Spelling correction

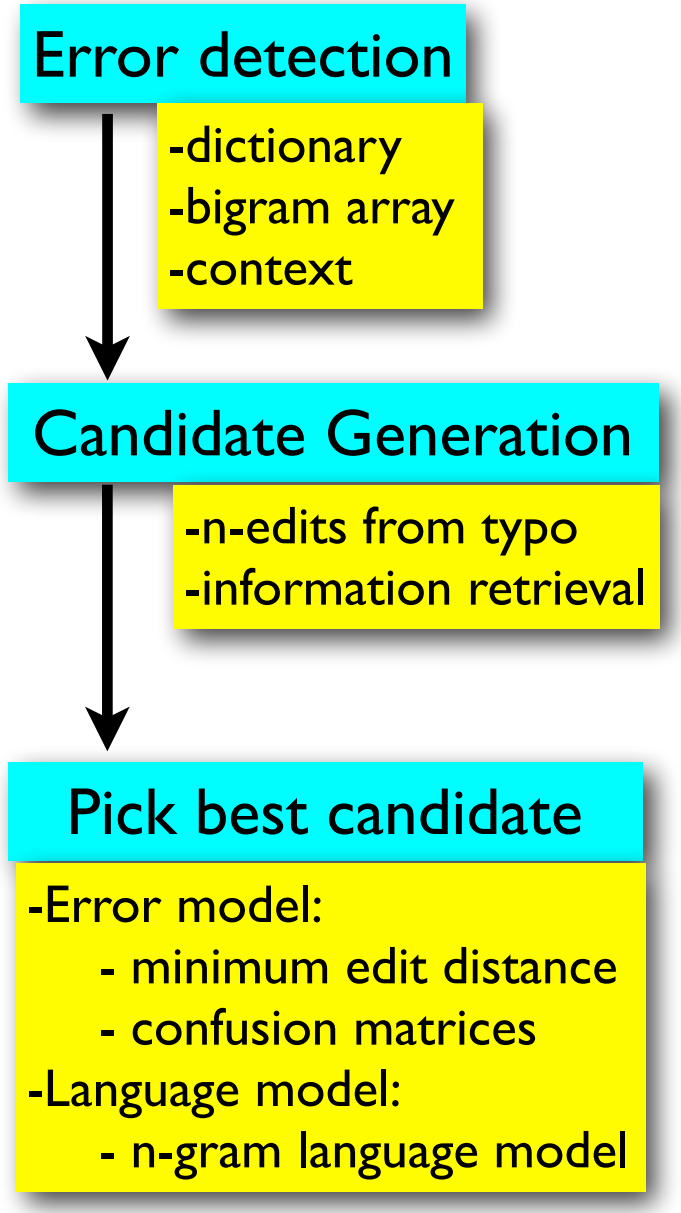
Jason Baldridge
UT Austin
Language and Computers

Slides to supplement Markus Dickinson's slides

The big picture

- Speling iz a bigg problim:
  - People make spelling mistakes all the time.
  - Optical character recognition might recognize words incorrectly, leading to misspelled OCR text.
  - Speech recognition, e.g. for helping produce close-captioned text, can have similar errors.
- We'd like to find ways to fix errors automatically.
- Here's a running example:
  - "She is there favorite acress in tonw."
  - We'll focus mainly on "tonw"

Stages of *automated* spelling correction



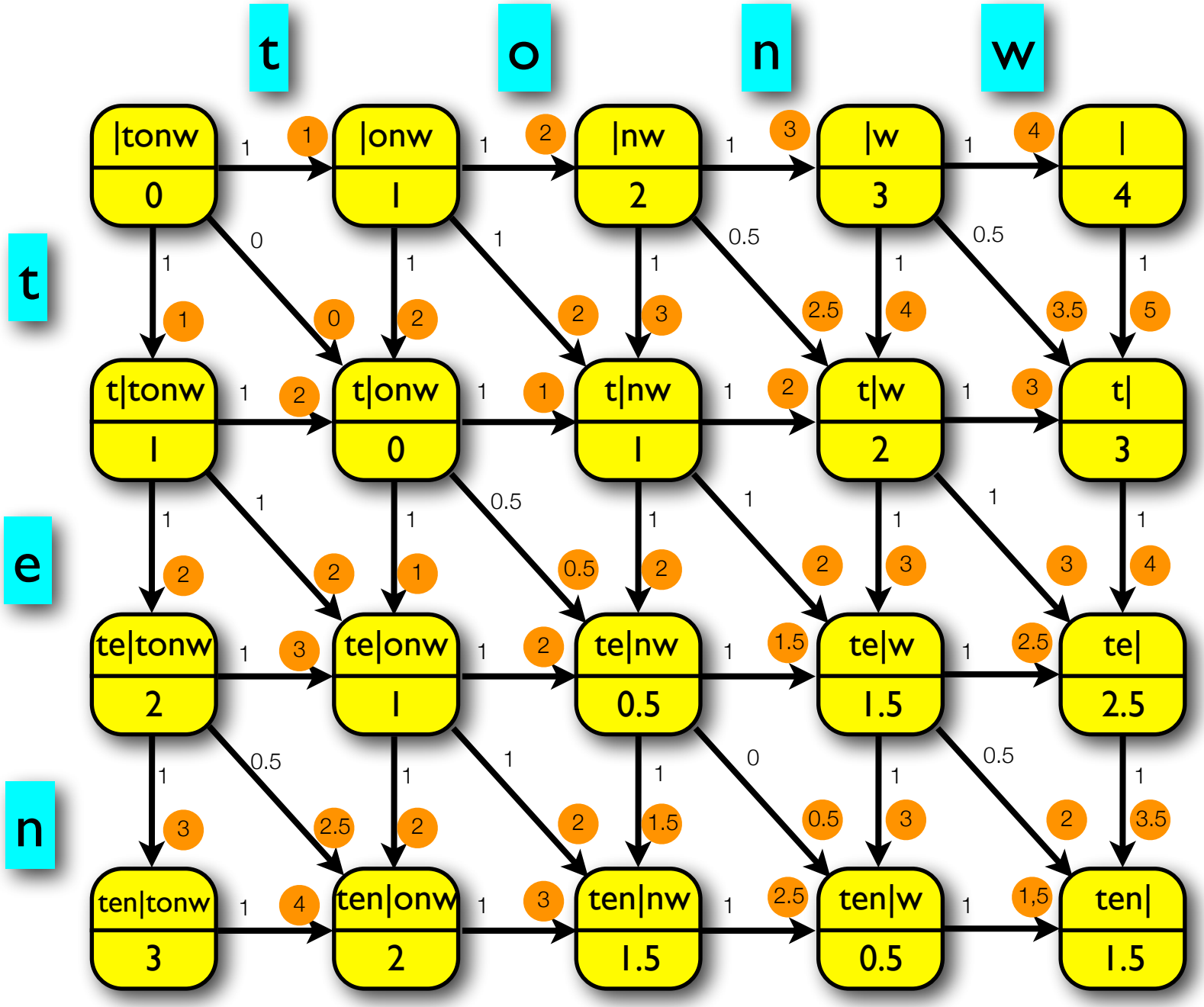
Minimum edit distance

- In determining how likely a candidate was intended when a typo was produced, we are usually interested in calculating the distance from the typo to the candidate.
  - we do this in terms of three basic string editing operations: deletion, insertion, and substitution. (see Dickinson slides for discussion)
  - intuitively, “actress” is closer to “acress” than “arrest” is to “acress”; in turn, “arrest” is closer to “acress” than “bob” is to “acress”.
- To turn “arrest” into “acress”, we (humans) know that a reasonable way to do this is to turn the first “r” into a “c” and then the last “s” into a “t” and leave the rest (two edits).
- A dumb algorithm might do stuff like:
  - arrest -> barrest -> barrest -> barrist -> barrista -> arrista -> acrista -> acrissa -> acriss -> bacriss -> bacrissp -> acrissp -> acressp -> acress
- We instead use a directed acyclic graph to find the minimum edit distance efficiently by storing

Minimum edit distance

- See Dickinson's slides for more discussion.
- The next slides will walk you through how to set up and use the graph for computing the minimum edit distance between "tonw" and "ten".
- The nodes of the graph in these slides show what the string looks like at each node after the edits have been performed. They also store the minimum edit distance for each node.

Minimum edit distance for “tonw” to “ten”: static view



Notes

The word on top is the original string. The word on the left is the one we will create by editing the original string.

The nodes are the yellow, rounded squares.

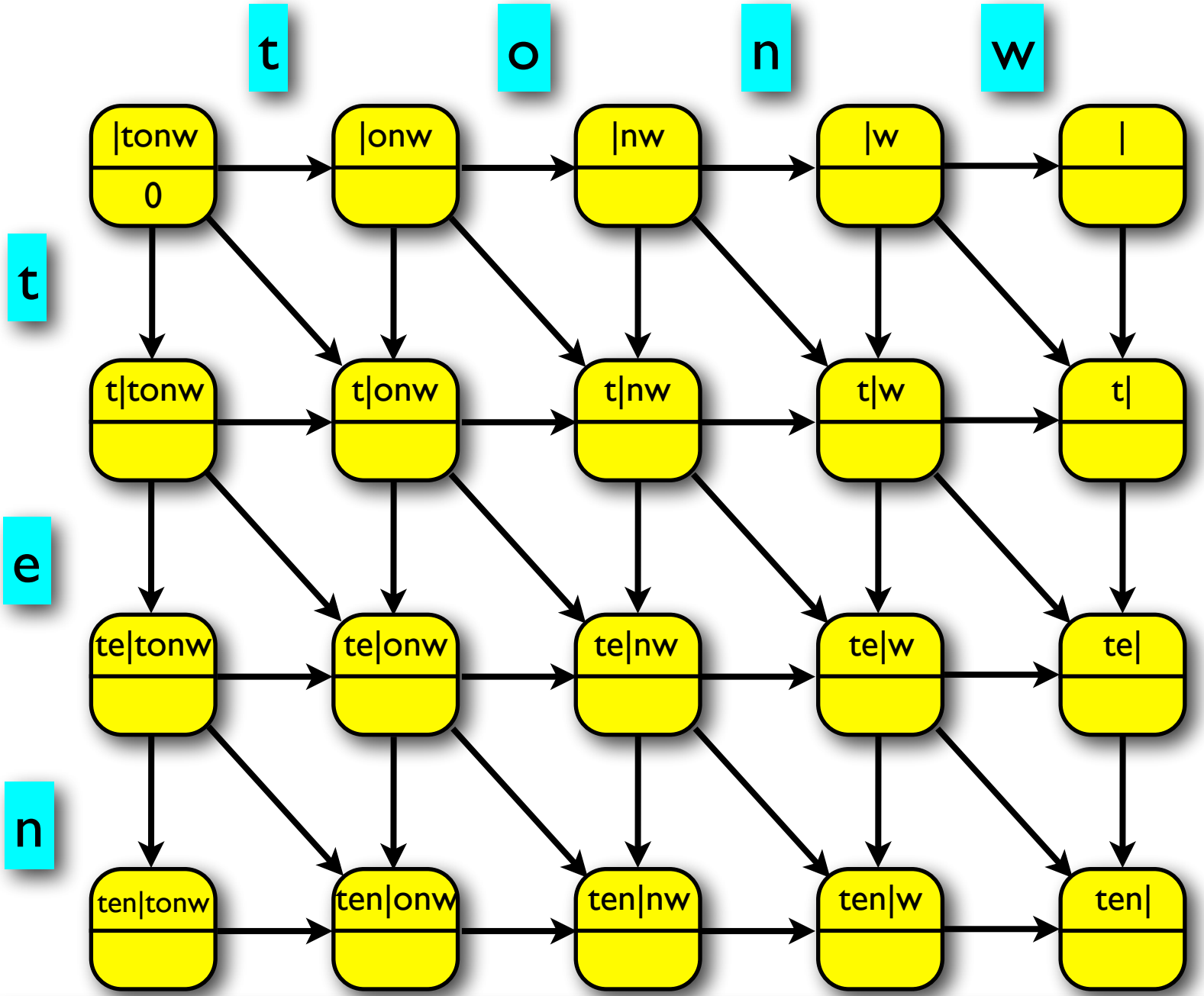
The upper part of a node contains the representation of the string after making the edits to get to that node. The | symbol represents the cursor, like in an editor.

The number in the lower half of a node represents the minimum edit distance to create the string in that node.

Right arrows represent deletion, down arrows represent insertion, and diagonal arrows represent substitution.

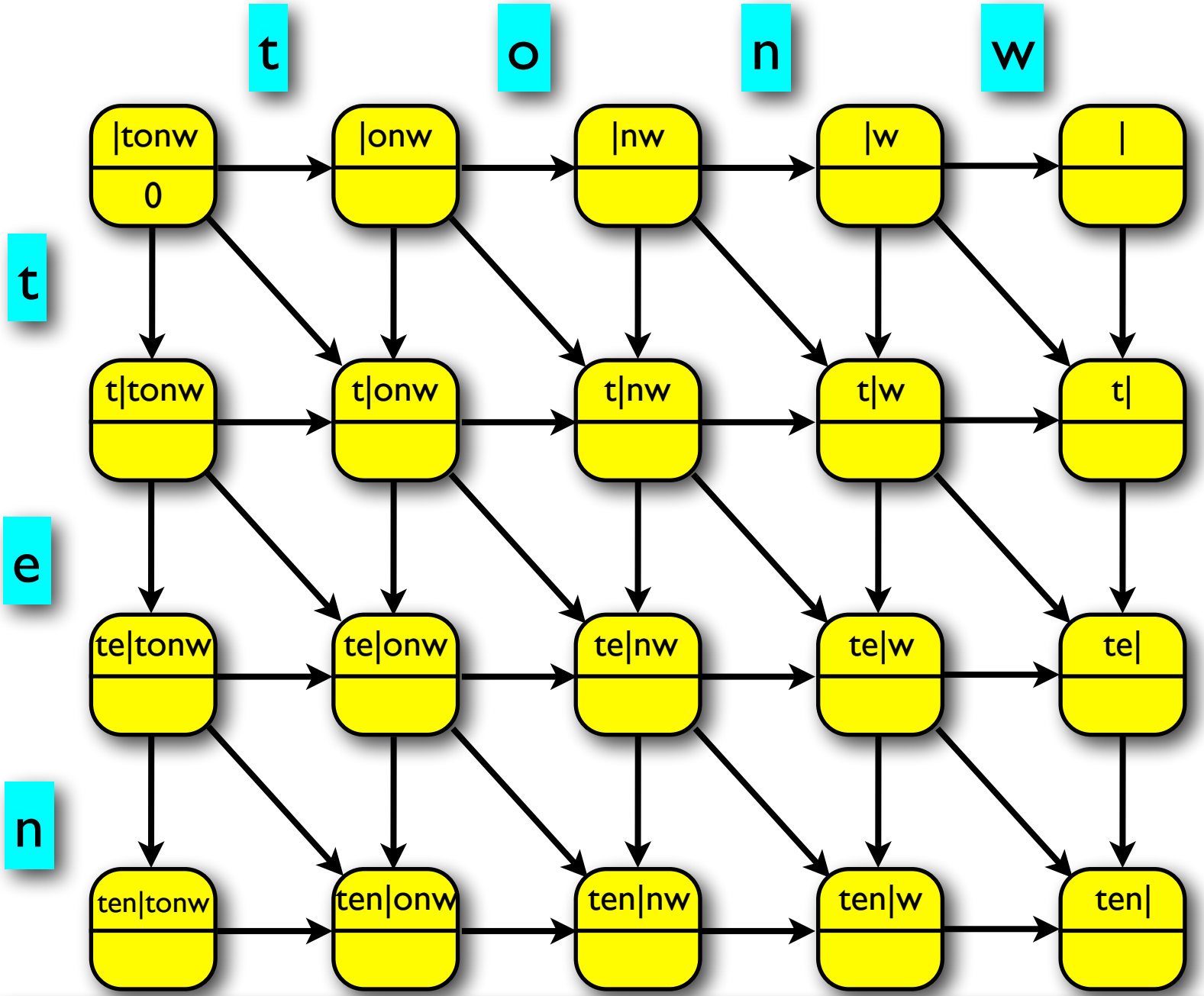
The numbers in orange are the edit distances coming from the previous nodes plus the cost of the edit represented by the arrow.

Minimum edit distance for “tonw” to “ten”: animated



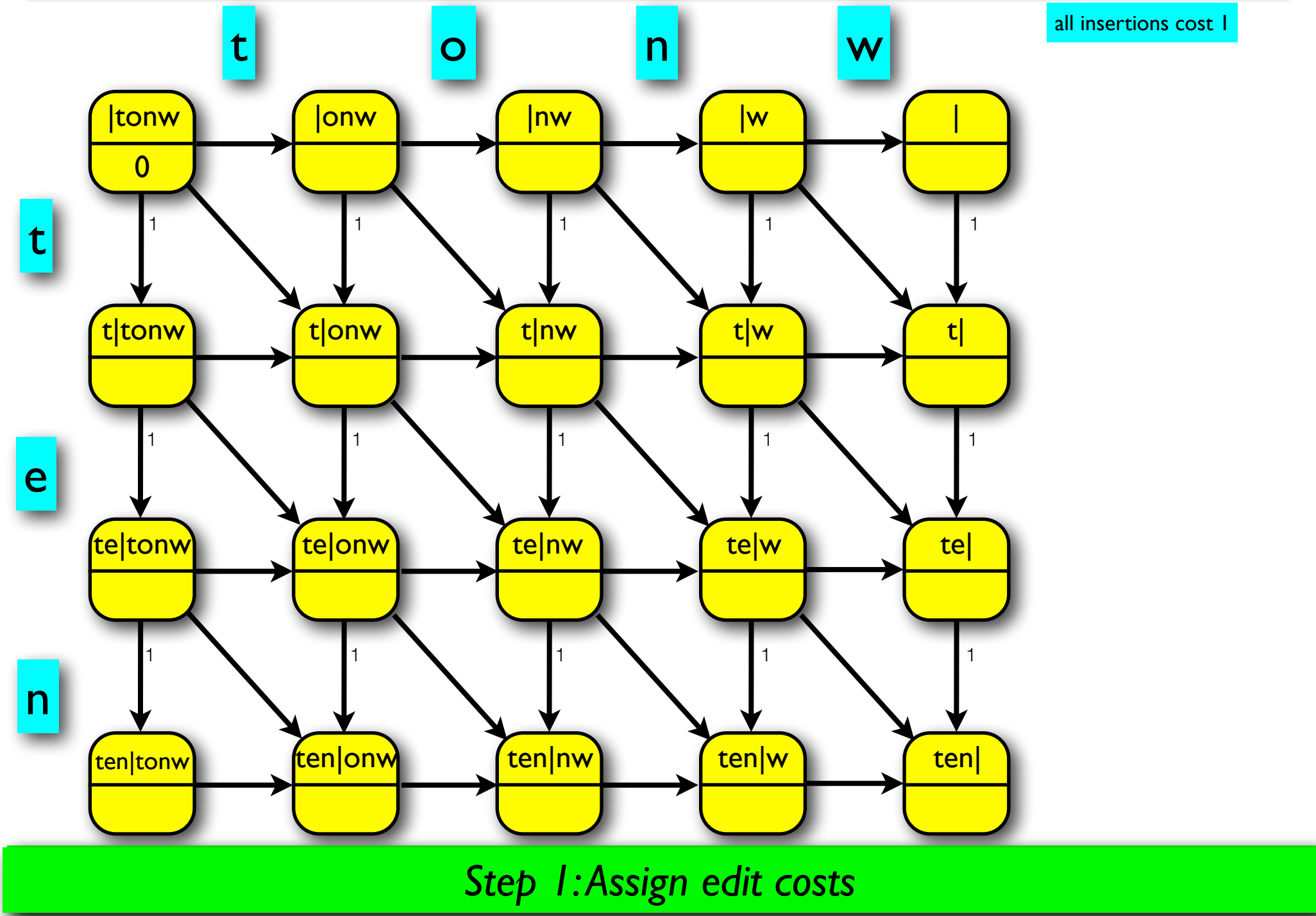
Step 1: Assign edit costs

Minimum edit distance for “tonw” to “ten”: animated

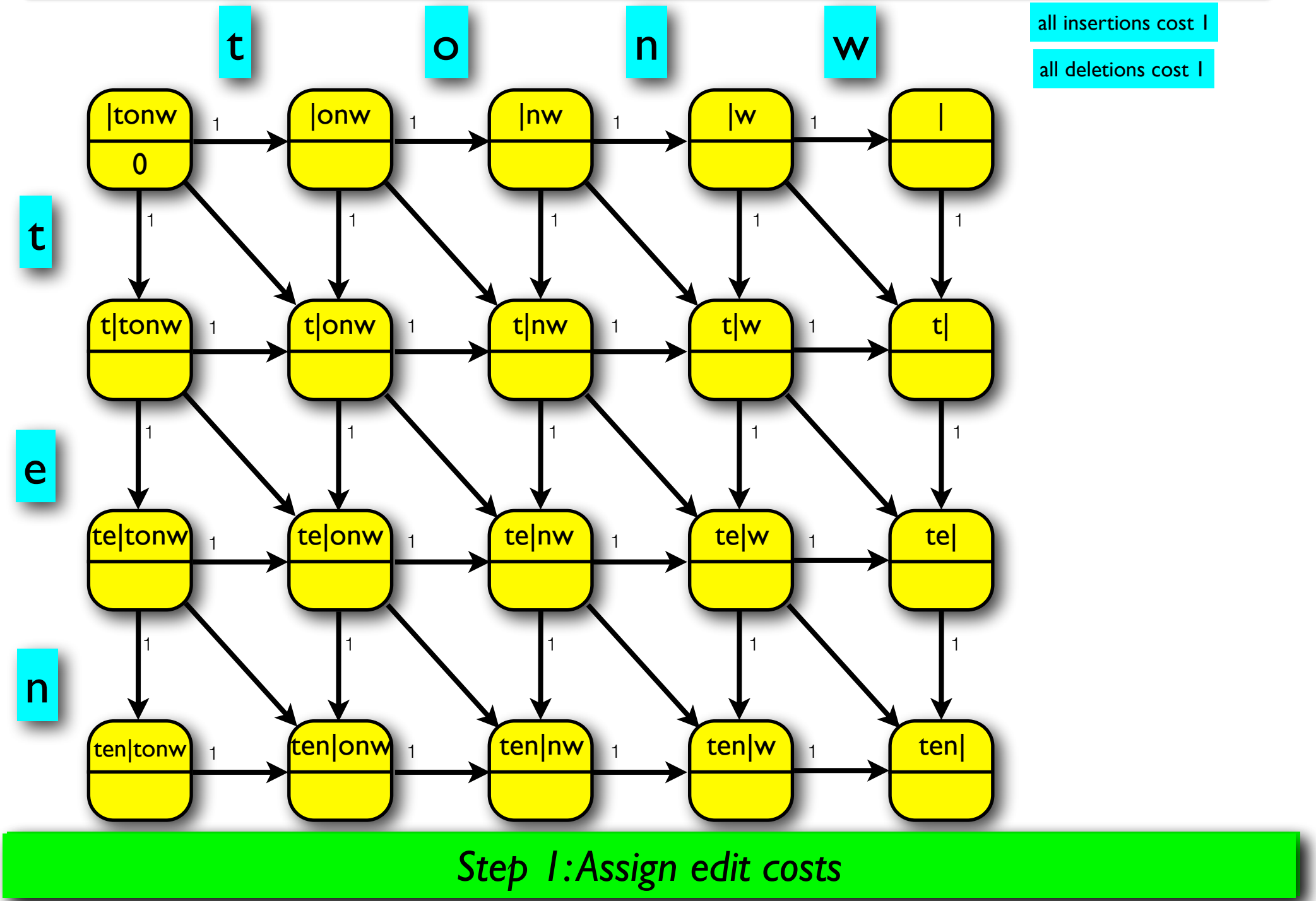


Step 1: Assign edit costs

Minimum edit distance for “tonw” to “ten”: animated

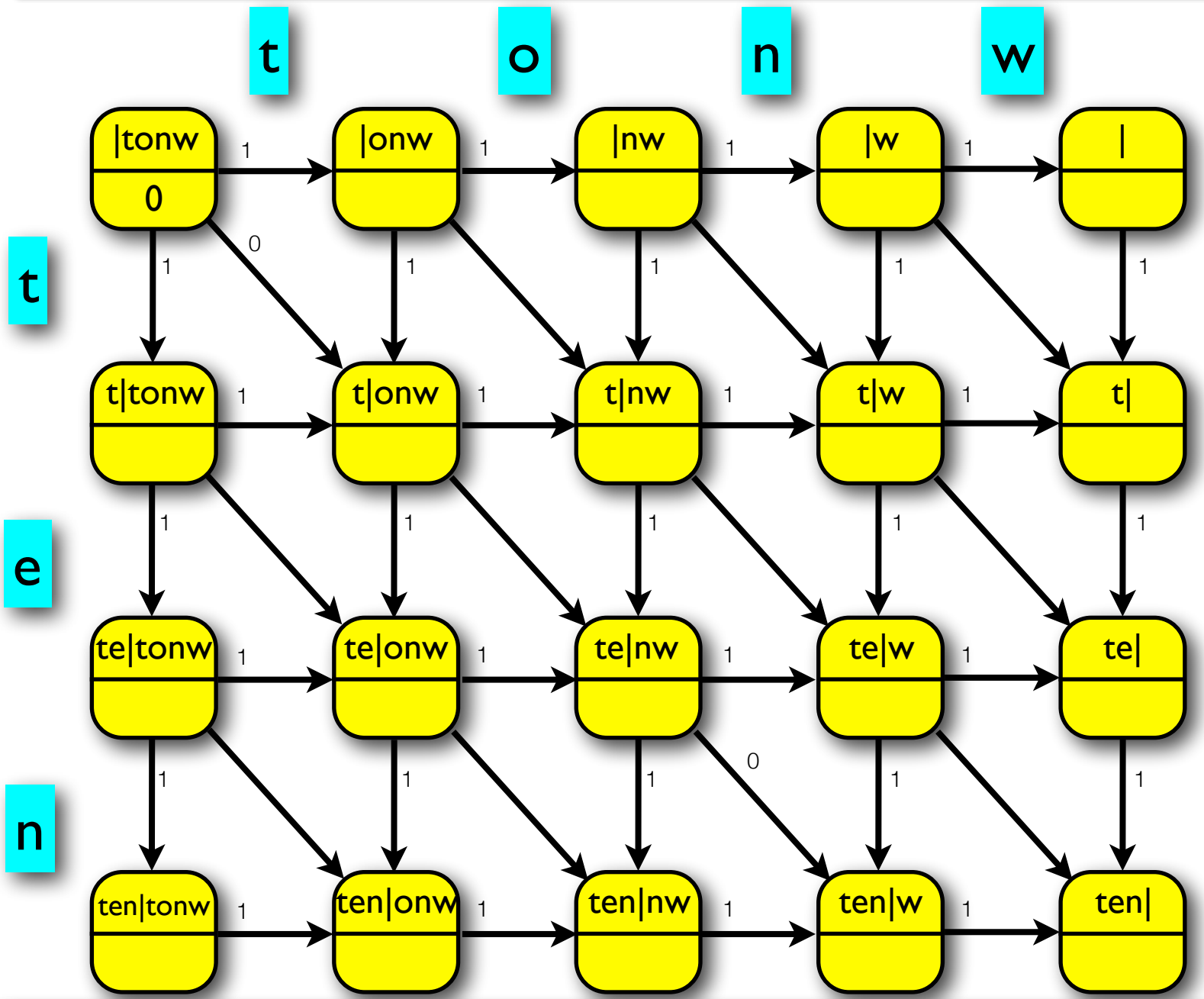


Minimum edit distance for “tonw” to “ten”: animated



Step 1: Assign edit costs

Minimum edit distance for “tonw” to “ten”: animated



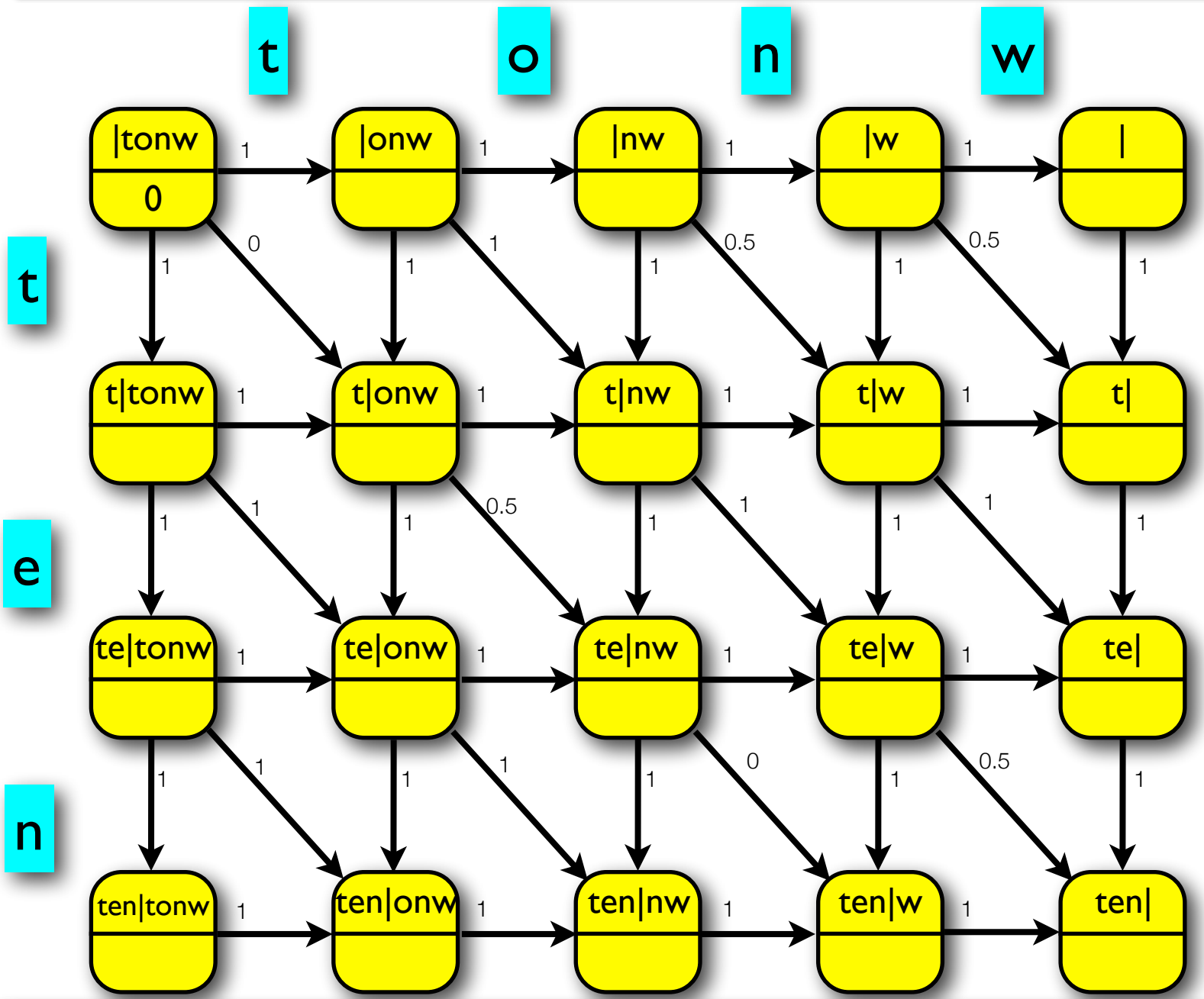
all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

Step 1: Assign edit costs

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

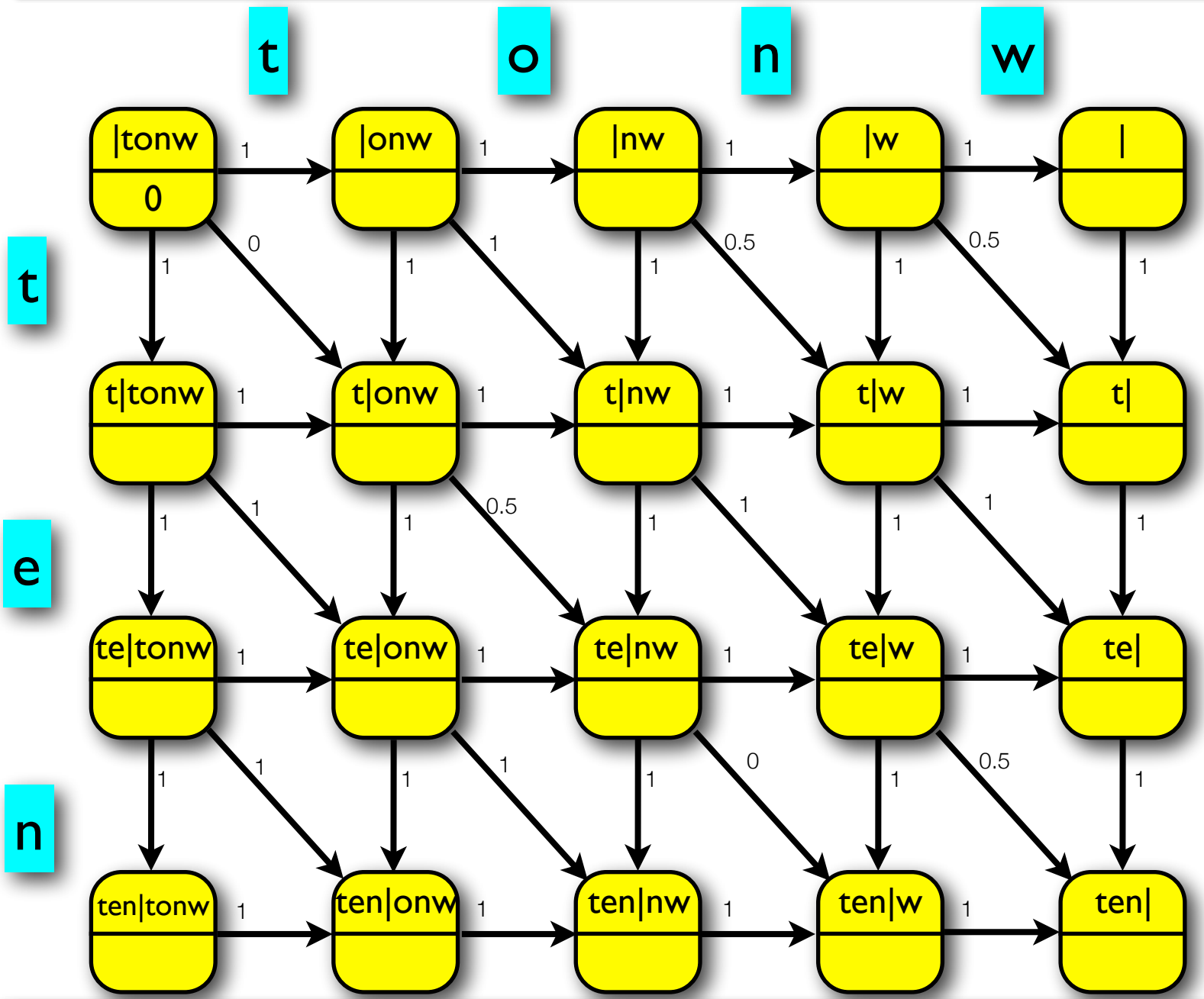
all deletions cost 1

substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

Step 1: Assign edit costs

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

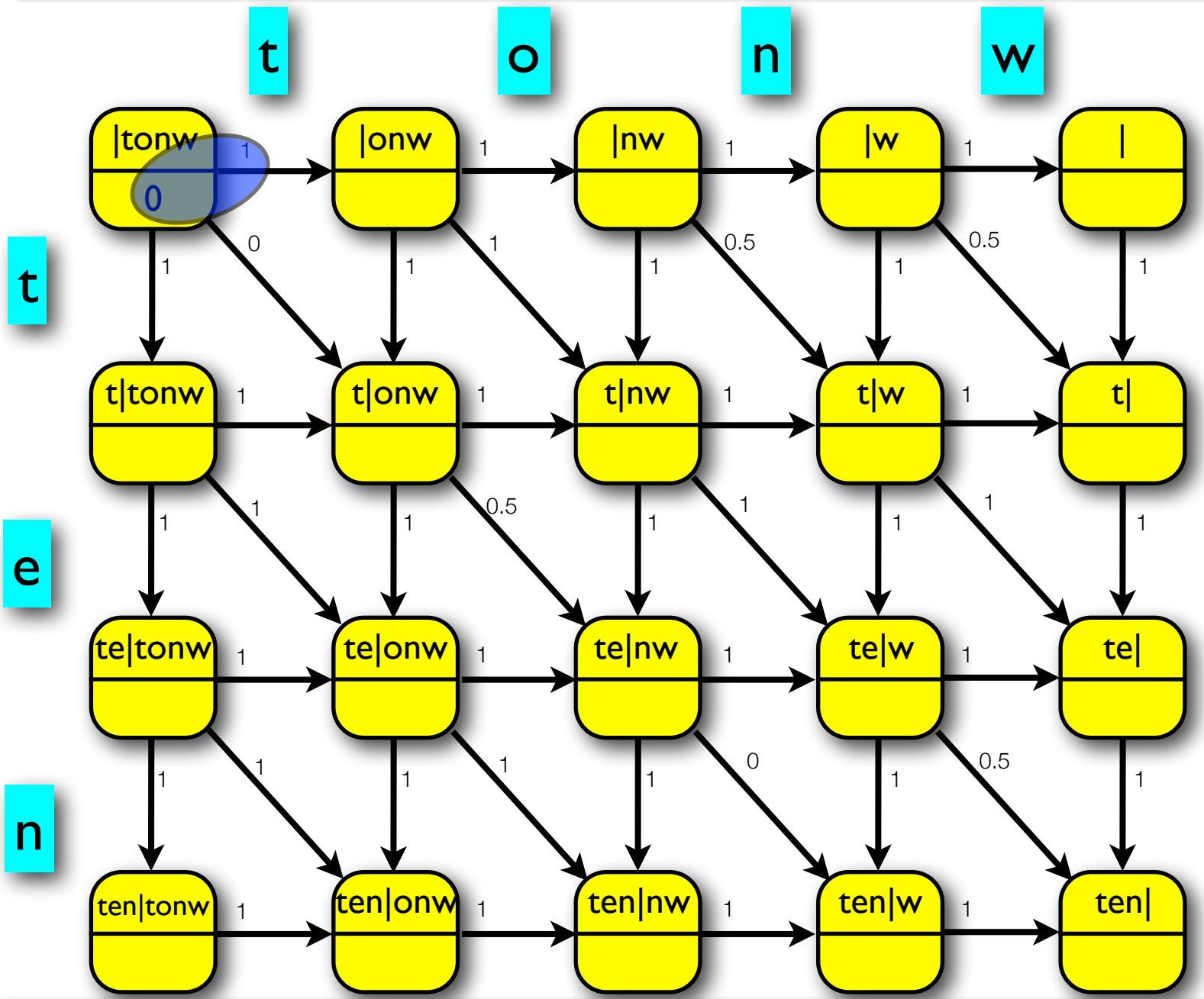
all deletions cost 1

substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

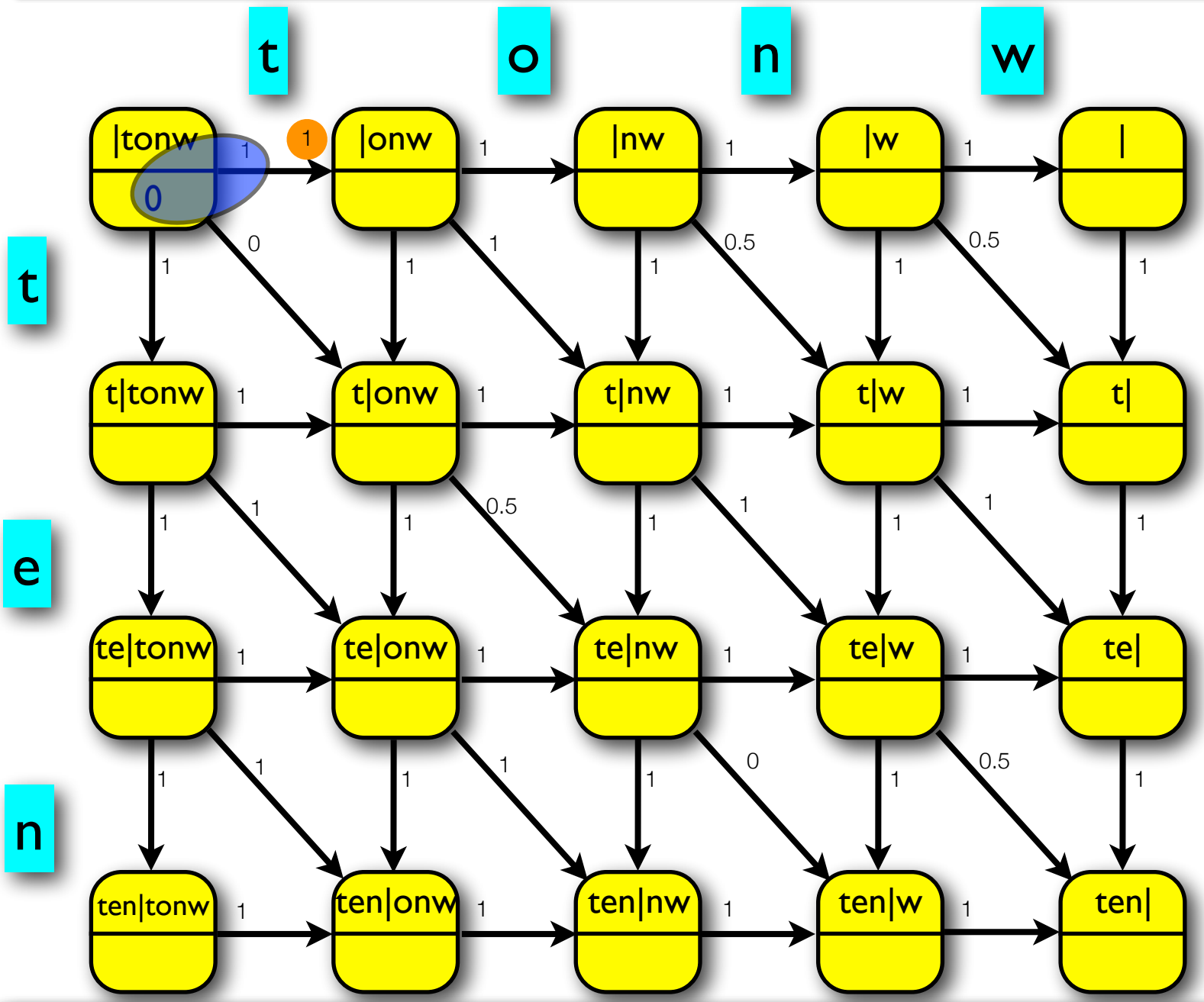
all deletions cost 1

substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

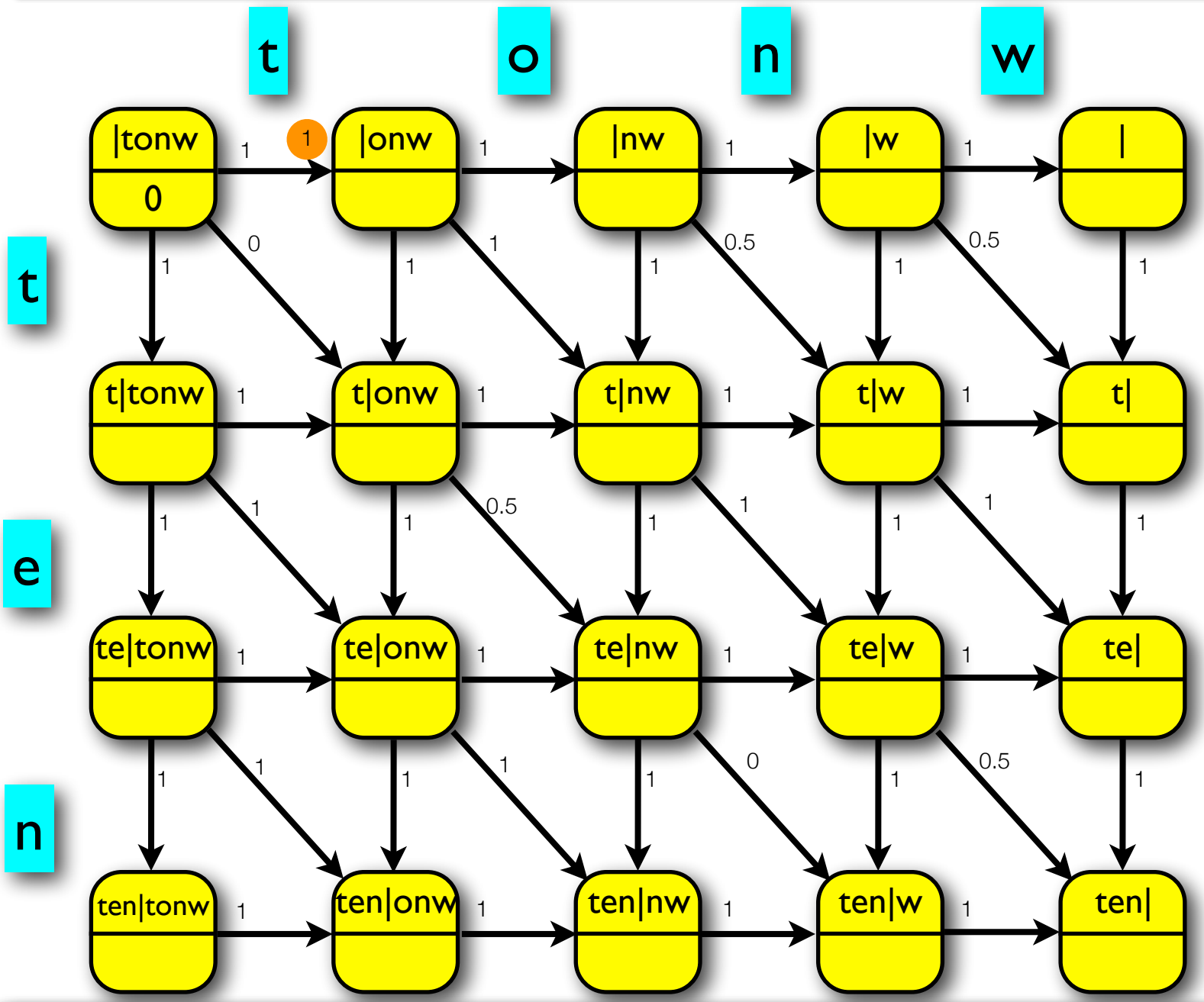
all deletions cost 1

substitutions cost 0 if same letter (e.g.“t”→ “t”,“n”→ “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g“o” → “e”,“n” → “t”); otherwise, cost is 1 (e.g.“t”→ “e”,“e”→ “t”)

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

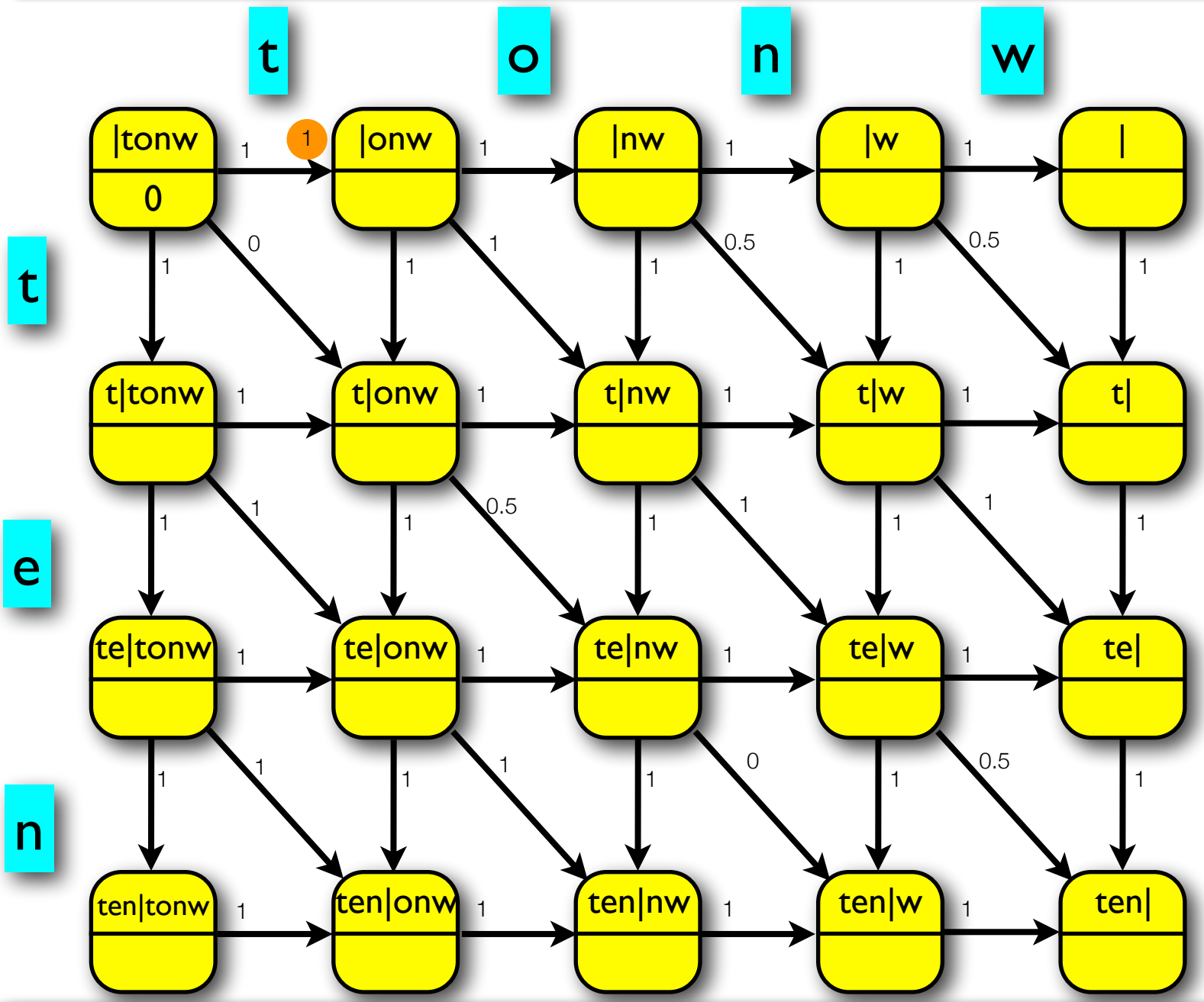
all deletions cost 1

substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

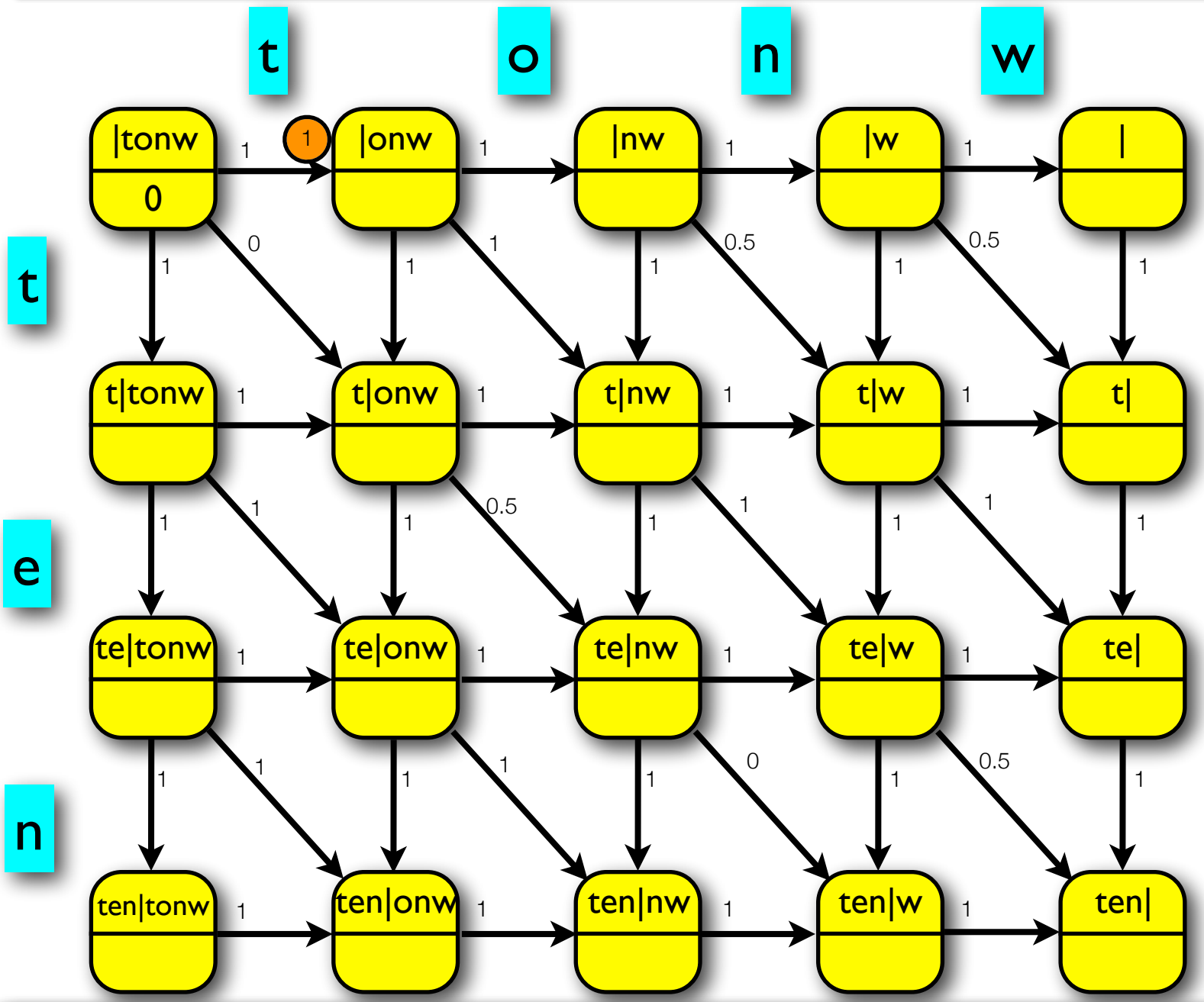
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

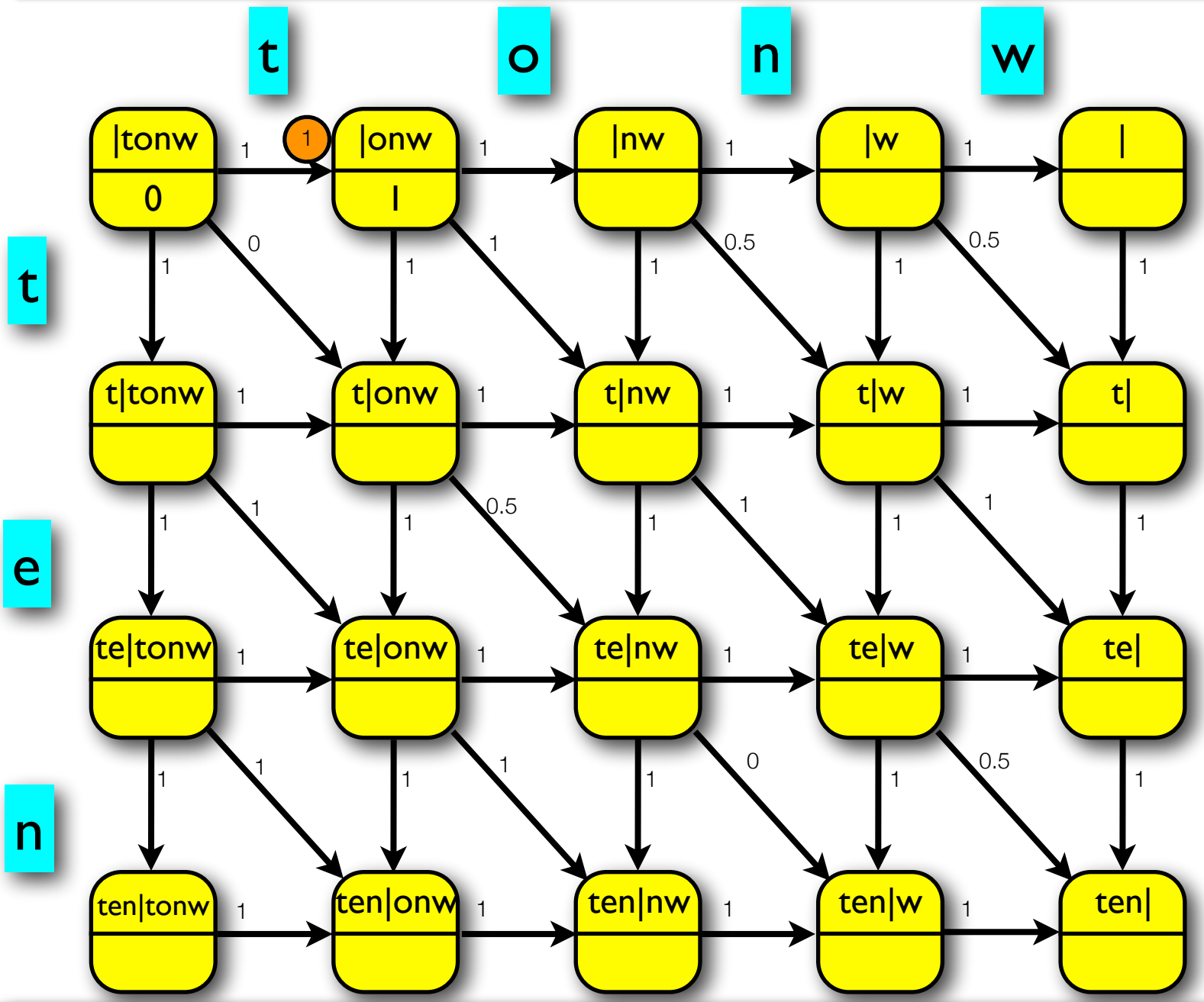
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

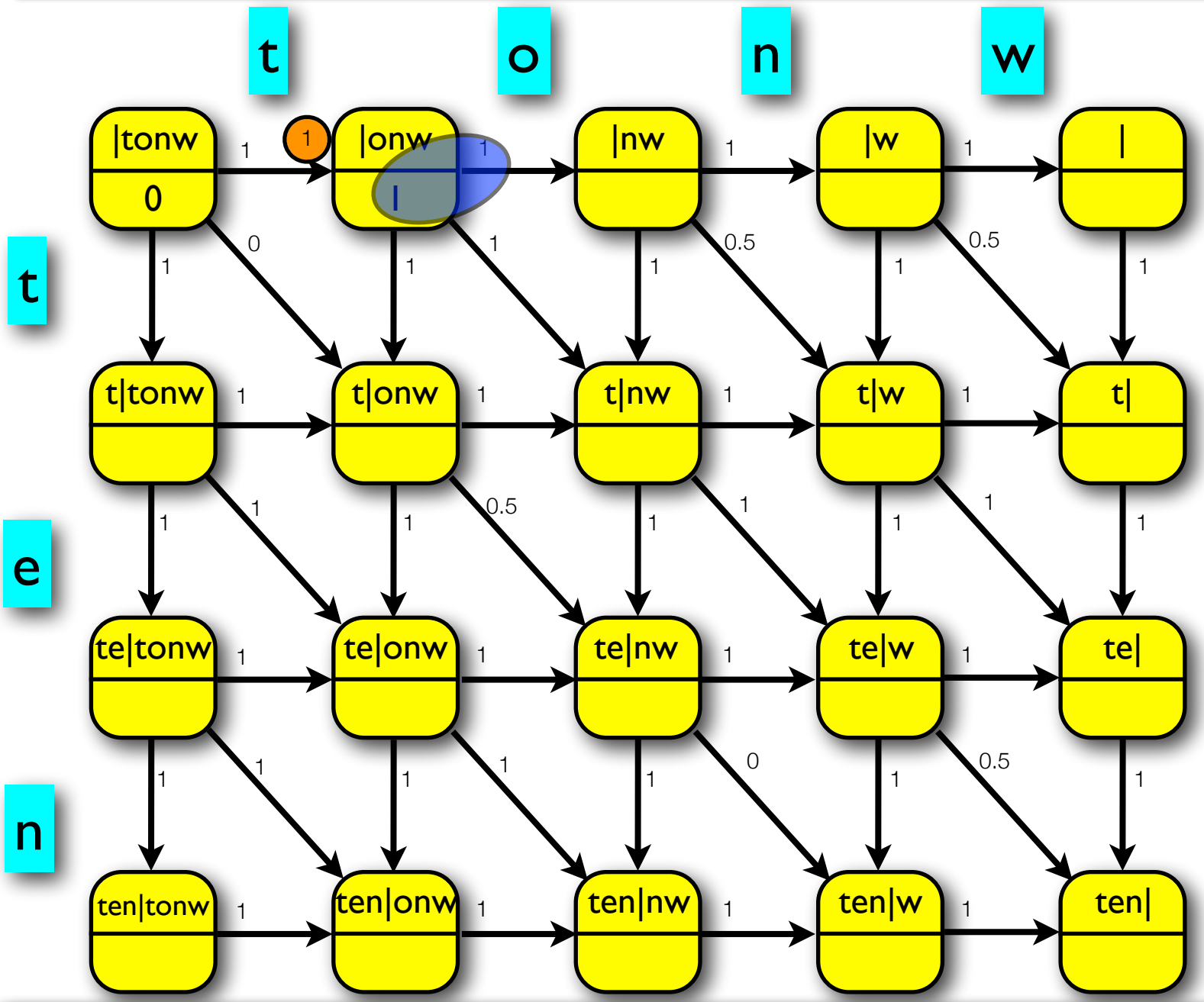
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

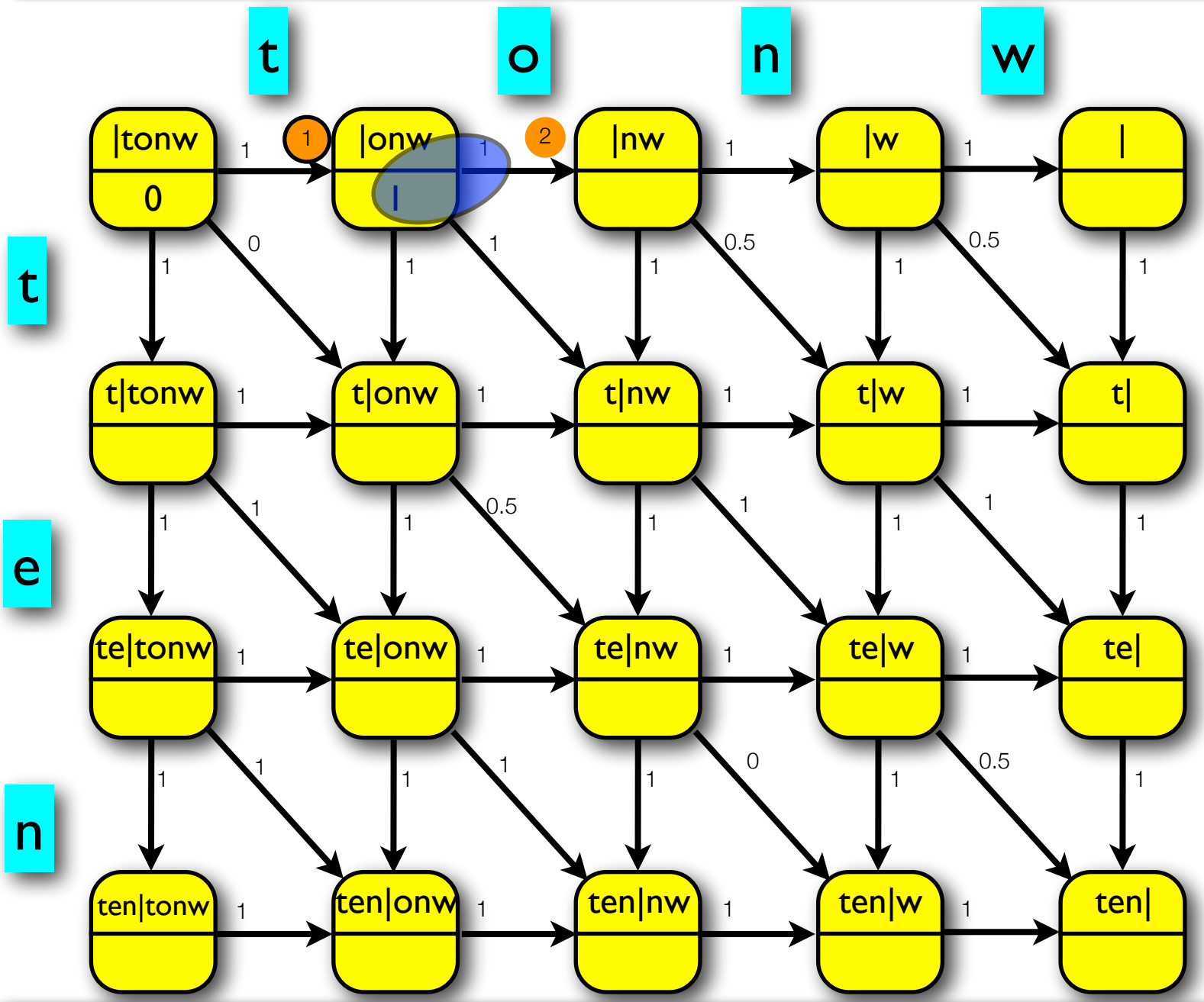
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

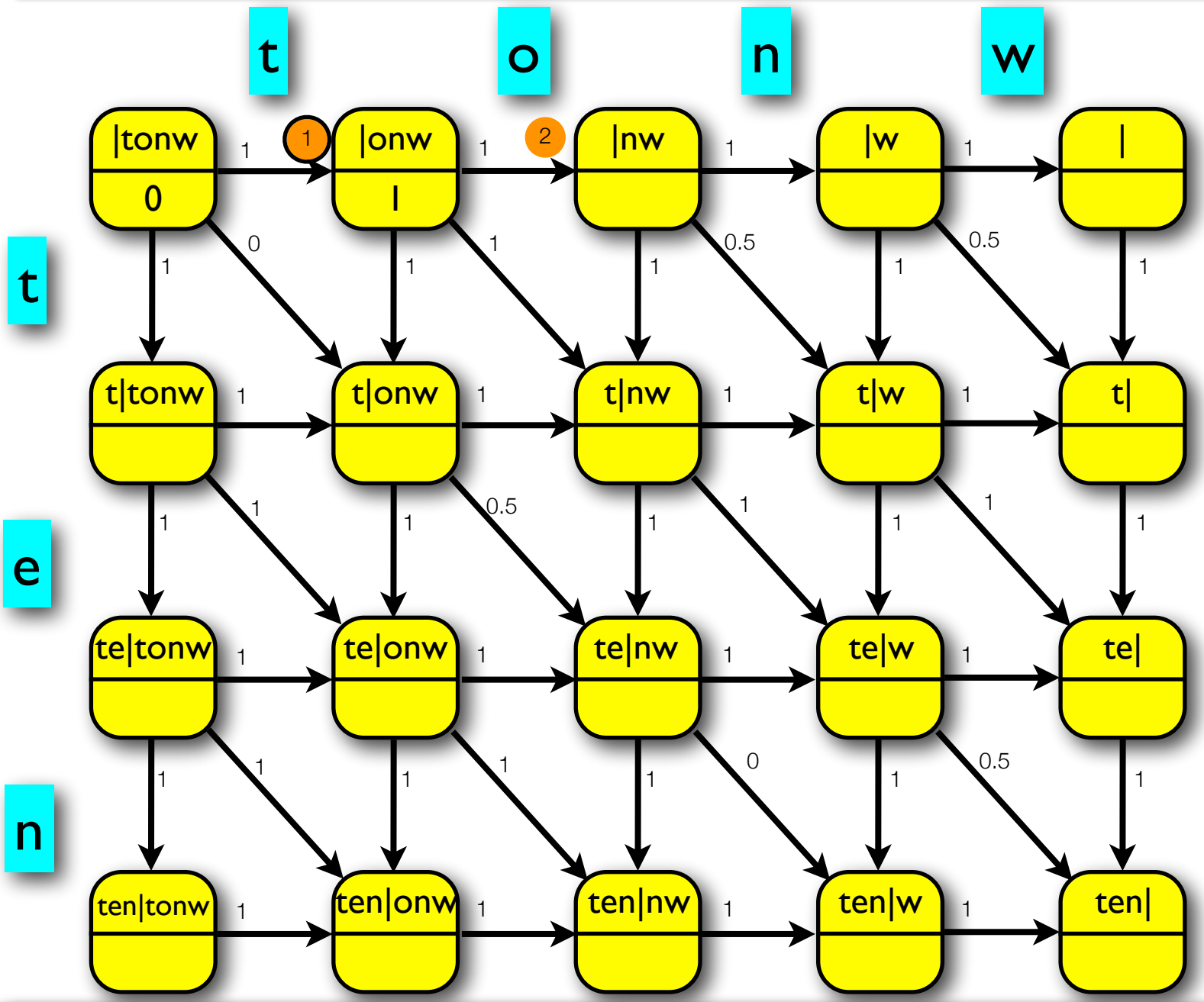
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

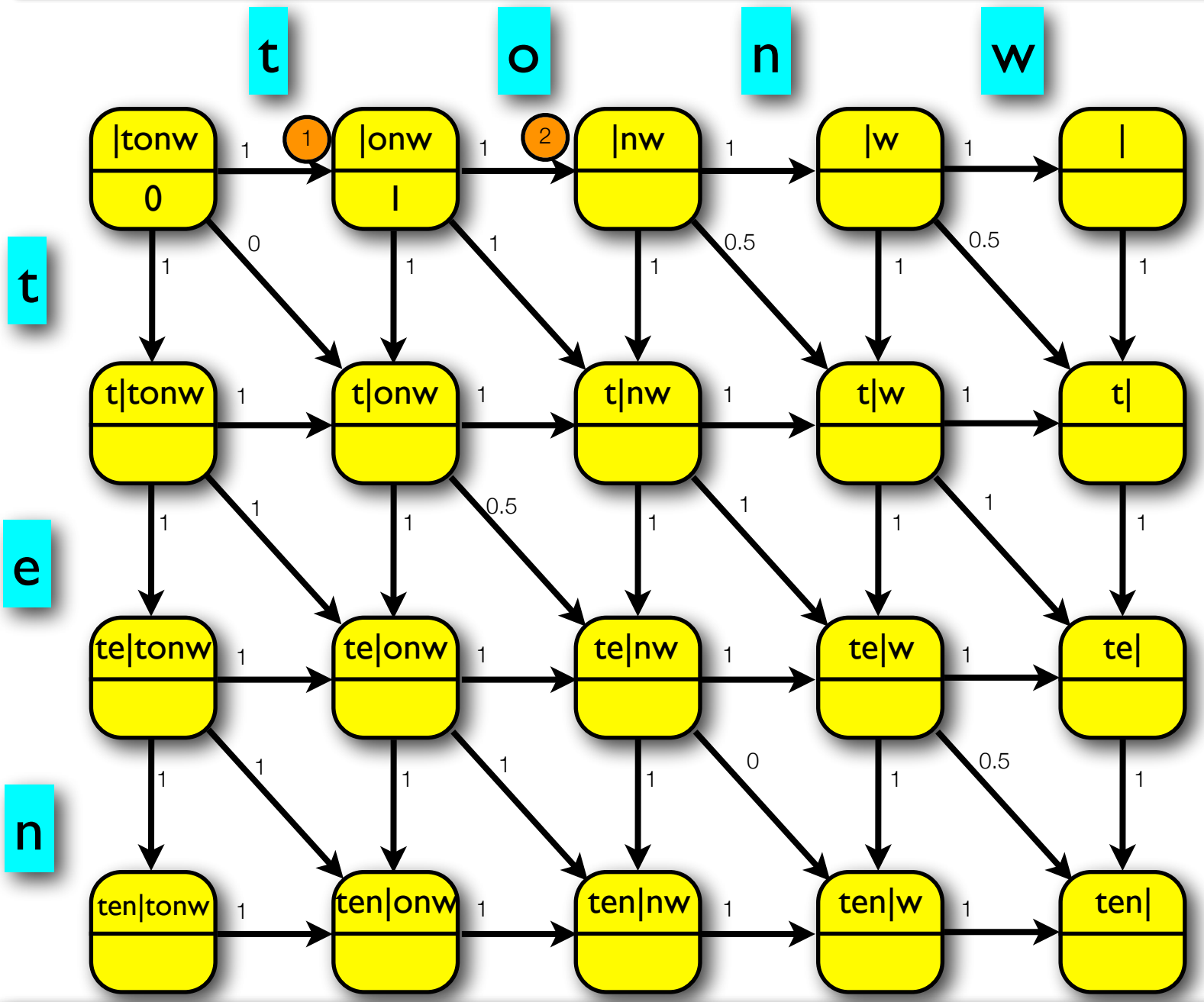
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

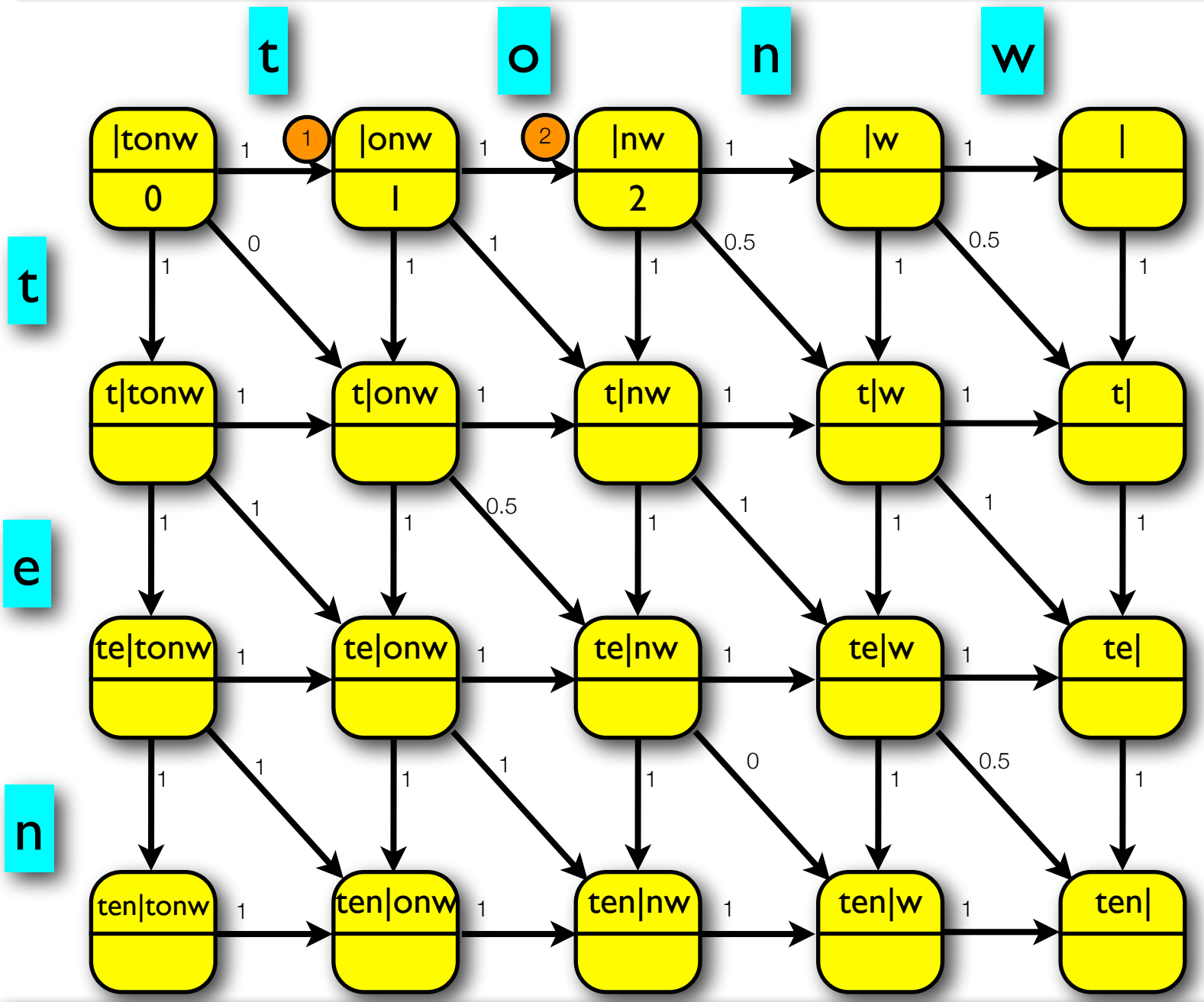
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

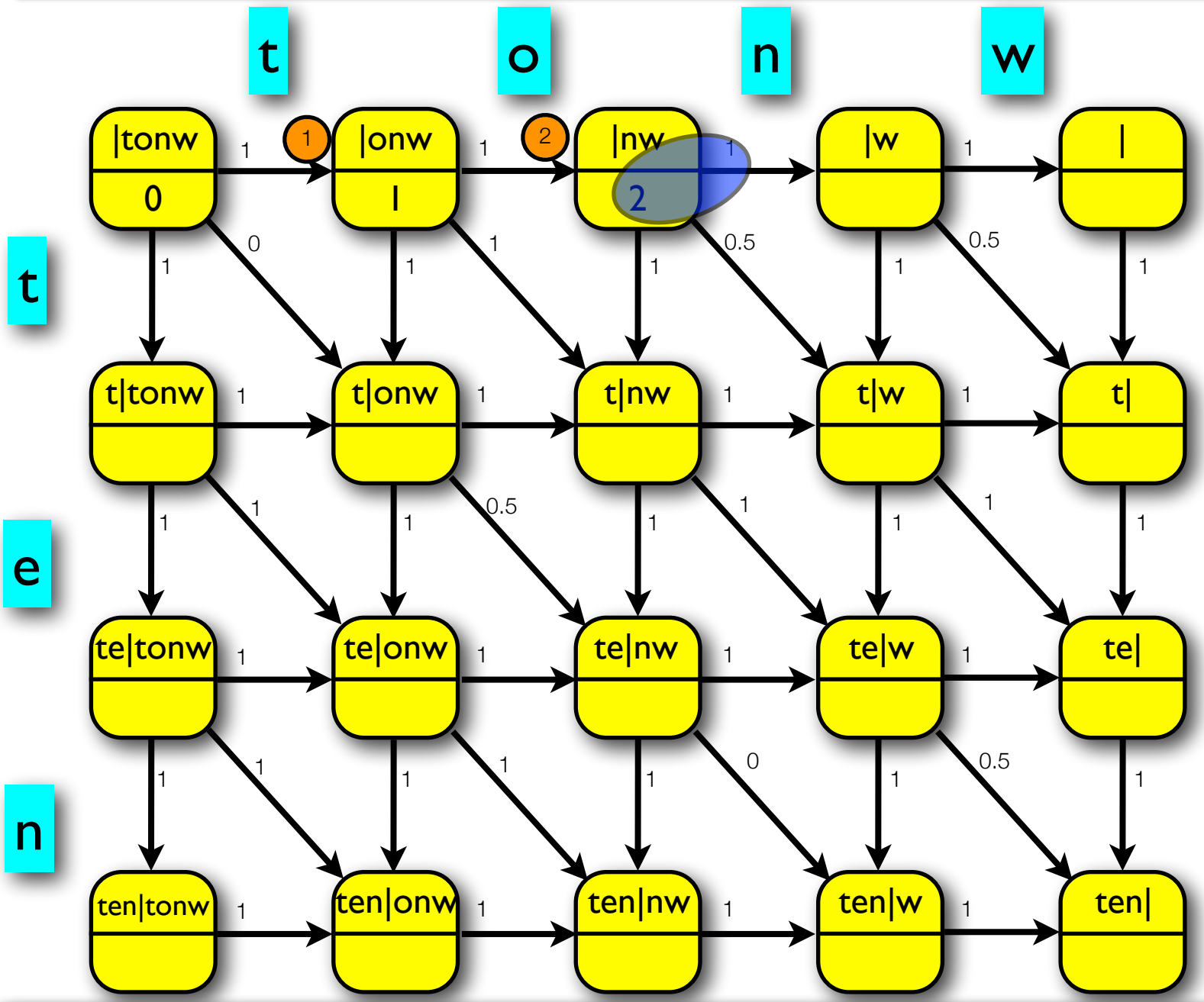
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

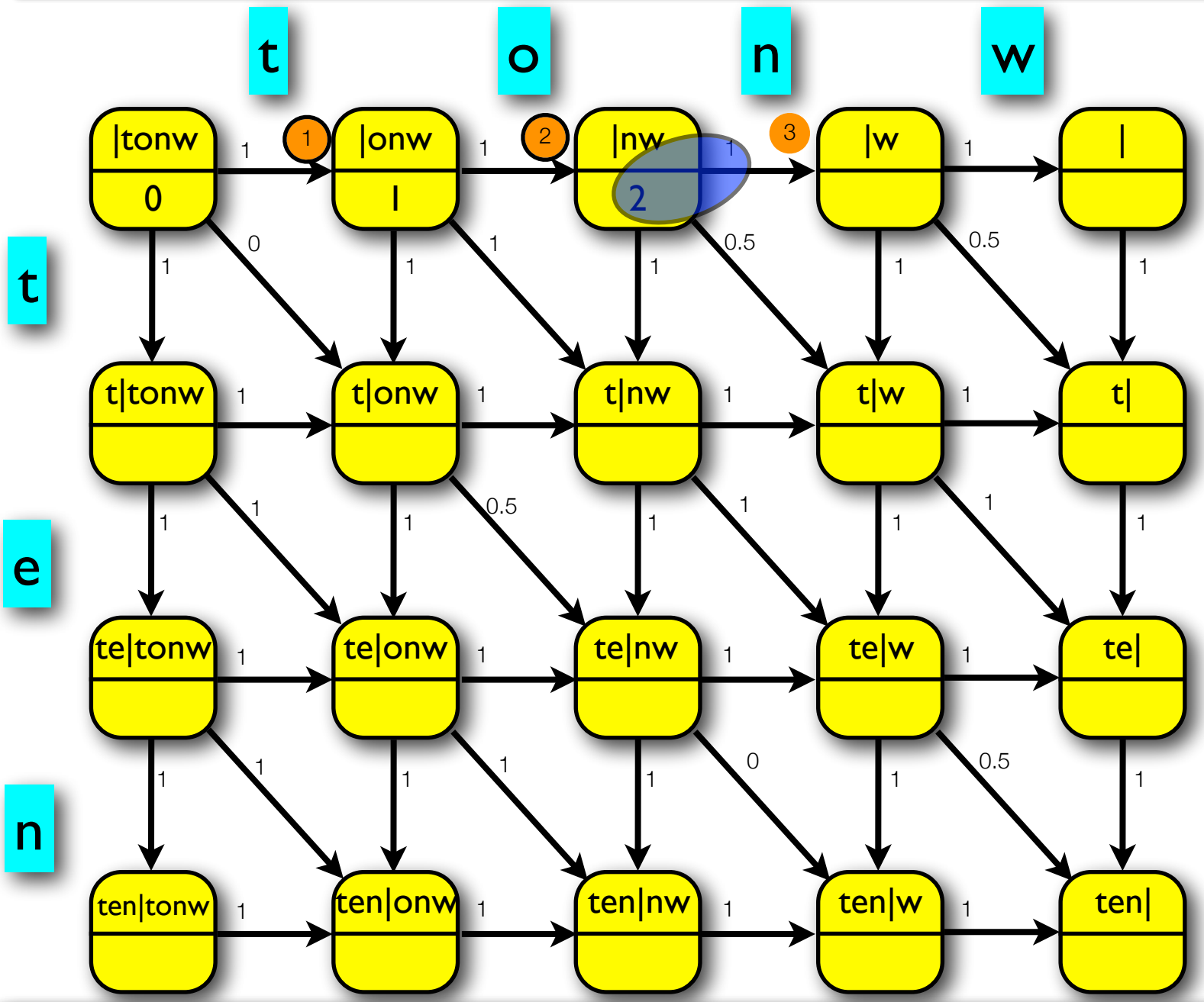
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

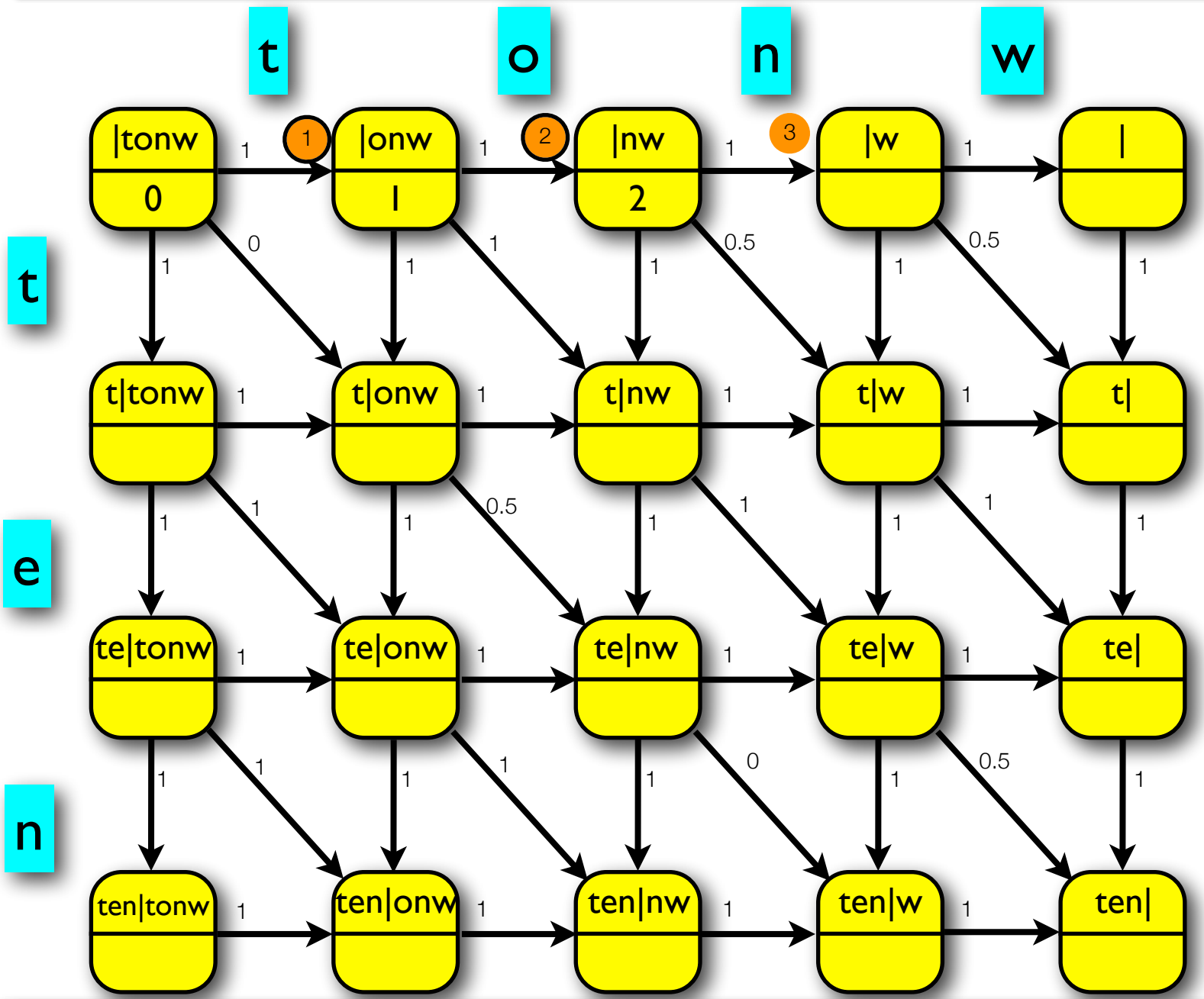
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

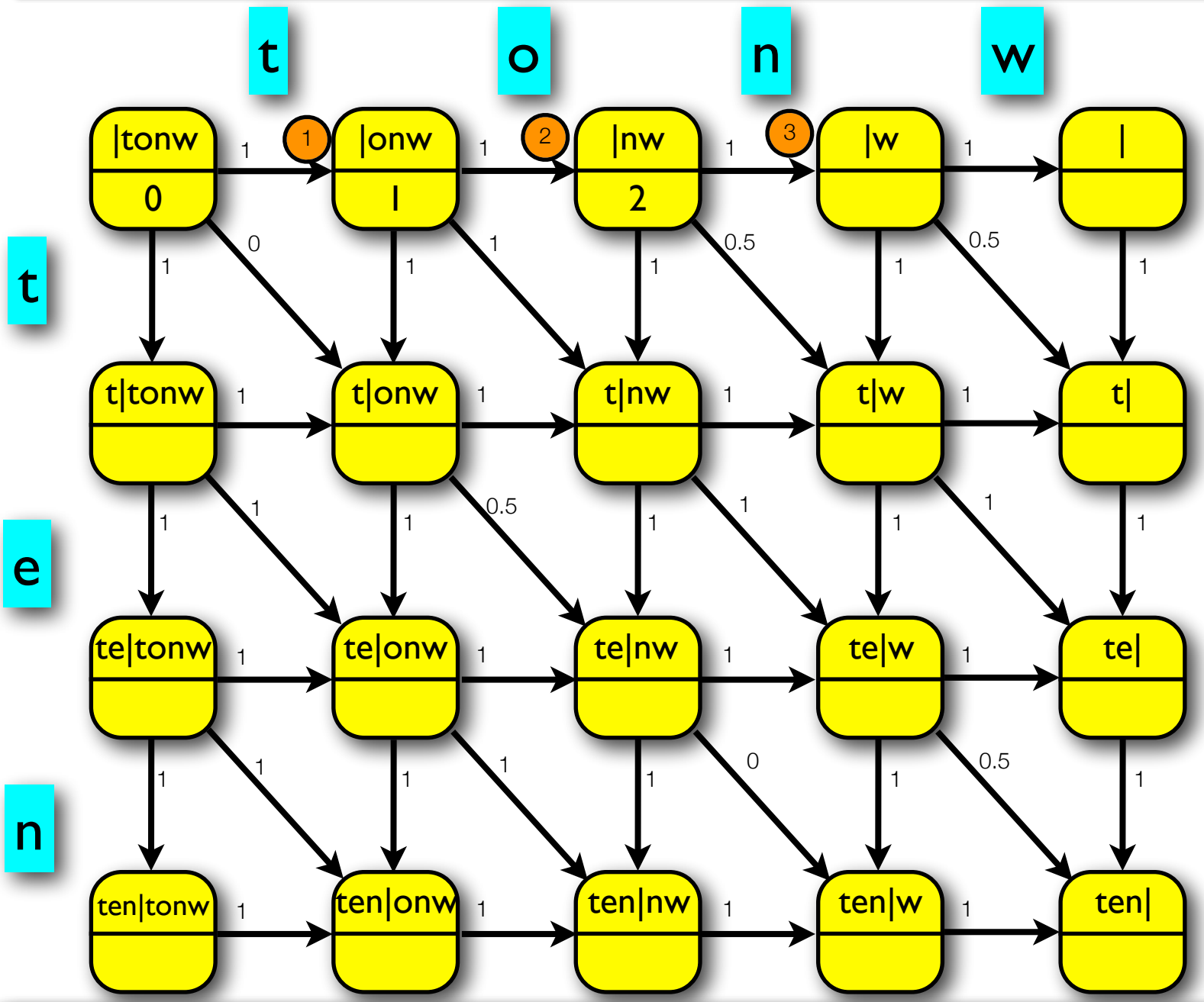
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

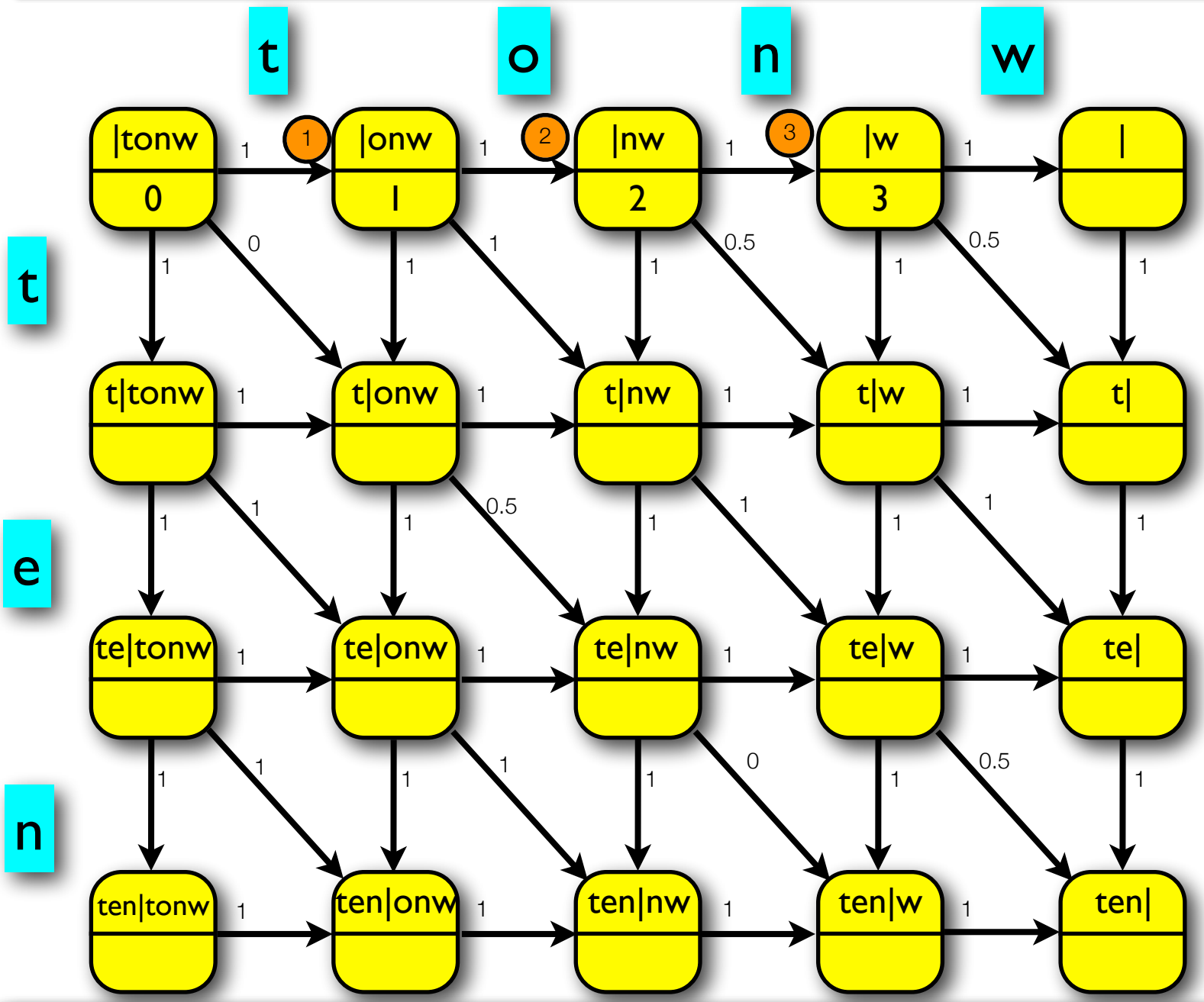
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

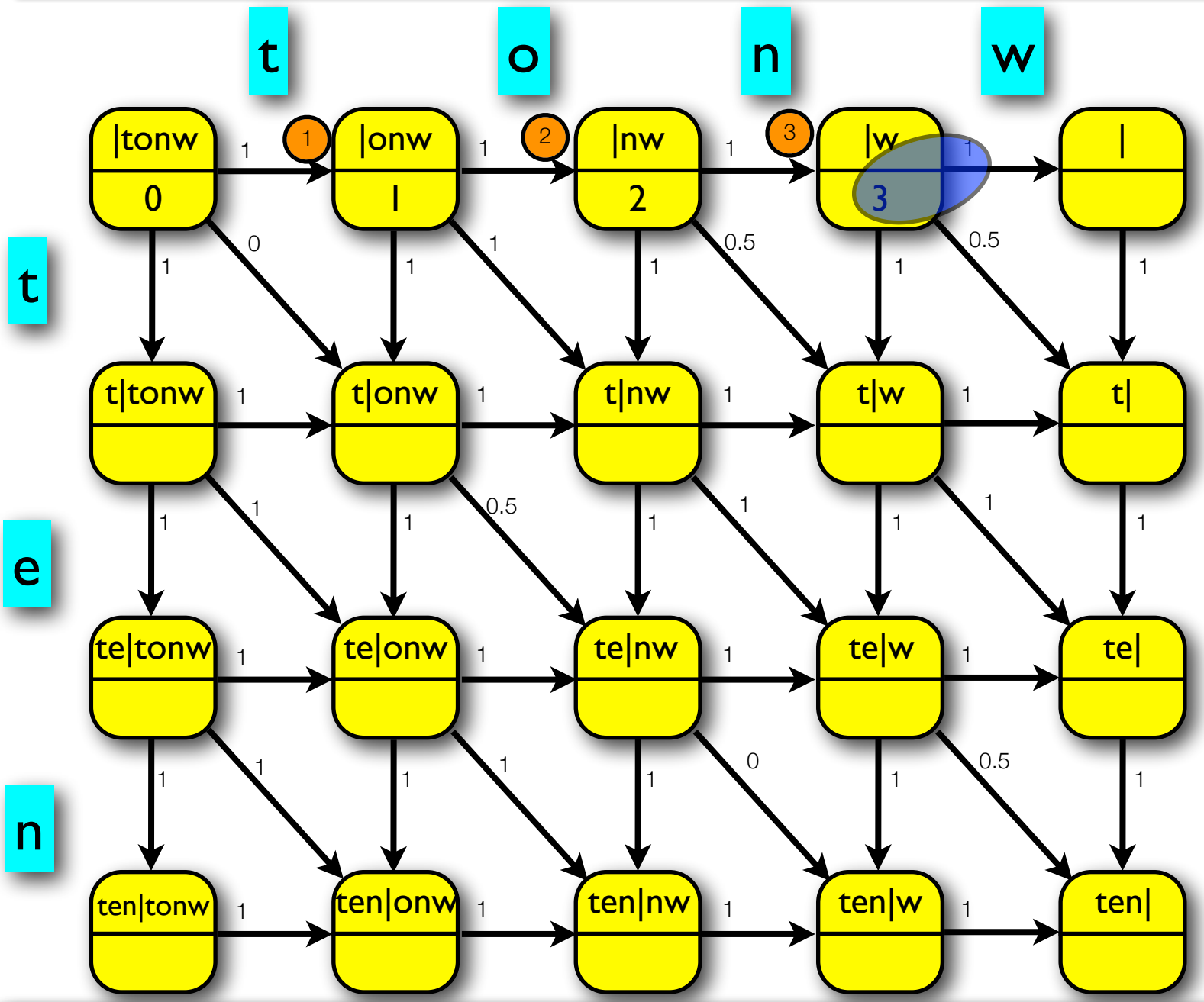
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

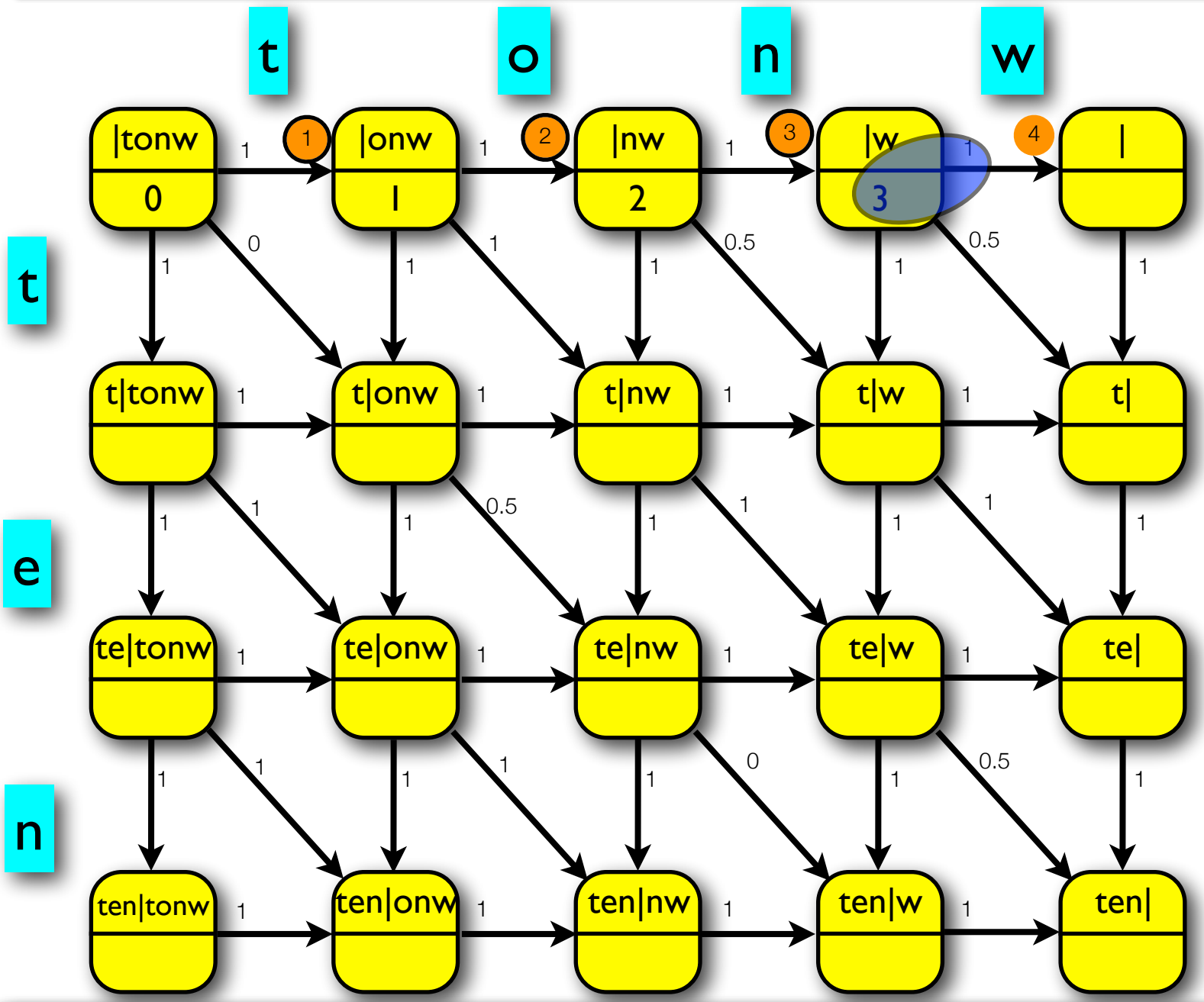
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

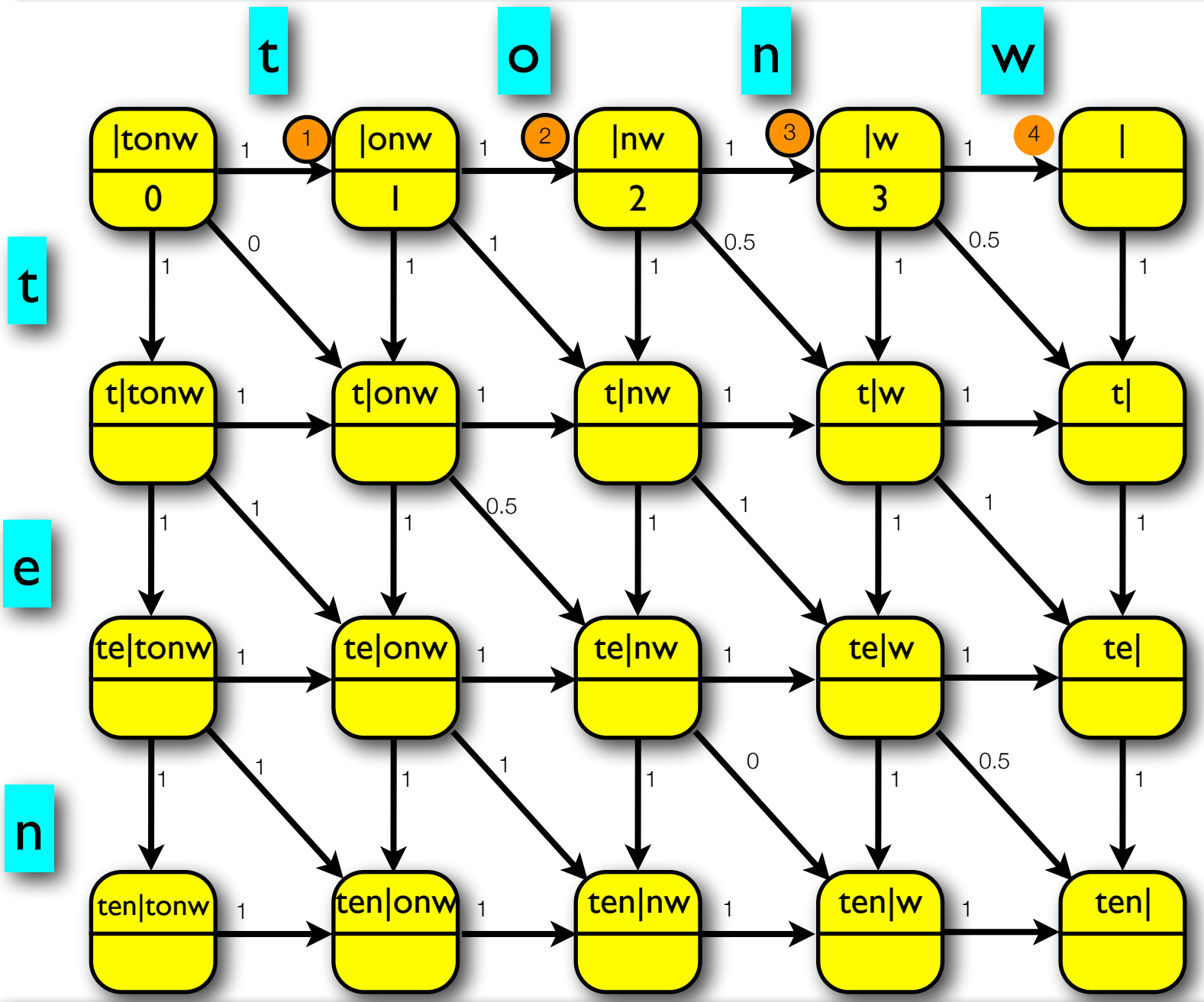
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

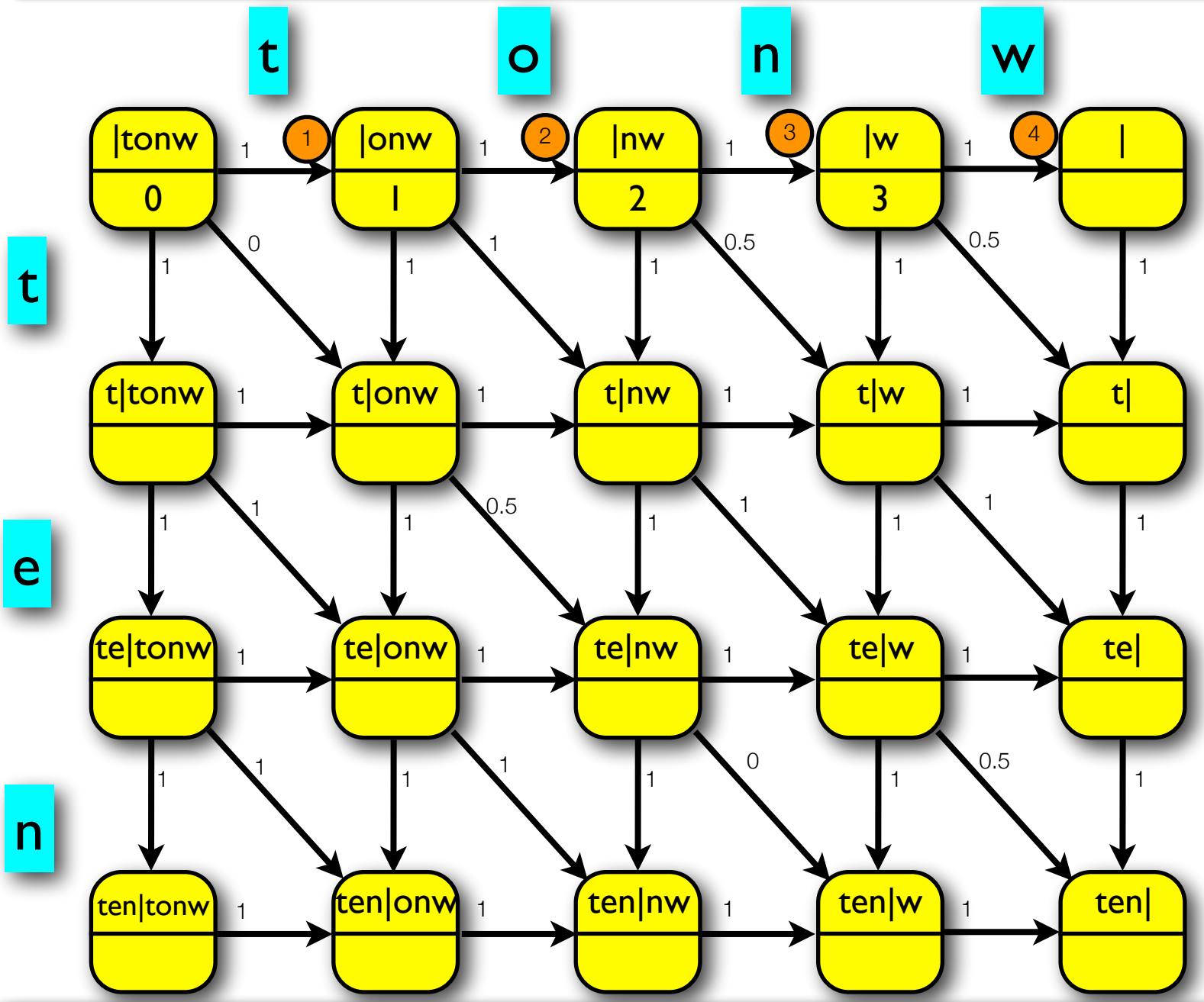
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

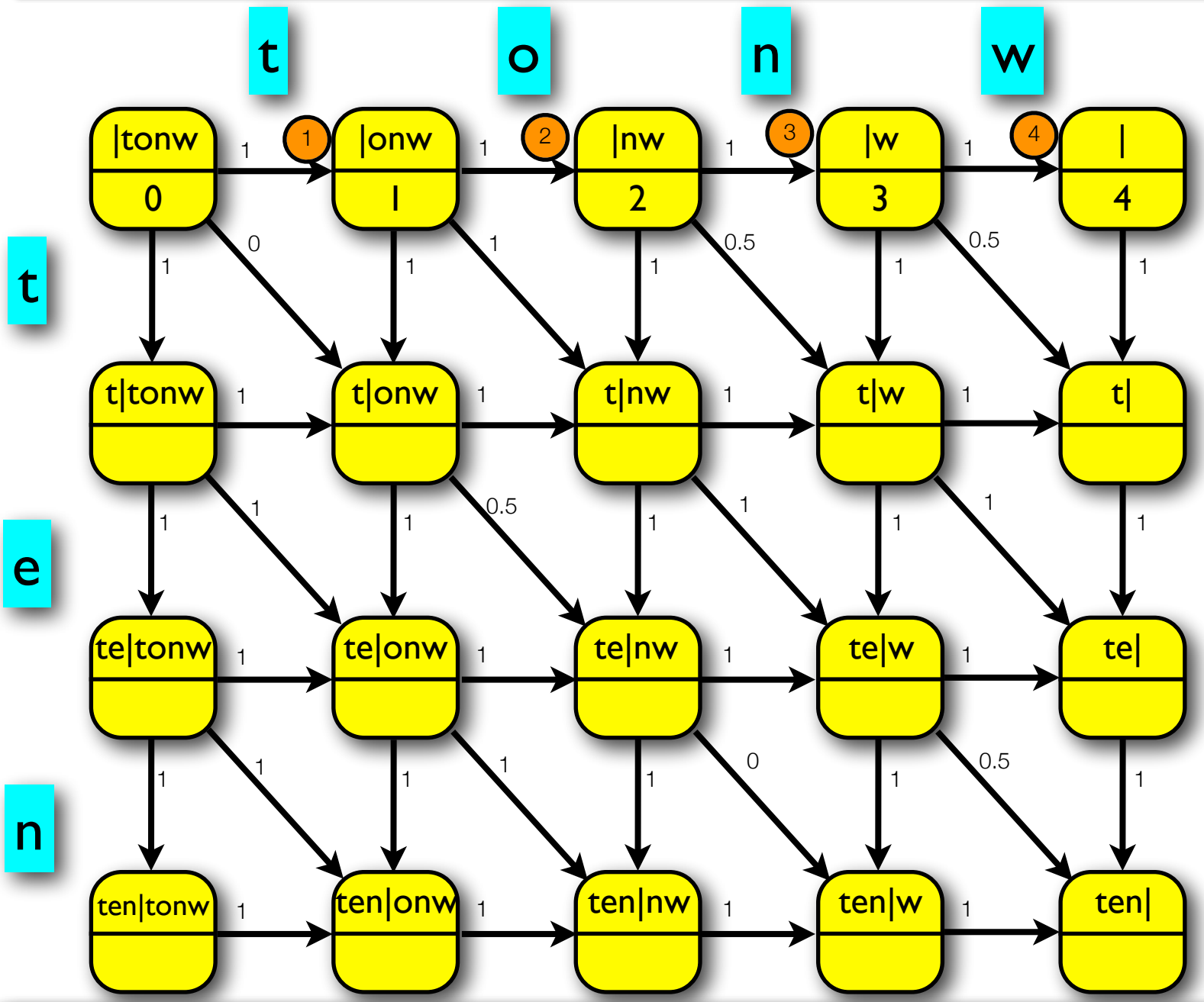
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

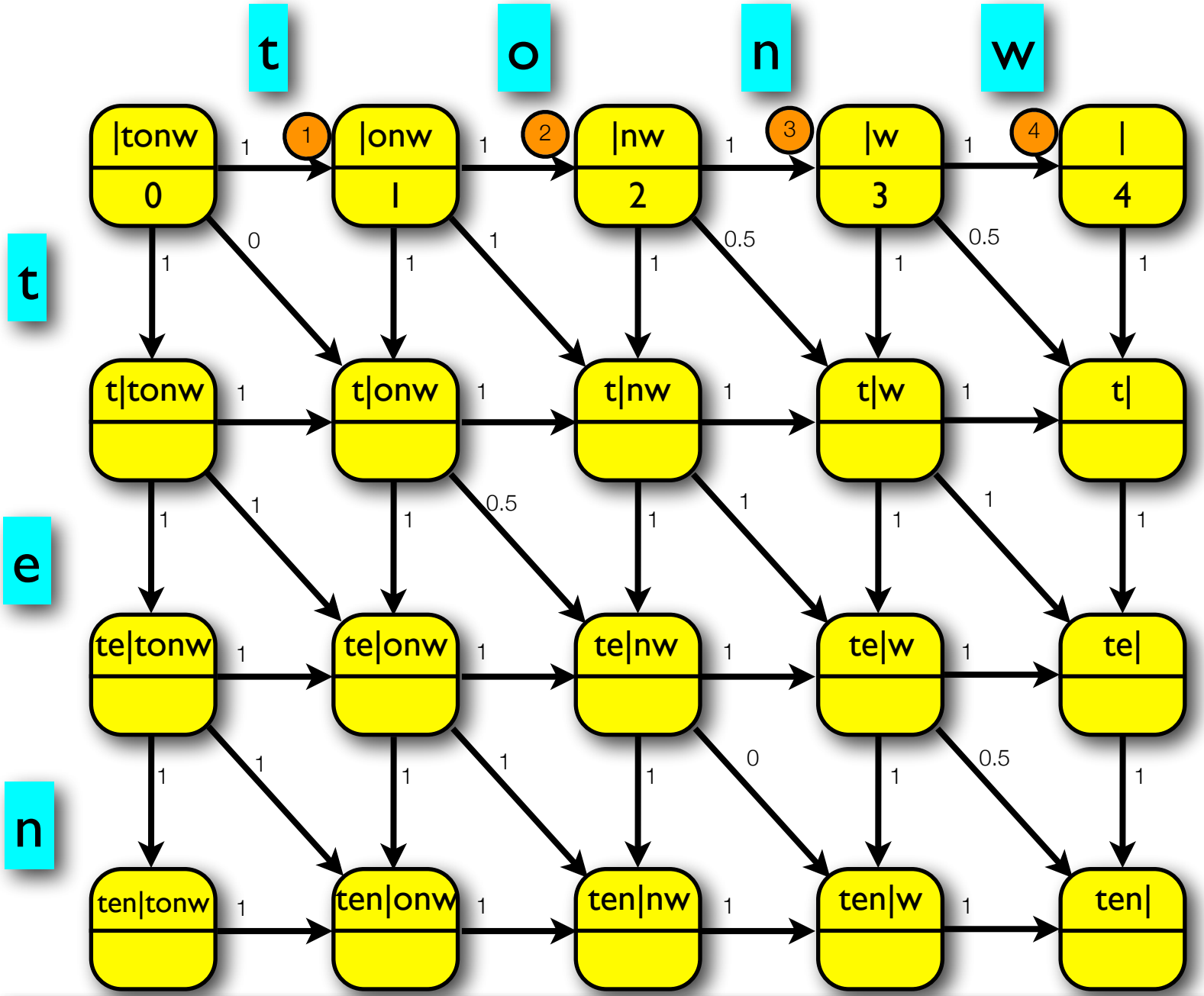
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 2: Calculate minimum edit distance for top row (easy)

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

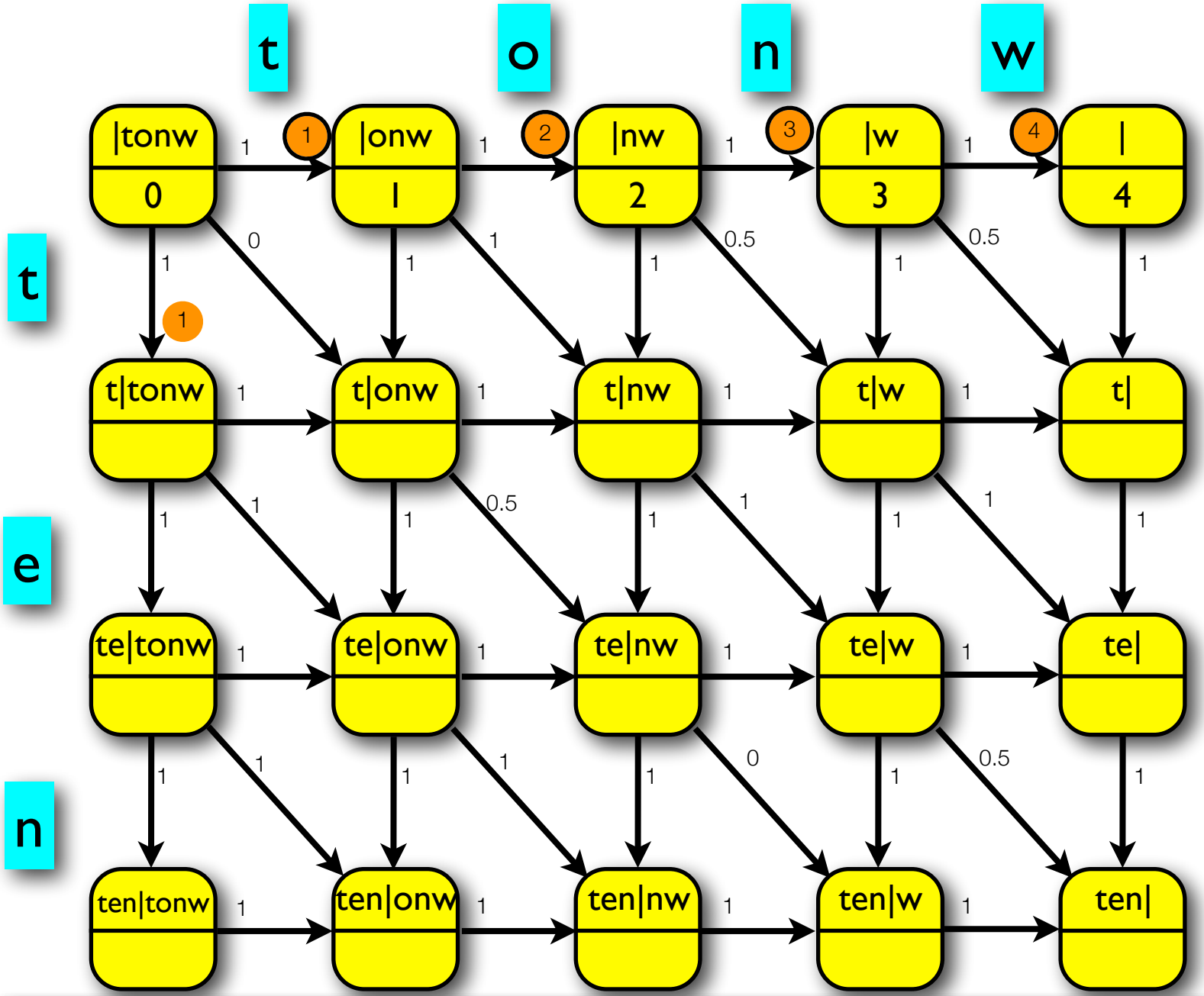
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

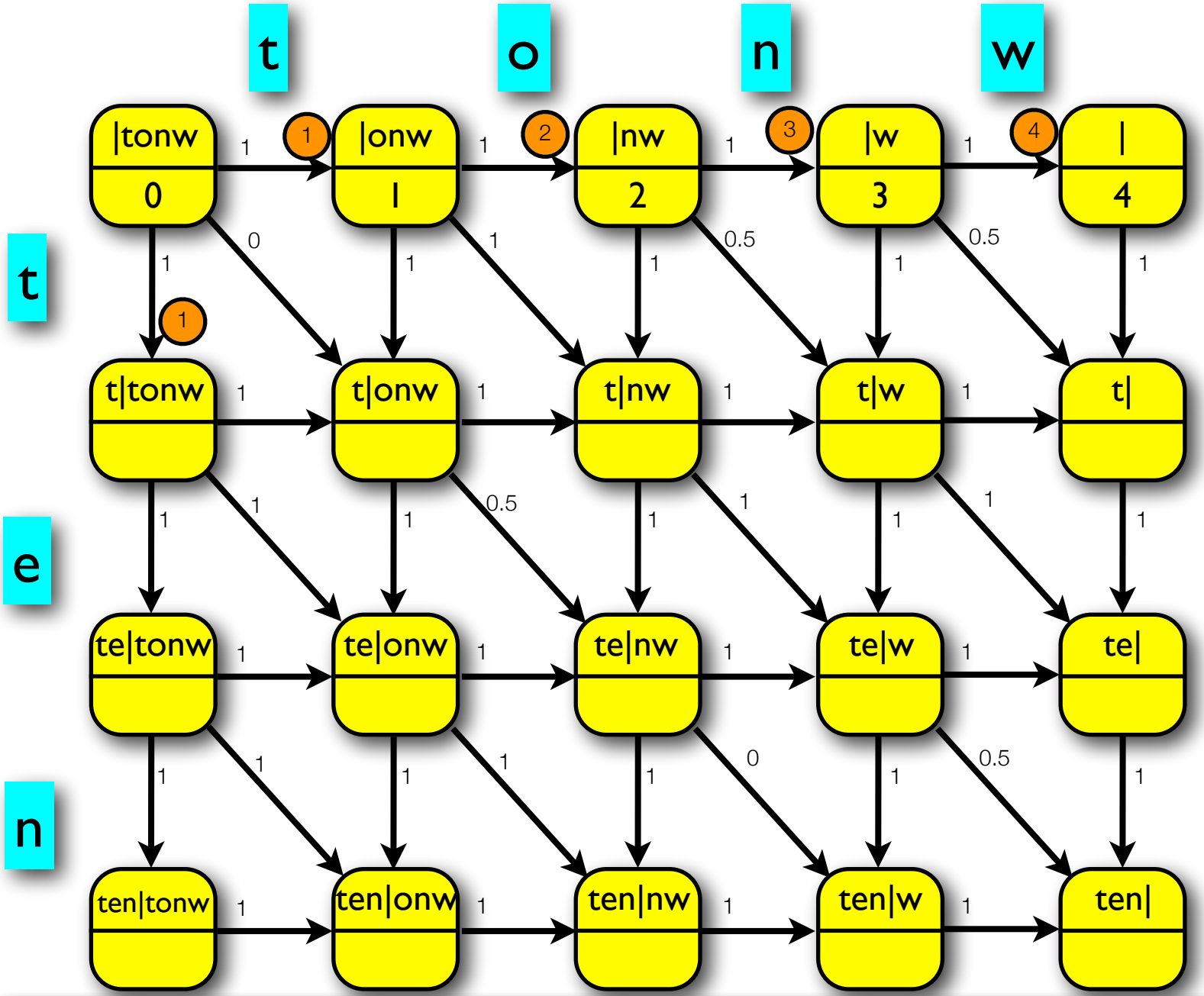
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

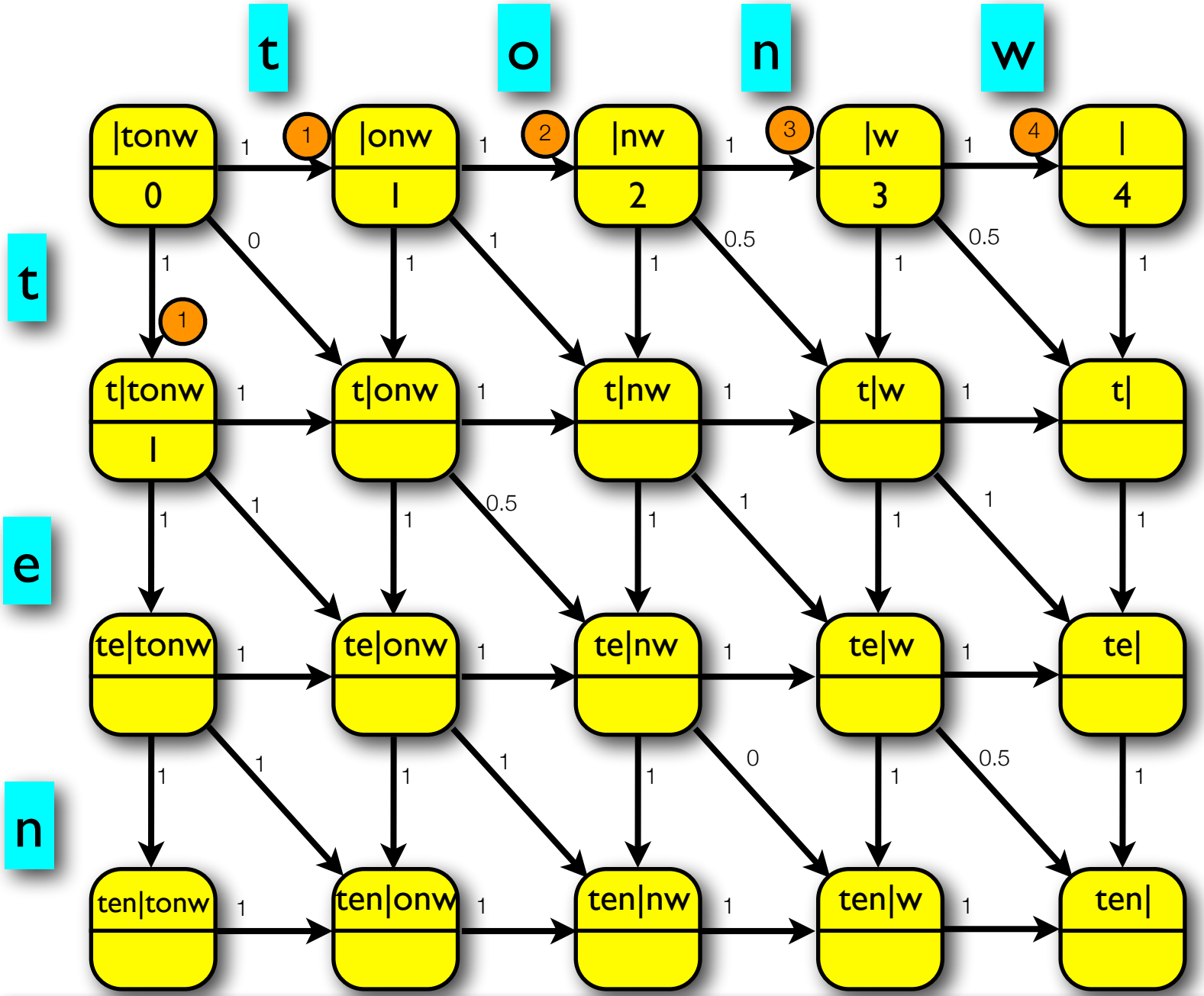
substitutions cost 0 if same letter (e.g. “t”→ “t”, “n”→ “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t”→ “e”, “e”→ “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

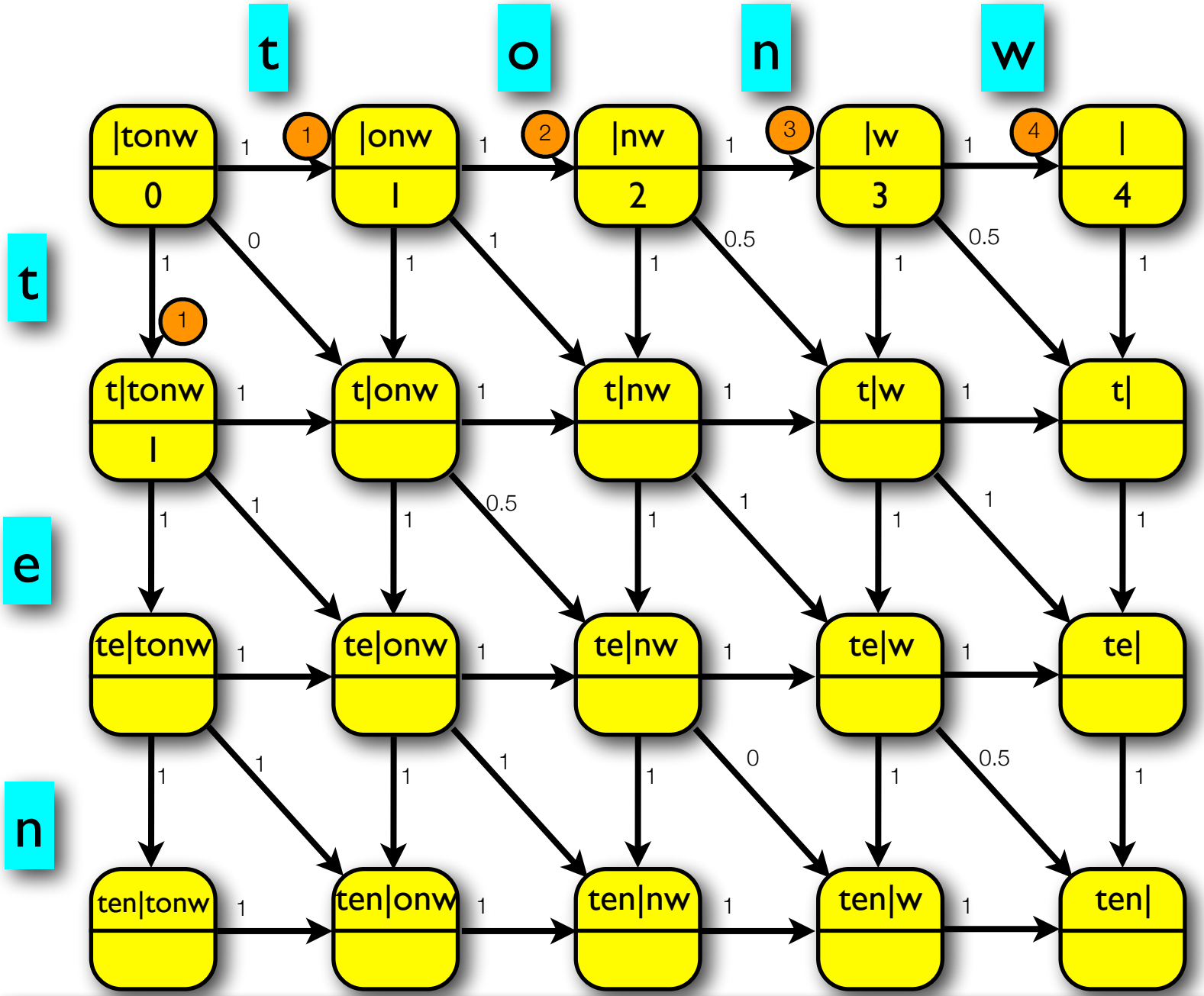
substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

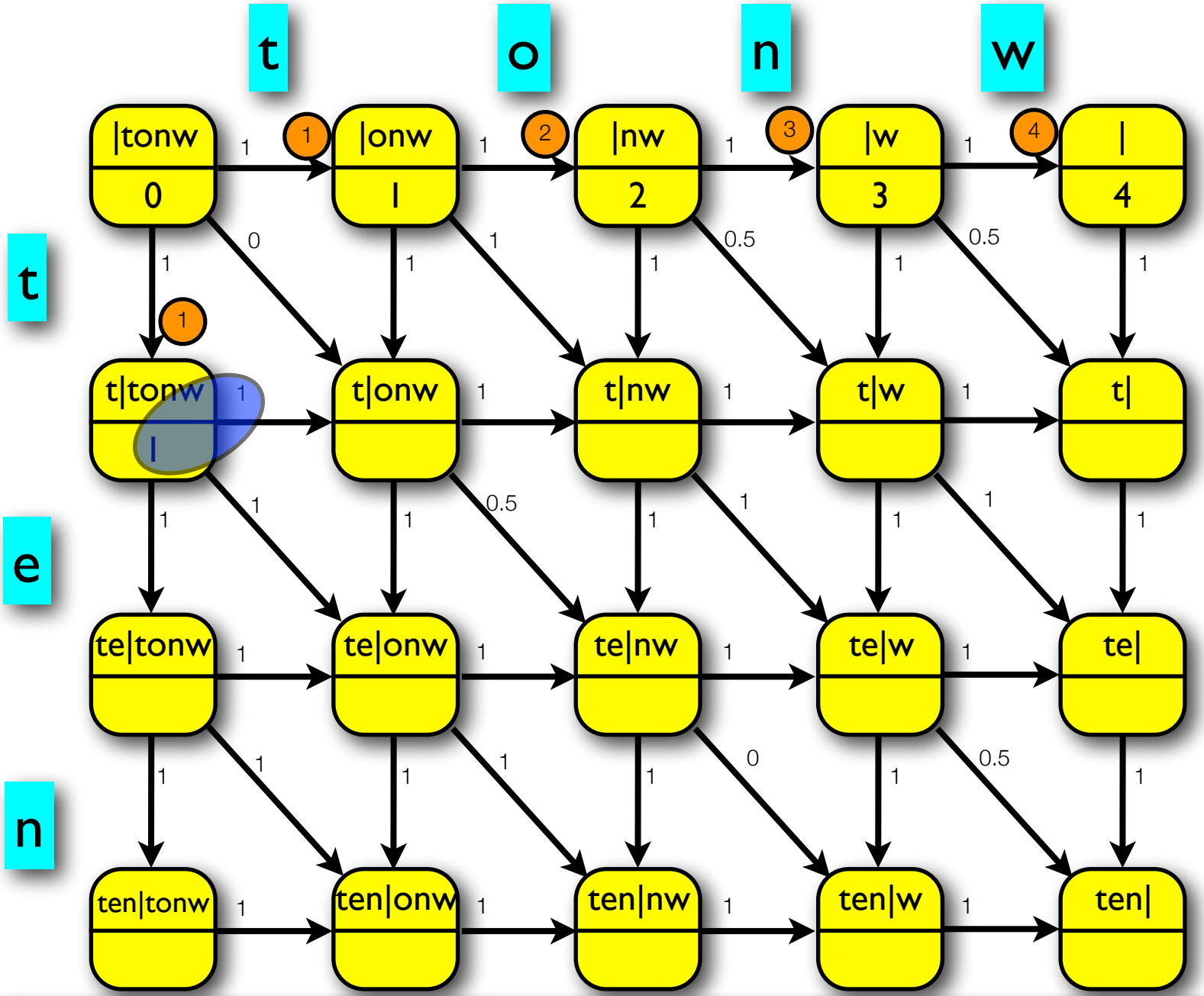
substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

The next node requires choosing one of three paths.

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t”→ “t”, “n”→ “n”)

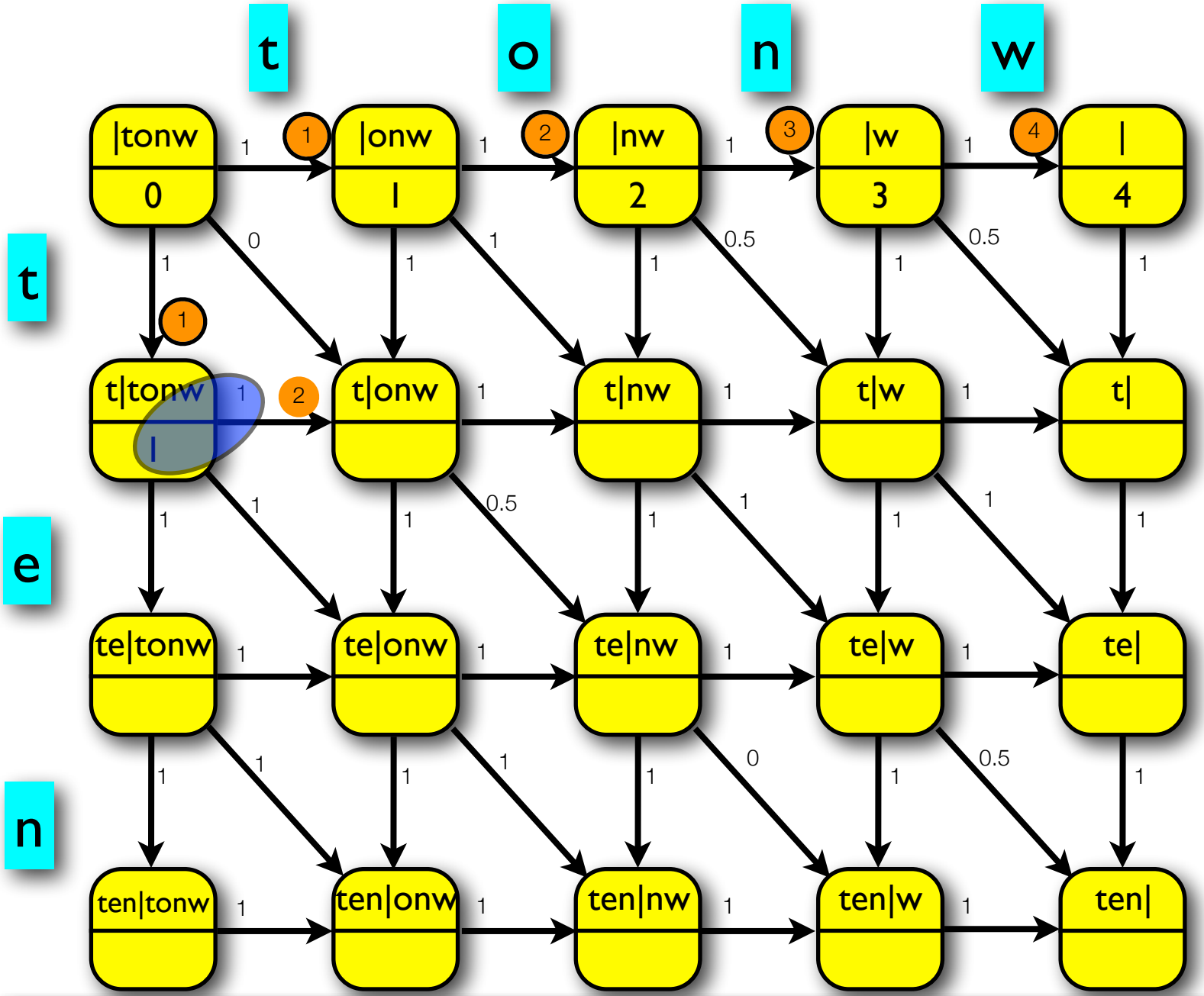
substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t”→ “e”, “e”→ “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

The next node requires choosing one of three paths.

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t”→ “t”, “n”→ “n”)

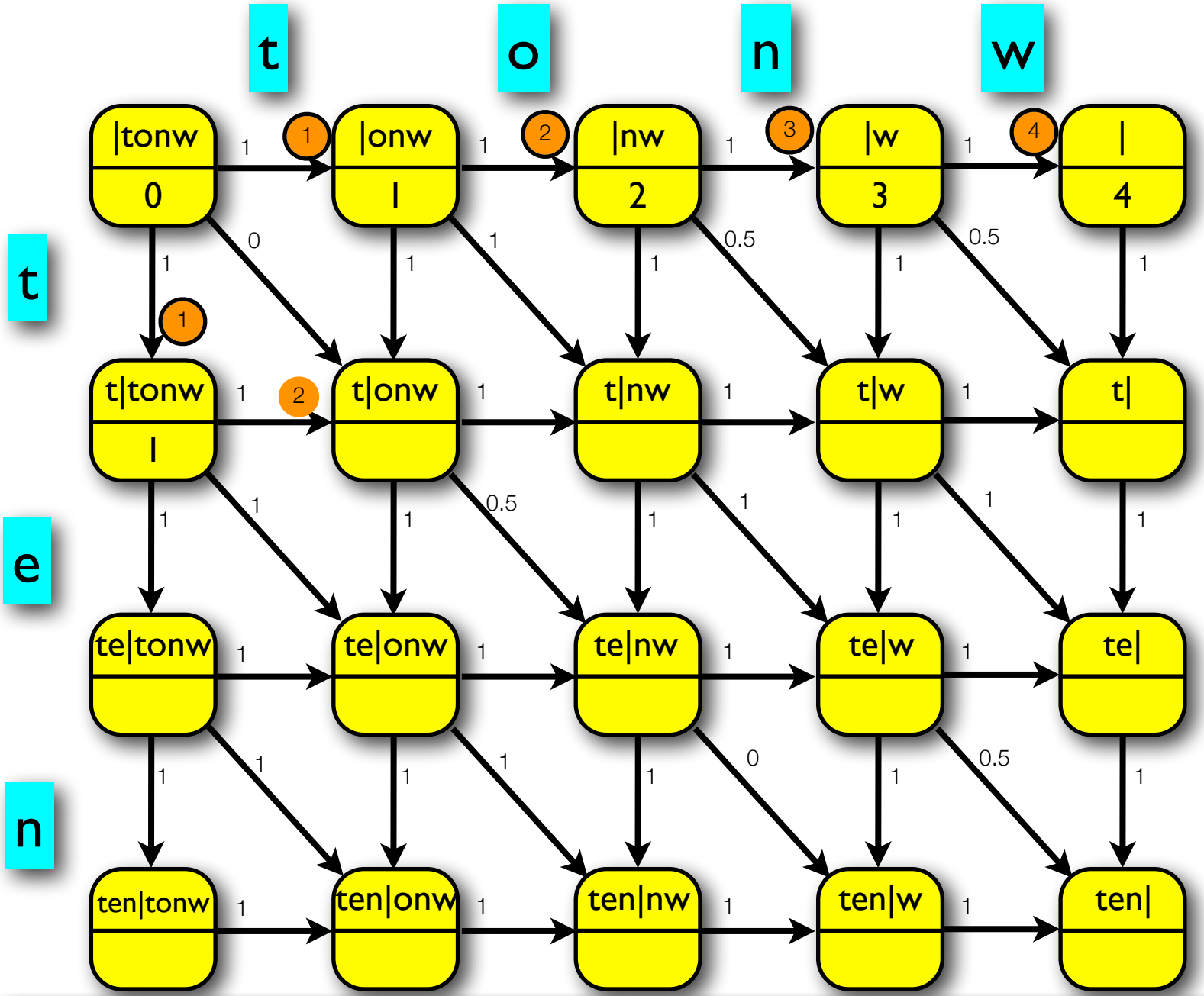
substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t”→ “e”, “e”→ “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

The next node requires choosing one of three paths.

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

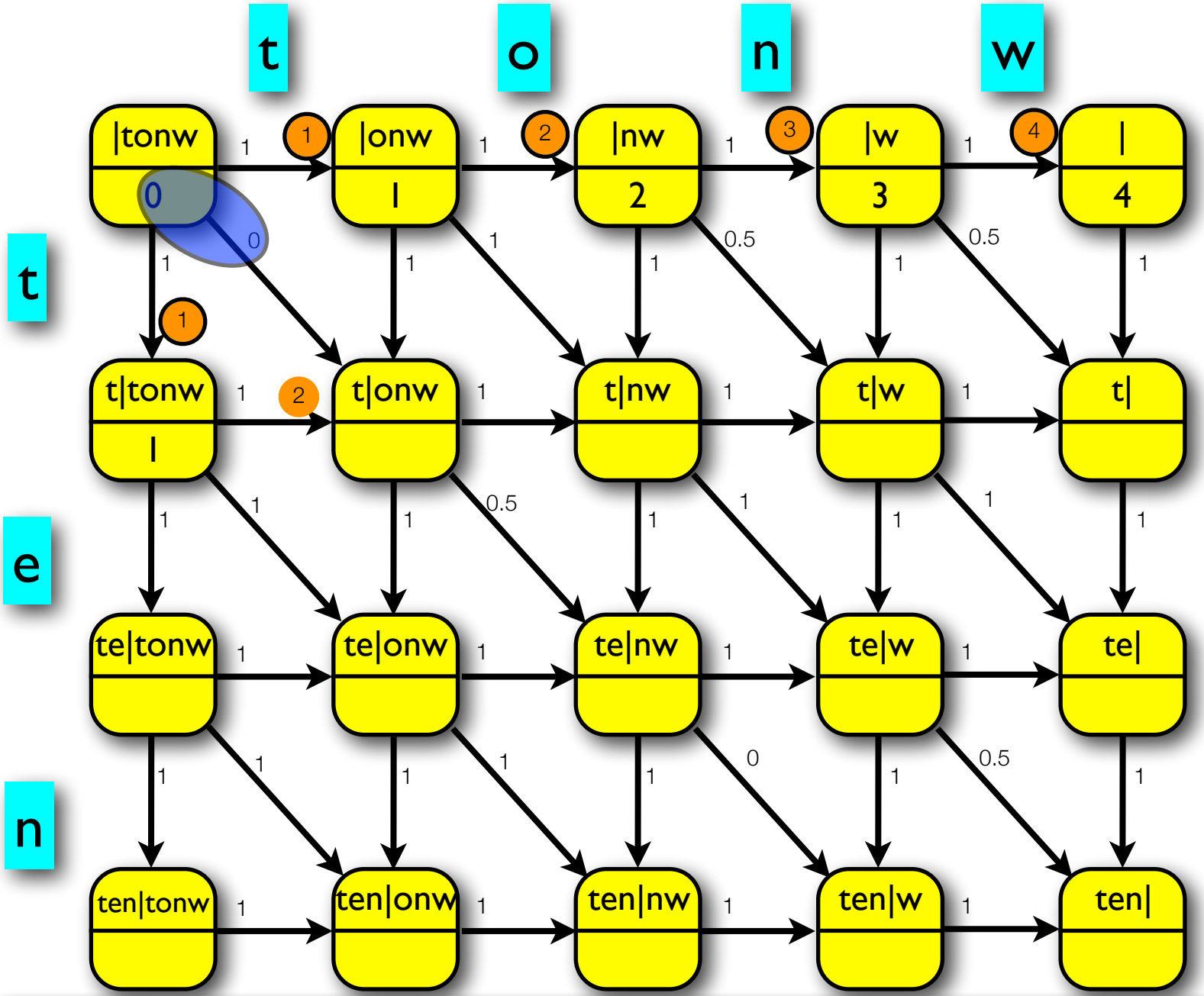
substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

The next node requires choosing one of three paths.

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t”→ “t”, “n”→ “n”)

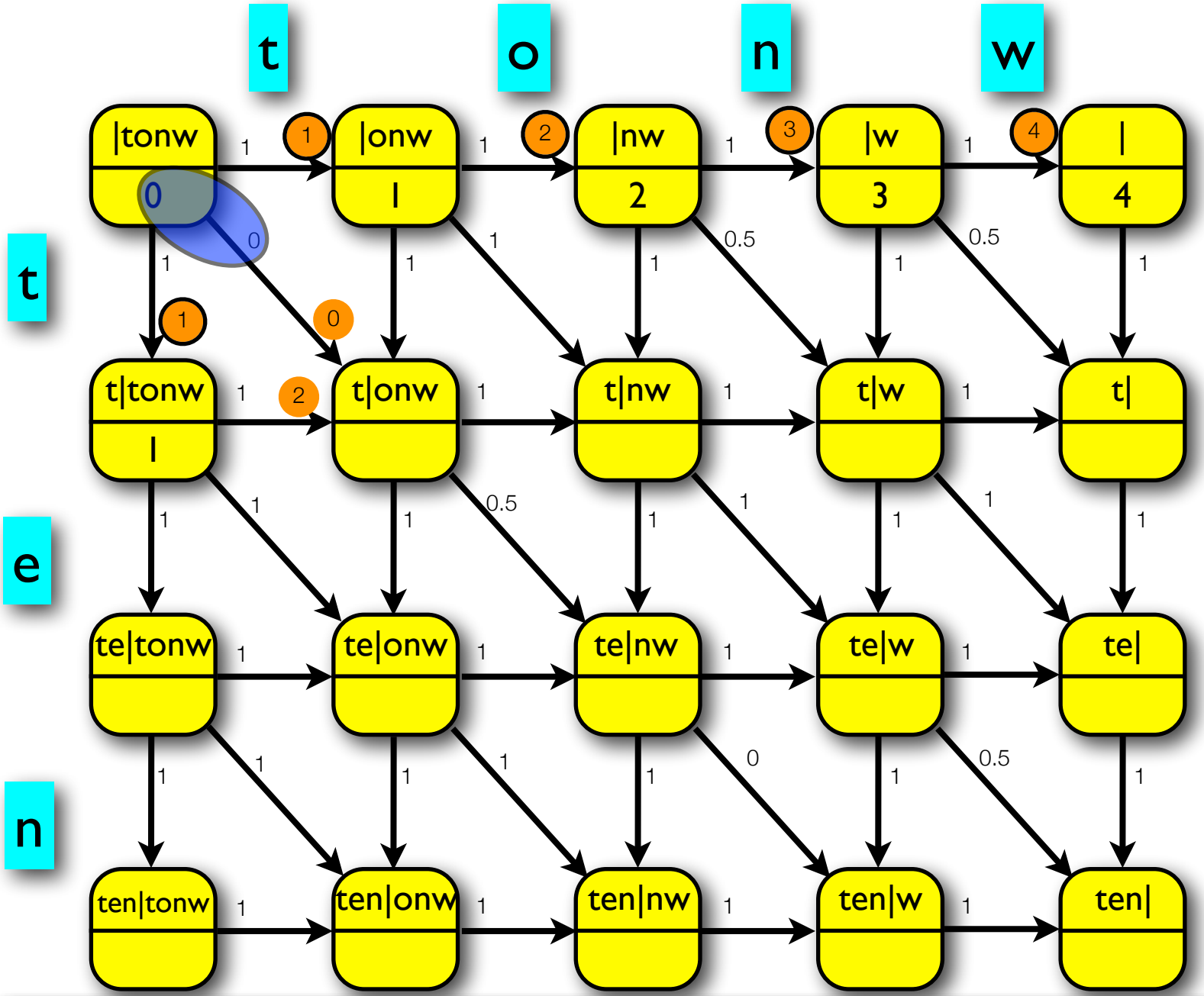
substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t”→ “e”, “e”→ “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

The next node requires choosing one of three paths.

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t”→ “t”, “n”→ “n”)

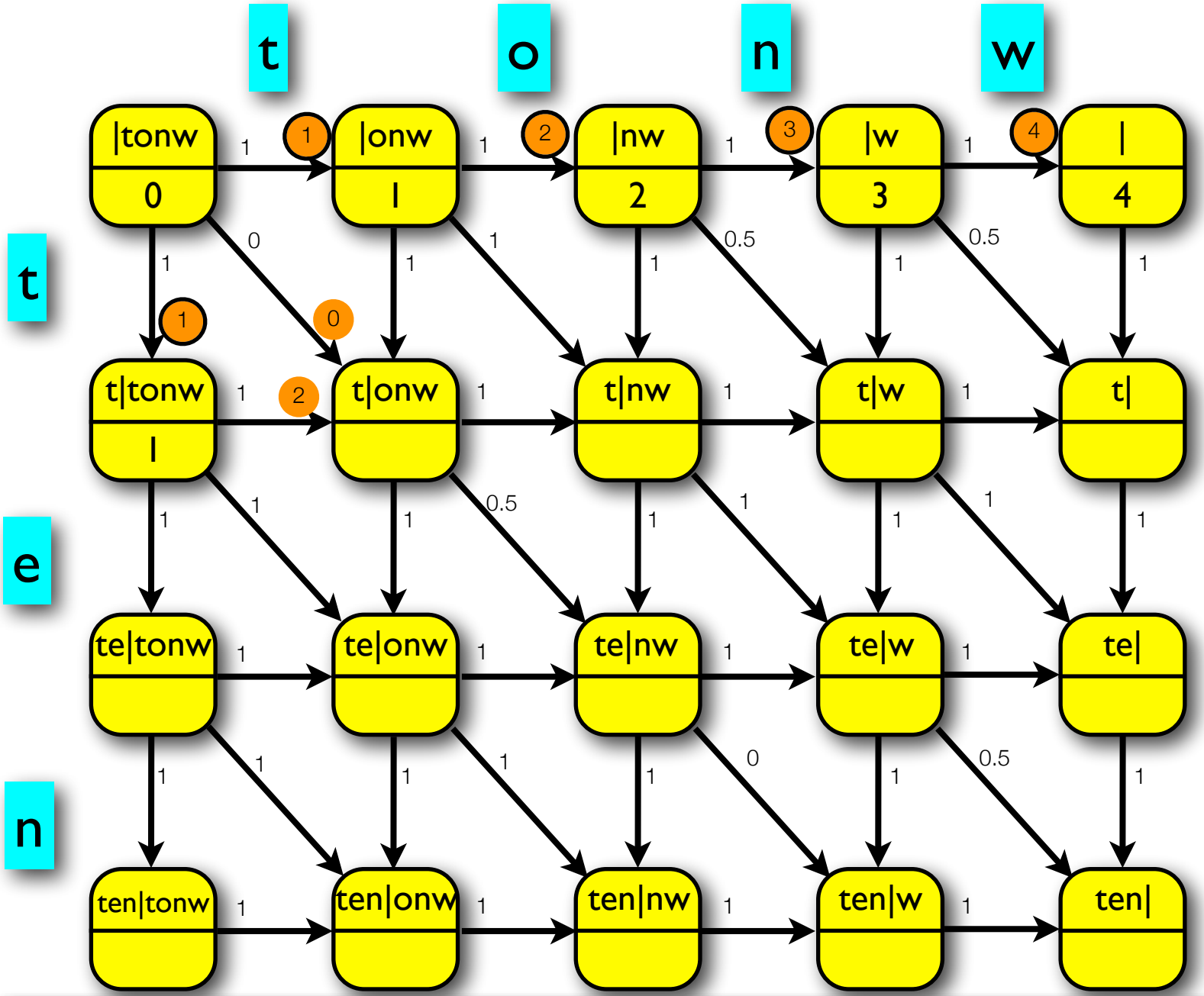
substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t”→ “e”, “e”→ “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

The next node requires choosing one of three paths.

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t”→ “t”, “n”→ “n”)

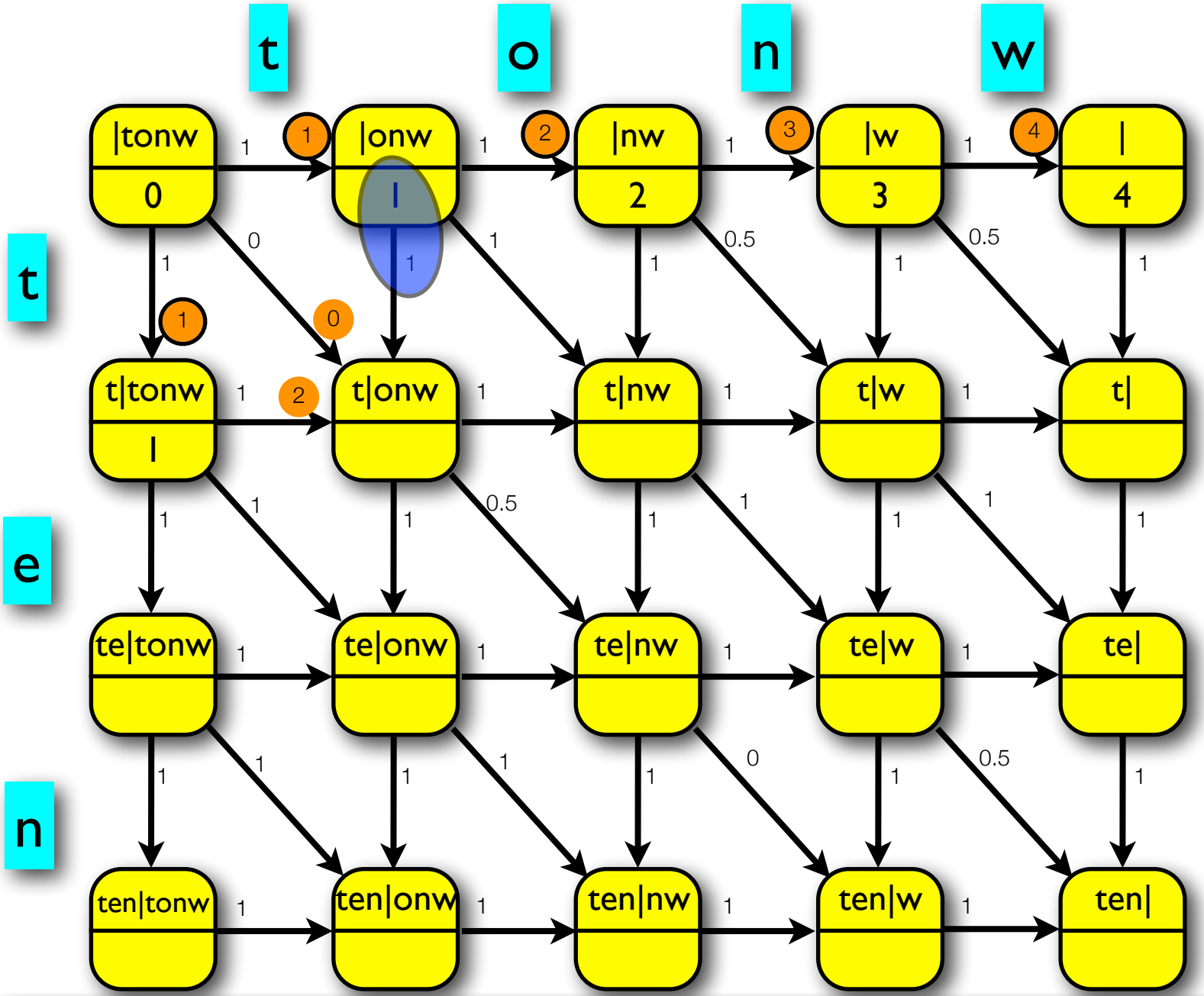
substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t”→ “e”, “e”→ “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

The next node requires choosing one of three paths.

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t”→ “t”, “n”→ “n”)

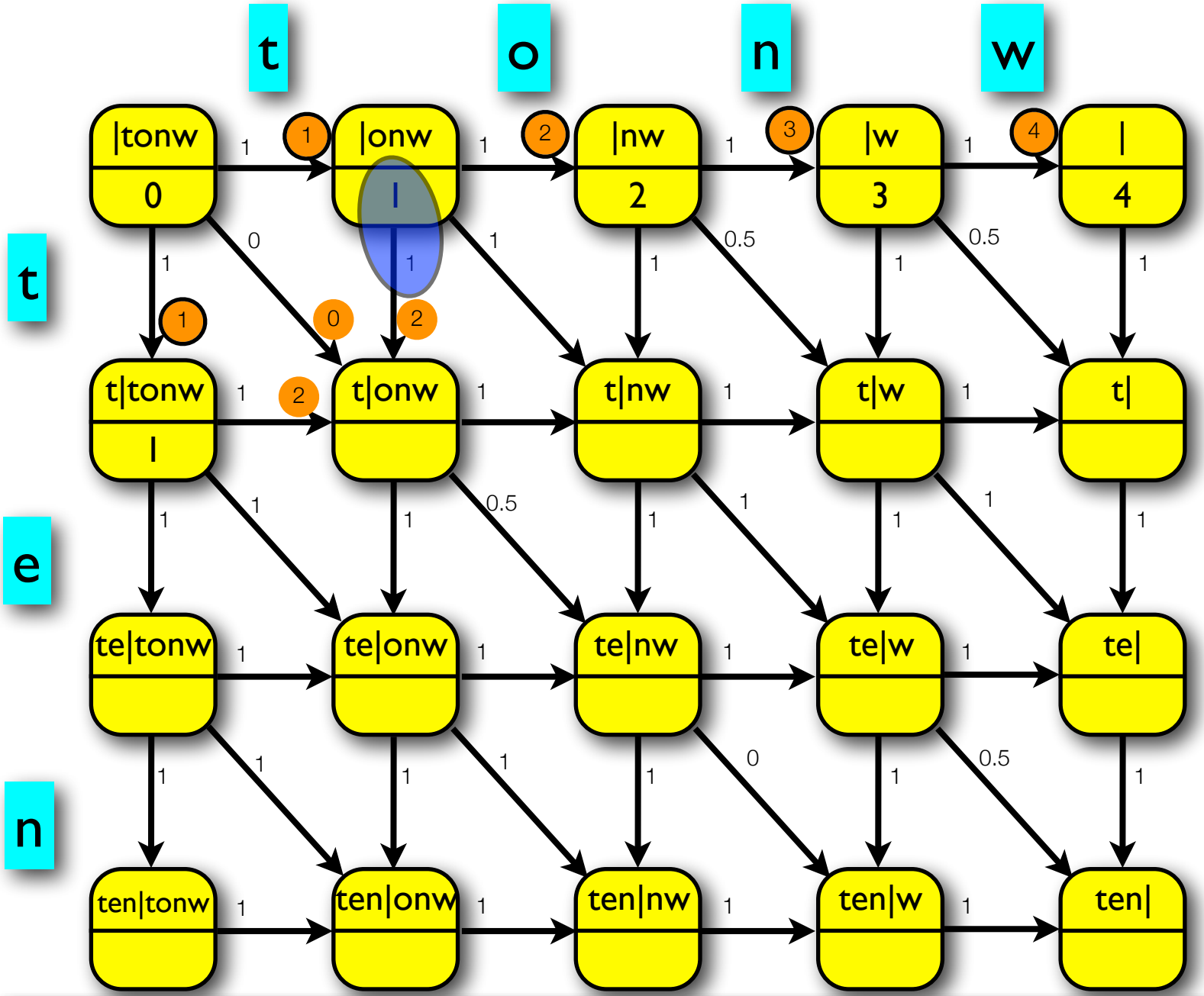
substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t”→ “e”, “e”→ “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

The next node requires choosing one of three paths.

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t”→ “t”, “n”→ “n”)

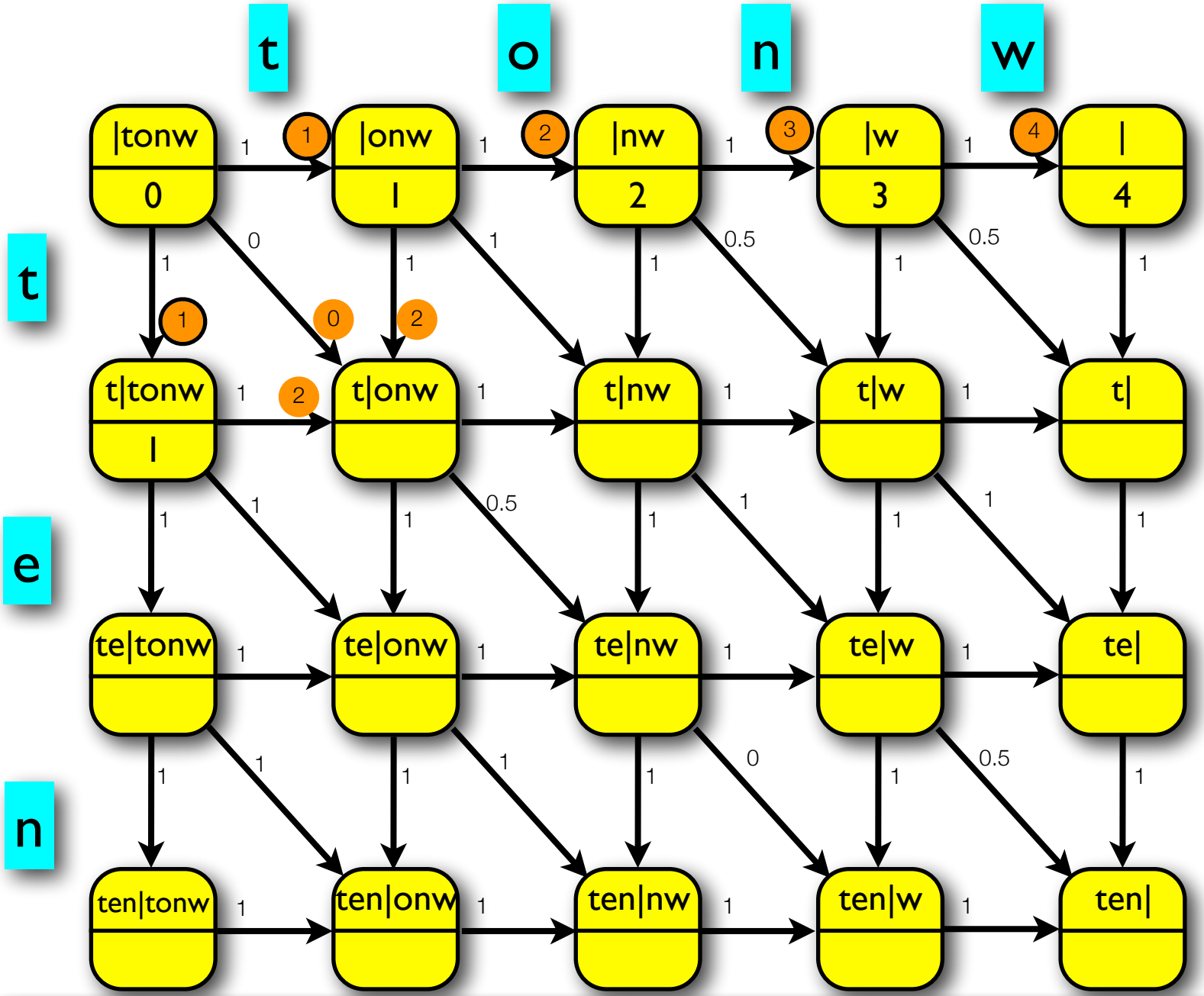
substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t”→ “e”, “e”→ “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

The next node requires choosing one of three paths.

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

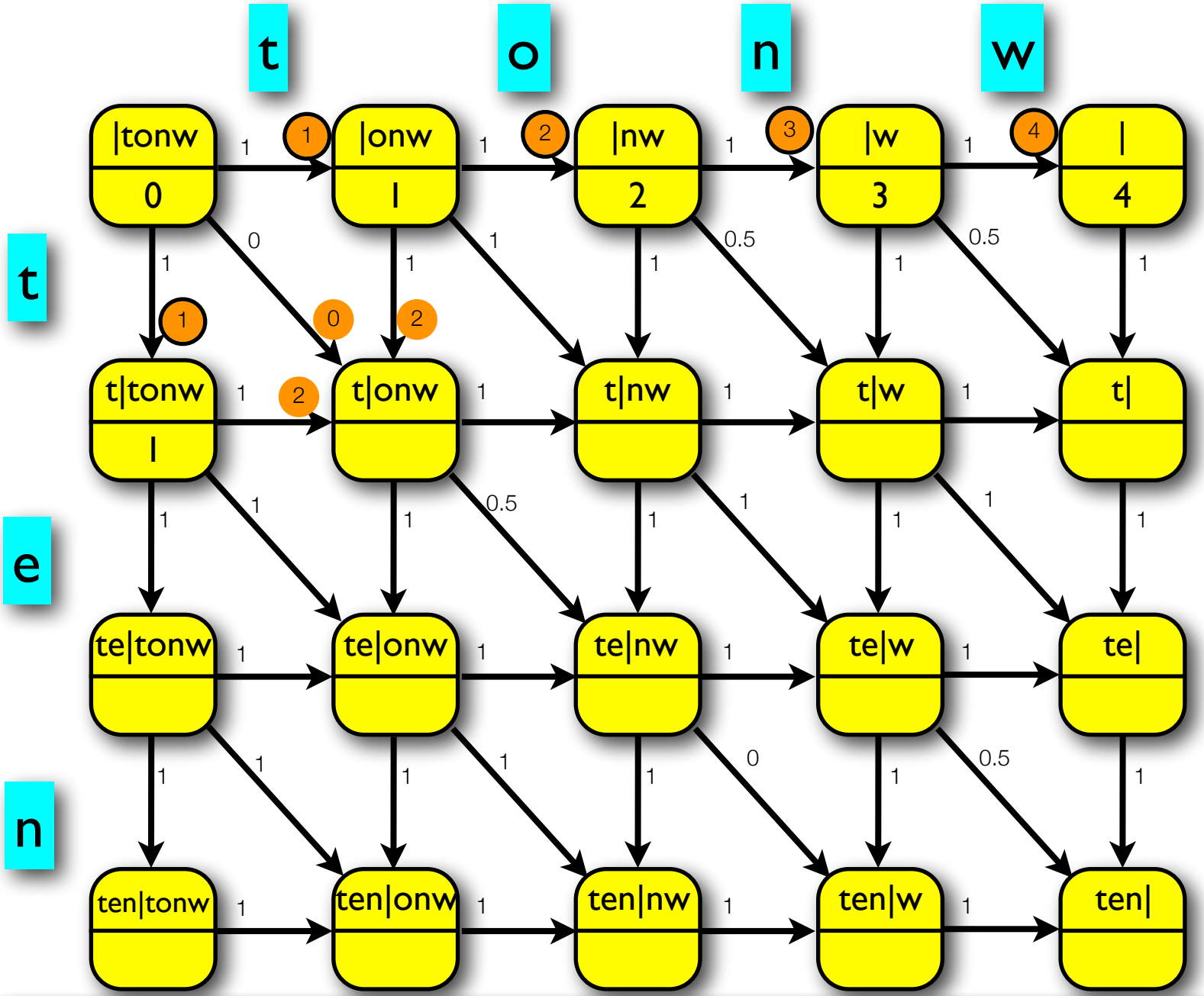
substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

The next node requires choosing one of three paths.

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t”→ “t”, “n”→ “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t”→ “e”, “e”→ “t”)

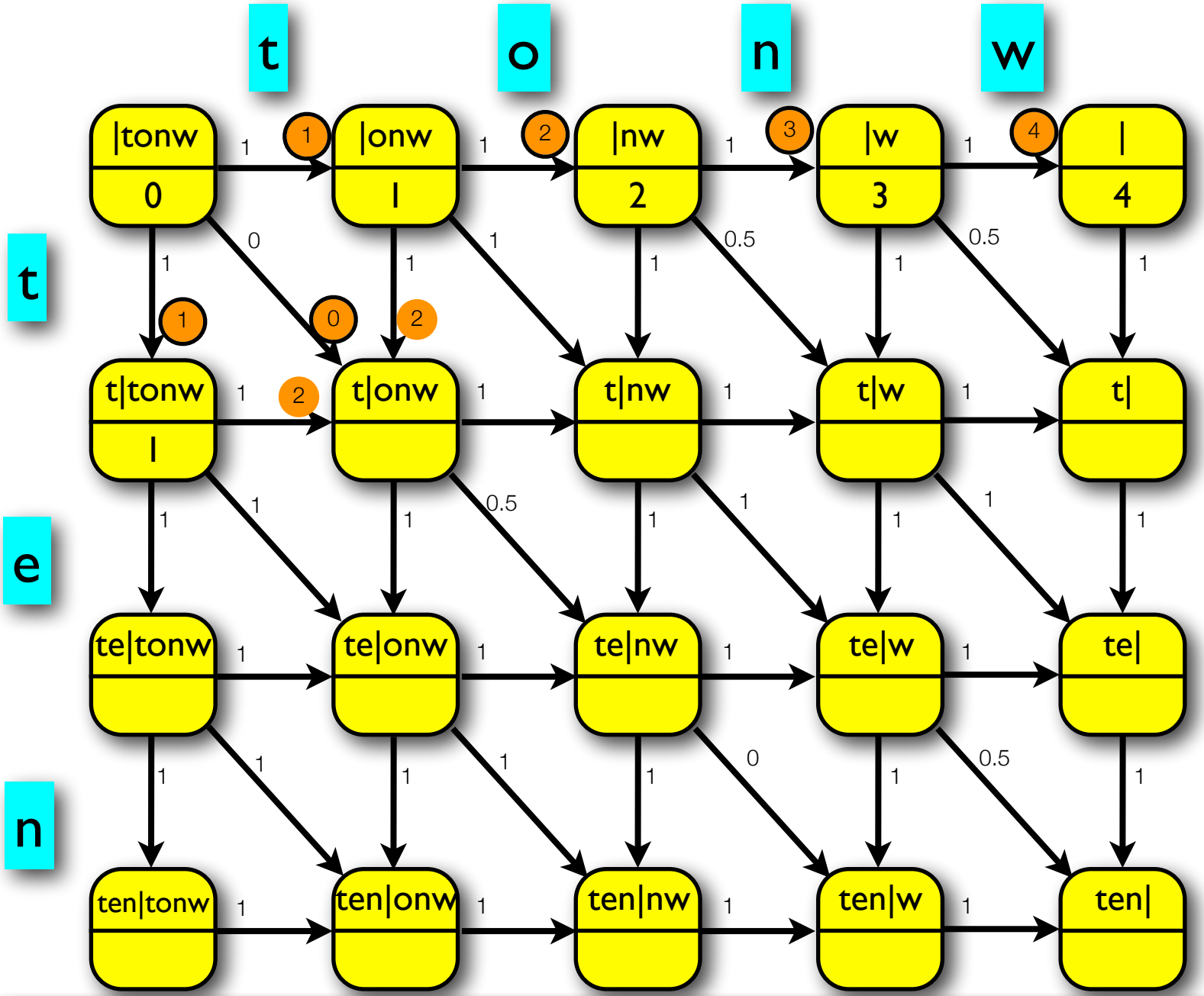
minimum edit distance is trivial for top: only one previous node to choose from!

The next node requires choosing one of three paths.

Pick the cheapest: 0 (substituting “t” for “t”)

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t”→ “t”, “n”→ “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t”→ “e”, “e”→ “t”)

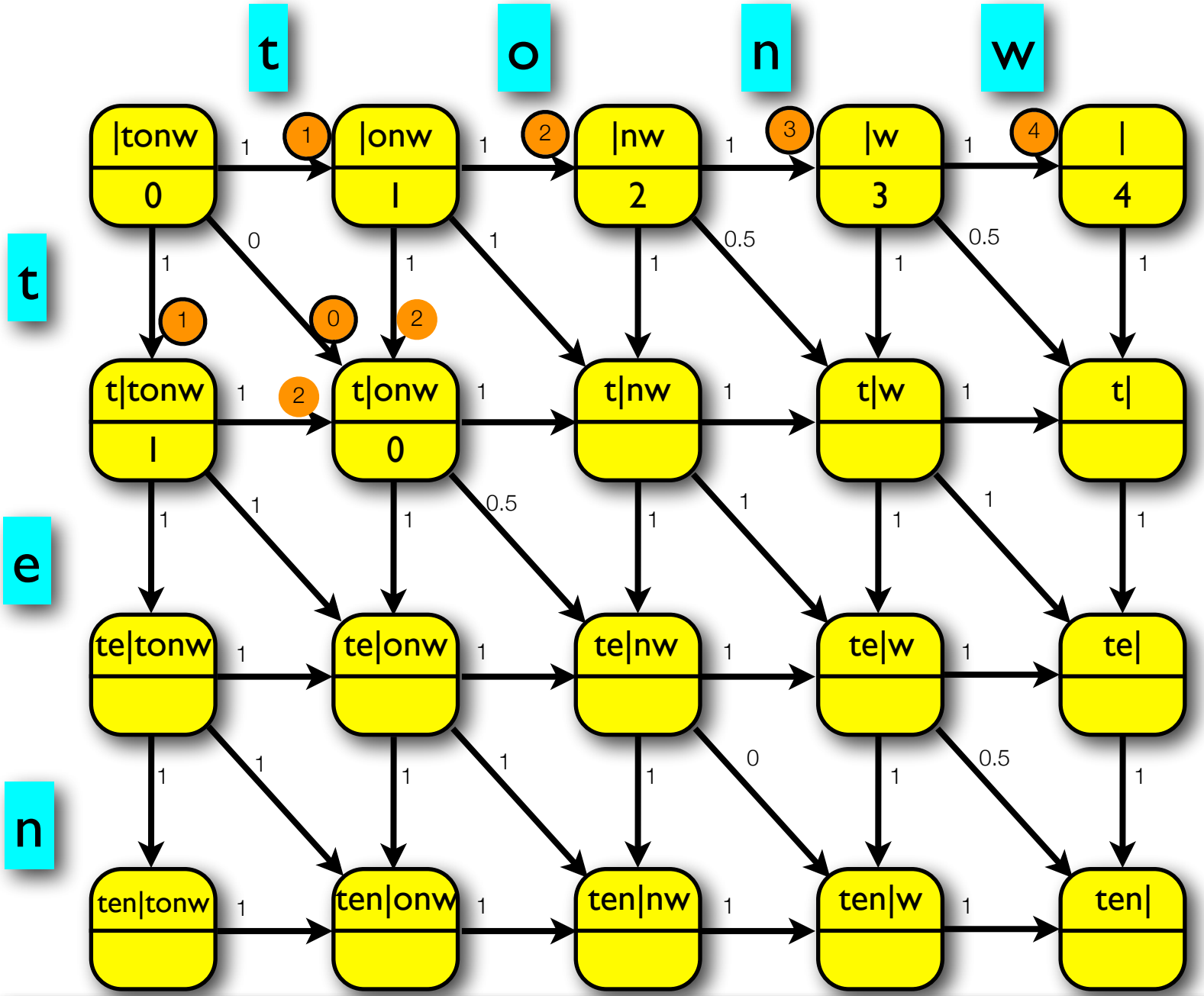
minimum edit distance is trivial for top: only one previous node to choose from!

The next node requires choosing one of three paths.

Pick the cheapest: 0 (substituting “t” for “t”)

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t”→ “t”, “n”→ “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t”→ “e”, “e”→ “t”)

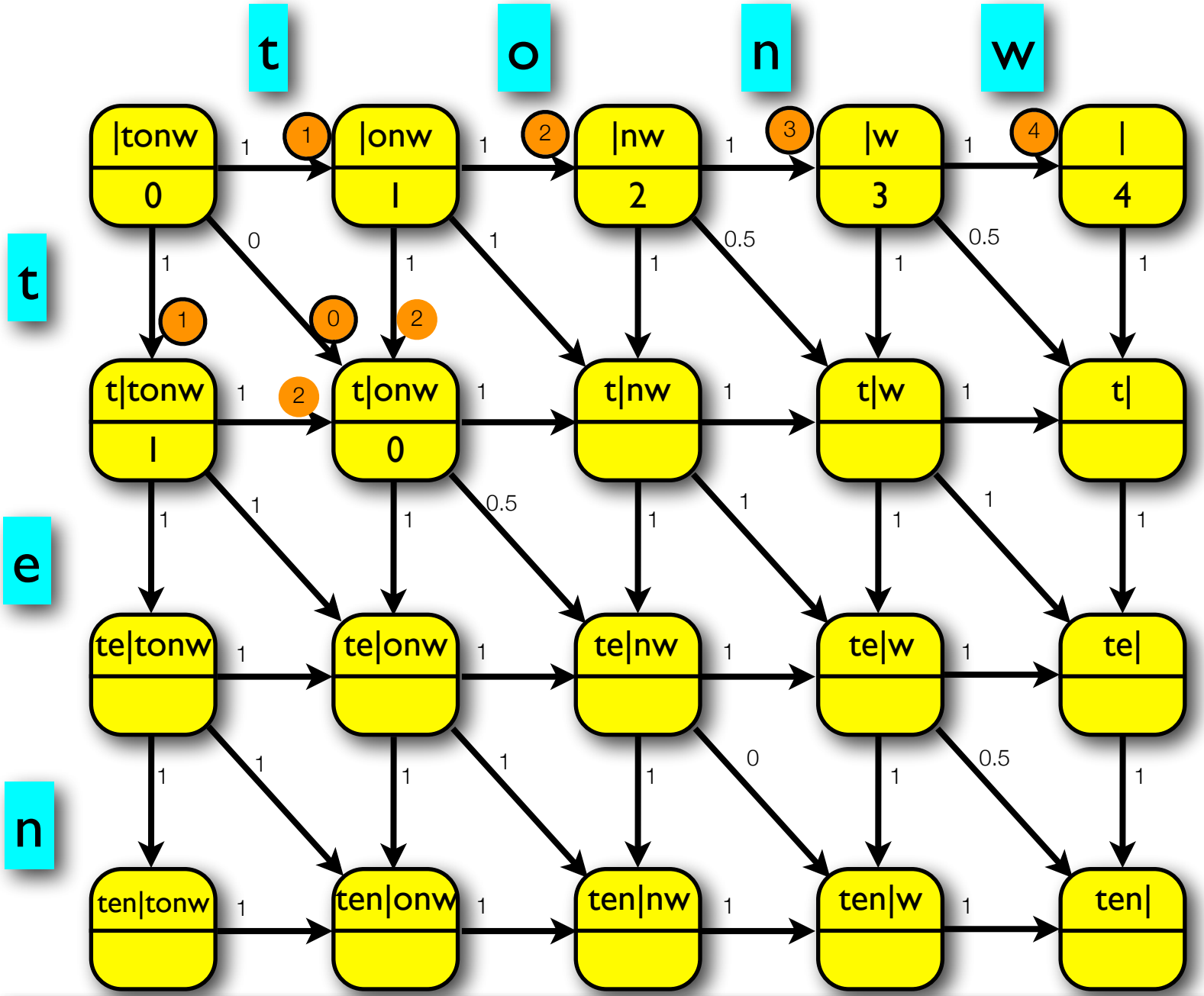
minimum edit distance is trivial for top: only one previous node to choose from!

The next node requires choosing one of three paths.

Pick the cheapest: 0 (substituting “t” for “t”)

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t”→ “t”, “n”→ “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t”→ “e”, “e”→ “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

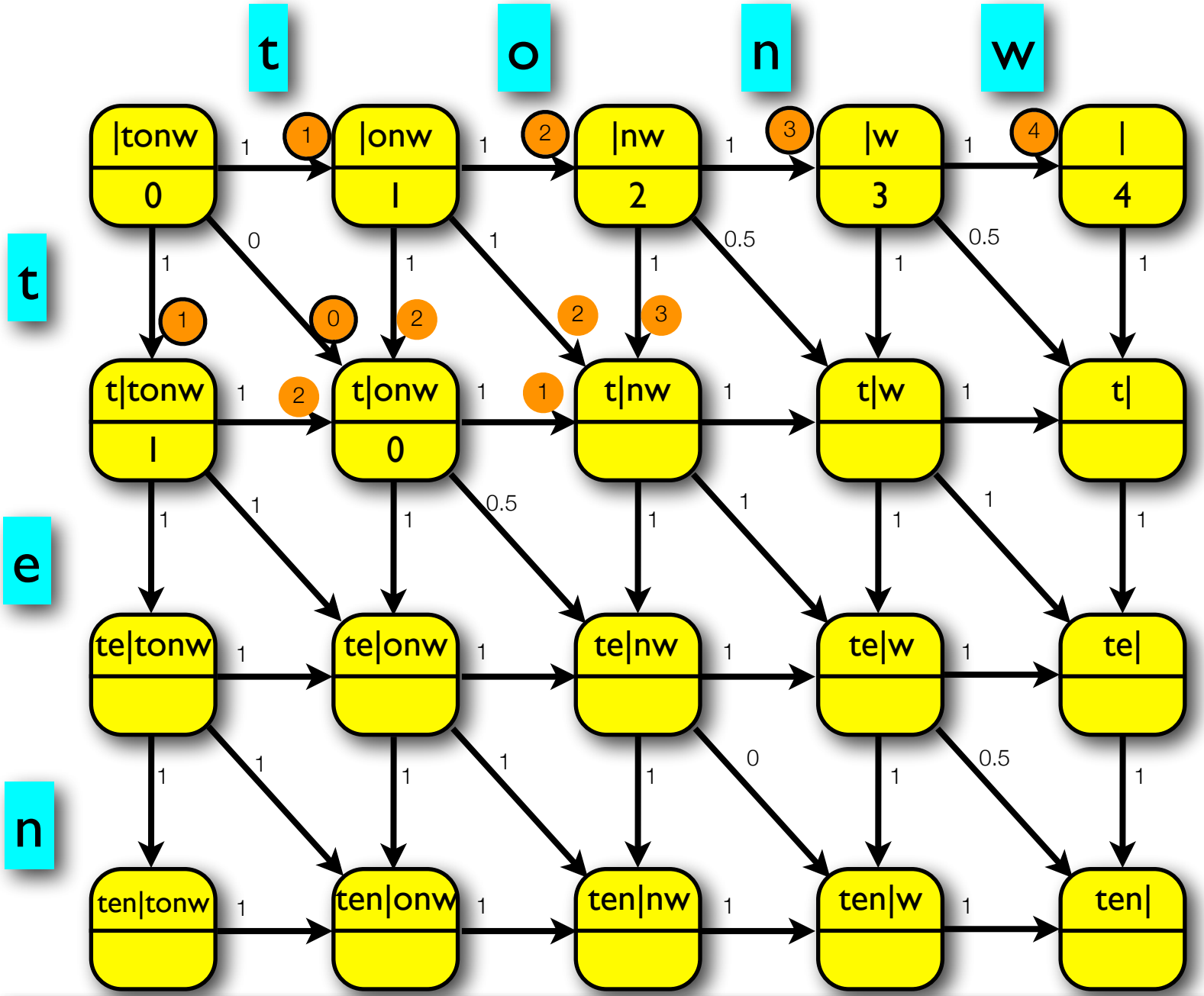
The next node requires choosing one of three paths.

Pick the cheapest: 0 (substituting “t” for “t”)

Same thing for the rest of the row (more quickly now).

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t”→ “t”, “n”→ “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t”→ “e”, “e”→ “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

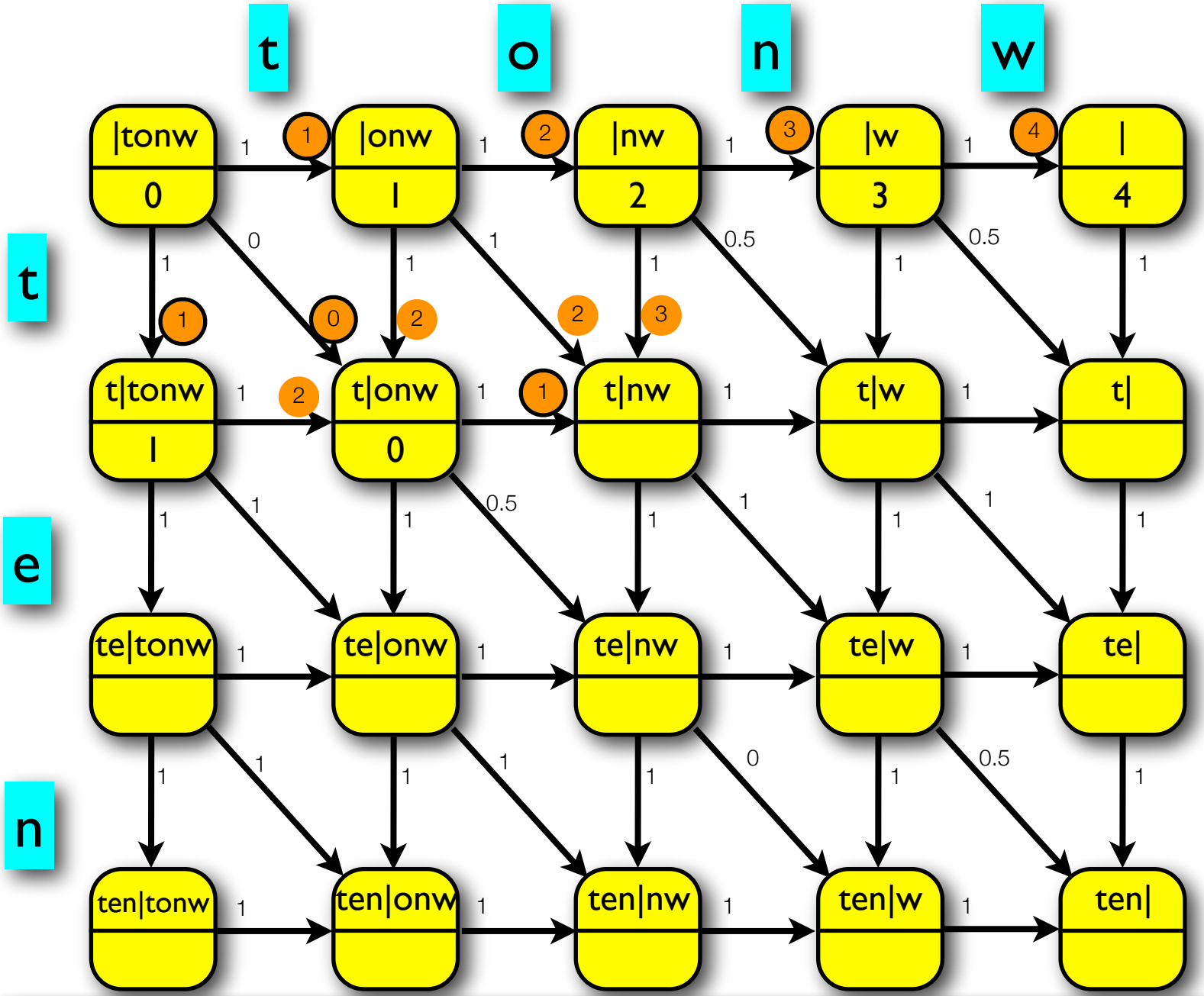
The next node requires choosing one of three paths.

Pick the cheapest: 0 (substituting “t” for “t”)

Same thing for the rest of the row (more quickly now).

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t”→ “t”, “n”→ “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t”→ “e”, “e”→ “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

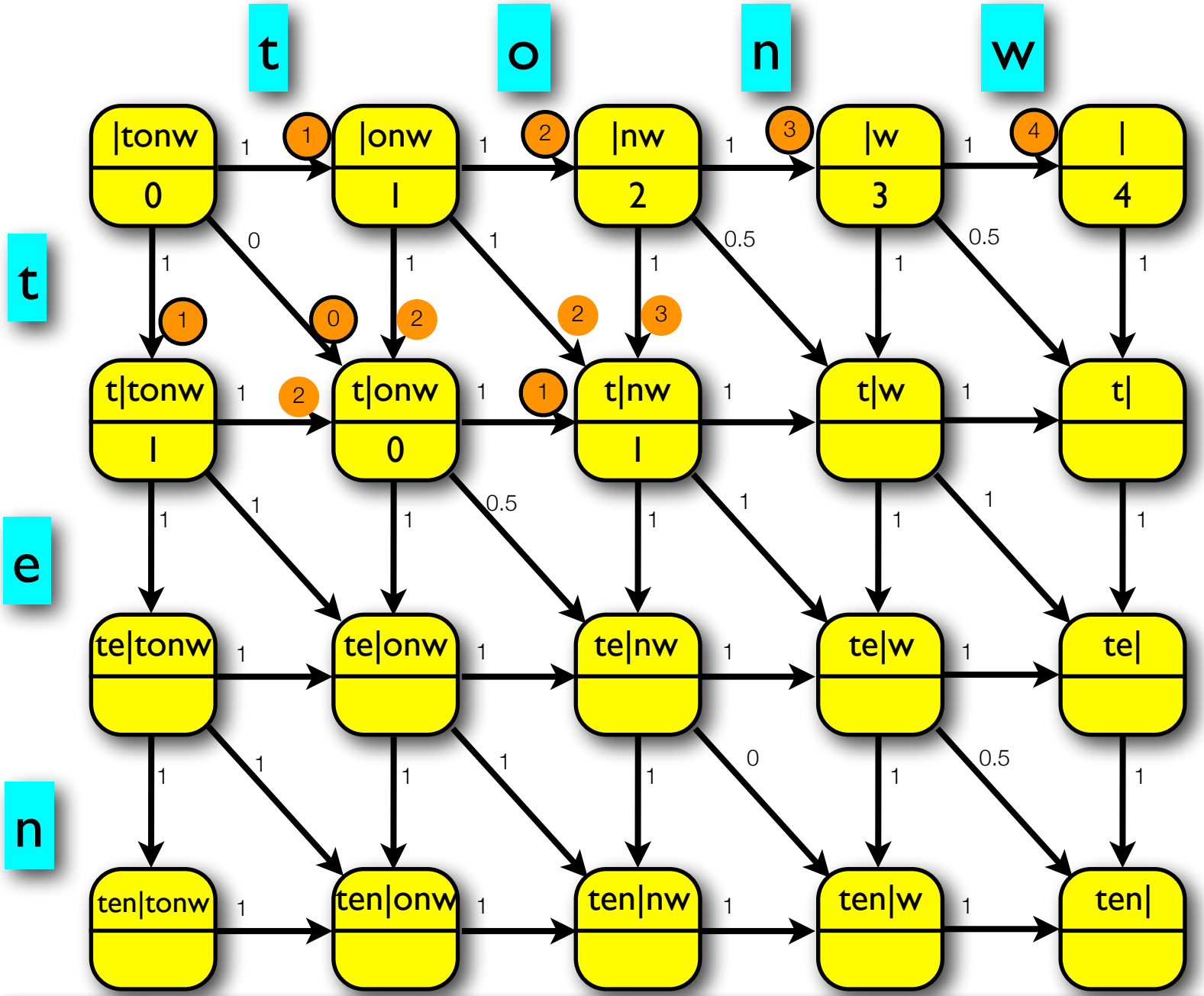
The next node requires choosing one of three paths.

Pick the cheapest: 0 (substituting “t” for “t”)

Same thing for the rest of the row (more quickly now).

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t”→ “t”, “n”→ “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t”→ “e”, “e”→ “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

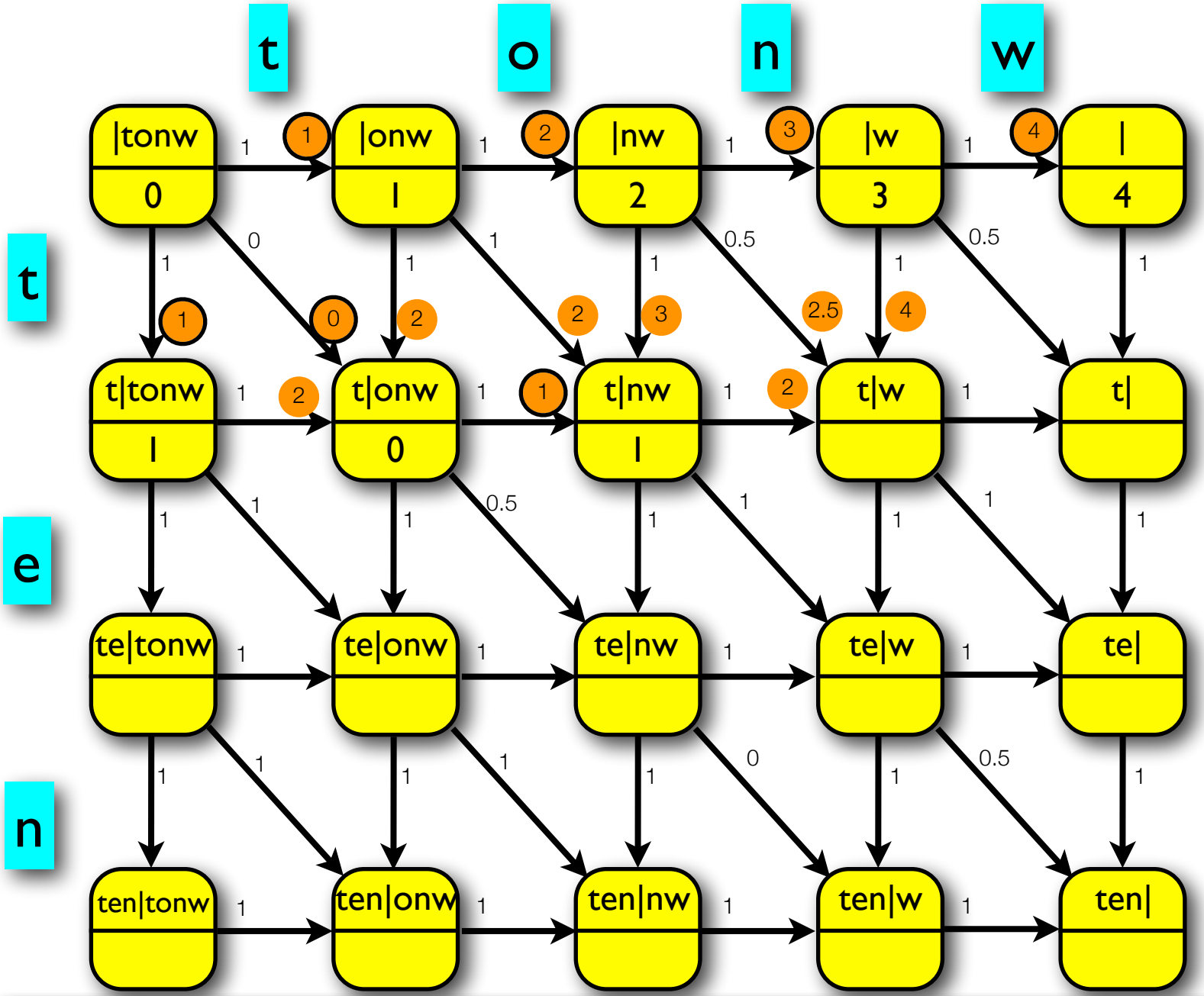
The next node requires choosing one of three paths.

Pick the cheapest: 0 (substituting “t” for “t”)

Same thing for the rest of the row (more quickly now).

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t”→ “t”, “n”→ “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t”→ “e”, “e”→ “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

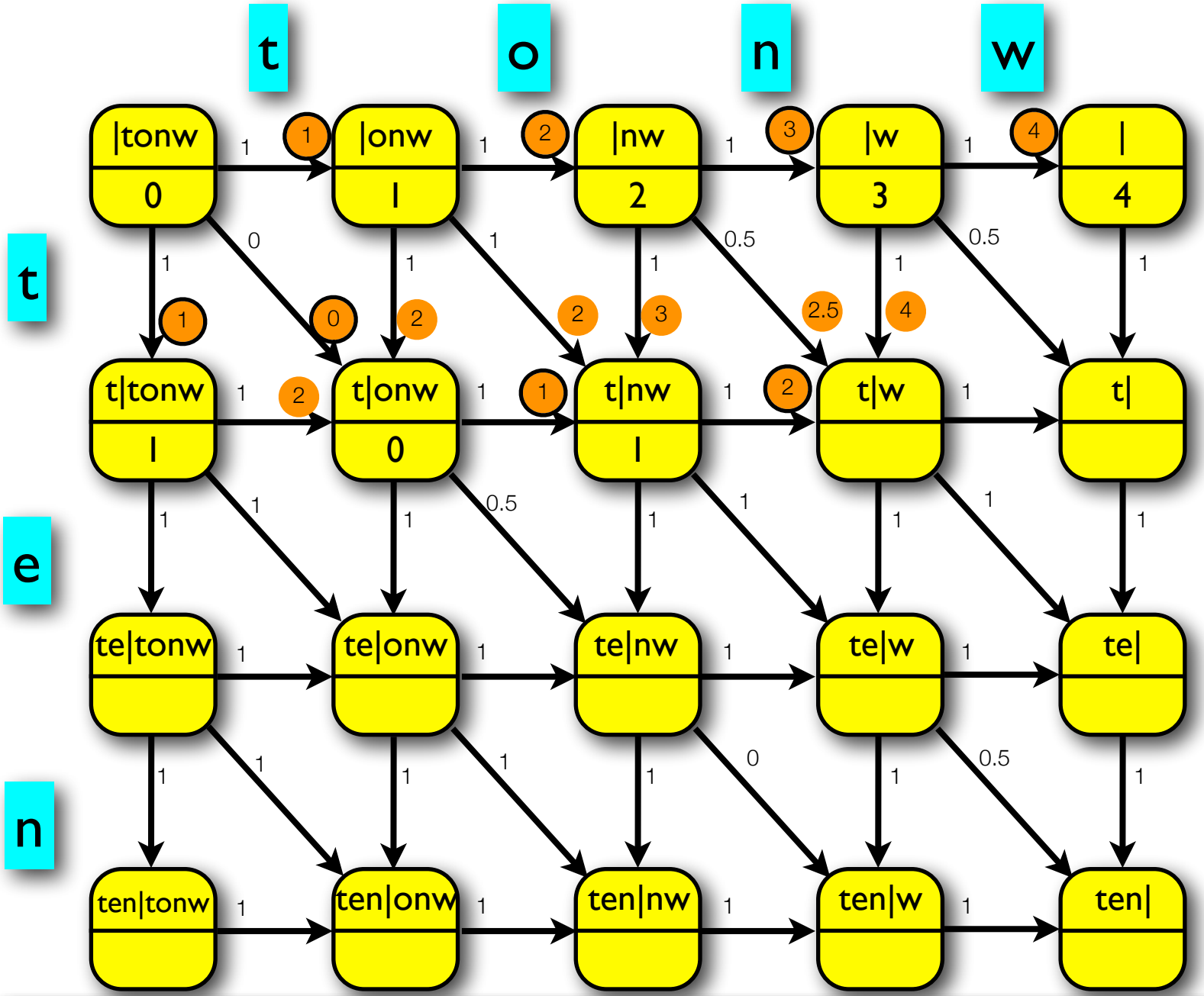
The next node requires choosing one of three paths.

Pick the cheapest: 0 (substituting “t” for “t”)

Same thing for the rest of the row (more quickly now).

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

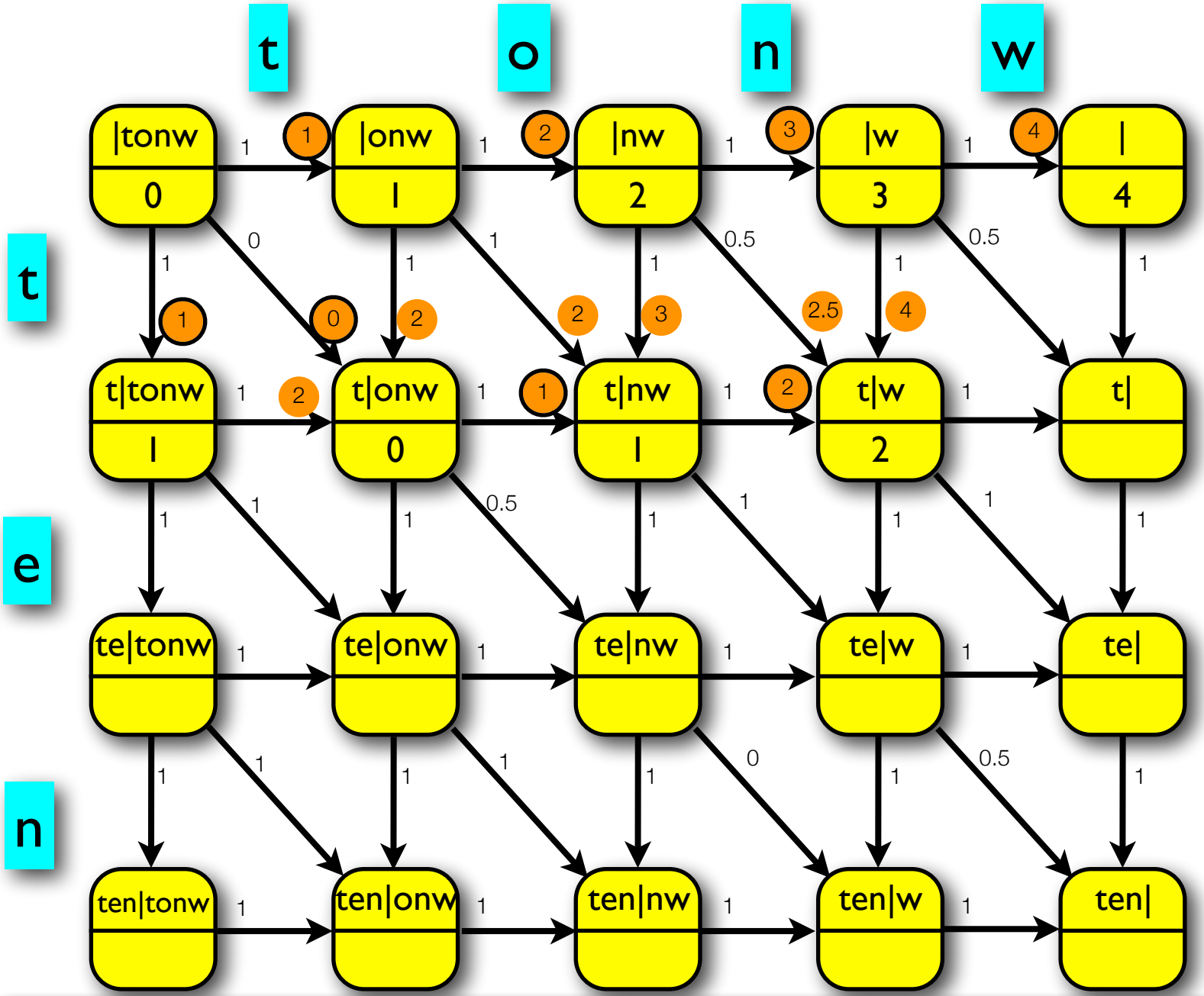
The next node requires choosing one of three paths.

Pick the cheapest: 0 (substituting “t” for “t”)

Same thing for the rest of the row (more quickly now).

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

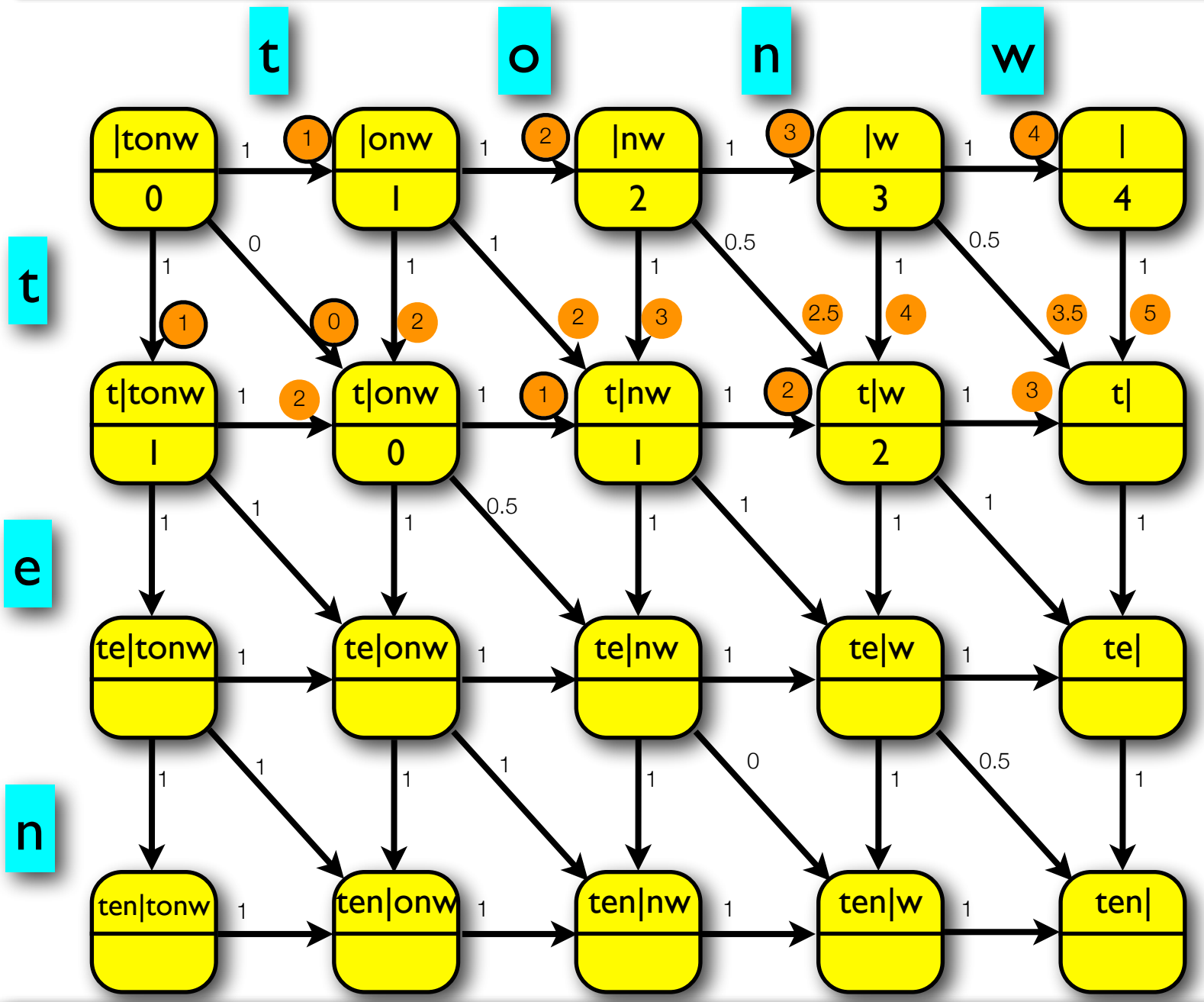
The next node requires choosing one of three paths.

Pick the cheapest: 0 (substituting “t” for “t”)

Same thing for the rest of the row (more quickly now).

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

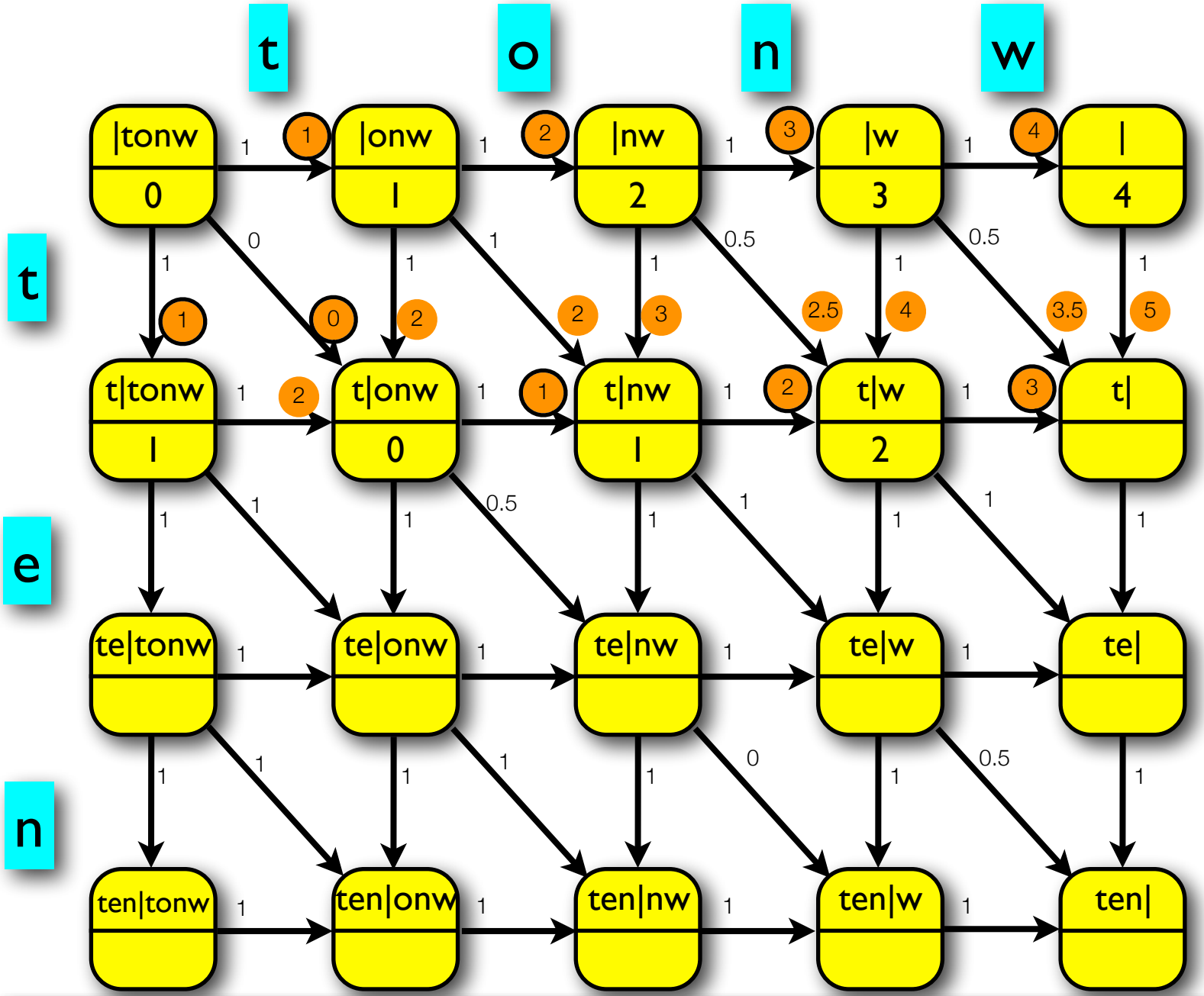
The next node requires choosing one of three paths.

Pick the cheapest: 0 (substituting “t” for “t”)

Same thing for the rest of the row (more quickly now).

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

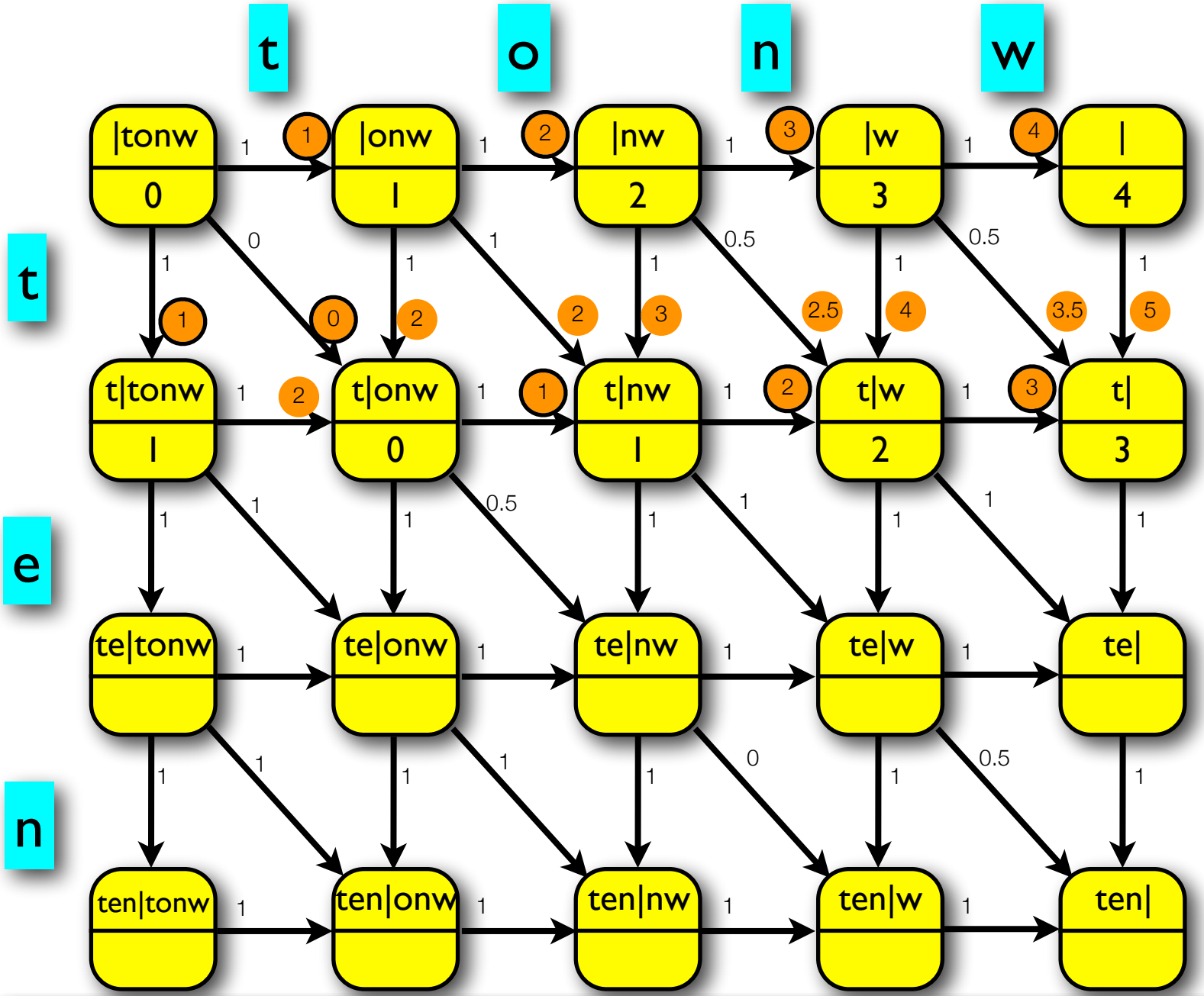
The next node requires choosing one of three paths.

Pick the cheapest: 0 (substituting “t” for “t”)

Same thing for the rest of the row (more quickly now).

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t”→ “t”, “n”→ “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t”→ “e”, “e”→ “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

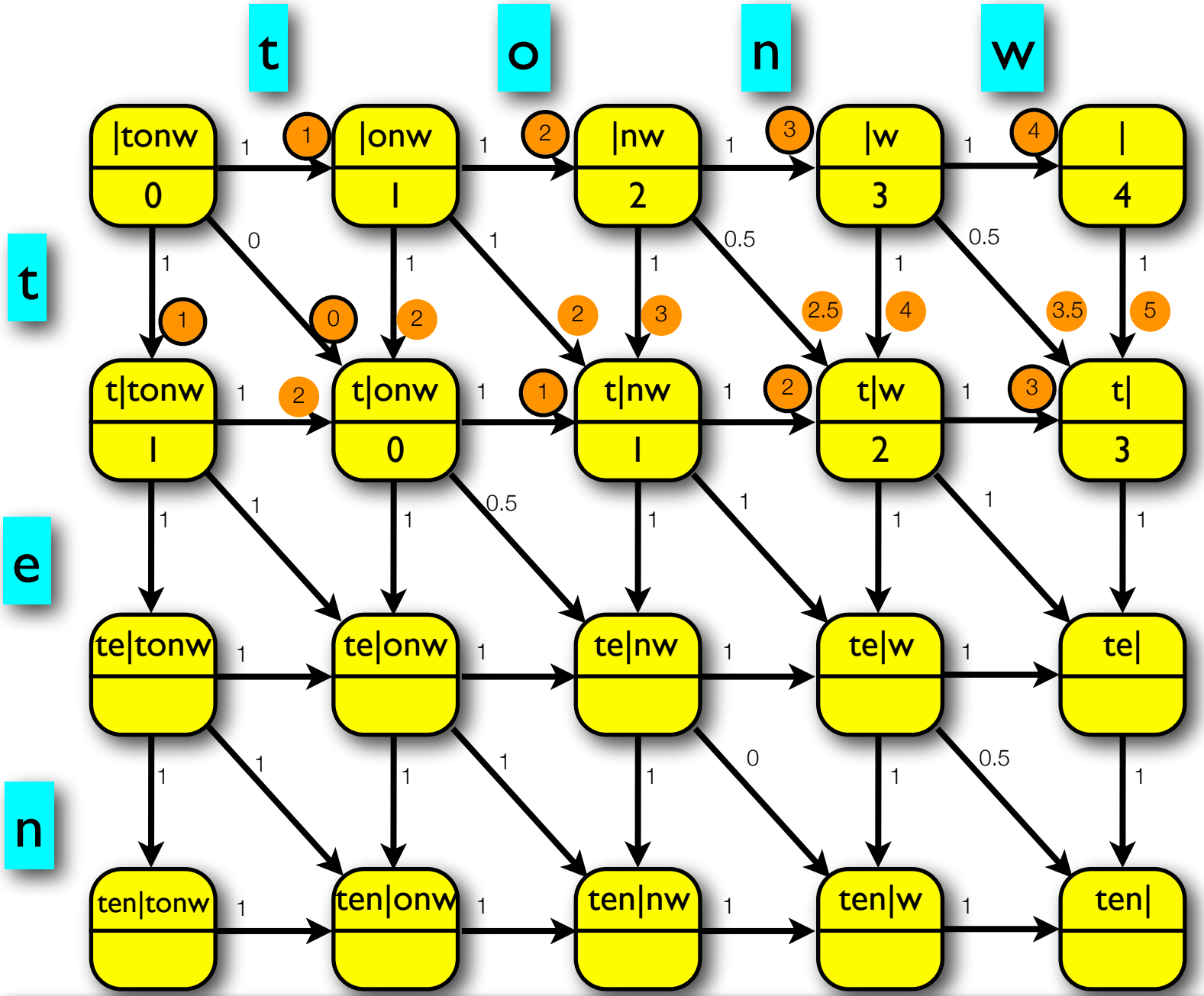
The next node requires choosing one of three paths.

Pick the cheapest: 0 (substituting “t” for “t”)

Same thing for the rest of the row (more quickly now).

Step 3: Do second row (1st column is trivial).

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

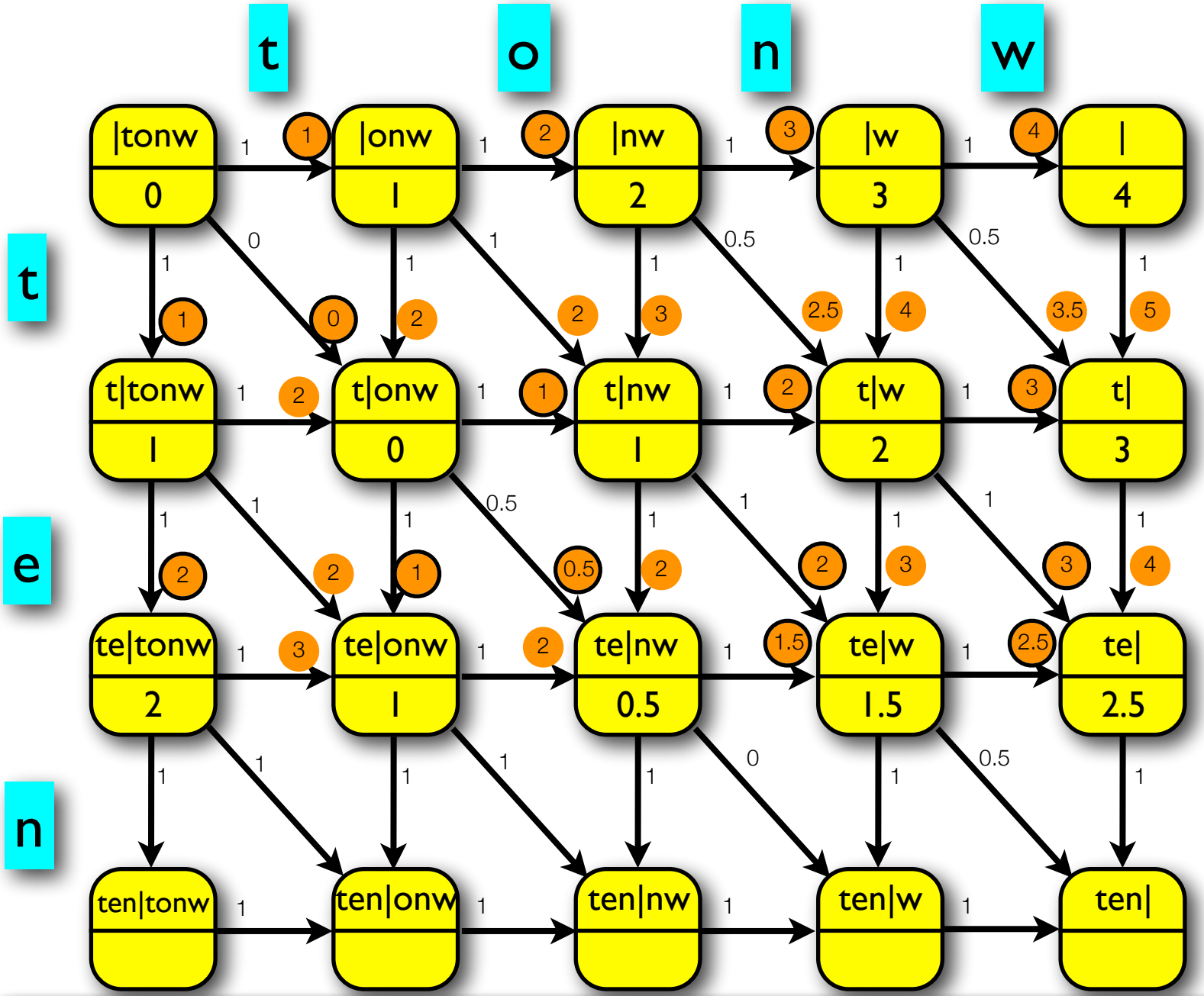
The next node requires choosing one of three paths.

Pick the cheapest: 0 (substituting “t” for “t”)

Same thing for the rest of the row (more quickly now).

Step 4: Keep going row-by-row until you reach the bottom right.

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t” → “t”, “n” → “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t” → “e”, “e” → “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

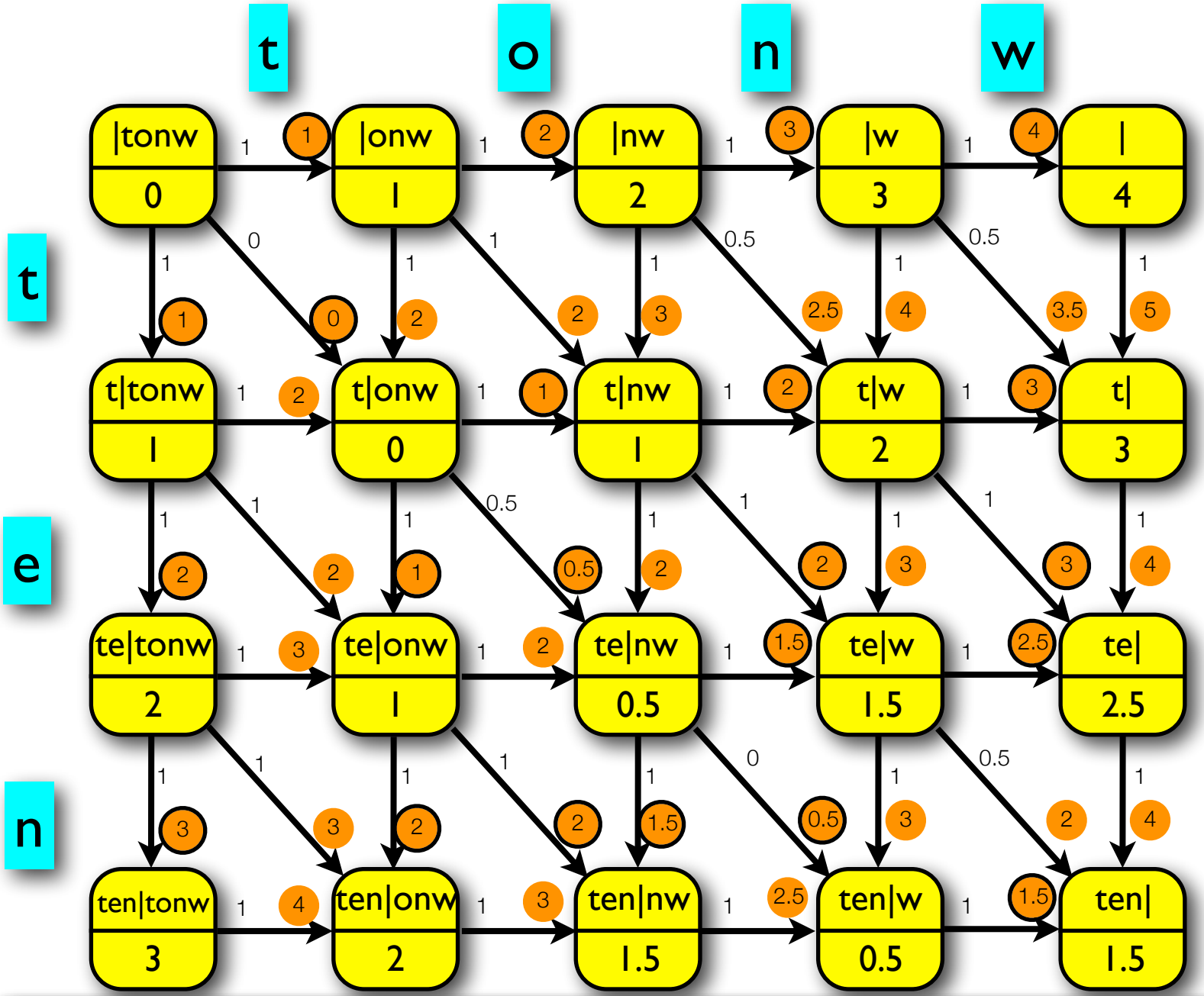
The next node requires choosing one of three paths.

Pick the cheapest: 0 (substituting “t” for “t”)

Same thing for the rest of the row (more quickly now).

Step 4: Keep going row-by-row until you reach the bottom right.

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t”→ “t”, “n”→ “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t”→ “e”, “e”→ “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

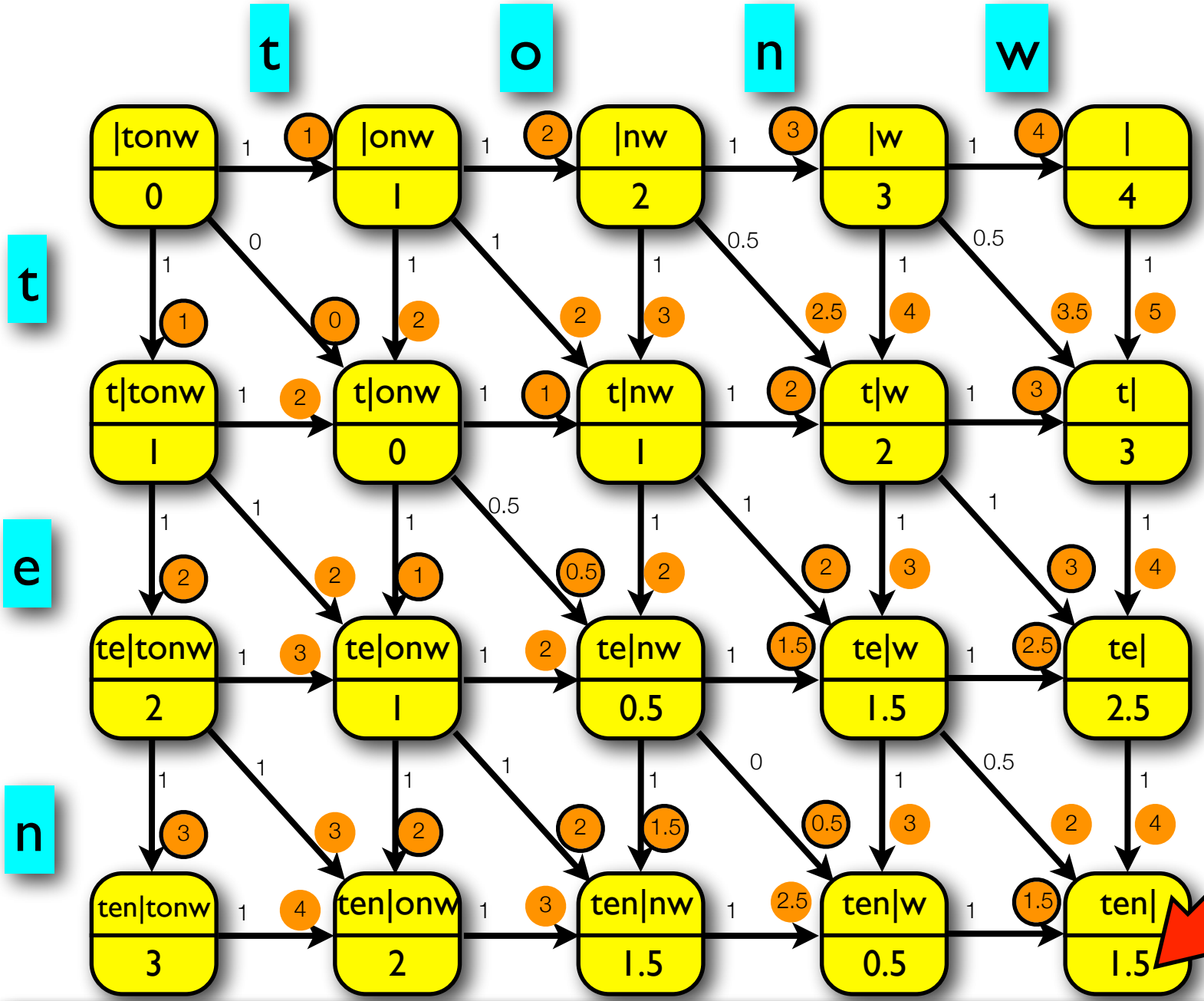
The next node requires choosing one of three paths.

Pick the cheapest: 0 (substituting “t” for “t”)

Same thing for the rest of the row (more quickly now).

Step 4: Keep going row-by-row until you reach the bottom right.

Minimum edit distance for “tonw” to “ten”: animated



all insertions cost 1

all deletions cost 1

substitutions cost 0 if same letter (e.g. “t”→ “t”, “n”→ “n”)

substitutions cost 0.5 if substituting vowel for vowel or consonant for consonant (e.g “o” → “e”, “n” → “t”); otherwise, cost is 1 (e.g. “t”→ “e”, “e”→ “t”)

minimum edit distance is trivial for top: only one previous node to choose from!

The next node requires choosing one of three paths.

Pick the cheapest: 0 (substituting “t” for “t”)

Same thing for the rest of the row (more quickly now).

This is the minimum edit distance for turning “tonw” into “ten”.

Step 4: Keep going row-by-row until you reach the bottom right.

What minimum edit distance is good for.

- Candidate generation phase: We can use it to rank candidates, e.g. such that we only consider candidates that are a maximum of 2 edits from the typo.
- Picking the best candidate phase: We can use edit distance as part of calculating the probabilities for the error model.
  - Alternatively, we can use a simple confusion matrix -- see Dickinson's slides.

Correcting “tonw”

- Let’s say we are considering only the candidates “town”, “ten” and “tow”.
- We need error model probabilities and language model probabilities in order to compute the score for each candidate:
  - *town*: p(typo=“tonw” | cand=“town”) x p(cand= “town”)
  - *ten*: p(typo=“tonw” | cand=“ten”) x p(cand= “ten”)
  - *town*: p(typo=“tonw” | cand=“tow”) x p(cand= “tow”)

Error model probabilities

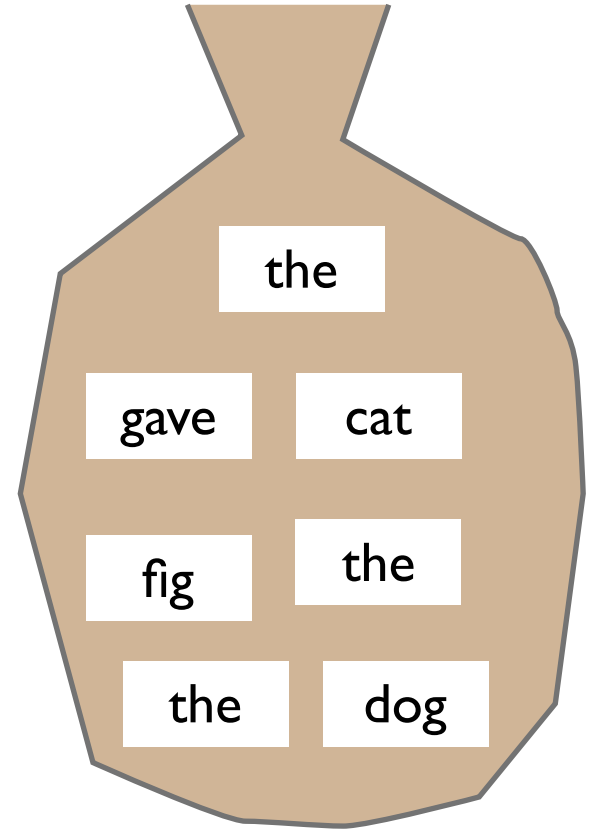
- You'll just be given some values for the error model probabilities. Here they are for the "tonw" problem:
  - p(typo="tonw" | cand="town") = .0021
  - p(typo="tonw" | cand="ten") = .0018
  - p(typo="tonw" | cand="tow") = .0132
- These are made up, but are meant to be somewhat plausible (at least in relation to one another): *town* and *ten* are two edits from *tonw*, while *tow* is one edit away.
  - Also note that I've made *town* just a bit more likely that *ten* because actually "wn" -> "nw" is a transposition: easy to make such a mistake.

Language model

- Language models tell us whether some word sequences are more or less probable than others.
  - For example, you are more likely to hear someone say “I worked in town today” than “I worked in tow trucks” because it is more usual to talk about *town* and *today* than *tow* and *trucks*.
- N-gram models predict the probability of a word in a sentence based on the previous n-1 words.
  - unigram model $p(word_i \mid context) = p(word_i)$
  - bigram model $p(word_i \mid context) = p(word_i \mid word_{i-1})$
  - trigram model $p(word_i \mid context) = p(word_i \mid word_{i-1}, word_{i-2})$
- A unigram model uses no context, so order doesn’t matter: “I worked in town today” is as probable as “town worked I today in”. They are very easy, so let’s start with a unigram language model for fixing “tonw”.

Unigram language model (a bag of words)

- Imagine that I write “the cat gave the dog the fig” on a piece of paper, then cut out each word and put them into a bag.
- Next, I ask you to reach into the bag and pull out one of the slips of paper.
- What is the probability that the word written on the piece of paper is “the”?
  - Easy: 3/7
- Now, I’ll give you numbers for a corpus with many more words, but we can get the probability of each word in that corpus using the same logic.



Unigram language model for a larger corpus

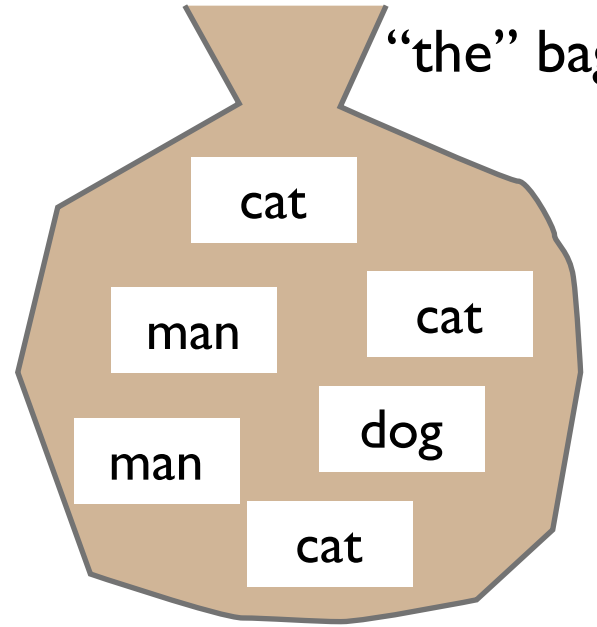
- You are going to estimate the probabilities of the candidates using a corpus which has the following counts:
  - Total number of words: 8753
  - Number of tokens of *town*: 23
  - Number of tokens of *ten*: 42
  - Number of tokens of *tow*: 2
- Unigram probabilities:
  - p(cand = “town”) = 23/8753 = .00263
  - p(cand = “ten”) = 42/8753 = .00480
  - p(cand = “tow”) = 2/8753 = .00023

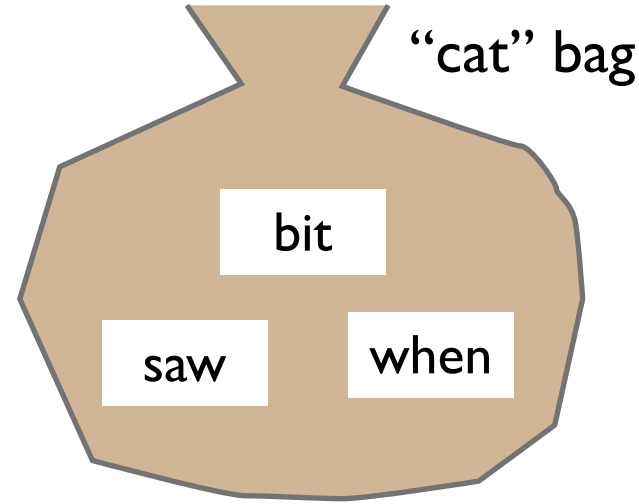
Picking the best candidate with the unigram model

- Recall we want the candidate that maximizes the value p(typo | cand) x p(cand).
- Putting the error model and the language model together:
  - town: p(typo="tonw" | cand="town") x p(cand= "town") = .0021 x .00263 = .00000552
  - ten: p(typo="tonw" | cand="ten") x p(cand= "ten") = .0018 x .00480 = .00000864
  - tow: p(typo="tonw" | cand="tow") x p(cand= "tow") = .0132 x .00023 = .00000304
- With these values, we can rank the candidates:
  - ten > town > tow
  - So, we pick "ten" automatically as the best correction for the typo "tonw" in the sentence "She is there favorite acress in tonw."
- That might seem odd -- we mortals might expect *town* to be the correction... let's try adding more context.

Bigram language model - (context-dependent bags of words)

- A bigram model uses the previous word in determining which candidate is the most probable.
- Now, we have a bag of words for each word in the vocabulary. These words represent the word tokens that occur after the word "labeling" the bag. For example, with a sentence like:
  - "the man says the cat bit the cat when the dog and the cat saw the man"
  - the bag of words that follow "the" is that on the upper right, and the bag of words that follow "cat" is the bottom right.
  - Notice that the number of words in each bag is the same as the number of times the word labeling the bag occurred.
- Using this (very small) corpus, what is the probability of "cat" occurring after "the"? 3/6
- Of "bit" occurring after "cat"? 1/3





Bigram language model -- back to the bigger corpus

- Our corpus has 8753 words totally.
- We want to exploit the knowledge that the word before the typo “tonw” is “in” -- that means we need to collect a bag of word tokens that occur after “in”.
- Restricting our attention to the candidates we find:
  - Number of times “in” occurs: 317 times
  - Number of times “in town” occurs: 11 times
  - Number of times “in ten” occurs: 5 times
  - Number of times “in tow” occurs: 1 time
- Bigram probabilities p(cand | prev = “in):
  - p(cand=“town” | prev =“in”) = 11/317 = .0347
  - p(cand=“ten” | prev =“in”) = 5/317 = .0158
  - p(cand=“tow” | prev =“in”) = 1/317 = .0032

Picking the best candidate with the bigram model

- Recall we want the candidate that maximizes the value p(typo | cand) x p(cand). Now, p(cand) = p(cand|prev).
- Putting the error model and the language model together:
  - town: p(typo="tonw" | cand="town") x p(cand= "town" | prev="in") = .0021 x .0347 = .0000729
  - ten: p(typo="tonw" | cand="ten") x p(cand= "ten" | prev="in") = .0018 x .0158 = .0000284
  - tow: p(typo="tonw" | cand="tow") x p(cand= "tow" | prev="in") = .0132 x .0032 = .0000422
- With these values, we can rank the candidates:
  - town > tow > ten
  - So, now we pick "town" automatically as the best correction for the typo "tonw" in the sentence "She is there favorite acress in tonw."