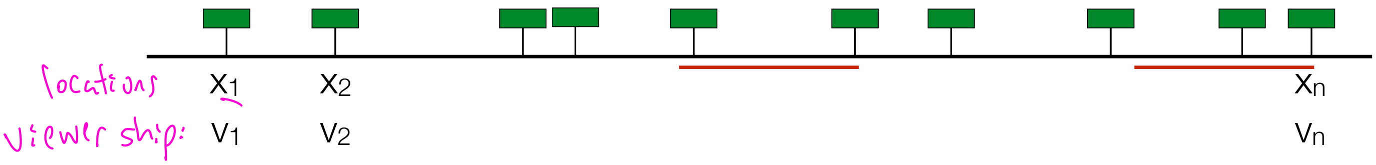
L8 5800

feb 11/14 2022

shelat

Billboard problem

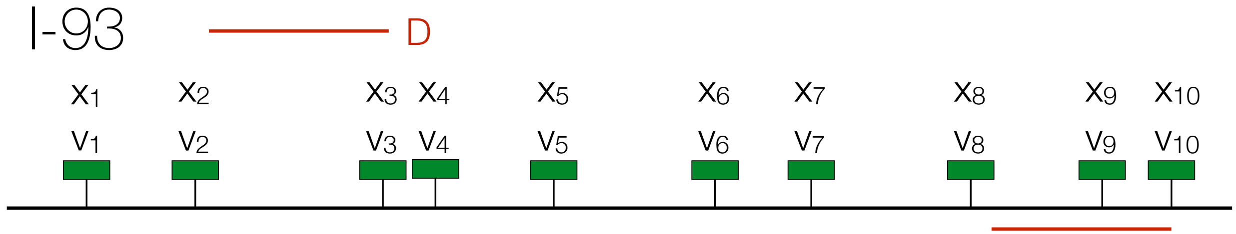
I-93



D

distance parameter

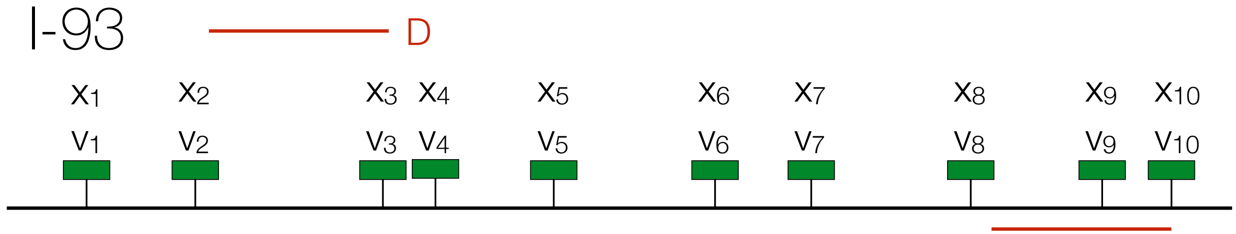
Cannot place ads that are closer than D miles apart



locations viewership distance constraint

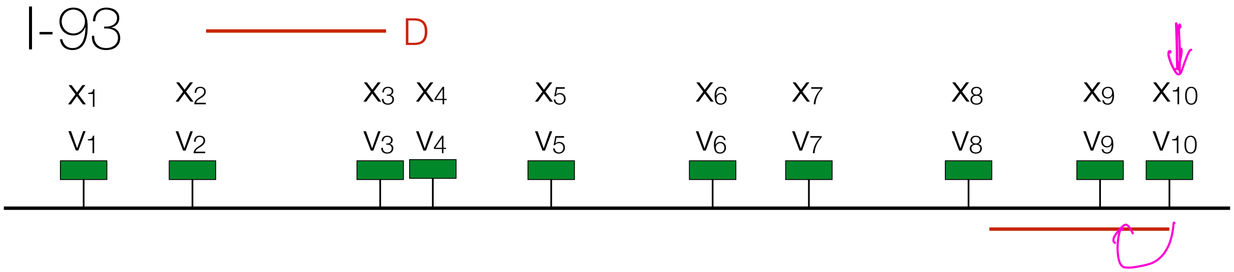
Input is $((x_1, \ldots, x_n)(v_1, \ldots, v_n), D)$

Best_n = maximum viewers one can obtain for an ad that appears no closer than D along the highway using billboards 1...n.



Input is $((x_1, \ldots, x_n)(v_1, \ldots, v_n), D)$

Best$_n$ = Max viewers for a campaign that uses billboards {1…n} with separation D.

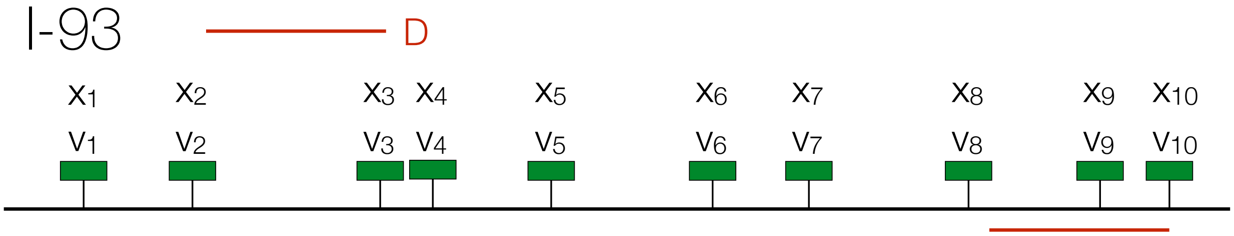


Input is $((x_1, \ldots, x_n)(v_1, \ldots, v_n), D)$

Best_n = Max viewers for a campaign that uses billboards {1…n} with separation D.

$$Best_n = \max \begin{cases} Best_{n-1} \\ v_n + Best_{closest_D(n)} \end{cases}$$

the closest billboard to n that is D away



Input is $((x_1, \ldots, x_n)(v_1, \ldots, v_n), D)$

Best$_n$ = Max viewers for a campaign that uses billboards {1…n} with separation D.

$$Best_n = \max \begin{cases} Best_{n-1} \\ v_n + Best_{closest_D(n)} \end{cases}$$

Familiar?

Familiar?

$$Best_n =$$

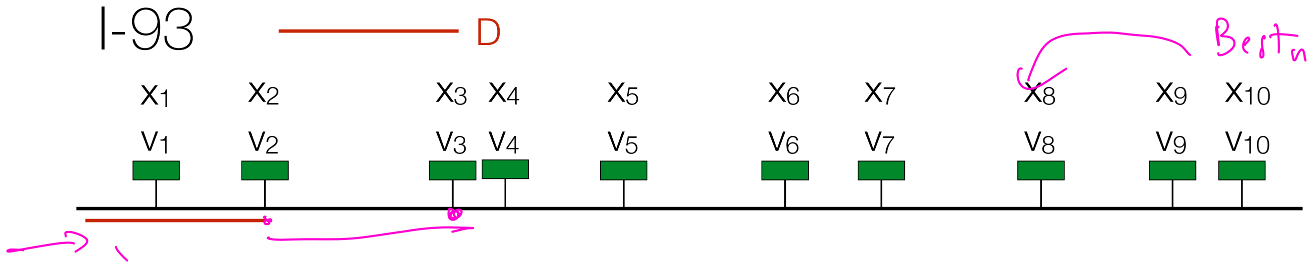
Familiar?

$$Best_n = \max \begin{cases} Best_{n-1} \\ v_n + Best_{closest_D(n)} \end{cases}$$

Familiar?

$$Best_n = \max \begin{cases} Best_{n-1} \\ v_n + Best_{closest_D(n)} \end{cases}$$

This equation is very similar to the log-cutter equation, with one difference. We cannot simply use the price to pick the sub-problem, we have to use D:



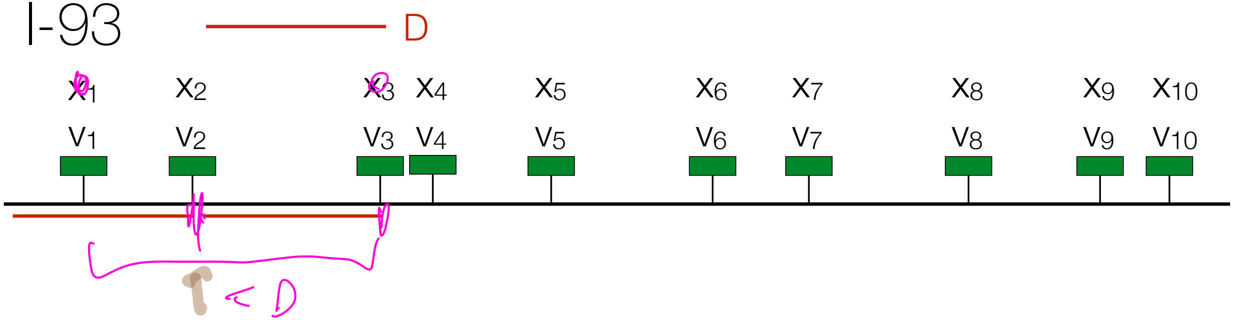
$Best_0 = 0$

$$Best_1 = \max \begin{Bmatrix} Best_0 \\ v_1 \end{Bmatrix} = v_1$$

$$Best_2 = \max \begin{Bmatrix} Best_1 \\ v_2 + Best_{closest(2)} \end{Bmatrix} = \begin{Bmatrix} v_1 \\ Best_0 + v_2 \end{Bmatrix} = \max \begin{Bmatrix} v_1 \\ v_2 \end{Bmatrix}$$

$$Best_3 = \max \Big\{$$

I-93



$\text{Best}_1 =$

$\text{Best}_2 =$

$$\text{Best}_3 = \max \begin{cases} \text{Best}_2 \\ v_3 + \text{Best}_1 \end{cases}$$

Billboard Problem

$$\text{BEST}_j = \max \begin{cases} \text{BEST}_{j-1} \\ v_j + \text{BEST}_{cl(j)} \end{cases}$$

```
best[0] = 0
for i=1 to n
        cl = i-1
        while ( x_i - x_cl < D  and cl > 0) { cl = cl - 1 }
        Best_i = max ( Best_{i-1} , v_i + Best_[cl] )
                                                ?
return best[n]
```

Billboard Problem

$$\mathrm{BEST}_j = \max \begin{cases} \mathrm{BEST}_{j-1} \\ v_j + \mathrm{BEST}_{cl(j)} \end{cases}$$

```
best[0] = 0
for i=1 to n
    cl = i-1
    while( (x[i]-x[cl])< D  && cl>0) cl=cl-1
    best[i] = max(best[i-1], vi+best[cl])
return best[n]
```

loop has n iterations

Run time can be i iterations

Total worse case running time: $\Theta(n^2)$

Billboard Problem

$$\mathrm{BEST}_j = \max \begin{cases} \mathrm{BEST}_{j-1} \\ v_j + \mathrm{BEST}_{cl(j)} \end{cases}$$

```
best[0] = 0
for i=1 to n
    cl = i-1
    while( (x[i]-x[cl])< D  && cl>0) cl=cl-1
    best[i] = max(best[i-1], vi+best[cl])
return best[n]
```

This line can take $\Theta(i)$ steps in the worst case.

Running time (worst case): $\Theta(n^2)$

Billboard Problem

$$\text{BEST}_j = \max \begin{cases} \text{BEST}_{j-1} \\ v_j + \text{BEST}_{cl(j)} \end{cases}$$

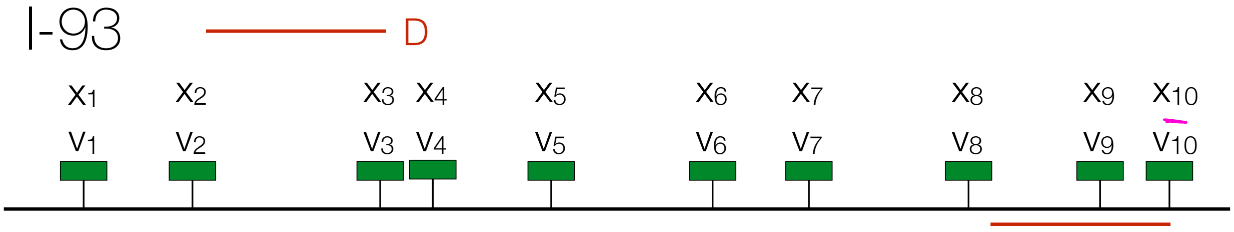
make this step faster.

```
best[0] = 0
for i=1 to n
    cl = i-1
    while( (x[i]-x[cl])< D  && cl>0) cl=cl-1
    best[i] = max(best[i-1], vi+best[cl])
return best[n]
```

This line can take $\Theta(i)$ steps in the worst case.

How can we improve?

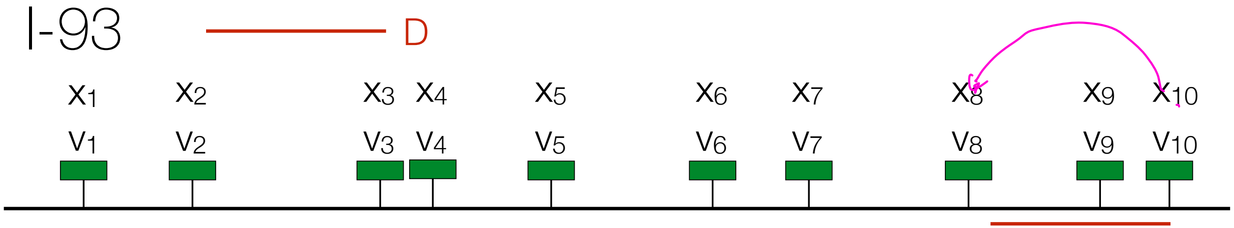
Running time (worst case): $\Theta(n^2)$



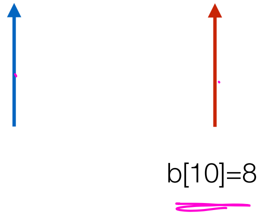
Faster way to find each billboard's buddy:

Pre-process to find every board's buddy.

right = n, left = n



Faster way to find each billboard's buddy:

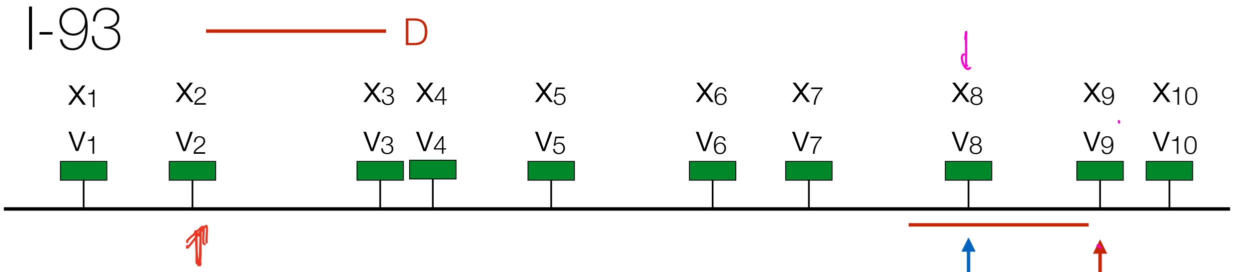


Pre-process to find every board's buddy.

right = n, left = n

move left until dist(x[right], x[left]) $\geq$ D

buddy[right] = left



Faster way to find each billboard's buddy:

b[10]=8

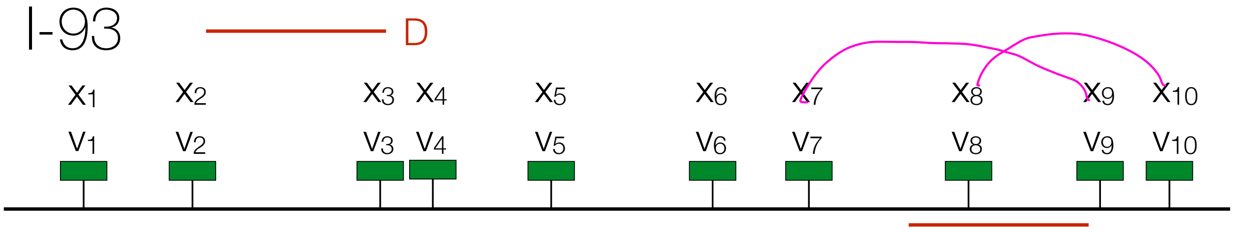
Pre-process to find every board's buddy.

right = n, left = n

move left until dist(x[right], x[left]) $\geq$ D

buddy[right] = left

move right to right



Faster way to find each billboard's buddy:

Pre-process to find every board's buddy.

right = n, left = n

while right and left are valid

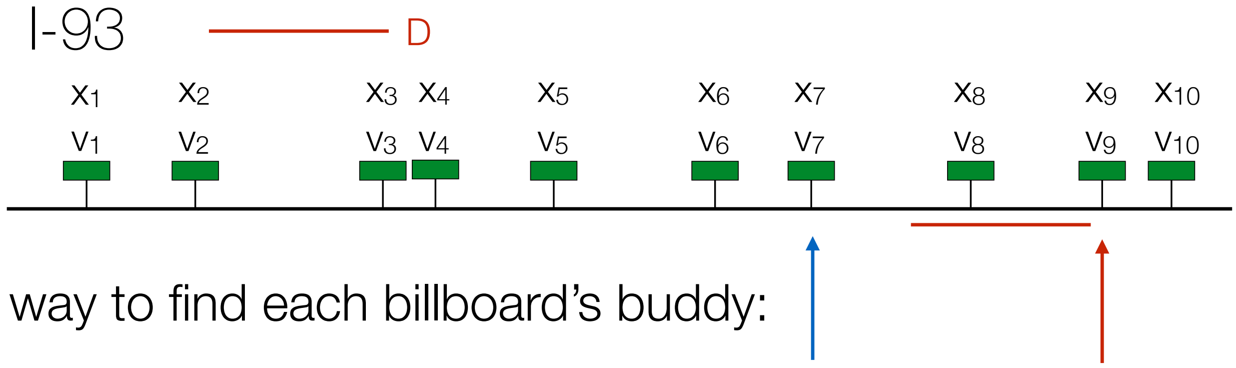
move left until dist(x[right], x[left]) $\geq$ D

buddy[right] = left

move right to right

b[10]=8

b[9]=7



Faster way to find each billboard's buddy:

b[10]=8

$\Theta(n)$

Pre-process to find every board's buddy.

right = n, left = n

while right and left are valid

move left until dist(x[right], x[left]) $\geq$ D

buddy[right] = left

move right to right

handle all of the remaining buddies for right

Better Billboard

$$\text{BEST}_j = \max \begin{cases} \text{BEST}_{j-1} \\ v_j + \text{BEST}_{cl(j)} \end{cases}$$

```
<Preprocess buddies>
best[0] = 0
for i=1 to n
    cl = i-1
    while( (x[i]-x[cl])< D  && cl>0) cl=cl-1
    best[i] = max(best[i-1], v[i]+best[buddy[i]])
return best[n]
```

$\Theta(1)$

the billboard index that is $\geq D$ away.

running time: $\Theta(n)$

Typesetting

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.

margin

slack

overrun

overfill

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.

First rule of typesetting

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.

0	0
0	0
2	4
12	144
2	4
1	1
6	36
2	4
2	4
	197

penalty for this typesetting of the first 9 lines.

______ is slack.

slack^2 is penalized

```
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.
```

is

Penalty is the square of the total slack.

It was the best of times, it was the worst	0	0
of times, it was the age of wisdom, it was	0	0
the age of foolishness, it was the epoch	2	4
of belief, it was the epoch of	12	144
incredulity, it was the season of Light,	2	4
it was the season of Darkness, it was the	1	1
spring of hope, it was the winter of	6	36
despair, we had everything before us, we	2	4
had nothing before us, we were all going	2	4
direct to heaven, we were all going direct	0	0
the other way - in short, the period was		
so far like the present period, that some of its		197
noisiest authorities insisted on its being received,		
for good or for evil, in the superlative degree of		
comparison only.		

1	It was the best of times, it was the	6	36
$l_1 =$ 10	worst of times, it was the age of wisdom,	1	1
$l_2 =$ 19	it was the age of foolishness, it was the	1	1
	epoch of belief, it was the epoch of	6	36
	incredulity, it was the season of Light,	2	4
	it was the season of Darkness, it was the	1	1
	spring of hope, it was the winter of	6	36
	despair, we had everything before us, we	2	4
	had nothing before us, we were all going	2	4
	direct to heaven, we were all going direct	0	0
	the other way - in short, the period was		
	so far like the present period, that some		
	of its noisiest authorities insisted on		123
	its being received, for good or for evil,		
	in the superlative degree of comparison		
	only.		

Typesetting problem

input: word list, $W = \{w_1, w_2, w_3 \ldots w_n\}$, margin M

output: list of words for each line, $L = (w_1 \ldots w_{\ell_1}), (w_{\ell_1+1} \ldots w_{\ell_2}) \ldots$

such that

$$c_i \leq M \quad \text{and} \quad \min \sum (M - c_i)^2$$

over all ways to typeset.

$$c_i = \sum_{j=\ell_i+1}^{\ell_{i+1}} w_j + \underline{(\ell_{i+1} - \ell_i - 1)}$$

sum of word length on the line.

space between words

Typesetting problem

input: $W = \{w_1, w_2, w_3, \ldots, w_n\}$ M

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: $W = \{w_1, w_2, w_3, \ldots, w_n\}$ $\quad M$

words

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 $c_i \leq M \quad \forall i$

$$\min \sum (M - c_i)^2$$

slack on line i

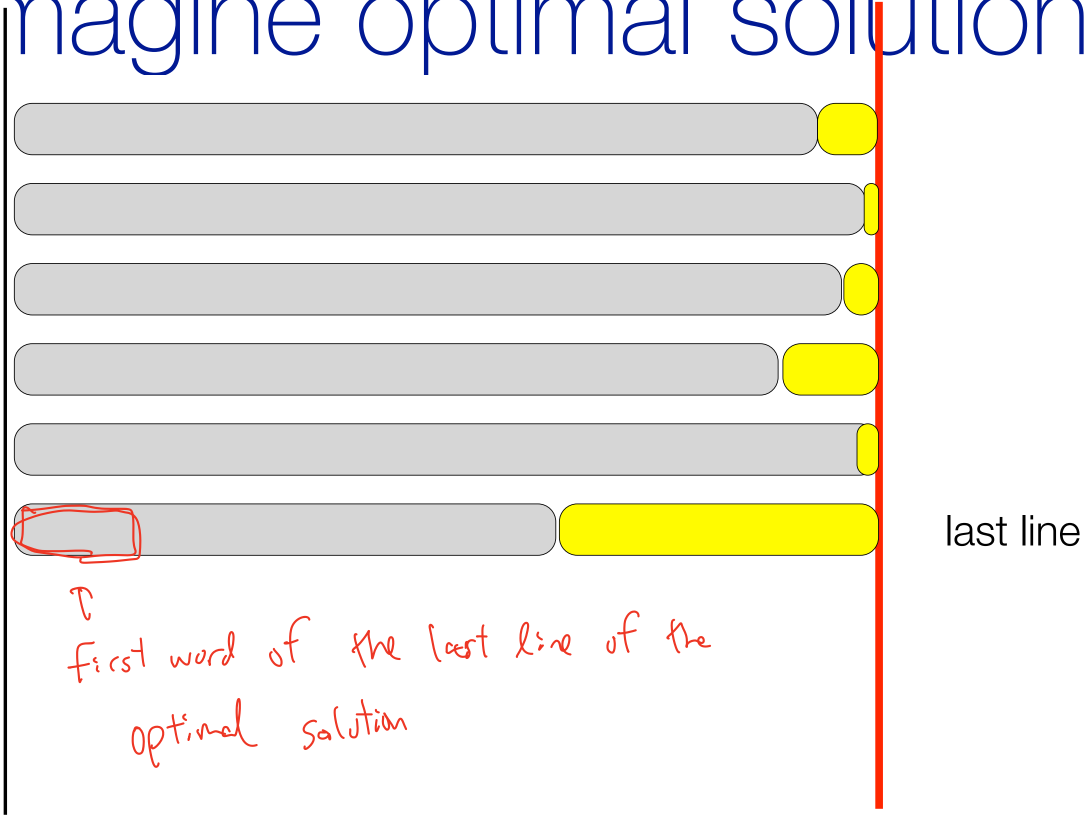
how to solve

define the right variable:

$Best_n$ = min penalty for typesetting the first n words.

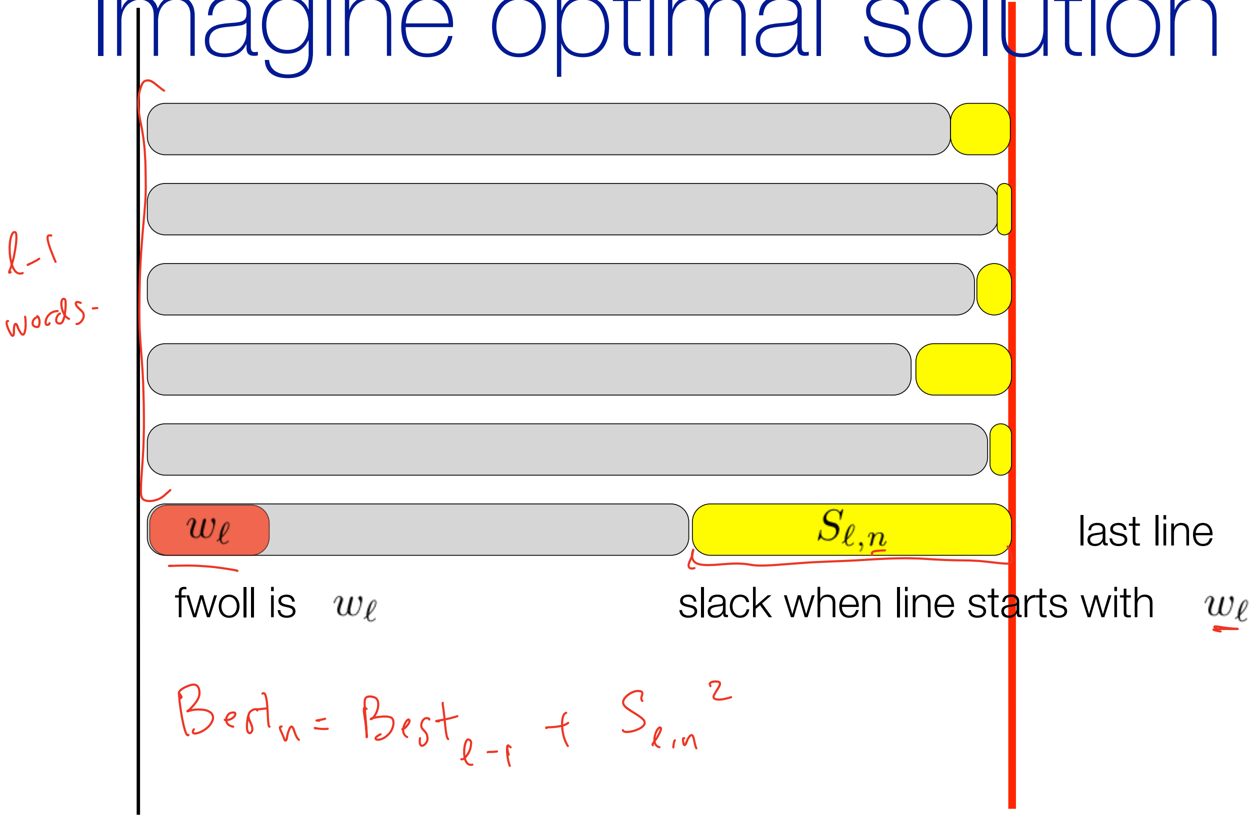
Imagine optimal solution

Imagine optimal solution



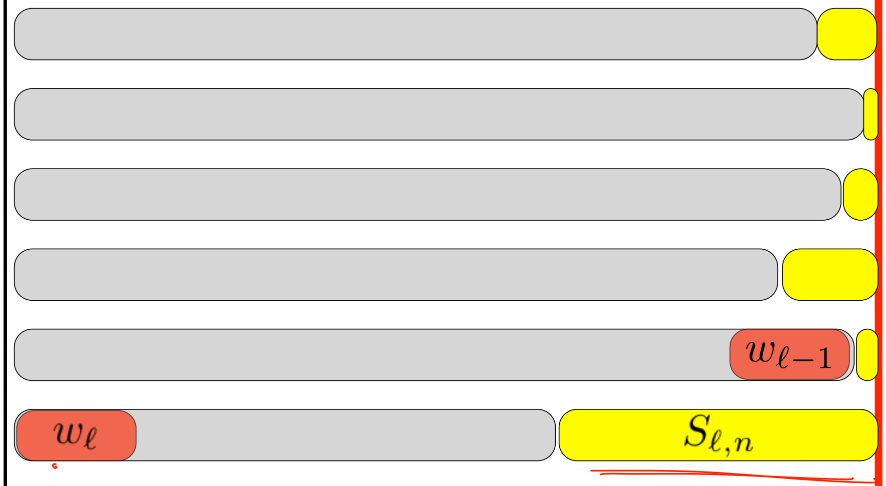
Some word has to be the first-word-of-last-line (fwoll)

Imagine optimal solution



$$Best_n = Best_{\ell-1} + S_{\ell,n}^2$$

Imagine optimal solution



last line

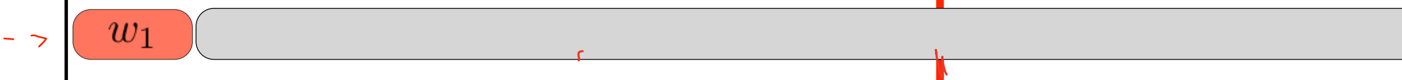
fwoll is w_ℓ

slack when line starts with w_ℓ

$$\mathrm{BEST}_n = \mathrm{BEST}_{\ell-1} + S_{\ell,n}^2$$

How many candidates are there for the fwoll?

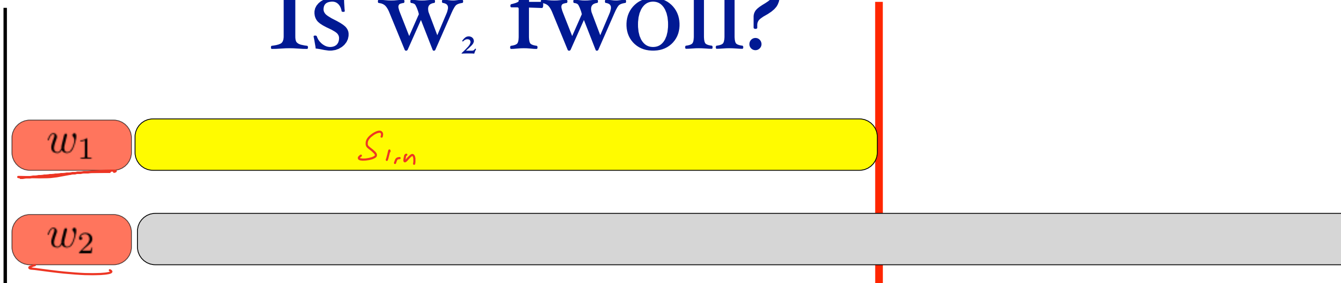
Is w_1 fwoll?



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

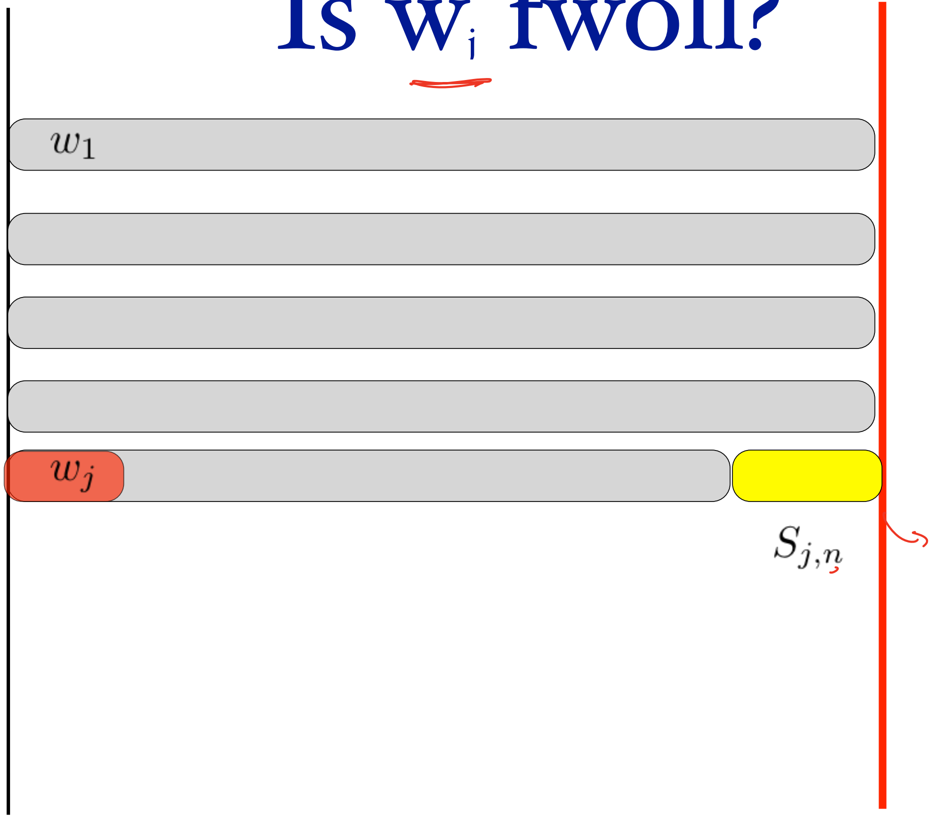
define $S_{1,n} = \infty$ if this happens

Is w_2 fwoll?



$$S_{2,n} = \infty$$

Is w_j fwoll?



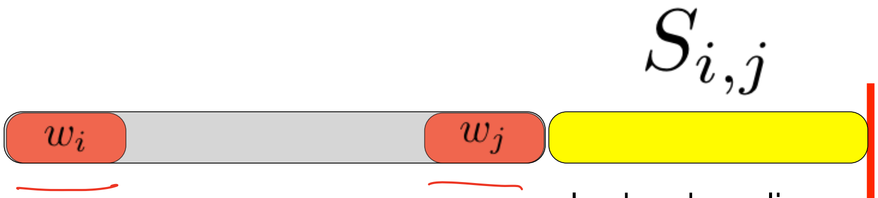
Which word is fwoll?

$$\mathrm{BEST}_n = \min \begin{cases} \mathrm{Best}_0 + (S_{1,n})^2 \\ \mathrm{Best}_1 + (S_{2,n})^2 \\ \mathrm{Best}_2 + (S_{3,n})^2 \\ \mathrm{Best}_3 + (S_{4,n})^2 \\ \vdots \\ \mathrm{Best}_{n-1} + (S_{n,n})^2 \end{cases}$$

Which word is fwoll?

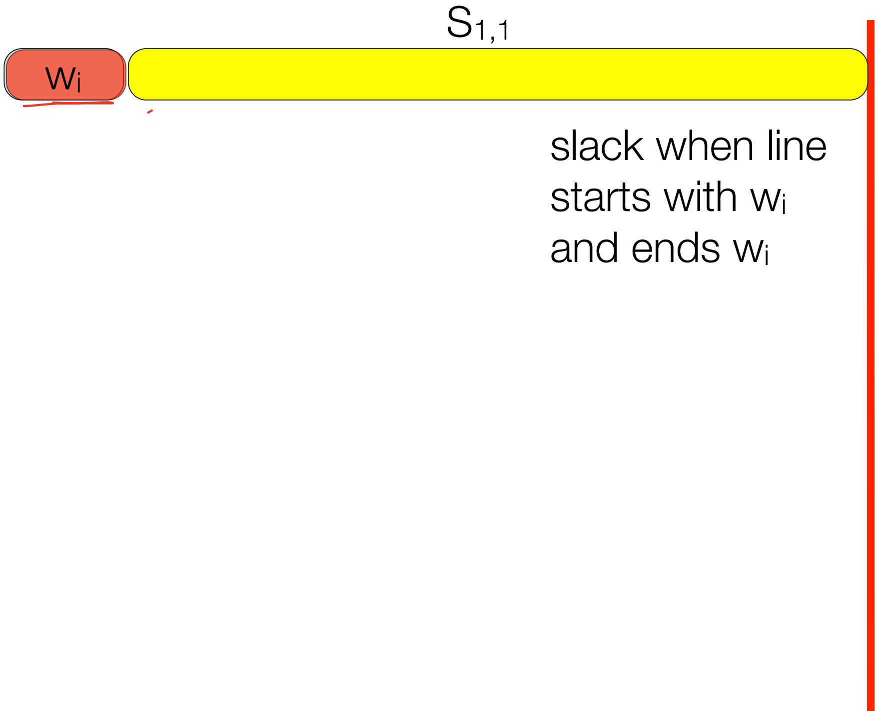
$$\mathrm{BEST}_n = \min \begin{cases} \mathrm{BEST}_0 + S_{1,n}^2 \\ \mathrm{BEST}_1 + S_{2,n}^2 \\ \mathrm{BEST}_2 + S_{3,n}^2 \\ \quad \dots \\ \mathrm{BEST}_{\ell-1} + S_{\ell,n}^2 \\ \\ \quad \dots \\ \\ \mathrm{BEST}_{n-1} + S_{n,n}^2 \end{cases}$$

How to compute $S_{i,j}$

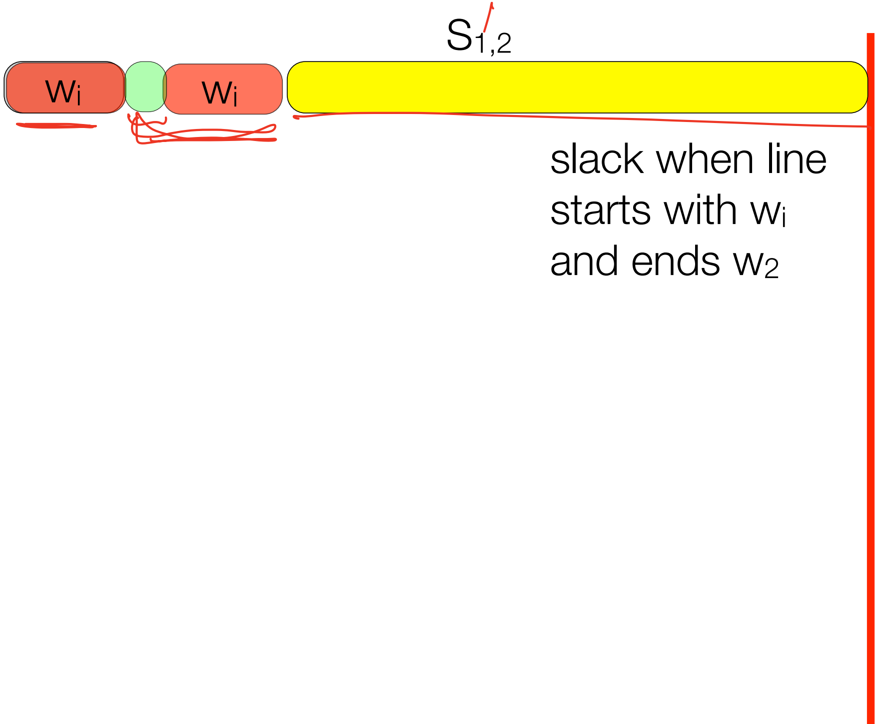


slack when line starts with w_i and ends w_j

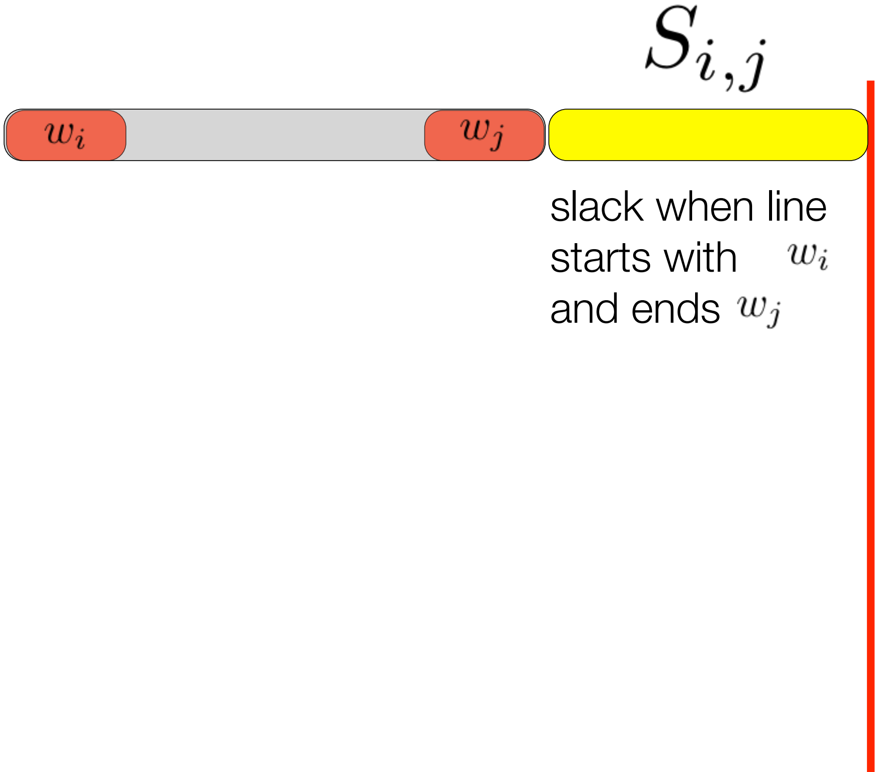
Simplest case



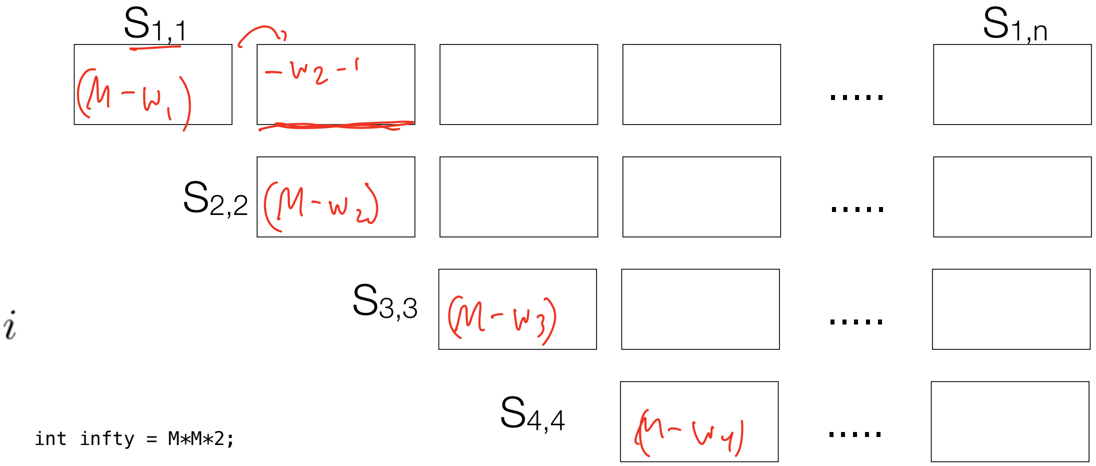
Simplest case



how to compute $S_{i,j}$



slack when line starts with w_i and ends w_j




```
int infty = M*M*2;

// compute S_ij
int S[][] = new int[n+1][n+1];
for(int i=1;i<=n;i++) {
    S[i][i] = M - lens[i];
    for(int j=i+1; j<=n; j++) {
        S[i][j] = S[i][j-1] - lens[j] - 1;
        if (S[i][j]<0) {
            while(j<=n) { S[i][j++] = infty; }
        }
    }
}
```

Typesetting algorithm

make table for $S_{i,j}$

for $i = 1 \ldots n$

$$\text{best}_i = \min_{j=1 \ldots i-1} \left\{ \text{best}_j + (S_{j+1,i})^2 \right\}$$

Typesetting algorithm

make table for $S_{i,j}$

for `i=1` to `n`

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

```
// 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;
  }
```

Example

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

$w = \{$ 2 3 3 4 2 6 2 3 3 5 2 6 2 3 3 3 2 7 2 3 3 3
2 12 2 3 3 5 2 7 2 3 3 5 2 12 2 3 3 6 2 $\}$

$M = 42$

first step: make$S_{i,j}$

42-2

	1	2	3	4	5	6	7	8	9	10	11	12	...
1	40	36	32	27	24	17	14	10	6	0	[illegible]	[illegible]	[illegible]

40-3-1

2 3 3 4 2 6 2 3 3 5 2 6 2 3 3 3 2 7 2 3 3 3
2 12 2 3 3 5 2 7 2 3 3 5 2 12 2 3 3 6 2

$M = 42$

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

$$S_{i,j} = S_{i,j-1} - 1 - |w_j|$$

First step: make $S_{i,j}$

S

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	40	36	32	27	24	17	14	10	6	0	99	99	99
2		39	35	30	27	20	17	13	9	3	0	99	99
3			39	34	31								

like ∞ here. It is larger than the values in our input.

2 3 3 4 2 6 2 3 3 5 2 6 2 3 3 3 2 7 2 3 3 3
2 12 2 3 3 5 2 7 2 3 3 5 2 12 2 3 3 6 2

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

$$S_{i,j} = S_{i,j-1} - 1 - |w_j|$$

$M = 42$

First step: make $S_{i,j}$

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	40	36	32	27	24	17	14	10	6	0	99	99	99
2		39	35	30	27	20	17	13	9	3	0	99	99
3													

2 3 3 4 2 6 2 3 3 5 2 6 2 3 3 3 2 7 2 3 3 3
2 12 2 3 3 5 2 7 2 3 3 5 2 12 2 3 3 6 2

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

$$S_{i,j} = S_{i,j-1} - 1 - |w_j|$$

second step: compute

	0	1	2	3	4	5	6	7	8	9	10	...
best	0	$\boxed{1600}$										

$$\min \begin{cases} \text{Best}_0 + (S_{1,2})^2 = 0 + 36^2 = 1296 \\ \text{Best}_1 + (S_{2,2})^2 = 1600 + 39^2 \end{cases}$$

$$\text{Best}_0 + (S_{1,1})^2 = 0 + 40^2 = 1600$$

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

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	40	36	32	27	24	17	14	10	6	0	99	99	99
2		39	35	30	27	20	17	13	9	3	0	99	99

$$\mathrm{BEST}_i = \min_{j=0}^{i-1}\left\{\mathrm{BEST}_j + S^2_{j+1,i}\right\}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
best	0	1600	1256											
choice	0	0	0	0	0	0	0	0	0	0	0			

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	40	36	32	27	24	17	14	10	6	0	99	99	99
2		39	35	30	27	20	17	13	9	3	0	99	99

$$\mathrm{BEST}_i = \min_{j=0}^{i-1} \left\{ \mathrm{BEST}_j + S^2_{j+1,i} \right\}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
best	0	1600	1296	1024										
choice	0	0	0	0	0	0	0	0	0	0	0			

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

It was

$$\mathrm{Best}_3 = \min \begin{cases} \mathrm{Best}_0 + (S_{1,3})^2 \quad \checkmark \\ \mathrm{Best}_1 + (S_{2,3})^2 \\ \mathrm{Best}_2 + (S_{3,3})^2 \end{cases}$$

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	40	36	32	27	24	17	14	10	6	0	99	99	99
2		39	35	30	27	20	17	13	9	3	0	99	99

$$\mathrm{BEST}_i = \min_{j=0}^{i-1} \left\{ \mathrm{BEST}_j + S^2_{j+1,i} \right\}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
best	0	1600	1296	1024										
choice	0	0	0	0	0	0	0	0	0	0	0			

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

It was the

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	40	36	32	27	24	17	14	10	6	0	99	99	99
2		39	35	30	27	20	17	13	9	3	0	99	99

$$\mathrm{BEST}_i = \min_{j=0}^{i-1} \left\{ \mathrm{BEST}_j + S_{j+1,i}^2 \right\}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
best	0	1600	1296	1024	729	576	289	196	100	36	0			
choice	0	0	0	0	0	0	0	0	0	0	0			

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

```
It was the best
```

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	40	36	32	27	24	17	14	10	6	0	99	99	99
2		39	35	30	27	20	17	13	9	3	0	99	99

$$\mathrm{BEST}_i = \min_{j=0}^{i-1} \left\{ \mathrm{BEST}_j + S_{j+1,i}^2 \right\}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
best	0	1600	1296	1024	729	576	289	196	100	36	0			
choice	0	0	0	0	0	0	0	0	0	0	0			

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

```
It was the best of
```

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	40	36	32	27	24	17	14	10	6	0	99	99	99
2		39	35	30	27	20	17	13	9	3	0	99	99

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

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
best	0	1600	1296	1024	729	576	289	196	100	36	0			
choice	0	0	0	0	0	0	0	0	0	0	0			

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

```
It was the best of times, it was the worst
```

$$\mathrm{BEST}_i = \min_{j=0}^{i-1} \left\{ \mathrm{BEST}_j + S_{j+1,i}^2 \right\}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
best	0	1600	1296	1024	729	576	289	196	100	36	0			
choice	0	0	0	0	0	0	0	0	0	0	0			

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

```
It was the best of times, it was the worst
of
```

$\mathrm{Best}_{10} + (S_{11,11})^2 = 0 + 40^2 = 1600$

$\mathrm{Best}_{11} = \min \{$

$$\text{BEST}_i = \min_{j=0}^{i-1} \left\{ \text{BEST}_j + S^2_{j+1,i} \right\}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
best	0	1600	1296	1024	729	576	289	196	100	36	0			
choice	0	0	0	0	0	0	0	0	0	0	0			

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

```
It was the best of times, it was the
worst of
```

$$\text{BEST}_{11} = \min \begin{cases} \text{BEST}_{10} + S^2_{11,11} & = 160 \\ \text{BEST}_9 + S^2_{10,11} & 36 + 34^2 \\ \text{BEST}_8 + S^2_{9,11} \\ \text{BEST}_7 + S^2_{8,11} \\ \text{BEST}_6 + S^2_{7,11} \\ \ldots \end{cases}$$

$$\text{BEST}_i = \min_{j=0}^{i-1} \left\{ \text{BEST}_j + S^2_{j+1,i} \right\}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
best	0	1600	1296	1024	729	576	289	196	100	36	0			
choice	0	0	0	0	0	0	0	0	0	0	0			

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

```
It was the best of times, it was
the worst of
```

$$\text{BEST}_{11} = \min \begin{cases} \text{BEST}_{10} + S^2_{11,11} \\ \text{BEST}_9 + S^2_{10,11} \\ \text{BEST}_8 + S^2_{9,11} \\ \text{BEST}_7 + S^2_{8,11} \\ \text{BEST}_6 + S^2_{7,11} \\ \ldots \end{cases}$$

$$\mathrm{BEST}_i = \min_{j=0}^{i-1} \left\{ \mathrm{BEST}_j + S_{j+1,i}^2 \right\}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
best	0	1600	1296	1024	729	576	289	196	100	36	0	818		
choice	0	0	0	0	0	0	0	0	0	0	0			

```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
It was the best of times,
it was the worst of
```

$$\mathrm{BEST}_{11} = \min \begin{cases} \mathrm{BEST}_{10} + S_{11,11}^2 \\ \mathrm{BEST}_{9} + S_{10,11}^2 \\ \mathrm{BEST}_{8} + S_{9,11}^2 \\ \mathrm{BEST}_{7} + S_{8,11}^2 \\ \mathrm{BEST}_{6} + S_{7,11}^2 \\ \ldots \end{cases}$$

$= 289 + 23^2 = 289 + 569$

$$\text{BEST}_i = \min_{j=0}^{i-1} \left\{ \text{BEST}_j + S^2_{j+1,i} \right\}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
best	0	1600	1296	1024	729	576	289	196	100	36	0	818	545	
choice	0	0	0	0	0	0	0	0	0	0	0	6	6	

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

```
It was the best of times,
it was the worst of times, it
```

$$\text{BEST}_{13} = \min \begin{cases} \text{BEST}_{12} + S^2_{13,13} \\ \text{BEST}_{11} + S^2_{12,13} \\ \ldots \\ \text{BEST}_{7} + S^2_{8,13} \\ \text{BEST}_{6} + S^2_{7,13} \end{cases}$$

$$\boxed{\mathrm{BEST}_i = \min_{j=0}^{i-1}\left\{\mathrm{BEST}_j + S_{j+1,i}^2\right\}}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
best	0	1600	1296	1024	729	576	289	196	100	36	0	818	545	
choice	0	0	0	0	0	0	0	0	0	0	0	6	6	

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

```
It was the best of times, it
was the worst of times, it
```

$$\mathrm{BEST}_{13} = \min \begin{cases} \mathrm{BEST}_{12} + S_{13,13}^2 \\ \mathrm{BEST}_{11} + S_{12,13}^2 \\ \dots \\ \mathrm{BEST}_{7} + S_{8,13}^2 \\ \mathrm{BEST}_{6} + S_{7,13}^2 \end{cases}$$

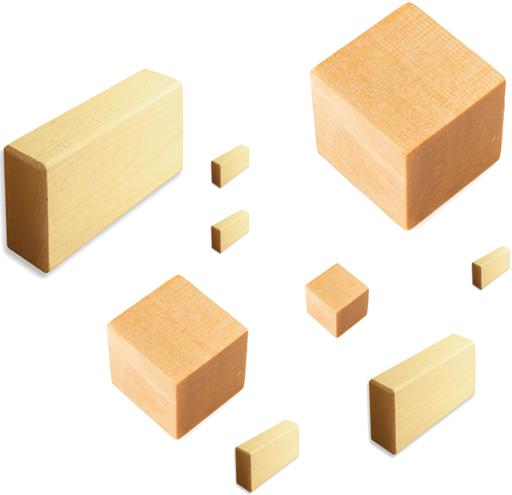
```
d-172-25-159-219:typeset abhi$ java typeset charly 42
0 best: 0 ch 0
1 best: 1600 ch 0
2 best: 1296 ch 0
3 best: 1024 ch 0
4 best: 729 ch 0
5 best: 576 ch 0
6 best: 289 ch 0
7 best: 196 ch 0
8 best: 100 ch 0
9 best: 36 ch 0
10 best: 0 ch 0
11 best: 818 ch 6
12 best: 545 ch 6
13 best: 452 ch 7
14 best: 340 ch 7
15 best: 244 ch 8
16 best: 164 ch 8
17 best: 117 ch 9
18 best: 37 ch 9
19 best: 16 ch 10
20 best: 0 ch 10
21 best: 509 ch 14
22 best: 413 ch 15
23 best: 344 ch 15
24 best: 133 ch 17
25 best: 118 ch 17
26 best: 62 ch 18
27 best: 32 ch 19
28 best: 4 ch 20
29 best: 444 ch 23
30 best: 348 ch 23
31 best: 277 ch 24
32 best: 197 ch 24
33 best: 149 ch 24
34 best: 87 ch 26
35 best: 66 ch 26
36 best: 446 ch 31
37 best: 377 ch 31
38 best: 297 ch 32
39 best: 233 ch 32
```

```
0 best: 0 ch 0
1 best: 1600 ch 0        It
2 best: 1296 ch 0        It was
3 best: 1024 ch 0        It was the
4 best: 729 ch 0         It was the best
5 best: 576 ch 0         It was the best of
6 best: 289 ch 0         It was the best of times,
7 best: 196 ch 0         It was the best of times, it
8 best: 100 ch 0         It was the best of times, it was
9 best: 36 ch 0          It was the best of times, it was the
10 best: 0 ch 0          It was the best of times, it was the worst
11 best: 818 ch 6        It was the best of times,\nit was the worst of
12 best: 545 ch 6        It was the best of times,\nit was the worst of times,
13 best: 452 ch 7        It was the best of times, it\nwas the worst of times, it
14 best: 340 ch 7        It was the best of times, it\nwas the worst of times, it was
15 best: 244 ch 8        It was the best of times, it was\nthe worst of times, it was the
16 best: 164 ch 8        It was the best of times, it was\nthe worst of times, it was the age
17 best: 117 ch 9        It was the best of times, it was the\nworst of times, it was the age of
18 best: 37 ch 9         It was the best of times, it was the\nworst of times, it was the age of wisdom,
19 best: 16 ch 10        It was the best of times, it was the worst\nof times, it was the age of wisdom, it
20 best: 0 ch 10         It was the best of times, it was the worst\nof times, it was the age of wisdom, it was
21 best: 509 ch 14       It was the best of times, it\nwas the worst of times, it was\nthe age of wisdom, it was the
22 best: 413 ch 15       It was the best of times, it was\nthe worst of times, it was the\nage of wisdom, it was the age
23 best: 344 ch 15       It was the best of times, it was\nthe worst of times, it was the\nage of wisdom, it was the age of
24 best: 133 ch 17       It was the best of times, it was the\nworst of times, it was the age of\nwisdom, it was the age of foolishness,
25 best: 118 ch 17       It was the best of times, it was the\nworst of times, it was the age of\nwisdom, it was the age of foolishness, it
26 best: 62 ch 18        It was the best of times, it was the\nworst of times, it was the age of wisdom,\nit was the age of foolishness, it was
```

```
// read input

 try {
 BufferedReader bin = new BufferedReader(new FileReader(args[0]));
 String line = bin.readLine();
 String words[] = line.split(" ");
 int n = words.length;
 int M = Integer.parseInt(args[1]);
 int lens[] = new int[n+1];
 for(int i=1;i<=n; i++) {
     lens[i] = words[i-1].length();
     if (lens[i]>M) {
         System.out.println("word too long");
         System.exit(1);
     }
 }

 int infty = M*M*2;

 // compute S_ij
 int S[][] = new int[n+1][n+1];
 for(int i=1;i<=n;i++) {
     S[i][i] = M - lens[i];
     for(int j=i+1; j<=n; j++) {
         S[i][j] = S[i][j-1] - lens[j] - 1;
         if (S[i][j]<0) {
             while(j<=n) { S[i][j++] = infty; }
         }
     }
 }
```

```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;
}

// backtrack to output linebreaks
int end = n;
int start   = choice[end]+1;
String lines[] = new String[n];
int cnt = 0;
while (end>0) {
    StringBuffer buf = new StringBuffer();
    for(int j=start; j<=end; j++) {
        buf.append(words[j-1] + " ");
    }
    lines[cnt++] = buf.toString();
    end = start-1;
    start = choice[end]+1;
}
```

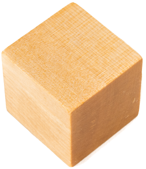
Knapsack



Sack has Capacity W

W

w_1 v_1



w_2 v_2

w_3 v_3

Each item
has a weight w_i
and a value v_i

Goal is to select
a set of items that
"**fit**" into the Knapsack
and have the **greatest** value.
Can only use each item once
(versus Logcutter)

Define a quantity that captures the optimal solution:

Best({1,…,n}, C) :

Consider the very first item. Is it part of the max solution?

Define a quantity that captures the optimal solution:

Best({1,…,n}, C) : max value obtainable from items {1…n} that fit in sack of size C

Consider the very first item. Is it part of the max solution?

w_1 v_1

Define a quantity that captures the optimal solution:

Best({1,…,n}, C) : max value obtainable from items {1…n} that fit in sack of size C

Consider the very first item. Is it part of the max solution?

w_1 v_1

$$B(1\ldots n, C) = \max \left\{ \begin{array}{cc} B(2\ldots n, C) & \text{if not included} \\ v_1 + B(2\ldots n-1, C - w_1) & \text{if in} \end{array} \right\}$$

Recursive structure

Either the best solution doesn't include item i

B({i…n}, W) = max

Or, it includes item i and the best solution for the remaining space, $W-w_i$

Recursive structure

Either the best solution doesn't include item i

$$B(\{i\ldots n\}, W) = \max \begin{cases} B(\{i+1\ldots n\}, W) \\ v_i + B(\{i+1\ldots n\}, W - w_i) & \text{If } W - w_i > 0 \end{cases}$$

Or, it includes item i and the best solution for the remaining space, $W-w_i$

Pick an order

Start from the last item

	0												W
B({ }, 0…W)	0	0	0	0	0	0	0	0	0	0	…	0	0

	0						w_n						W
B({ n }, 0…W)											…		

If $W - w_n > 0$

Pick an order

Start from the last item

B({ }, 0…W)

0												W
0	0	0	0	0	0	0	0	0	0	…	0	0

B({ n }, 0…W)

0						w_n						W
										…		

$$B(\{n\}, W) = \max \begin{cases} B(\{\}, W) \\ v_n + B(\{\}, W - w_n) & \text{If } W - w_n > 0 \end{cases}$$

Pick an order

Start from the last item

	0												W
B({ }, 0…W)	0	0	0	0	0	0	0	0	0	0	…	0	0

	0						w_n						W
B({ n }, 0…W)	0	0	0	0	0	0	v_n	v_n	v_n	v_n	…	v_n	v_n

$$B(\{n\}, W) = \max \begin{cases} B(\{\}, W) \\ v_n + B(\{\}, W - w_n) \quad \text{If } W - w_n > 0 \end{cases}$$

Pick an order

Start from the last item

	0												W
B({ }, 0…W)	0	0	0	0	0	0	0	0	0	0	…	0	0

	0						w_n						W
B({ n }, 0…W)	0	0	0	0	0	0	v_n	v_n	v_n	v_n	…	v_n	v_n

							15		17				40
	0						w_n		w_{n-1}				W
B({ n-1…n }, 0…W)											…		

Pick an order

Start from the last item

B({ }, 0…W)

0												W
0	0	0	0	0	0	0	0	0	0	…	0	0

B({ n }, 0…W)

0						w_n						W
0	0	0	0	0	0	v_n	v_n	v_n	v_n	…	v_n	v_n

B({ n-1…n }, 0…W)

						15		17				40
0						w_n		w_{n-1}				W
										…		

$$B(\{i \ldots n\}, W) = \max \begin{cases} B(\{i+1 \ldots n\}, W) \\ v_i + B(\{i+1 \ldots n\}, W - w_i) \end{cases}$$

Example

	1	2	3	4	5	6
v_i	4	5	9	1	10	3
w_i	1	3	2	4	7	2

$$C = 15$$

$B(\{\}, \ldots)$

0					5					10					W=15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$B(\{6\}, \ldots)$

0					5					10					W=15

$$B(\{i \ldots n\}, W) = \max \begin{cases} B(\{i+1 \ldots n\}, W) \\ v_i + B(\{i+1 \ldots n\}, W - w_i) \end{cases}$$

If $W - w_i > 0$

Example

	1	2	3	4	5	6
v_i	4	5	9	1	10	3
w_i	1	3	2	4	7	2

$$C = 15$$

$B(\{\}, \ldots)$

0					5					10					W=15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$B(\{6\}, \ldots)$

0					5					10					W=15
0	0	3	3	3	3	3	3	3	3	3	3	3	3	3	3

$$B(\{i\ldots n\}, W) = \max \begin{cases} B(\{i+1\ldots n\}, W) \\ v_i + B(\{i+1\ldots n\}, W - w_i) \end{cases}$$

If $W - w_i > 0$

Example

	1	2	3	4	5	6
v_i	4	5	9	1	10	3
w_i	1	3	2	4	7	2

$$C = 15$$

$B(\{6\}, \ldots)$

0					5					10					W=15
0	0	3	3	3	3	3	3	3	3	3	3	3	3	3	3

$B(\{5,6\}, \ldots)$

0					5					10					W=15

$$B(\{i \ldots n\}, W) = \max \begin{cases} B(\{i+1 \ldots n\}, W) \\ v_i + B(\{i+1 \ldots n\}, W - w_i) \end{cases}$$

If $W - w_i > 0$

Example

	1	2	3	4	5	6
v_i	4	5	9	1	10	3
w_i	1	3	2	4	7	2

$$C = 15$$

$B(\{6\}, \ldots)$

0					5					10					W=15
0	0	3	3	3	3	3	3	3	3	3	3	3	3	3	3

$B(\{5,6\}, \ldots)$

0					5					10					W=15
0	0	3	3	3	3	3	10	10	13	13	13	13	13	13	13

$$B(\{i\ldots n\}, W) = \max \begin{cases} B(\{i+1\ldots n\}, W) \\ v_i + B(\{i+1\ldots n\}, W - w_i) \end{cases}$$

If $W - w_i > 0$

Example

	1	2	3	4	5	6
v_i	4	5	9	1	10	3
w_i	1	3	2	4	7	2

$$C = 15$$

$B(\{5,6\}, \ldots)$

0					5					10					W=15
0	0	3	3	3	3	3	10	10	13	13	13	13	13	13	13

$B(\{4,5,6\}, \ldots)$

0					5					10					W=15

$$B(\{i\ldots n\}, W) = \max \begin{cases} B(\{i+1\ldots n\}, W) \\ v_i + B(\{i+1\ldots n\}, W - w_i) \end{cases}$$

If $W - w_i > 0$

Knapsack($\{w_i, v_i\}_n$, W)

Initialize B({n-1}, 0…W) = 0

for i from n to 1

for j from 0 to W

$$B(i, j) = \max \begin{cases} B(i+1, j) \\ v_i + B(i+1, j - w_i) \end{cases}$$

as long as $j > w_i$
because otherwise,
this term is negative

Return B(1,W)

Knapsack($\{w_i, v_i\}_n$, W)

Initialize B({n-1}, 0…W) = 0

for i from n to 1

    for j from 0 to W

        B(i, j) = B(i+1, j)

        if j > wi and B(i+1, j-wi)+ vi > S(i, j)
            B(i,j) = B(i+1, j-wi)+ vi

Return B(1,W)

How can we determine WHICH items are selected?

Knapsack($\{w_i, v_i\}_n$, W)

Initialize B({n-1}, 0…W) = 0

for i from n to 1

for j from 0 to W

B(i, j) = B(i+1, j)

pick(i, j) = false

if j > wi and B(i+1, j-wi)+ vi > B(i, j)

B(i,j) = B(i+1, j-wi)+ vi

pick(i, j) = true

//Backtrack to find solution

cap = W, sol = {}

for i from 1 to n

if picked(i,cap) = true { sol = sol + {i}; cap = cap - w_i; }

PROBLEM: REDUCE IMAGE WIDTH

scaling: distortion

deleting column: distortion

delete the most invisible seam

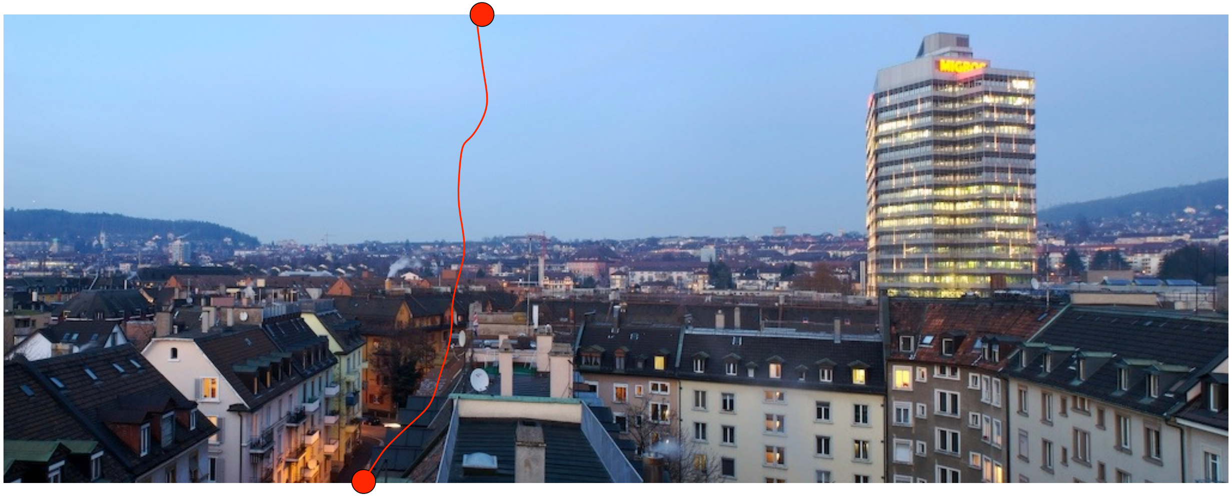
http://www.youtube.com/watch?v=qadw0BRKeMk

http://www.youtube.com/watch?v=qadw0BRKeMk

DEMO?

http://rsizr.com/

WHICH SEAM TO DELETE?



ENERGY OF AN IMAGE

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

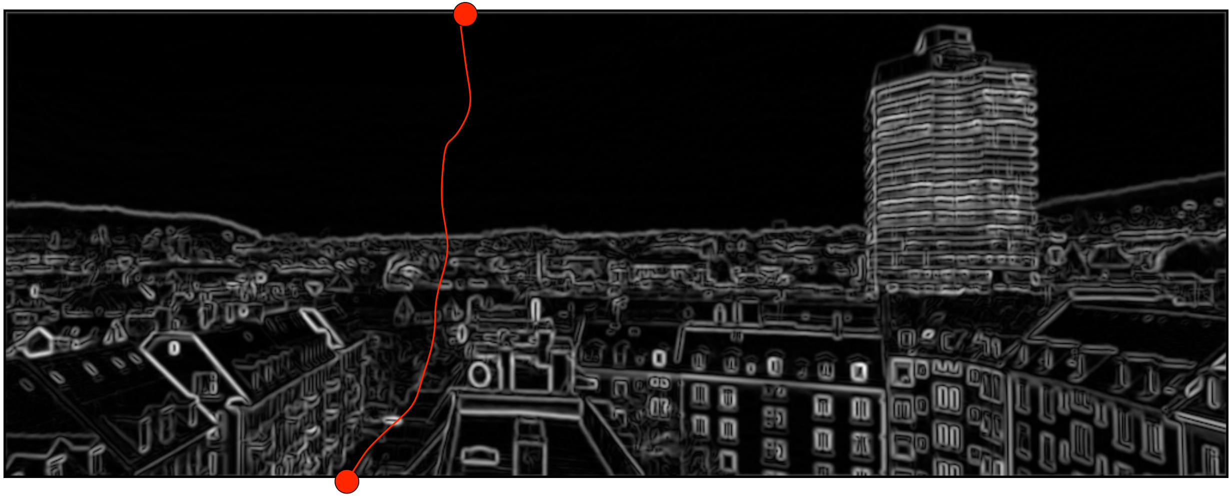
"magnitude of gradient at a pixel"

$$\frac{\partial}{\partial x} I_{x,y} = I_{x-1,y} - I_{x+1,y}$$

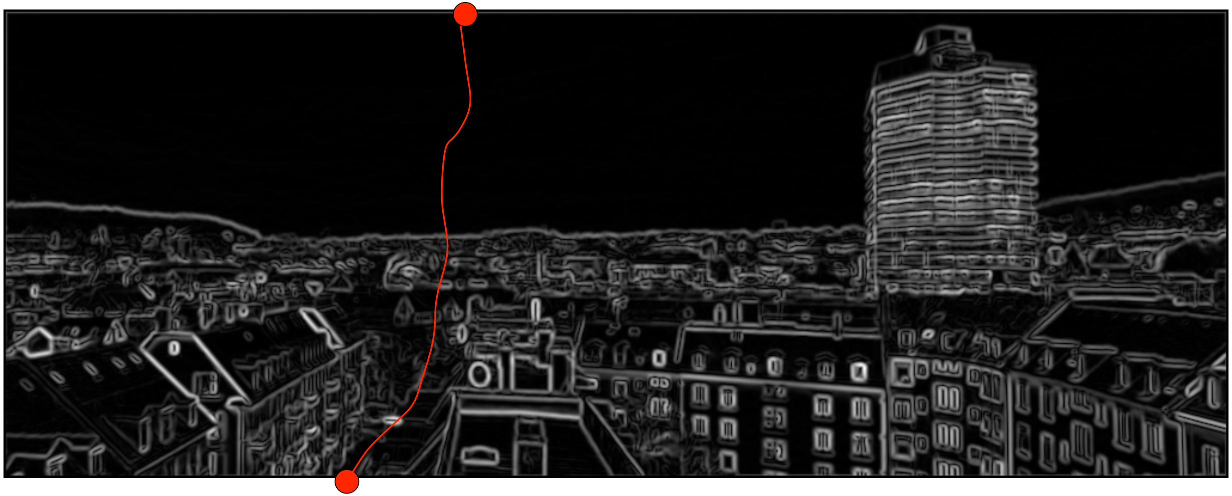
energy of sample image

thanks to Jason Lawrence for gradient software

BEST SEAM HAS LOWEST ENERGY



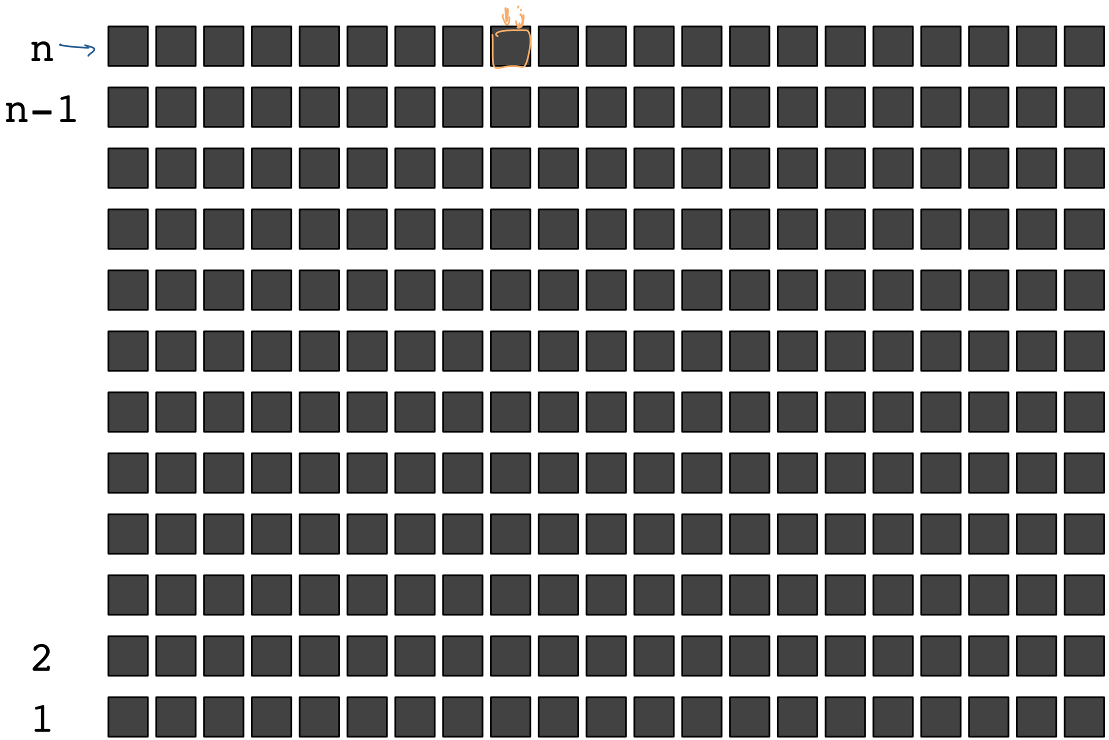
Finding lowest energy seam?



Define a variable:

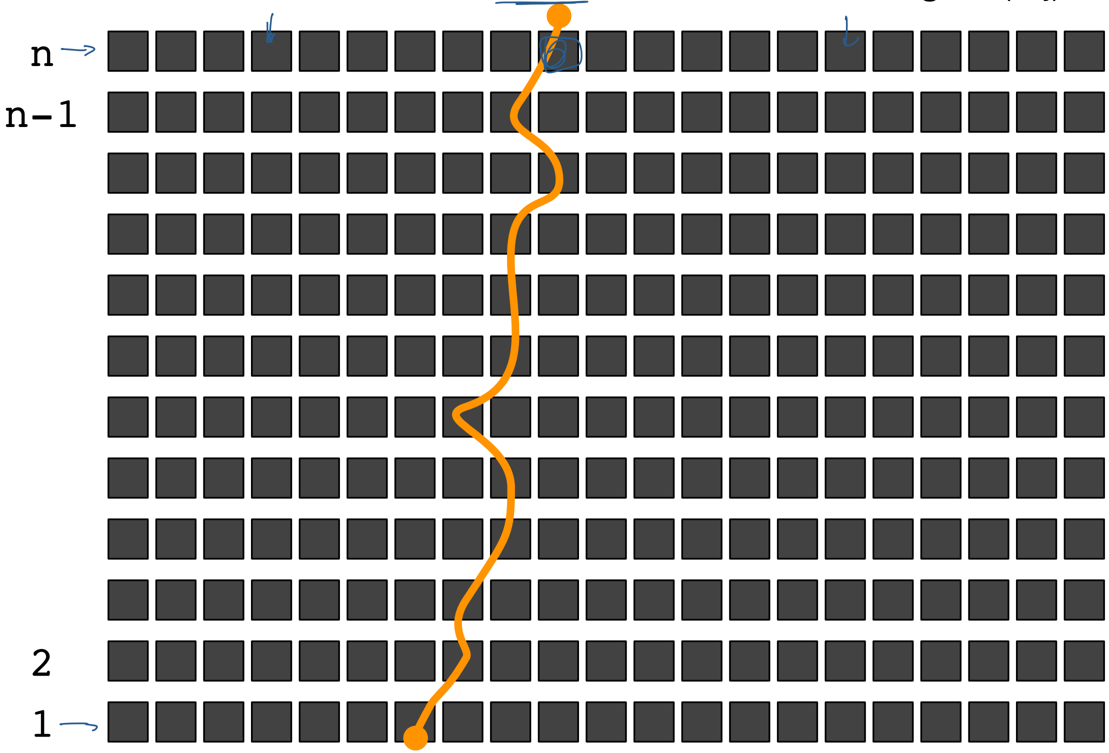
$$S_i(j)$$

definition: $S_n(j)$



definition:

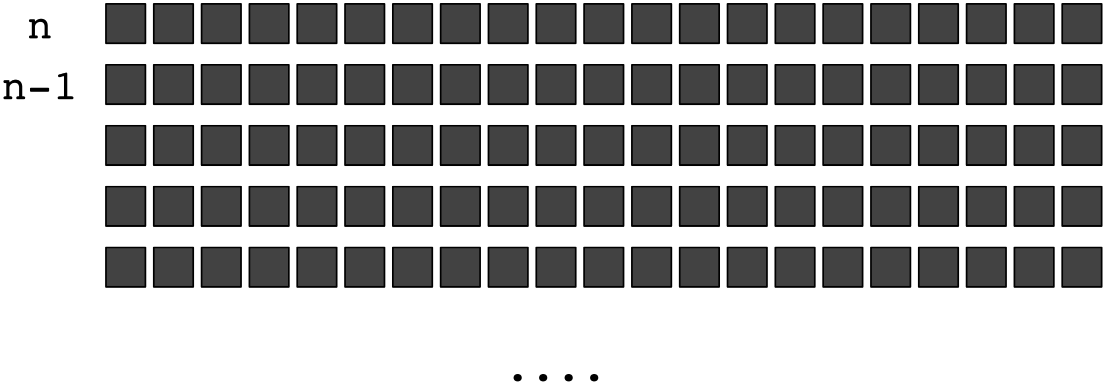
$S_n(j)$ best seam ending at (n,j)



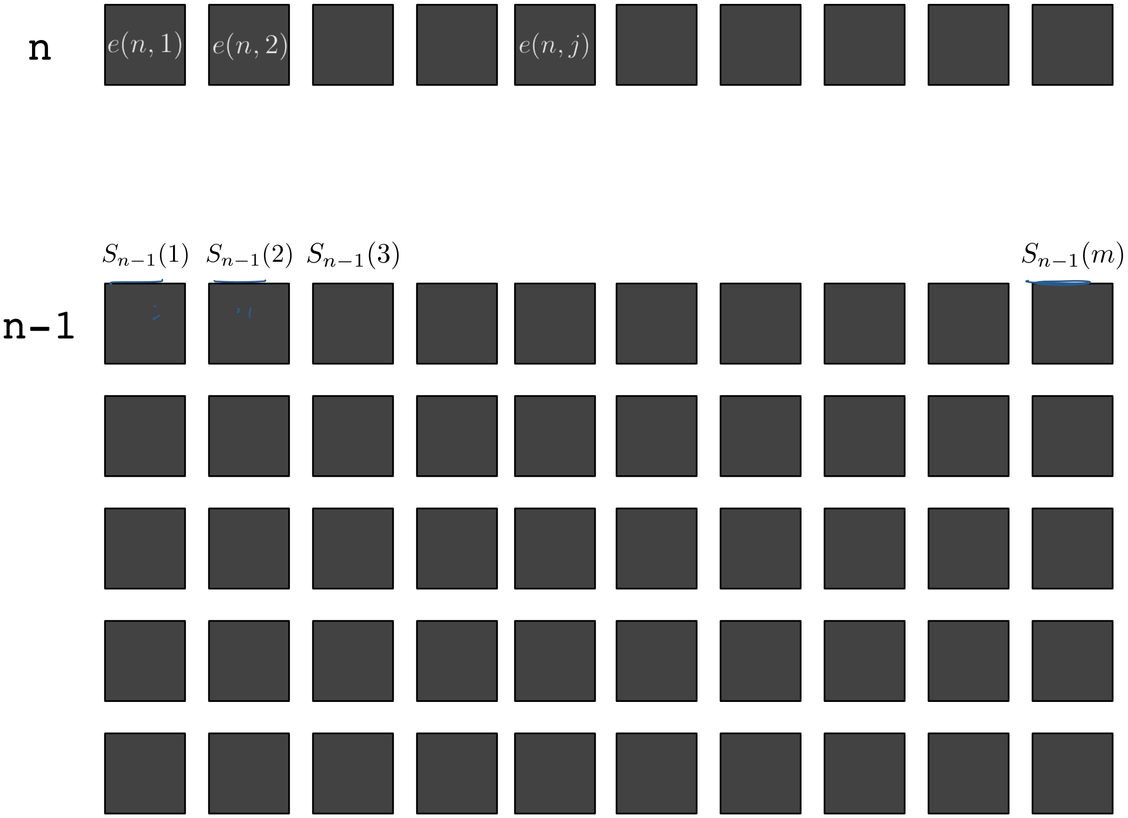
BEST SEAM TO DELETE HAS TO BE THE BEST AMONG

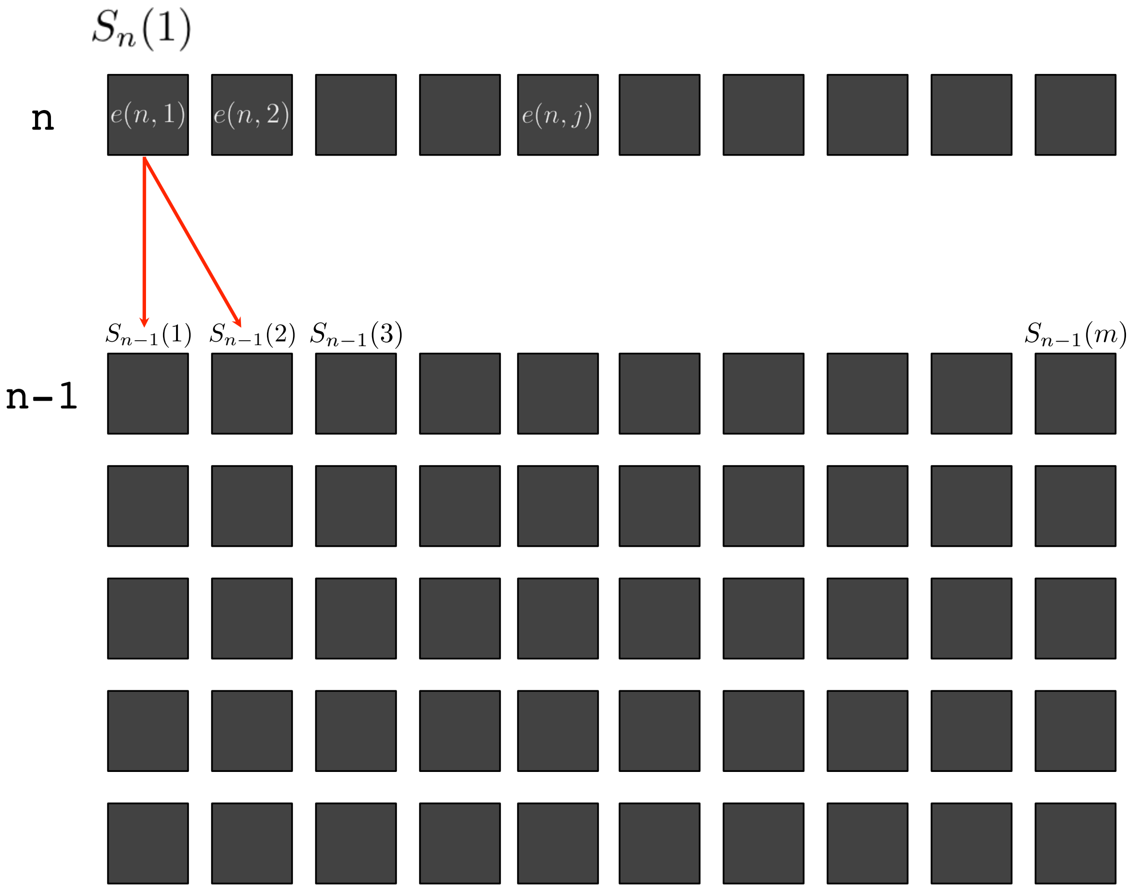
$$S_n(1), S_n(2), \ldots, S_n(m)$$

IDEA: COMPUTE + COMPARE

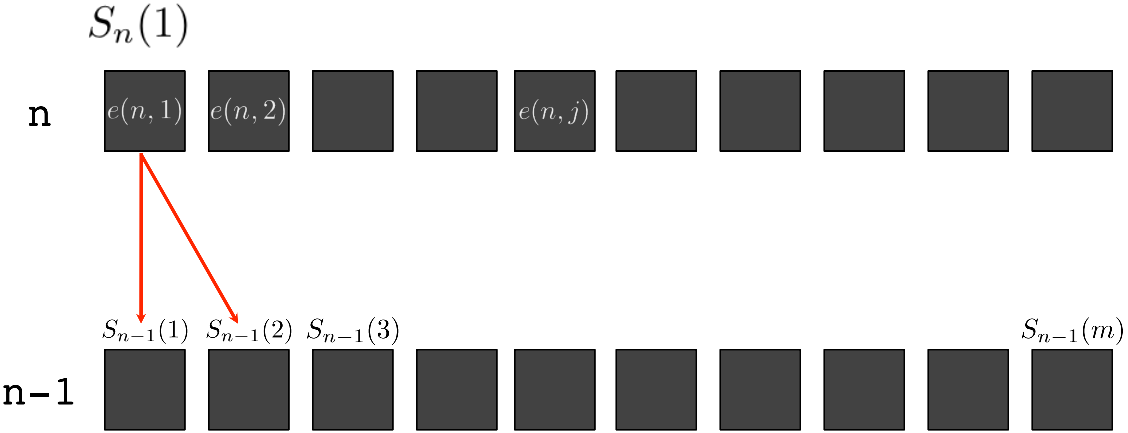


IMAGINE YOU HAVE THE SOLUTION TO THE FIRST N-1 ROWS

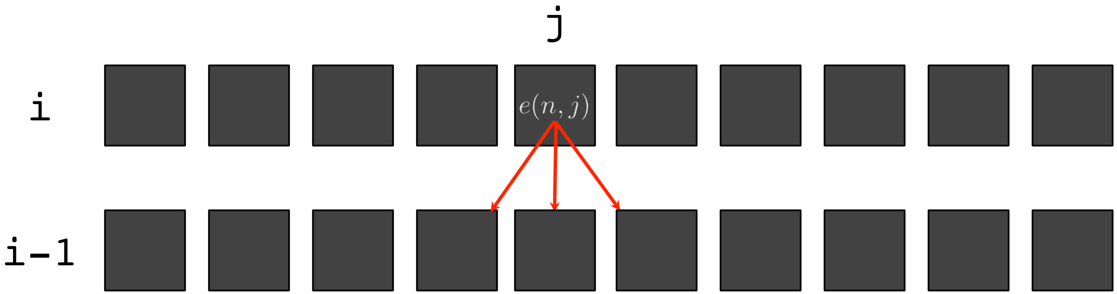

n
$e(n,1)$
$e(n,2)$
$e(n,j)$
$S_{n-1}(1)$
$S_{n-1}(2)$
$S_{n-1}(3)$
$S_{n-1}(m)$
n-1



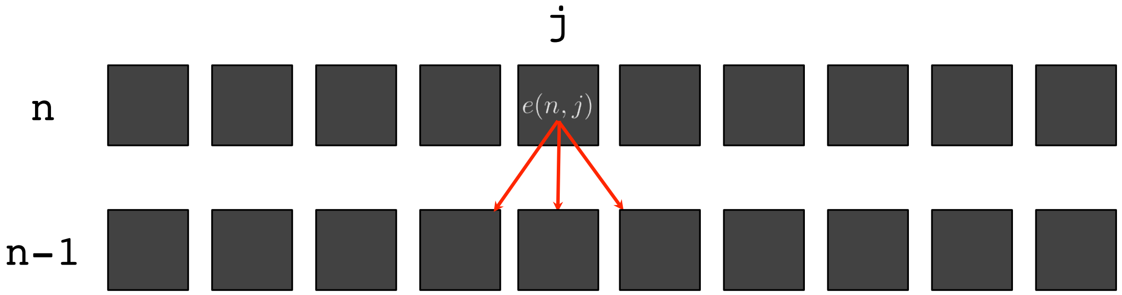
$S_n(1)$
n
$e(n,1)$
$e(n,2)$
$e(n,j)$
$S_{n-1}(1)$
$S_{n-1}(2)$
$S_{n-1}(3)$
$S_{n-1}(m)$
n-1



$$S_n(1) = e(n,1) + \min\{S_{n-1}(1), S_{n-1}(2)\}$$



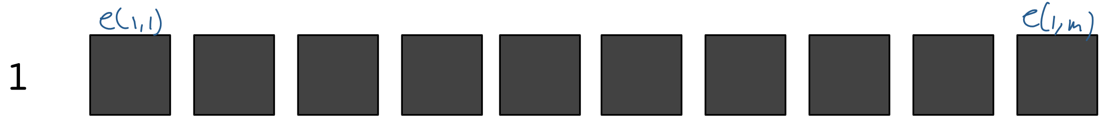
$$S_i(j) =$$



$$S_i(j) = e(i,j) + \min \begin{cases} S_{i-1}(j-1) \\ S_{i-1}(j) \\ S_{i-1}(j+1) \end{cases}$$

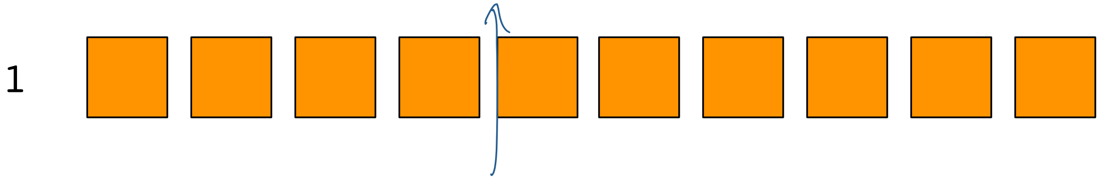
ALGORITHM

start at bottom of picture



ALGORITHM

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

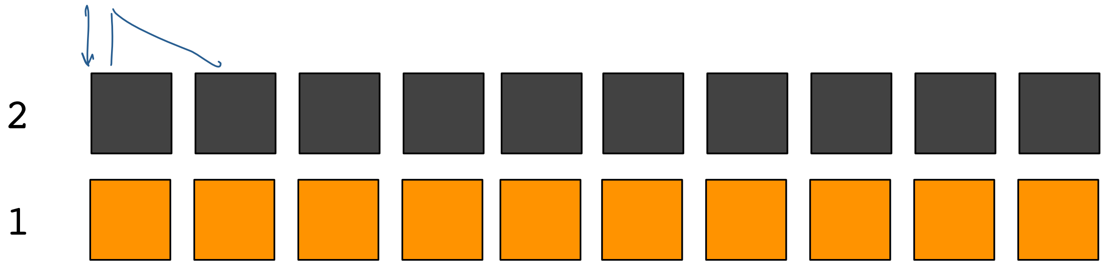


ALGORITHM

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

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

$$S_i(j) = e(i,j) + \min \begin{cases} S_{i-1}(j-1) \\ S_{i-1}(j) \\ S_{i-1}(j+1) \end{cases}$$

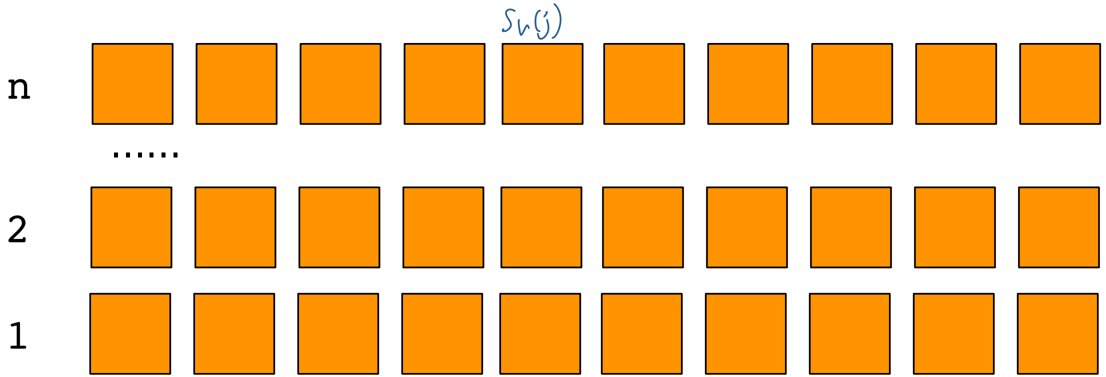


ALGORITHM

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

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

$$S_i(j) = e(i,j) + \min \begin{cases} S_{i-1}(j-1) \\ S_{i-1}(j) \\ S_{i-1}(j+1) \end{cases}$$



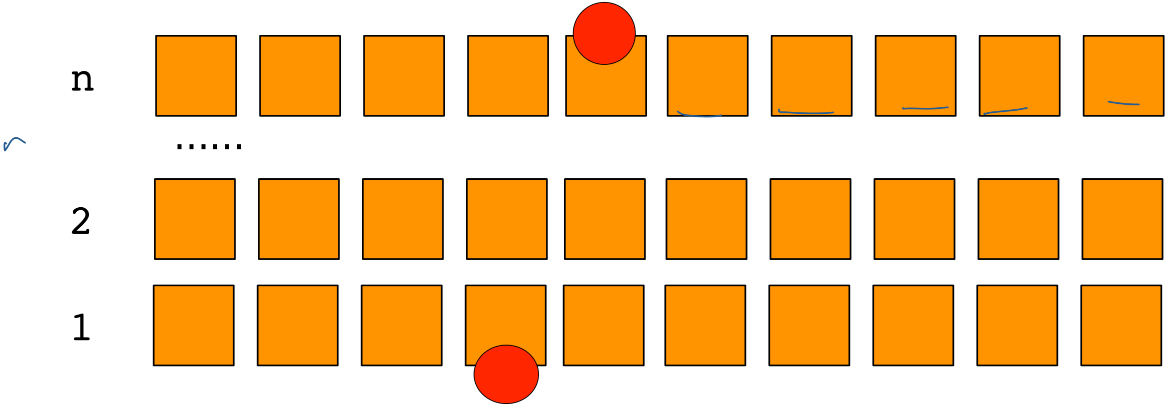
ALGORITHM

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

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

$$S_i(j) = e(i,j) + \min \begin{cases} S_{i-1}(j-1) \\ S_{i-1}(j) \\ S_{i-1}(j+1) \end{cases}$$

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 $S_{i+1}(\cdot)$

$$S_i(j) = e(i,j) + \min \begin{cases} S_{i-1}(j-1) \\ S_{i-1}(j) \\ S_{i-1}(j+1) \end{cases}$$

pick best among top row, backtrack.