make $S_{i,j}$

|   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$M = 42$

$S_{i,i} = M - |w_i|$  

$S_{i,j} = S_{i,j-1} - 1 - |w_j|$
first step: make $S_{i,j}$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40</td>
<td>36</td>
<td>32</td>
<td>27</td>
<td>24</td>
<td>17</td>
<td>14</td>
<td>10</td>
<td>6</td>
<td>0</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$M = 42$
first step: make $S_{i,j}$
second step: compute

\[
\text{BEST}_i = \min_{j=0}^{i-1} \left\{ \text{BEST}_j + S^2_{j+1,i} \right\}
\]
second step: compute

\[
\text{BEST}_i = \min_{j=0}^{i-1} \{ \text{BEST}_j + S_{j+1}^2, i \}
\]

\[
\begin{array}{cccccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & \ldots & 40 & 1 & 1 & \text{best} \\
1 & 40 & 36 & 32 & 27 & 24 & 17 & 14 & 10 & 6 & 0 & 99 & 99 & 99 & 0 & 0 \\
2 & 39 & 35 & 30 & 27 & 20 & 17 & 13 & 9 & 3 & 0 & 99 & 99 & \text{best} & & \\
\end{array}
\]
second step: compute

\[
\text{BEST}_i = \min_{j=0}^{i-1} \left\{ \text{BEST}_j + S_{j+1,i}^2 \right\}
\]
Running time

make table for $S_{i,j}$

for $i=1$ to $n$

    best[$i$] = min{ best[$j$] + $s[j+1][i]^2$ }
PROBLEM: REDUCE IMAGE

scaling: distortion
deleting column: distortion
delete the most invisible seam
Shai Avidan
Mitsubishi Electric Research Lab
Ariel Shamir
The interdisciplinary Center & MERL
WHICH SEAM TO DELETE?
ENERGY OF AN IMAGE

\[ e(I) = \left| \frac{\partial}{\partial x} I \right| + \left| \frac{\partial}{\partial y} I \right| \]

"magnitude of gradient at a pixel"

\[ \frac{\partial}{\partial x} I_{x,y} = I_{x-1,y} - I_{x+1,y} \]
energy of sample image
thanks to Jason Lawrence for gradient software
BEST SEAM HAS LOWEST ENERGY
FINDING LOWEST ENERGY SEAM?
definition: $S_n(j)$
definition:

\[ S_n(j) \text{ best seam ending at } (n,j) \]
BEST SEAM TO DELETE HAS TO BE THE BEST AMONG $S_n(1), S_n(2), \ldots, S_n(m)$
IDEA: COMPUTE + COMPARE
SMALLER PROBLEM APPROACH
IMAGINE YOU HAVE THE SOLUTION TO THE FIRST $n-1$ ROWS
\[ n \quad e(n, 1) \quad e(n, 2) \quad e(n, j) \]

\[ n-1 \quad S_{n-1}(1) \quad S_{n-1}(2) \quad S_{n-1}(3) \quad S_{n-1}(m) \]
\[ S_n(1) = e(n, 1) + \min\{S_{n-1}(1), S_{n-1}(2)\} \]
$S_i(j) = e(n, j)$
\[ S_i(j) = e(i, j) + \min \left\{ S_{i-1}(j - 1), S_{i-1}(j), S_{i-1}(j + 1) \right\} \]
ALGORITHM

start at bottom of picture
ALGORITHM

start at bottom of picture. initialize $S_1(i) = e(1, i)$
ALGORITHM

start at bottom of picture. initialize $S_1(i) = e(1, i)$

for $i=2,n$ use formula to compute $S_{i+1}( \cdot )$

\[
S_i(j) = e(i, j) + \min \left\{ \begin{array}{c}
S_{i-1}(j - 1) \\
S_{i-1}(j) \\
S_{i-1}(j + 1)
\end{array} \right\}
\]
ALGORITHM

1. Start at the bottom of the picture.
2. Initialize $S_1(i) = e(1, i)$
3. For $i = 2, n$ use the formula to compute $S_{i+1}(\cdot)$
   
   
   $S_i(j) = e(i, j) + \min \left\{ \begin{array}{c} S_{i-1}(j-1) \\ S_{i-1}(j) \\ S_{i-1}(j+1) \end{array} \right\}$

   
   
   

\[ \begin{array}{cccccccccc} n & & & & & \cdots & & & & \hline 2 & & & & & \cdots & & & \hline 1 & & & & & \cdots & & & \hline \end{array} \]
ALGORITHM

start at bottom of picture. initialize \( S_1(i) = e(1, i) \)

for \( i = 2, n \) use formula to compute \( S_{i+1}(\cdot) \)

\[
S_i(j) = e(i, j) + \min \left\{ \begin{array}{c} S_{i-1}(j - 1) \\ S_{i-1}(j) \\ S_{i-1}(j + 1) \end{array} \right. 
\]

pick best among top row, backtrack.
RUNNING TIME

start at bottom of picture. initialize $S_1(i) = e(1,i)$

for $i=2,n$ use formula to compute $S_{i+1}(\cdot)$

$$S_i(j) = e(i,j) + \min \left\{ \begin{array}{l} S_{i-1}(j-1) \\ S_{i-1}(j) \\ S_{i-1}(j+1) \end{array} \right\}$$

pick best among top row, backtrack.