9 February 2024 Edit Distance

Plan

* Edit Distance

* Announcements

* ED Dynamic Program
  - ↳ Recurrence
  - ↳ Reconstructing the sequence of edits.
  - ↳ Space Complexity?

Genomics

* Genome Sequences $S \in \{A, C, G, T\}^*$

$$S_{cat} = A\ C\ C\ G\ A\ T\ C\ G\ A\ T\ \dots$$

$$S_{dog} = A\ C\ G\ G\ A\ A\ T\ C\ G\ G\ T\ \dots$$

Genomics

* Genome Sequences $S \in \{A, C, G, T\}^*$

$$S_{cat} = A\ C\ C\ G\ A\ T\ C\ G\ A\ T \ldots$$

$$S_{dog} = A\ C\ G\ G\ A\ A\ T\ C\ G\ G\ T \ldots$$

How related are two species?

↳ How "similar" are their genomes?

Genomics

* Genome Sequences $S \in \{A, C, G, T\}^*$

$$S_{cat} = A\ C\ C\ G\ A\ T\ C\ G\ A\ T \ldots$$

$$S_{dog} = A\ C\ G\ G\ A\ A\ T\ C\ G\ G\ T \ldots$$

How related are two species?

$\hookrightarrow$ How "similar" are their genomes?

BLAST: Basic Local Alignment Search Tool $\quad > 110$ K citations

Edit Distance

How easily can we edit string S into string t?

Edit Distance

How easily can we edit string S into string t?

Allowed Edits

* Insertion / Deletion
* Changes

~~E~~DIT → DIST

M~~IN~~IMUM → MAXIMUM

~~S~~NO~~W~~ → NO

Edit Distance

How easily can we edit string S into string t?

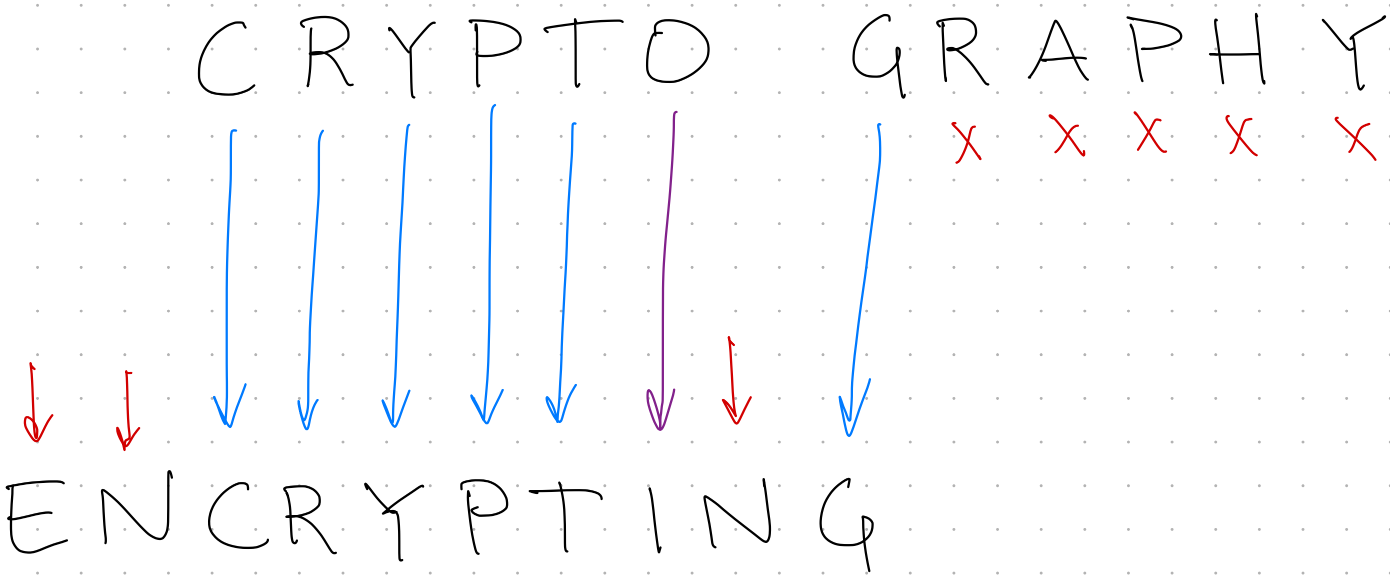
CRYPTOGRAPHY

?

ENCRYPTING

Edit Distance

How easily can we edit string s into string t?



3 insertions

5 deletion

1 change

O → I

Given two strings S and T

Compute minimum cost edits from $S \to T$

insertion / deletion : γ

"changing" a to b : Δ_{ab}

S AGGCTAATC ...

T GAGCTAAGCC ...

Changing a to a costs $\Delta_{aa} = 0$.

Announcements

* HW 2 due last night. Late deadline Sunday.
* HW 3 out after lecture.
* Prelim Tues Feb 20 7:30 pm
  → Last call to schedule a make-up
  See Ed post from Sara Perkins

Given two strings S and T

Compute minimum cost edits from $S \to T$

insertion / deletion : γ

changing a to b : Δ_{ab}

S | AGGCTAATC ...

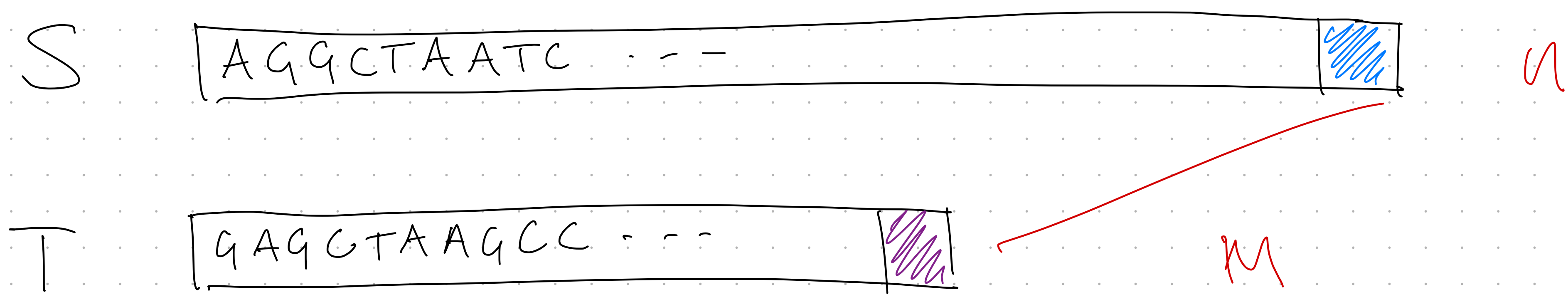
T | GAGCTAAGCC ...

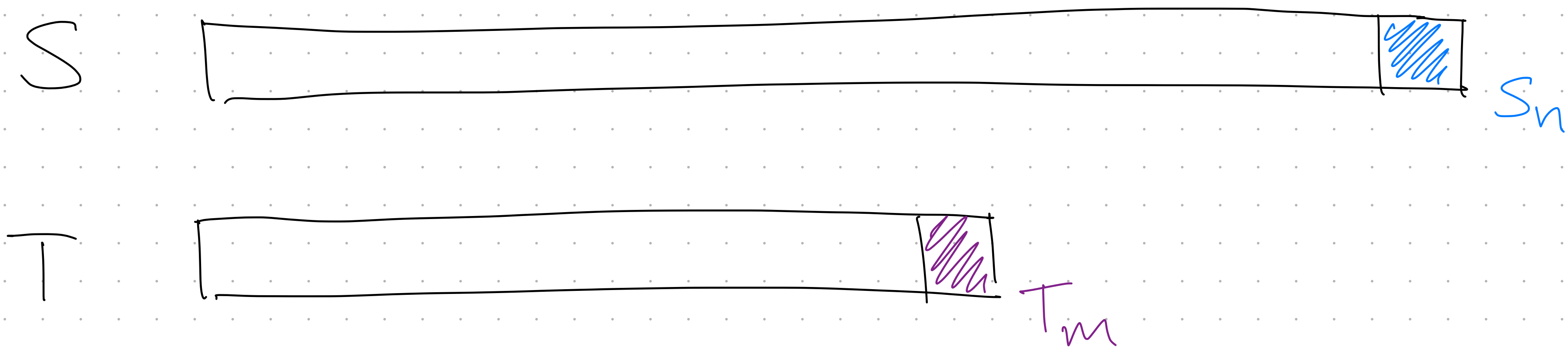
Given two strings S and T

Compute minimum cost edits from $S \to T$

insertion / deletion : γ

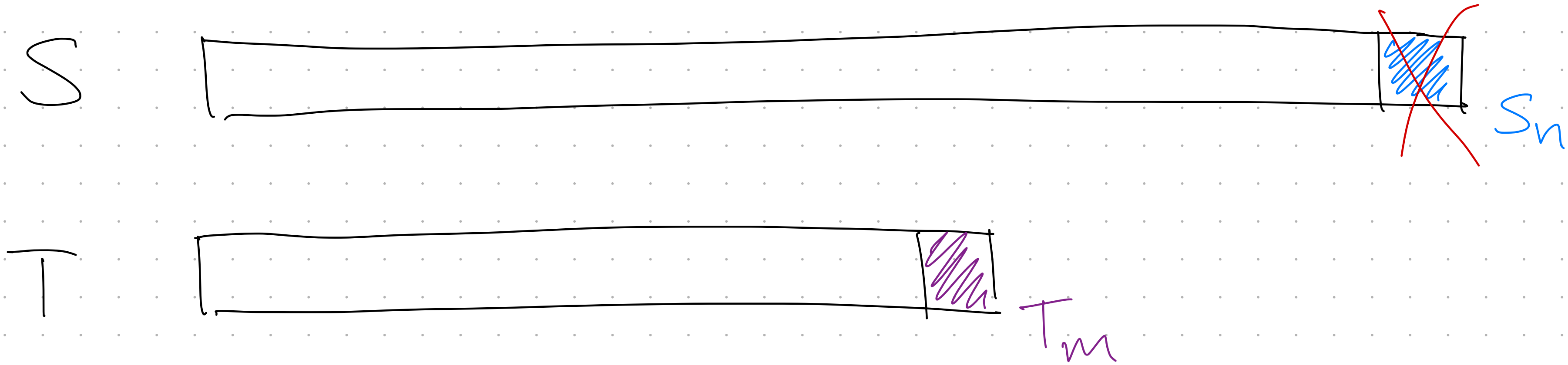
changing a to b : Δ_{ab}





Fact. Fix an optimal set of edits from S to T.

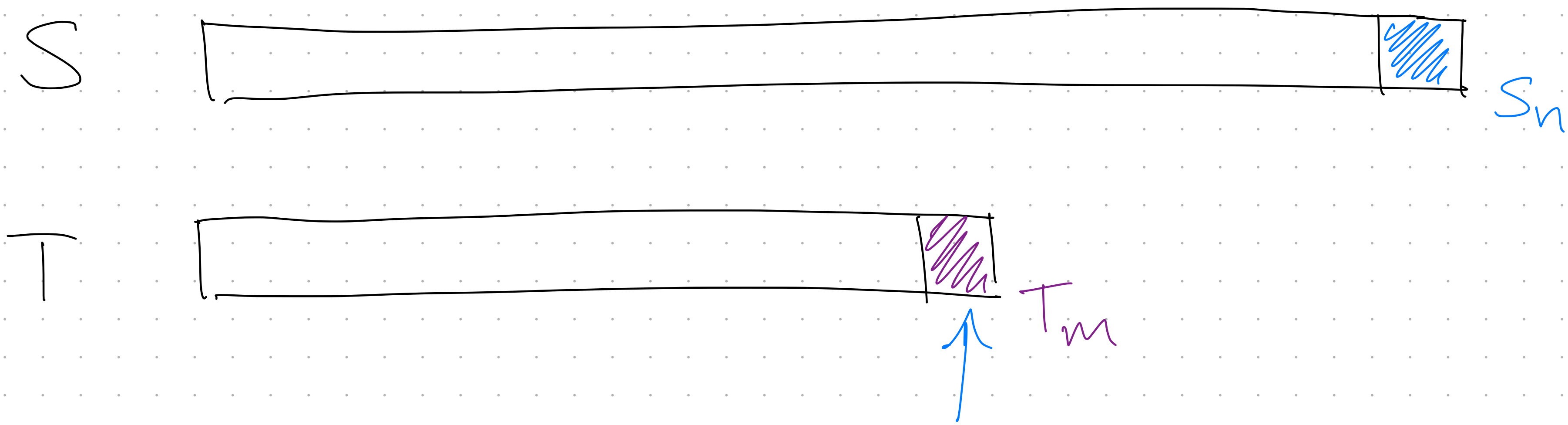
One of the following is true:



Fact. Fix an optimal set of edits from S to T.

One of the following is true:

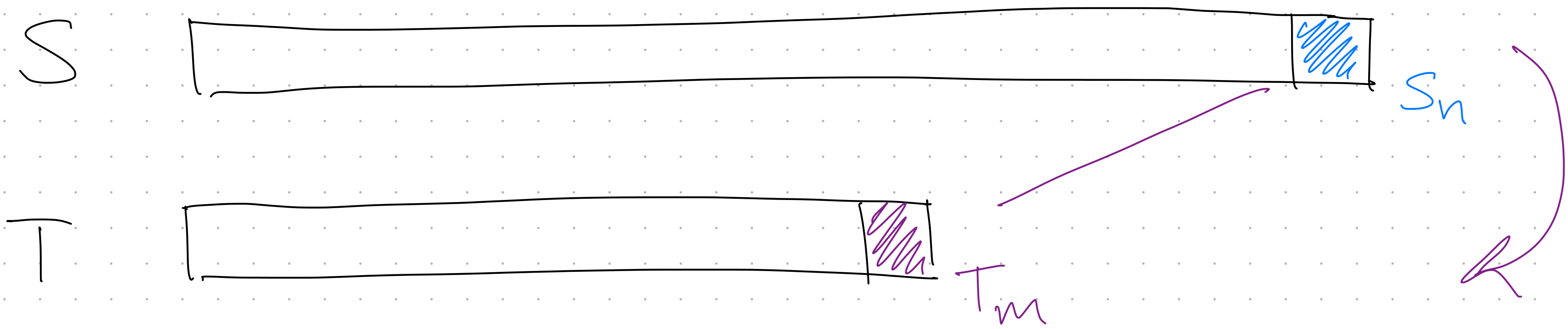
* S_n is deleted at cost δ



<u>Fact</u>. Fix an optimal set of edits from S to T.

One of the following is true:

* S_n is deleted at cost γ
* T_m is inserted at cost γ



Fact. Fix an optimal set of edits from S to T.

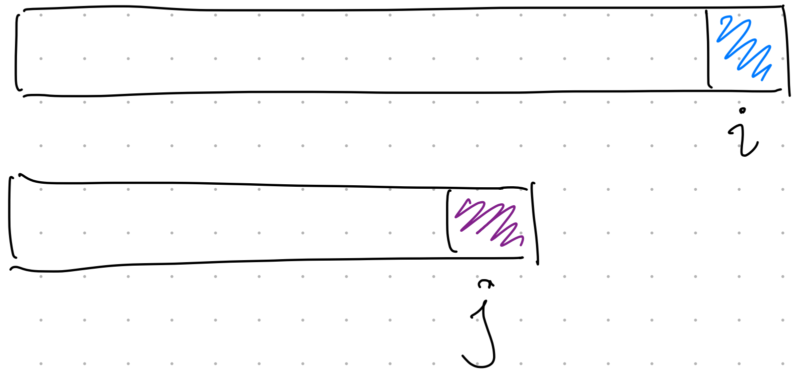
One of the following is true:

* S_n is deleted at cost γ
* T_m is inserted at cost γ
* S_n "changes" to T_m at cost $\Delta_{S_n T_m}$

The Edit Distance Recurrence

$S[1:i]$

$T[1:j]$



$ED(i, j)$

↓

Edit distance between prefixes $S[1:i]$ and $T[1:j]$

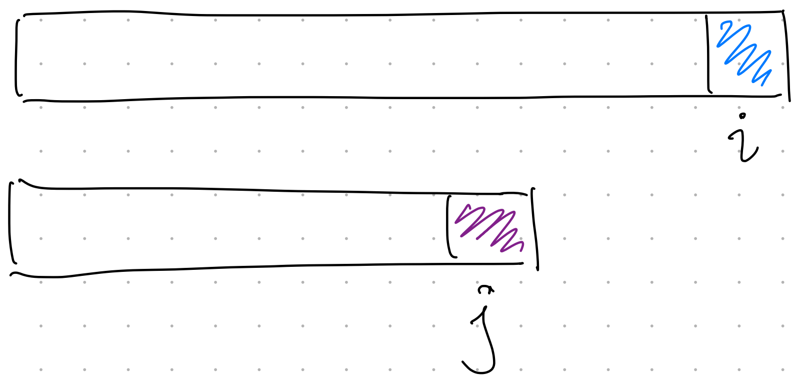
Possibilities

* S_i changes to T_j
* S_i deleted
* T_j inserted

The Edit Distance Recurrence

$S[1:i]$

$T[1:j]$



$ED(i,j)$ minimizes

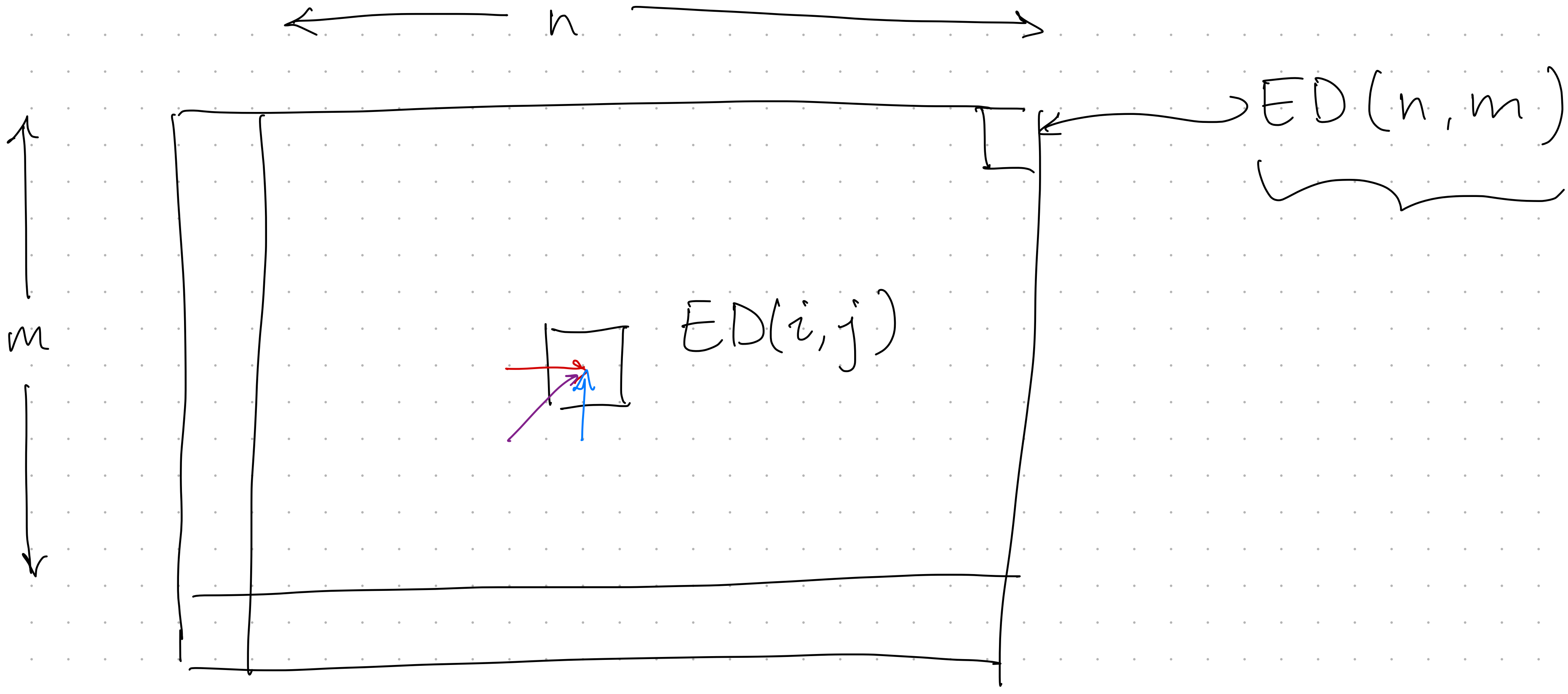
Possibilities

* S_i changes to T_j $\longrightarrow$ $ED(i-1, j-1) + \Delta_{S_i T_j}$
* S_i deleted $\longrightarrow$ $ED(i-1, j) + \gamma$
* T_j inserted $\longrightarrow$ $ED(i, j-1) + \gamma$

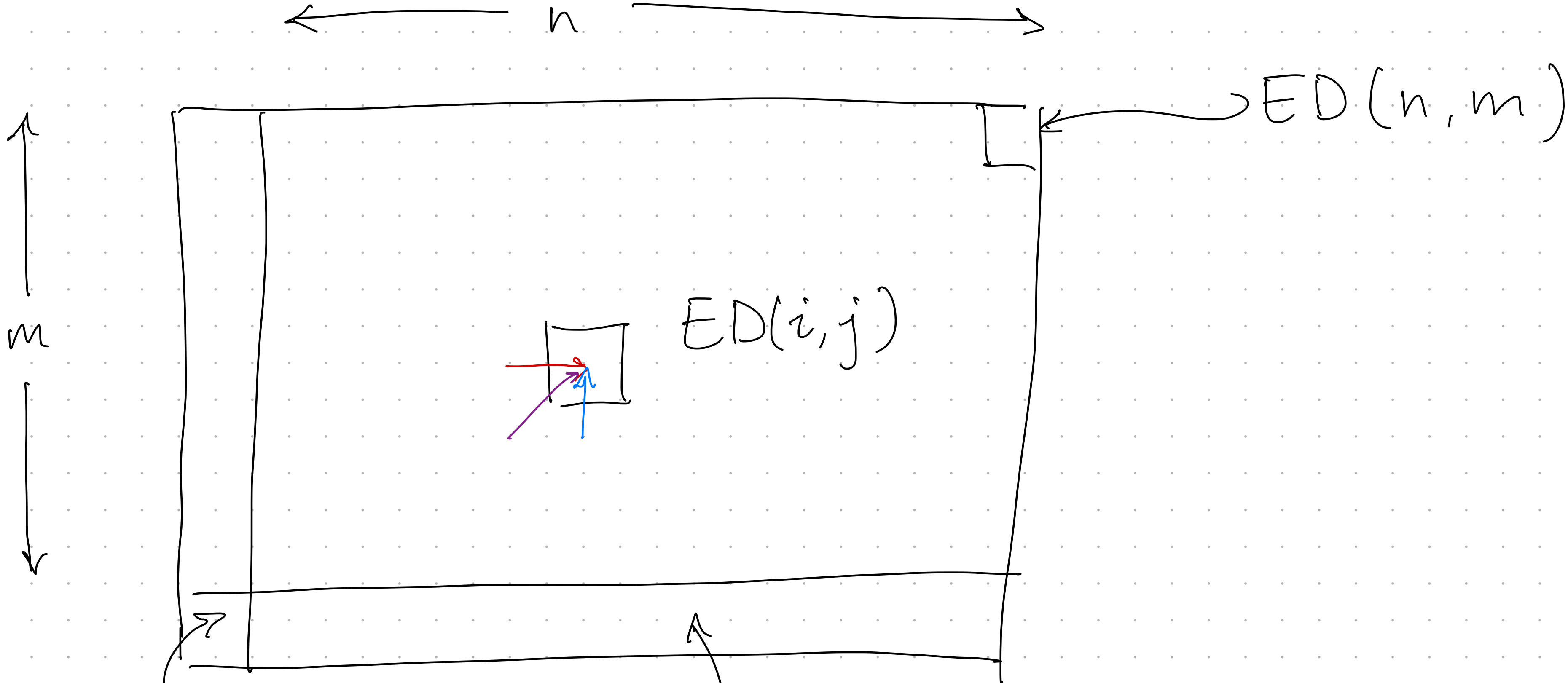
$$ED(i,j) = \min \left\{ \begin{array}{l} ED(i-1, j-1) + \Delta_{s_i, T_j}, \\ ED(i-1, j) + \gamma, \\ ED(i, j-1) + \gamma \end{array} \right\}$$

Dynamic Programming Table?

Base Cases?



$$ED(i,j) = \min \begin{cases} ED(i-1,j-1) + \Delta_{s_i T_j}, \\ ED(i-1,j) + \gamma, \\ ED(i,j-1) + \gamma \end{cases}$$



$ED(0,0) = 0$

$ED(i,0) = i \cdot \gamma$

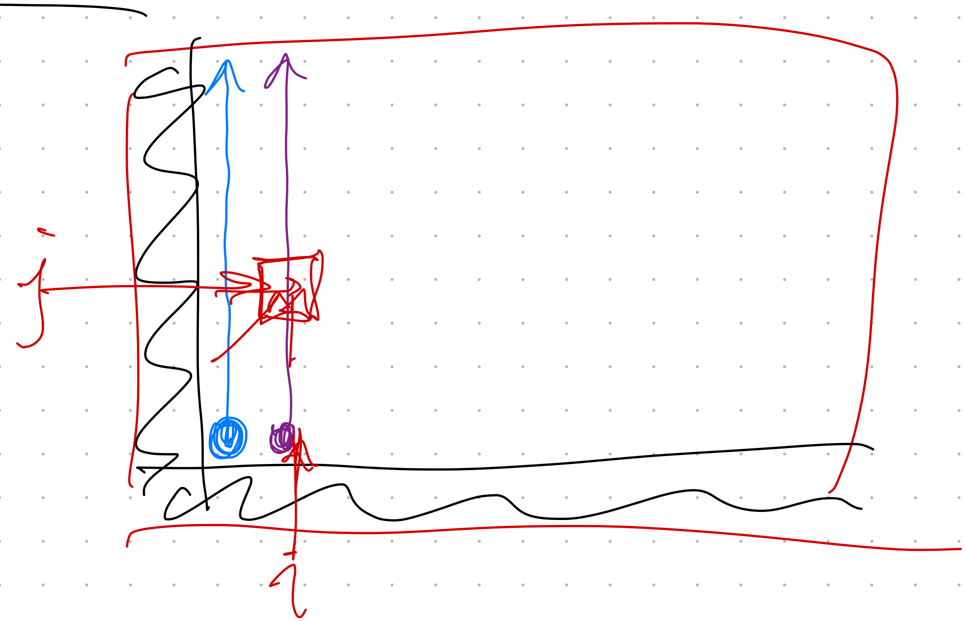
$ED(0,j) = j \cdot \gamma$

Edit Distance Algorithm

$ED(0,0) = 0$

$ED(i,0) = i \cdot \gamma$

$ED(0,j) = j \cdot \gamma$



For $i = 1 \longrightarrow n$

  For $j = 1 \longrightarrow m$

    $ED(i,j) = \min \begin{cases} ED(i-1, j-1) + \Delta_{S_i T_j}, \\ ED(i-1, j) + \gamma, \\ ED(i, j-1) + \gamma \end{cases}$

return $ED(n,m)$

Edit Distance Algorithm

$ED(0,0) = 0$

$ED(i,0) = i \cdot \gamma$

$ED(0,j) = j \cdot \gamma$

For $i = 1 \to n$

  For $j = 1 \to m$

    $ED(i,j) = \min \left\{ ED(i-1,j-1) + \Delta_{S_i T_j},\ ED(i-1,j) + \gamma,\ ED(i,j-1) + \gamma \right\}$

return $ED(n,m)$

Running Time.

- $1+n+m$ initialization
- n outer iterations
  m inner iterations
  $O(1)$ per iteration

Edit Distance Algorithm

$ED(0,0) = 0$
$ED(i,0) = i \cdot \gamma$
$ED(0,j) = j \cdot \gamma$

For $i = 1 \to n$
  For $j = 1 \to m$
    $ED(i,j) = \min \left\{ \begin{array}{l} ED(i-1,j-1) + \Delta_{S_i T_j}, \; ED(i-1,j) + \gamma, \; ED(i,j-1) + \gamma \end{array} \right\}$

return $ED(n,m)$

Running Time

- $1+n+m$ initialization
- n outer iterations
  m inner iterations
  $O(1)$ per iteration

Space

DP Table has $(m+1) \times (n+1)$ cells

$O(mn)$

Edit Distance ≡ Sequence Alignment

```
- - C R Y P T - O G R A P H Y
    | | | | |   | |
E N C R Y P T I N G - - - - -
```

An __alignment__ is a non-crossing matching.

Edit Distance ≡ Sequence Alignment

An <u>alignment</u> is a non-crossing matching.

Edit Distance ≡ Sequence Alignment

```
- - C R Y P T - O G R A P H Y
    | | | | |   | |
E N C R Y P T I N G - - - - -
```

changes

An <u>alignment</u> is a non-crossing matching.

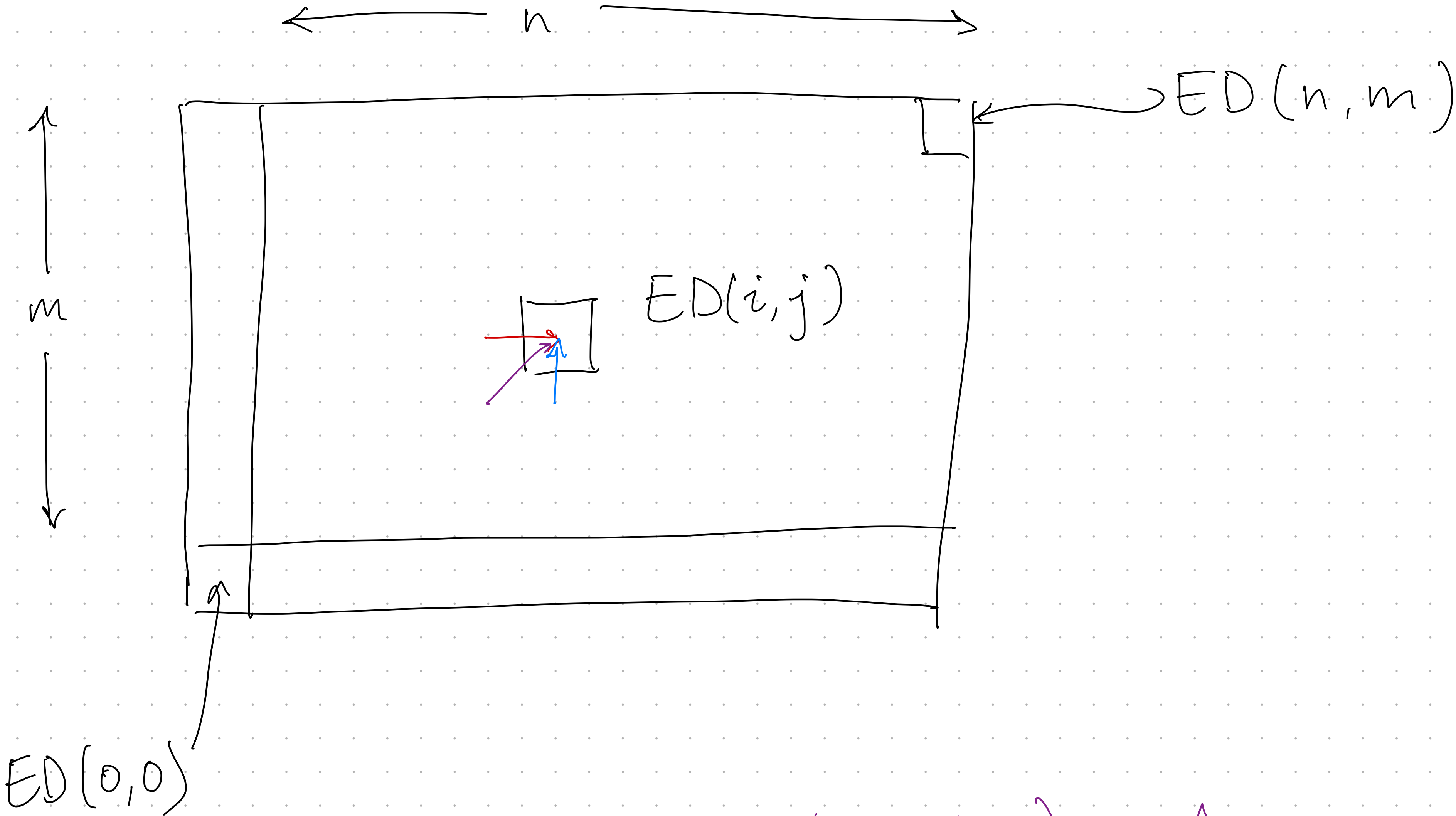
Edit Distance ≡ Sequence Alignment

```
  CRYPT OGRAPHY
ENCRYPTING
```

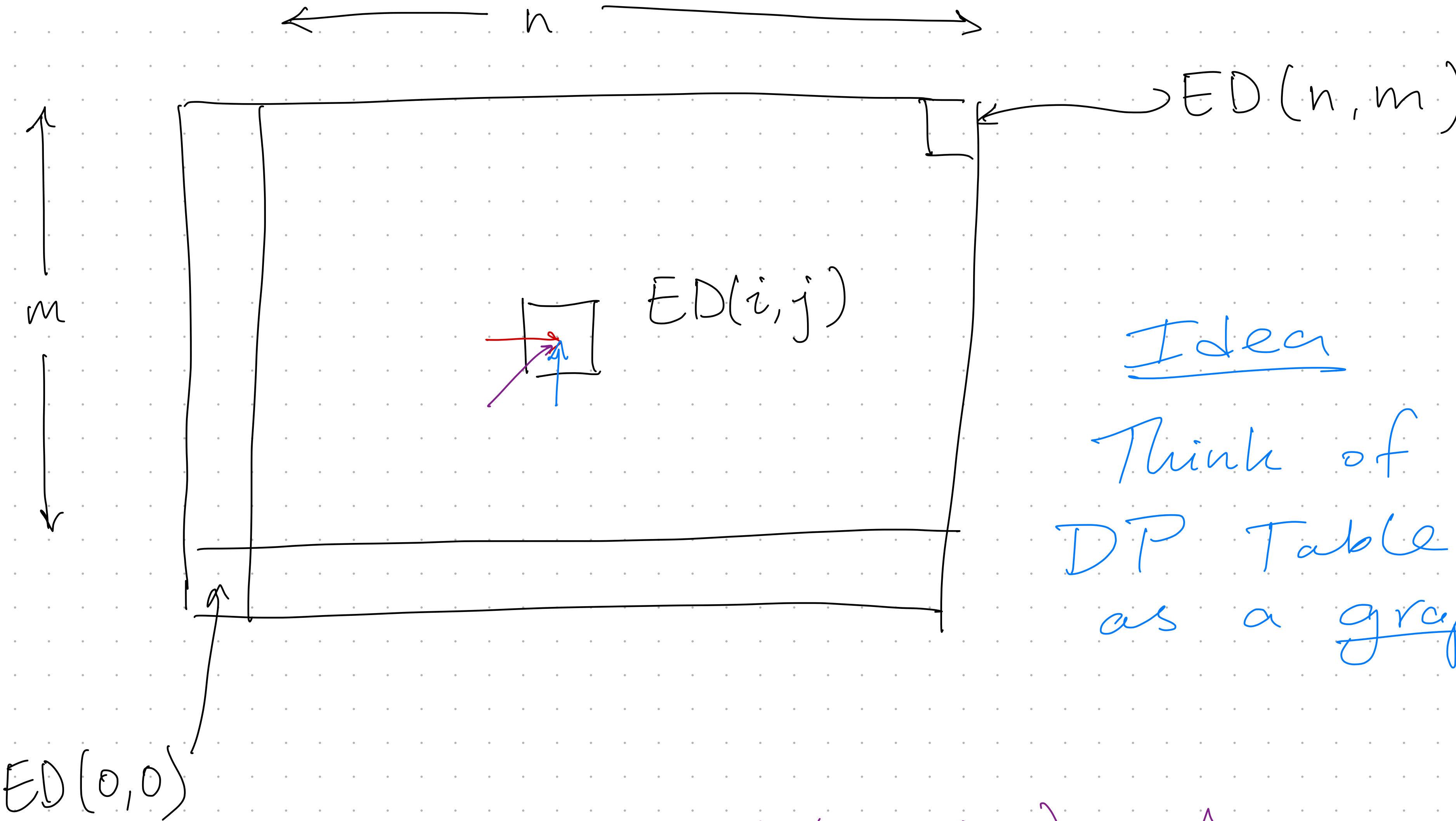
An <u>alignment</u> is a non-crossing matching.

ED Algorithm computed the <u>distance</u>

Can we compute the alignment?



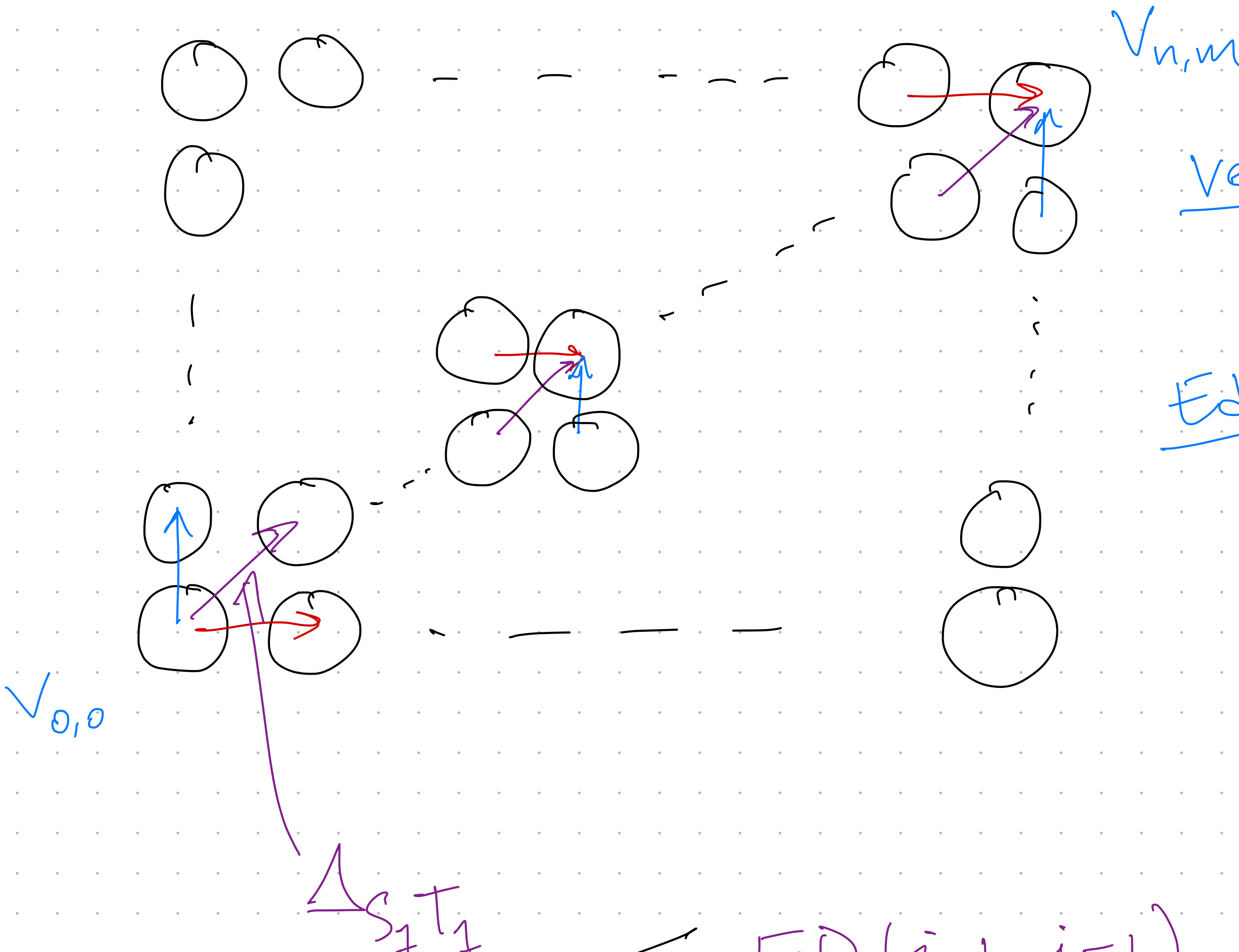
$$ED(i,j) = \min \begin{cases} ED(i-1, j-1) + \Delta_{S_i T_j}, \\ ED(i-1, j) + \gamma, \\ ED(i, j-1) + \gamma \end{cases}$$



Idea

Think of DP Table as a graph

$$ED(i,j) = \min \left\{ \begin{array}{l} ED(i-1, j-1) + \Delta_{s_i t_j}, \\ ED(i-1, j) + \gamma, \\ ED(i, j-1) + \gamma \end{array} \right\}$$



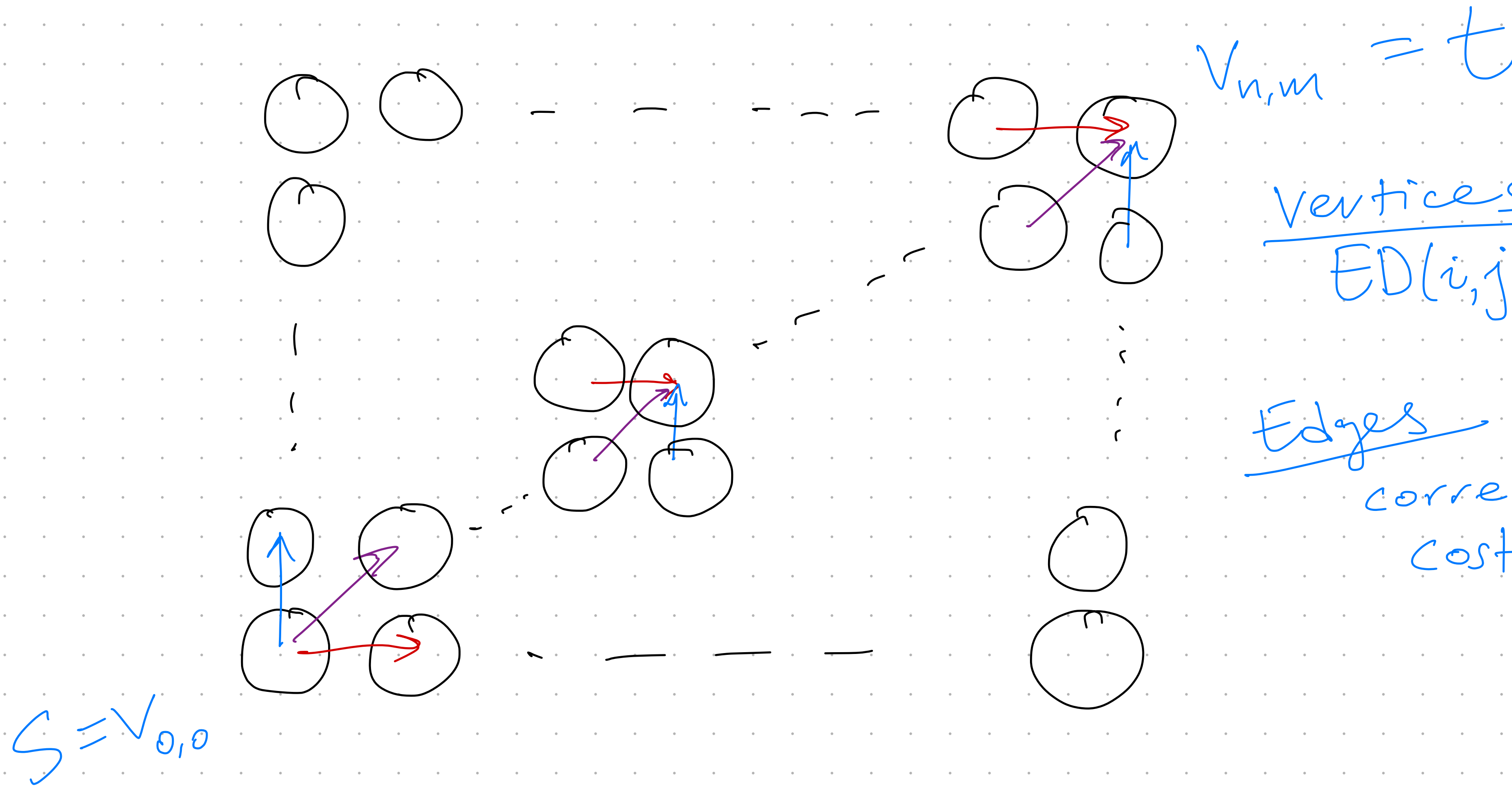
Vertices

$ED(i,j) \equiv v_{i,j}$

Edges

correspond to cost of edit

$$ED(i,j) = \min \left\{ \begin{array}{l} ED(i-1, j-1) + \Delta_{S_i T_j}, \\ ED(i-1, j) + \gamma, \\ ED(i, j-1) + \gamma \end{array} \right\}$$



Vertices

$ED(i,j) \equiv v_{i,j}$

Edges

correspond to cost of edit

Minimum Cost Alignment

$\equiv$

Shortest st-Path in ED Graph

Theorem. Shortest path from v_{00} to v_{ij} equals $ED(i,j)$.

Pf. By induction on $i+j$.

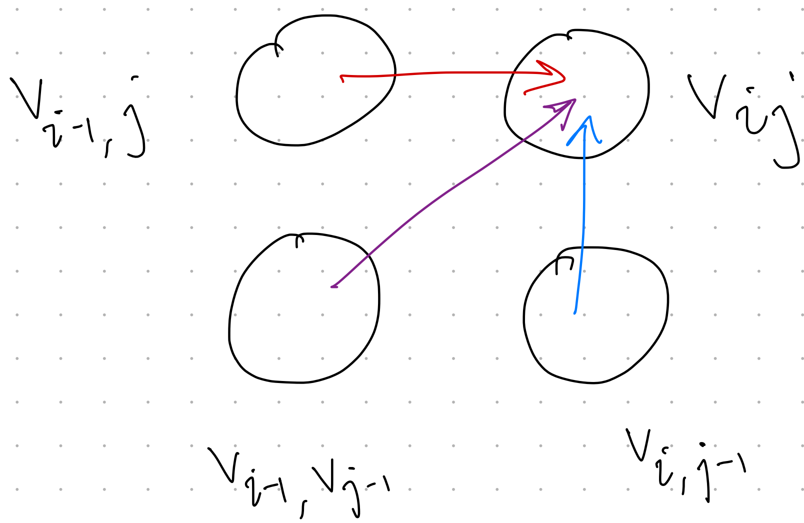
Base Case. Shortest path from $v_{00} \to v_{00} = 0$
$= ED(0,0)$

Theorem. Shortest path from v_{00} to v_{ij} equals $ED(i,j)$.

Pf. By induction on $i+j$.

Base Case. Shortest path from $v_{00} \to v_{00} = 0$
$= ED(0,0)$

Inductive Step. Suppose shortest path to $v_{kl} = ED(k,l)$ for all $k+l < i+j$.



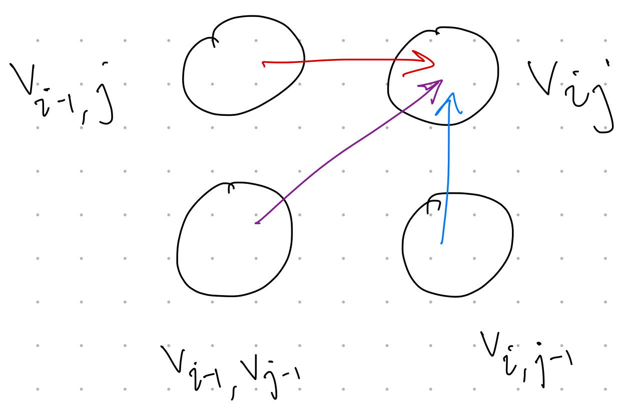
Theorem. Shortest path from v_{00} to v_{ij} equals $ED(i,j)$.

Pf. By induction on $i+j$.

Base Case. Shortest path from $v_{00} \to v_{00} = 0$
$= ED(0,0)$

Inductive Step. Suppose shortest path to $v_{kl} = ED(k,l)$ for all $k+l < i+j$.

$$SP(v_{00} \to v_{ij}) = \min \begin{cases} SP(v_{00} \to v_{i-1,j-1}) + \Delta_{S_i T_j}, \\ SP(v_{00} \to v_{i-1,j}) + \gamma \\ SP(v_{00} \to v_{i,j-1}) + \gamma \end{cases}$$



Theorem. Shortest path from v_{00} to v_{ij} equals $ED(i,j)$.

Pf. By induction on $i+j$.

Base Case. Shortest path from $v_{00} \to v_{00} = 0$
$= ED(0,0)$

Inductive Step. Suppose shortest path to $v_{kl} = ED(k,l)$ for all $k+l < i+j$.

$$SP(v_{00} \to v_{ij}) = \min \begin{Bmatrix} ED(i-1, j-1) + \Delta_{S_i T_j}, \\ ED(i-1, j) + \gamma \\ ED(i, j-1) + \gamma \end{Bmatrix}$$

Theorem. Shortest path from v_{00} to v_{ij} equals $ED(i,j)$.

Pf. By induction on $i+j$.

Base Case. Shortest path from $v_{00} \to v_{00} = 0$
$= ED(0,0)$

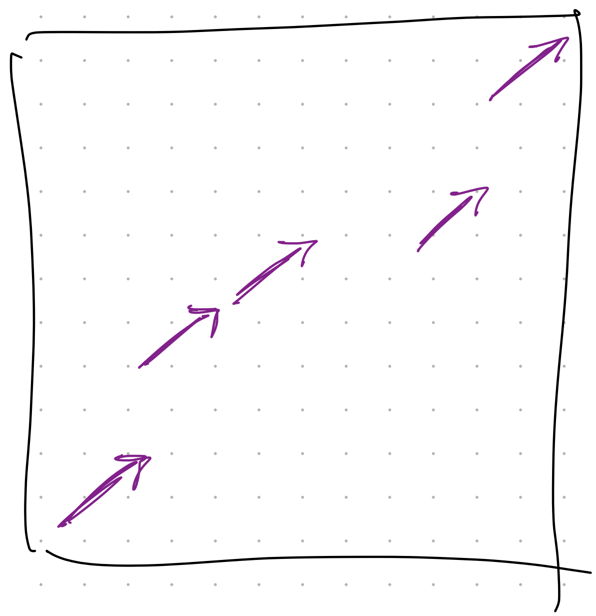
Inductive Step. Suppose shortest path to $v_{kl} = ED(k,l)$ for all $k+l < i+j$.

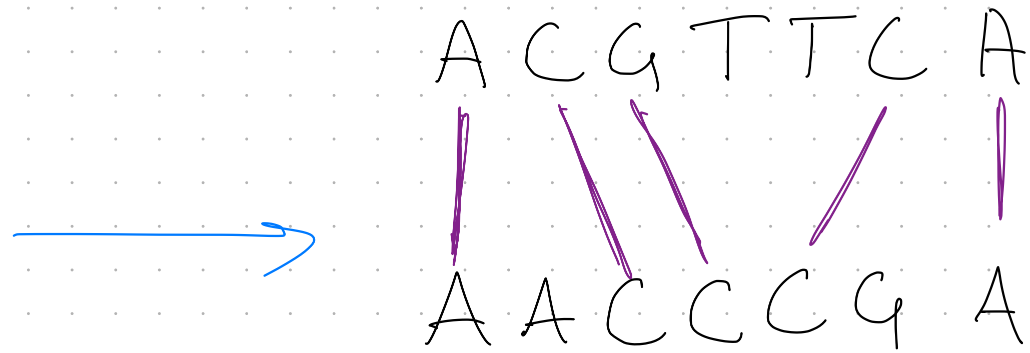
$$SP(v_{00} \to v_{ij}) = \min \left\{ \begin{array}{l} ED(i-1,j-1) + \Delta_{S_i T_j}, \\ ED(i-1,j) + \gamma \\ ED(i,j-1) + \gamma \end{array} \right\}$$

$$= ED(i,j)$$

Theorem. Shortest path from v_{00} to v_{ij} equals $ED(i,j)$.

Corollary. Diagonal edges on shortest path from $v_{00} \to v_{nm}$ correspond to alignment.





Edit Distance Algorithm

$ED(0,0) = 0$

$ED(i,0) = i \cdot \gamma$

$ED(0,j) = j \cdot \gamma$

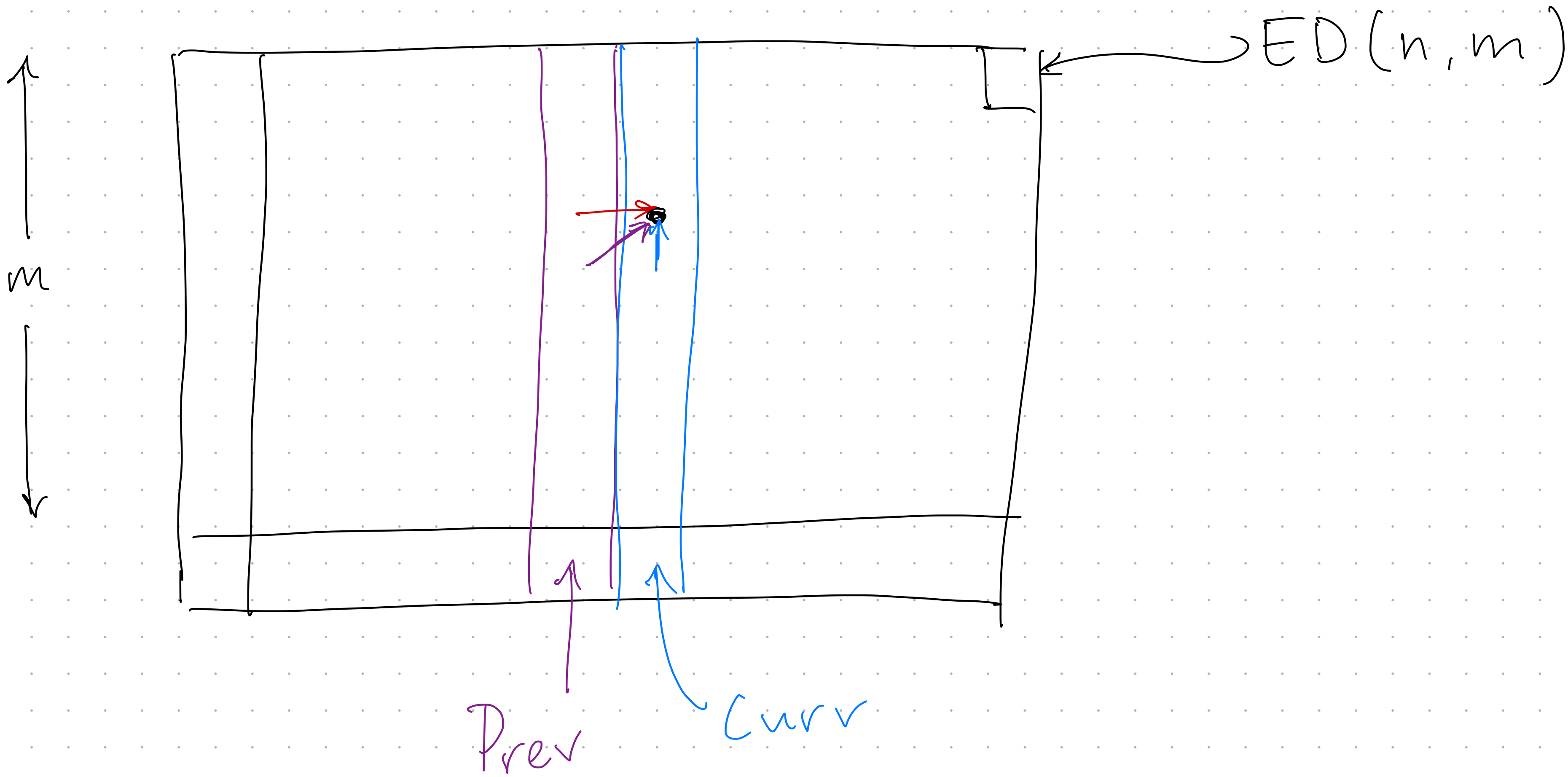
How much space did we really need?

For $i = 1 \rightarrow n$

  For $j = 1 \rightarrow m$

    $ED(i,j) = \min \left\{ ED(i-1,j-1) + \Delta_{S_i T_j},\ ED(i-1,j) + \gamma,\ ED(i,j-1) + \gamma \right\}$

return $ED(n,m)$



$$\mathrm{Curr}(j) = \min \begin{cases} \mathrm{Prev}(j-1) + \Delta_{s_i T_j}, \\ \mathrm{Prev}(j) + \gamma, \\ \mathrm{Curr}(j-1) + \gamma \end{cases}$$

Linear Space ED.

$\text{Prev}(j) = j \cdot \gamma$ // m-entry 1D arrays

$\text{Curr}(j) = 0$

For $i = 1 \longrightarrow n$

  $\text{Curr}(0) = i \cdot \gamma$

  For $j = 1 \longrightarrow m$

    $\text{Curr}(j) = \min \left\{ \text{Prev}(j-1) + \Delta_{S_i T_j},\ \text{Prev}(j) + \gamma,\ \text{Curr}(j-1) + \gamma \right\}$

  $\text{Prev} \leftarrow \text{Curr}.$

return $\text{Curr}(m)$.