Applied Algorithms

Lecture 4: Hirshberg's Algorithm

UPDATED 2/26

Admin

- Office hours Tuesday are now 3-4
- THIS week they are 3-5 Tuesday
- Wednesday office hours THIS week are 10:30-2:30
- No class Thursday   (I'm at a review panel Thursday-Friday)
- Assignment 1 is due Saturday
  - I'll be around Saturday if you need help

Some simple efficiency principles

Time vs Space themed

Cost of operations

- Adding? Multiplying? Floats? Ints?
- Dividing? Modulo?

Touching memory

Notes on how this works

- Allocation itself is essentially O(1)
- Writing to *lots of places* in memory is expensive
- How expensive is it?
  - Let's say we do a modulo, and an if, and a memory store (but in a small number of places)
  - Which is more expensive?

Edit Distance

Problem

- Given two strings A, B
- Edit: insert, delete, replace (each costs 1)
- What is the minimum number of edits to get from A to B?

Example

OCURRANCE vs OCCURRENCE

OCURRANCE

Insert C here

OCCURRANCE

OCCURRENCE

Replace A with E

Algorithm: Dynamic Programming

- How can you build up edit distance recursively?

Base case:

If X has length 0, what is the edit distance between X and a string Y?

Recursion: characters match

If the last characters of X and Y match, what is the edit distance between X and Y?

OCCURRA**N**

OCCURRE**N**

Recursion: characters don’t match

If the last characters of X and Y do not match, what is the edit distance between X and Y?

OCCURR**A**

OCCURR**E**

Dynamic programming

- Entry (i, j) in the table is the edit distance between the first i characters of X and the first j characters of Y

Dynamic programming: example

		O	C	C	U	R	R	E	N	C	E
	0	1	2	3	4	5	6	7	8	9	10
O	1	0	1	2	3	4	5	6	7	8	9
C	2	1	0	1	2	3	4	5	6	7	8
U	3	2	1	1	1	2	3	4	5	6	7
R	4	3	2	2	2	1	2	3	4	5	6
R	5	4	3	3	3	2	1	2	3	4	5
A	6	5	4	4	4	3	2	2	3	4	5
N	7	6	5	5	5	4	3	3	2	3	4
C	8	7	6	6	6	5	4	4	3	2	3
E	9	8	7	7	7	6	5	4	4	3	2

Analysis

Two strings of length m and n

- Time: $O(mn)$
- Space: $O(mn)$

Can this be improved?

Can this be improved?

Time: No. (Probably not)

Edit Distance Cannot Be Computed in Strongly Subquadratic Time (unless SETH is false)

Arturs Backurs
MIT
backurs@mit.edu

Piotr Indyk
MIT
indyk@mit.edu

ABSTRACT

The edit distance (a.k.a. the Levenshtein distance) between two strings is defined as the minimum number of insertions, deletions or substitutions of symbols needed to transform one string into another. The problem of computing the ... with many applications in computational biology, natural language processing and information theory. The problem of computing the edit distance between two strings is a classical computational task, with a well-known algorithm based on dynamic programming. Unfortunately, that algorithm runs in quadratic time, which is prohibitive for long sequences

Can this be improved?

Space?

Question: how do you fill out the dynamic programming table?

Dynamic programming: example

		O	C	C	U	R	R	E	N	C	E
	0	1	2	3	4	5	6	7	8	9	10
O	1	0	1	2	3	4	5	6	7	8	9
C	2	1	0	1	2	3	4	5	6	7	8
U	3	2	1	1	1	2	3	4	5	6	7
R	4	3	2	2	2	1	2	3	4	5	6
R	5	4	3	3	3	2	1	2	3	4	5
A	6	5	4	4	4	3	2	2	3	4	5
N	7	6	5	5	5	4	3	3	2	3	4
C	8	7	6	6	6	5	4	4	3	2	3
E	9	8	7	7	7	6	5	4	4	3	2

Can this be improved?

		O	C	C	U	R	R	E	N	C	E
O											
C											
U											
R											
R	4	3	2	2	2	1	2	3	4	5	6
A	5										
N											
C											
E											

Can this be improved?

Space?

- Need only keep two lines in memory. Space $O(\min\{n, m\})$

Time?

- Same. (Do we lose any constants in terms of operations?)
- This generally significantly improves running time in practice

Recovering the edits

How can we figure out the actual inserts, deletes, etc. to get from one string to the other?

Recovering the edits

		O	C	C	U	R	R	E	N	C	E
	0	1	2	3	4	5	6	7	8	9	10
O	1	0	1	2	3	4	5	6	7	8	9
C	2	1	0	1	2	3	4	5	6	7	8
U	3	2	1	1	1	2	3	4	5	6	7
R	4	3	2	2	2	1	2	3	4	5	6
R	5	4	3	3	3	2	1	2	3	4	5
A	6	5	4	4	4	3	2	2	3	4	5
N	7	6	5	5	5	4	3	3	2	3	4
C	8	7	6	6	6	5	4	4	3	2	3
E	9	8	7	7	7	6	5	4	4	3	2

Recovering the edits

		O	C	C	U	R	R	E	N	C	E
	0	1	2	3	4	5	6	7	8	9	10
O	1	0	1	2	3	4	5	6	7	8	9
C	2	1	0	1	2	3	4	5	6	7	8
U	3	2	1	1	1	2	3	4	5	6	7
R	4	3	2	2	2	1	2	3	4	5	6
R	5	4	3	3	3	2	1	2	3	4	5
A	6	5	4	4	4	3	2	2	3	4	5
N	7	6	5	5	5	4	3	3	2	3	4
C	8	7	6	6	6	5	4	4	3	2	3
E	9	8	7	7	7	6	5	4	4	3	2

Recovering the edits

		O	C	C	U	R	R	E	N	C	E
	0	1	2	3	4	5	6	7	8	9	10
O	1	0	1	2	3	4	5	6	7	8	9
C	2	1	0	1	2	3	4	5	6	7	8
U	3	2	1	1	1	2	3	4	5	6	7
R	4	3	2	2	2	1	2	3	4	5	6
R	5	4	3	3	3	2	1	2	3	4	5
A	6	5	4	4	4	3	2	2	3	4	5
N	7	6	5	5	5	4	3	3	2	3	4
C	8	7	6	6	6	5	4	4	3	2	3
E	9	8	7	7	7	6	5	4	4	3	2

Path gives you the edits

Match
Match
Insert
Match
Match
Match
Replace
Match
Match
Match

		O	C	C	U	R	R	E	N	C	E
	0	1	2	3	4	5	6	7	8	9	10
O	1	0	1	2	3	4	5	6	7	8	9
C	2	1	0	1	2	3	4	5	6	7	8
U	3	2	1	1	1	2	3	4	5	6	7
R	4	3	2	2	2	1	2	3	4	5	6
R	5	4	3	3	3	2	1	2	3	4	5
A	6	5	4	4	4	3	2	2	3	4	5
N	7	6	5	5	5	4	3	3	2	3	4
C	8	7	6	6	6	5	4	4	3	2	3
E	9	8	7	7	7	6	5	4	4	3	2

Recovering the edits

- This takes lots of space!
  - Which is inefficient
- Can we get the best of both worlds—linear space as well as recovering the edits?

Recovering just one edit

- Let's say I want just one piece: what is the (rightmost) square in the middle row on the solution path?

- Can I do this in $O(\min\{n, m\})$ space?

Recovering the edits

		O	C	C	U	R	R	E	N	C	E
	0	1	2	3	4	5	6	7	8	9	10
O	1	0	1	2	3	4	5	6	7	8	9
C	2	1	0	1	2	3	4	5	6	7	8
U	3	2	1	1	1	2	3	4	5	6	7
R	4	3	2	2	2	1	2	3	4	5	6
R	5	4	3	3	3	2	1	2	3	4	5
A	6	5	4	4	4	3	2	2	3	4	5
N	7	6	5	5	5	4	3	3	2	3	4
C	8	7	6	5	6	5	4	4	3	2	3
E	9	8	7	6	6	6	5	4	4	3	2

Recovering the edits

		O	C	C	U	R
	0	1	2	3	4	5
O	1	0	1	2	3	4
C	2	1	0	1	2	3
U	3	2	1	1	1	2
R	4	3	2	2	2	1

Recovering the edits

The entry we are looking for $(i, n/2)$ is the one that minimizes:

(Edit distance from first $n/2$ characters of string 1 to first i characters of string 2) + (Edit distance from last $n - n/2$ characters of string 1 to last $m - i$ characters of string 2)

How efficiently can we get these?

- Edit distance from $(0,0)$ to $(i, n/2)$ for all i:
  - $O(nm)$ time, $O(n+m)$ space

0	1	2	3	4	5	6	7	8	9	10
1	0	1	2	3	4	5	6	7	8	9

How efficiently can we get these?

- Edit distance from $(0,0)$ to $(i, n/2)$ for all i:
  - $O(nm)$ time, $O(n+m)$ space

1	0	1	2	3	4	5	6	7	8	9
2	1	0	1	2	3	4	5	6	7	8

How efficiently can we get these?

- Edit distance from $(0{,}0)$ to $(i, n/2)$ for all i:
  - $O(nm)$ time, $O(n+m)$ space

2	1	0	1	2	3	4	5	6	7	8
3	2	1	1	1	2	3	4	5	6	7

How efficiently can we get these?

- Edit distance from $(0,0)$ to $(i, n/2)$ for all i:
  - $O(nm)$ time, $O(n+m)$ space

3	2	1	1	1	2	3	4	5	6	7
4	3	2	2	2	1	2	3	4	5	6

How efficiently can we get these?

- Edit distance from $(0,0)$ to $(i, n/2)$ for all i:
  - $O(nm)$ time, $O(n+m)$ space

4	3	2	2	2	1	2	3	4	5	6

How efficiently can we get these?

- How can we get edit distance from $(i, n/2)$ to (n, m) for all i?
  - (Be careful about off-by-one! Remember that this should be testing the cost of matching the "rest" of string 1 to the "rest" of string 2)
- Issue: we don't want to start over from every starting point
- Idea: run it backwards!
- (edit distance stays the same if we reverse both strings)

Recovering the edits

		O	C	C	U	R	R	E	N	C	E
O											
C											
U											
R	4	3	2	2	2	1	2	3	4	5	6
R											
A											
N											
C											
E											

Recovering the edits

		E	C	N	E	R	R	U	C	C	O
	0	1	2	3	4	5	6	7	8	9	10
E	1	0	1	2	3	4	5	6	7	8	9
C											
N											
A											
R											

Recovering the edits

		E	C	N	E	R	R	U	C	C	O
E	1	0	1	2	3	4	5	6	7	8	9
C	2	1	0	1	2	3	4	5	6	7	8
N											
A											
R											

Recovering the edits

E C N E R R U C C O

E
C
N
A
R

2	1	0	1	2	3	4	5	6	7	8
3	2	1	0	1	2	3	4	5	6	7

Recovering the edits

		E	C	N	E	R	R	U	C	C	O
E											
C											
N	3	2	1	0	1	2	3	4	5	6	7
A	4	3	2	1	1	2	3	4	5	6	7
R											

Recovering the edits

		E	C	N	E	R	R	U	C	C	O
E											
C											
N											
A											
R	5	4	3	2	2	1	2	3	4	5	6

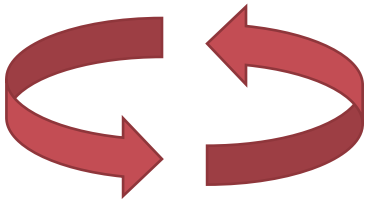
Recovering the edits: Putting it together

4	3	2	2	2	1	2	3	4	5	6

Cost of second half of reversed X to reversed Y

Cost from (0,0) to (i,n/2) for all i

5	4	3	2	2	1	2	3	4	5	6



Recovering the edits: Putting it together

4	3	2	2	2	1	2	3	4	5	6

Cost from (0,0) to (i,n/2) for all i

Cost from (i,n/2) to (m,n) for all i

6	5	4	3	2	1	2	2	3	4	5

Recovering the edits: Putting it together

4	3	2	2	2	1	2	3	4	5	6

+

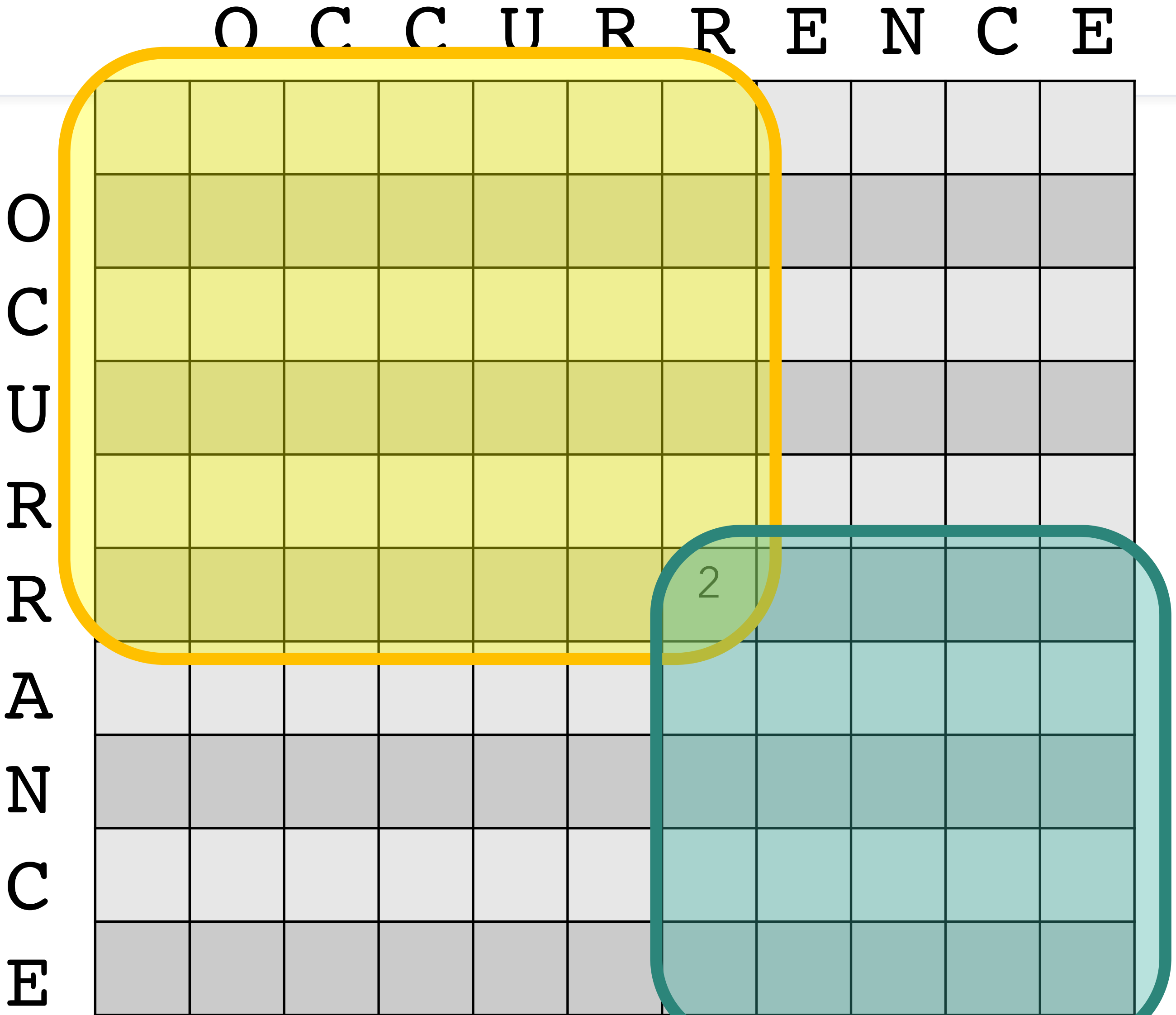
6	5	4	3	2	1	2	2	3	4	5

=

10	8	6	5	4	2	4	5	7	9	11

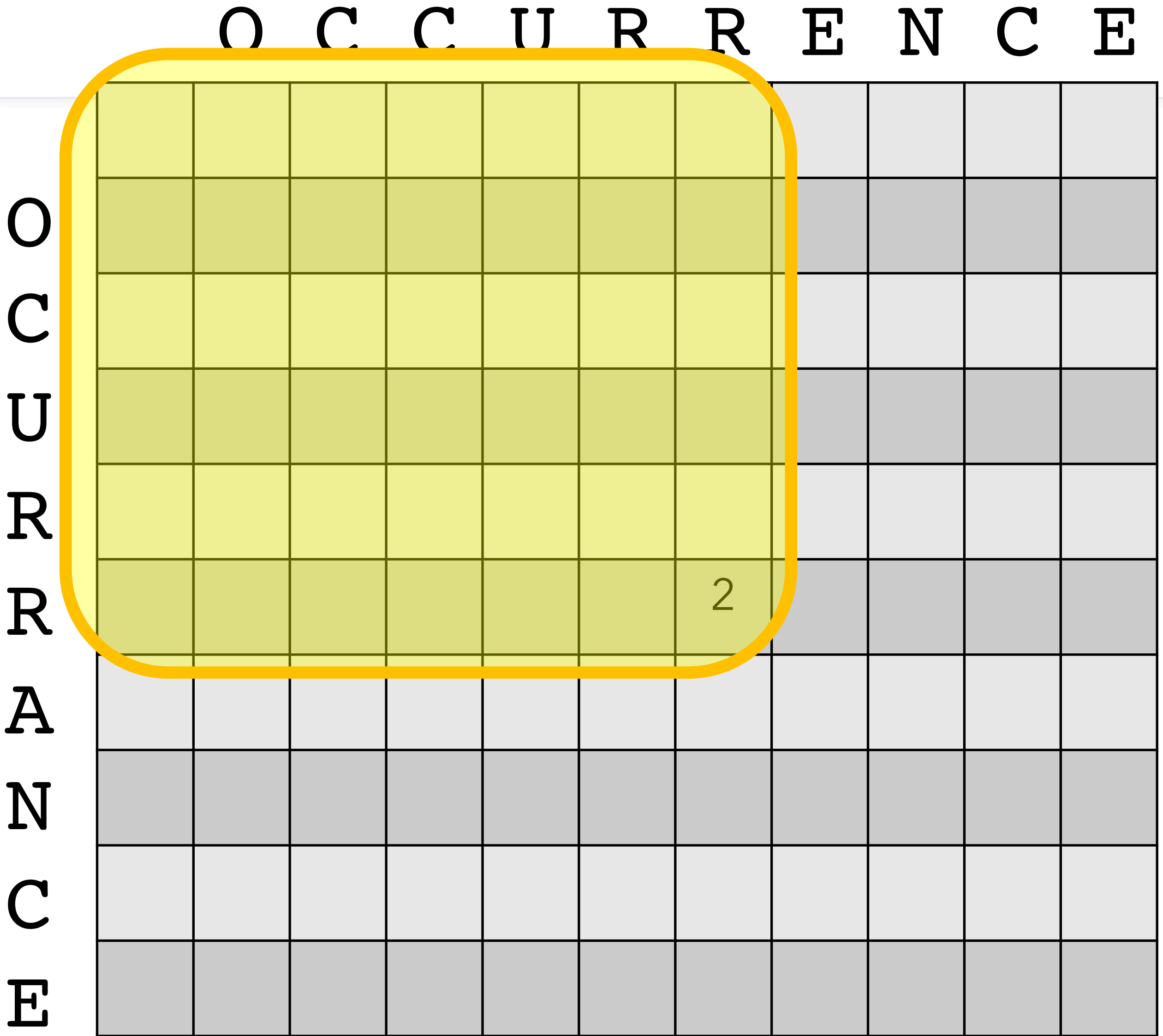
Careful–need to line up arrays so that we never reuse a character of String 2
(They line up exactly here)

Recovering the edits



- We know the optimal path goes through this entry
- Now what?
- Recurse!!
  - Where?
- Why don't we need to look at the rest of the table?

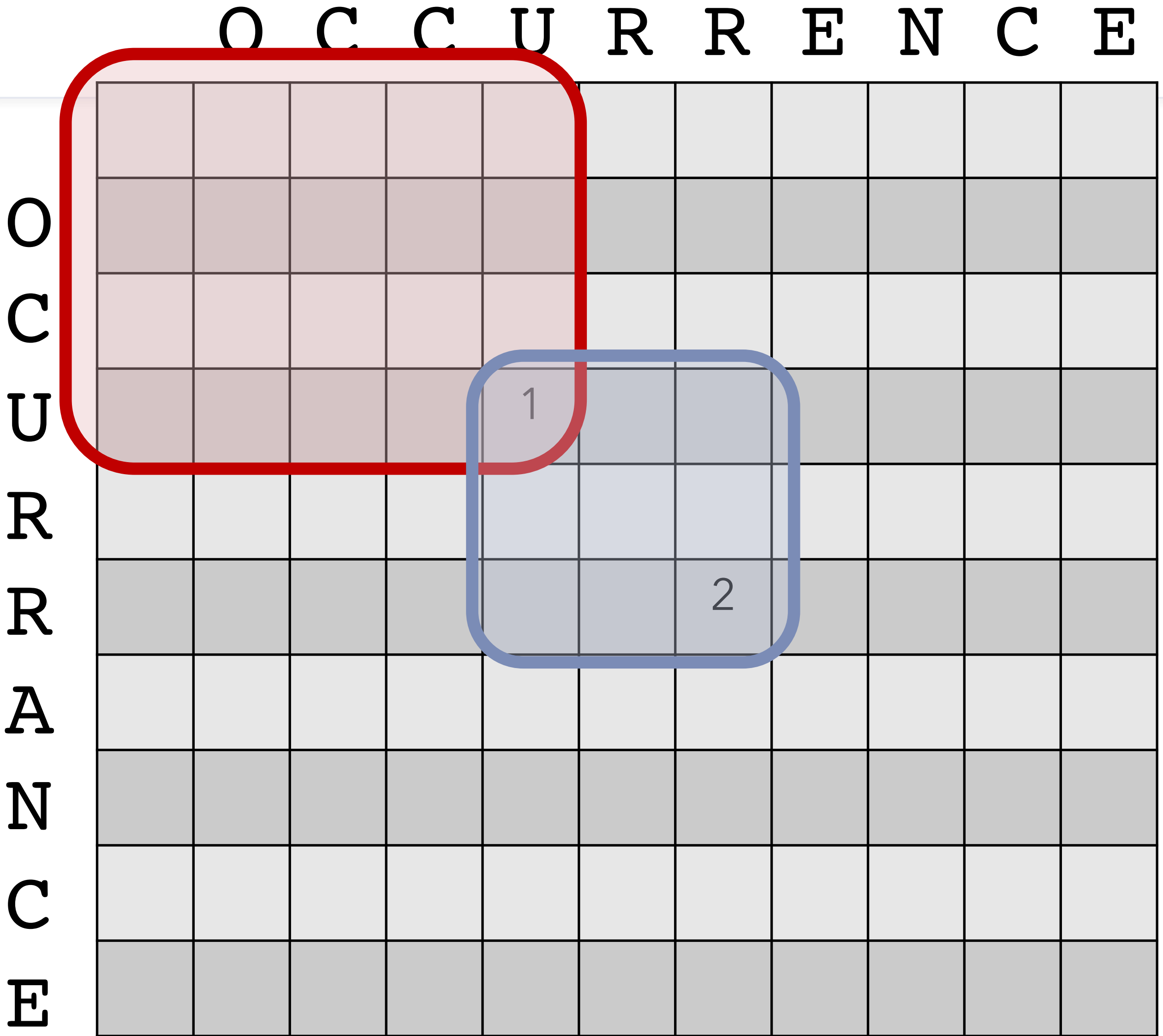
Recovering the edits



Recovering the edits

		O	C	C	U	R	R	E	N	C	E
O											
C											
U					1						
R											
R							2				
A											
N											
C											
E											

Recovering the edits



And so on, until we have the entire path

Recovering the edits

- If one string is empty, return trivial edits
- Otherwise:
  - Run space-efficient edit distance on top-right
  - Run space-efficient edit distance backwards on bottom-left
  - Lowest total edit distace is where the path goes through
  - Recurse to find remaining path of top-left and bottom right

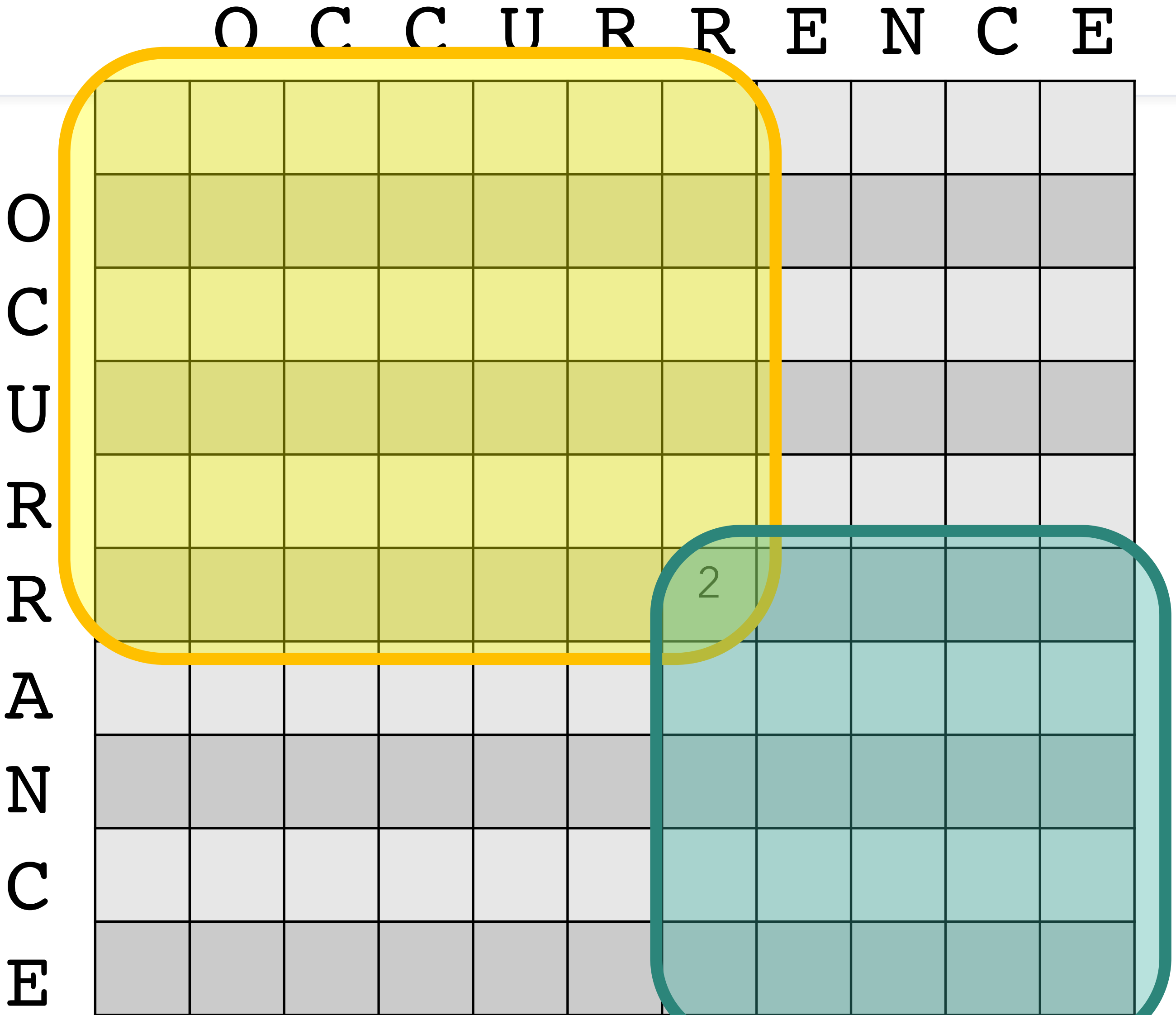
Space?

- $O(\min\{n, m\})$
  - Space-efficient edit distance algorithm
  - Keep track of the current path

Time?

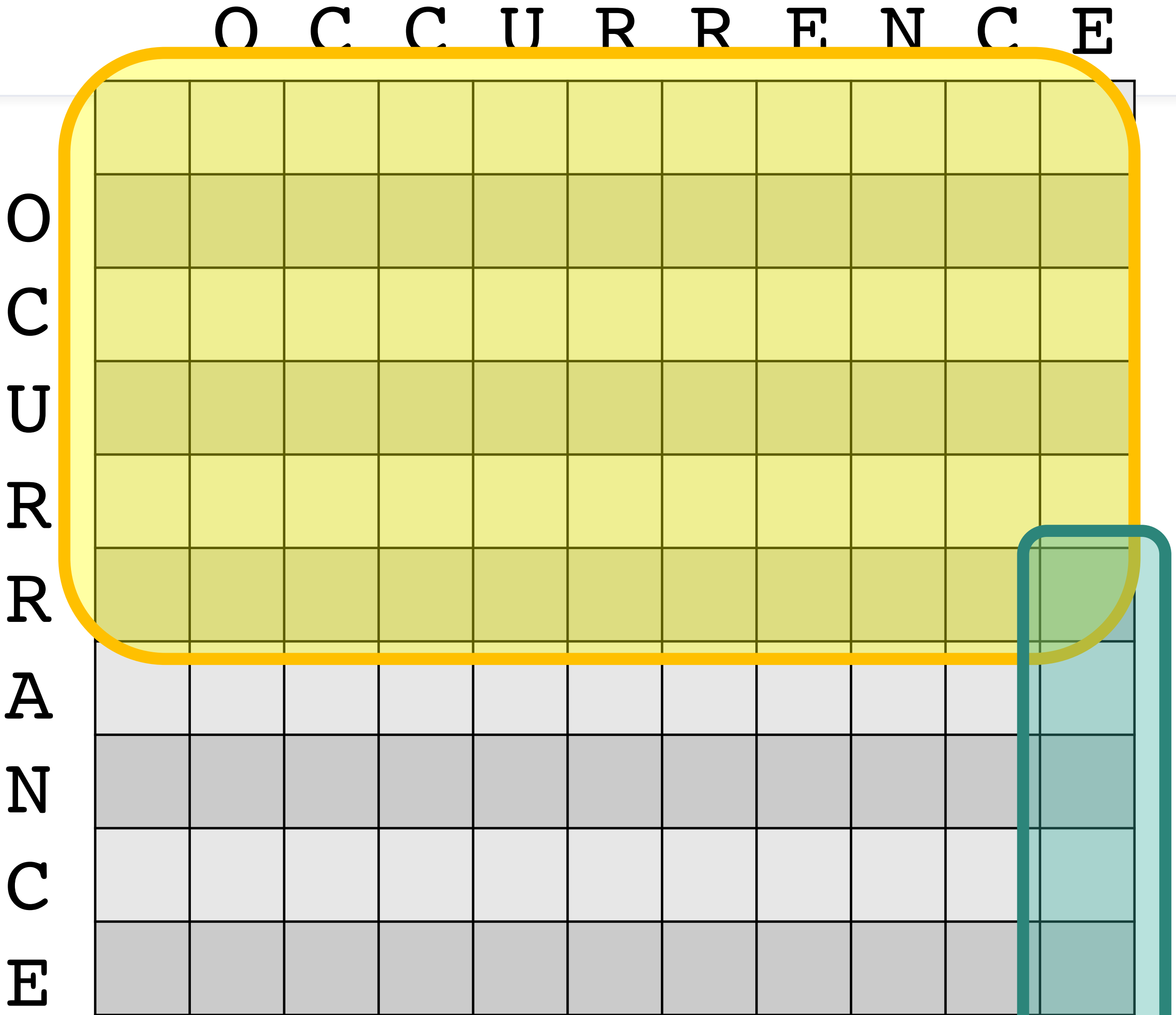
- Intuitively, what do you think it is?
  - Linear (in the table size) for each value of the path obtained
  - But, it’s getting smaller
  - How much smaller does the table get every time?

Recovering the edits



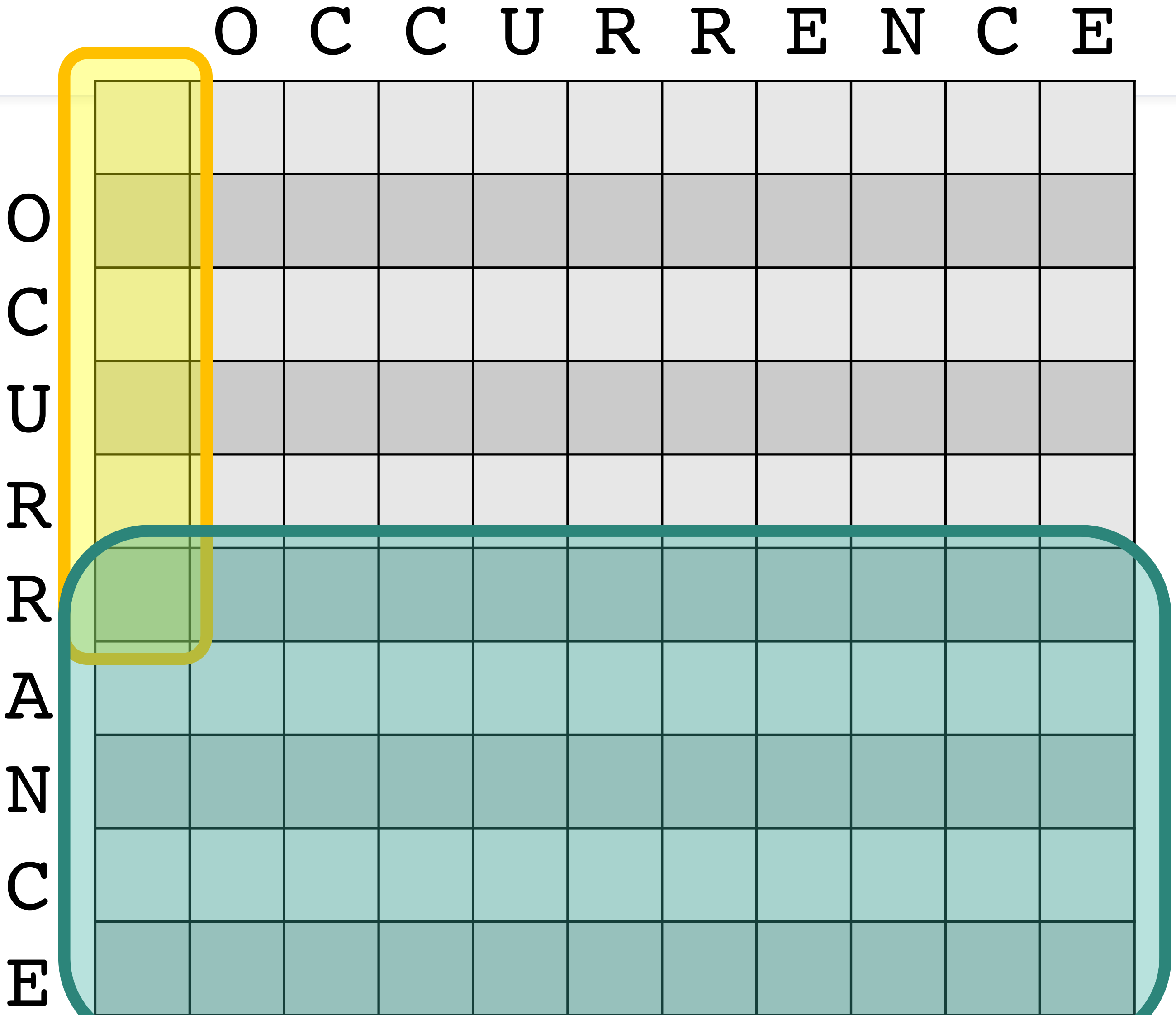
How much smaller does the table get every time?

Recovering the edits



How much smaller does the table get every time?

Recovering the edits



How much smaller does the table get every time?

Time: recurrence

- Let's say the path goes through element k
- $T(n, m) = T\left(\frac{n}{2}, k\right) + T\left(\frac{n}{2}, m - k\right) + O(nm)$

Solving the recurrence

- $T(n,m) = T\left(\frac{n}{2}, k\right) + T\left(\frac{n}{2}, m-k\right) + c_2 nm$
- Assume that $T(n,m) \leq c_1 nm$, let $c_1 = 2c_2$
- Key part of inductive proof:
- $c_1 nm \leq \frac{c_1 nk}{2} + \frac{c_1 n(m-k)}{2} + c_2 nm$
- $c_1 nm \leq \frac{c_1 nk}{2} + \frac{c_1 n(m-k)}{2} + \frac{c_1 nm}{2} = c_1 nm$

Is this actually good?

- Space efficiency is linear instead of quadratic
- Is the time higher?
  - Asymptotics: no
  - Constants? Absolutely

- Is the tradeoff worth it?
  - You'll find out in Assignment 2.
  - (It probably is)