How many ways can I get from KB to BG moving only up and left?

\[ B(i,j) = \text{# of ways to get to (i,j)} \]

\[ = B(i-1,j) + B(i,j-1) \]

11 moves, 6 of which are up

\[ \binom{11}{6} \]

? if CP wasn't there.
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.
do not typeset in margin
typeset every word
minimize the slack between margin and last word on a line
one paragraph at a time
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: \[ W = \{w_1, w_2, w_3, \ldots, w_n\} \]

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

such that

No line exceeds the margin

\[ C_i < M \]

\[ \min \sum (M - c_i)^2 \rightarrow \text{minimize the sum the slack on each line} \]
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 \( c_i \leq M \ \forall i \)

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

define the right variable:

Best \( n \): smallest penalty for which the first \( n \) words can be typeset.
imagine optimal solution

\[ \text{Best}_n = \text{Best}_{n-1} + \left( \text{slack}_{n-1} \right)^2 \]

\( \text{fwoll} - 1 \)

\( \text{fwoll} \) - first word of the last line.

\( \text{slack for the last line if the last line begins w/fwoll.} \)
imagine optimal solution

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

slack when line starts with \( w_\ell \), and goes to word \( n \).
imagine optimal solution

fwoll is slack when line starts with $w\ell$  

$$\operatorname{BEST}_n = \operatorname{BEST}_{n-1} + S_{\ell,n}^2$$
how many candidates are there for the fwoll?

\[ \text{Best}_n = \min_{k=1}^{n} \left\{ \text{Best}_k + (S_{kn}, n)^2 \right\} \]

\[ \text{slack when typesetting the line that begins with word } \text{w1} \text{ and ends w/word } n. \]
is \( w_1 \) well?

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

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

\( S_{2, n} = \infty \)
is $w_j$ full?
which word is fwoll?

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

\[
\text{BEST}_n = \min \begin{cases} 
0 \\
\text{BEST}_0 + S_{1,n}^2 \\
\text{BEST}_1 + S_{2,n}^2 \\
\text{BEST}_2 + S_{3,n}^2 \\
\text{...} \\
\text{BEST}_{\ell-1} + S_{\ell,n}^2 \\
\text{...} \\
\text{BEST}_{n-1} + S_{n,n}^2 
\end{cases} + \left(S_{(j-1),n}\right)^2
\]
typesetting algorithm
typesetting algorithm

make table for $S_{i,j}$

\[ \text{for } i=1 \text{ to } n \]
\[ \text{best}[i] = \min_{j=1}^{i} \{ \text{best}[j] + S[j+1][i]^2 \} \]

// 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;
}
how to compute $S_{i,j}$

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

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

\[ S_{1,1} = M - |v_i| \]

\[ S_{1,2} = S_{1,1} - 1 - |v_2| \]
Simplest case

slack when line starts with $w_1$ and ends $w_2$

$S_{1,1}$

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

Slack when line starts with $w_i$ and ends $w_j$

$$S_{i,j} = \begin{cases} M - |w_i| & \text{if } i = j \\ S_{i,j-1} - 1 - |w_j| & \text{otherwise} \end{cases}$$
How to compute $S_{i,j}$

```java
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; }
        }
    }
}
```

// 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; }
        }
    }
}
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
The first step is to make

\[ S_{i,j} \]

where

\[ S_{i,i} = M - |w_i| \]

and

\[ S_{i,j} = S_{i,j-1} - 1 - |w_j| \]

and

\[ S_{22} \] should be slacked when typesetting word 2 as word 2 on a line.
first step: make \( S_{i,j} \)

\[
\begin{array}{cccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\
\hline
1 & 40 & 36 & 32 & 27 & 24 & 17 & 14 & 10 & 6 & 0 & 99 & 99 & 99 \\
2 & & & & & & & & & & & & 39 \\
\end{array}
\]

\[
S_{i,i} = M - |w_i| \\
S_{i,j} = S_{i,j-1} - 1 - |w_j|
\]
first step: make $S_{i,j}$
The table shows the values for different positions in the matrix. The expressions $S_{2,2} = M - 3 = 39$ and $S_{3,3} = M - 3 = 39$ indicate that the values at these positions are calculated by subtracting 3 from the maximum value ($M$) in each respective row and column. The diagram illustrates the layout of the matrix with circled values highlighting the positions where the expressions are applied. The note at the bottom right corner seems to be a comment on the process of adding columns, suggesting attention to the details of setting up the problem.
second step: compute

\[
\text{Best}_1 = \text{Best}_0 + (S_{1,1})^2 = 0 + 0^2 = 0
\]
\[
\text{Best}_2 = \text{Best}_0 + (S_{1,2})^2 = 0 + 36^2 = 1296
\]
\[
\text{Best}_i = \min_{j=0}^{i-1} \left\{ \text{Best}_j + S_{j+1,i}^2 \right\}
\]

\[
\begin{array}{cccccccccccc}
  & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & \ldots \\
\hline
\text{Best} & 0 & 1600 & 1296 & 1024 & 800 & 625 & 49 & 36 & 25 & 16 & 9999999
\end{array}
\]
second step: compute

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

\[
\begin{array}{cccccccccccccccc}
1 & 40 & 36 & 32 & 27 & 24 & 17 & 14 & 10 & 6 & 0 & 99 & 99 & 99 \\
2 & 39 & 35 & 30 & 27 & 20 & 17 & 13 & 9 & 3 & 0 & 99 & 99 \\
\end{array}
\]
second step: compute

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

make table for \( S_{i,j} \)

for \( i=1 \) to \( n \)

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

\( \Theta(n^2) \)

\( \Theta(n^2) \)

\( \Theta(n^2) \)
PROBLEM: REDUCE IMAGE

scaling: distortion
deleting column: distortion
delete the most invisible seam
DEMO?

http://rsizr.com/
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?
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

\[ n \]
\[ n-1 \]

....
SMALLER PROBLEM APPROACH
IMAGINE YOU HAVE THE SOLUTION TO THE FIRST $n-1$ ROWS
\( S_{n-1}(1) \quad S_{n-1}(2) \quad S_{n-1}(3) \quad \cdots \quad S_{n-1}(m) \)
\( S_n(1) \)

\[
\begin{array}{cccc}
  e(n, 1) & e(n, 2) & & e(n, j) \\
  \hline
  S_{n-1(1)} & S_{n-1(2)} & S_{n-1(3)} & \cdots & S_{n-1(m)} \\
\end{array}
\]
\[ S_n(1) = e(n, 1) + \min\{S_{n-1}(1), S_{n-1}(2)\} \]
\[ S_i(j) = \]
\[ S_i(j) = e(i, j) + \min \left\{ S_{i-1}(j - 1), S_{i-1}(j), S_{i-1}(j + 1) \right\} \]
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 \( S_{i+1}(j) = S_i(j) + \min \left\{ \begin{array}{l} S_{i-1}(j-1) \\ S_{i-1}(j) \\ S_{i-1}(j+1) \end{array} \right\} \)
ALGORITHM

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

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

$$S_i(j) = c(i, j) + \min \left\{ \frac{S_{i-1}(j - 1)}{S_{i-1}(j)}, \frac{S_{i-1}(j)}{S_{i-1}(j + 1)} \right\}$$
ALGORITHM

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

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

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

pick best among top row, backtrack.

\begin{center}
\begin{tabular}{cccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
n & n & n & n & n & n & n & n & n & n \\
\end{tabular}
\end{center}
RUNNING TIME

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

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

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

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

\[
S_{i+1}(j) = e(i, j) + \min \left\{ S_{i-1}(j - 1), S_{i-1}(j), S_{i-1}(j + 1) \right\}
\]

pick best among top row, backtrack.
Gerrymander
Map of Charlottesville Precincts and

Instructions on Finding Your Street:
- Set the Zoom
- Find a Near exit you feel
- Zoom in Click
- Follow Fam
GERRYMANDER PROBLEM

given:

output:
GERRYMANDER PROBLEM

given: \( m \), \( A_1, A_2, \ldots, A_n \) \( n \) is even

output: \( D_1, D_2 \)

such that \( |D_1| = |D_2| \)

\[
A(D_1) > \frac{mn}{4}
\]

\[
A(D_2) > \frac{mn}{4}
\]

or “failure” if no such solution is possible
THE TECHNIQUE
GERRYMANDER

imagine very last precinct and how it is assigned:
GERRYMANDER

\[ S_{j,k,x,y} = \]
GERRYMANDER

\[ S_{j,k,x,y} = \text{there is a split of first } j \text{ precincts in which } |D_1| = k \text{ and } \]
\[ x \text{ people in } D_1 \text{ vote A and } y \text{ people in } D_2 \text{ vote A} \]
GERRYMANDER (P, A, m)

initialize array S[0,0,0,0]

\[ S_{j,k,x,y} = S_{j-1,k-1,x-A_j,y} \lor S_{j-1,k,x,y-A_j} \]
$S_{j,k,x,y} = S_{j-1,k-1,x-A_j,y} \lor S_{j-1,k,x,y-A_j}$

**GERRYMANDER(P,A,m)**

initialize array $S[0,0,0,o,o]$
for $j=1,...,n$
  for $k=1,...,n/2$
    for $x=0,...,jm$
      for $y=0,...,jm$
        fill table according to equation
        search for true entry at $S[n,n/2,>mn/4,>mn/4]$
        $S_{j,k,x,y} = S_{j-1,k-1,x-A_j,y} \lor S_{j-1,k,x,y-A_j}$
Scheduling
<table>
<thead>
<tr>
<th>Course</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>sy333</td>
<td>2</td>
<td>3.25</td>
</tr>
<tr>
<td>en162</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>ma123</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>cs4102</td>
<td>3.5</td>
<td>4.75</td>
</tr>
<tr>
<td>cs4402</td>
<td>4</td>
<td>5.25</td>
</tr>
<tr>
<td>cs6051</td>
<td>4.5</td>
<td>6</td>
</tr>
<tr>
<td>sy333</td>
<td>5</td>
<td>6.5</td>
</tr>
<tr>
<td>cs1011</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>
problem statement

\[(a_1, \ldots, a_n)\]
\[(s_1, s_2, \ldots, s_n)\]
\[(f_1, f_2, \ldots, f_n)\] (sorted) \( s_i < f_i \)

(compatible)

find largest subset of activities \( C=\{a_i\} \) such that
problem statement

\[(a_1, \ldots, a_n)\]
\[(s_1, s_2, \ldots, s_n)\]  
\[(f_1, f_2, \ldots, f_n) \text{ (sorted)} \quad s_i < f_i\]  
\[(\text{compatible})\]

find largest subset of activities \(C = \{a_i\}\) such that

\[a_i, a_j \in C, i < j\]
\[f_i \leq s_j\]
problem statement

\((a_1, \ldots, a_n)\)

\((s_1, s_2, \ldots, s_n)\)

\((f_1, f_2, \ldots, f_n)\) \(\forall i \in \mathbb{R}^n. \quad s_i < f_i\)
dynamic programming
Dynamic programming

\[ \text{BEST}_{f_n} = \max \text{BEST}_{s_n} + 1 \quad a_n \text{ in:} \]

\[ \text{BEST}_{e_t} \quad a_n \text{ out:} \]
greedy solution:
greedy solution:

\( s_1 \rightarrow f_1 \rightarrow f_2 \)

\[ \text{SOLT}N_{i,j} \]

\[ \text{SOLT}N_{0,2n} \]