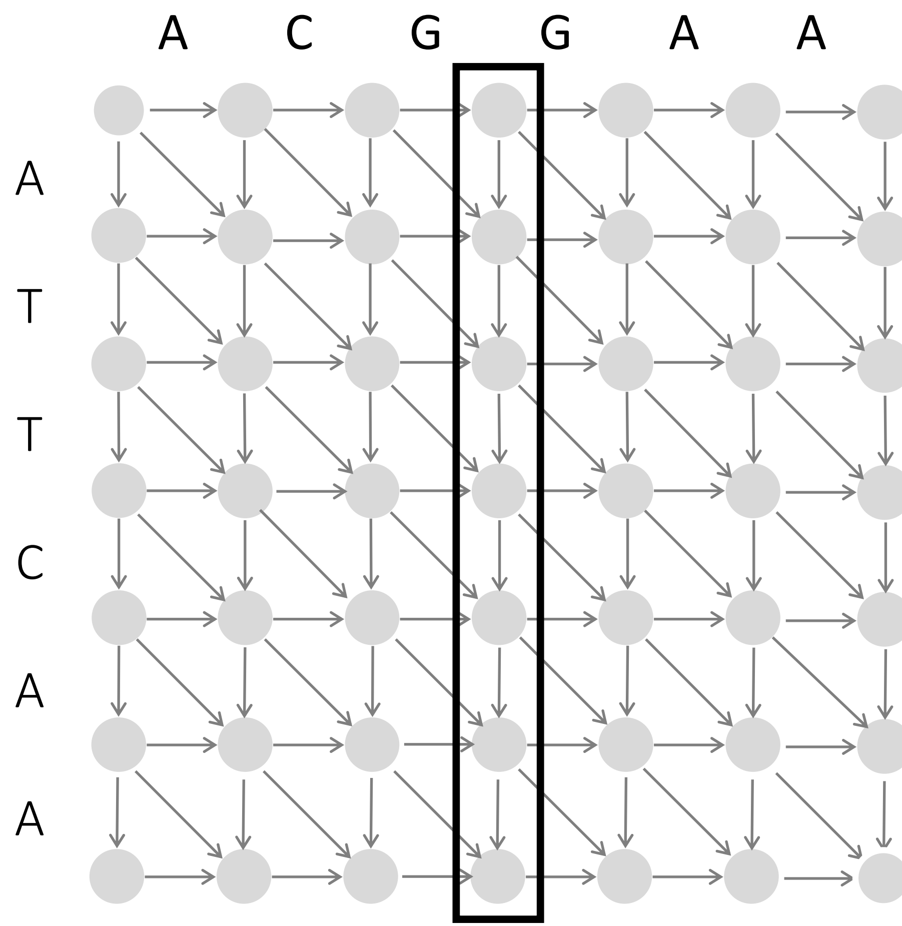
Linear-space sequence alignment

Joe Song
Visiting faculty member
Department of Computer Science, Czech Technical University
Professor, Department of Computer Science,
New Mexico State University

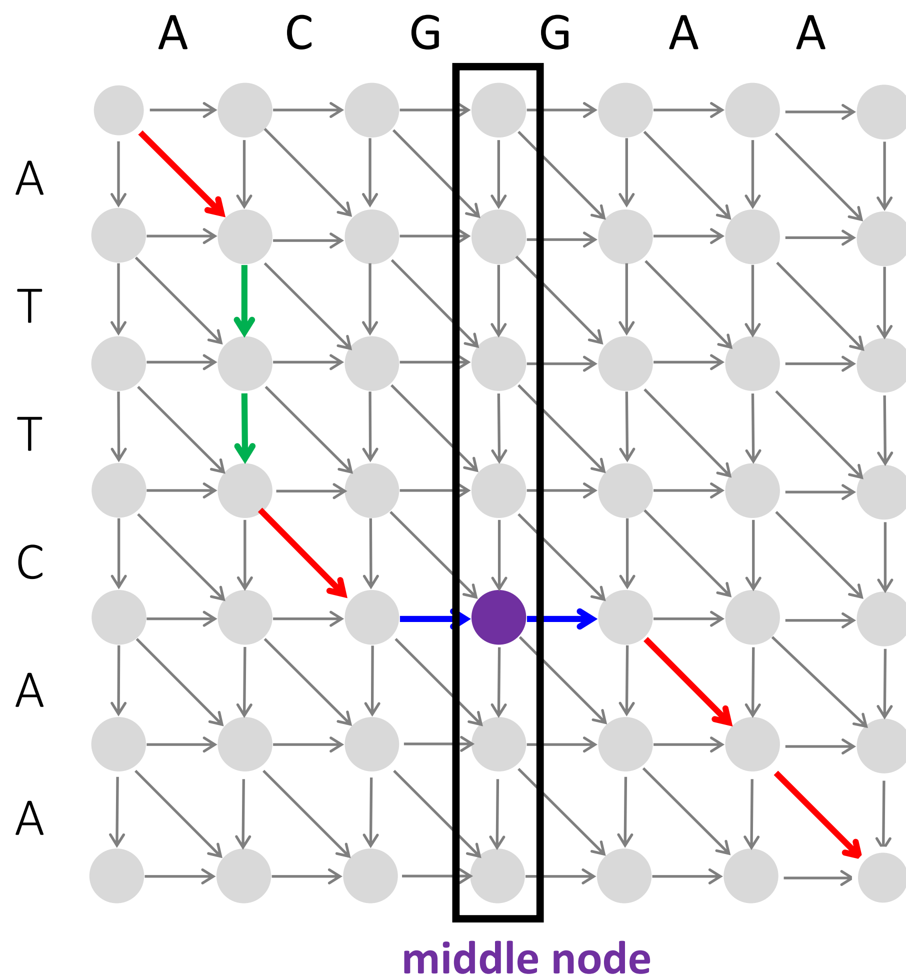
Adapted from Phillip Compeau and Pavel Pevzner. *Bioinformatics Algorithms: An Active Learning Approach*

Middle Column of the Alignment



middle column
(*middle*=#columns/2)

Middle Node of the Alignment

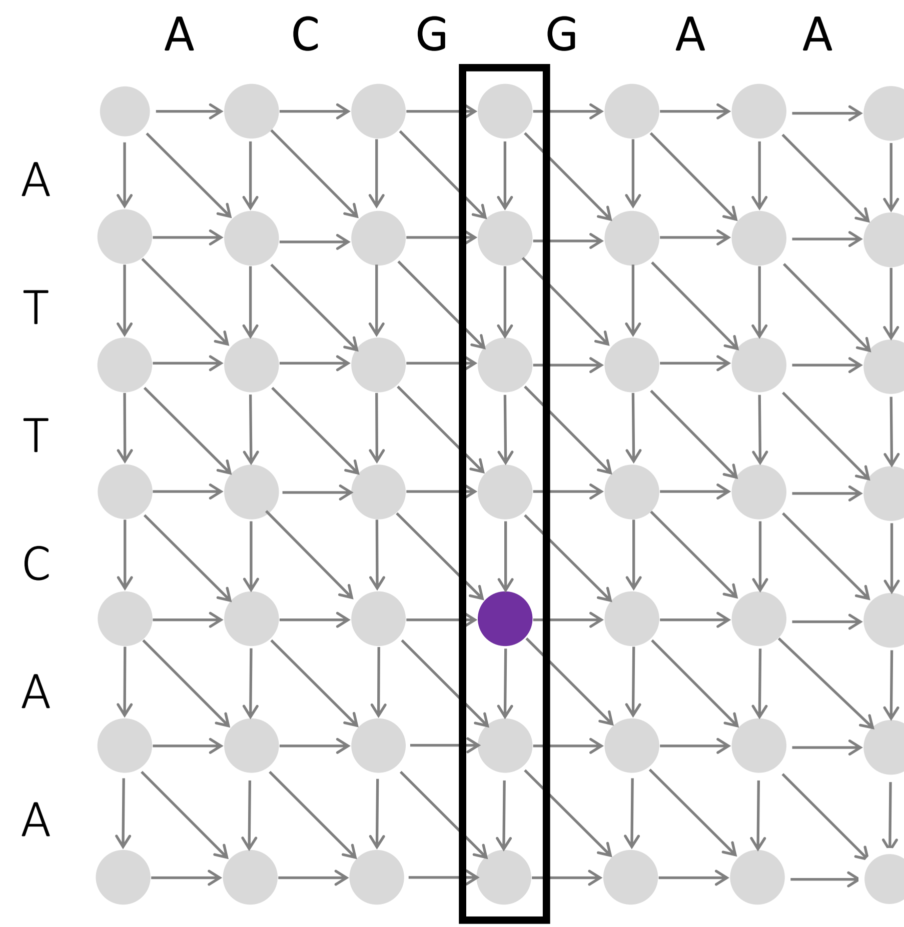


(a node where an optimal alignment path crosses the middle column)

Divide & Conquer for Sequence Alignment

AlignmentPath(*source, sink*)

find *MiddleNode*

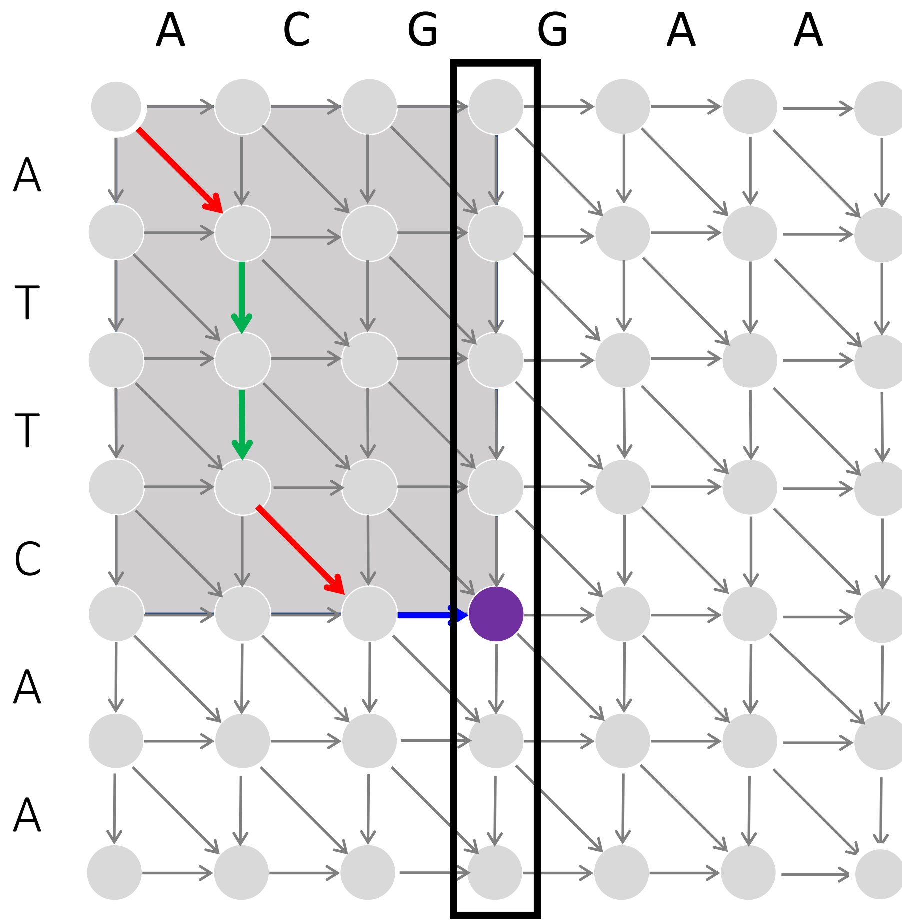


Divide and Conquer Approach to Sequence Alignment

AlignmentPath(*source, sink*)

find *MiddleNode*

AlignmentPath(*source, MiddleNode*)



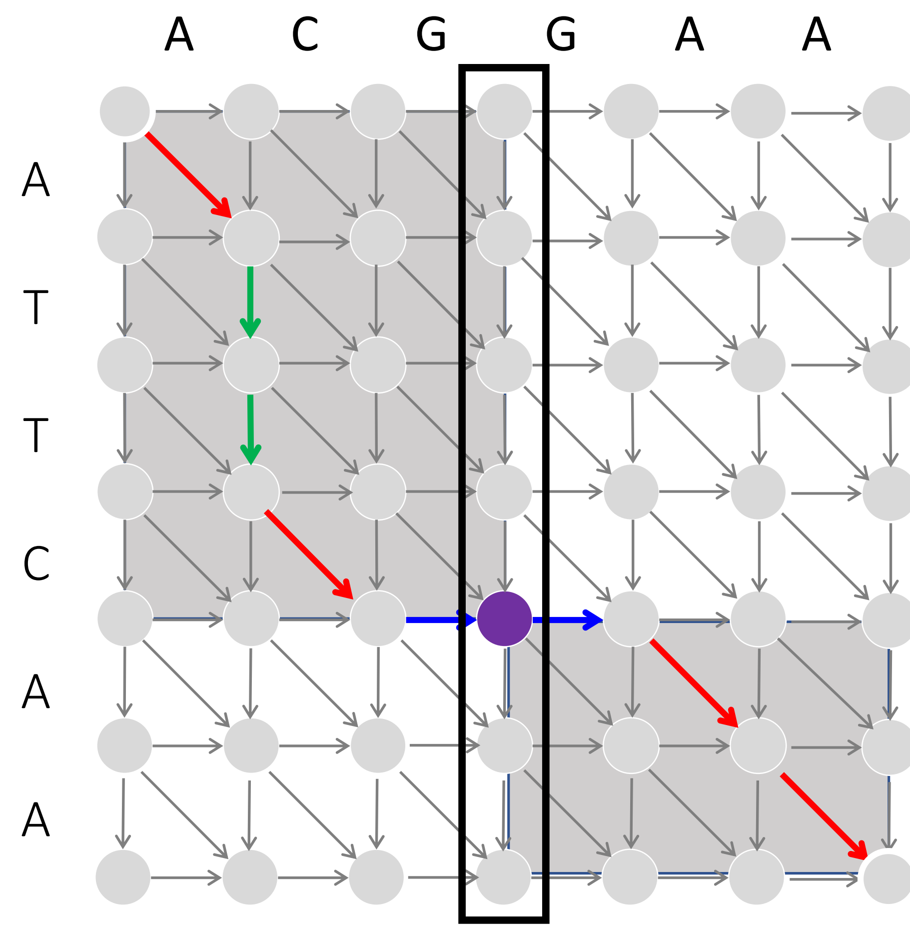
Divide and Conquer Approach to Sequence Alignment

AlignmentPath(*source*, *sink*)

find *MiddleNode*

AlignmentPath(*source*, *MiddleNode*)

AlignmentPath(*MiddleNode*, *sink*)



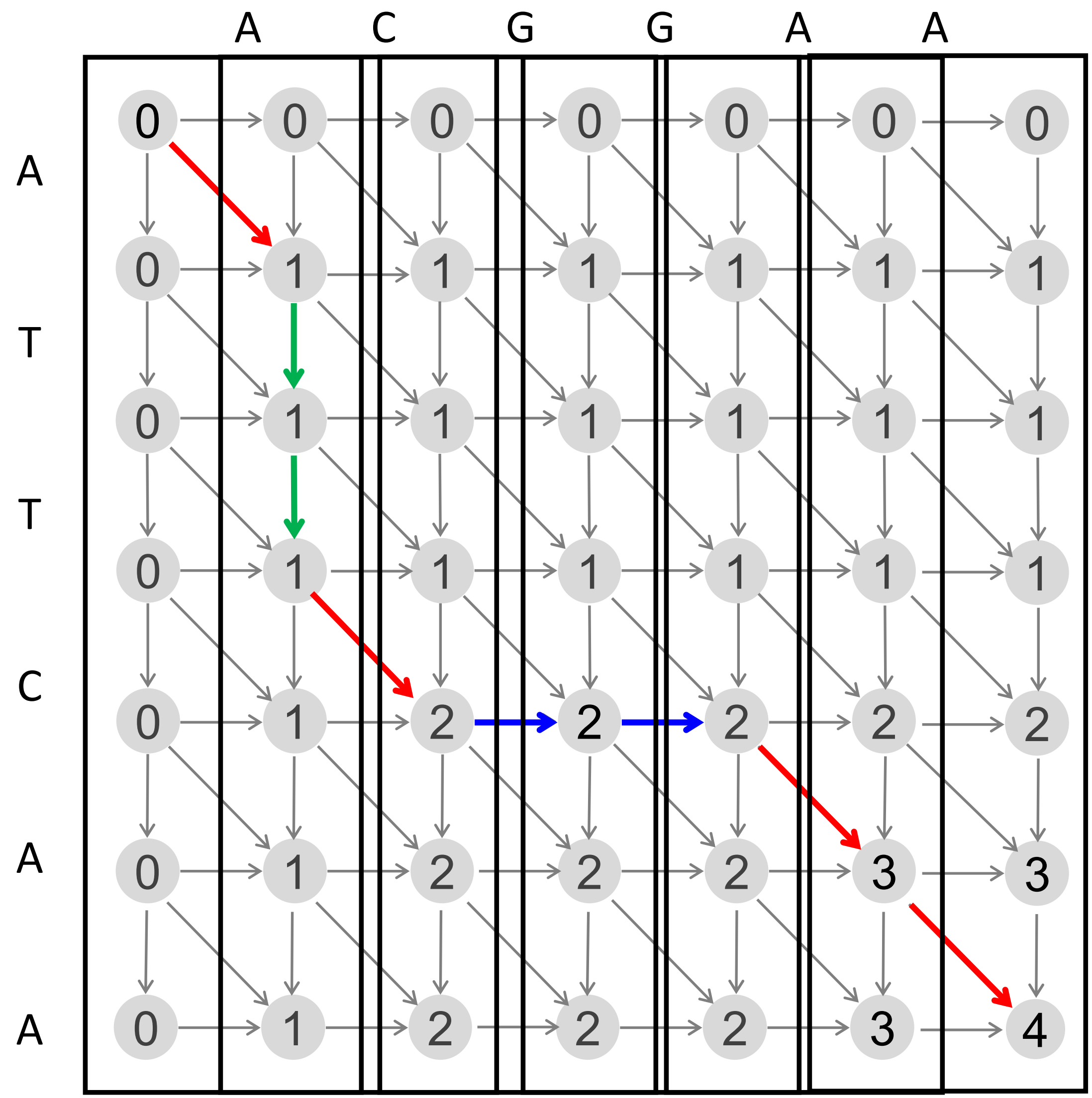
How do we find the middle node in **linear space?**

Computing Alignment Score in Linear Space

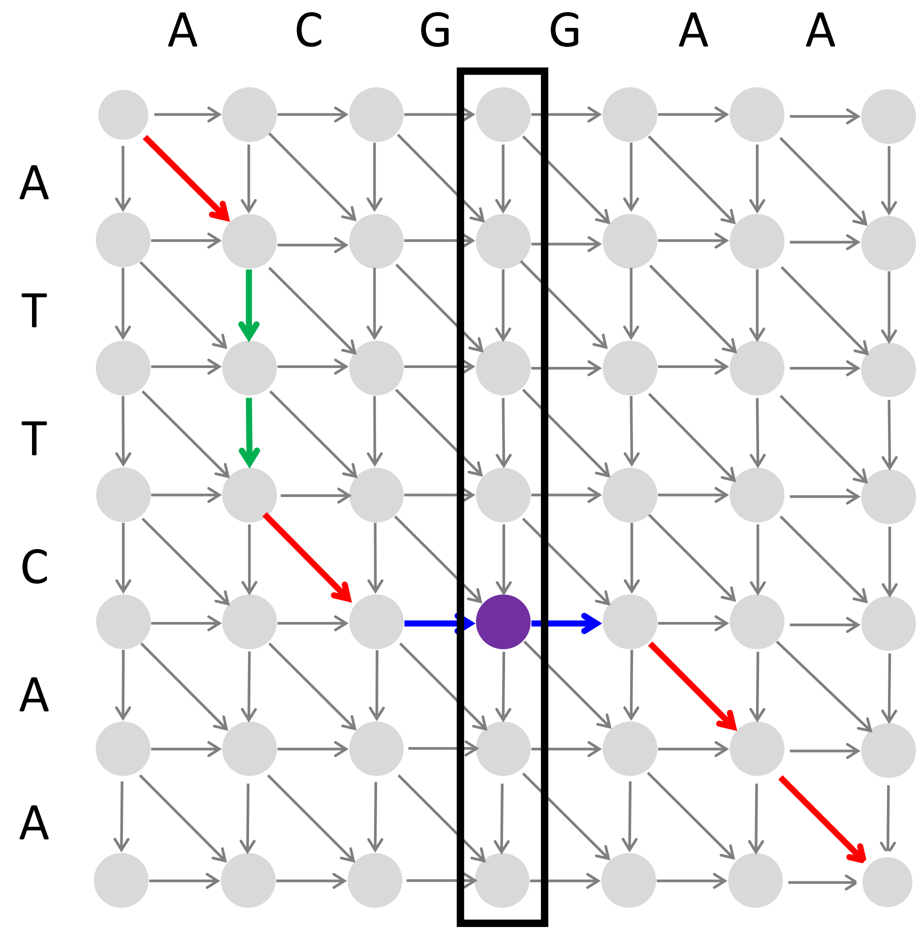
Finding the **longest path** in the alignment graph **requires** storing all backtracking pointers – O(*nm*) memory.

Finding the **length of the longest path** in the alignment graph **does not require** storing any backtracking pointers – O(*n*) memory.

Recycling the Columns in the Alignment Graph



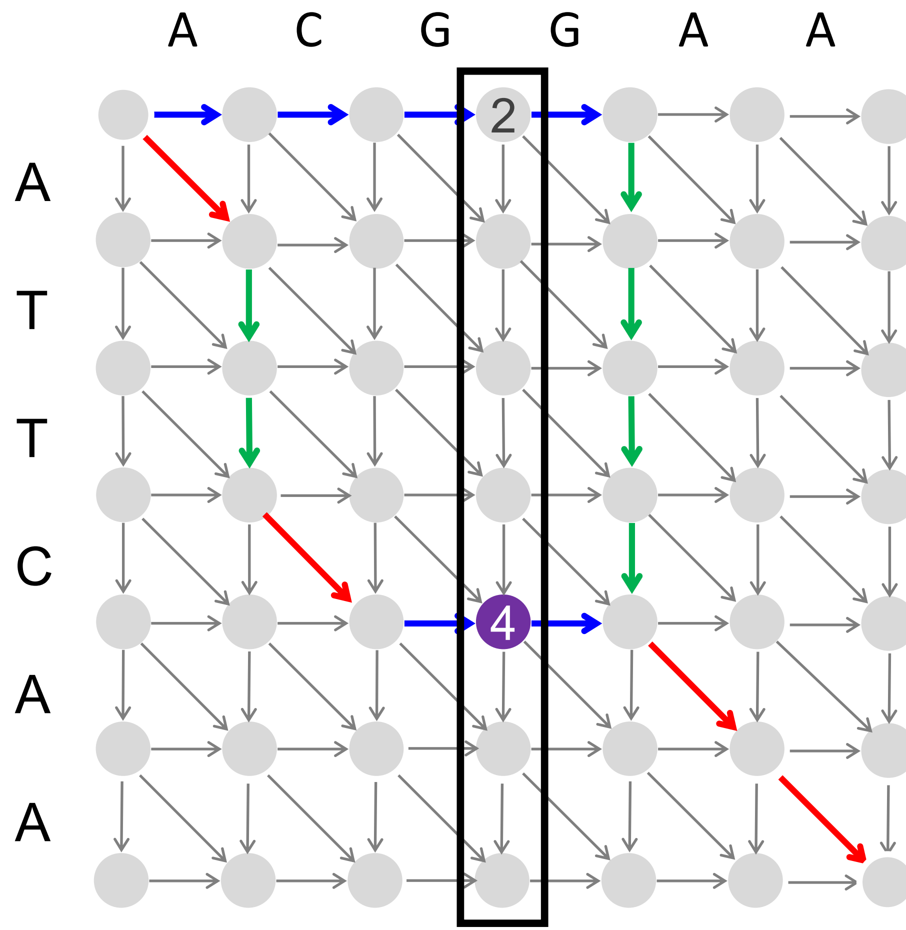
Can We Find the Middle Node without Constructing the Longest Path?



4-path that visits the node (4,middle) In the middle column

***i*-path** – a longest path among paths that visit the *i*-th node in the middle column

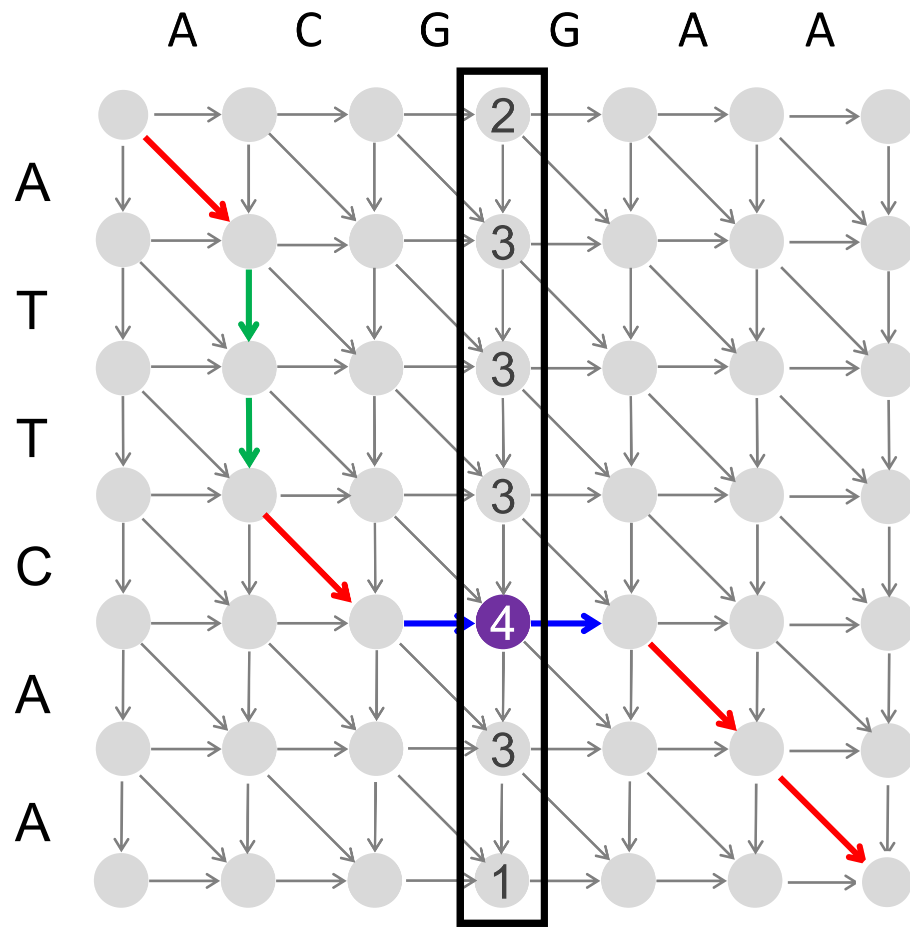
Can We Find The Lengths of All *i*-paths?



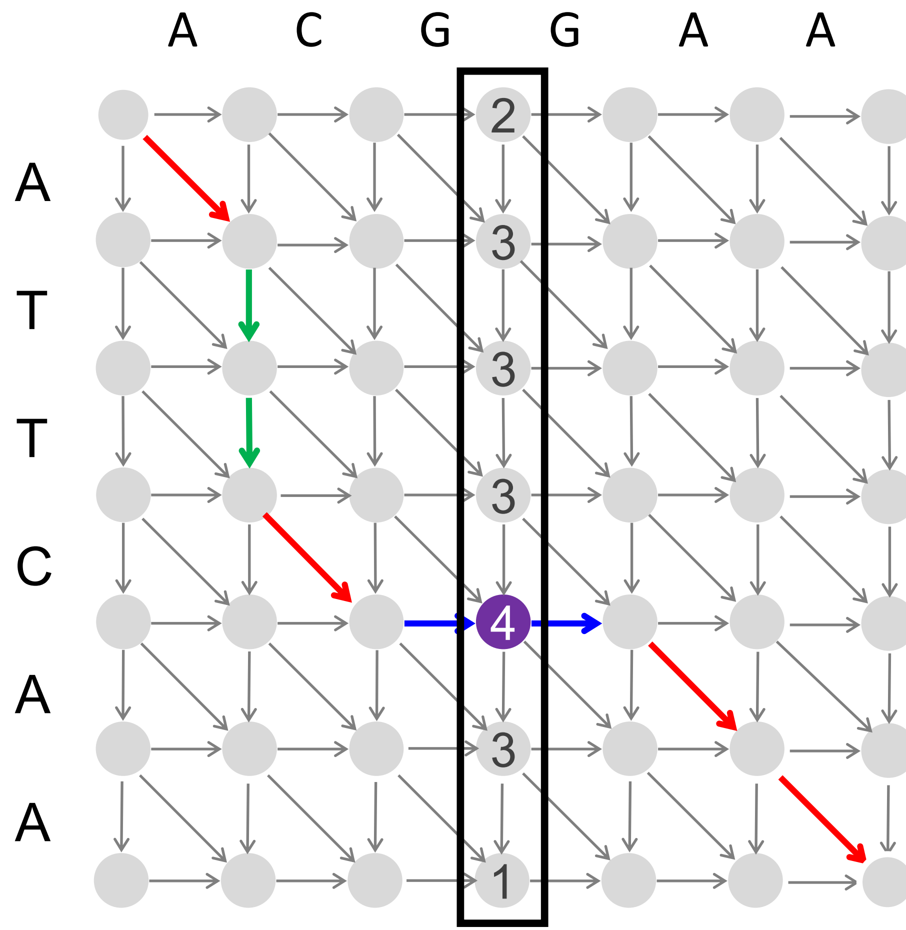
length(i):
length of an *i*-path:

length(0)=2
length(4)=4

Can We Find The Lengths of All *i*-paths?



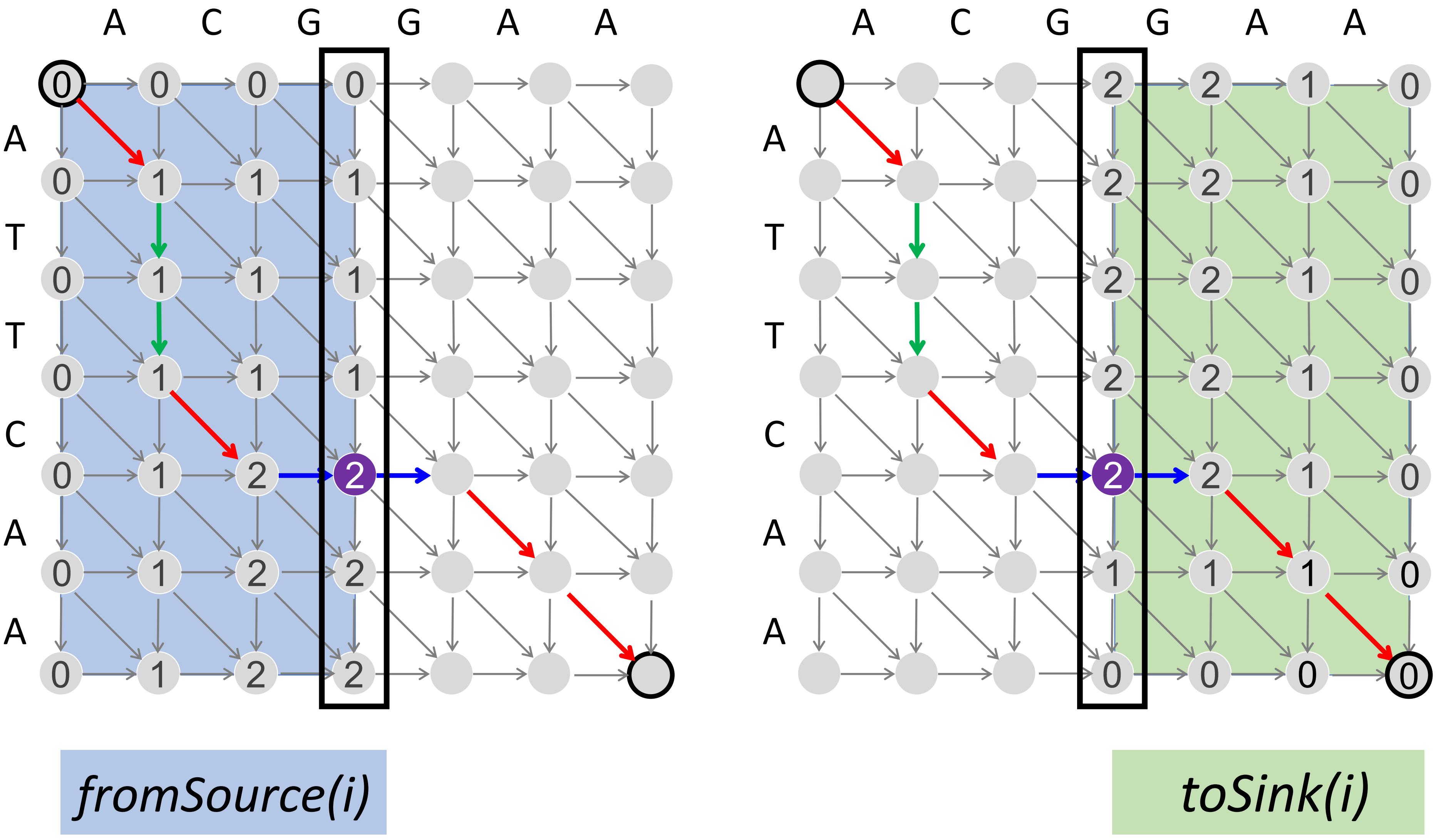
Can We Find The Lengths of *i*-paths?



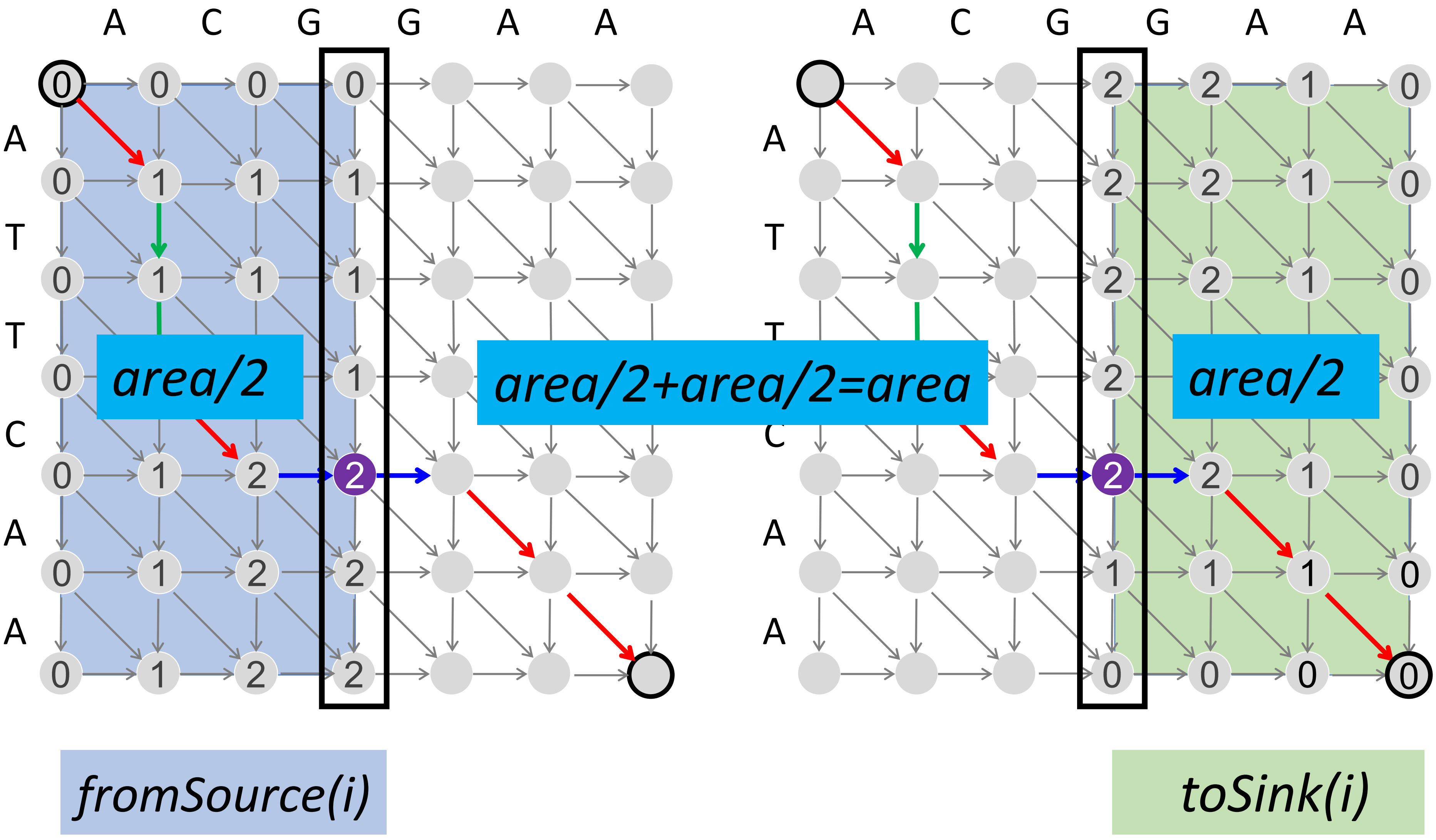
length(i):
length of an *i*-path

length(i)=fromSource(i)+toSink(i)

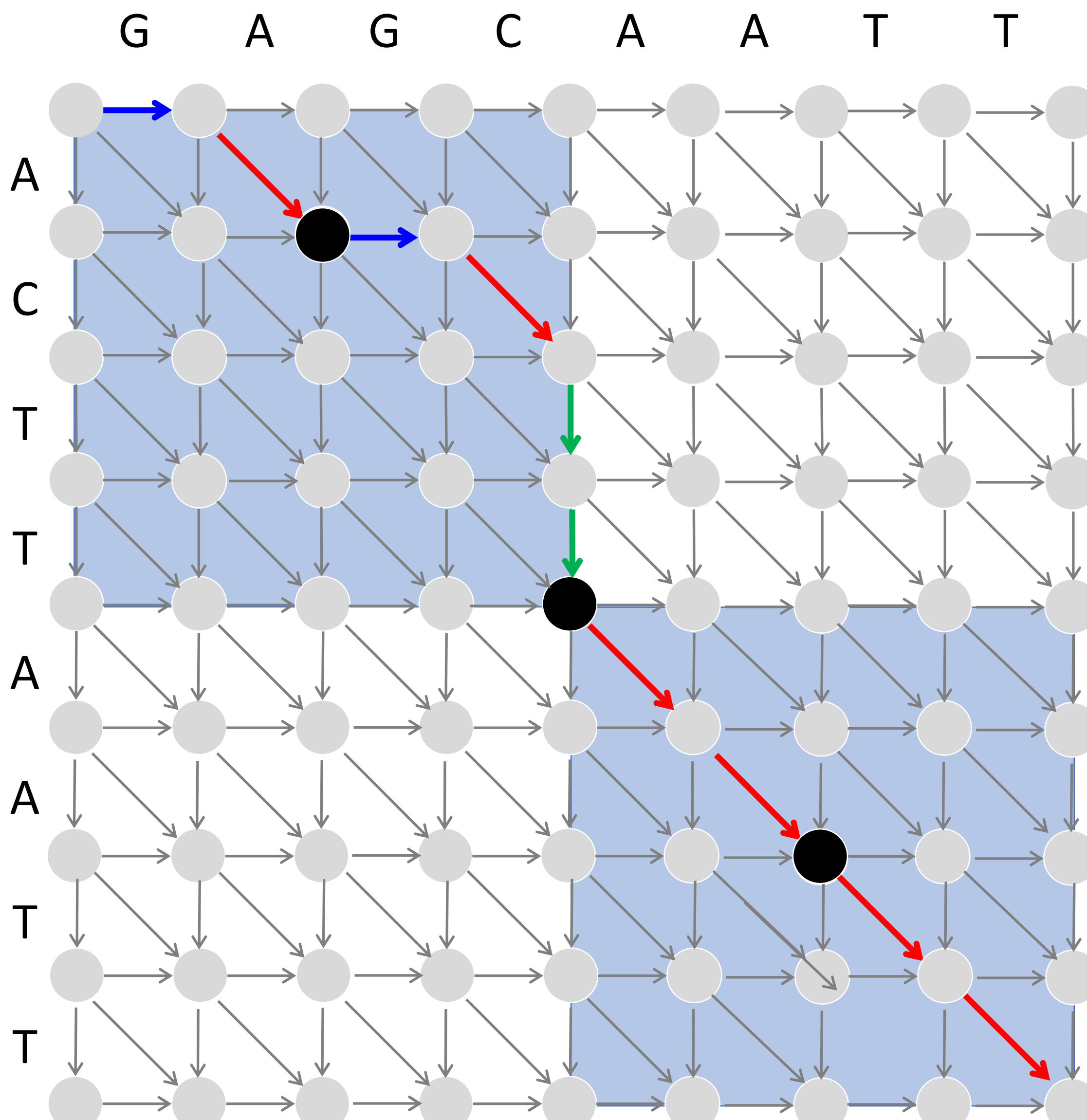
Computing *FromSource* and *toSink*



How Much Time Did It Take to Find the Middle Node ?



Laughable Progress: O(*nm*) Time to Find **ONE** Node!

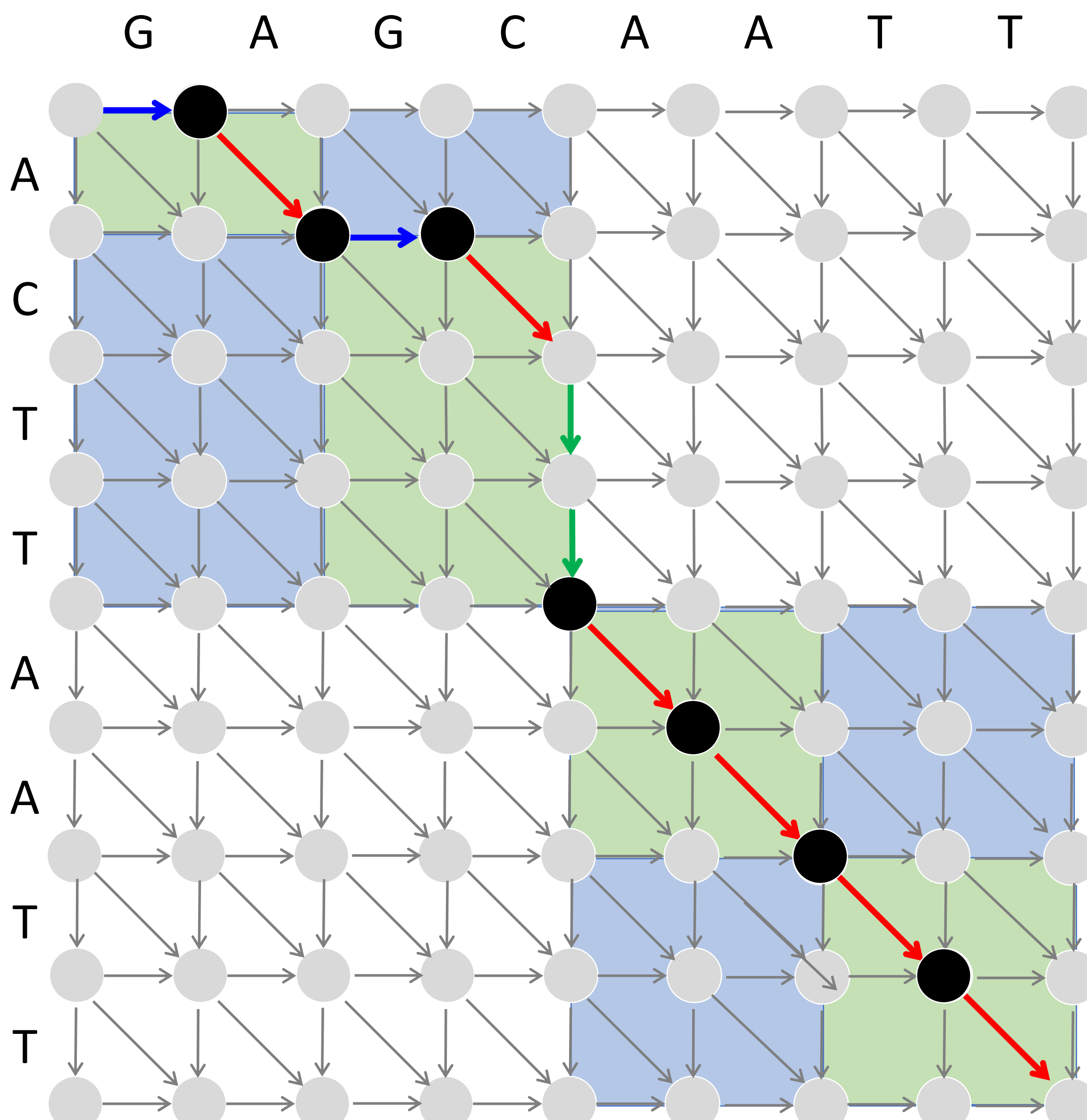


Each subproblem can be conquered in time proportional to its area:

area/4+area/4=
area/2

How much time would it take to conquer 2 subproblems?

Laughable Progress: O(*nm+nm/2*) Time to Find **THREE** Nodes!

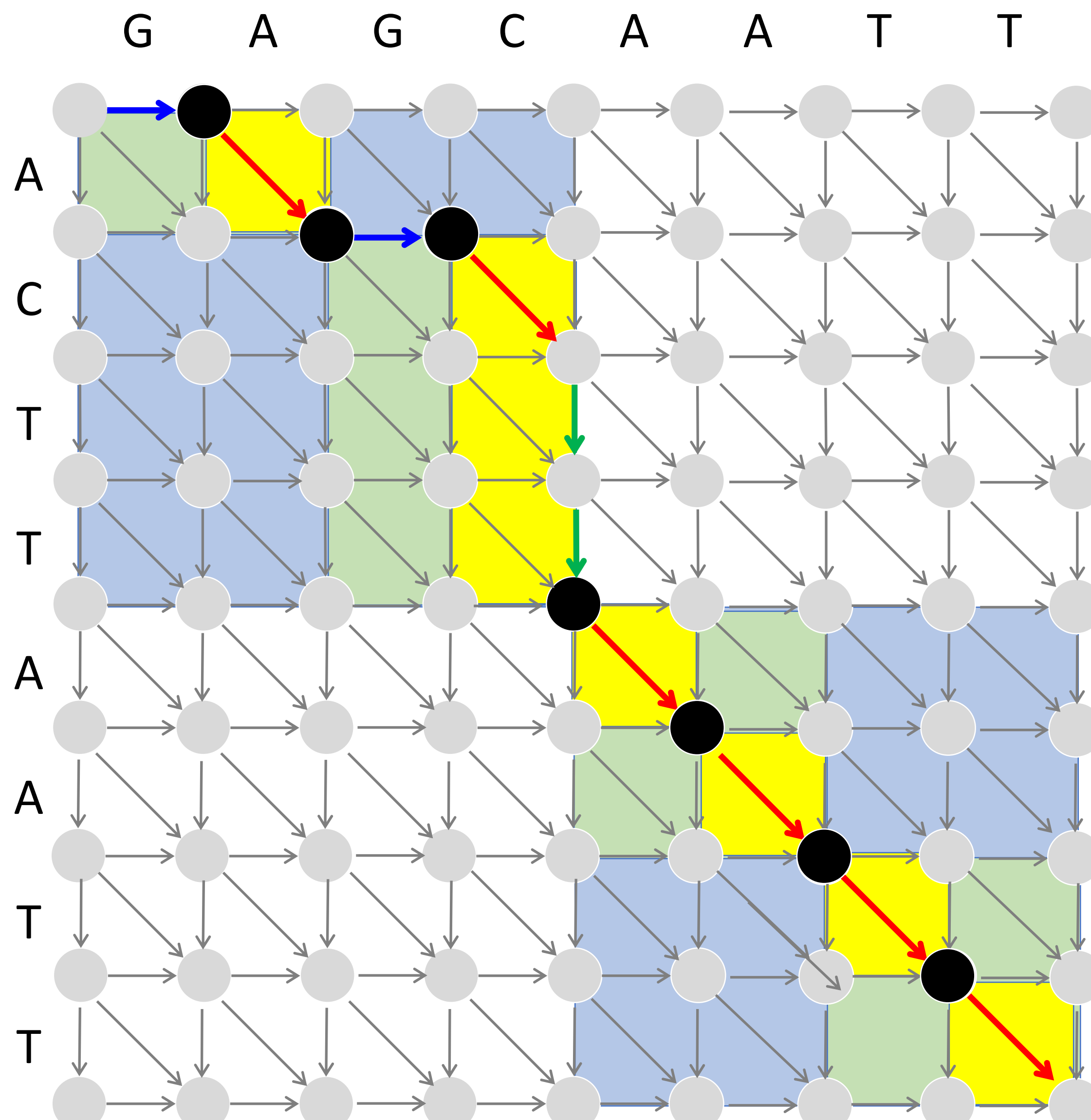


Each subproblem can be conquered in time proportional to its area:

area/8+area/8=
area/4

How much time would it take to conquer 4 subproblems?

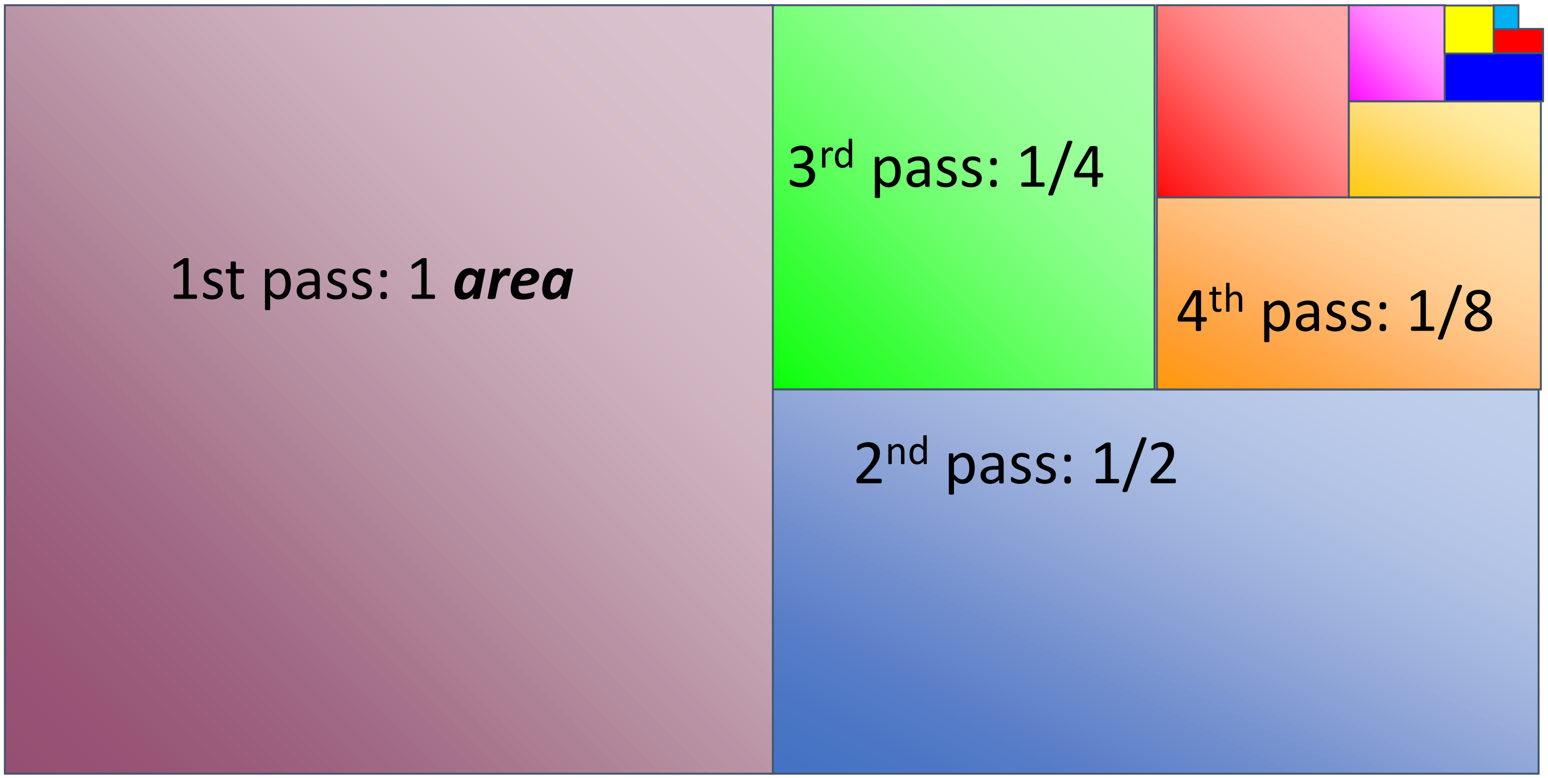
O($nm+nm/2+nm/4$) Time to Find **NEARLY ALL** Nodes!



area+
area/2
+area/4
+area/8
+area/16
....
+area/2^{lgn}

How much time would it take to conquer ALL subproblems?

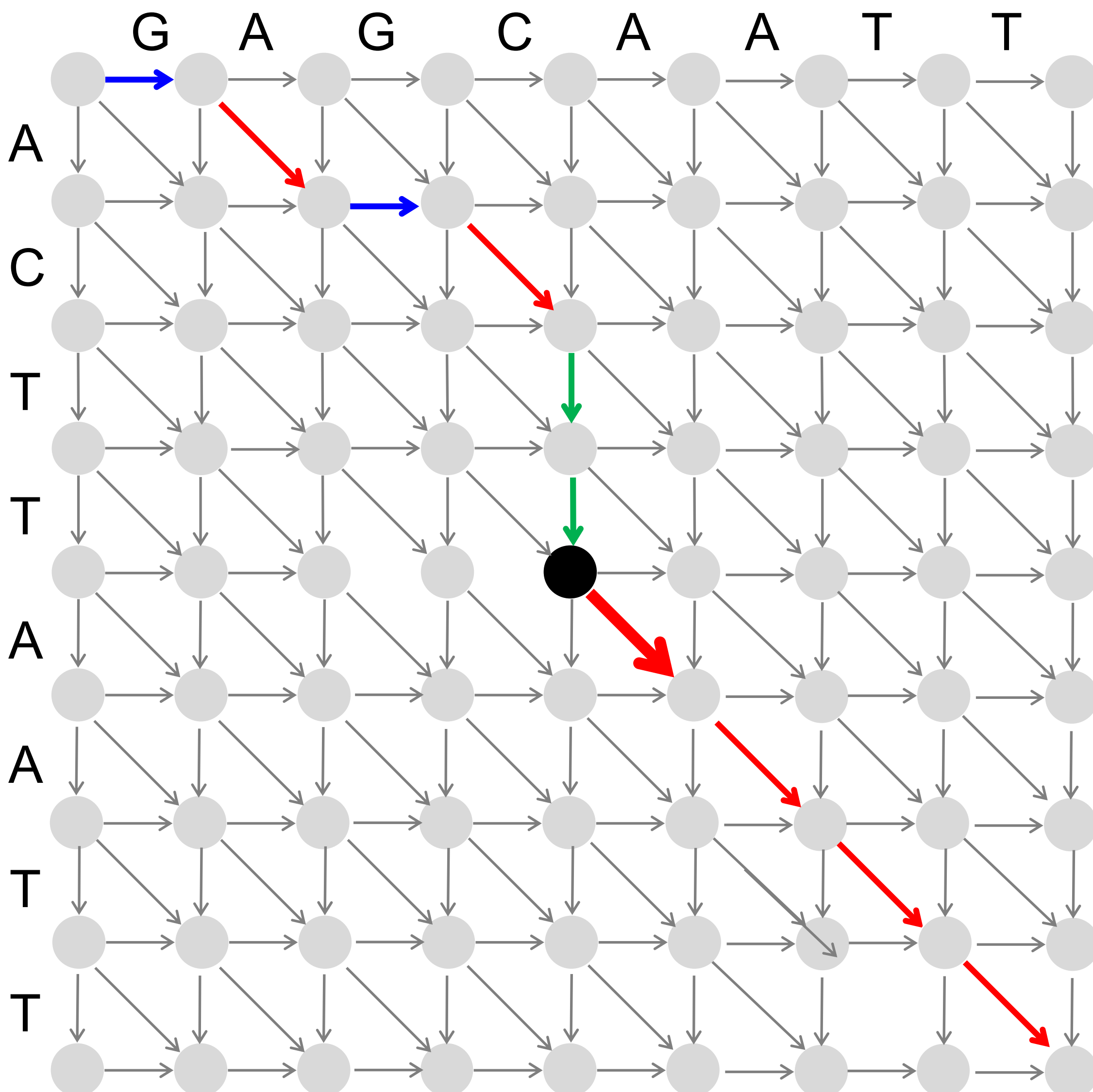
Total Time: ***area(1+1/2+1/4+1/8+1/16+…)***



$$1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^{\lg n}} = \frac{1 - \frac{1}{2n}}{1 - \frac{1}{2}} = 2 - \frac{1}{n} \ < \ 2$$

Still O(mn)!

The Middle Edge

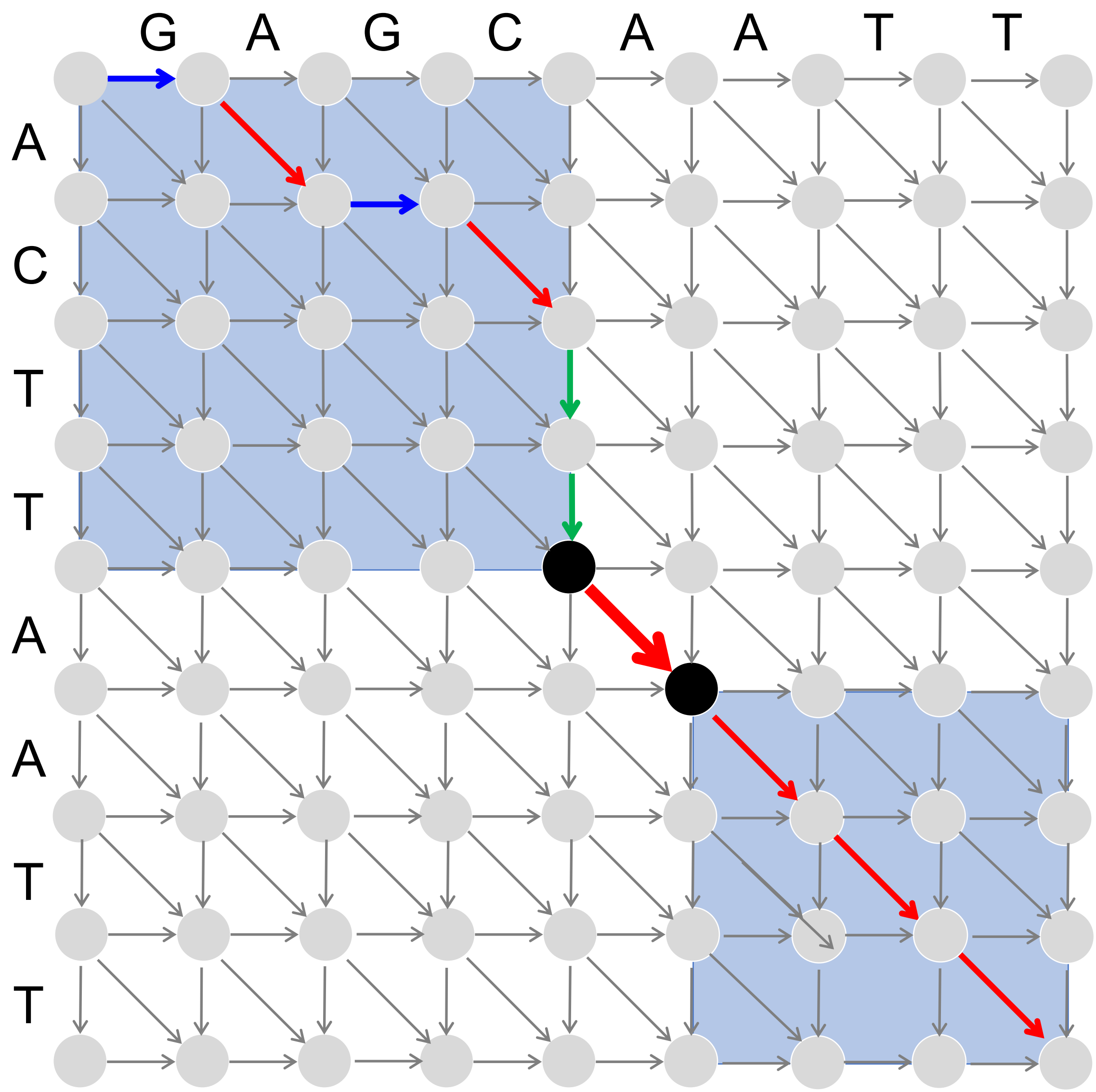


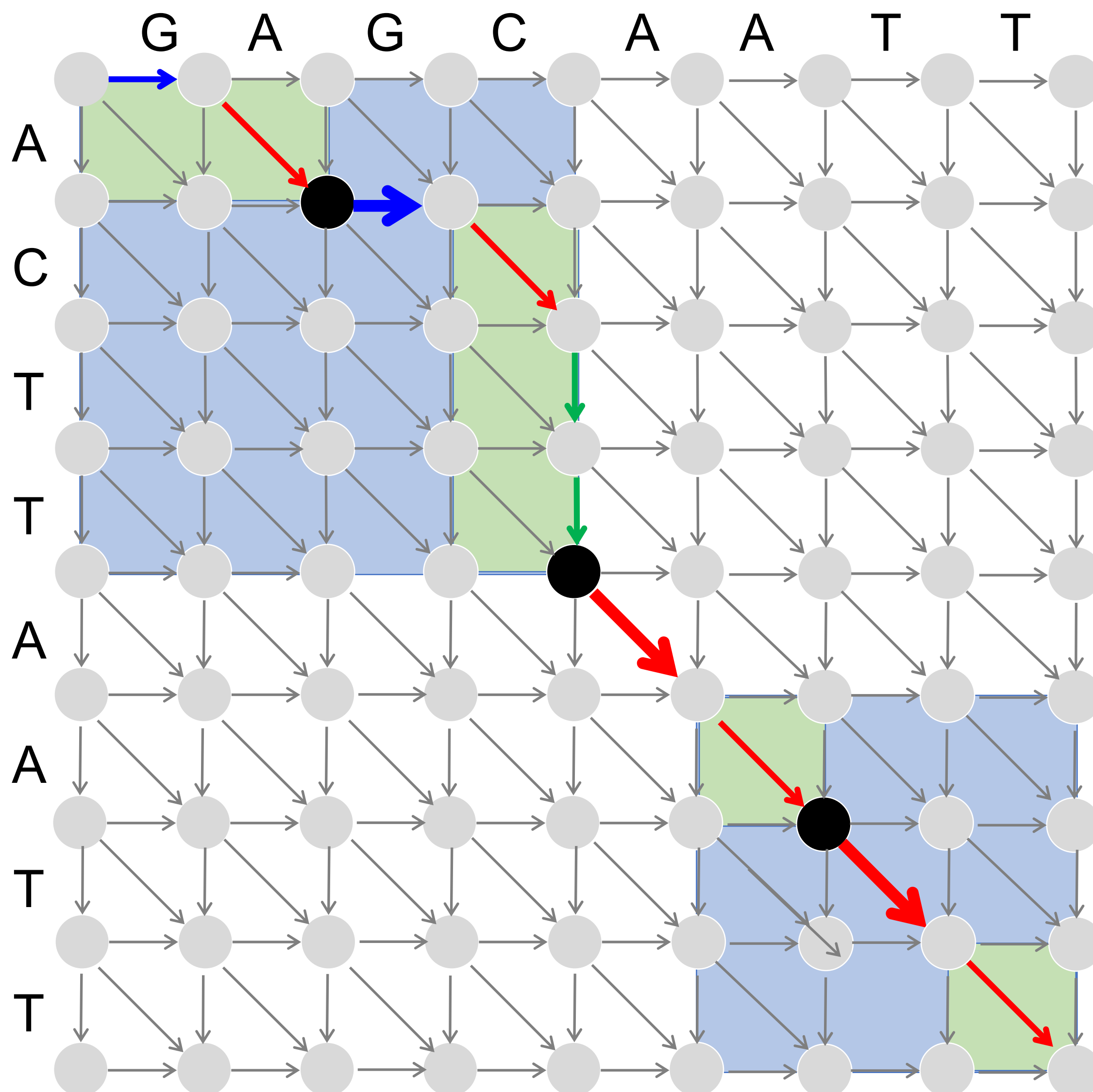
Middle Edge: an edge in an optimal alignment path starting at the middle node

The Middle Edge Problem

Middle Edge in Linear Space Problem. Find a middle edge in the alignment graph in linear space.

- **Input:** Two strings v and w and matrix ***score***.
- **Output:** A middle edge in the alignment graph of these strings (as defined by the matrix ***score***).


G A G C A A T T
A C T T A A T T


G A G C A A T T
A
C
T
T
A
A
T
T

Middle node and edge

- MiddleNodeEdge(v, w, top, bottom, left, right)
  - Finds the middle node and edge between

    $v_{top+1},\ldots,v_{bottom}$ and

    $w_{left+1}, \ldots, w_{right}$
  - mid <- $\lfloor (left+right)/2 \rfloor$
  - Apply linear-space dynamic programming with column reuse to get scores for aligning at the mid column:

    $v_{top+1},\ldots,v_{bottom}$ and $w_{left+1}, \ldots, w_{mid}$
  - Apply linear-space dynamic programming with column reuse to get scores for aligning at the mid column:

    $v_{bottom},\ldots, v_{top+1}$ and $w_{right}, \ldots, w_{mid}$
  - Identify both middle node and middle edge
  - Return (middleNode, middleEdge)

Recursive **LinearSpaceAlignment**

LinearSpaceAlignment(v, w, *top,bottom,left,right*)
  if *left = right*
    return alignment formed by *bottom-top* edges "↓"
  (midNode, midEdge) ← **MiddleNodeEdge***(v, w, top,bottom,left,right)*
  middle ← $\lfloor$*(left+right)/2*$\rfloor$
  LinearSpaceAlignment*(v, w, top,midNode,left,middle)*
  output *midEdge*
  if *midEdge = "→"* **or** *midEdge = "↘"*
    middle ← middle+1
  if *midEdge = "↓"* **or** *midEdge = "↘"*
    midNode ← midNode+1
  LinearSpaceAlignment*(v, w, midNode,bottom,middle,right)*