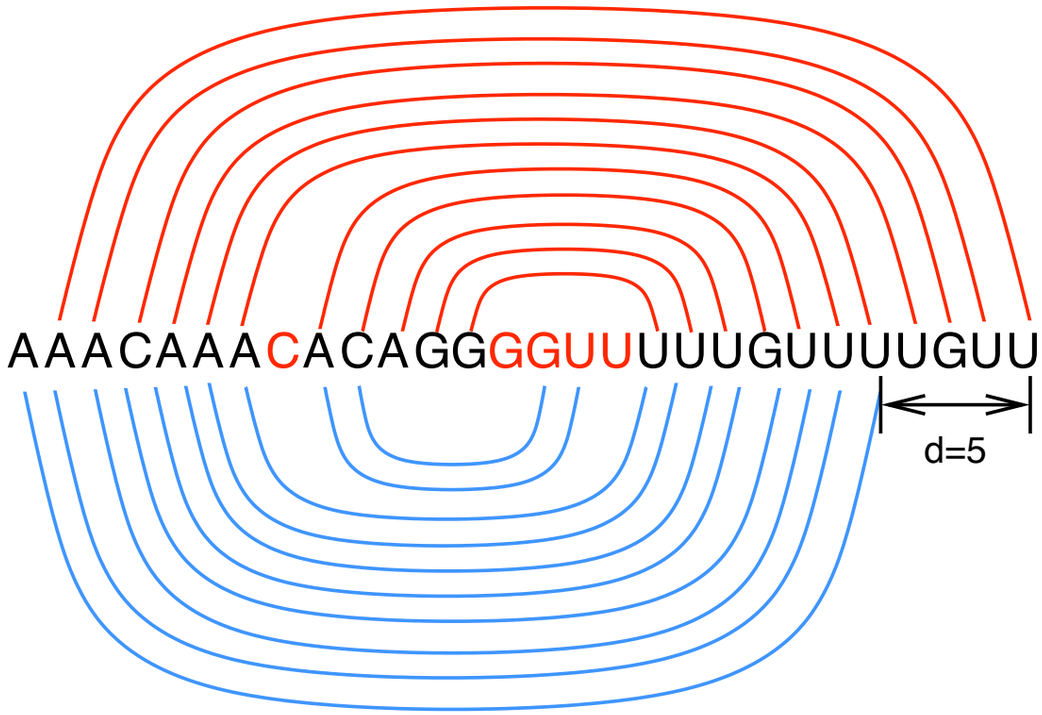
Fast RNA Structure Alignment for Crossing Input Structures



Rolf Backofen
Gad M. Landau
Mathias Möhl
Dekel Tsur
Oren Weimann

String Edit Distance

C o m b i n a t o r i a l O p t i m i s a t i o n

O p t i m a t o r i a l C o m b i n i s a t i o n

String Edit Distance

C o m b i n a t o r i a l O p t i m i s a t i o n

O p t i m a t o r i a l C o m b i n i s a t i o n

String Edit Distance

CombinatorialOptimisation

OptimatorialCombinisation

String Edit Distance

C o m b i n a t o r i a l O p t i m i s a t i o n

O p t i m a t o r i a l C o m b i n i s a t i o n

String Edit Distance

CombinatorialOptimisation

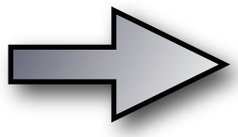
OptimatorialCombinisation

Every two prefixes

String Edit Distance

Combinatorial Optimisation

Optimatorial Combinisation

Every two prefixes  $O(n^2)$

String Edit Distance

CombinatorialOptimisation

OptimatorialCombinisation

Every two infixes

String Edit Distance

C o m b i n a t o r i a l O p t i m i s a t i o n

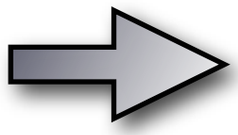
O p t i m a t o r i a l C o m b i n i s a t i o n

Every two infixes

String Edit Distance

C o m b i n a t o r i a l O p t i m i s a t i o n

O p t i m a t o r i a l C o m b i n i s a t i o n

Every two infixes  $O(n^4)$

RNA Edit Distance

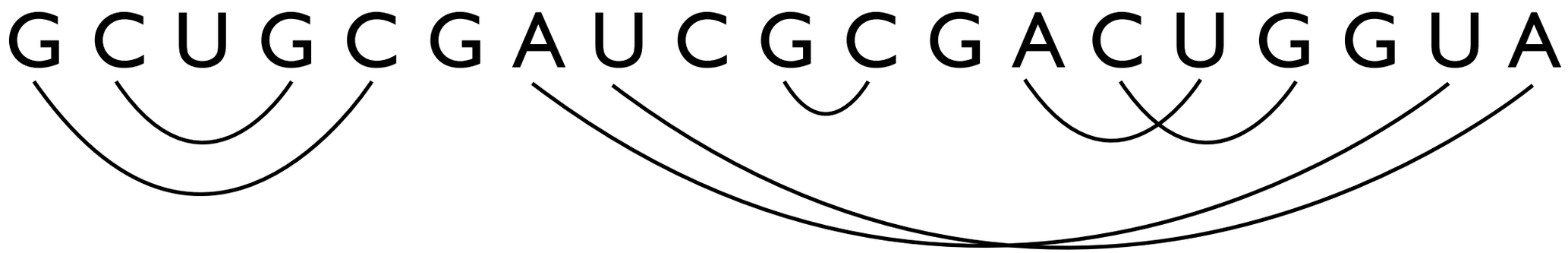
A G C U C A G G A U G U C A G U G A C

G C U G C G A U C G C G A C U G G U A

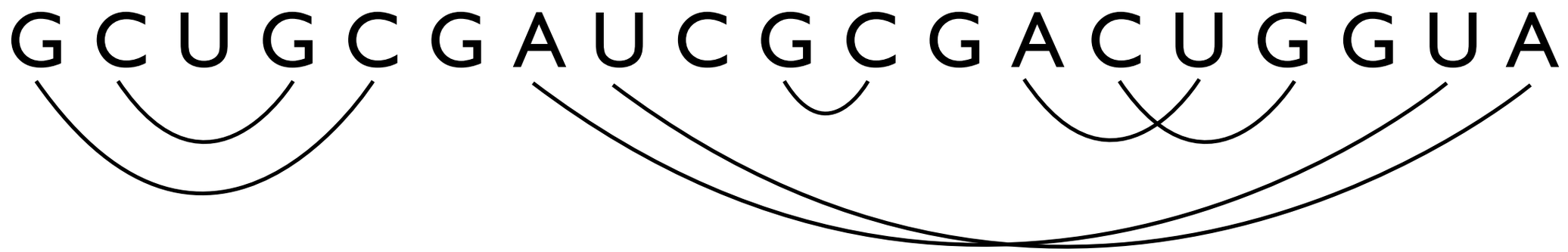
RNA Edit Distance

G C U G C G A U C G C G A C U G G U A

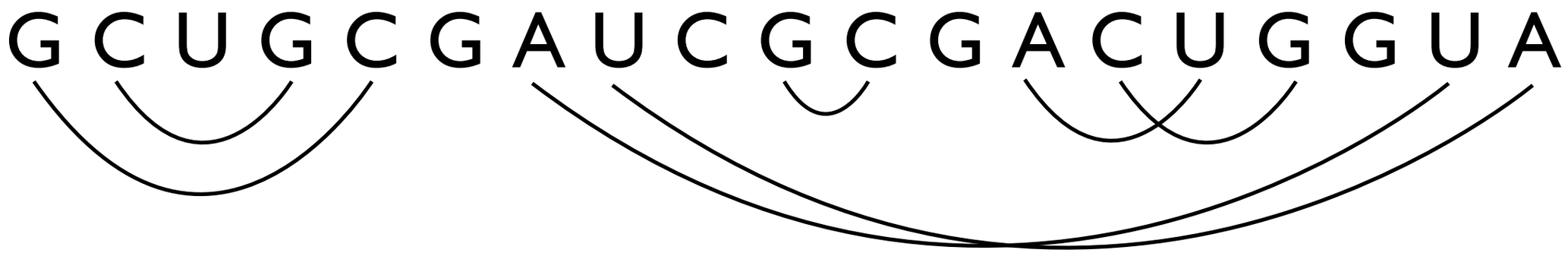
RNA Edit Distance



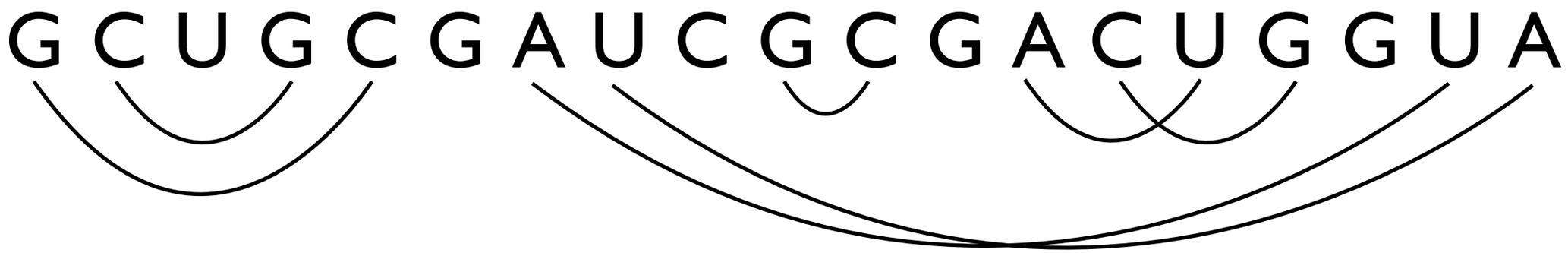
RNA Edit Distance



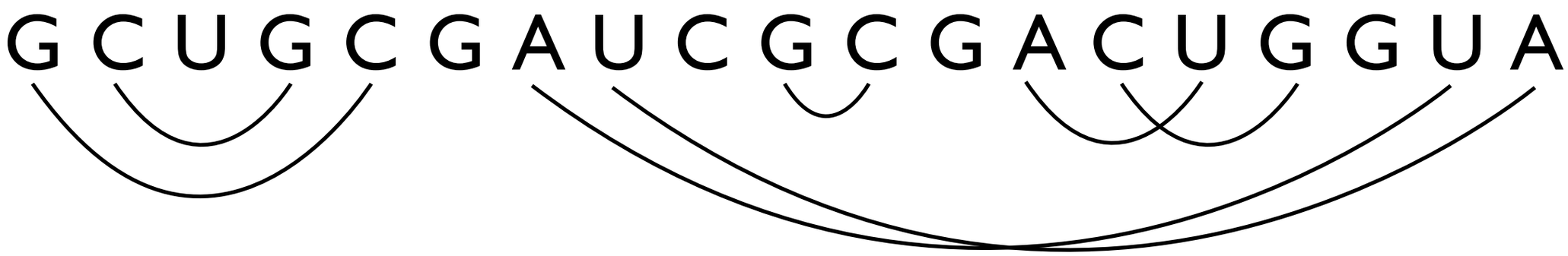
RNA Edit Distance



RNA Edit Distance

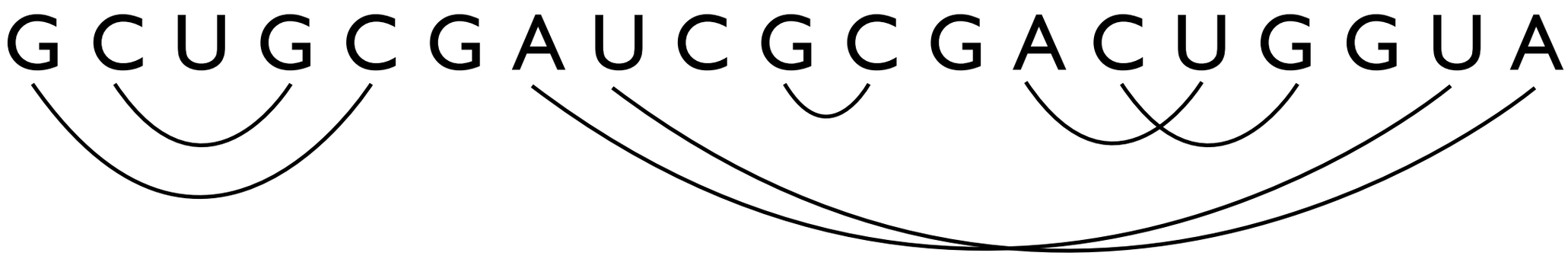


RNA Edit Distance



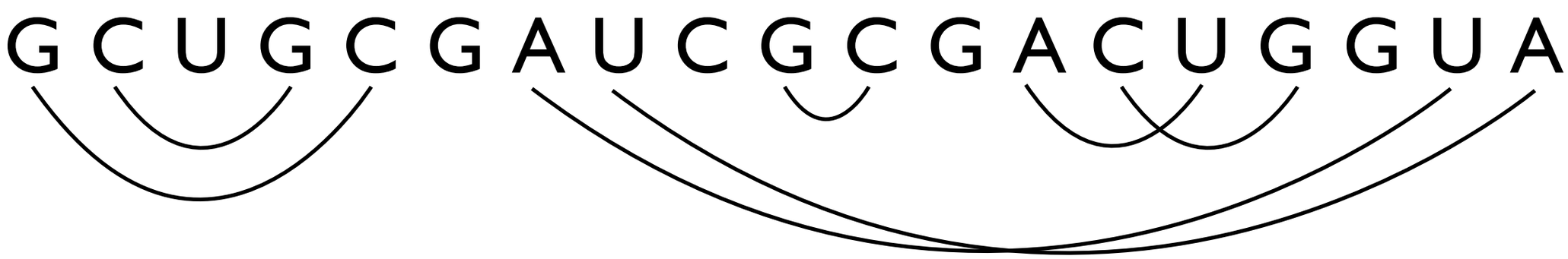
Additional edit operations:

RNA Edit Distance



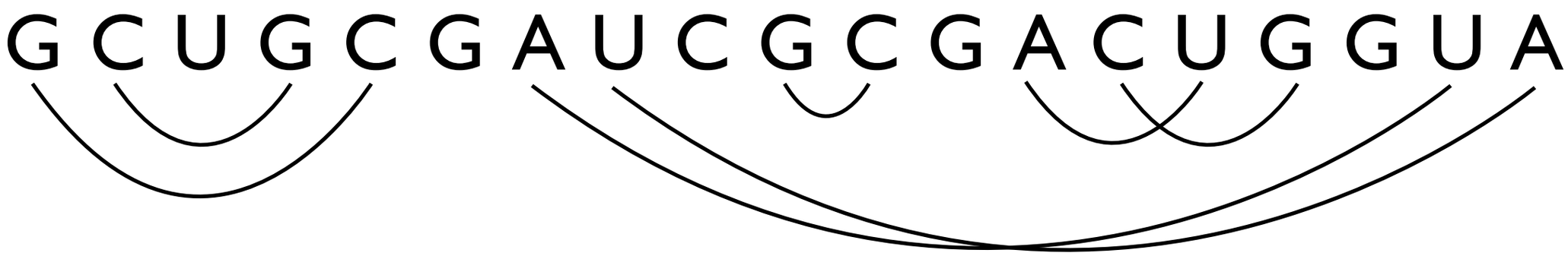
Additional edit operations: Arc Deletion / Insertion

RNA Edit Distance



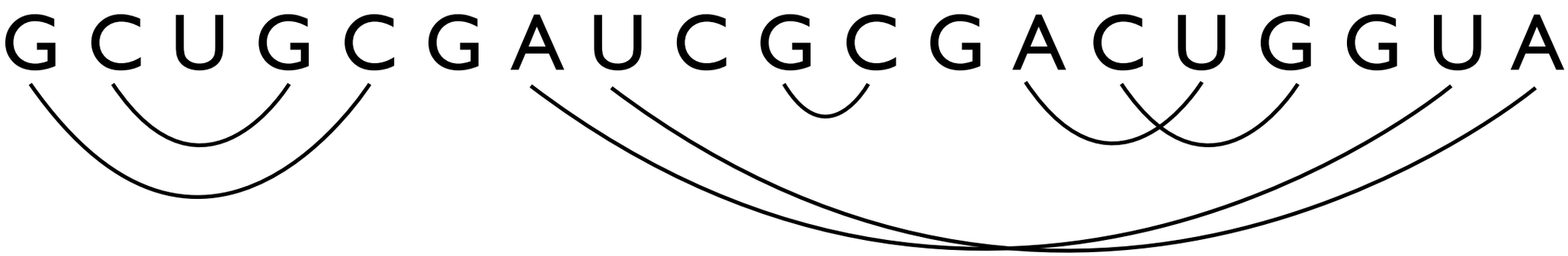
Additional edit operations: Arc Deletion / Insertion

RNA Edit Distance



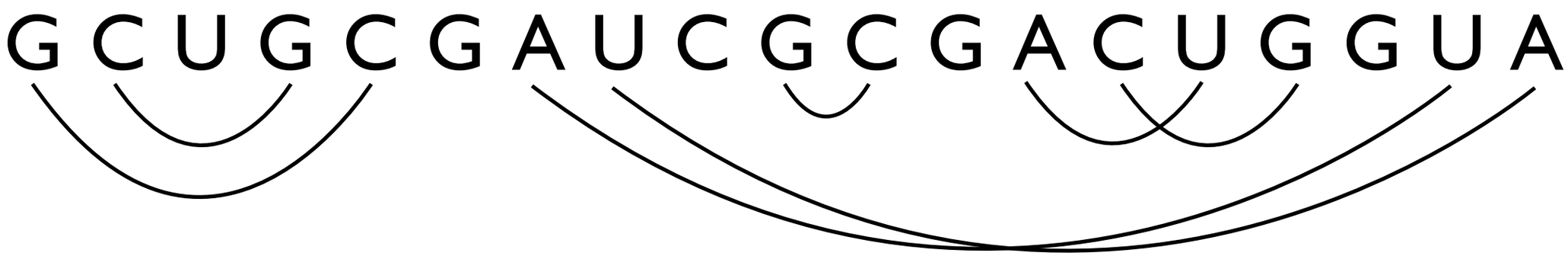
Additional edit operations:

RNA Edit Distance



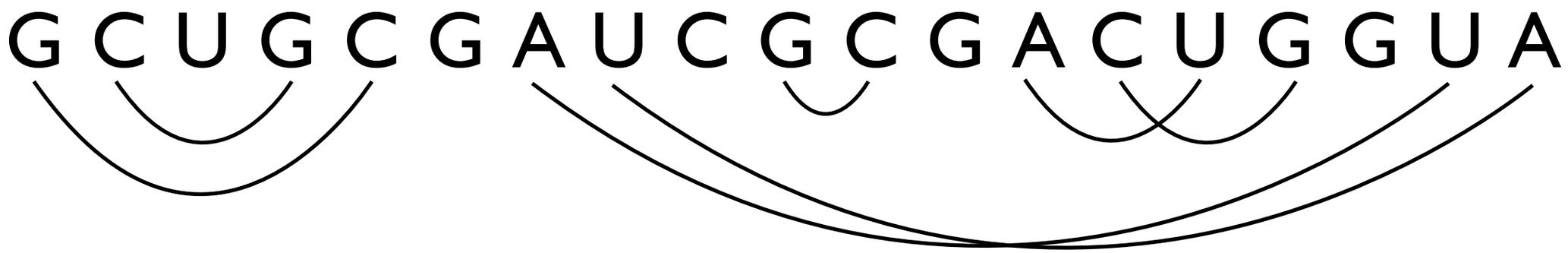
Additional edit operations: Arc Match / Relabel

RNA Edit Distance

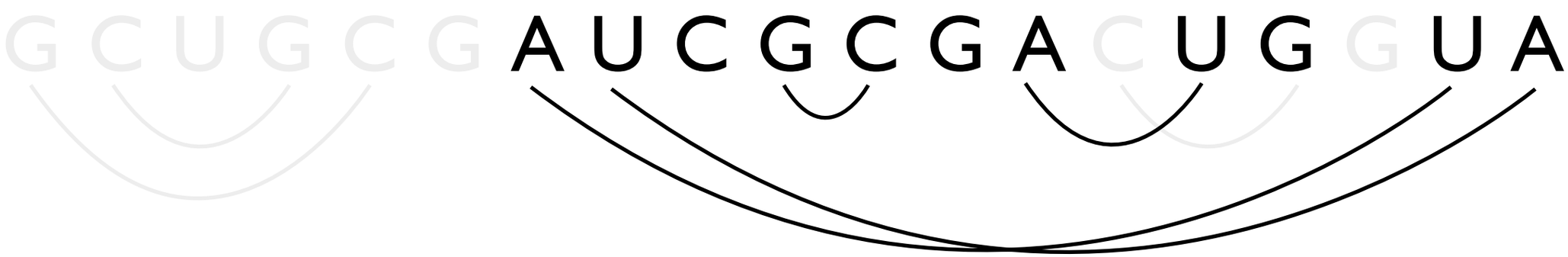


Additional edit operations: Arc Match / Relabel

RNA Edit Distance



RNA Edit Distance

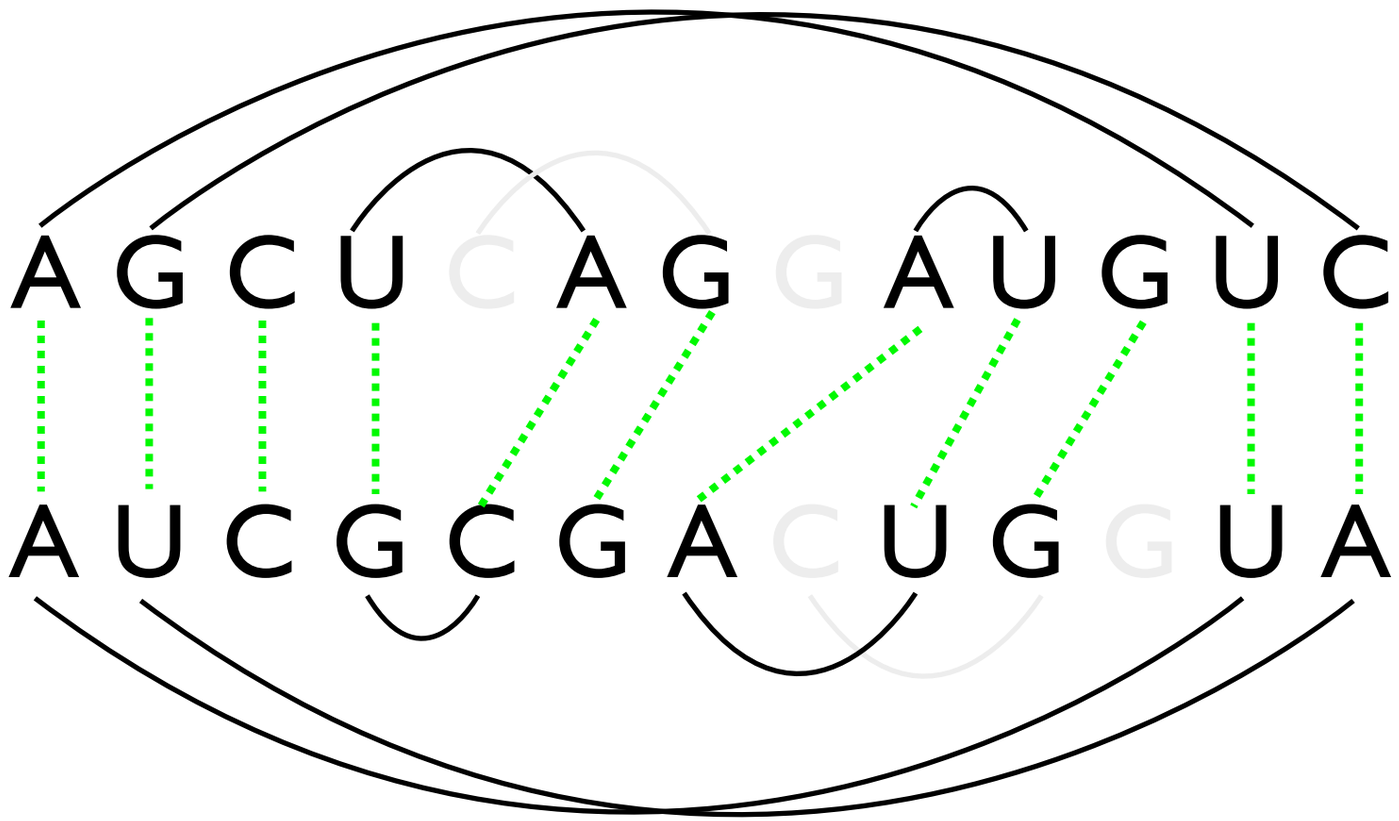


Consensus Structure

RNA Edit Distance

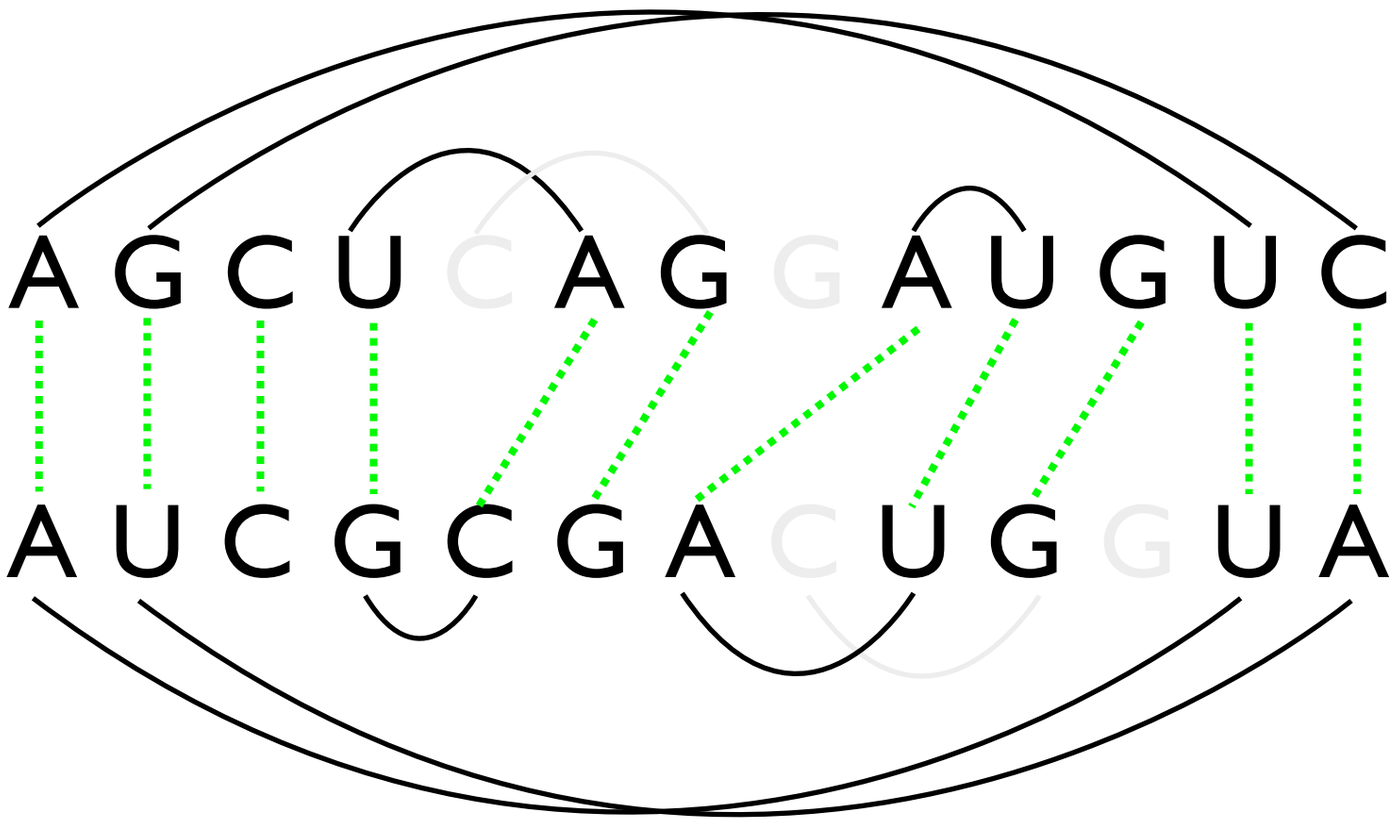
Consensus Structure

RNA Edit Distance



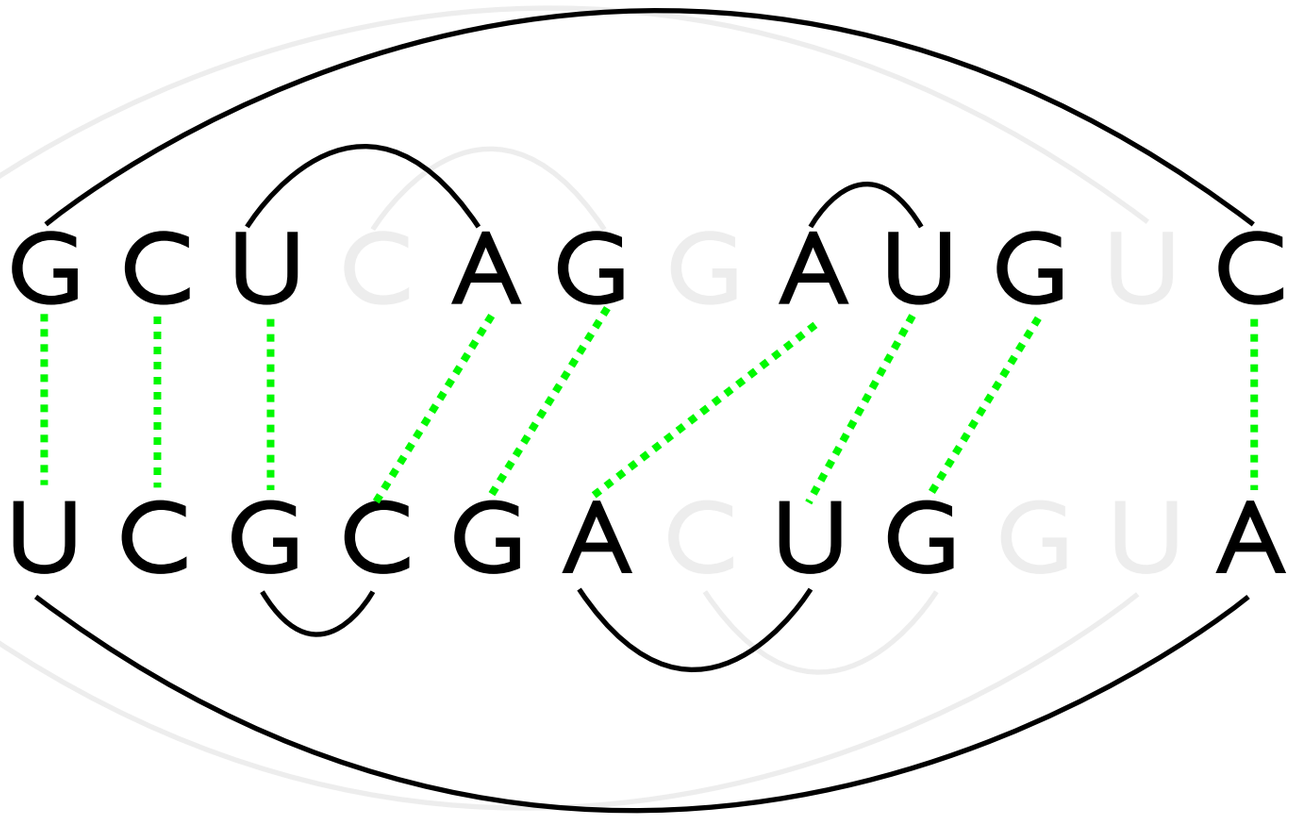
Consensus Structure

RNA Edit Distance



Arbitrary Consensus Structure: NP-hard

RNA Edit Distance



[Jiang *et al.* 2002]

~~Arbitrary~~ Consensus Structure: $O(n^4)$

Non-crossing

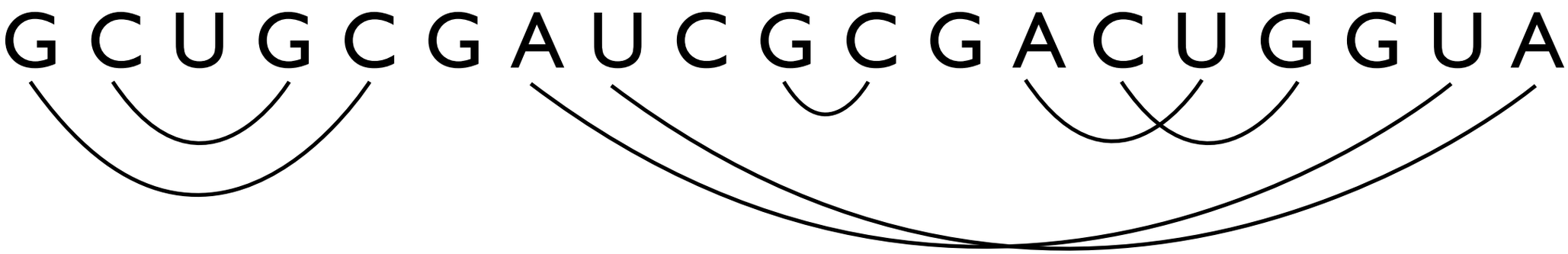
Non-Crossing Consensus

~~Arbitrary~~ Consensus Structure:
Non-crossing

[Jiang *et al.* 2002]

$O(n^4)$

Non-Crossing Consensus



[Jiang *et al.* 2002]

~~Arbitrary~~ Consensus Structure: $O(n^4)$

Non-crossing

Non-Crossing Consensus

Every two infixes $\Rightarrow$ $O(n^4)$

Non-Crossing Consensus

Every two infixes $\Rightarrow$ $O(n^4)$

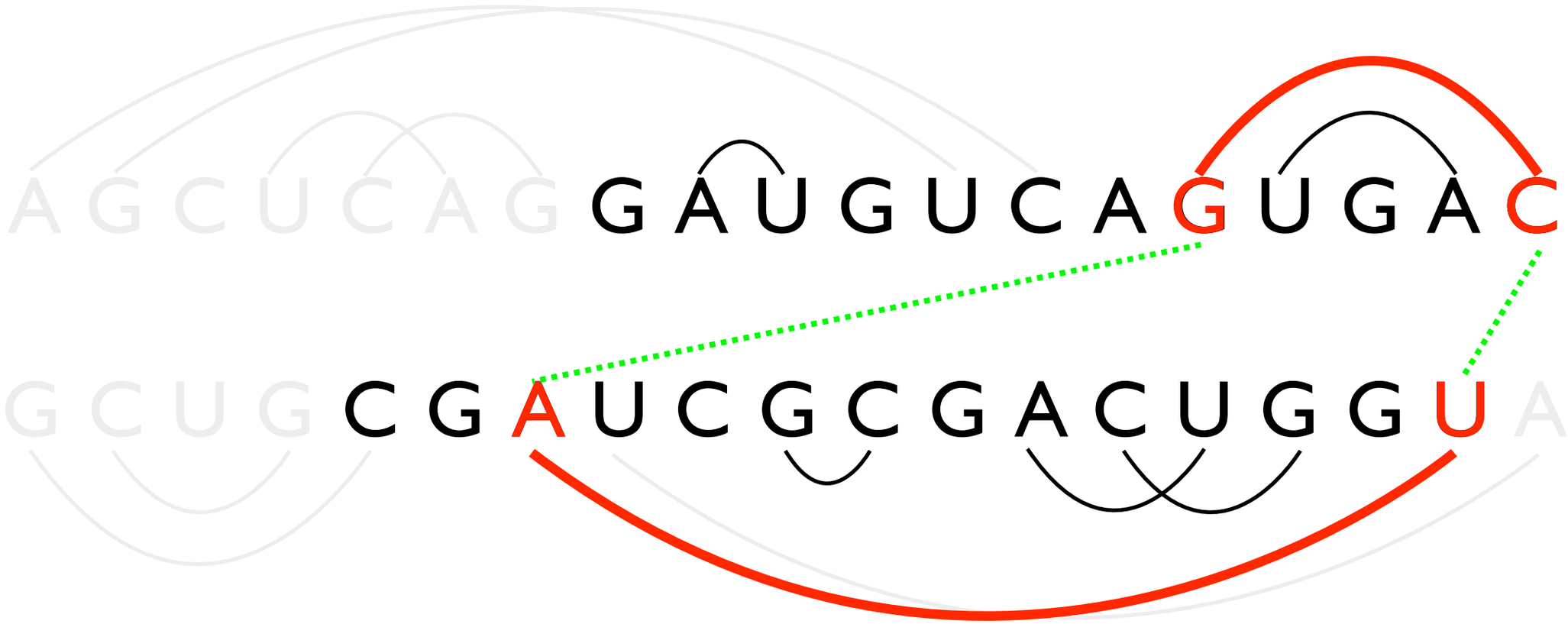
Non-Crossing Consensus

Every two infixes $\Rightarrow$ $O(n^4)$

Non-Crossing Consensus

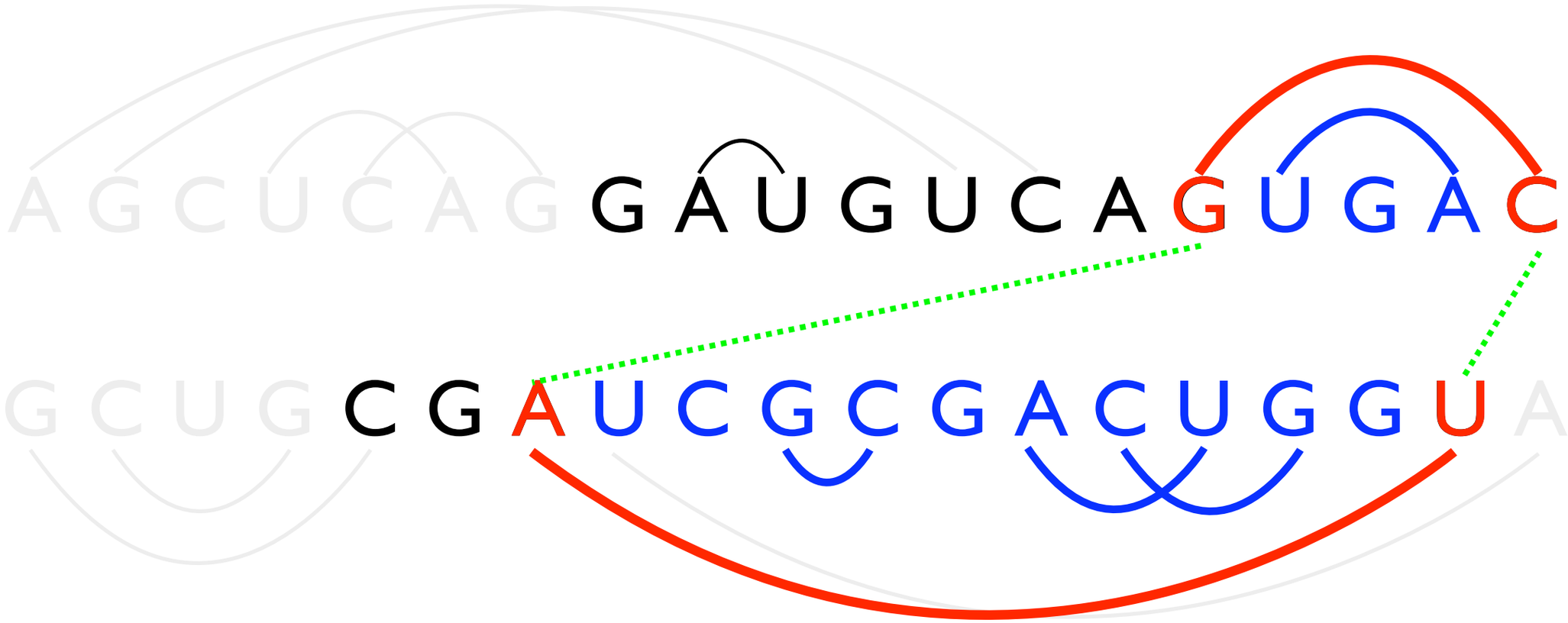
Arc Match / Relabel

Non-Crossing Consensus



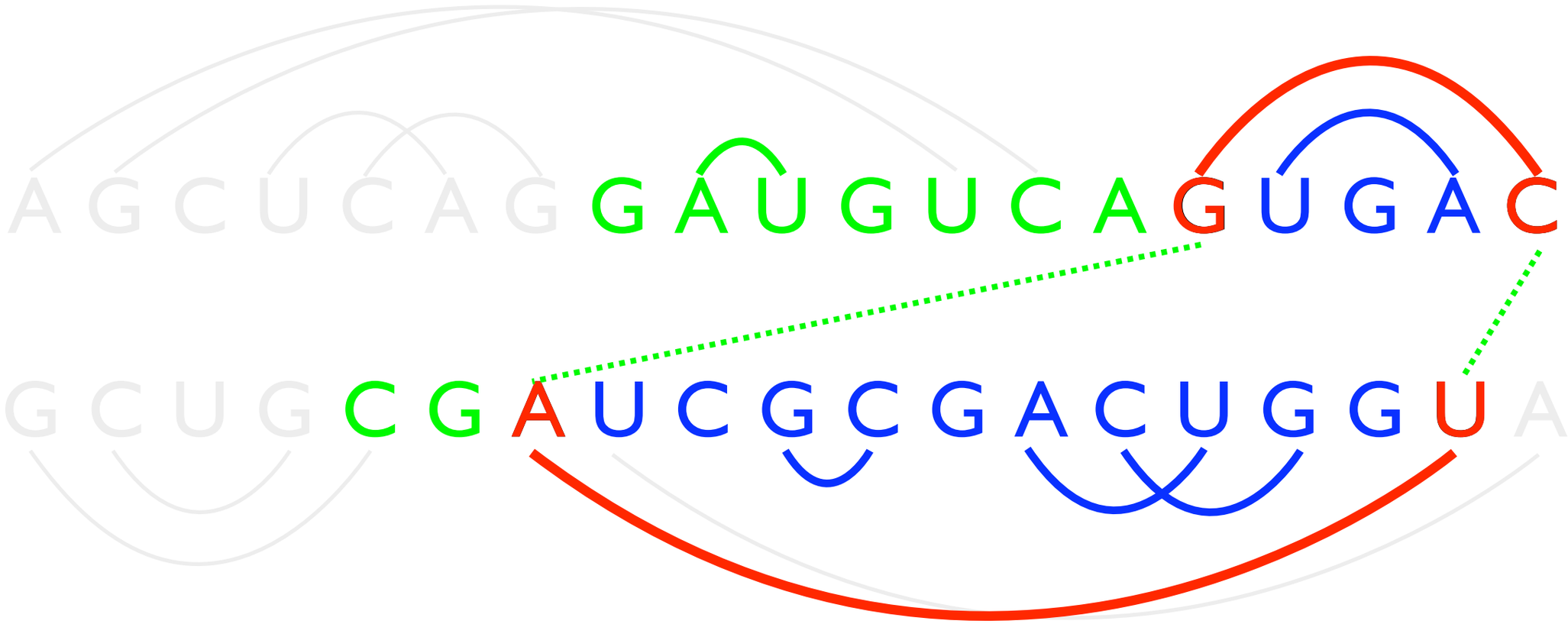
Arc Match / Relabel

Non-Crossing Consensus



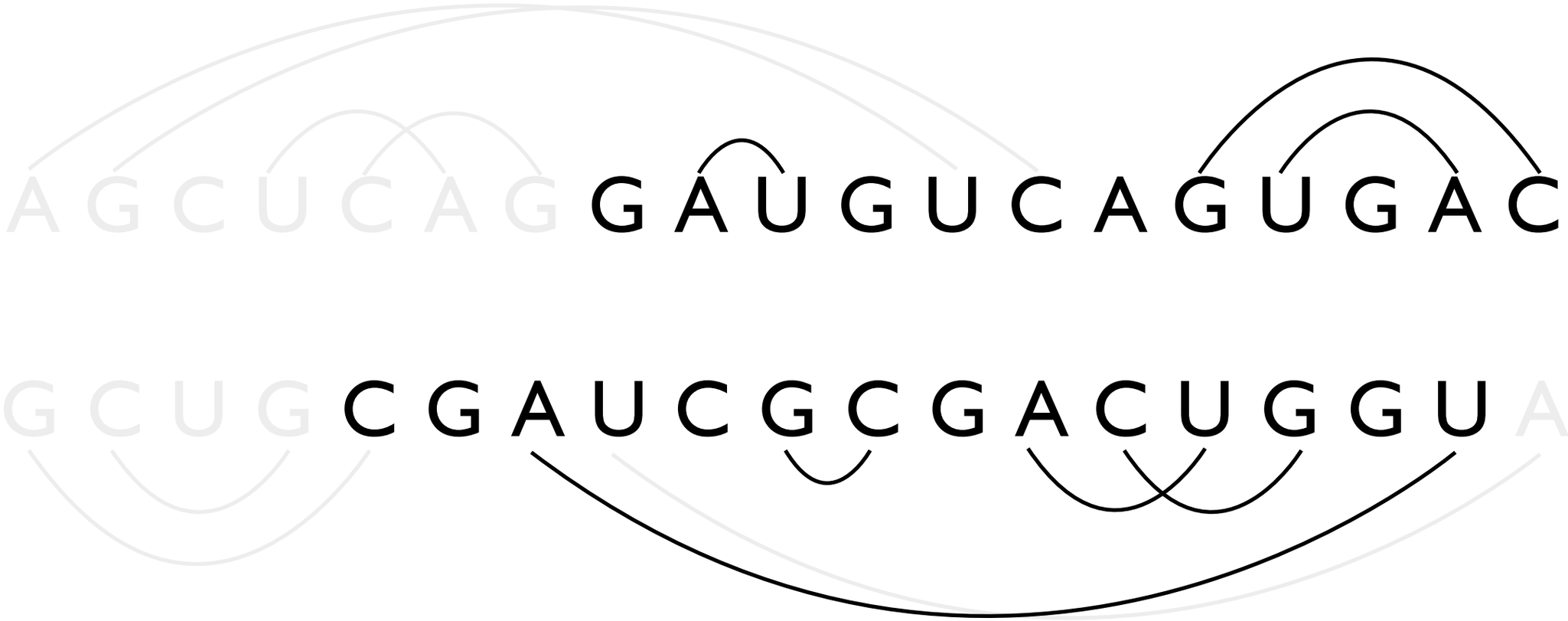
Arc Match / Relabel

Non-Crossing Consensus



Arc Match / Relabel

Non-Crossing Consensus



Character Match / Relabel

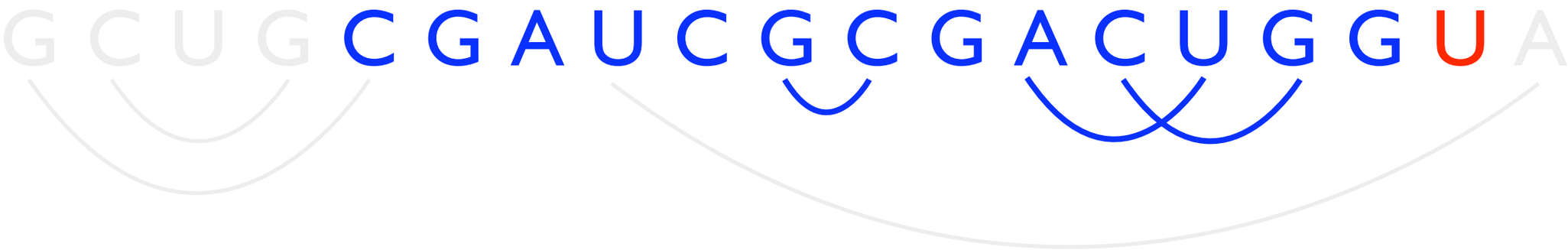
Non-Crossing Consensus

Character Match / Relabel

Non-Crossing Consensus

Character Match / Relabel

Non-Crossing Consensus



Character Match / Relabel

Non-Crossing Consensus

Character Deletion / Insertion

Non-Crossing Consensus

Character Deletion / Insertion

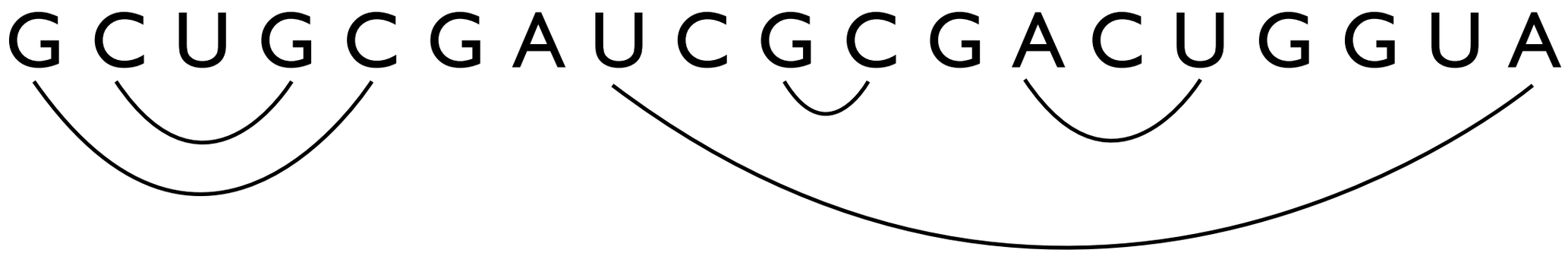
Non-Crossing Consensus

Character Deletion / Insertion

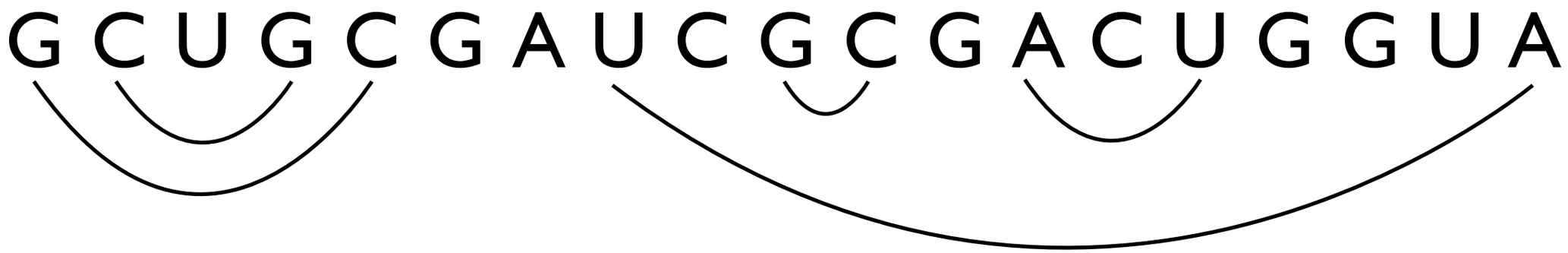
Non-Crossing Consensus

Character Deletion / Insertion

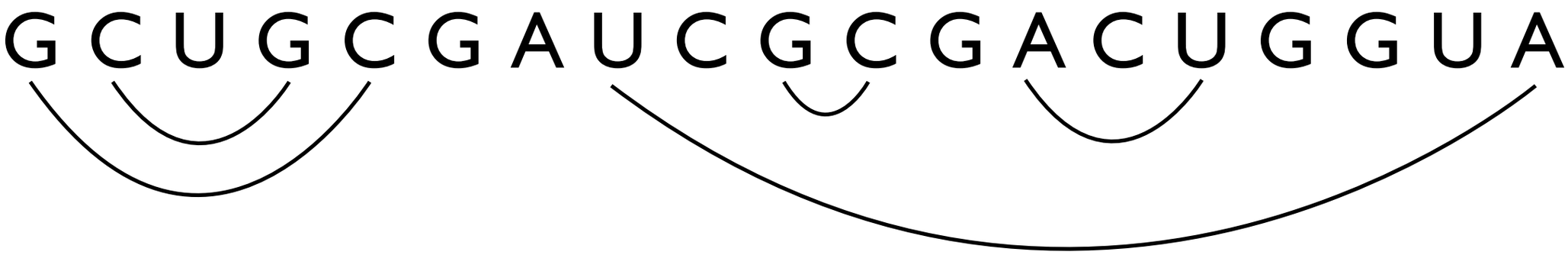
Non-Crossing Inputs



Non-Crossing Inputs
= Tree Edit distance

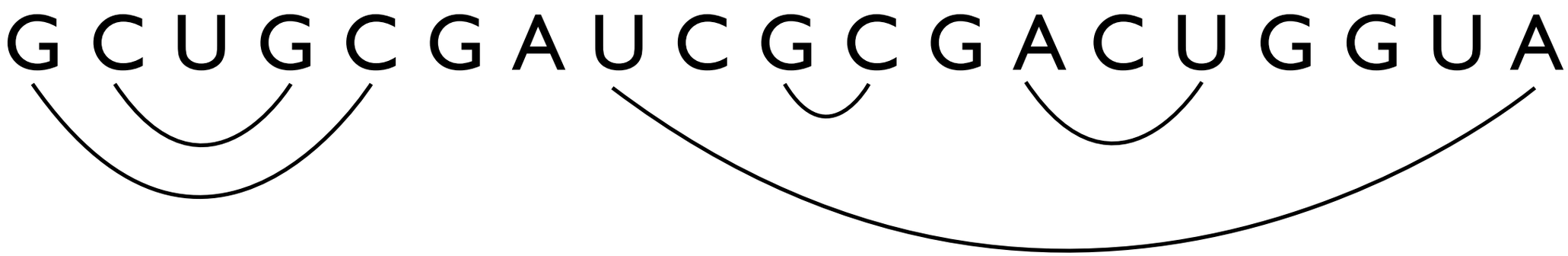


Non-Crossing Inputs
= Tree Edit distance



$O(n^6)$	[Tai 1979]
$O(n^4)$	[Shasha, Zhang 1989]
$O(n^3 \log n)$	[Klein 1998] [Dulucq, Touzet 2003] [Bille 2005]
$O(n^3)$	[Demaine, Mozes, Rossman, W, 2007]

Non-Crossing Inputs
= Tree Edit distance

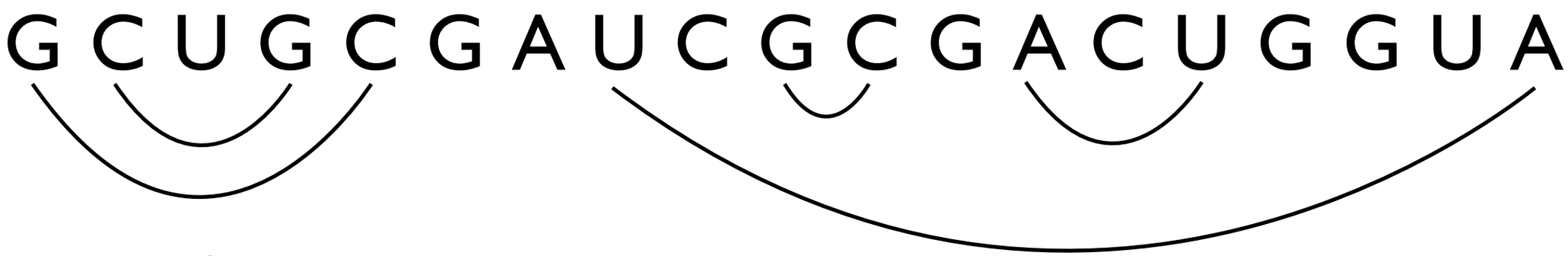


$O(n^3 \log n)$ [Klein 1998]

Non-Crossing Inputs
= Tree Edit distance

$O(n^3 \log n)$ [Klein 1998]

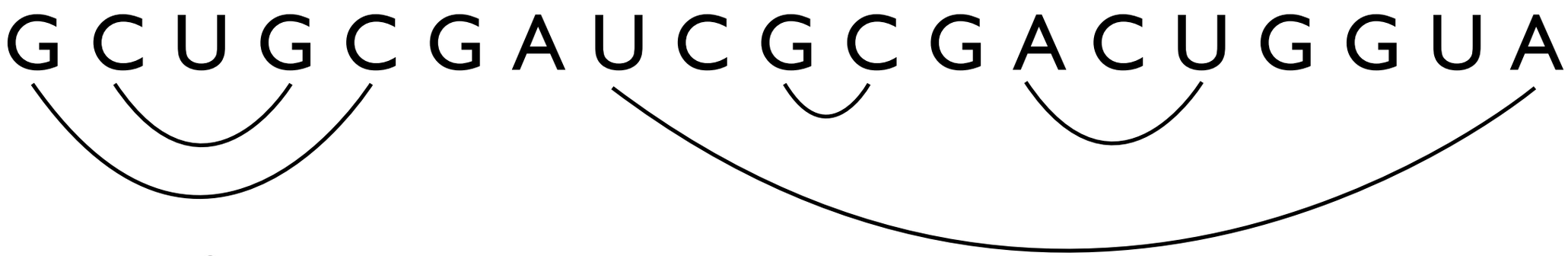
Non-Crossing Inputs
= Tree Edit distance



n^2

$O(n^3 \log n)$ [Klein 1998]

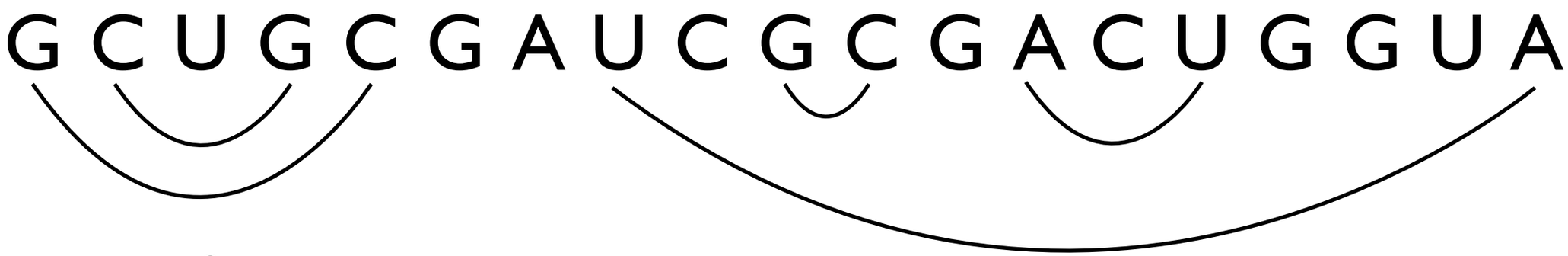
Non-Crossing Inputs
= Tree Edit distance



$n^2 \times n \log n$

$O(n^3 \log n)$ [Klein 1998]

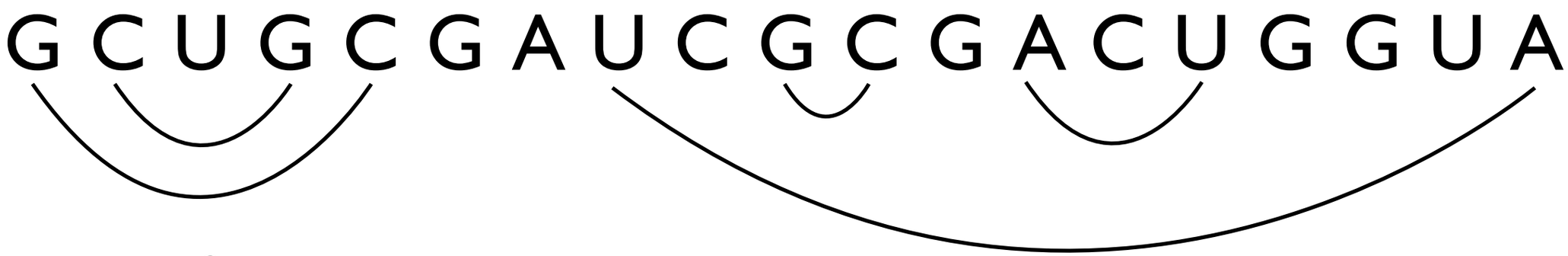
Non-Crossing Inputs
= Tree Edit distance



$n^2 \times n \log n$

$O(n^3 \log n)$ [Klein 1998]

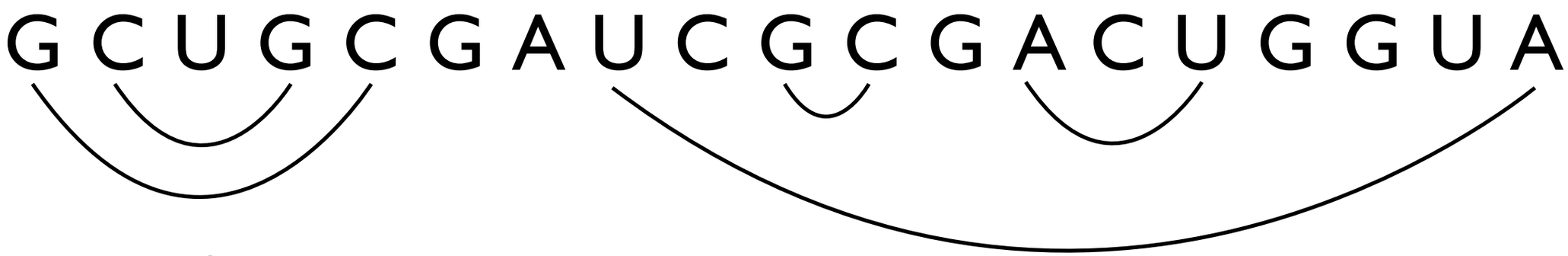
Non-Crossing Inputs
= Tree Edit distance



$n^2 \times n \log n$

$O(n^3 \log n)$ [Klein 1998]

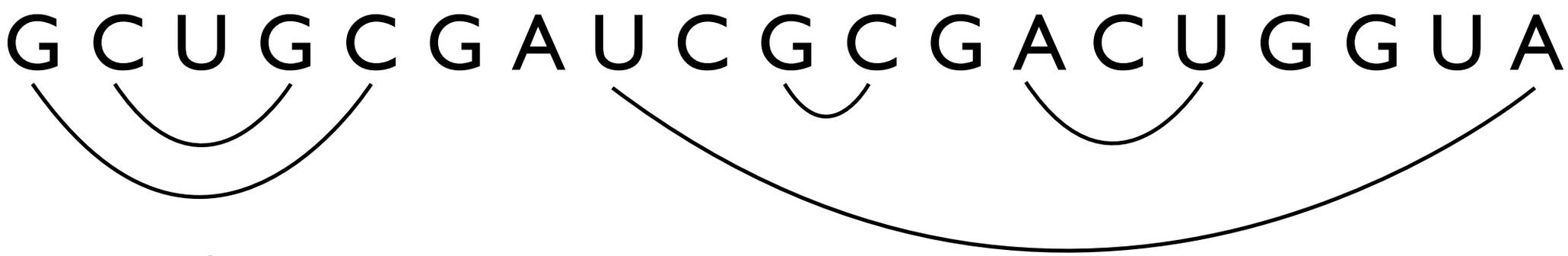
Non-Crossing Inputs
= Tree Edit distance



$n^2 \times n \log n$

$O(n^3 \log n)$ [Klein 1998]

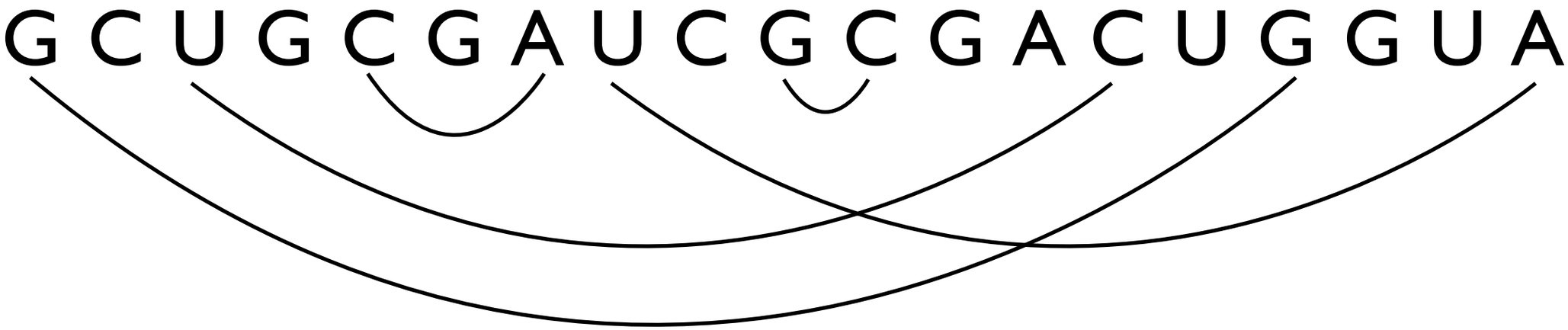
Non-Crossing Inputs
= Tree Edit distance



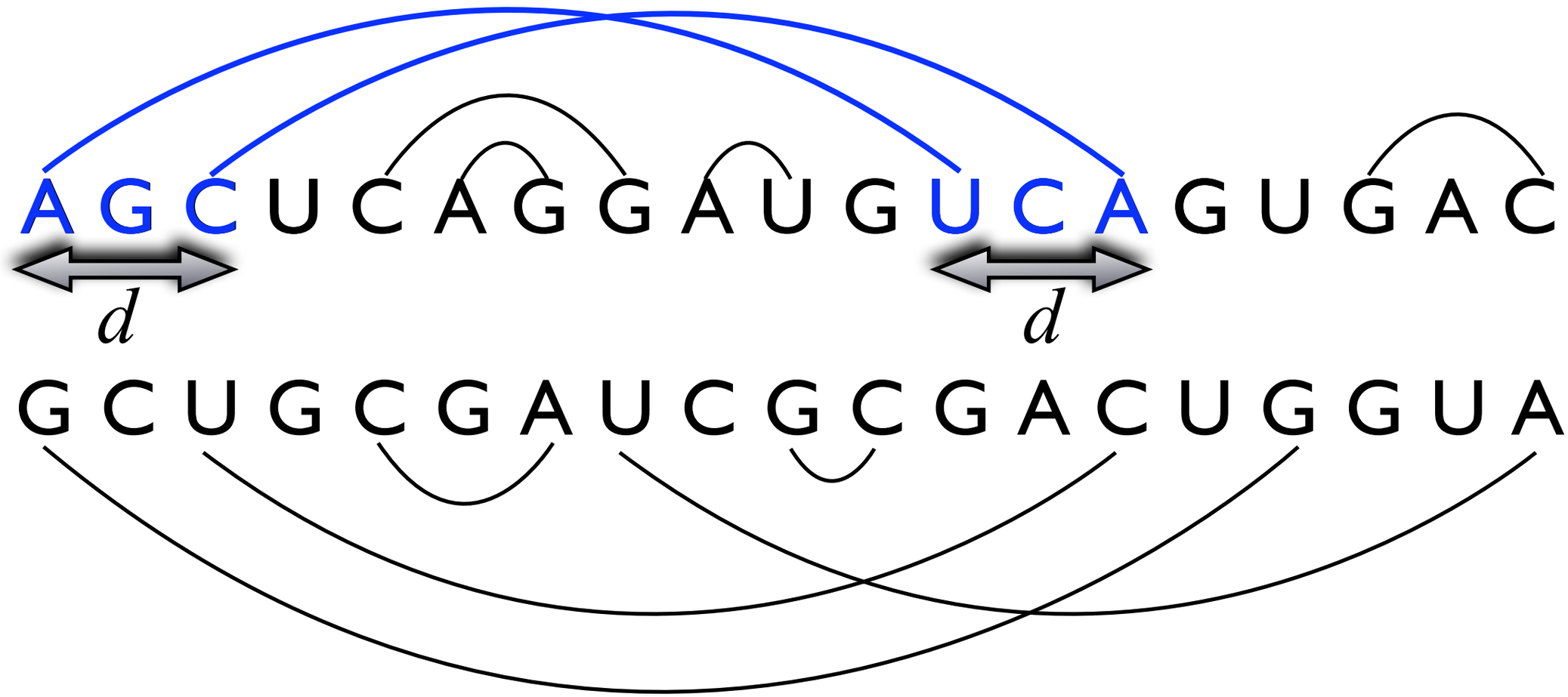
$n^2 \times n \log n$

$O(n^3 \log n)$ [Klein 1998]

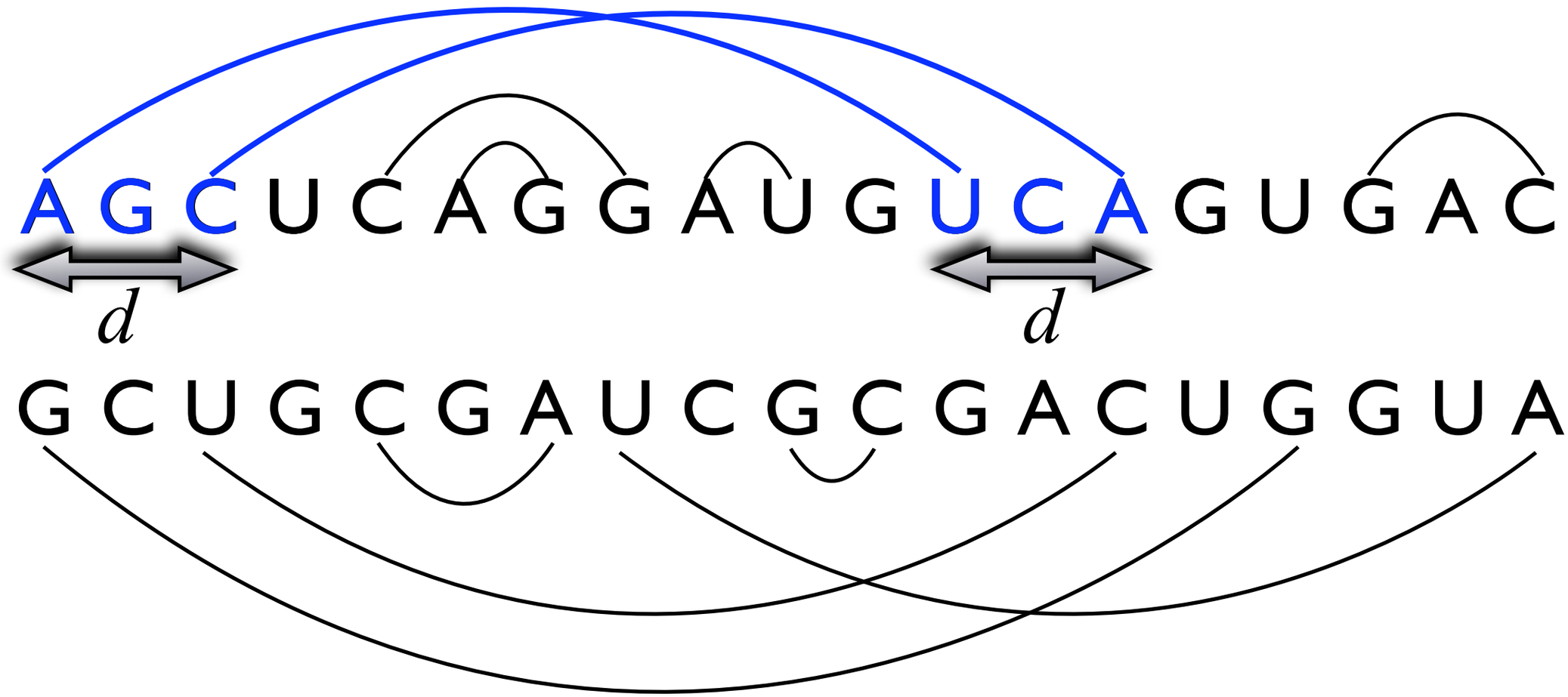
Our Result: d-crossing inputs



Our Result: d-crossing inputs

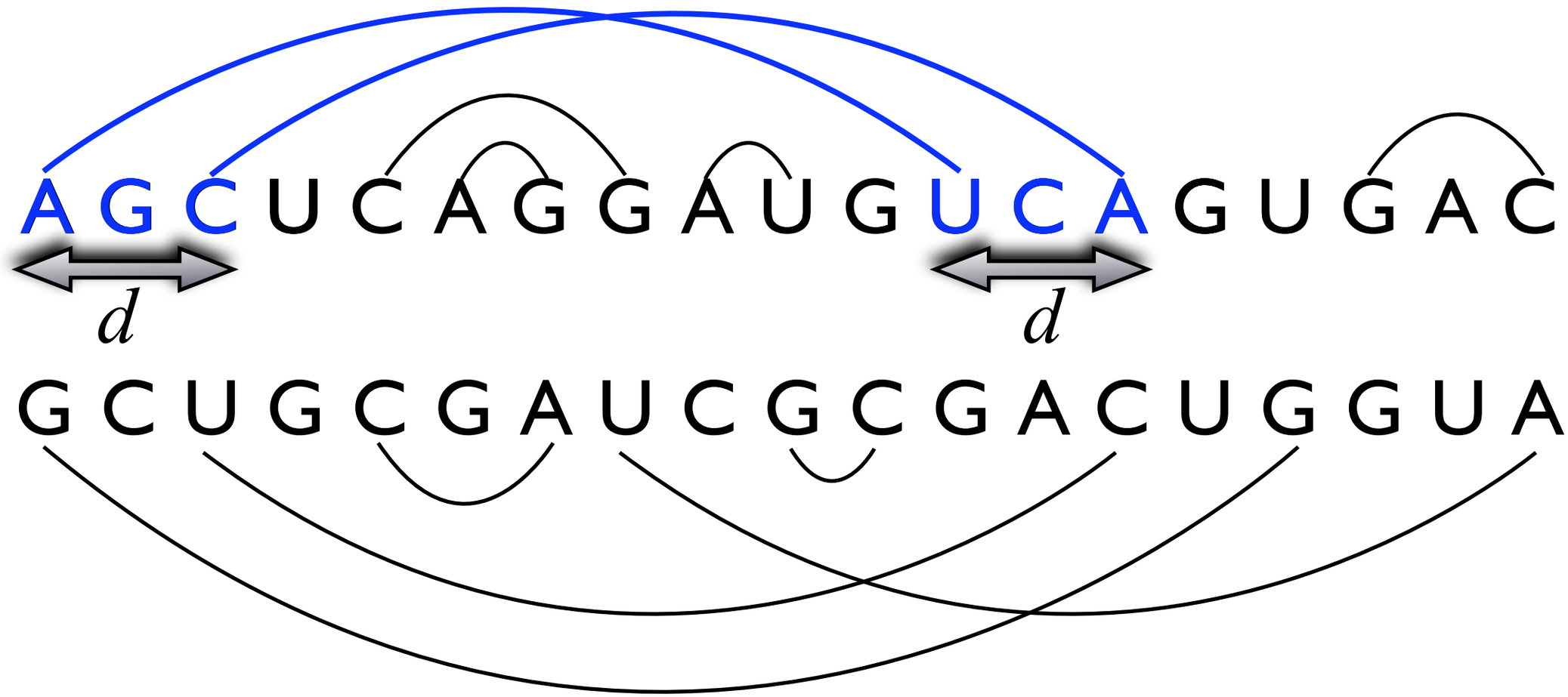


Our Result: d-crossing inputs



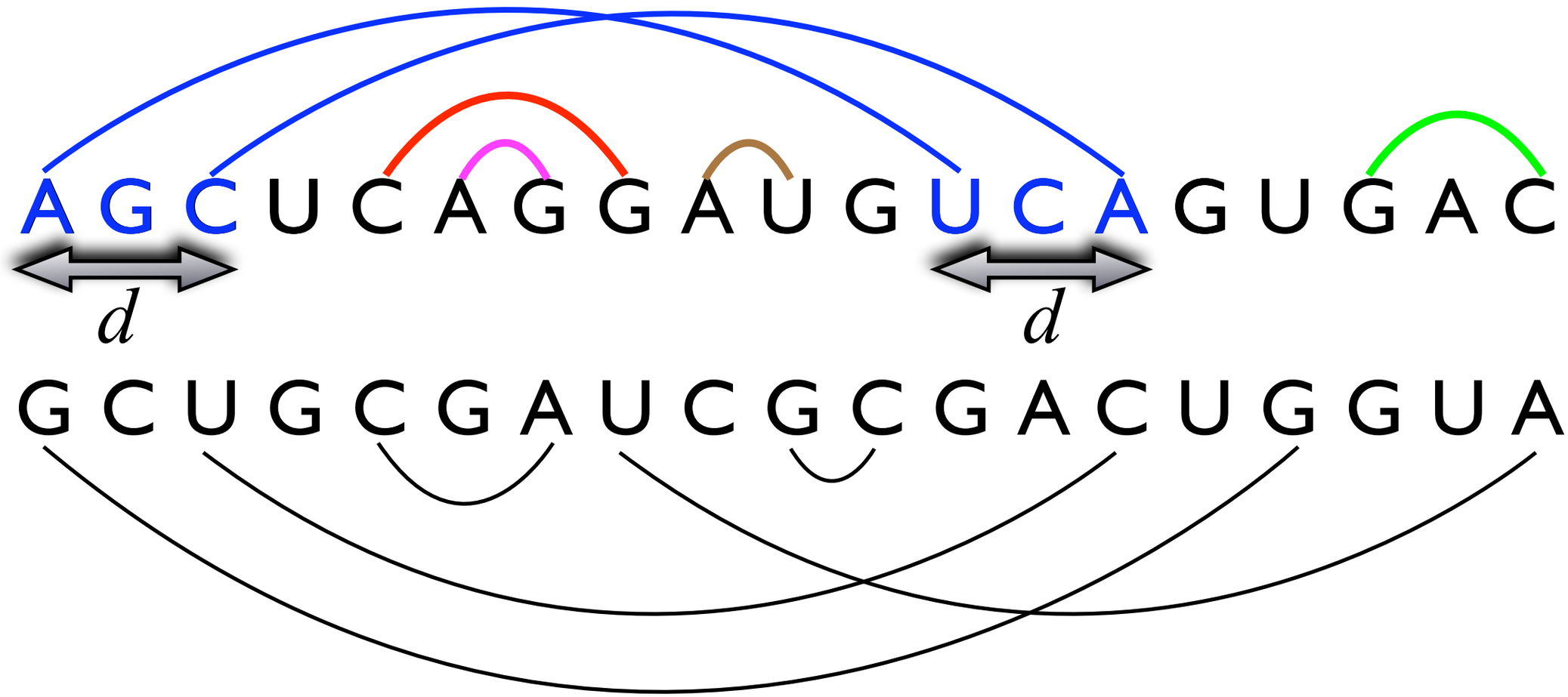
$$O(dn^3 \log n)$$

Our Result: d-crossing inputs



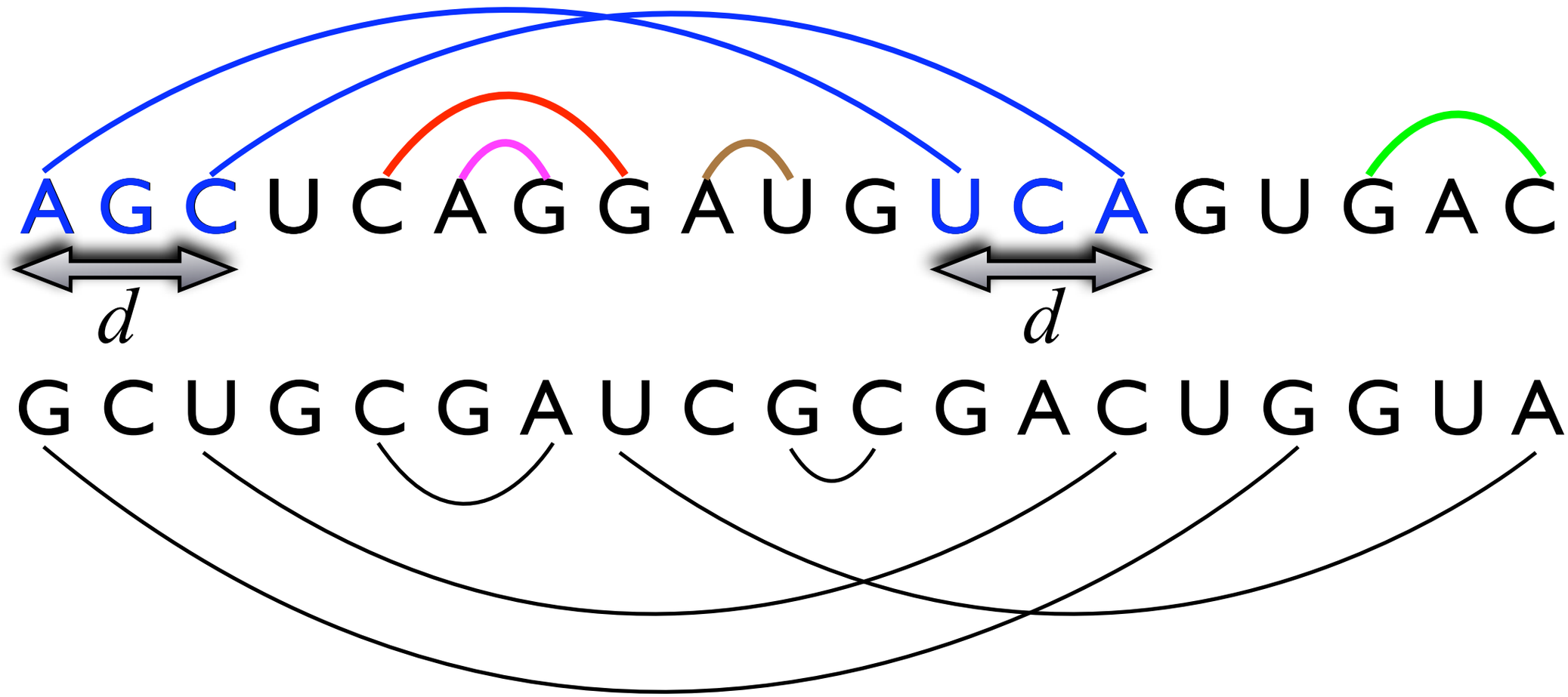
$O(dn^3 \log n)$ Vs. $O(n^4)$

Our Result: d-crossing inputs



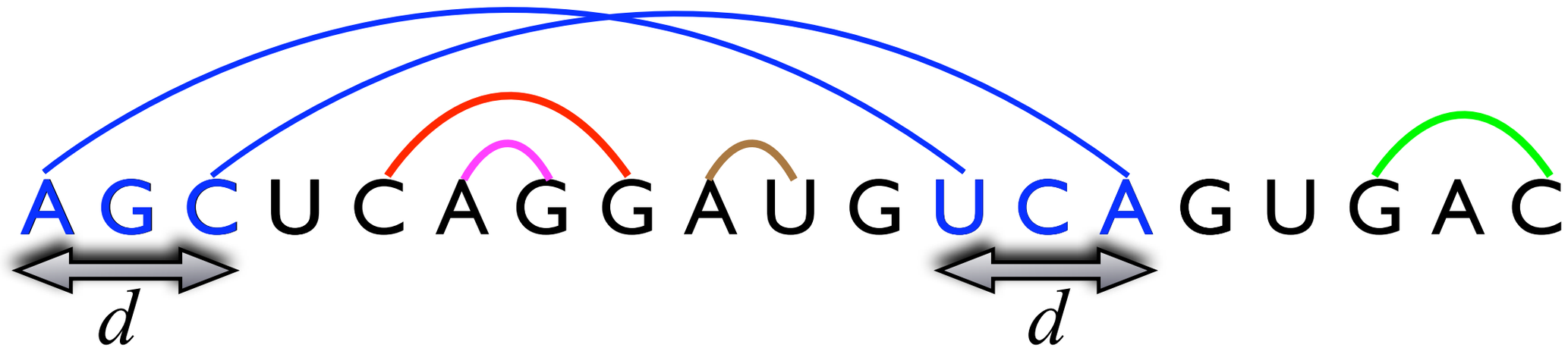
$O(dn^3 \log n)$

Our Result: d-crossing inputs



$$\mathrm{O}(dn^3 \log n) = n^2 \times dn \log n$$

Our Result: d-crossing inputs



$$\mathrm{O}(dn^3 \log n) = n^2 \times dn \log n$$

The Algorithm

The Algorithm

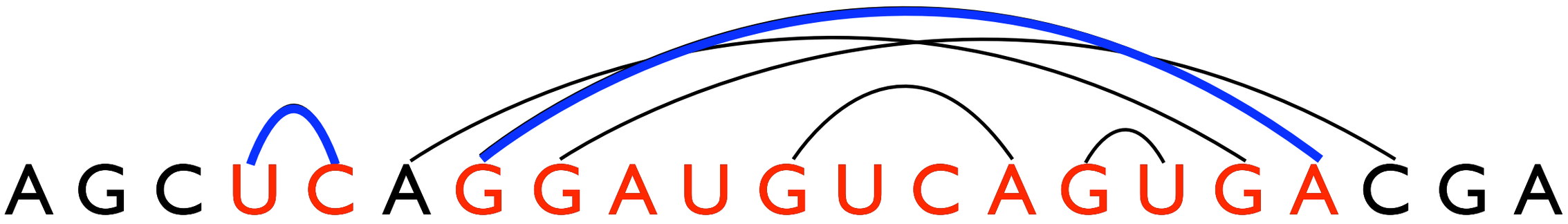
The Algorithm

1. Compute every infix of length $< 2d$ in $O(dn^3)$

The Algorithm

1. Compute every infix of length $< 2d$ in $O(dn^3)$
2. Pick largest arc in every cluster

The Algorithm



1. Compute every infix of length $< 2d$ in $O(dn^3)$
2. Pick largest arc in every cluster
3. Recursively compute infix below these arcs

The Algorithm

1. Compute every infix of length $< 2d$ in $O(dn^3)$
2. Pick largest arc in every cluster
3. Recursively compute infix below these arcs
   + its extension by d to both left and right

The Algorithm

1. Compute every infix of length $< 2d$ in $O(dn^3)$
2. Pick largest arc in every cluster
3. Recursively compute infix below these arcs
   + its extension by d to both left and right
4. Extend largest infix to the left and then to the right

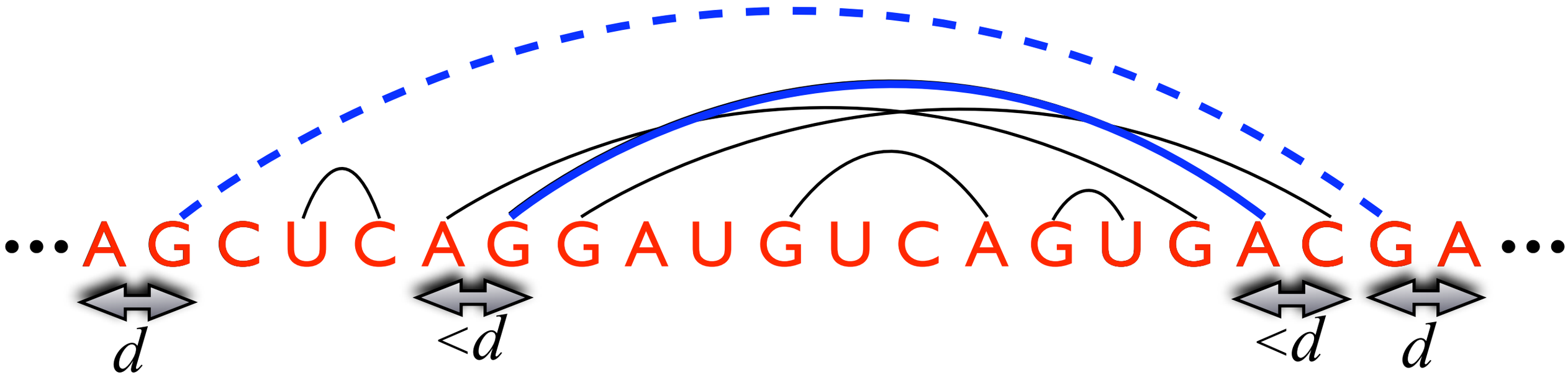
The Algorithm

1. Compute every infix of length $< 2d$ in $O(dn^3)$
2. Pick largest arc in every cluster
3. Recursively compute infix below these arcs
   + its extension by d to both left and right
4. Extend largest infix to the left and then to the right

The Algorithm

1. Compute every infix of length $< 2d$ in $O(dn^3)$
2. Pick largest arc in every cluster
3. Recursively compute infix below these arcs
   + its extension by d to both left and right
4. Extend largest infix to the left and then to the right

The Algorithm

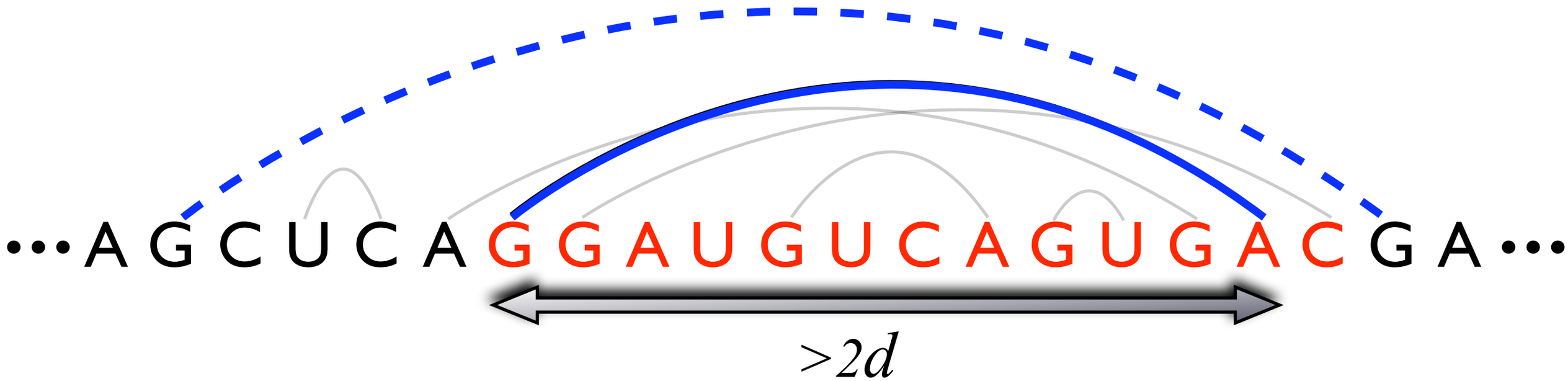


1. Compute every infix of length $< 2d$ in $O(dn^3)$
2. Pick largest arc in every cluster
3. Recursively compute infix below these arcs
   + its extension by d to both left and right
4. Extend largest infix to the left and then to the right

Why does this work?

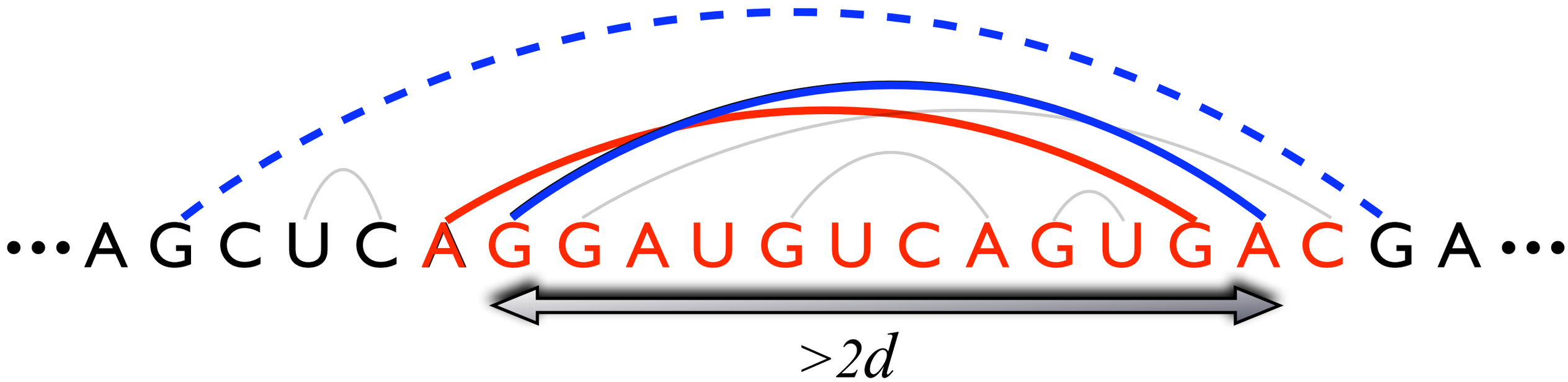
1. Compute every infix of length $< 2d$ in $O(dn^3)$
2. Pick largest arc in every cluster
3. Recursively compute infix below these arcs
   + its extension by d to both left and right
4. Extend largest infix to the left and then to the right

Why does this work?



1. Compute every infix of length $< 2d$ in $O(dn^3)$
2. Pick largest arc in every cluster
3. Recursively compute infix below these arcs
   + its extension by d to both left and right
4. Extend largest infix to the left and then to the right

Why does this work?



1. Compute every infix of length $< 2d$ in $O(dn^3)$
2. Pick largest arc in every cluster
3. Recursively compute infix below these arcs
   + its extension by d to both left and right
4. Extend largest infix to the left and then to the right

Why does this work?

1. Compute every infix of length $< 2d$ in $O(dn^3)$
2. Pick largest arc in every cluster
3. Recursively compute infix below these arcs
   + its extension by d to both left and right
4. Extend largest infix to the left and then to the right

Why does this work?

1. Compute every infix of length $< 2d$ in $O(dn^3)$
2. Pick largest arc in every cluster
3. Recursively compute infix below these arcs
   + its extension by d to both left and right
4. Extend largest infix to the left and then to the right

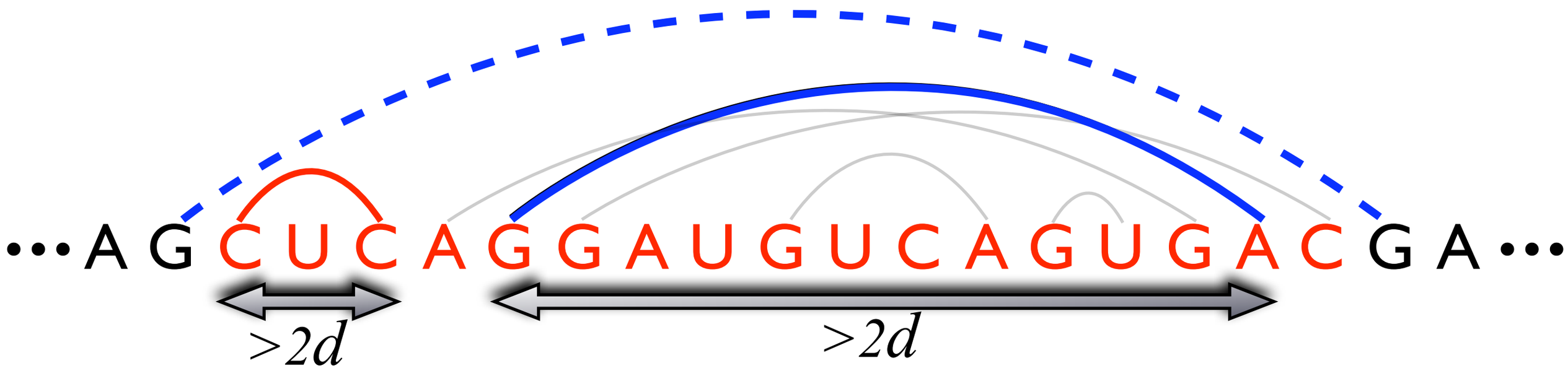
Why does this work?

1. Compute every infix of length $< 2d$ in $O(dn^3)$
2. Pick largest arc in every cluster
3. Recursively compute infix below these arcs
   + its extension by d to both left and right
4. Extend largest infix to the left and then to the right

Why does this work?

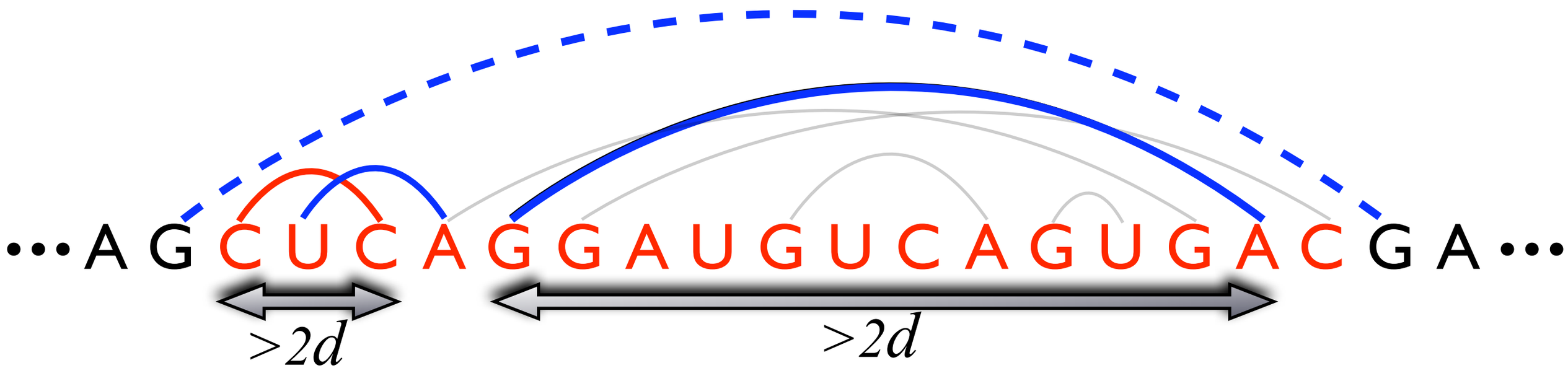
1. Compute every infix of length $< 2d$ in $O(dn^3)$
2. Pick largest arc in every cluster
3. Recursively compute infix below these arcs
   + its extension by d to both left and right
4. Extend largest infix to the left and then to the right

Why does this work?



1. Compute every infix of length $< 2d$ in $O(dn^3)$
2. Pick largest arc in every cluster
3. Recursively compute infix below these arcs
   + its extension by d to both left and right
4. Extend largest infix to the left and then to the right

Why does this work?

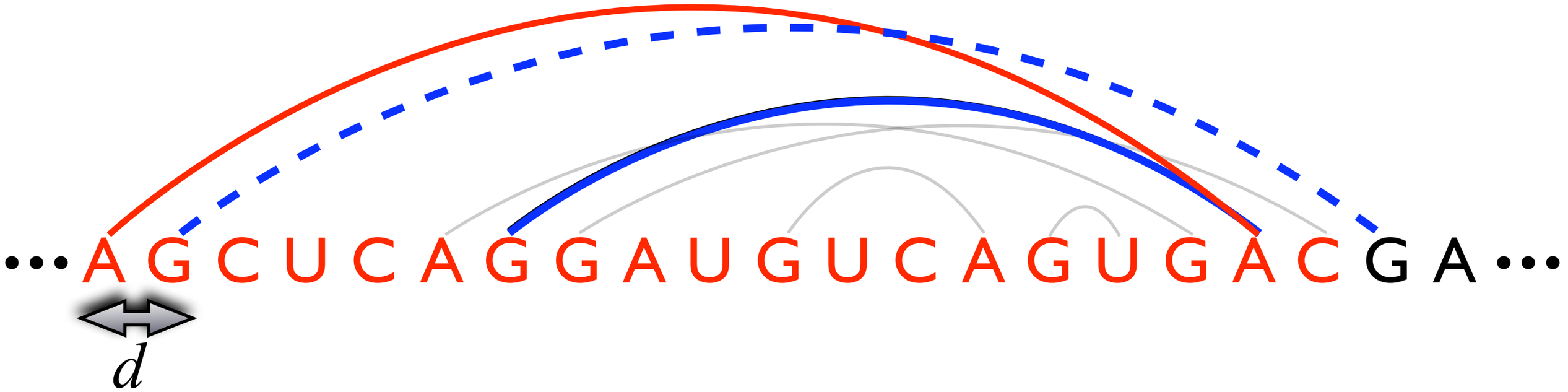


1. Compute every infix of length $< 2d$ in $O(dn^3)$
2. Pick largest arc in every cluster
3. Recursively compute infix below these arcs
   + its extension by d to both left and right
4. Extend largest infix to the left and then to the right

Why does this work?

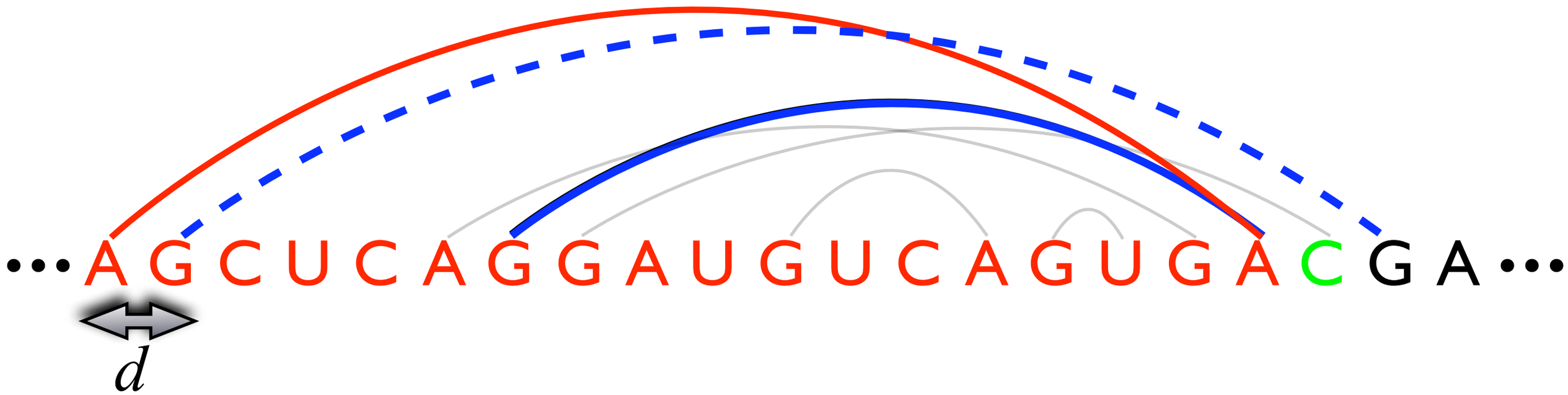
1. Compute every infix of length $< 2d$ in $O(dn^3)$
2. Pick largest arc in every cluster
3. Recursively compute infix below these arcs
   + its extension by d to both left and right
4. Extend largest infix to the left and then to the right

Why does this work?



1. Compute every infix of length $< 2d$ in $O(dn^3)$
2. Pick largest arc in every cluster
3. Recursively compute infix below these arcs
   + its extension by d to both left and right
4. Extend largest infix to the left and then to the right

Why does this work?



1. Compute every infix of length $< 2d$ in $O(dn^3)$
2. Pick largest arc in every cluster
3. Recursively compute infix below these arcs
   + its extension by d to both left and right
4. Extend largest infix to the left and then to the right

Thank You!

and happy birthday CPM