RNA folding & kinetics

Recap: energy terms

- Complementary base-pairing
- Base-pair stacking
- Entropic cost of loop closure
- Sequence-dependent loops and bulges

A "quick" exercise

- Consider the <u>**strict foldback**</u> structure with the most <u>**perfect Watson-Crick**</u> basepairs for this sequence:

A “quick” exercise

- Consider the **<u>strict foldback</u>** structure with the most **<u>perfect Watson-Crick</u>** basepairs for this sequence:

```
ACCACCUGCACAGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGUGUGCAGG
```

A “quick” exercise

- Consider the **strict foldback** structure with the most **perfect Watson-Crick** basepairs for this sequence:

```
ACCACCUGCACAGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGUGUGCAGG
```

- How many (perfect WC) basepairs does it have?
- Can you quickly find an upper or lower bound?
- What *is* the structure?

A “quick” exercise

- Consider the **strict foldback** structure with the most **perfect Watson-Crick** basepairs for this sequence:

```
ACCACCUGCACAGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGUGUGCAGG
```

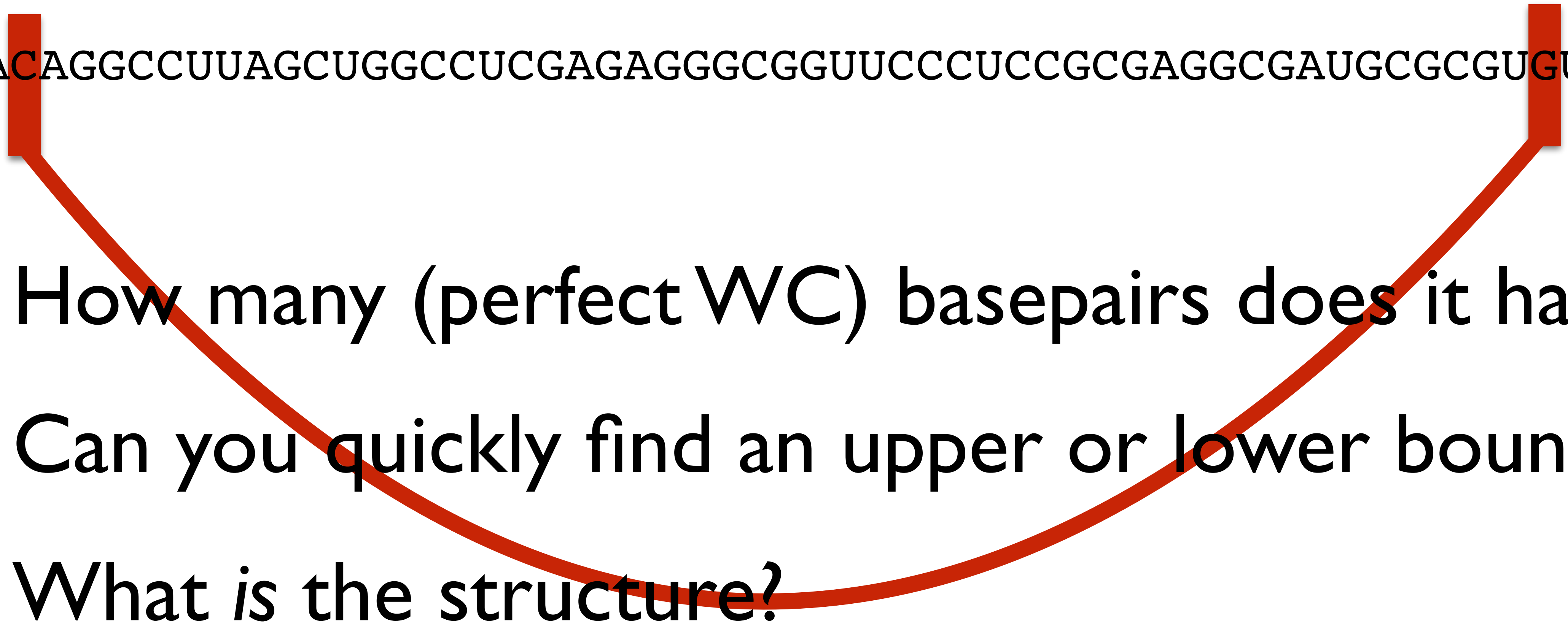
- How many (perfect WC) basepairs does it have?
- Can you quickly find an upper or lower bound?
- What *is* the structure?

HINTS…

A “quick” exercise

- Consider the **<u>strict foldback</u>** structure with the most **<u>perfect Watson-Crick</u>** basepairs for this sequence:

ACCACCUGCACAGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGUGUGCAGG



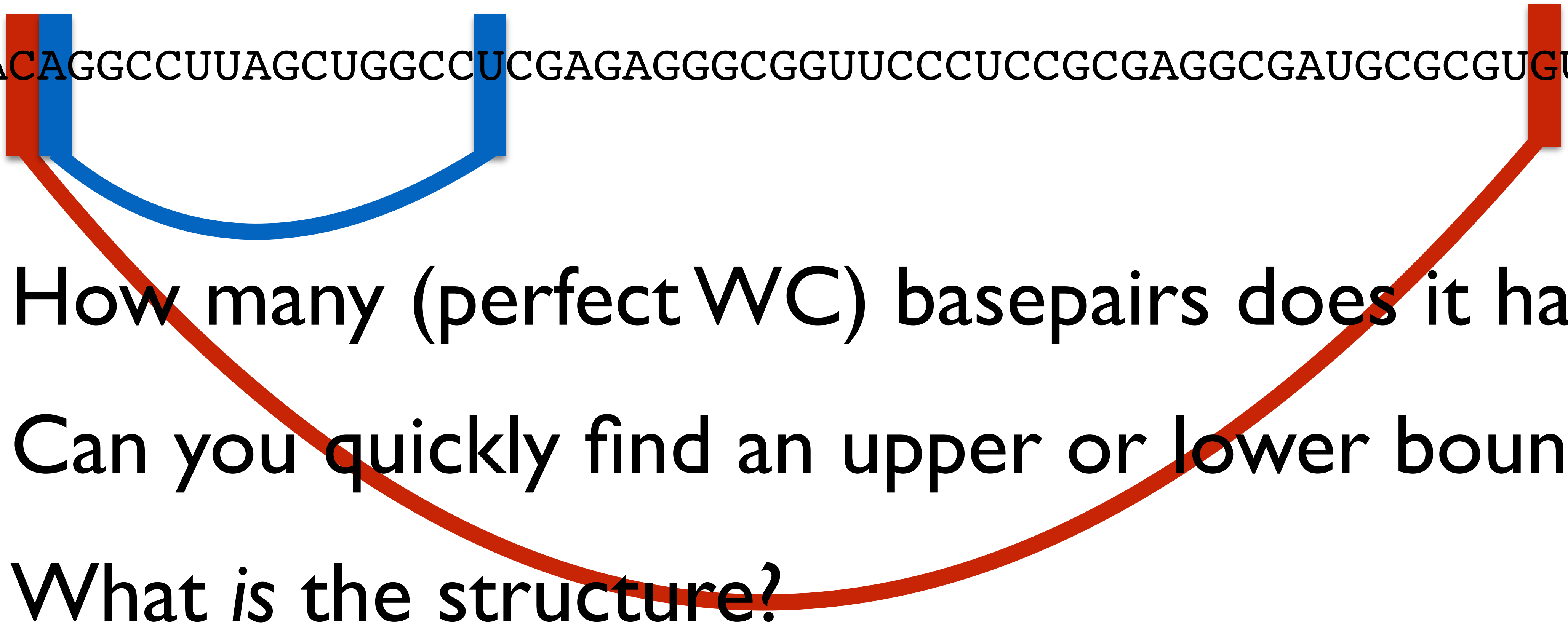
- How many (perfect WC) basepairs does it have?
- Can you quickly find an upper or lower bound?
- What *is* the structure?

HINTS…

A “quick” exercise

- Consider the **strict foldback** structure with the most **perfect Watson-Crick** basepairs for this sequence:

ACCACCUGCACAGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGUGUGCAGG



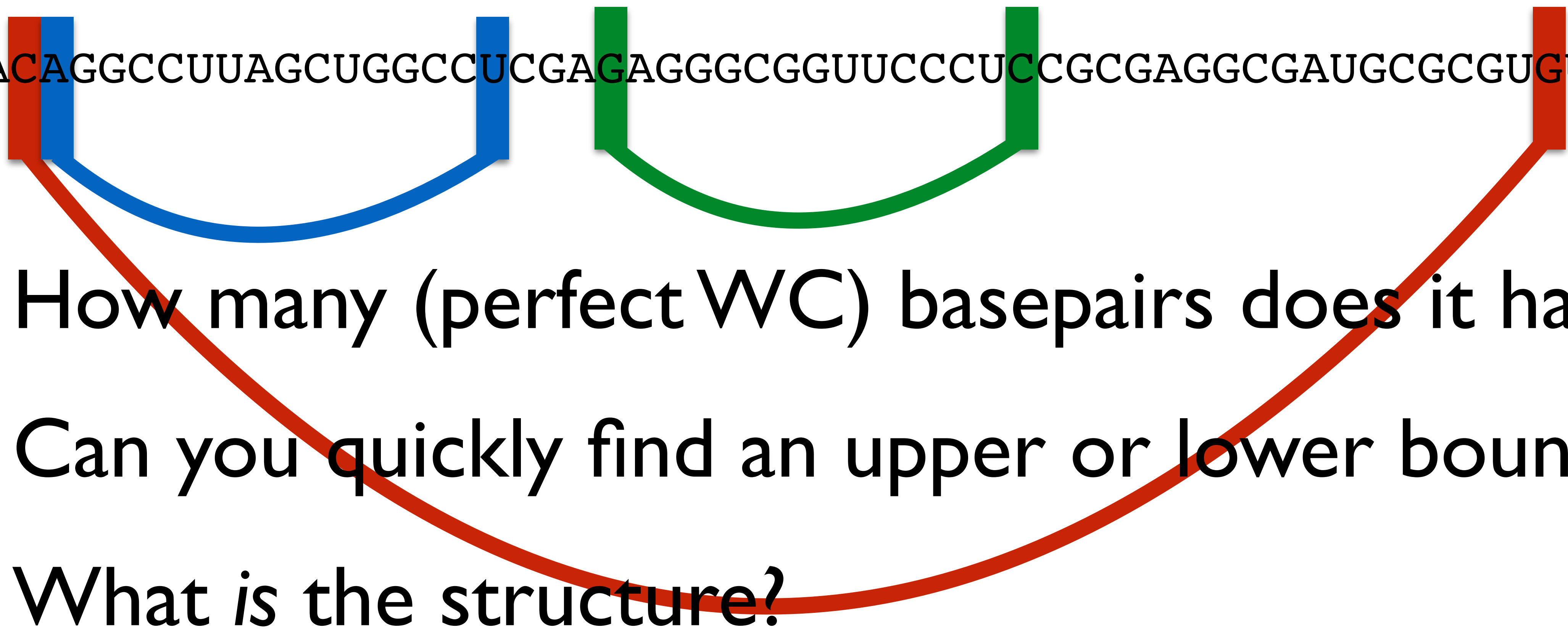
- How many (perfect WC) basepairs does it have?
- Can you quickly find an upper or lower bound?
- What *is* the structure?

HINTS…

A "quick" exercise

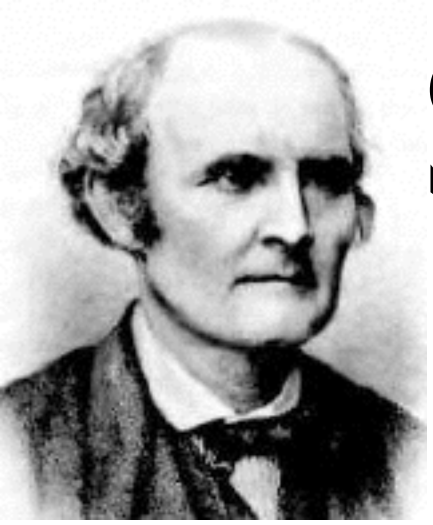
- Consider the **strict foldback** structure with the most **perfect Watson-Crick** basepairs for this sequence:

```
ACCACCUGCACAGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGUGUGCAGG
```



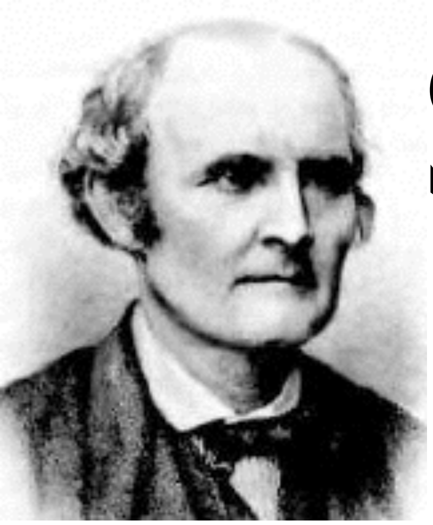
- How many (perfect WC) basepairs does it have?
- Can you quickly find an upper or lower bound?
- What *is* the structure?

HINTS…



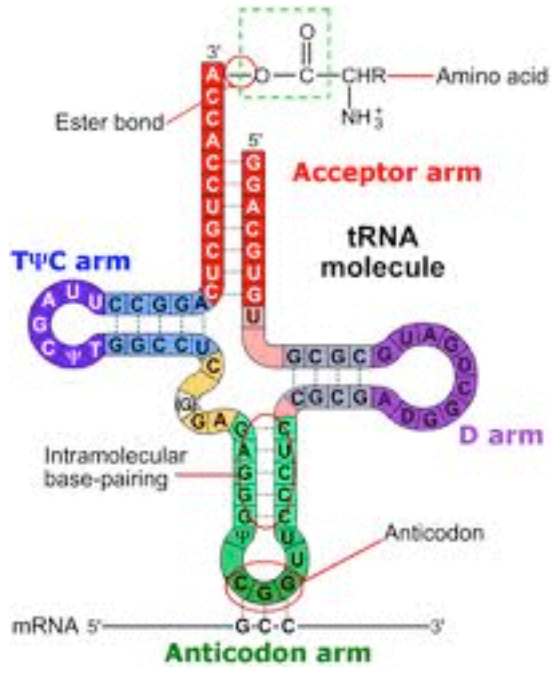
Secondary structure

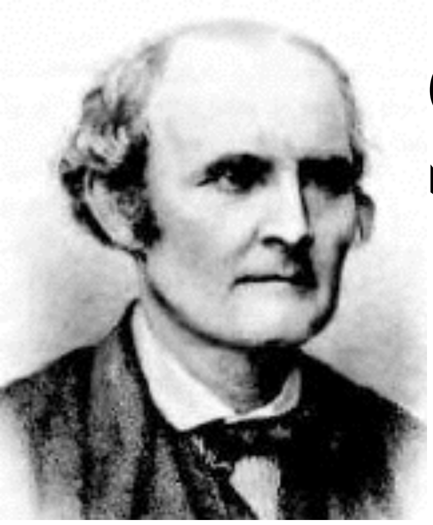
- Arthur Cayley: correspondence between trees & strings of balanced parentheses
- Implicit nested/hierarchical structure
- Noam Chomsky: *“Parse trees”*, *“Grammars”*



Secondary structure

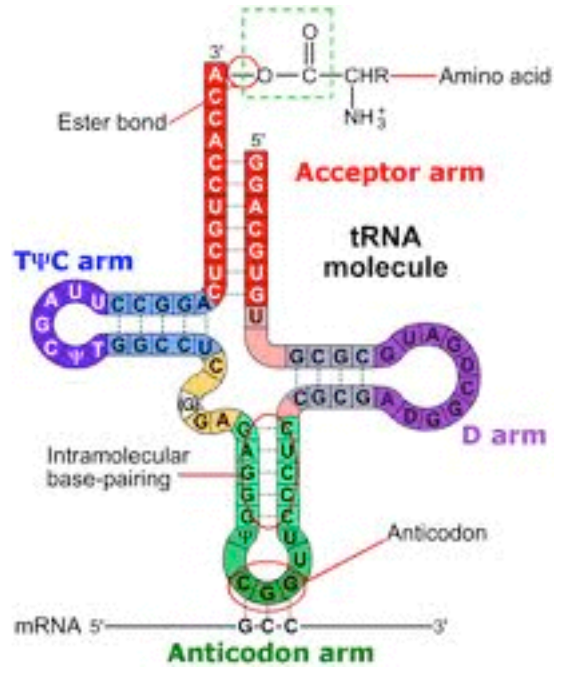
- Arthur Cayley: correspondence between trees & strings of balanced parentheses
- Implicit nested/hierarchical structure
- Noam Chomsky: *"Parse trees"*, *"Grammars"*



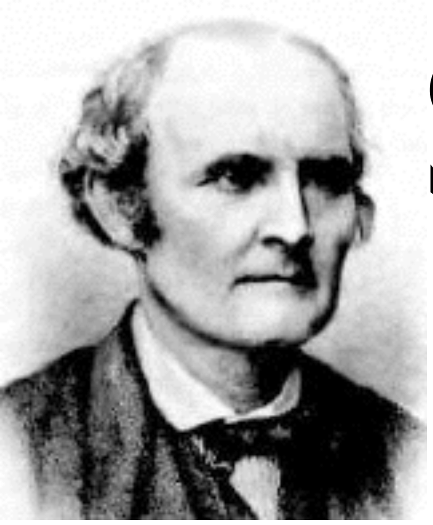


Secondary structure

- Arthur Cayley: correspondence between trees & strings of balanced parentheses
- Implicit nested/hierarchical structure
- Noam Chomsky: *"Parse trees"*, *"Grammars"*

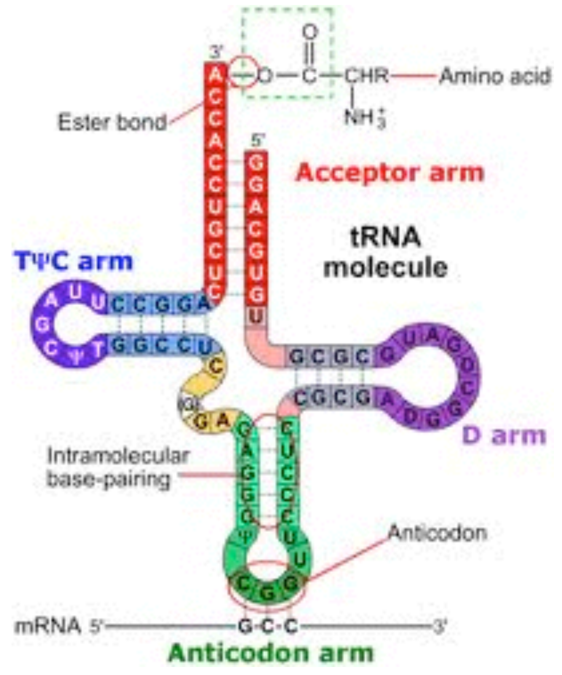


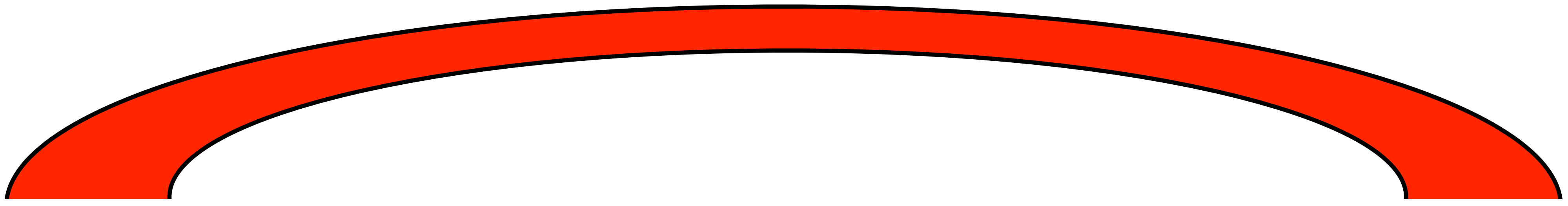
ACCACCUGCUCAGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGUGUGCAGG



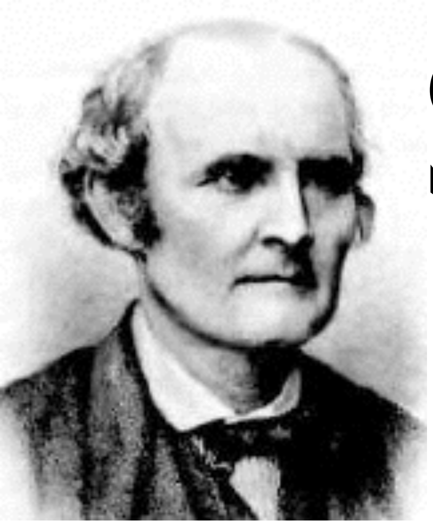
Secondary structure

- Arthur Cayley: correspondence between trees & strings of balanced parentheses
- Implicit nested/hierarchical structure
- Noam Chomsky: *“Parse trees”*, *“Grammars”*



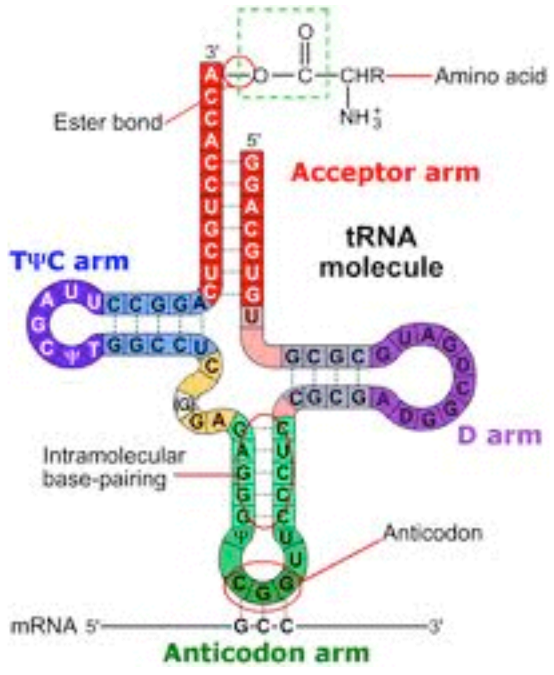


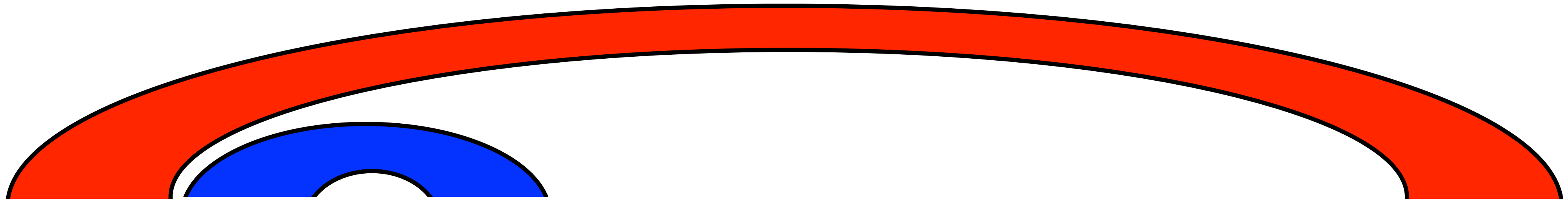
ACCACCUGCUCAGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGUGUGCAGG



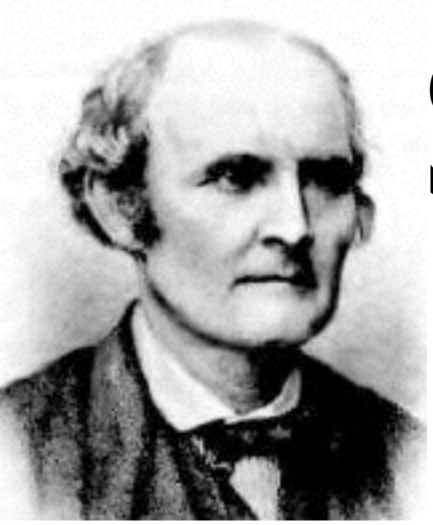
Secondary structure

- Arthur Cayley: correspondence between trees & strings of balanced parentheses
- Implicit nested/hierarchical structure
- Noam Chomsky: *“Parse trees”*, *“Grammars”*



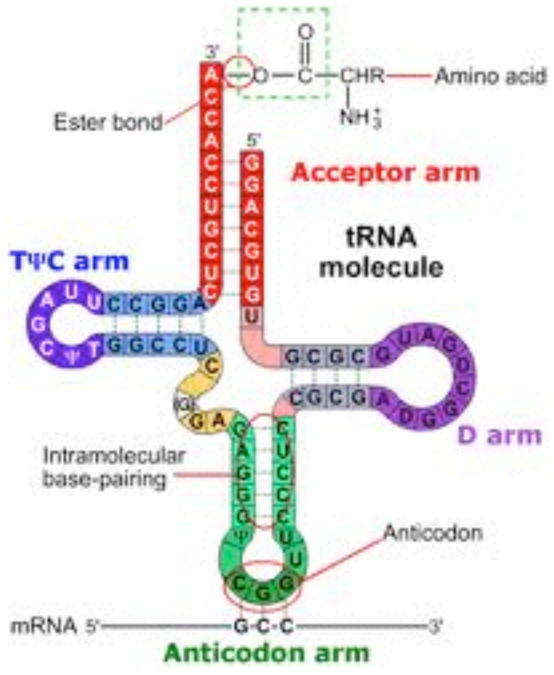


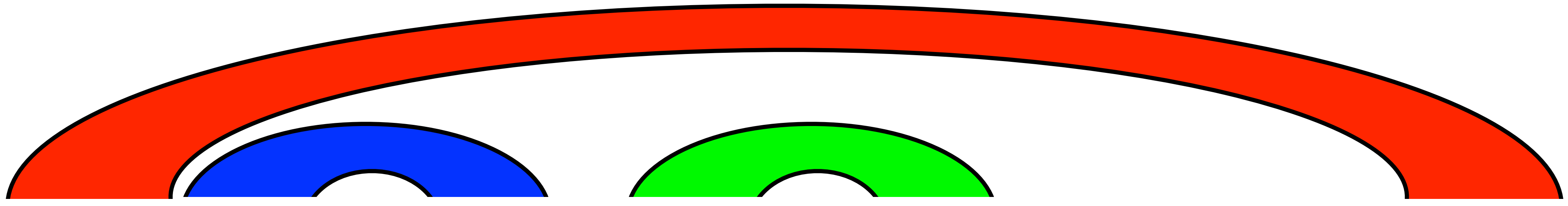
ACCACCUGCUCAGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGUGUGCAGG



Secondary structure

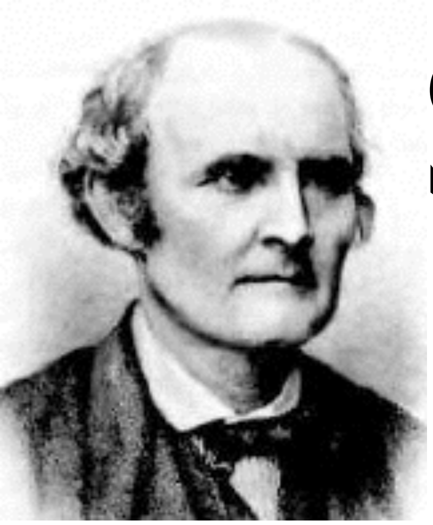
- Arthur Cayley: correspondence between trees & strings of balanced parentheses
- Implicit nested/hierarchical structure
- Noam Chomsky: *"Parse trees"*, *"Grammars"*



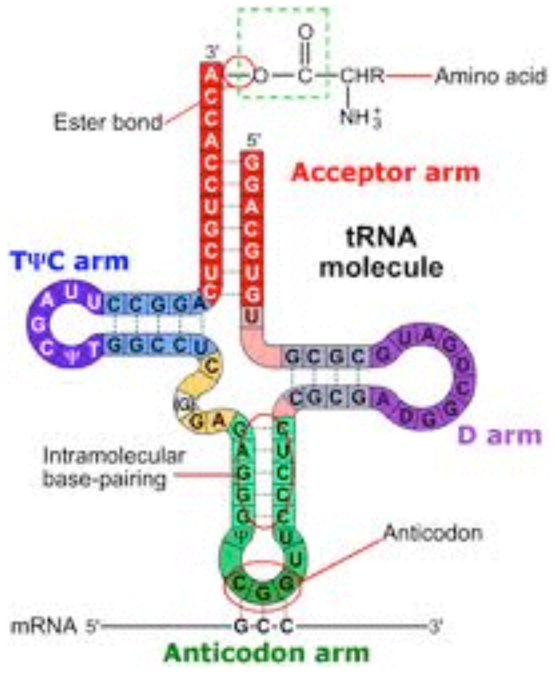


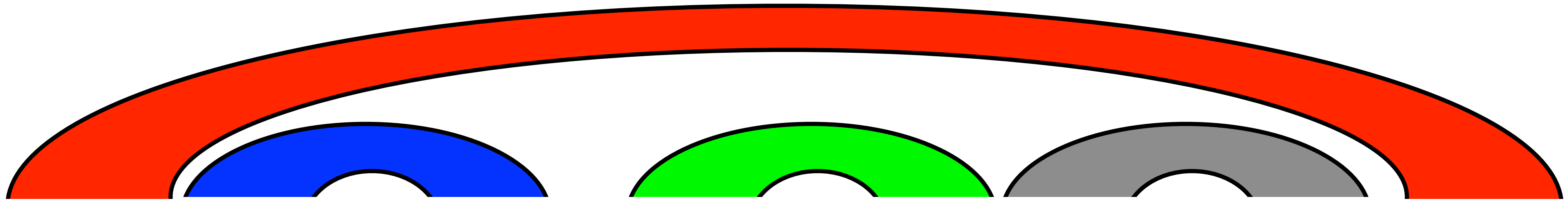
ACCACCUGCUCAGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGUGUGCAGG

Secondary structure

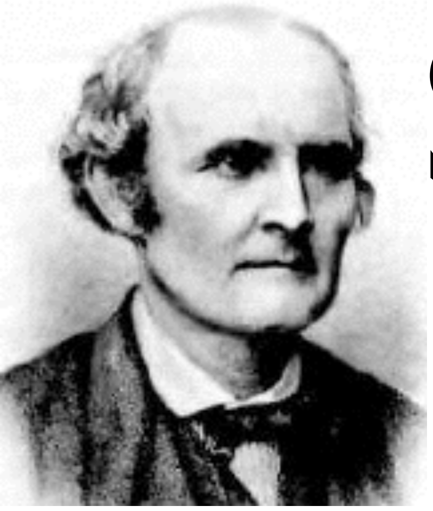


- Arthur Cayley: correspondence between trees & strings of balanced parentheses
- Implicit nested/hierarchical structure
- Noam Chomsky: *"Parse trees"*, *"Grammars"*



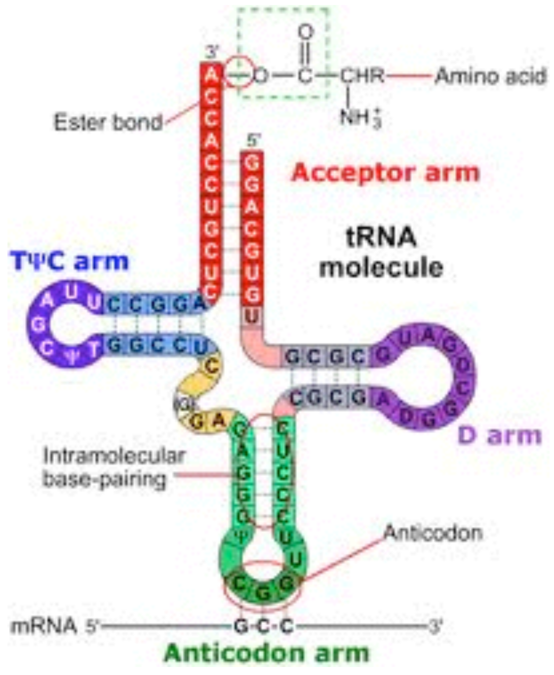


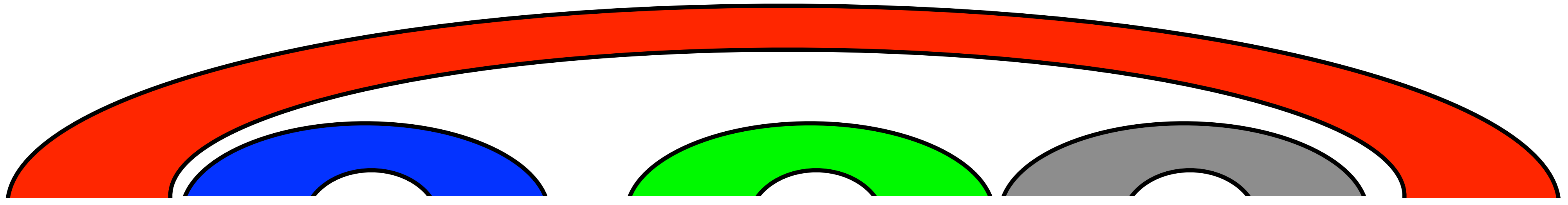
ACCACCUGCUCAGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGUGUGCAGG



Secondary structure

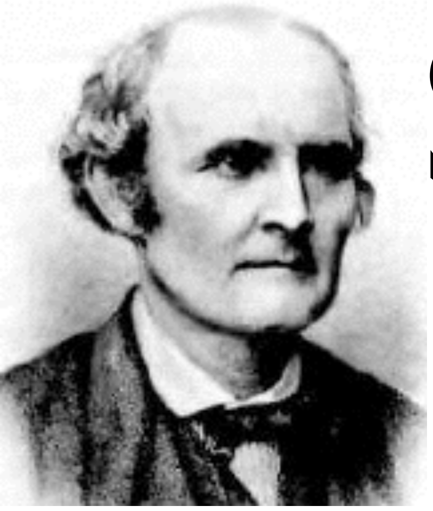
- Arthur Cayley: correspondence between trees & strings of balanced parentheses
- Implicit nested/hierarchical structure
- Noam Chomsky: *"Parse trees"*, *"Grammars"*



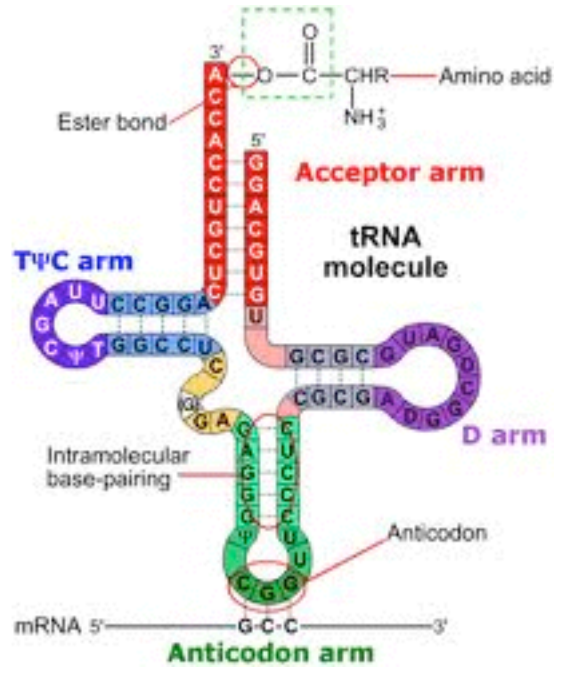


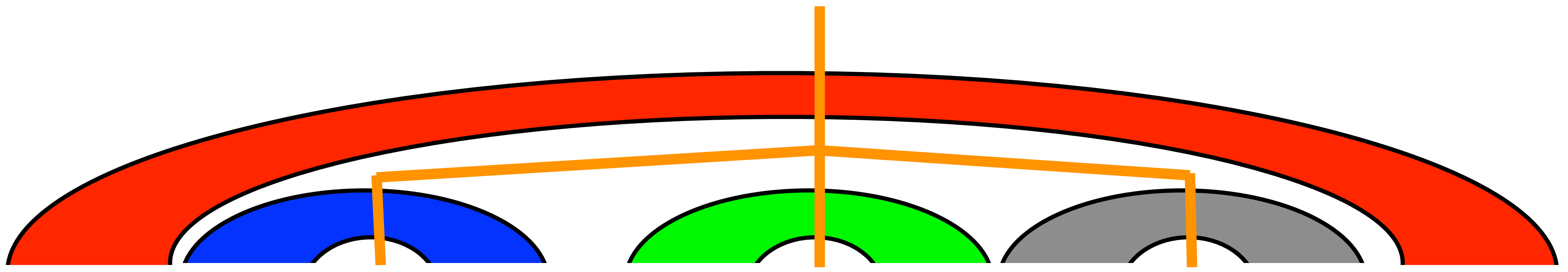
```
ACCACCUGCUCAGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGUGUGCAGG
....<<<<<<<<<<<<......>>>>>...<<<<<.....>>>>><<<<........>>>>.>>>>>>>
```

Secondary structure

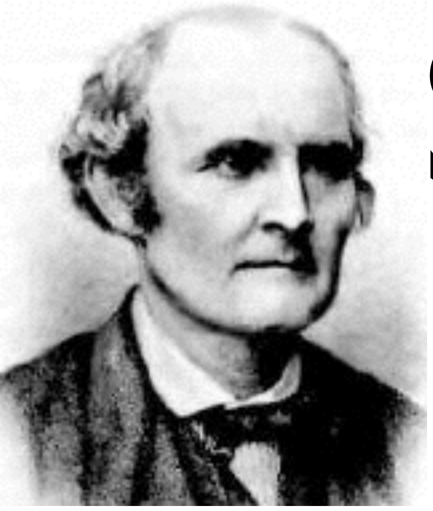


- Arthur Cayley: correspondence between trees & strings of balanced parentheses
- Implicit nested/hierarchical structure
- Noam Chomsky: *"Parse trees"*, *"Grammars"*



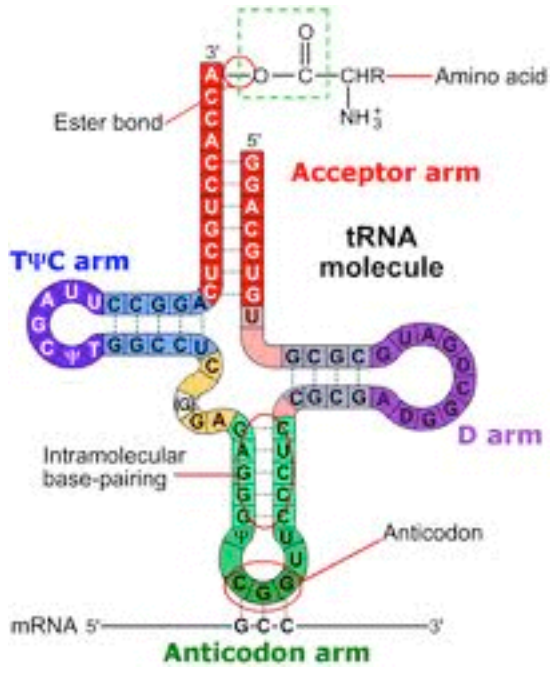


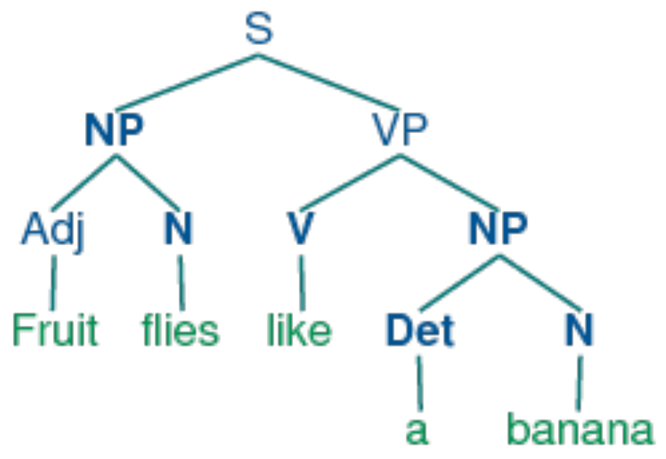
```
ACCACCUGCUCAGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGUGUGCAGG
....<<<<<<<<<<<<......>>>>>...<<<<<.....>>>>><<<<........>>>>.>>>>>>>
```

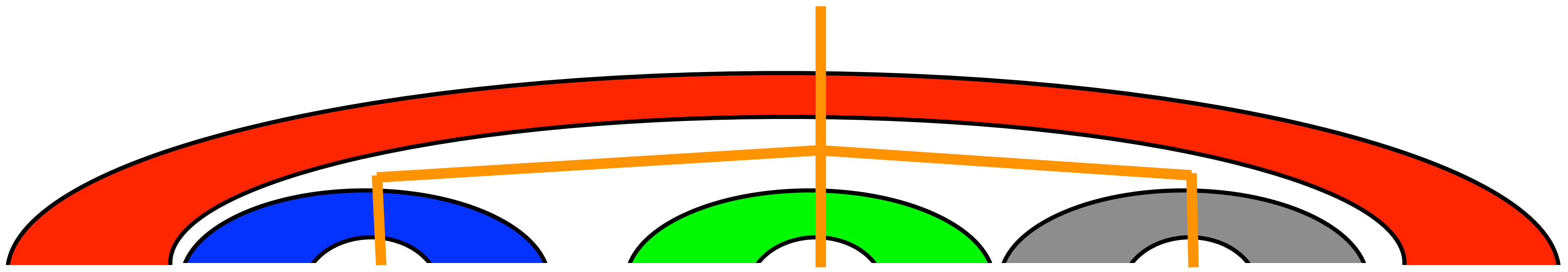


Secondary structure

- Arthur Cayley: correspondence between trees & strings of balanced parentheses
- Implicit nested/hierarchical structure
- Noam Chomsky: *"Parse trees"*, *"Grammars"*

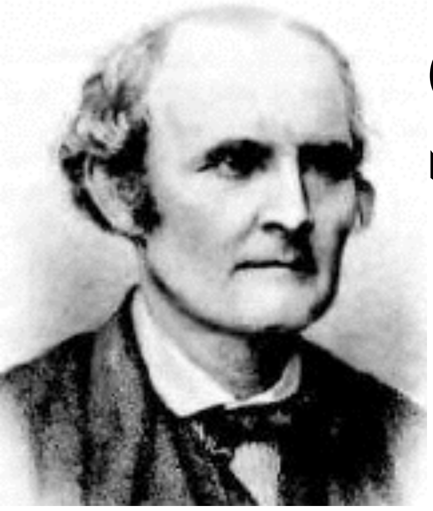




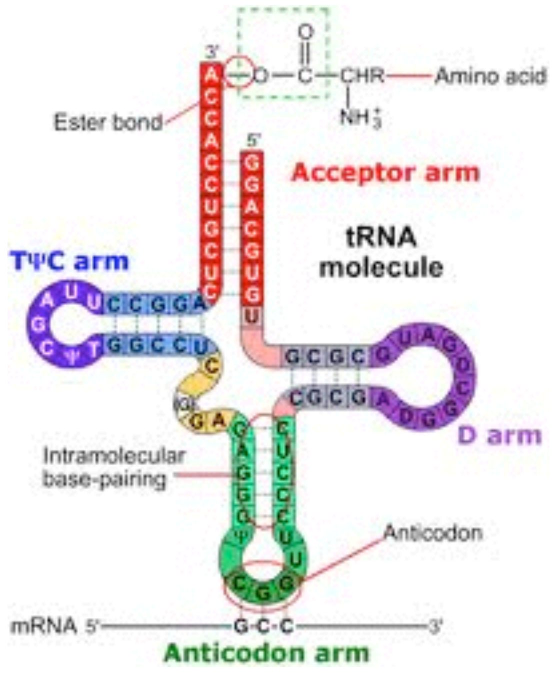


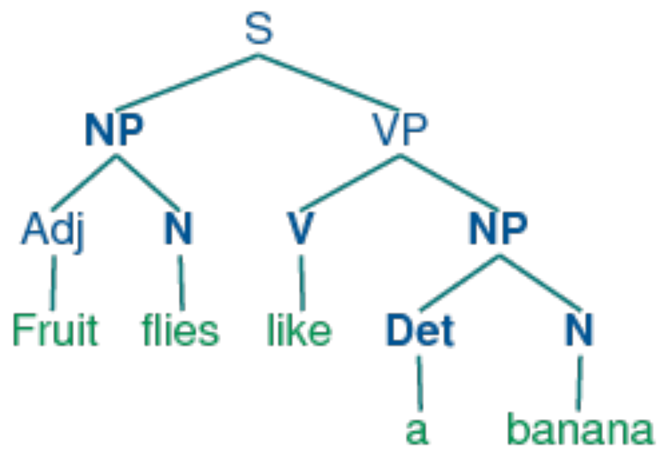
```
ACCACCUGCUCAGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGUGUGCAGG
....<<<<<<<<<<<<......>>>>>...<<<<<.....>>>>><<<<........>>>>.>>>>>>>
```

Secondary structure



- Arthur Cayley: correspondence between trees & strings of balanced parentheses
- Implicit nested/hierarchical structure
- Noam Chomsky: *"Parse trees"*, *"Grammars"*





S=sentence

NP=noun phrase

RC=relative clause

VP=verb phrase

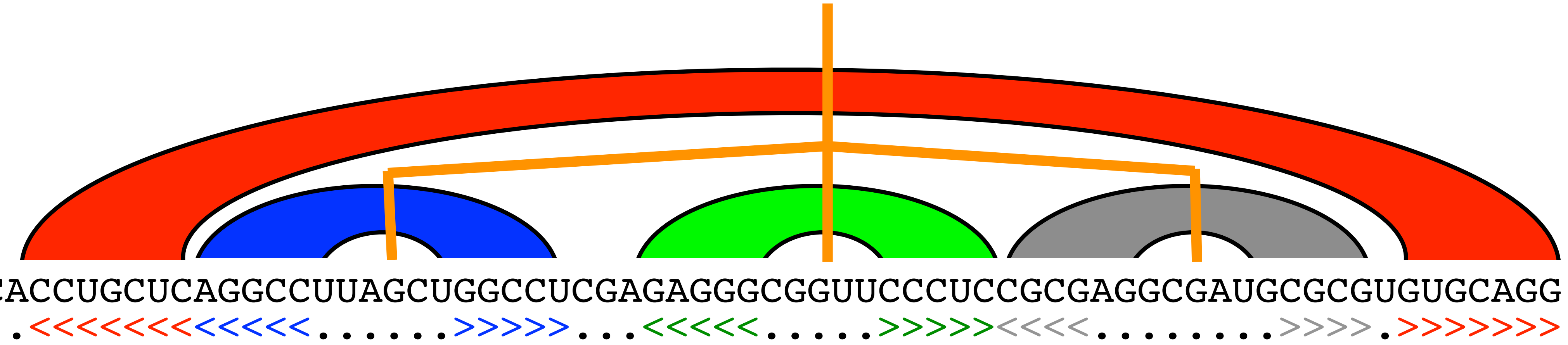
PN=proper noun

V=verb

Adj=Adjective

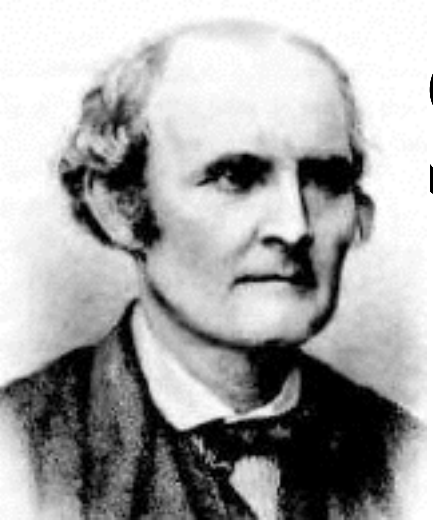
Det=Determiner

Prep=Preposition

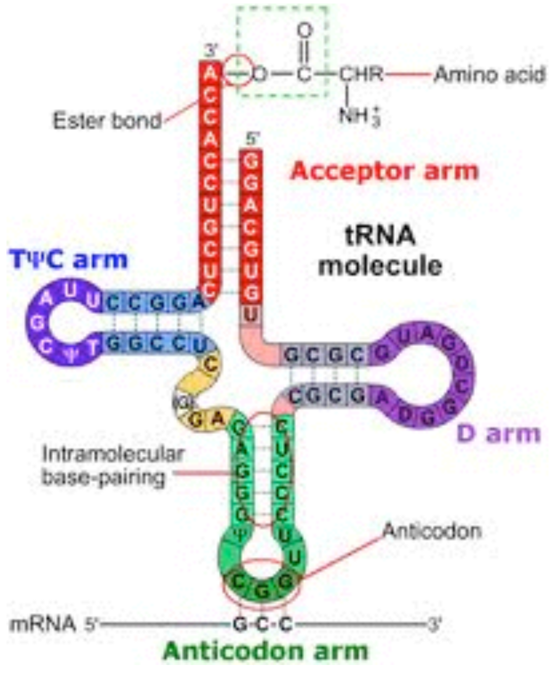


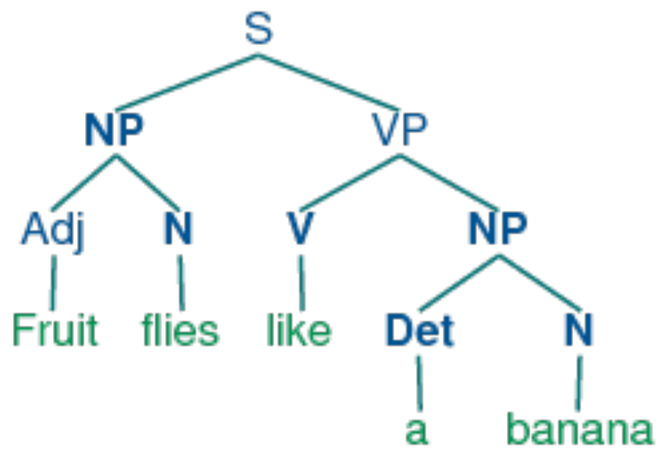
```
ACCACCUGCUCAGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGUGUGCAGG
....<<<<<<<<<<<<......>>>>>...<<<<<.....>>>>><<<<........>>>>.>>>>>>>
```

Secondary structure



- Arthur Cayley: correspondence between trees & strings of balanced parentheses
- Implicit nested/hierarchical structure
- Noam Chomsky: *"Parse trees"*, *"Grammars"*





S=sentence

NP=noun phrase

RC=relative clause

VP=verb phrase

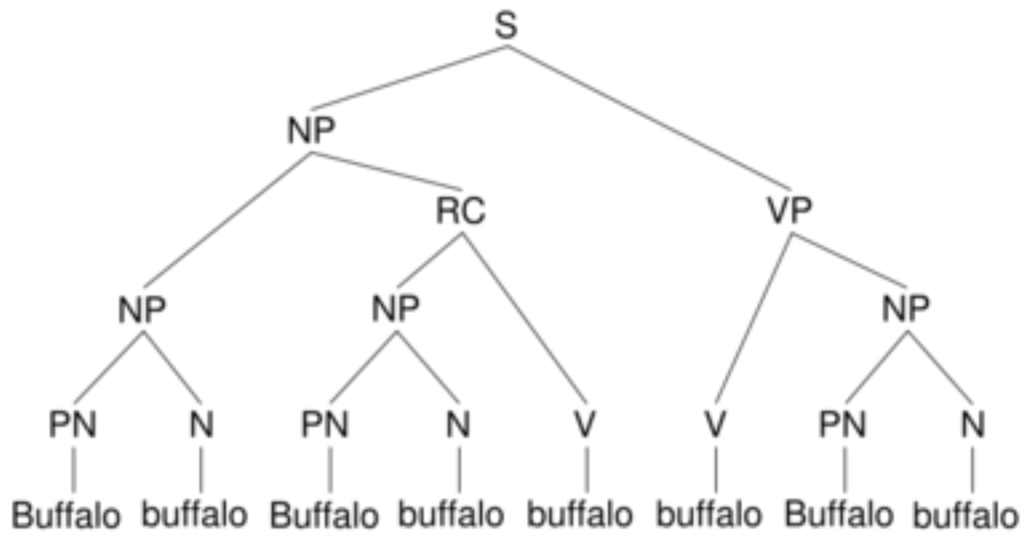
PN=proper noun

V=verb

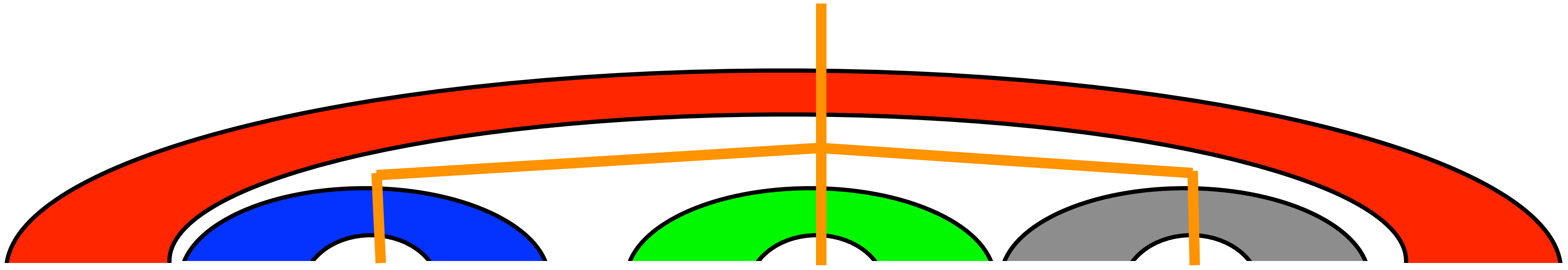
Adj=Adjective

Det=Determiner

Prep=Preposition



S
NP
RC
VP
NP
NP
NP
PN
N
PN
N
V
V
PN
N
Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo



ACCACCUGCUCAGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGUGUGCAGG

....<<<<<<<<<<<<.......>>>>>...<<<<<.....>>>>><<<<.......>>>>.>>>>>>>

Nussinov algorithm

Ruth Nussinov

- Statement of problem:
  "Given an RNA sequence, determine the foldback structure with the most number of Watson-Crick base pairs"
- Slightly different formulation:
  "Given an RNA sequence, determine the maximum number of WC basepairs it can form"
- Solving the 2nd problem will solve the 1st

Nussinov decomposition

- Let X be a sequence and let S be its best foldback structure. Consider 5’- & 3’-terminal bases of S.

- S involves either...

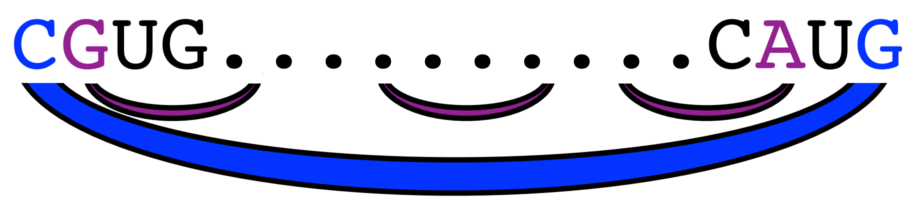
Nussinov decomposition

- Let X be a sequence and let S be its <u>best</u> foldback structure. Consider 5'- & 3'-terminal bases of S.
- S involves either...
  (a) terminal bases paired to each other,

(a)

X:

S:



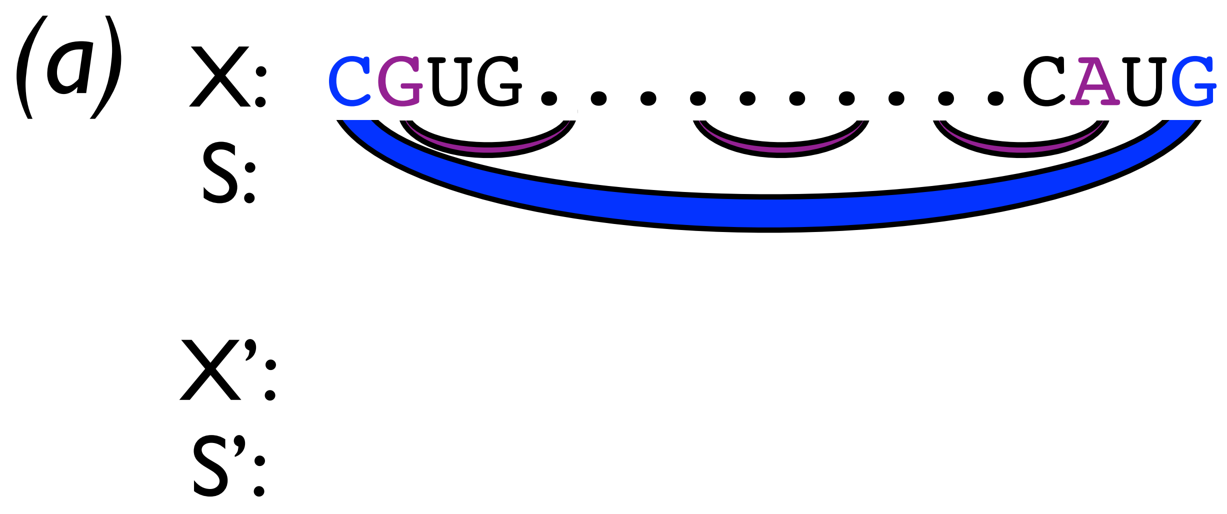
X':

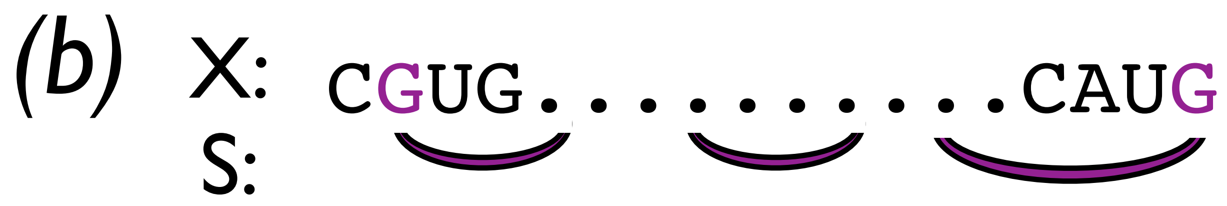
S':

Nussinov decomposition

- Let X be a sequence and let S be its best foldback structure. Consider 5’- & 3’-terminal bases of S.

- S involves either...
  (a) terminal bases paired to each other,
  (b) at least one unpaired terminal nucleotide, or

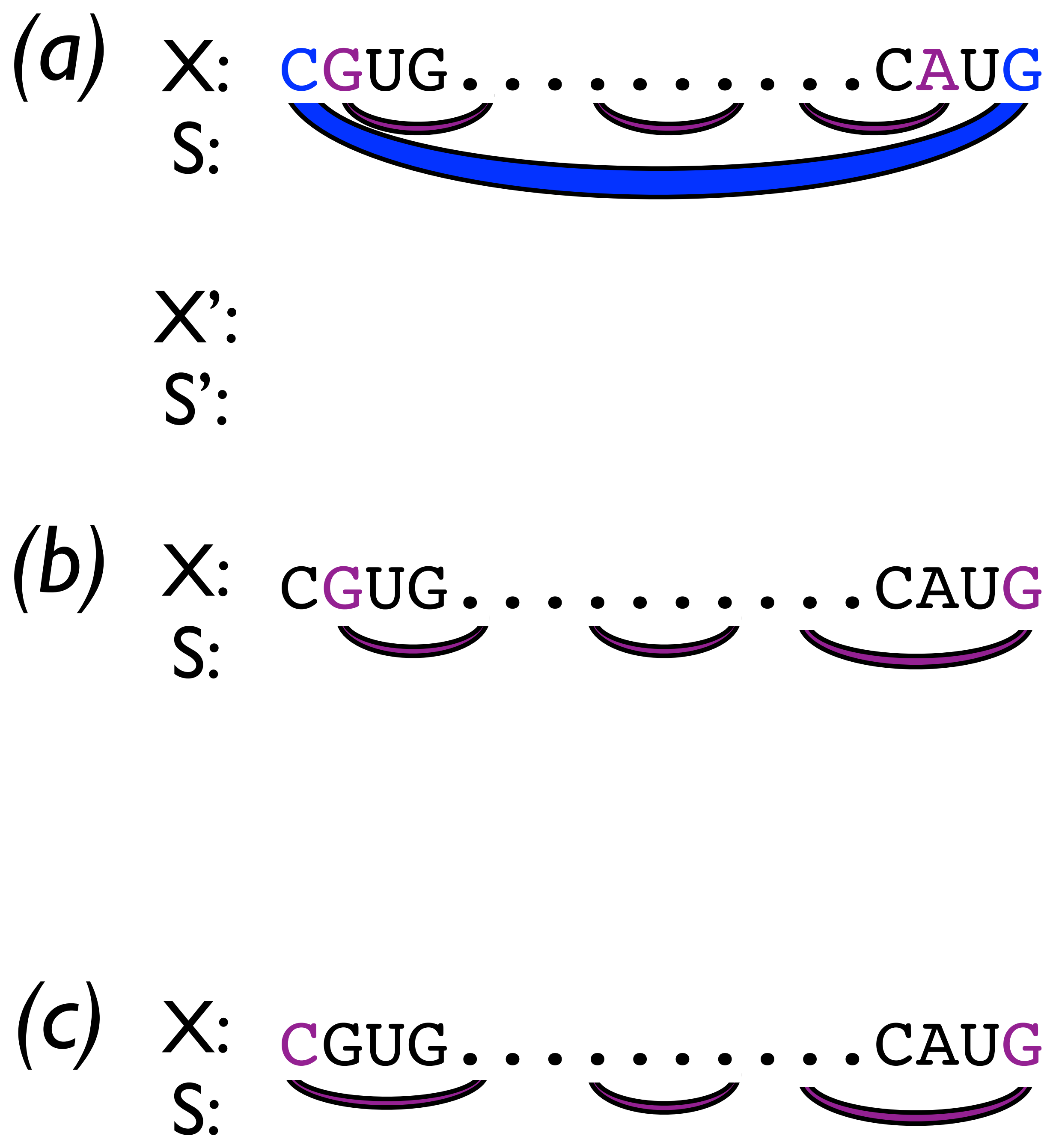




Nussinov decomposition

- Let X be a sequence and let S be its best foldback structure. Consider 5’- & 3’-terminal bases of S.

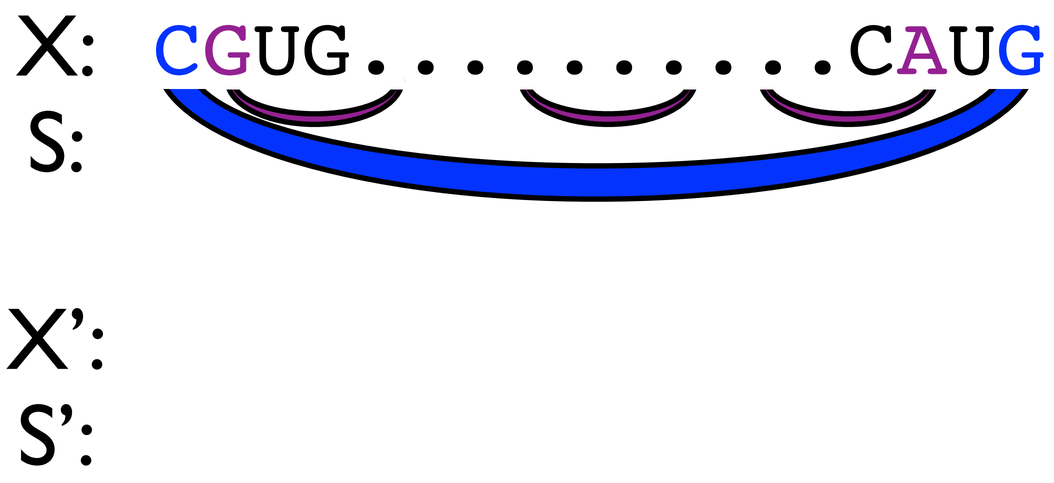
- S involves either...
  (a) terminal bases paired to each other,
  (b) at least one unpaired terminal nucleotide, or
  (c) terminal bases paired, but not to each other (implying a *bifurcation* in the structure).



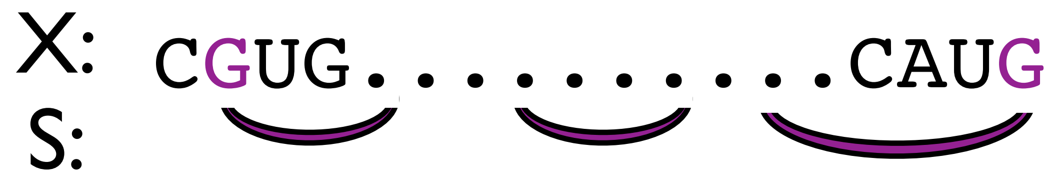
Nussinov decomposition

- Let X be a sequence and let S be its <u>best</u> foldback structure. Consider 5'- & 3'-terminal bases of S.
- S involves either...
  (a) terminal bases paired to each other,
  (b) at least one unpaired terminal nucleotide, or
  (c) terminal bases paired, but not to each other (implying a *bifurcation* in the structure).
- Cases *(a)* and *(b)*:
  - Removing the terminal nucleotides *(a)* or nucleotide *(b)* from S yields a subsequence X' with a structure S'
  - **S' is the optimal structure of X'**

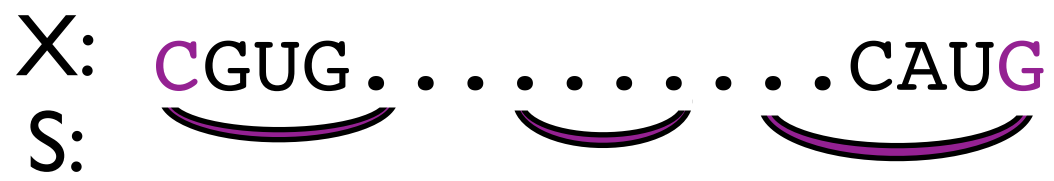
(a)



(b)



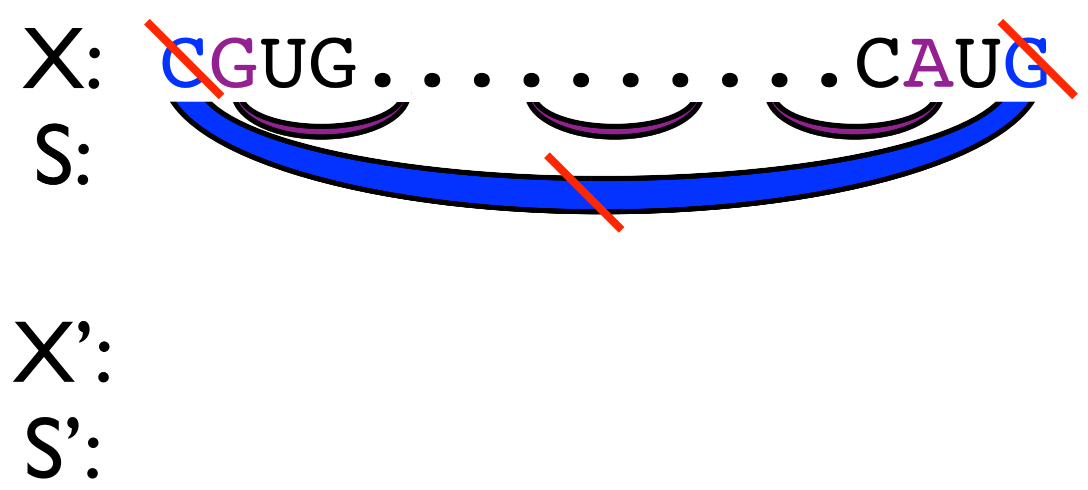
(c)



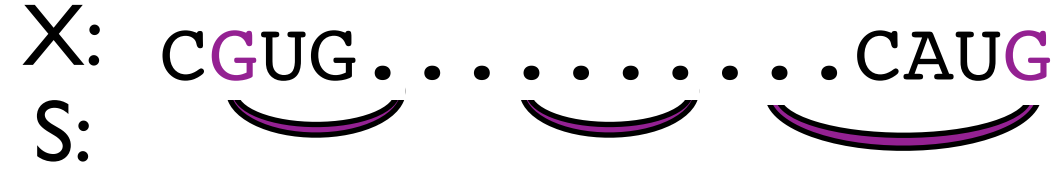
Nussinov decomposition

- Let X be a sequence and let S be its <u>best</u> foldback structure. Consider 5’- & 3’-terminal bases of S.
- S involves either...
  (a) terminal bases paired to each other,
  (b) at least one unpaired terminal nucleotide, or
  (c) terminal bases paired, but not to each other (implying a *bifurcation* in the structure).
- Cases *(a)* and *(b)*:
  - Removing the terminal nucleotides *(a)* or nucleotide *(b)* from S yields a subsequence X’ with a structure S’
  - **S’ is the optimal structure of X’**

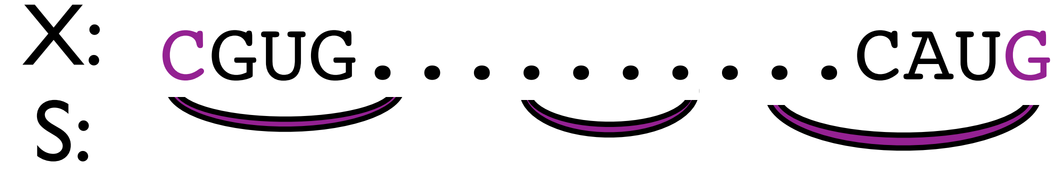
(a)



(b)

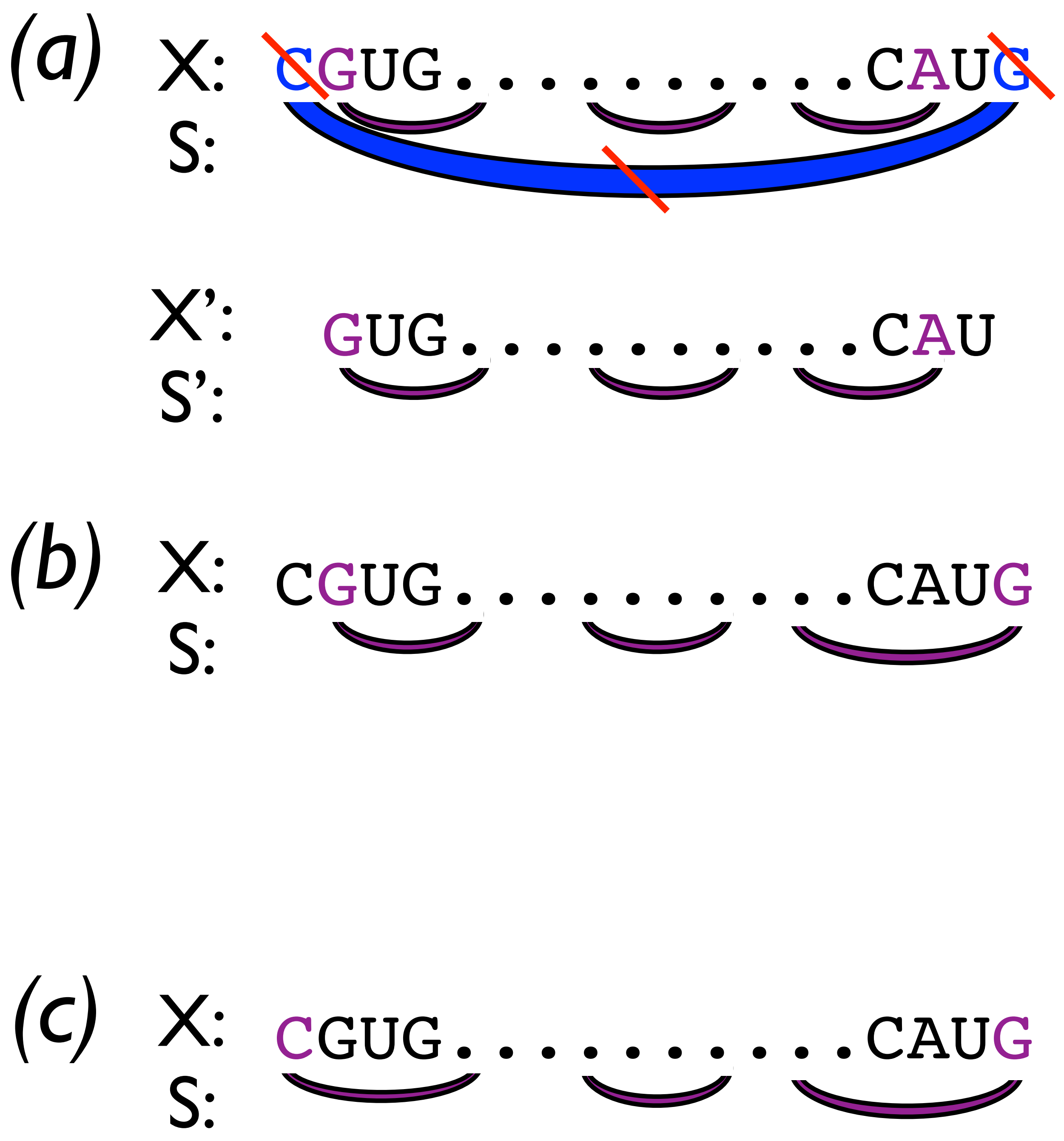


(c)



Nussinov decomposition

- Let X be a sequence and let S be its best foldback structure. Consider 5'- & 3'-terminal bases of S.
- S involves either...
  (a) terminal bases paired to each other,
  (b) at least one unpaired terminal nucleotide, or
  (c) terminal bases paired, but not to each other (implying a *bifurcation* in the structure).
- Cases *(a)* and *(b)*:
  - Removing the terminal nucleotides *(a)* or nucleotide *(b)* from S yields a subsequence X' with a structure S'
  - **S' is the optimal structure of X'**



Nussinov decomposition

- Let X be a sequence and let S be its <u>best</u> foldback structure. Consider 5’- & 3’-terminal bases of S.

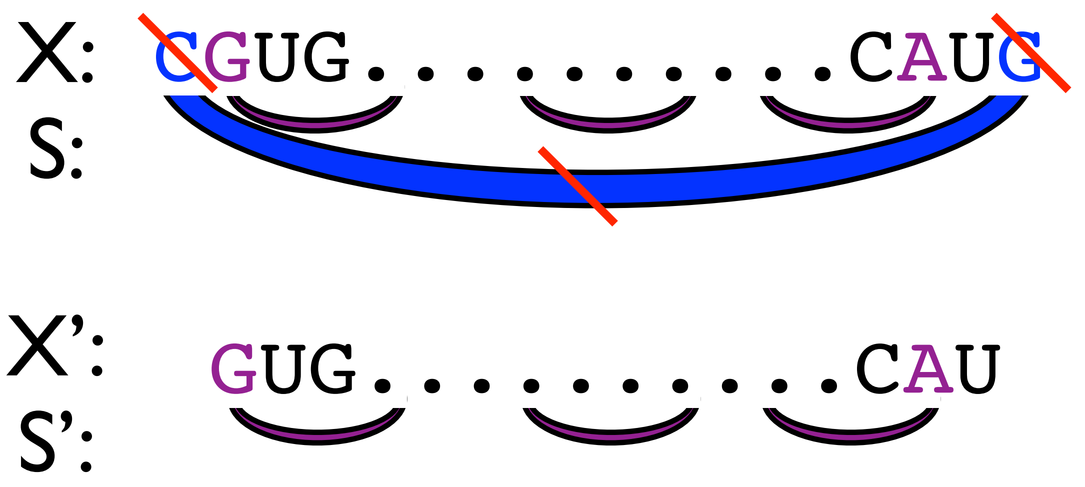
- S involves either...
  (a) terminal bases paired to each other,
  (b) at least one unpaired terminal nucleotide, or
  (c) terminal bases paired, but not to each other (implying a *bifurcation* in the structure).

- Cases *(a)* and *(b)*:

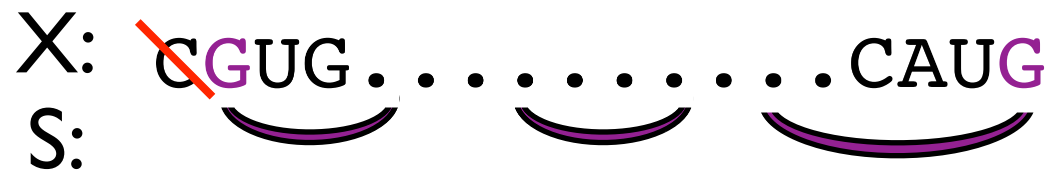
  - Removing the terminal nucleotides *(a)* or nucleotide *(b)* from S yields a subsequence X’ with a structure S’

  - **S’ is the optimal structure of X’**

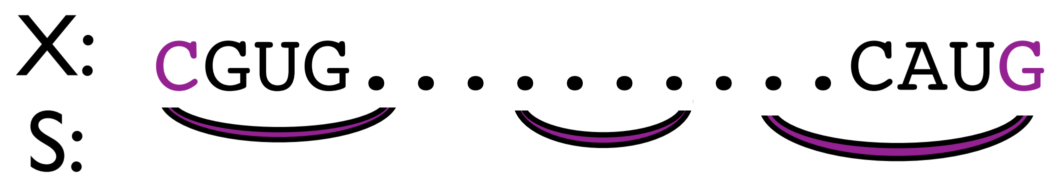
(a)



(b)

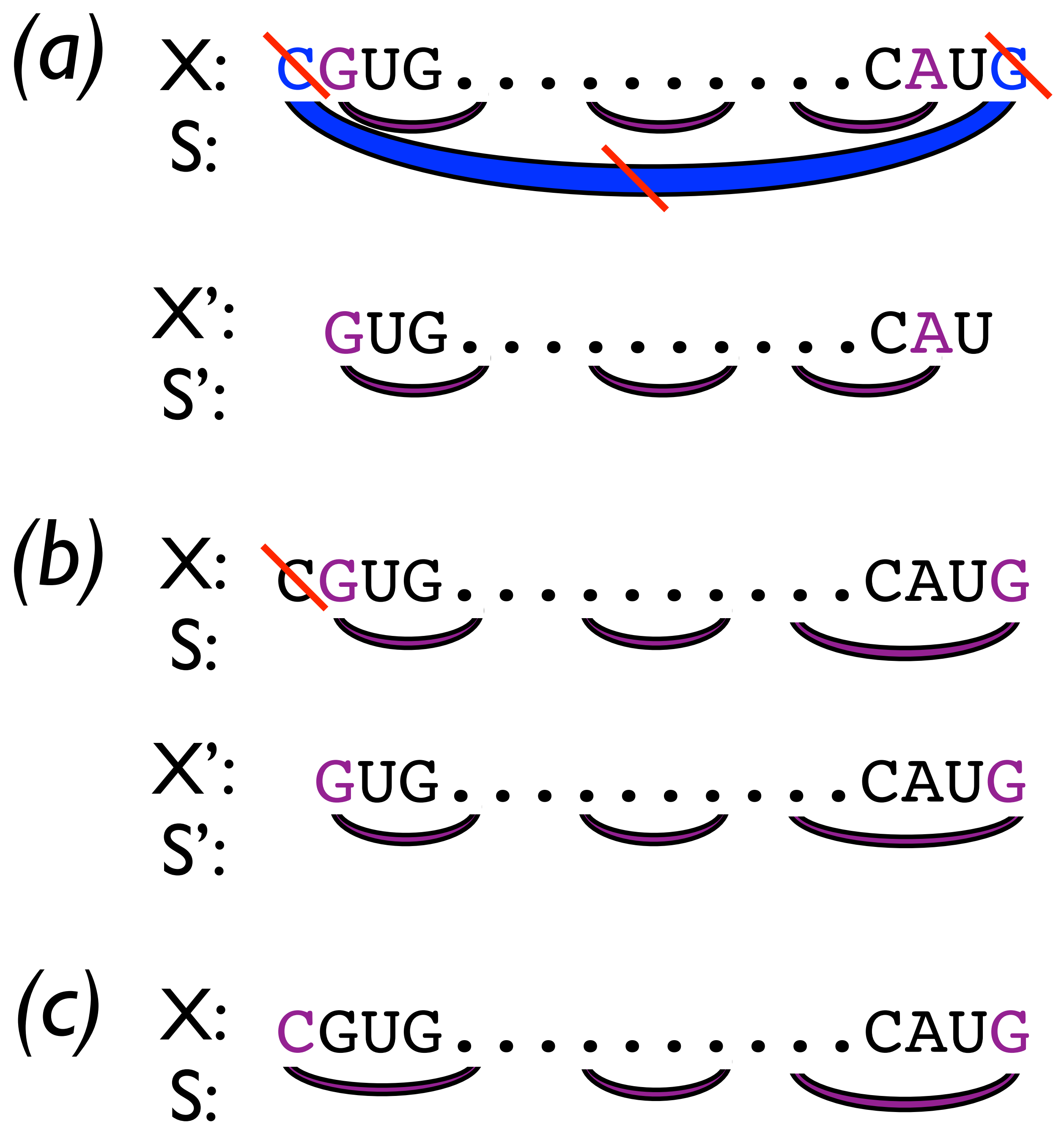


(c)



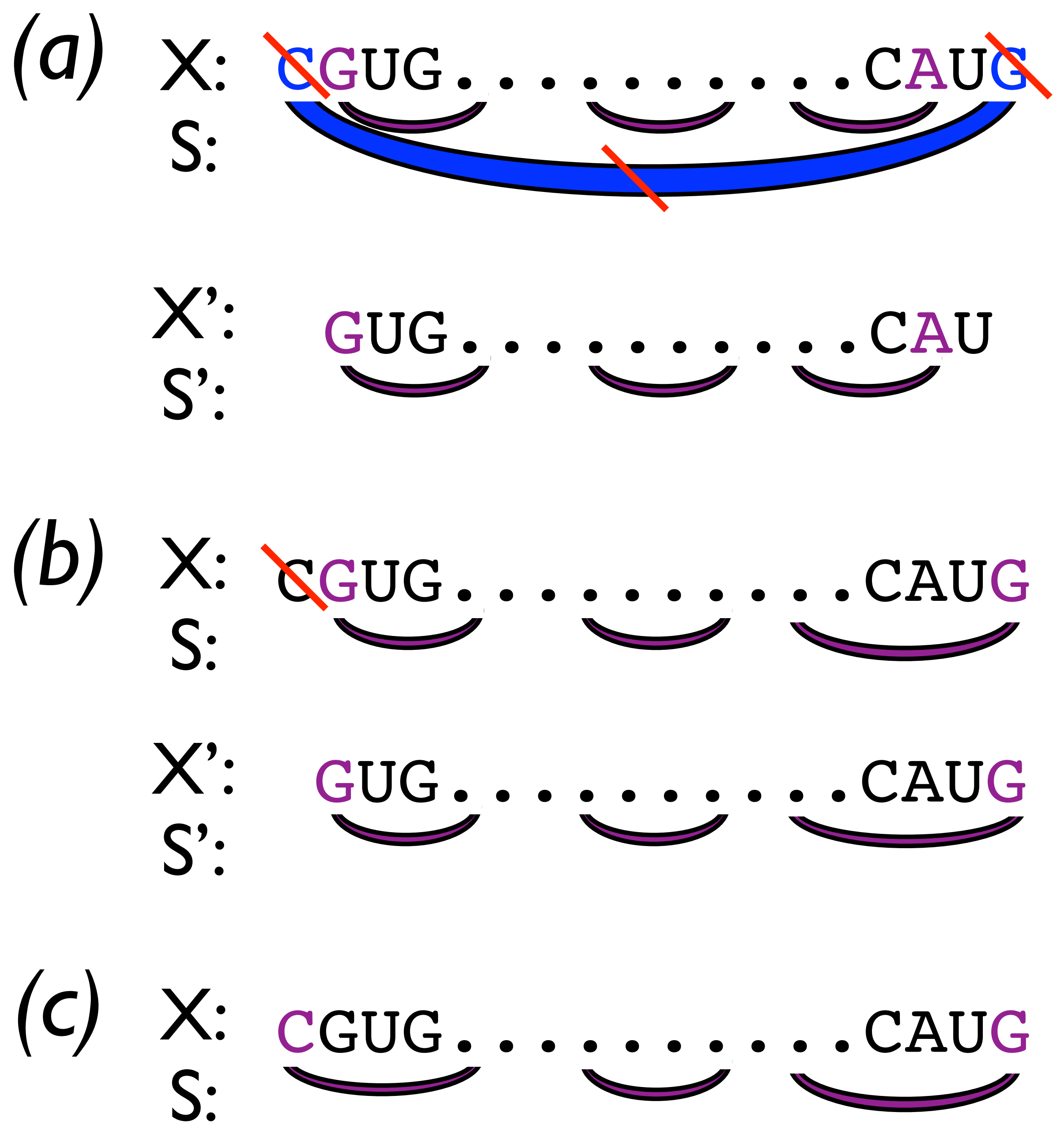
Nussinov decomposition

- Let X be a sequence and let S be its best foldback structure. Consider 5'- & 3'-terminal bases of S.
- S involves either...
  (a) terminal bases paired to each other,
  (b) at least one unpaired terminal nucleotide, or
  (c) terminal bases paired, but not to each other (implying a *bifurcation* in the structure).
- Cases *(a)* and *(b)*:
  - Removing the terminal nucleotides *(a)* or nucleotide *(b)* from S yields a subsequence X' with a structure S'
  - **S' is the optimal structure of X'**



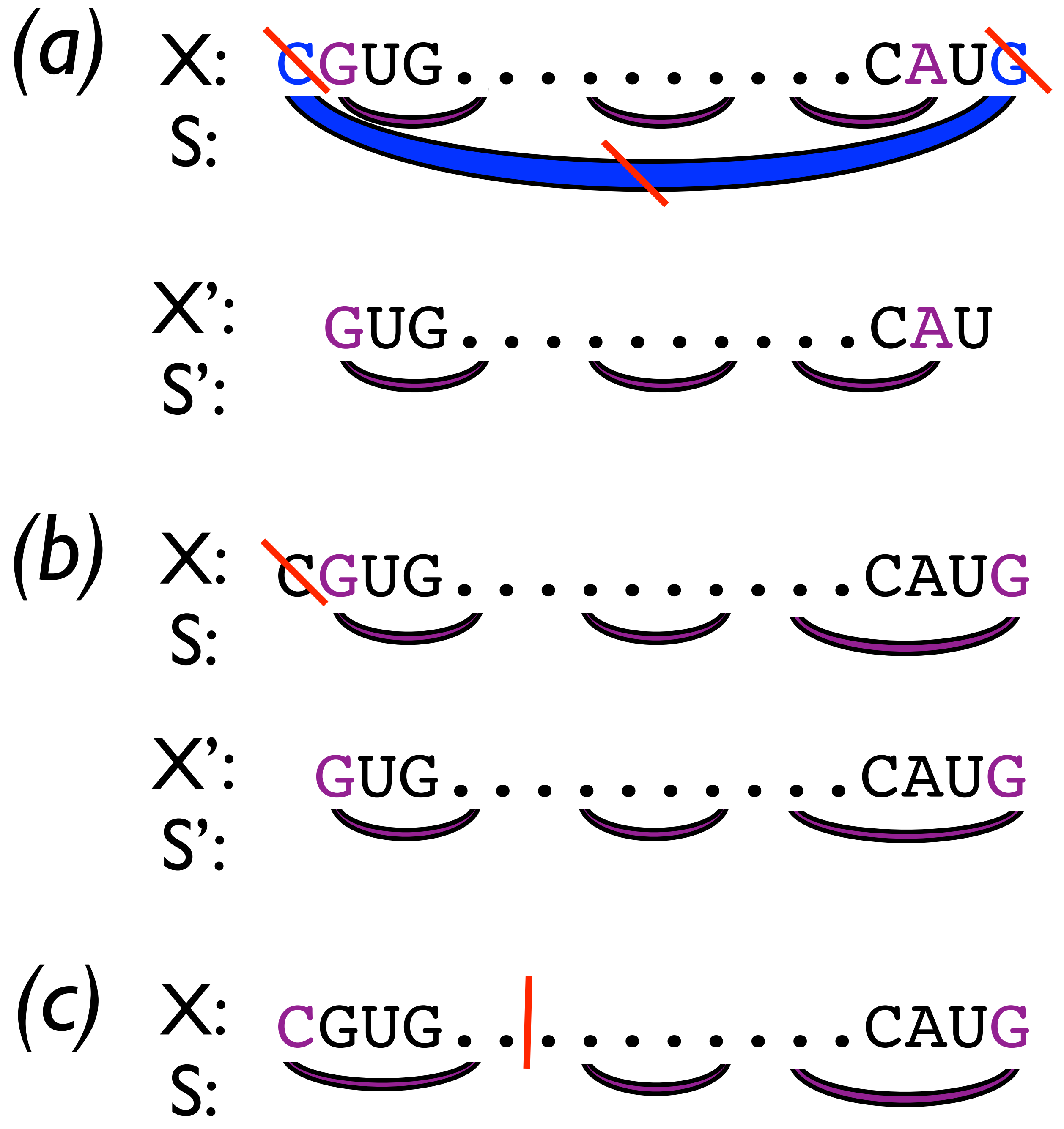
Nussinov decomposition

- Let X be a sequence and let S be its best foldback structure. Consider 5'- & 3'-terminal bases of S.
- S involves either...
  (a) terminal bases paired to each other,
  (b) at least one unpaired terminal nucleotide, or
  (c) terminal bases paired, but not to each other (implying a *bifurcation* in the structure).
- Cases *(a)* and *(b)*:
  - Removing the terminal nucleotides *(a)* or nucleotide *(b)* from S yields a subsequence X' with a structure S'
  - **S' is the optimal structure of X'**
- Case *(c)*:
  - Splitting at the bifurcation point yields two subsequences (X',X'') with structures (S',S'')
  - S' is optimal structure of X'; ditto S'' for X''



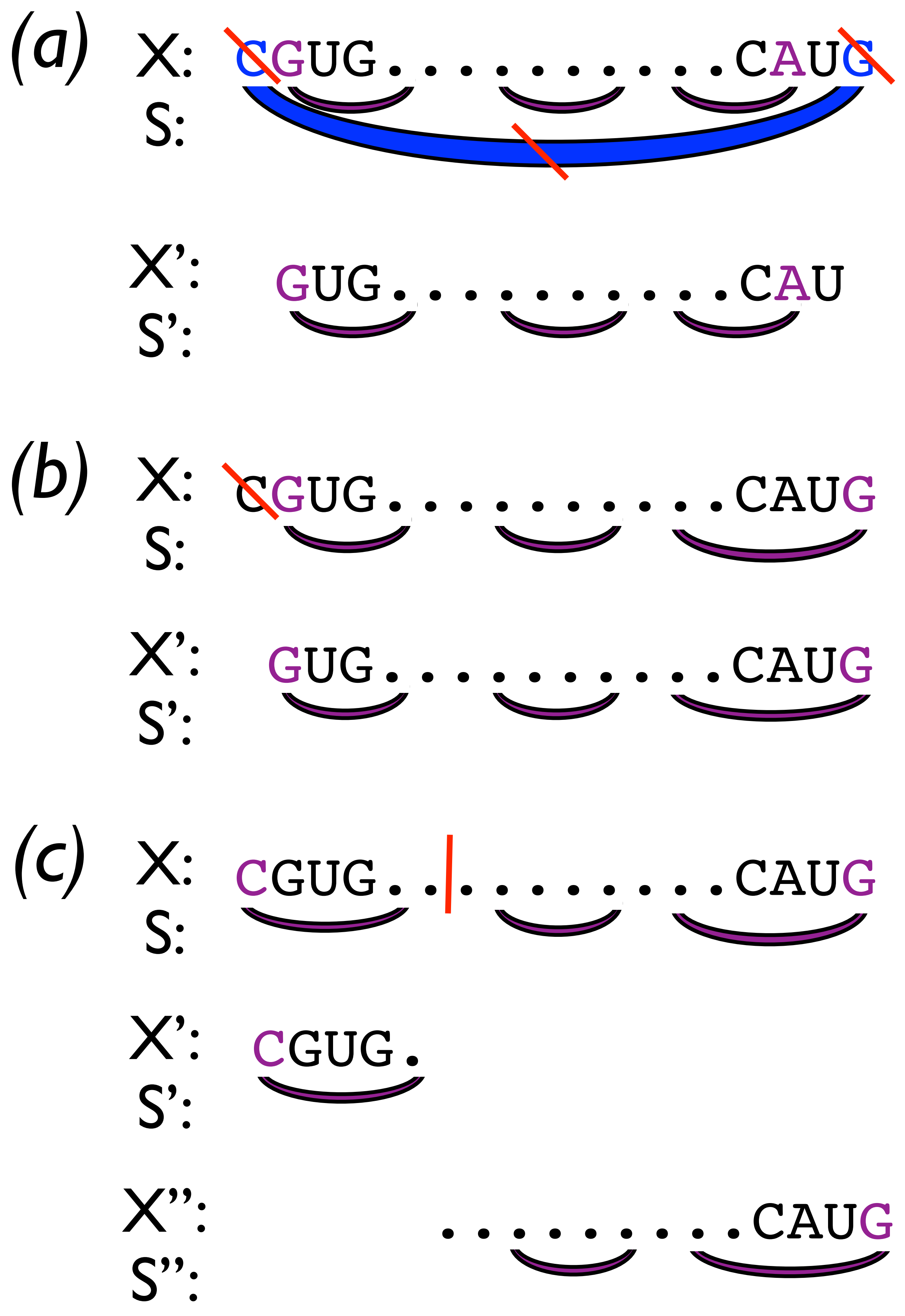
Nussinov decomposition

- Let X be a sequence and let S be its best foldback structure. Consider 5'- & 3'-terminal bases of S.
- S involves either...
  (a) terminal bases paired to each other,
  (b) at least one unpaired terminal nucleotide, or
  (c) terminal bases paired, but not to each other (implying a *bifurcation* in the structure).
- Cases *(a)* and *(b)*:
  - Removing the terminal nucleotides *(a)* or nucleotide *(b)* from S yields a subsequence X' with a structure S'
  - **S' is the optimal structure of X'**
- Case *(c)*:
  - Splitting at the bifurcation point yields two subsequences (X',X'') with structures (S',S'')
  - S' is optimal structure of X'; ditto S'' for X''



Nussinov decomposition

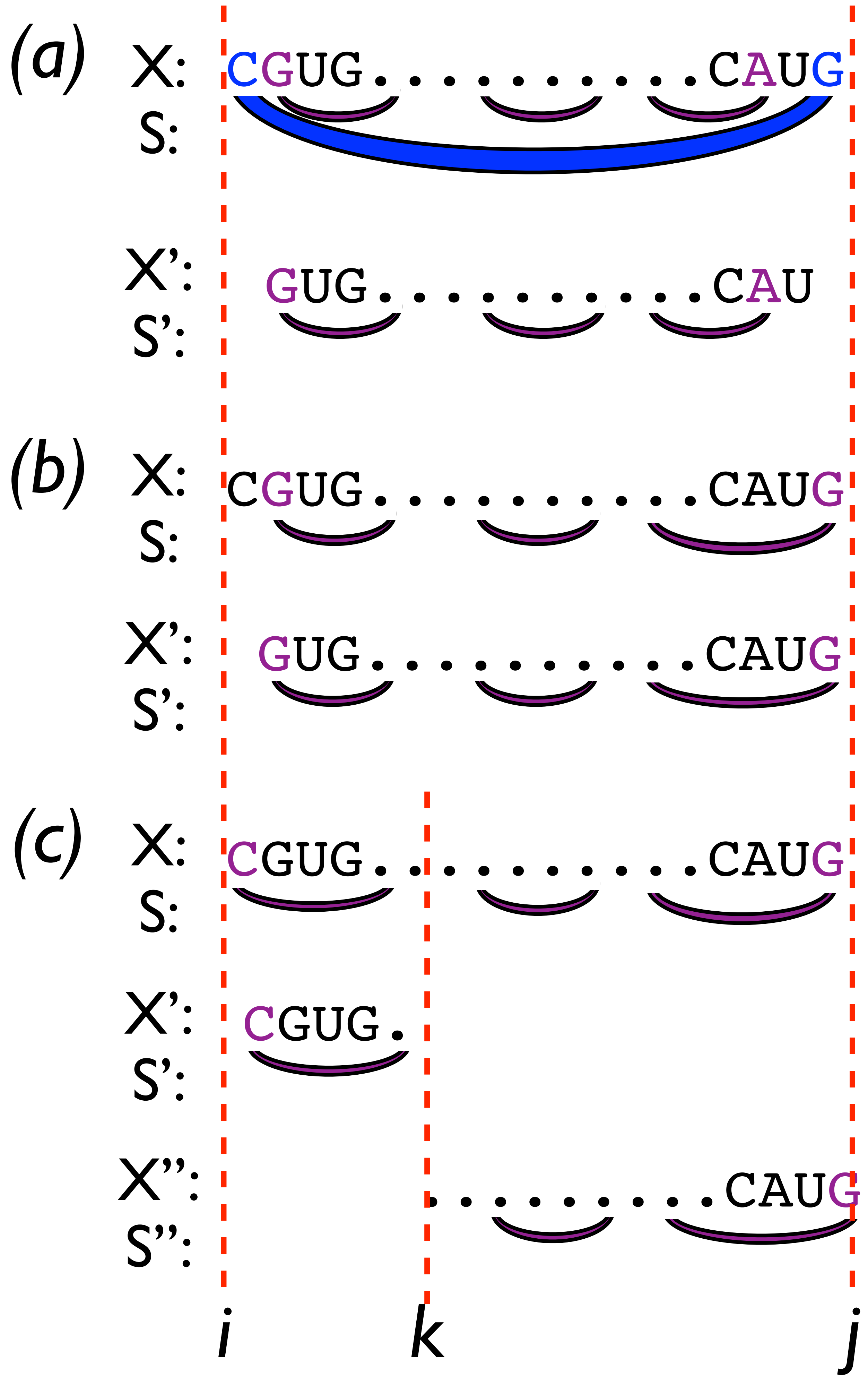
- Let X be a sequence and let S be its best foldback structure. Consider 5'- & 3'-terminal bases of S.
- S involves either...
  (a) terminal bases paired to each other,
  (b) at least one unpaired terminal nucleotide, or
  (c) terminal bases paired, but not to each other (implying a *bifurcation* in the structure).
- Cases *(a)* and *(b)*:
  - Removing the terminal nucleotides *(a)* or nucleotide *(b)* from S yields a subsequence X' with a structure S'
  - **S' is the optimal structure of X'**
- Case *(c)*:
  - Splitting at the bifurcation point yields two subsequences (X',X'') with structures (S',S'')
  - S' is optimal structure of X'; ditto S'' for X''



Nussinov decomposition

N_{ij} Max possible WC basepairs in subsequence from i to j

Nussinov decomposition

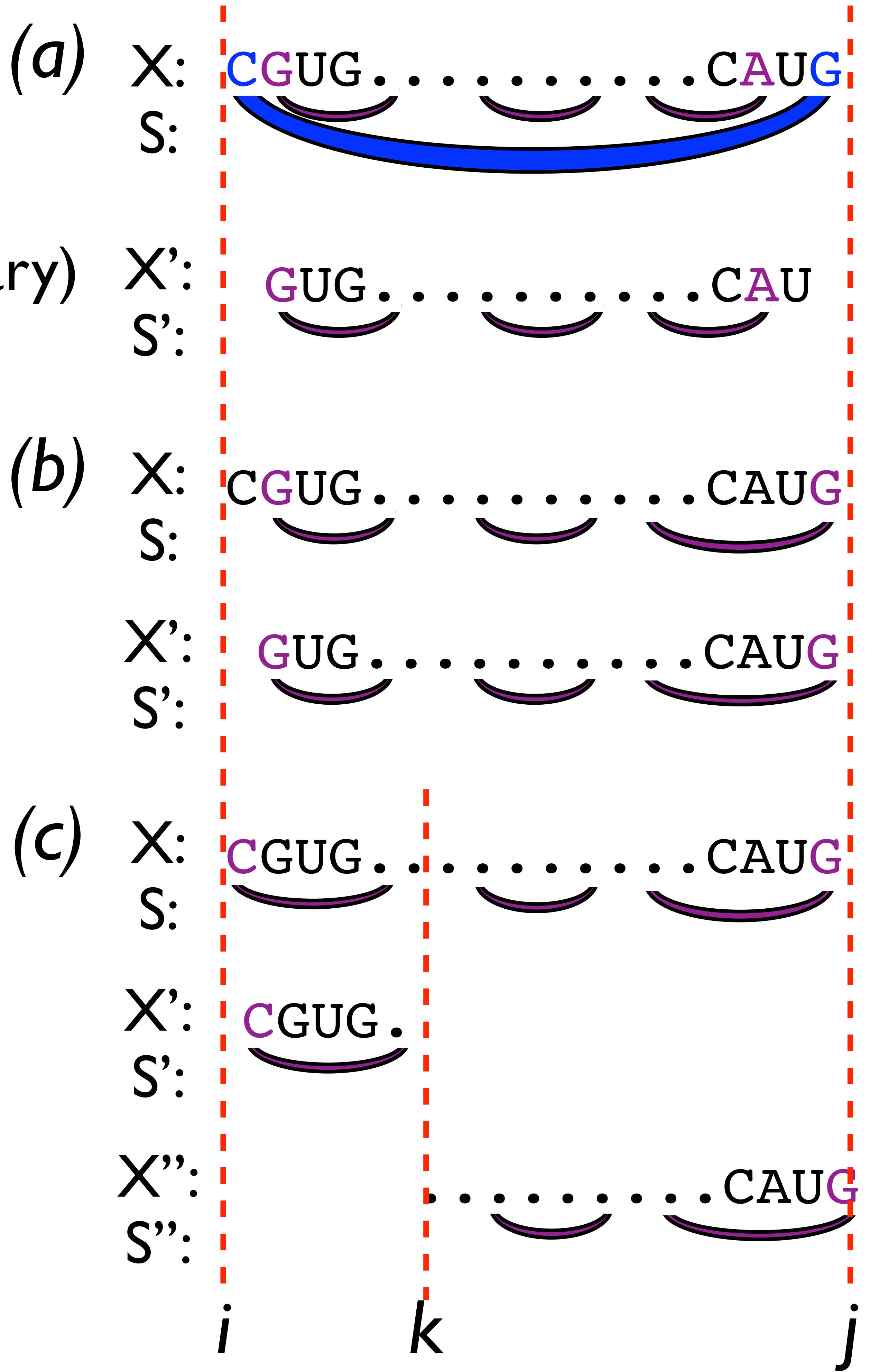


N_{ij} Max possible WC basepairs in subsequence from i to j

Nussinov decomposition

(a) $$N_{ij} = N_{i+1,j-1} + 1$$

(if nucleotides at i & j are complementary)



N_{ij} Max possible WC basepairs in subsequence from i to j

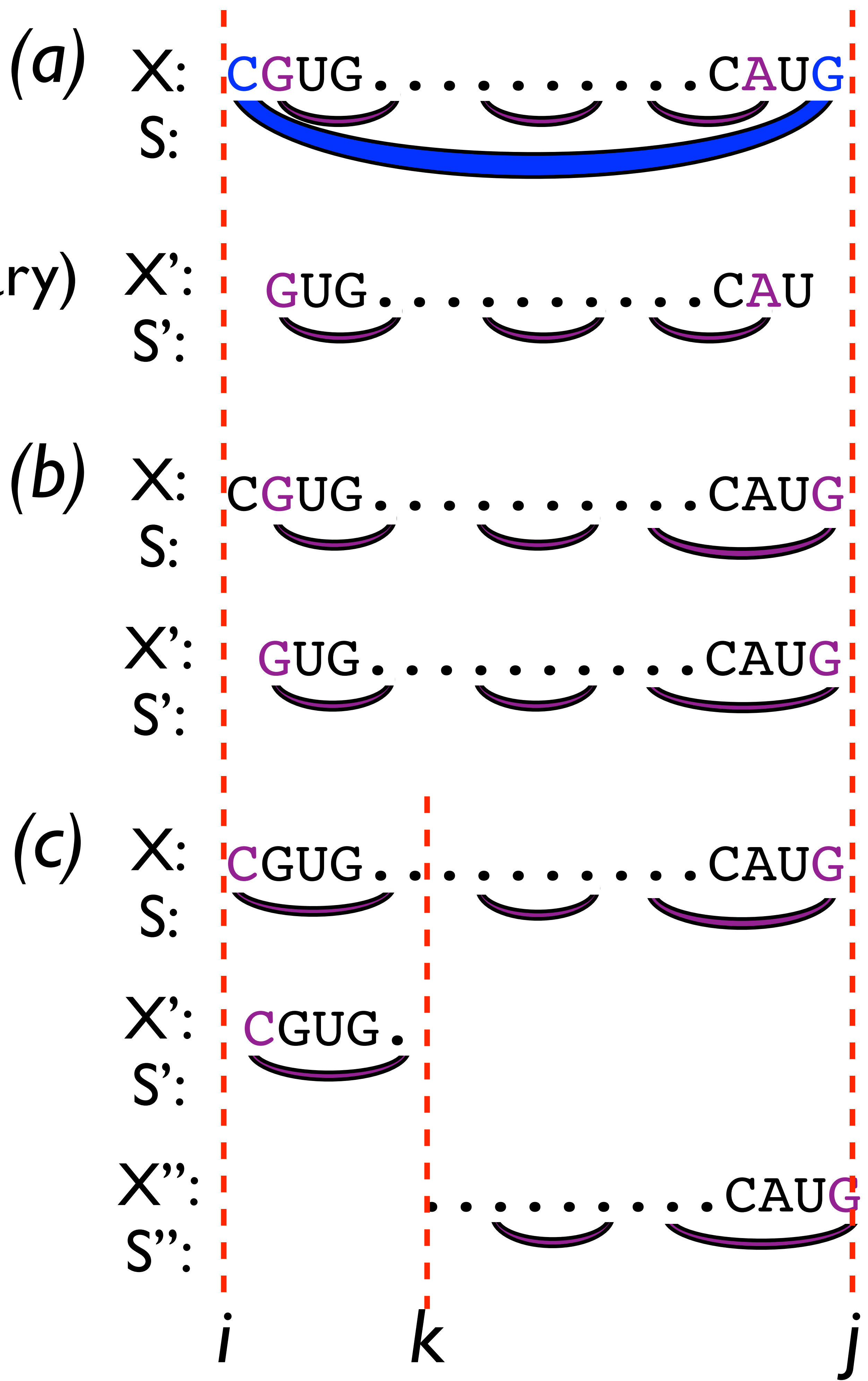
Nussinov decomposition

(a) $$N_{ij} = N_{i+1,j-1} + 1$$

(if nucleotides at i & j are complementary)

(b) $$N_{ij} = N_{i+1,j}$$

or $$N_{ij} = N_{i,j-1}$$



N_{ij} Max possible WC basepairs in subsequence from i to j

Nussinov decomposition

(a) $$N_{ij} = N_{i+1,j-1} + 1$$

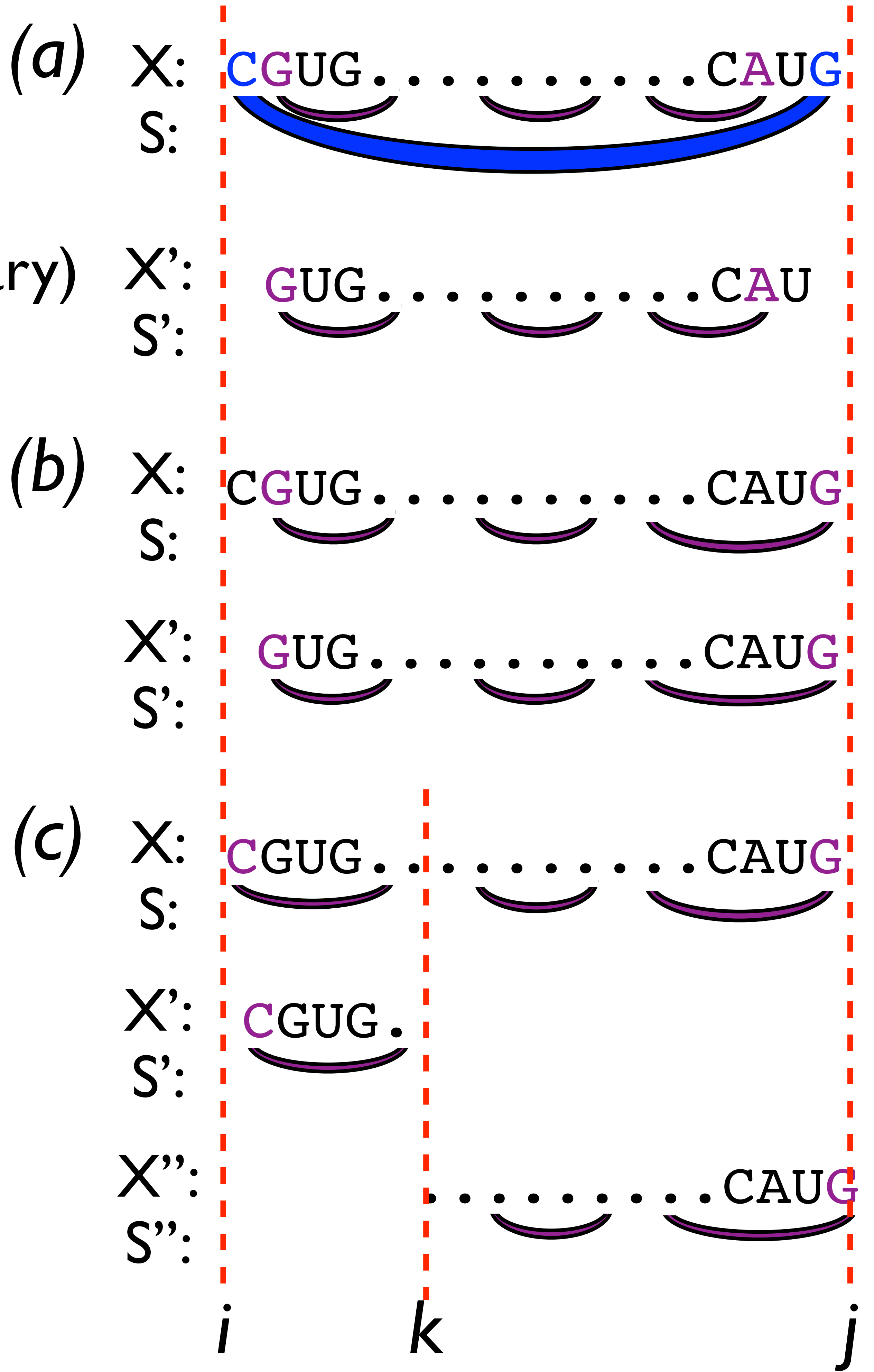
(if nucleotides at i & j are complementary)

(b) $$N_{ij} = N_{i+1,j}$$

or $$N_{ij} = N_{i,j-1}$$

(c) $$N_{ij} = N_{ik} + N_{k+1,j}$$

N_{ij} Max possible WC basepairs in subsequence from i to j



Nussinov decomposition

(a) $$N_{ij} = N_{i+1,j-1} + \delta_{WC}(X_i, X_j)$$

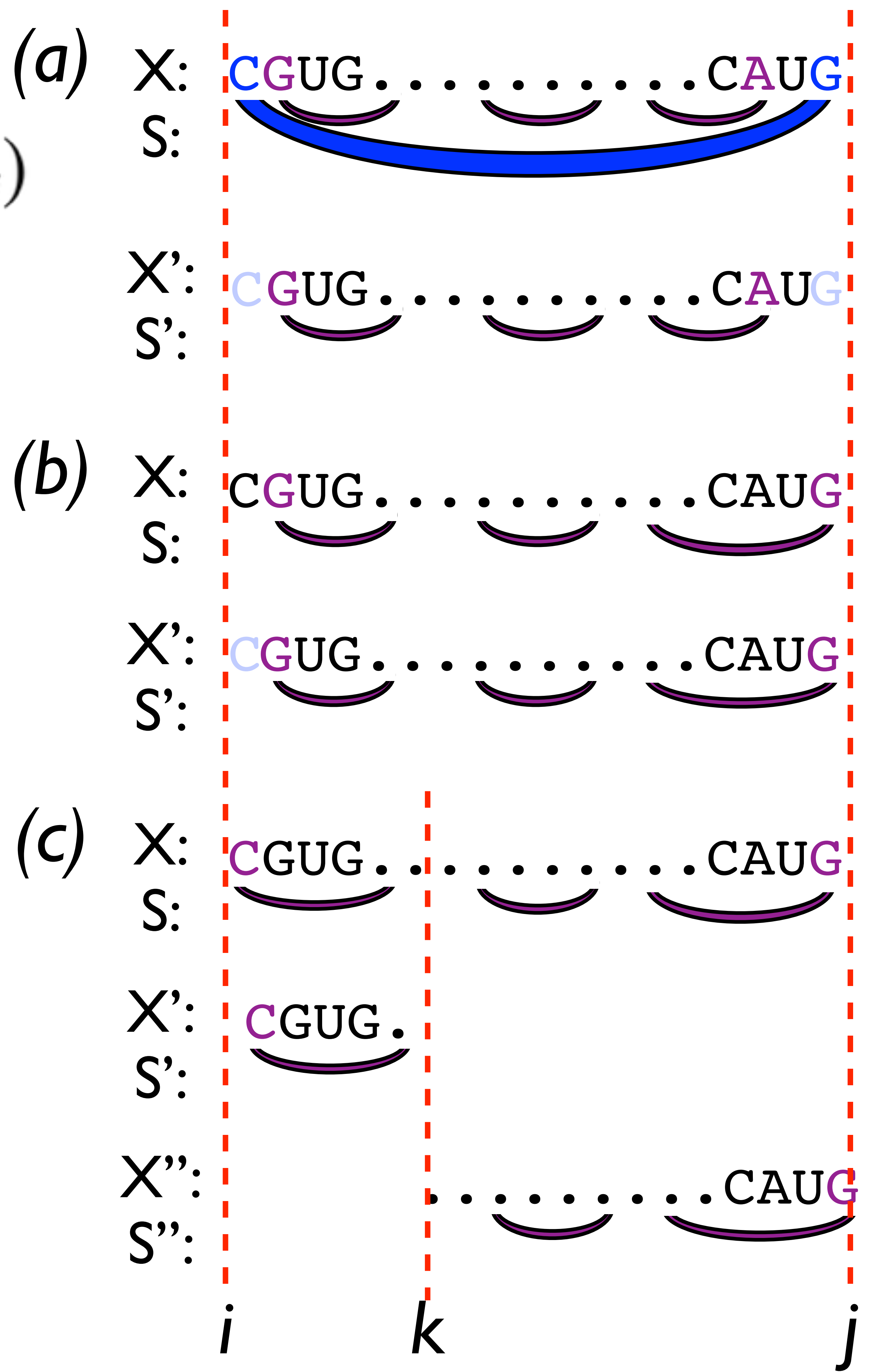
$\delta_{WC}(a,b)$ 1 if a and b are complementary
0 otherwise

(b) $$N_{ij} = N_{i+1,j}$$

or $$N_{ij} = N_{i,j-1}$$

(c) $$N_{ij} = N_{ik} + N_{k+1,j}$$

N_{ij} Max possible WC basepairs in subsequence from i to j



Nussinov decomposition

(a) $N_{ij} = N_{i+1,j-1} + \delta_{WC}(X_i, X_j)$

$\delta_{WC}(a,b)$ 1 if a and b are complementary
0 otherwise

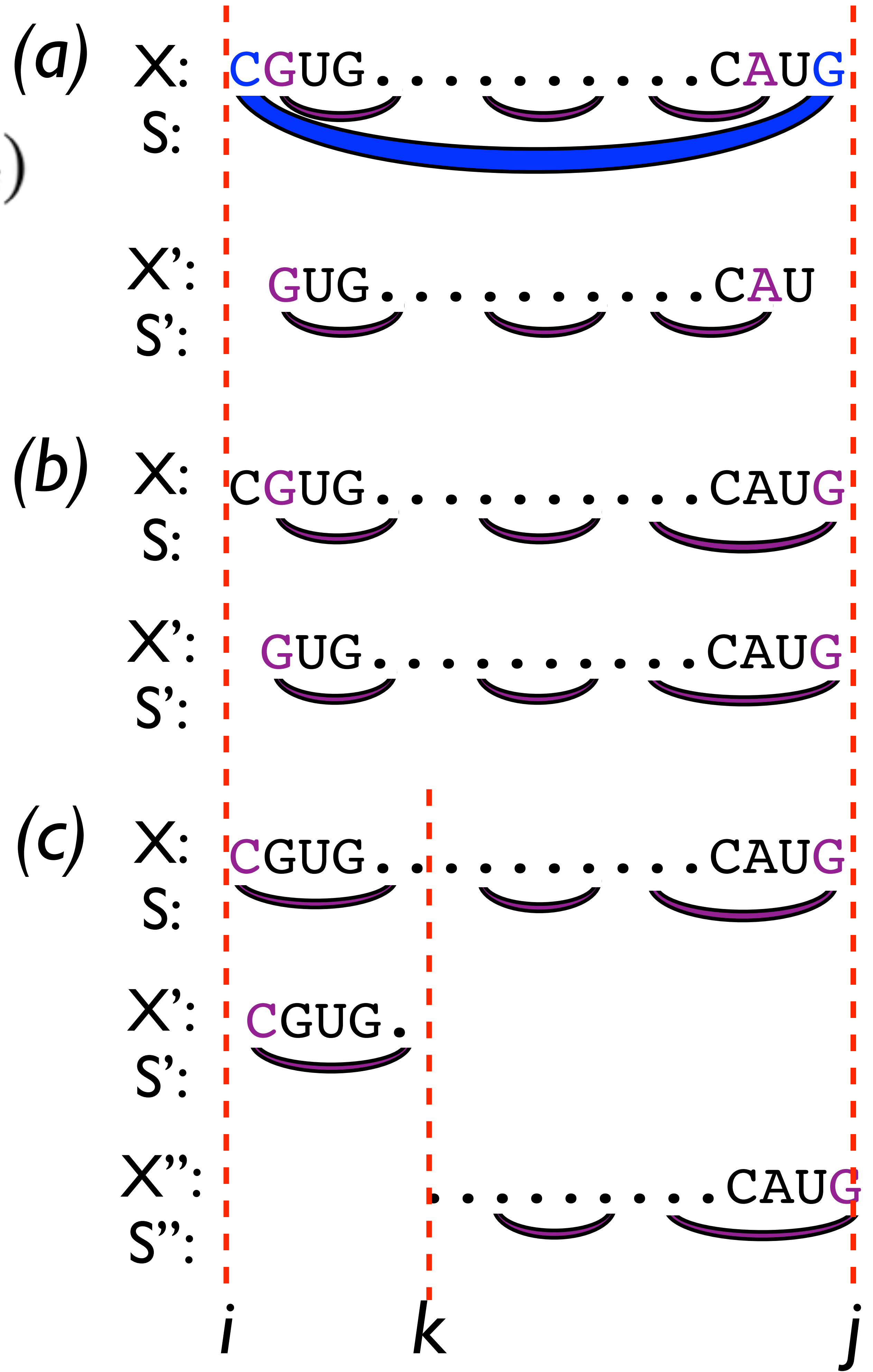
(b) $N_{ij} = N_{i+1,j}$

or $N_{ij} = N_{i,j-1}$

(c) $N_{ij} = \max_{i<k<j}(N_{ik} + N_{k+1,j})$

Since S is the best structure, the bifurcation point (k) must be the best possible bifurcation point...

N_{ij} Max possible WC basepairs in subsequence from i to j



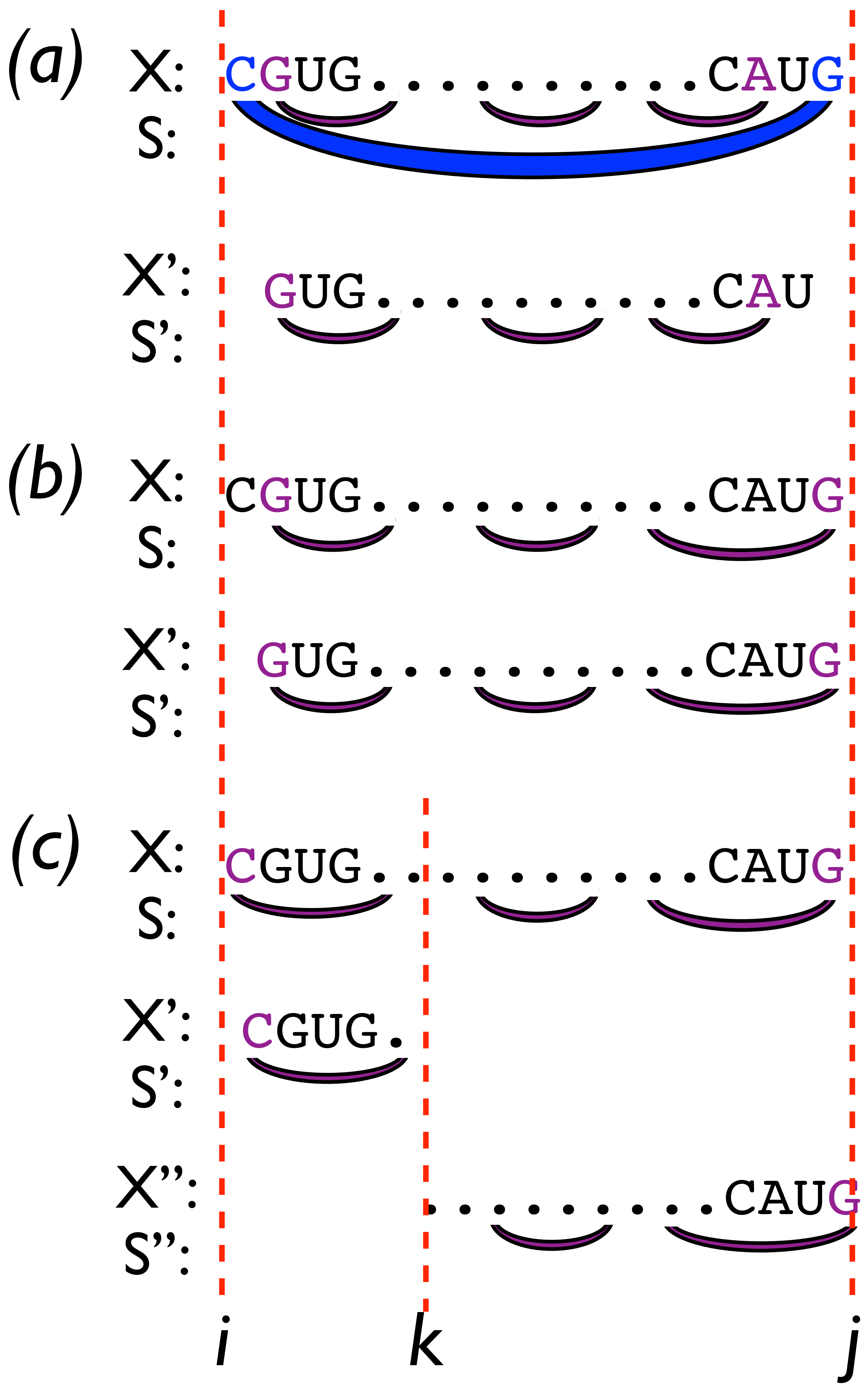
Nussinov decomposition

$$N_{ij} = \max \begin{pmatrix} N_{i+1,j-1} + \delta_{WC}(X_i, X_j) \\ N_{i+1,j} \\ N_{i,j-1} \\ \max_{i<k<j}(N_{ik} + N_{k+1,j}) \end{pmatrix} \text{(if } j > i\text{)}$$

$$N_{ii} = 0$$

...in fact, again because S is the best structure, we know it must have the most basepairs of all four scenarios (including both b-scenarios)

N_{ij} Max possible WC basepairs in subsequence from i to j



Nussinov recursion

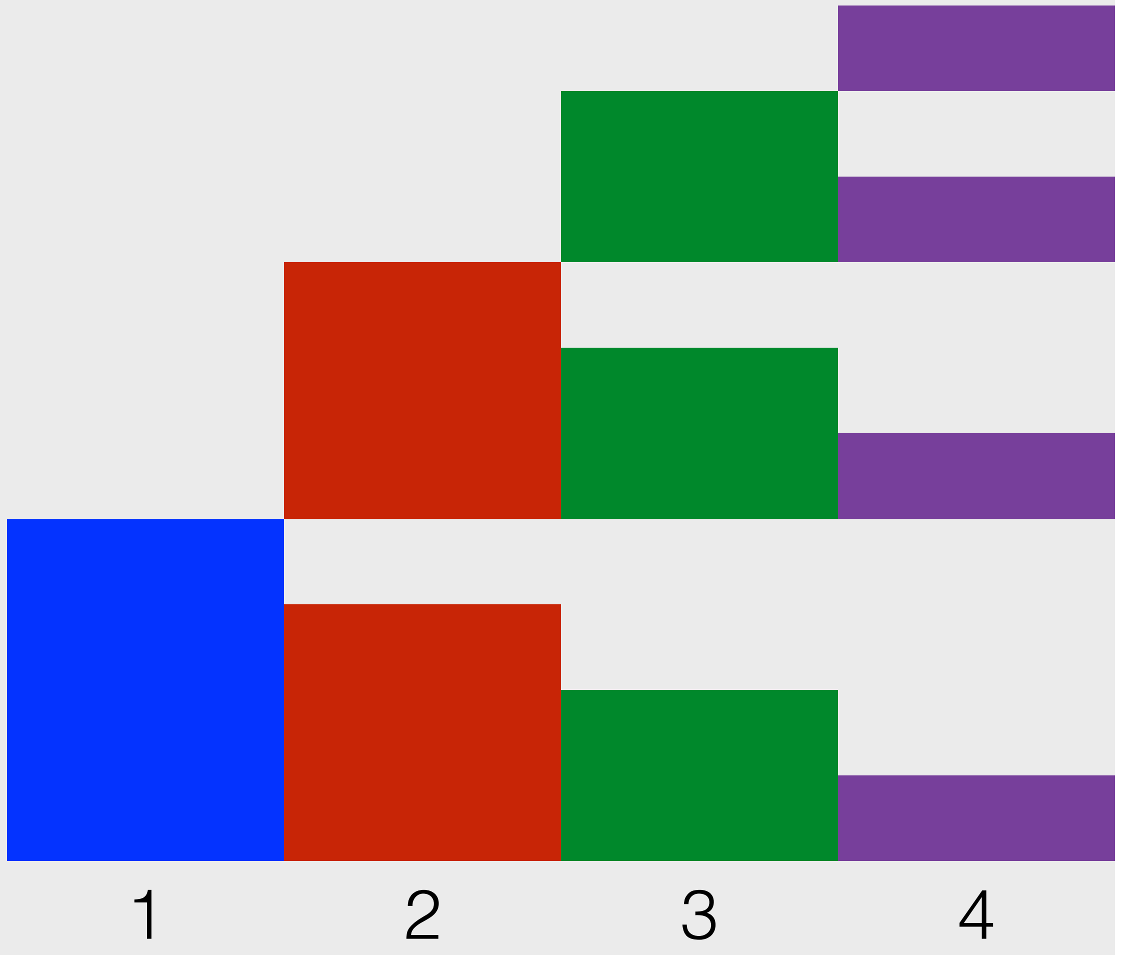
$$N_{ij} = \max\begin{pmatrix} N_{i+1,j-1} + \delta_{WC}(X_i, X_j) \\ N_{i+1,j} \\ N_{i,j-1} \\ \max_{i<k<j}(N_{ik} + N_{k+1,j}) \end{pmatrix} (\text{if } j > i)$$

$$N_{ii} = 0$$

N_{ij}	Max possible WC basepairs in subsequence from i to j
X_i	Nucleotide at position i
$\delta_{WC}(a,b)$	1 if a and b are WC complementary; 0 otherwise

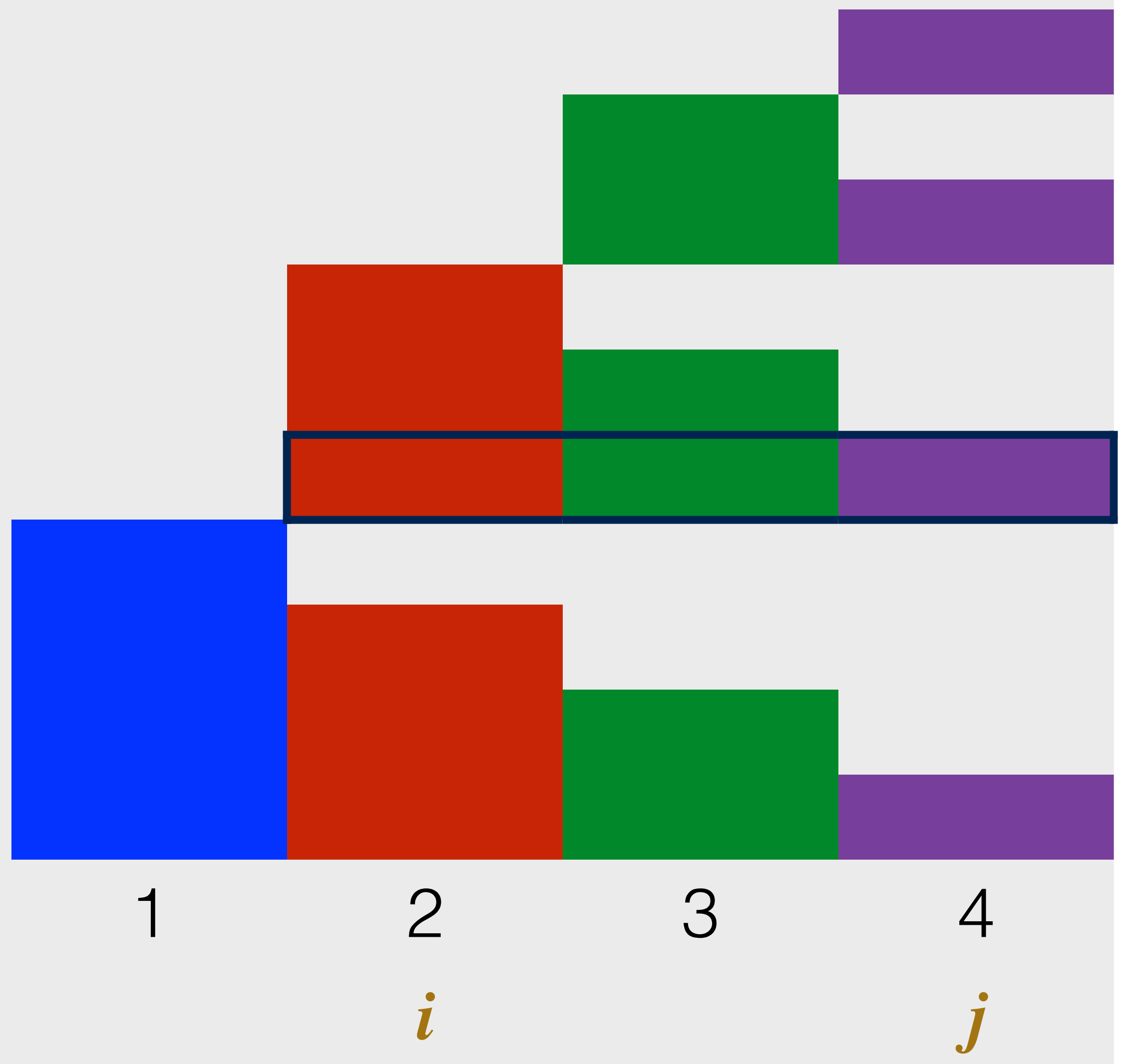
Nussinov fill order

i	j
4	4
3	3
3	4
2	2
2	3
2	4
1	1
1	2
1	3
1	4



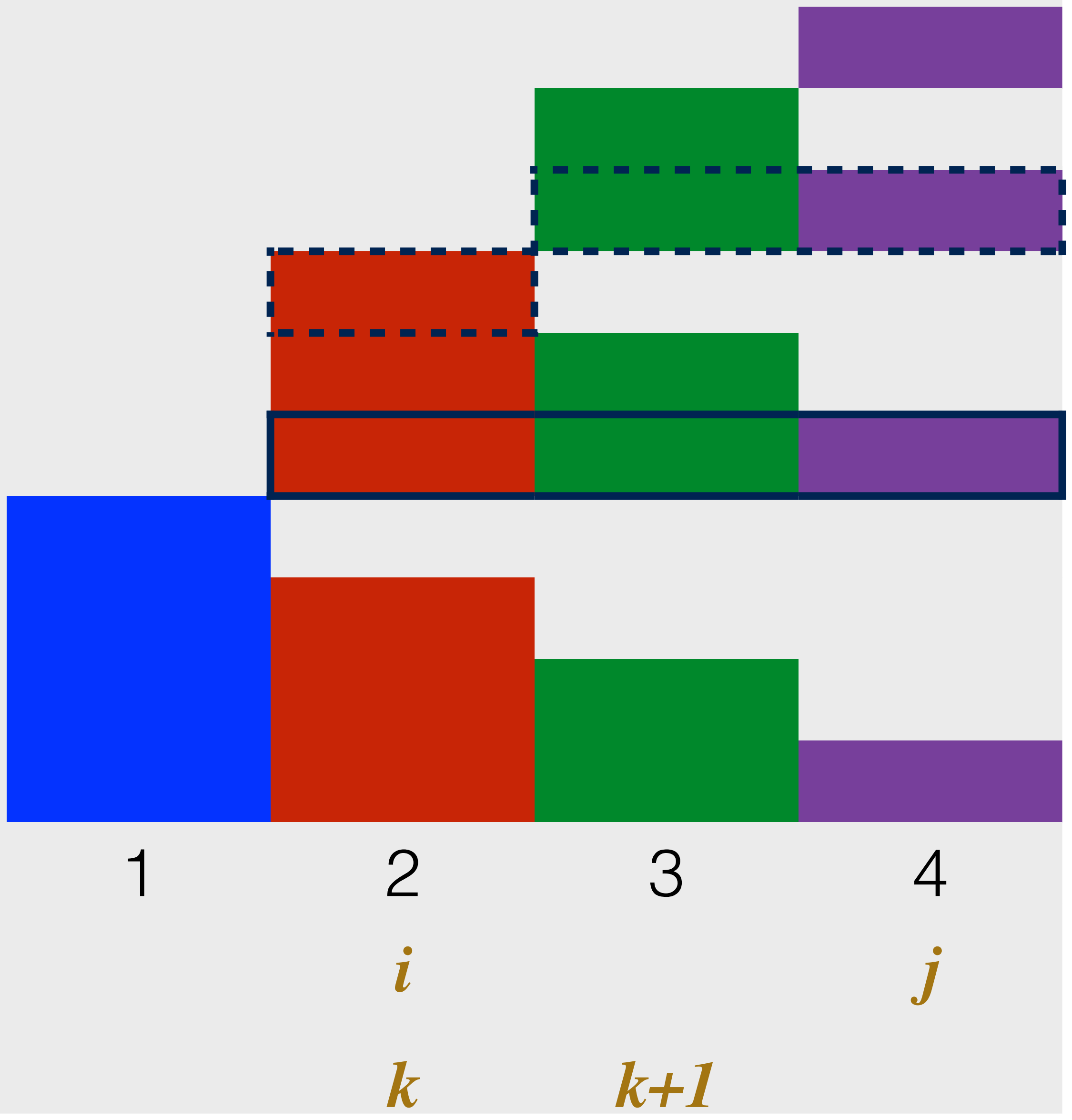
(i,j) = (2,4)

i	j
4	4
3	3
3	4
2	2
2	3
2	4
1	1
1	2
1	3
1	4

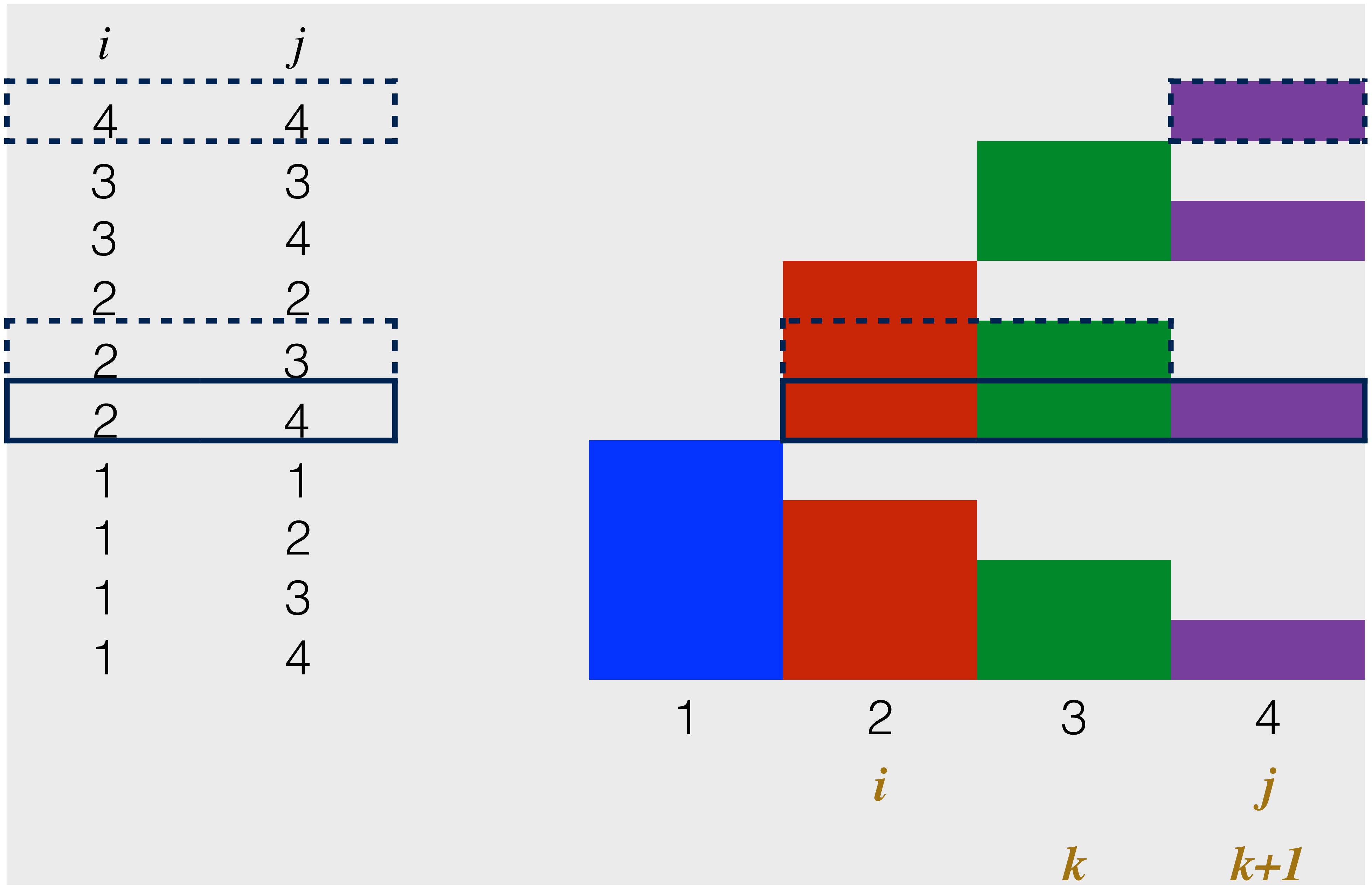


$(i,j) = (i,k).(k+1,j)$
$i=2, k=2, j=4$

i	j
4	4
3	3
3	4
2	2
2	3
2	4
1	1
1	2
1	3
1	4

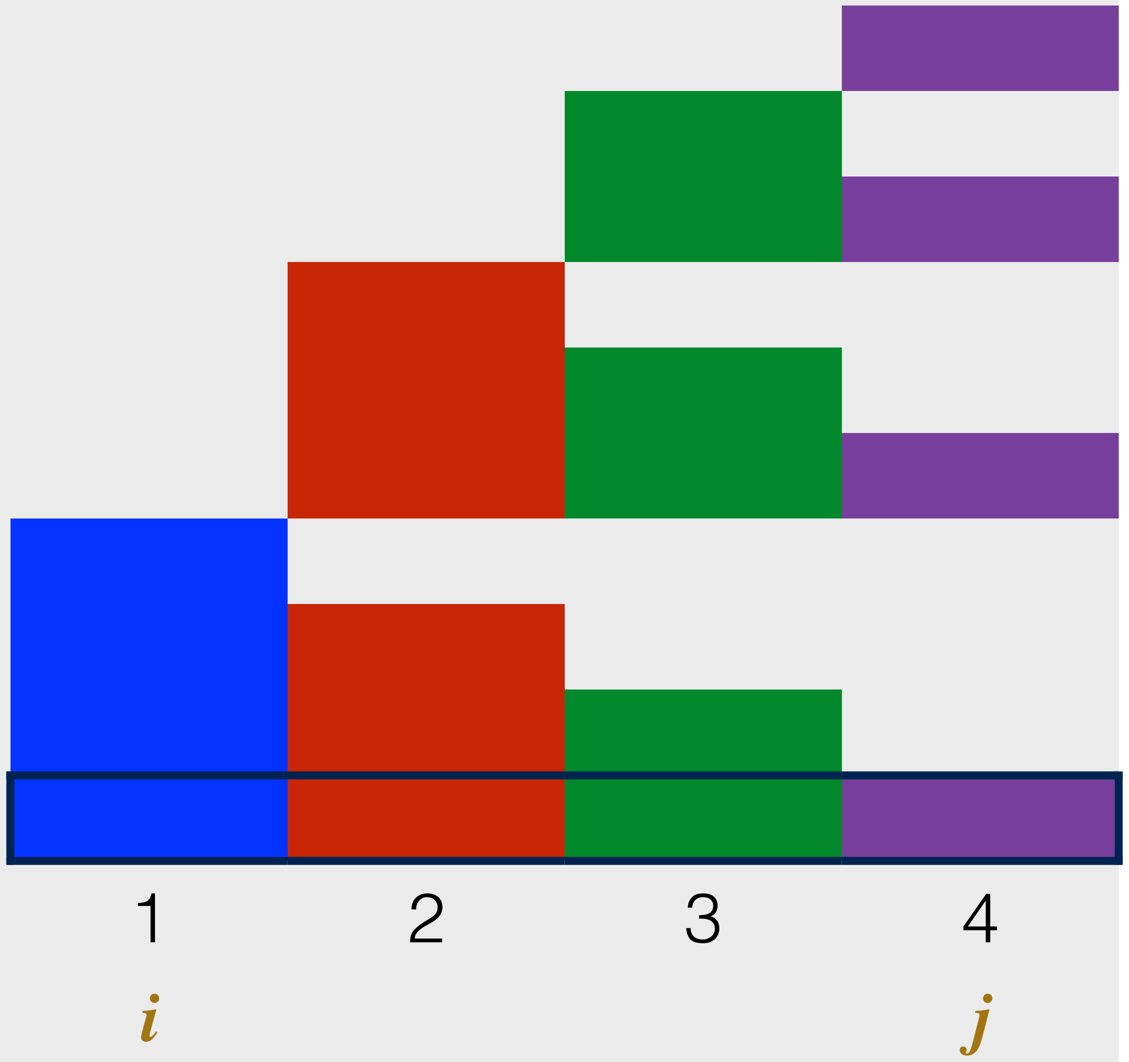


(i,j) = (i,k).(k+1,j)
i=2, k=3, j=4



(i,j) = (1,4)

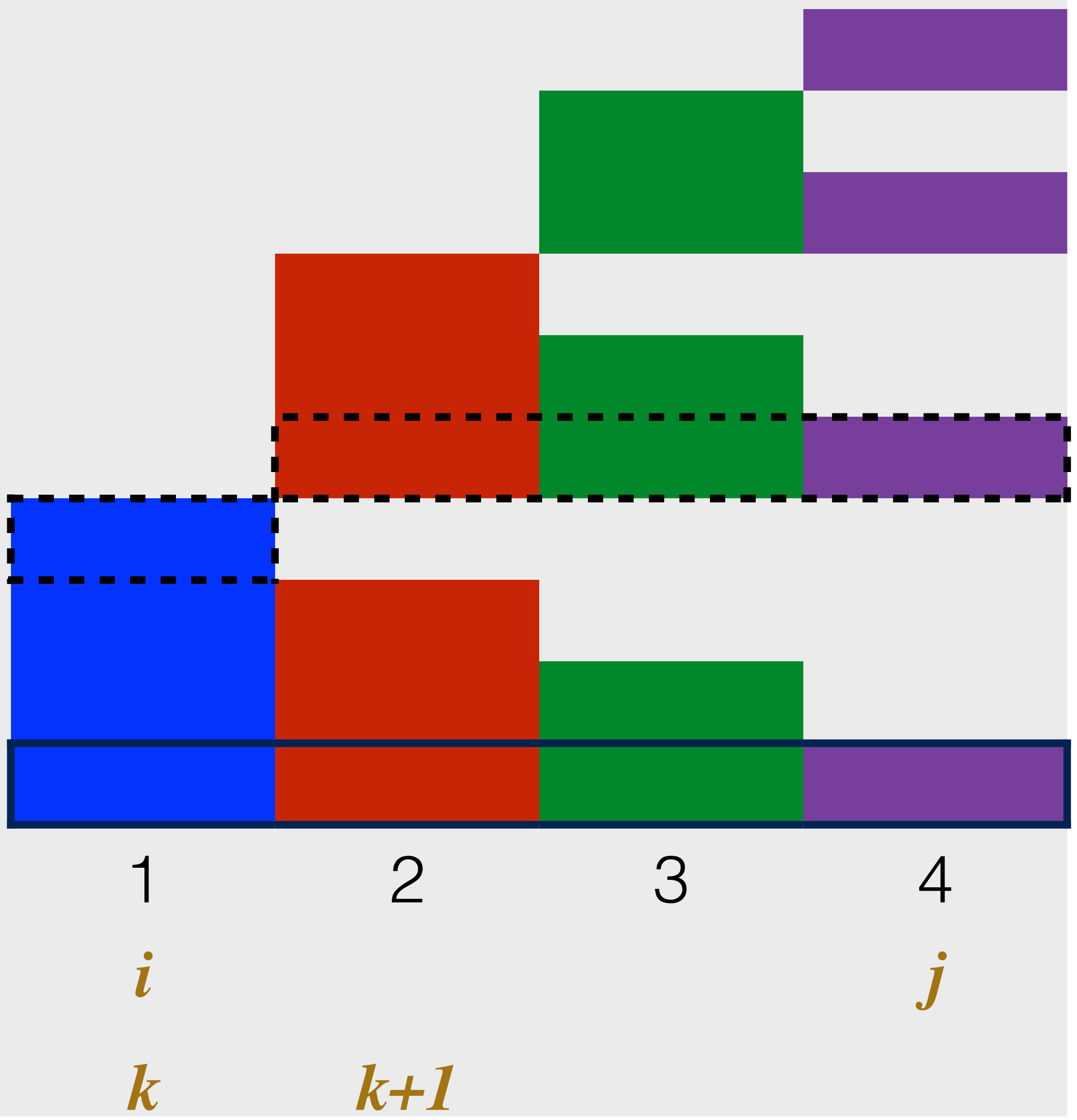
i	j
4	4
3	3
3	4
2	2
2	3
2	4
1	1
1	2
1	3
1	4



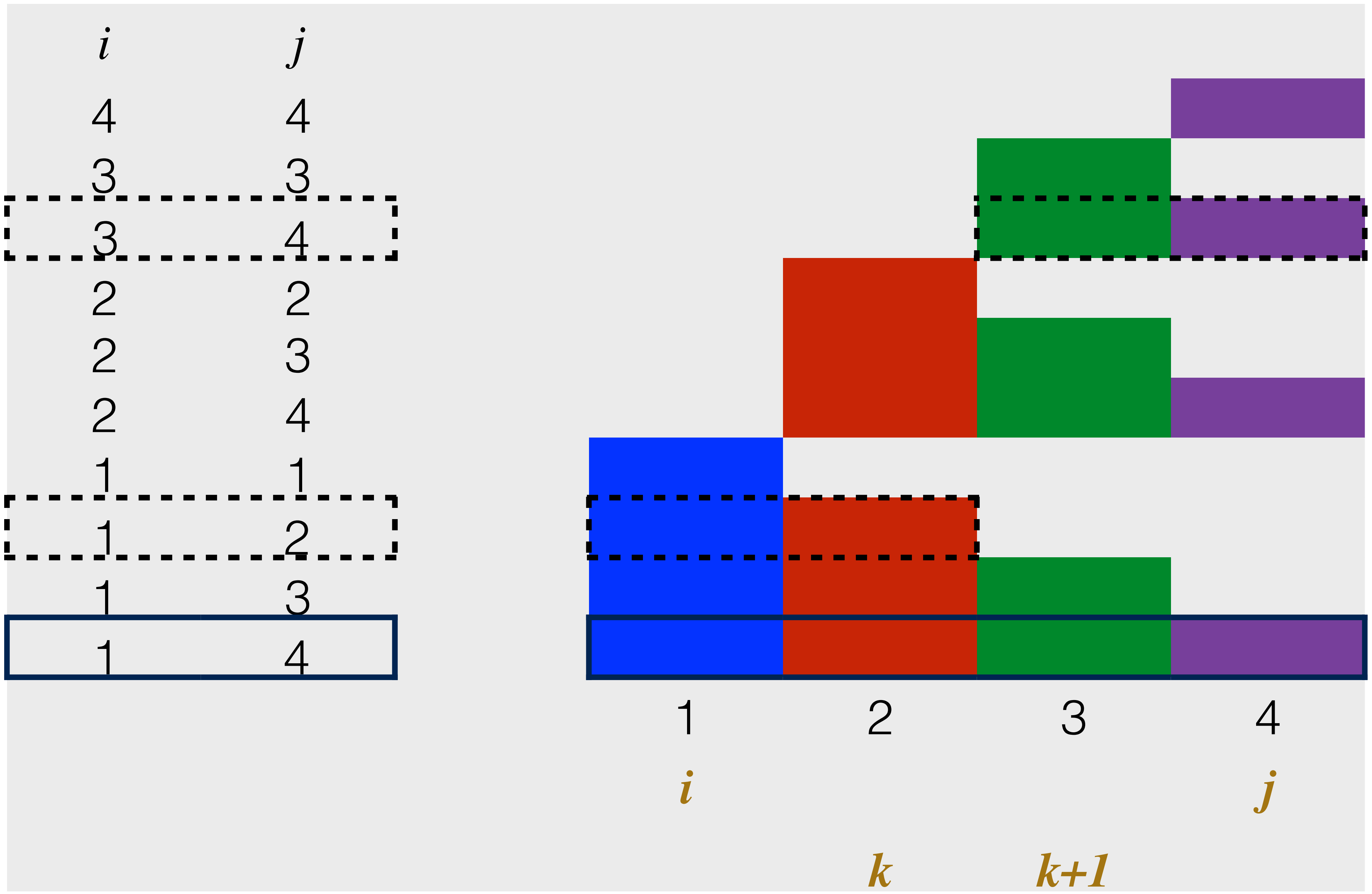
(i,j) = (i,k).(k+1,j)

i=1, k=1, j=4

i	*j*
4	4
3	3
3	4
2	2
2	3
2	4
1	1
1	2
1	3
1	4

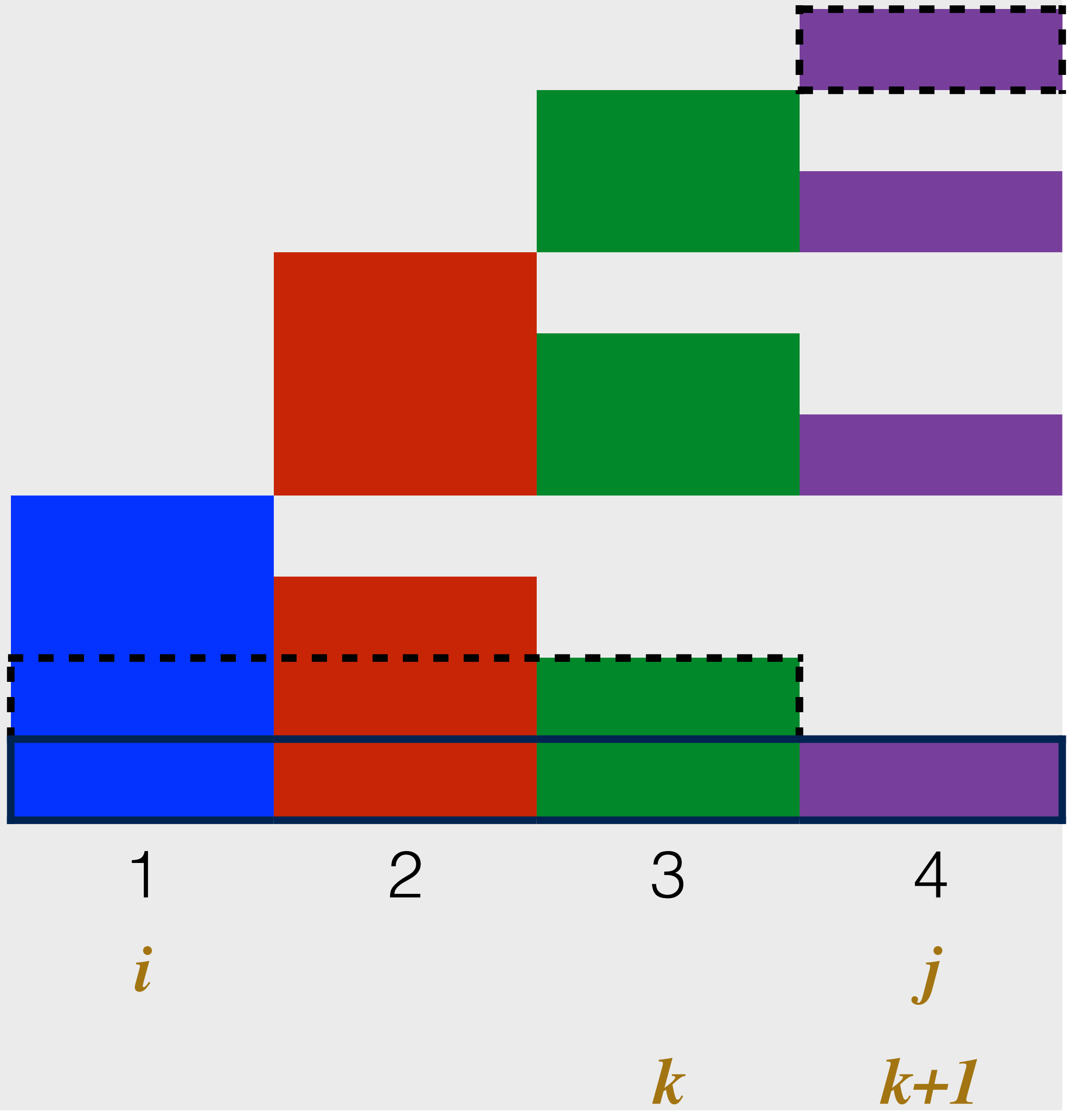


(i,j) = (i,k).(k+1,j)
i=1, k=2, j=4

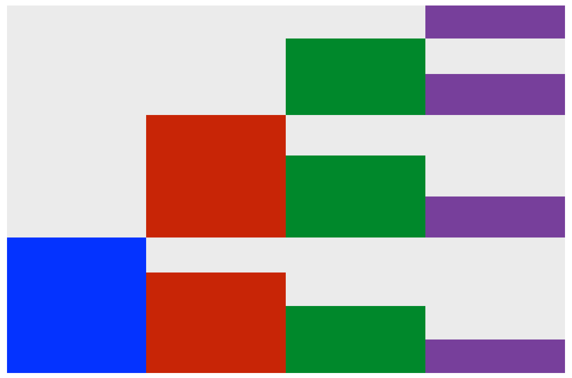


$(i,j) = (i,k).(k+1,j)$
$i=1, k=3, j=4$

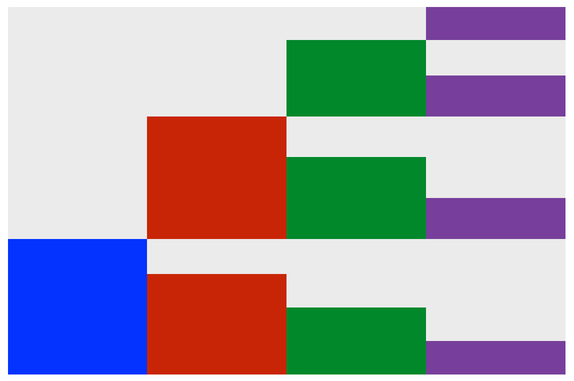
i	j
4	4
3	3
3	4
2	2
2	3
2	4
1	1
1	2
1	3
1	4

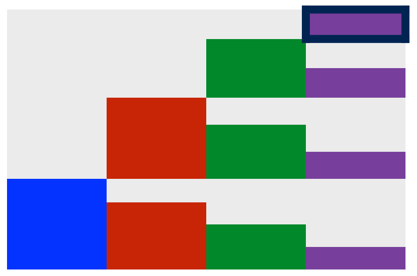


Nussinov fill order



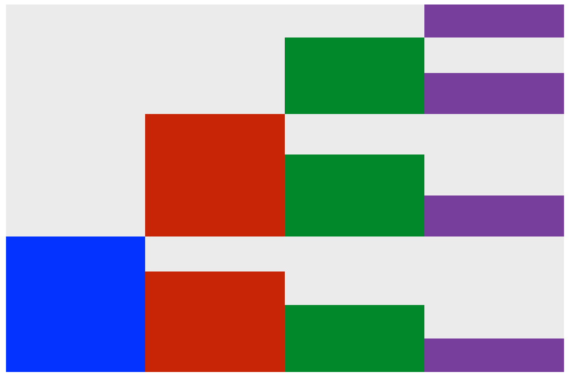
Nussinov fill order

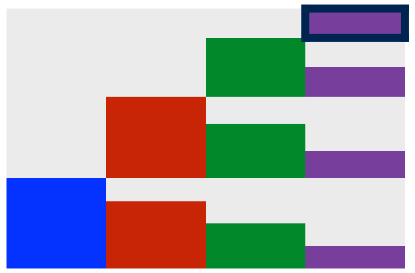




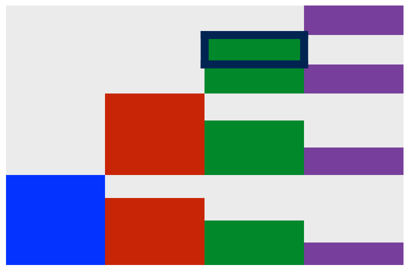
i=4,j=4

Nussinov fill order



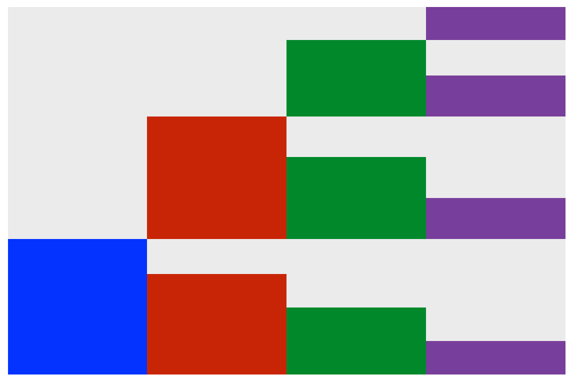


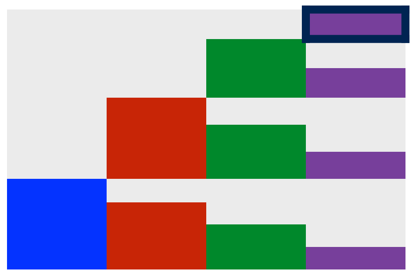
i=4,j=4

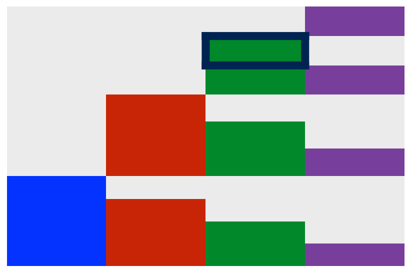


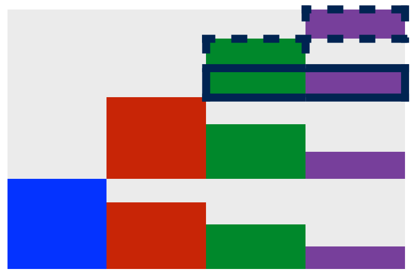
i=3,j=3

Nussinov fill order

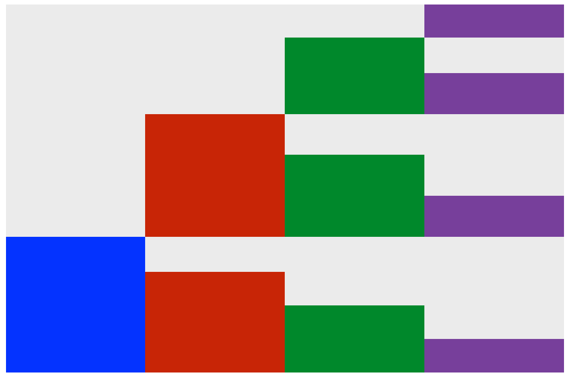


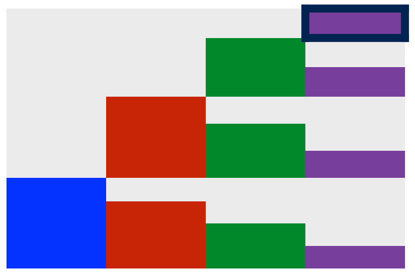

i=4,j=4


i=3,j=3

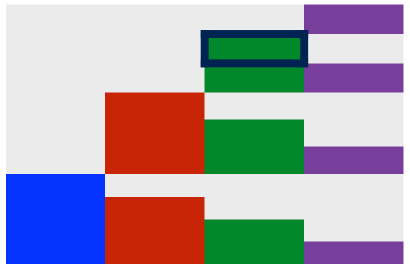

i=3,k=3,j=4

Nussinov fill order

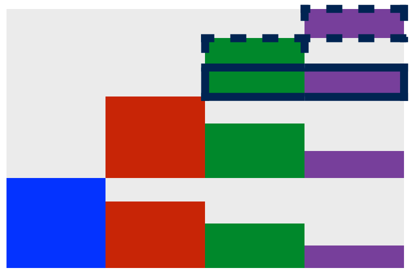




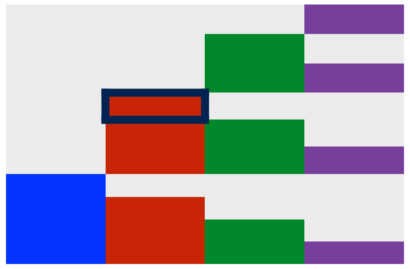
i=4,j=4



i=3,j=3

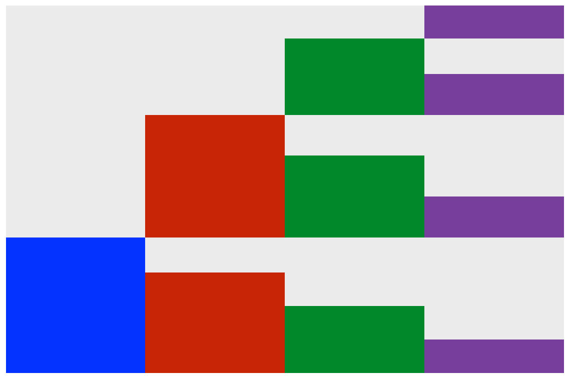


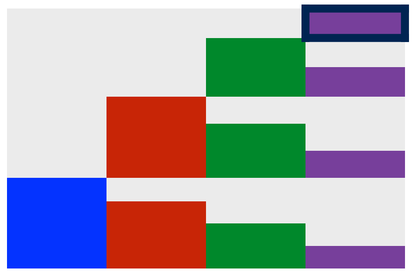
i=3,k=3,j=4

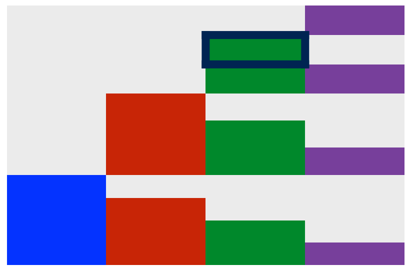


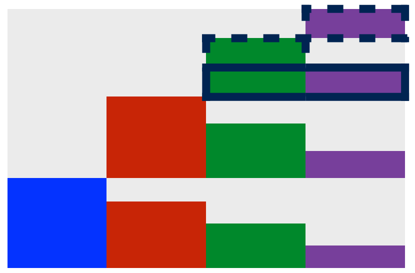
i=2,j=2

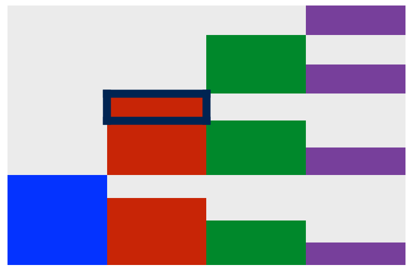
Nussinov fill order

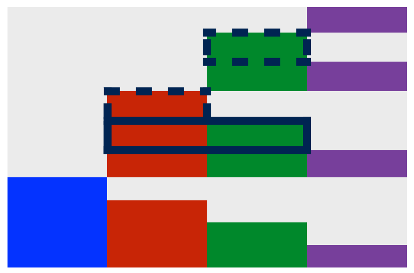



i=4,j=4

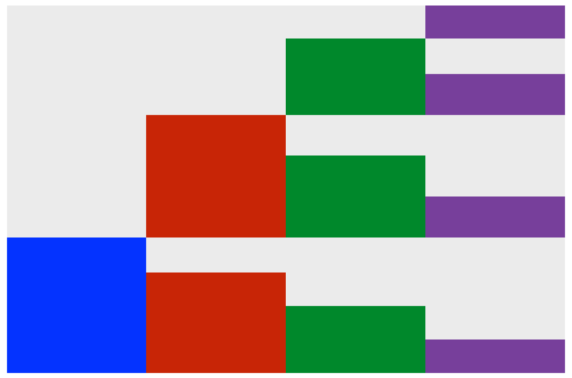

i=3,j=3

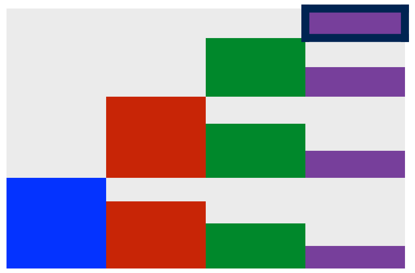

i=3,k=3,j=4

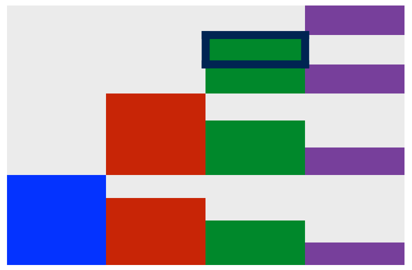

i=2,j=2

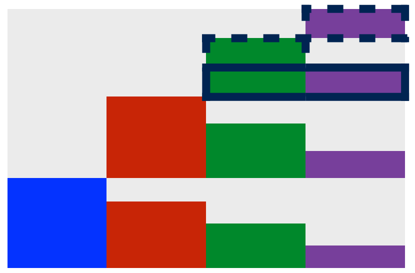

i=2,k=2,j=3

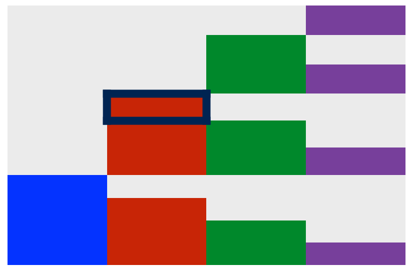
Nussinov fill order

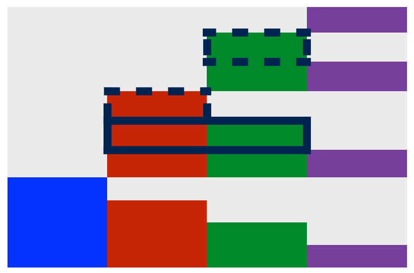


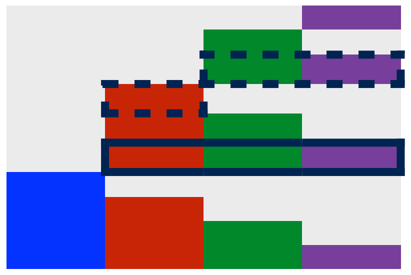

i=4,j=4


i=3,j=3

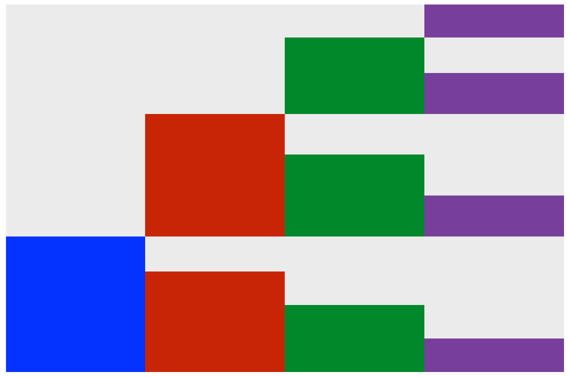

i=3,k=3,j=4

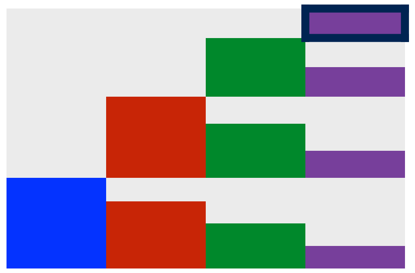

i=2,j=2


i=2,k=2,j=3

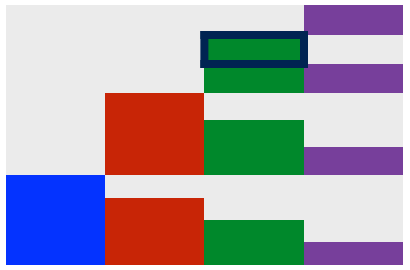

i=2,k=2,j=4

Nussinov fill order

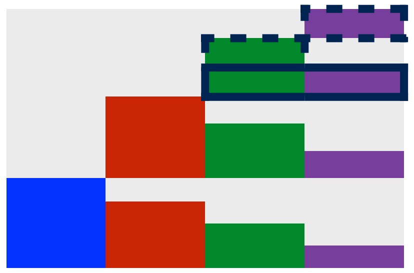




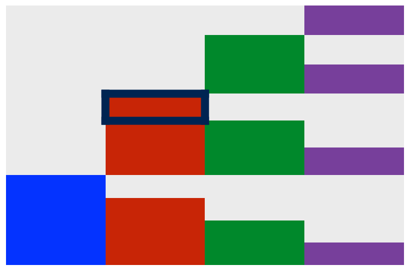
i=4,j=4



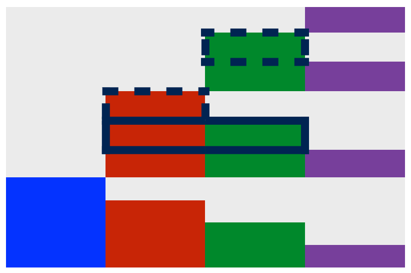
i=3,j=3



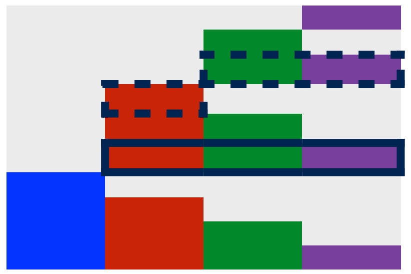
i=3,k=3,j=4



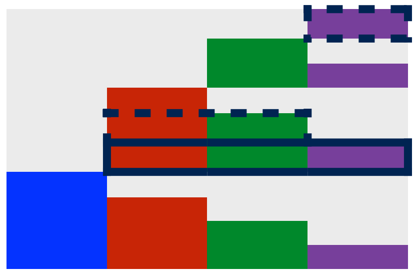
i=2,j=2



i=2,k=2,j=3

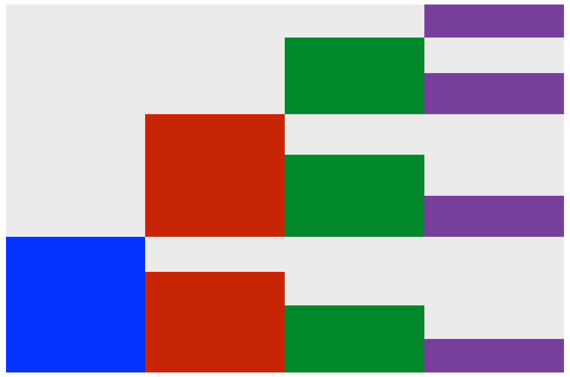


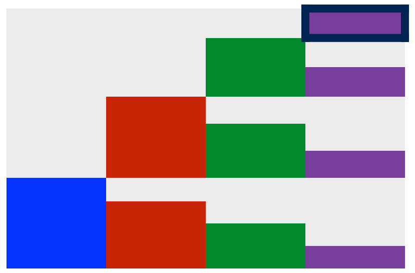
i=2,k=2,j=4

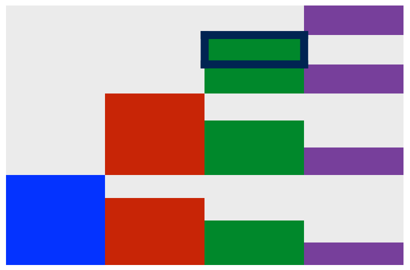


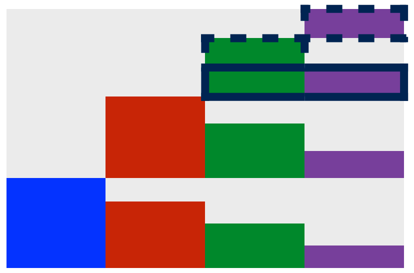
i=2,k=3,j=4

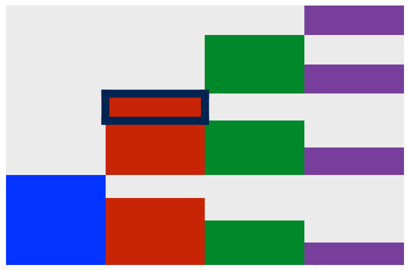
Nussinov fill order

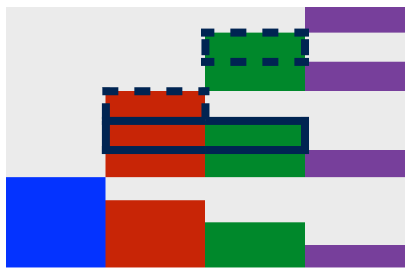


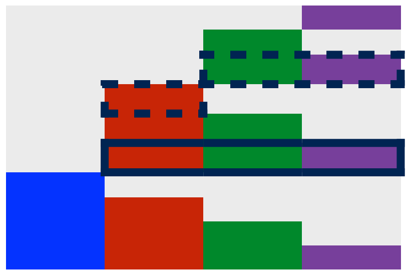

i=4,j=4

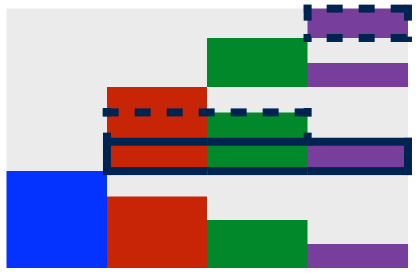

i=3,j=3

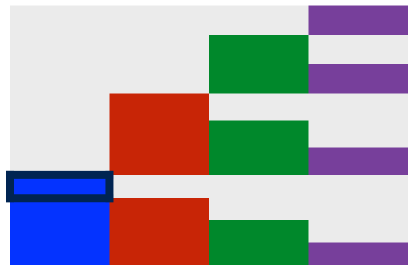

i=3,k=3,j=4


i=2,j=2

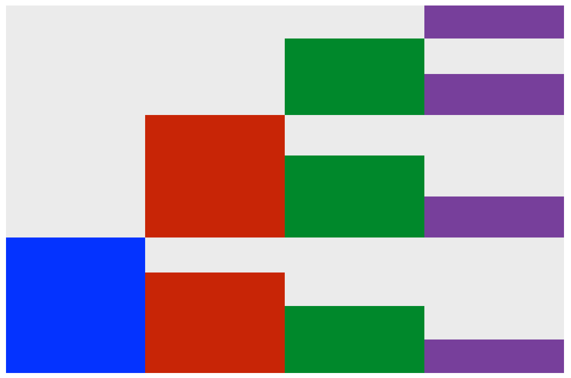

i=2,k=2,j=3

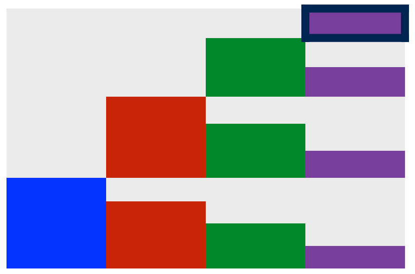

i=2,k=2,j=4


i=2,k=3,j=4

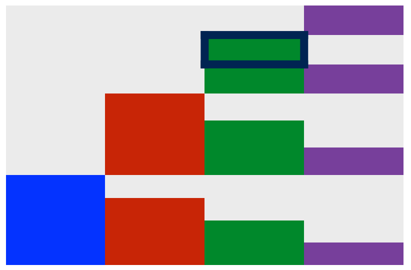

i=1,j=1

Nussinov fill order

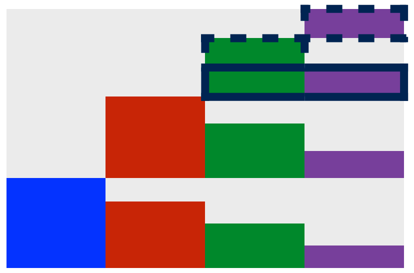




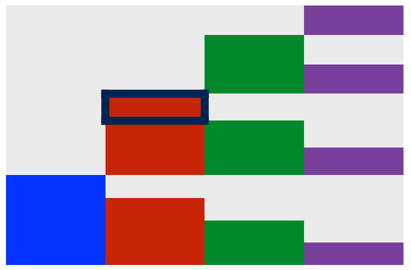
i=4,j=4



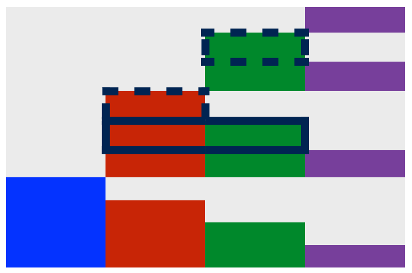
i=3,j=3



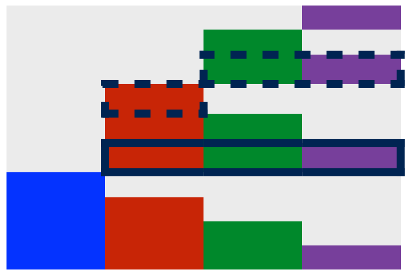
i=3,k=3,j=4



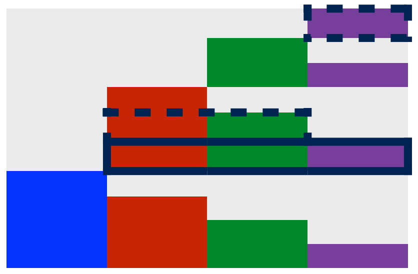
i=2,j=2



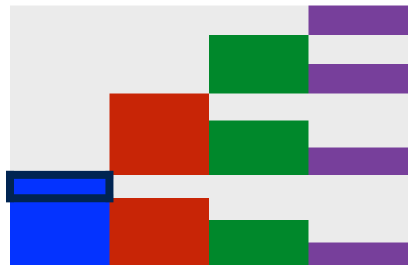
i=2,k=2,j=3



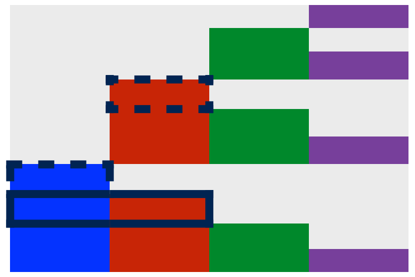
i=2,k=2,j=4



i=2,k=3,j=4

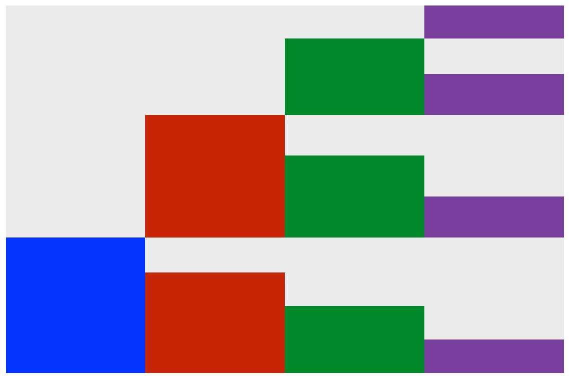


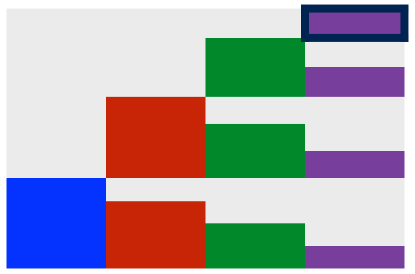
i=1,j=1



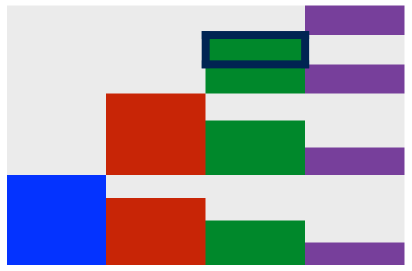
i=1,k=1,j=2

Nussinov fill order

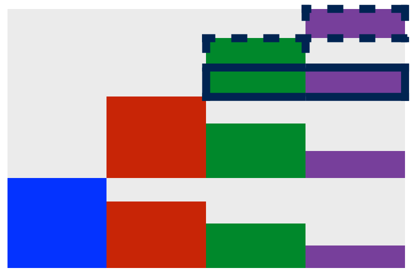




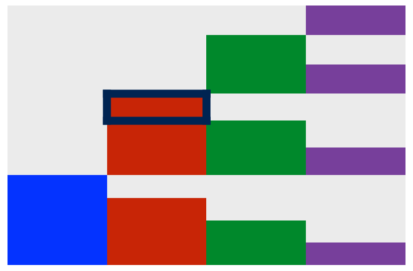
i=4,j=4



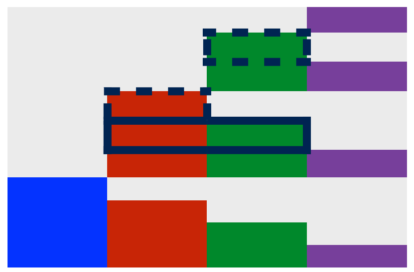
i=3,j=3



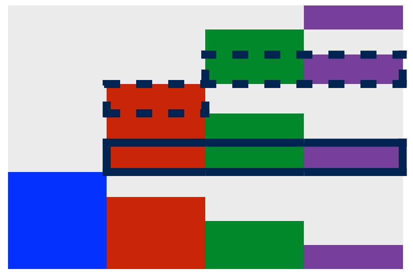
i=3,k=3,j=4



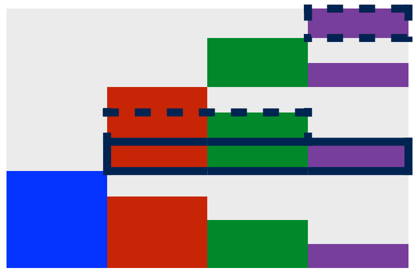
i=2,j=2



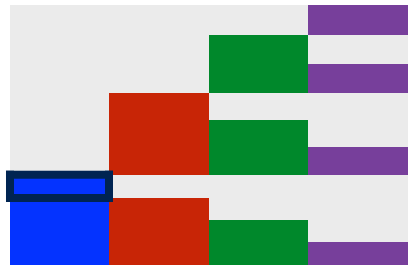
i=2,k=2,j=3



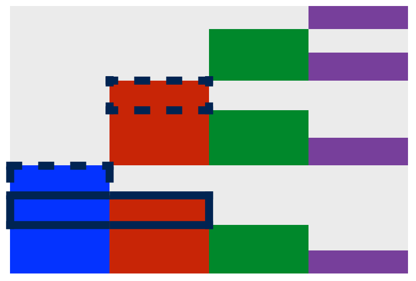
i=2,k=2,j=4



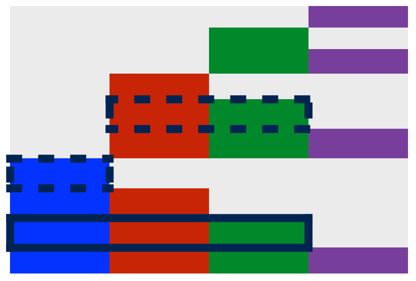
i=2,k=3,j=4



i=1,j=1

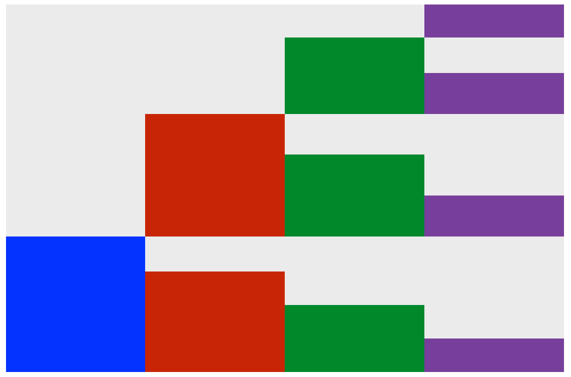


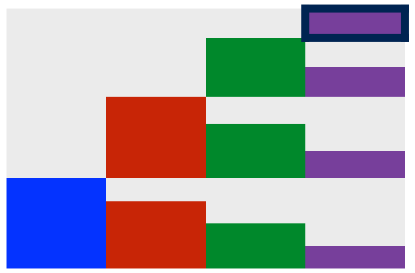
i=1,k=1,j=2

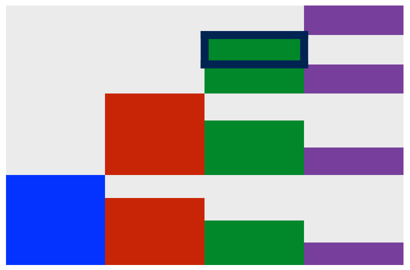


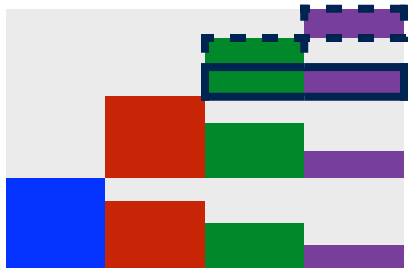
i=1,k=1,j=3

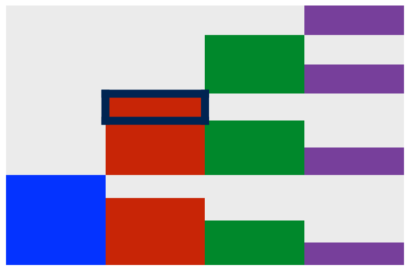
Nussinov fill order

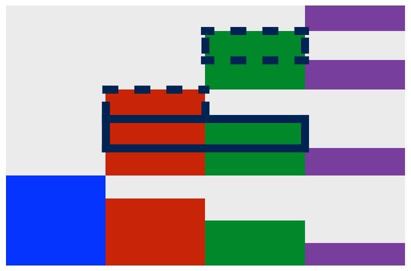


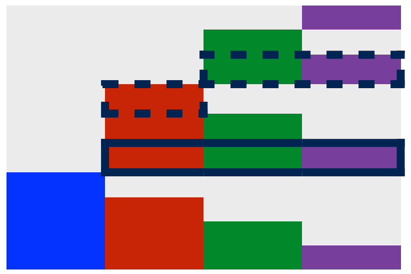

i=4,j=4

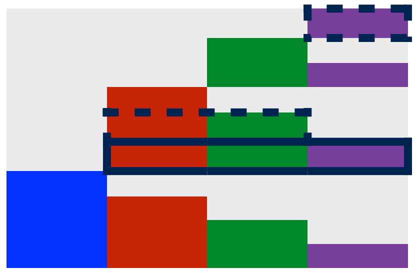

i=3,j=3

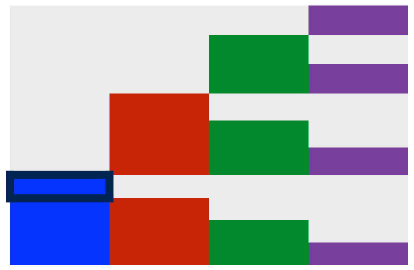

i=3,k=3,j=4

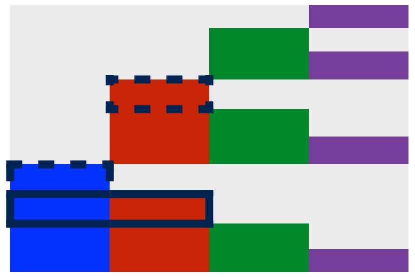

i=2,j=2

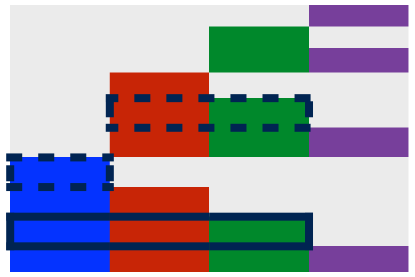

i=2,k=2,j=3

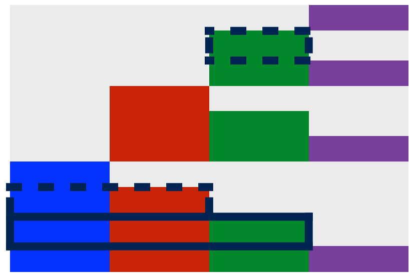

i=2,k=2,j=4


i=2,k=3,j=4

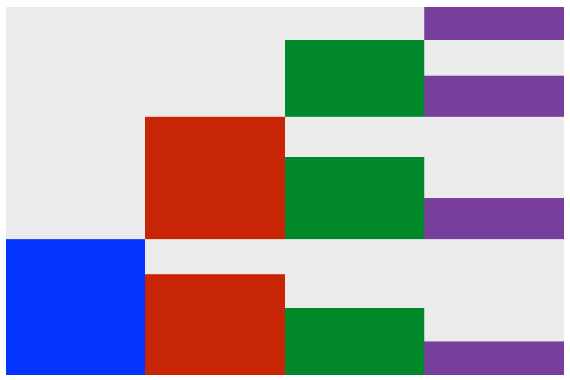

i=1,j=1

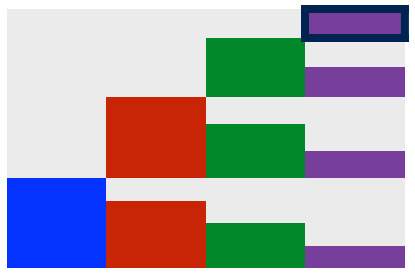

i=1,k=1,j=2

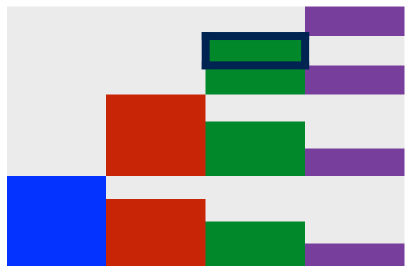

i=1,k=1,j=3

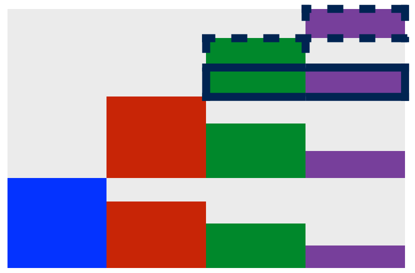

i=1,k=2,j=3

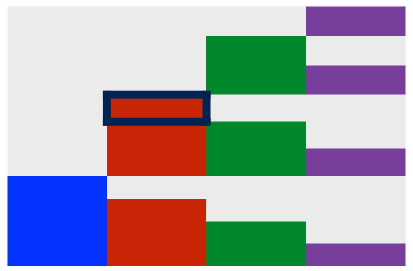
Nussinov fill order

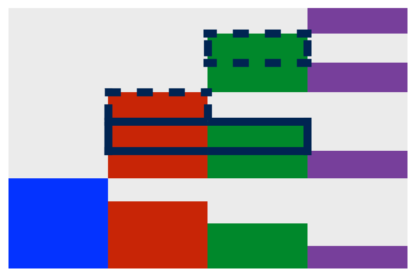


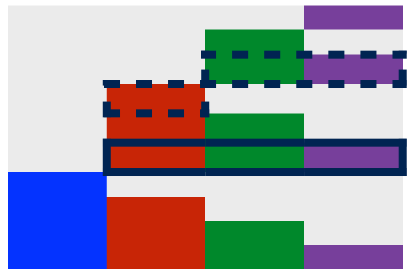

i=4,j=4

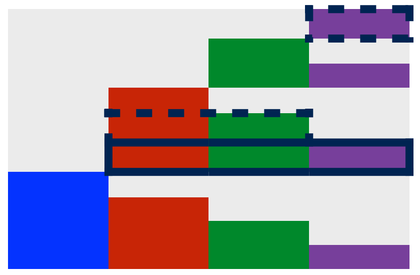

i=3,j=3

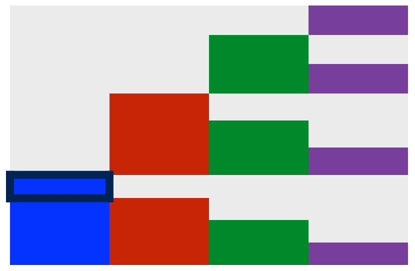

i=3,k=3,j=4

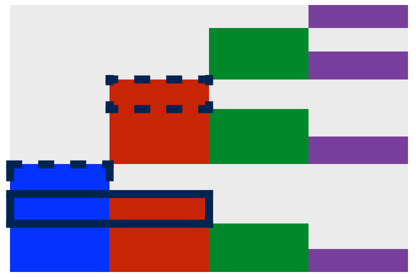

i=2,j=2

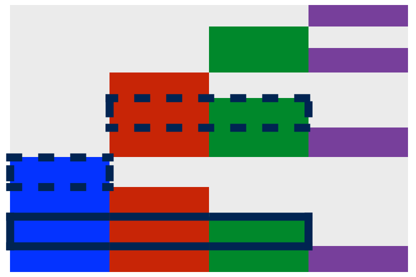

i=2,k=2,j=3

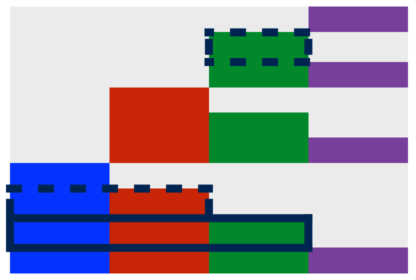

i=2,k=2,j=4

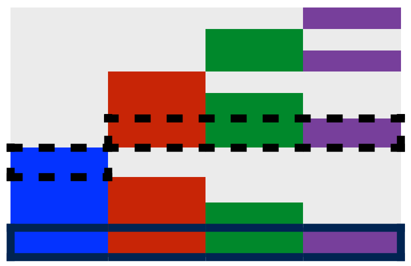

i=2,k=3,j=4


i=1,j=1

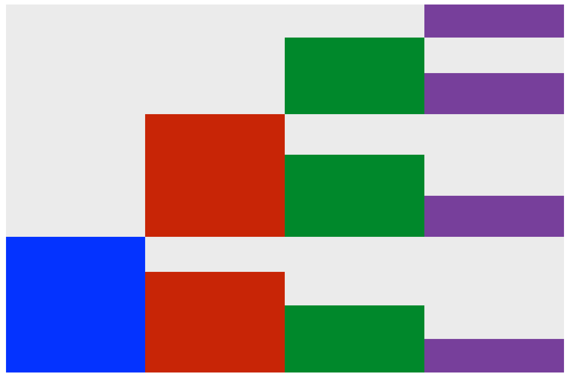

i=1,k=1,j=2

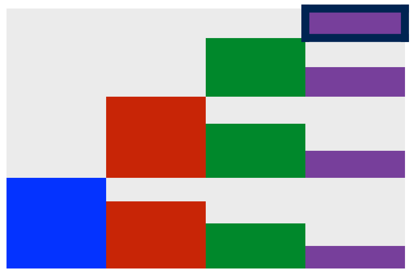

i=1,k=1,j=3

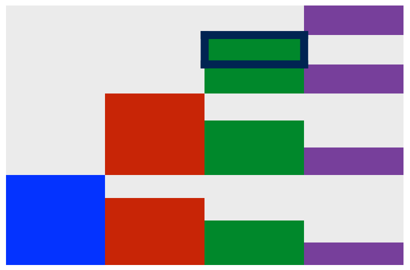

i=1,k=2,j=3

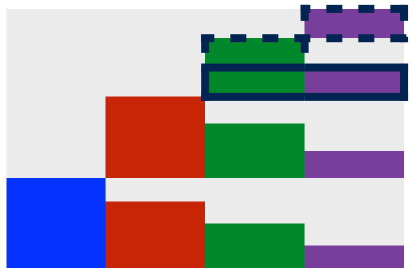

i=1,k=1,j=4

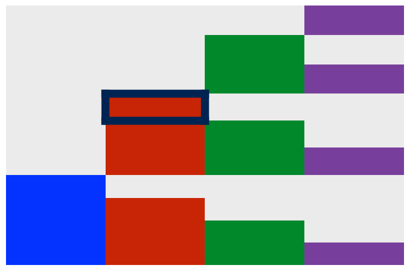
Nussinov fill order

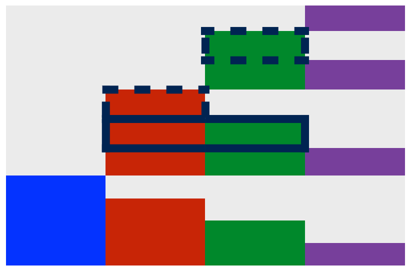


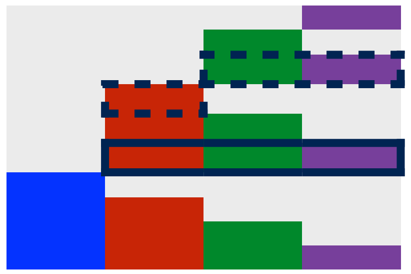

i=4,j=4

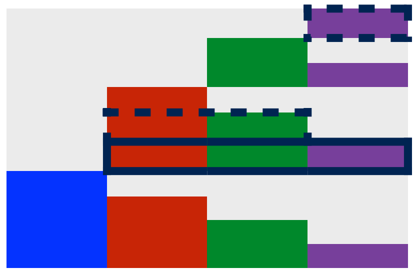

i=3,j=3

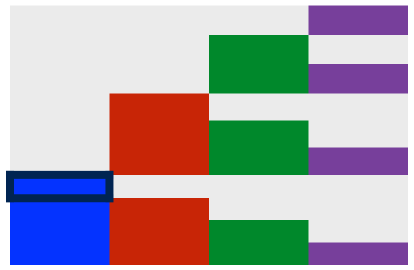

i=3,k=3,j=4

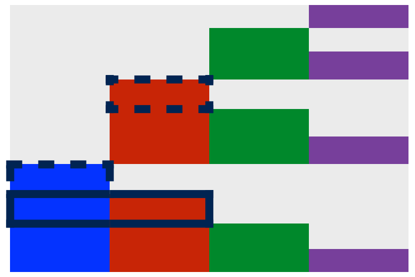

i=2,j=2

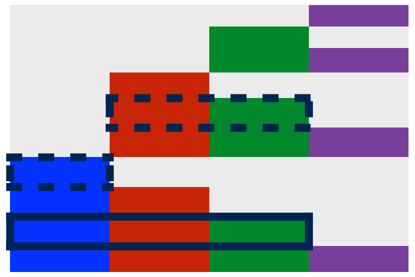

i=2,k=2,j=3

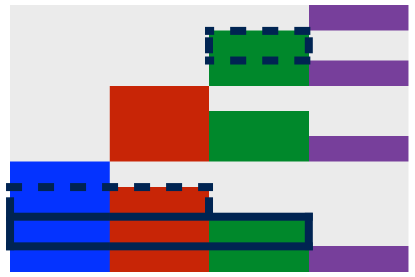

i=2,k=2,j=4

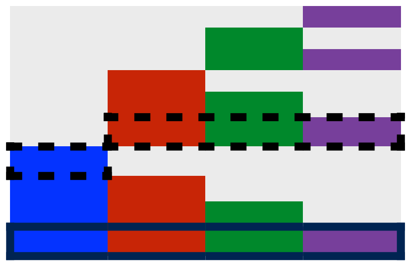

i=2,k=3,j=4

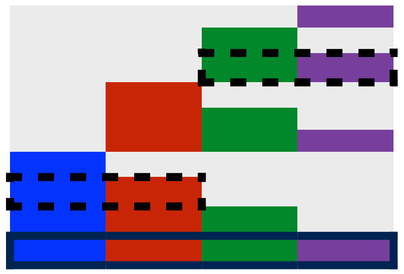

i=1,j=1


i=1,k=1,j=2

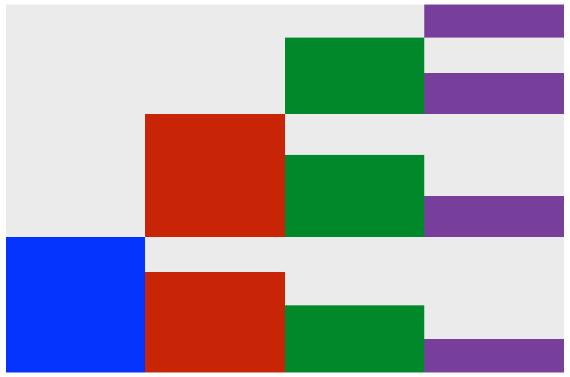

i=1,k=1,j=3

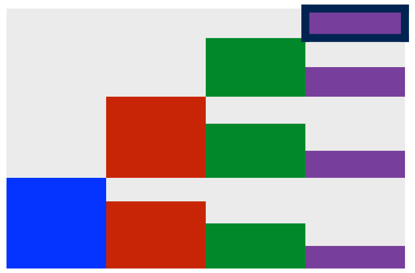

i=1,k=2,j=3

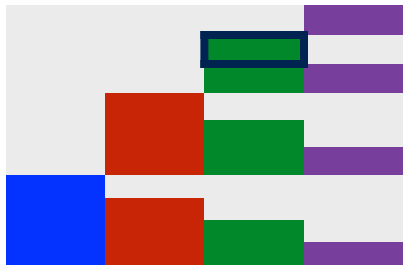

i=1,k=1,j=4

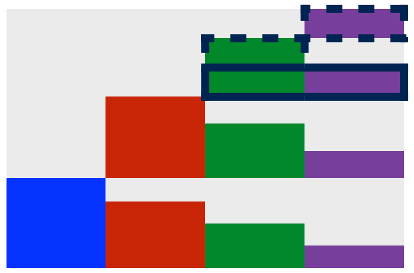

i=1,k=2,j=4

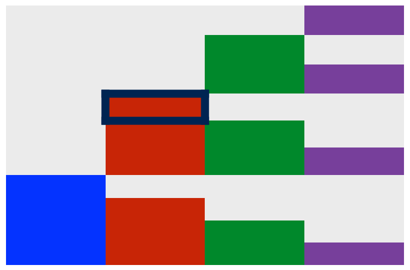
Nussinov fill order

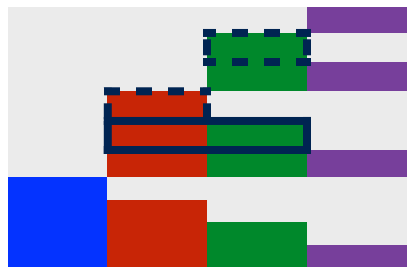


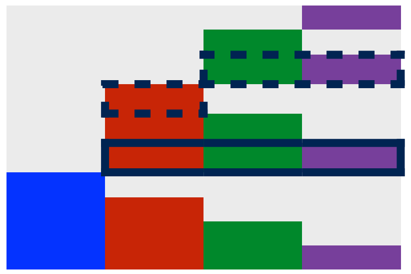

i=4,j=4

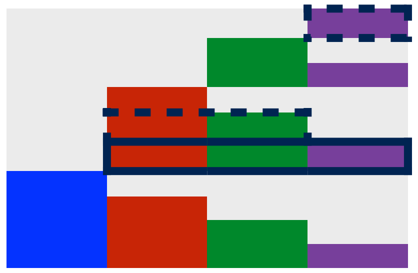

i=3,j=3

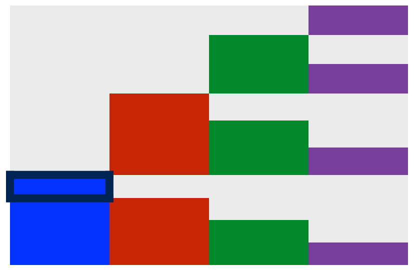

i=3,k=3,j=4

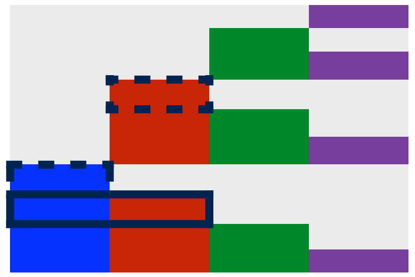

i=2,j=2

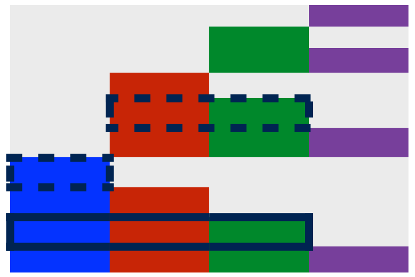

i=2,k=2,j=3

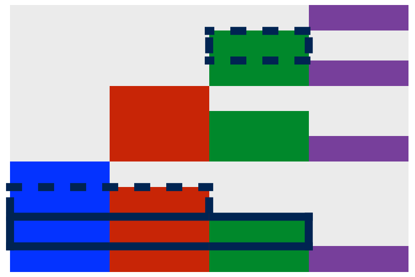

i=2,k=2,j=4

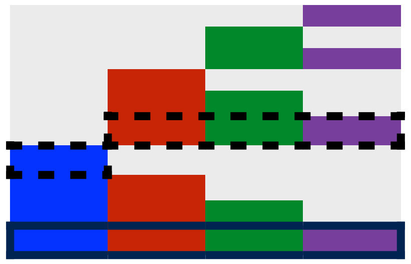

i=2,k=3,j=4

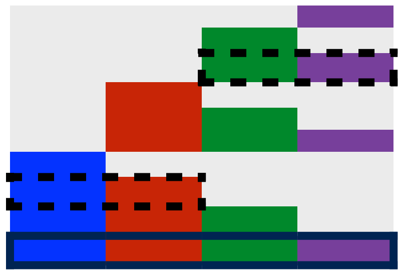

i=1,j=1

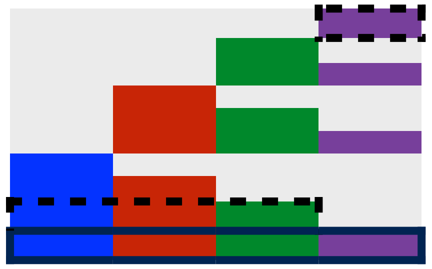

i=1,k=1,j=2


i=1,k=1,j=3

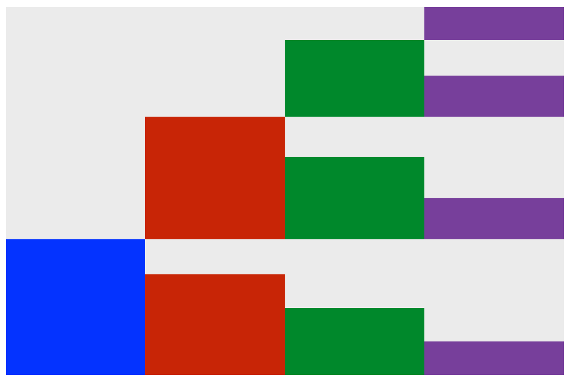

i=1,k=2,j=3

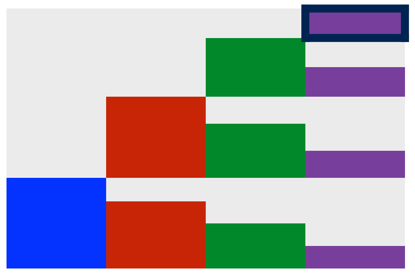

i=1,k=1,j=4

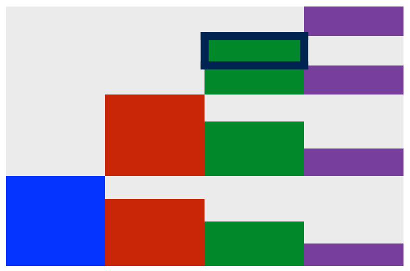

i=1,k=2,j=4

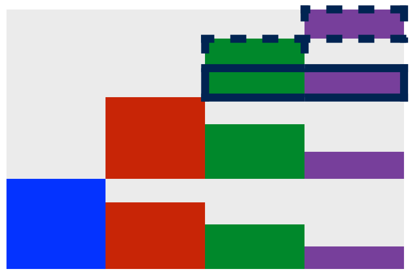

i=1,k=1,j=4

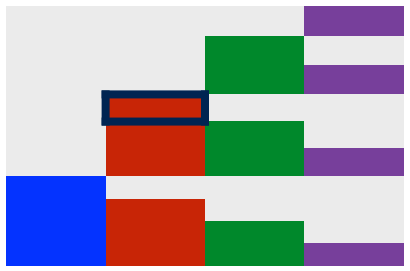
Nussinov fill order

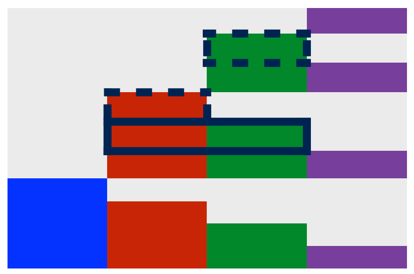


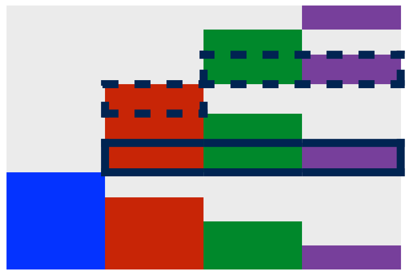

i=4,j=4

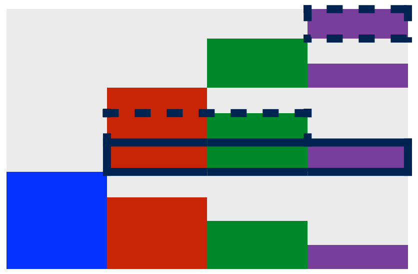

i=3,j=3

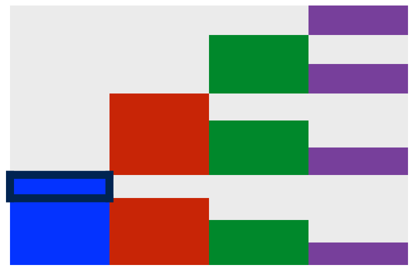

i=3,k=3,j=4

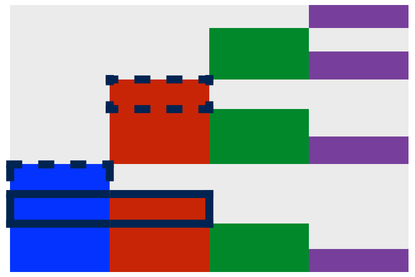

i=2,j=2

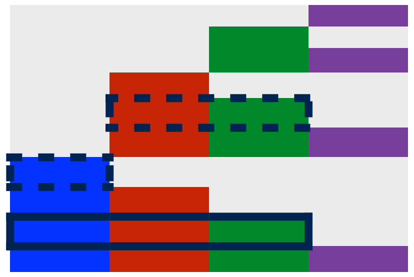

i=2,k=2,j=3

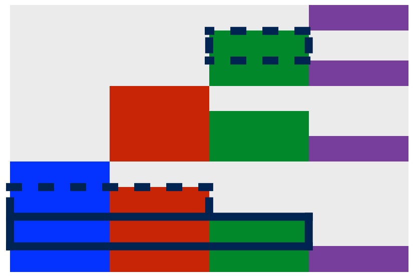

i=2,k=2,j=4

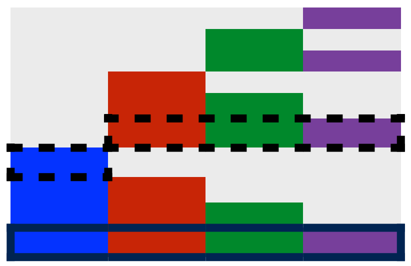

i=2,k=3,j=4

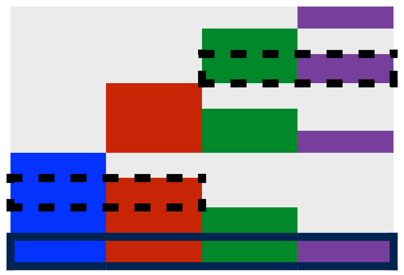

i=1,j=1

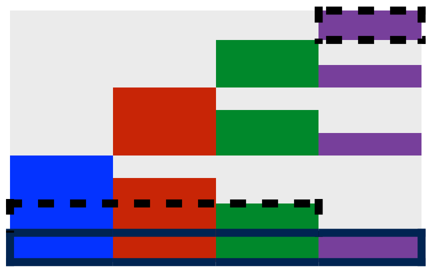

i=1,k=1,j=2


i=1,k=1,j=3


i=1,k=2,j=3

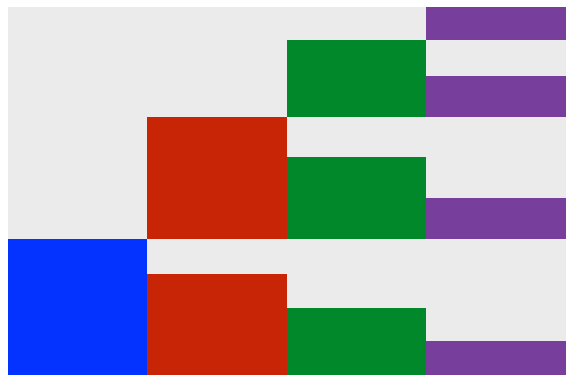

i=1,k=1,j=4

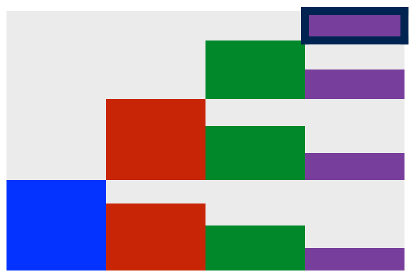

i=1,k=2,j=4

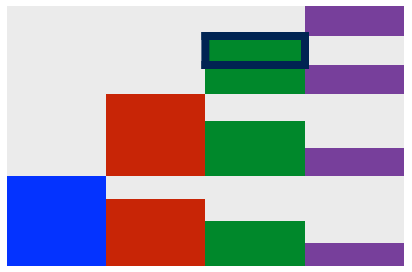

i=1,k=1,j=4

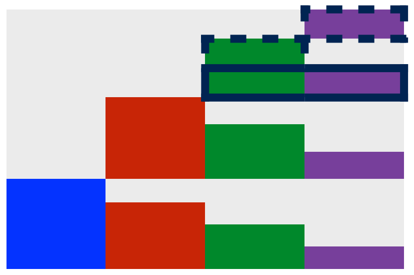
- For i = L to 1 (decreasing)
    - For j = i to L (increasing)
        - For k = i to j-1 (increasing)

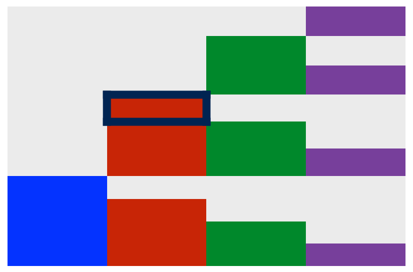
Nussinov fill order

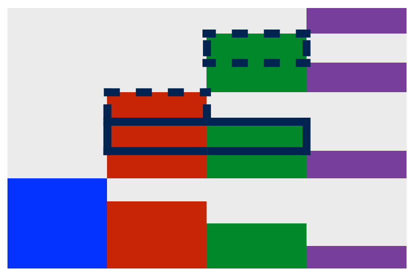


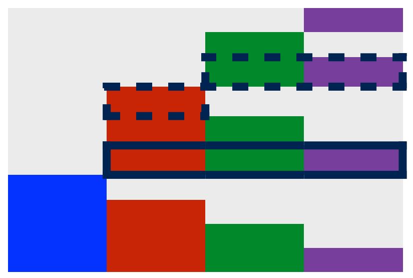

i=4,j=4

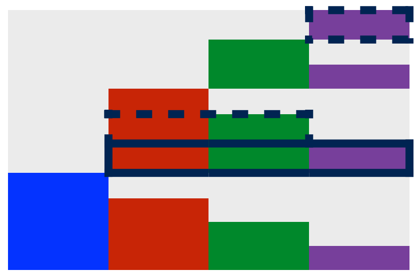

i=3,j=3

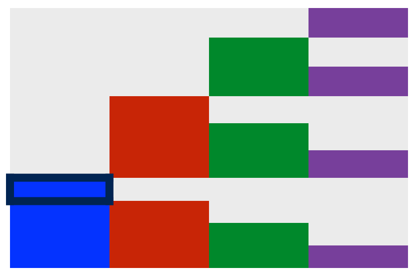

i=3,k=3,j=4

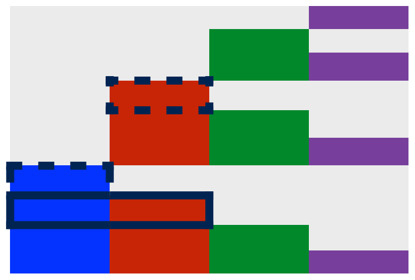

i=2,j=2

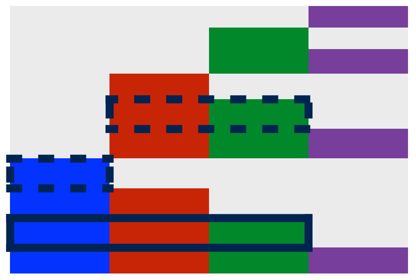

i=2,k=2,j=3

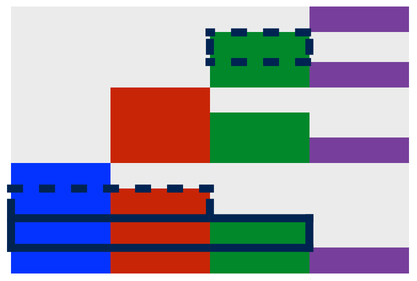

i=2,k=2,j=4

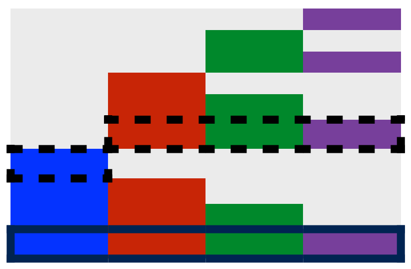

i=2,k=3,j=4

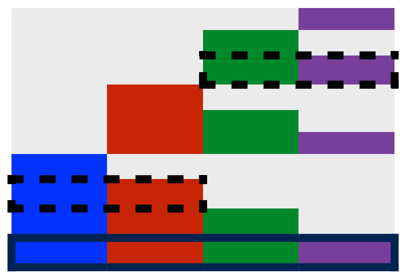

i=1,j=1

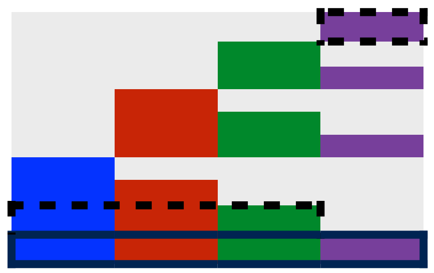

i=1,k=1,j=2


i=1,k=1,j=3


i=1,k=2,j=3


i=1,k=1,j=4


i=1,k=2,j=4


i=1,k=1,j=4

- For i = L to 1 (decreasing)
  - For j = i to L (increasing)
    - For k = i to j-1 (increasing)

$$N_{ij} = \max \begin{pmatrix} N_{i+1,j-1} + \delta_{WC}(X_i, X_j) \\ N_{i+1,j} \\ N_{i,j-1} \\ \max_{i<k<j}(N_{ik} + N_{k+1,j}) \end{pmatrix} \text{(if } j > i)$$

$$N_{ii} = 0$$

Nussinov fill

$$N_{ij} = \max\begin{pmatrix} N_{i+1,j-1} + \delta_{WC}(X_i, X_j) \\ N_{i+1,j} \\ N_{i,j-1} \\ \max_{i<k<j}(N_{ik} + N_{k+1,j}) \end{pmatrix} \text{(if } j > i\text{)}$$

$$N_{ii} = 0$$

```
For i = L to 1 (decreasing)
    For j = i to L (increasing)
        N(i,j) ← 0
        N(i,j) ← max ( N(i,j),  N(i+1,j-1) + $\partial_{WC}(X_i,X_j)$ )
        N(i,j) ← max ( N(i,j),  N(i+1,j) )
        N(i,j) ← max ( N(i,j),  N(i,j-1) )
        For k = i to j-1
            N(i,j) ← max ( N(i,j), N(i,k) + N(k+1,j) )
```

Nussinov fill

$$N_{ij} = \max \begin{pmatrix} N_{i+1,j-1} + \delta_{WC}(X_i, X_j) \\ N_{i+1,j} \\ N_{i,j-1} \\ \max_{i<k<j}(N_{ik} + N_{k+1,j}) \end{pmatrix} \text{(if } j > i\text{)}$$

$$N_{ii} = 0$$

```
• For i = L to 1 (decreasing)
    • For j = i to L (increasing)
        • N(i,j) ← 0
        • N(i,j) ← max ( N(i,j),  N(i+1,j-1) + $\partial_{WC}(X_i,X_j)$ )
        • N(i,j) ← max ( N(i,j),  N(i+1,j) )
        • N(i,j) ← max ( N(i,j),  N(i,j-1) )
        • For k = i to j-1
            • N(i,j) ← max ( N(i,j), N(i,k) + N(k+1,j) )
```

Dynamic programming

Nussinov traceback

- If you know how to calculate $N(i,j)$ then you also know whether (i,j) is a basepair
- Start with $N(1,L)$ and work inwards
  - Traceback through bifurcations requires a stack (or, equivalently, recursion)

Questions…

- How long does the Nussinov algorithm take to run (as a function of the input sequence lengths)?
- How much memory does it use?

$$N_{ij} = \max \begin{pmatrix} N_{i+1,j-1} + \delta_{WC}(X_i, X_j) \\ N_{i+1,j} \\ N_{i,j-1} \\ \max_{i<k<j}(N_{ik} + N_{k+1,j}) \end{pmatrix} (\text{if } j > i)$$

$$N_{ii} = 0$$

Algorithmic complexity

- Resource usage (**time**, **memory**, bandwidth, ...)
- Often more interested in asymptotic behavior (large datasets) than fine detail
- Asymptotic behavior describes scaling and order-of-magnitude differences

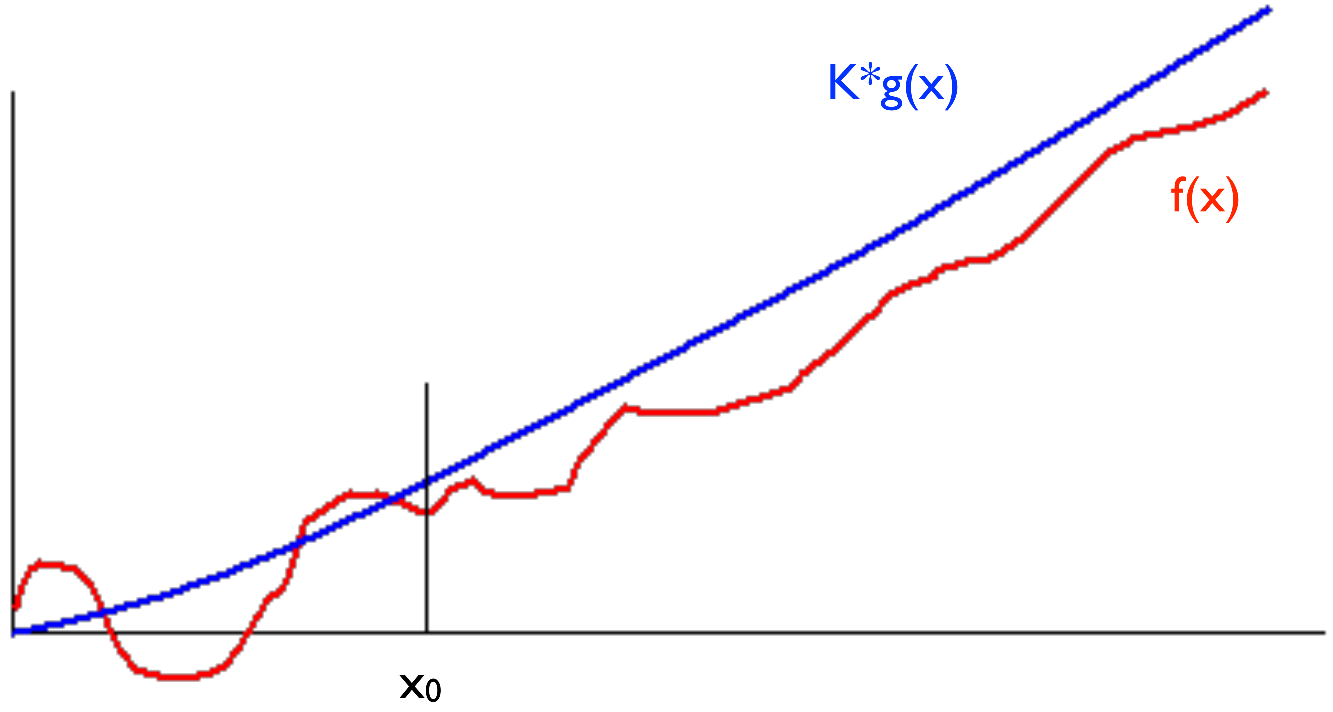
Big O notation

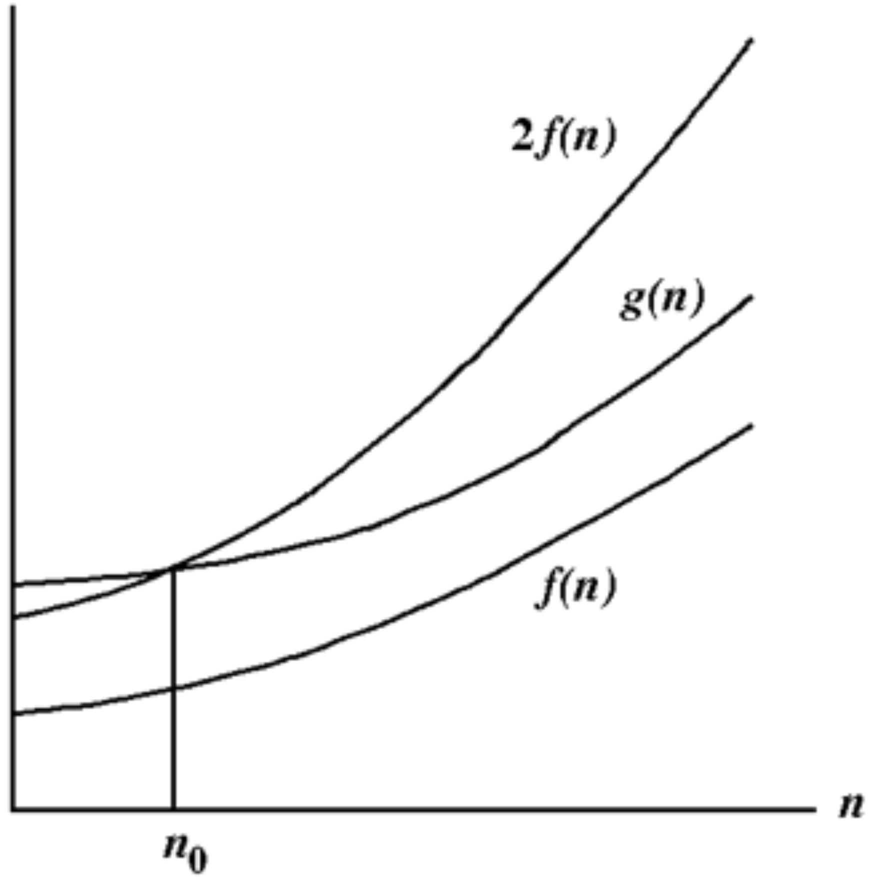
- In discussing the resource usage of algorithms, it is often useful to consider *asymptotic* behavior (e.g. on very large datasets), rather than every detail
- For example, suppose you have N items and you want to compare every item to every other item.
  This involves $N(N-1)/2$ comparisons, but it's often more illuminating to say it involves "of order N^2" comparisons. Or, more technically, $O(N^2)$ comparisons.
- Formally, a function $f(x)$ is said to be $O(g(x))$ if for large enough x ($x>x_0$) there is some constant K such that
  $f(x) < K*g(x)$ for all $x>x_0$

Big O notation

- Formally, we say $f(x)=O(g(x))$ if for large enough x ($x>x0$) there is some constant K such that $f(x) < K*g(x)$
- Example: if $f(x)$ is a polynomial, $f(x)=a+bx+cx^2+...+kx^m$ then $f(x)=O(x^m)$ **... only the highest power matters**
- Asymptotically (for large x), $f(x)$ will be dominated by the x^m term. For questions like "how does $f(x)$ change when we double x", the coefficient k does not matter.
  - In practice, the coefficient k, as well as the lower powers of x, might matter quite a lot. But in some sense they're "details"...
- Note that some functions can't be written as finite polynomials (e.g. $log(x)$, $exp(x)$, ...)

- Formally, a function $f(x)$ is said to be $O(g(x))$ if for large enough x ($x>x_0$) there is some constant K such that
  $f(x) < K*g(x)$ for all $x>x_0$





It comes down to **how fast the function f(x) grows for large x**

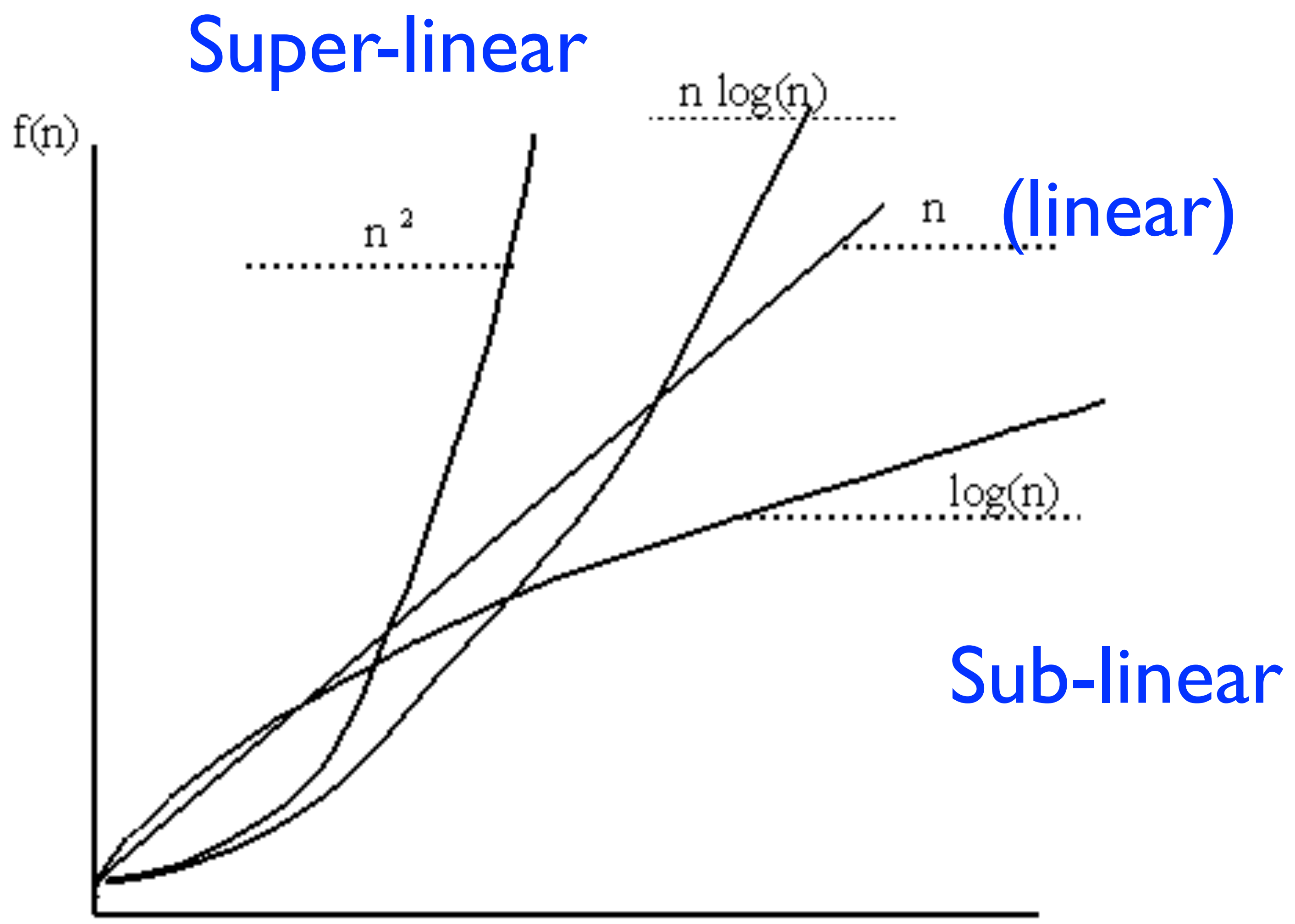
Big O notation

- We are often specifically interested in using Big O notation to describe...
    - *Runtime complexity* - the time an algorithm takes to run (note that we can only specify this to within a scaling coefficient that depends on the CPU clock rate of the hardware; Big O notation removes this scaling coefficient from the picture)
    - *Memory complexity* - the amount of RAM that an algorithm uses (again, this depends on hardware details - do we use 8, 16, 32 or 64 bits to store an integer? Big O notation saves us from such details)

Simplification rules

- If $f(x)$ is a sum of several terms, the one with the largest growth rate is kept, and all others omitted.
- If $f(x)$ is a product of several factors, any constants (terms in the product that do not depend on x) are omitted.

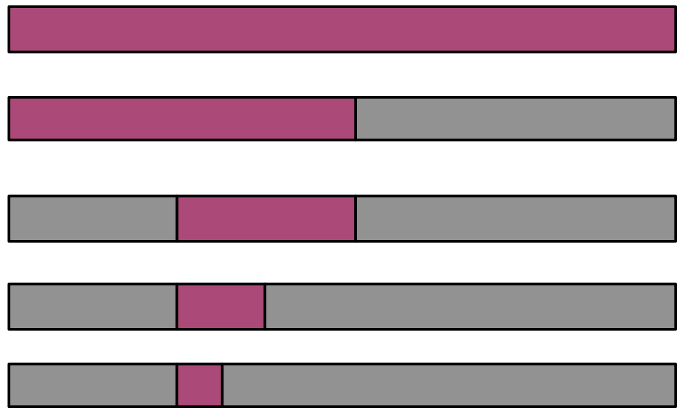
Some common Big-O formulae



exp(n) grows faster than any power n^k

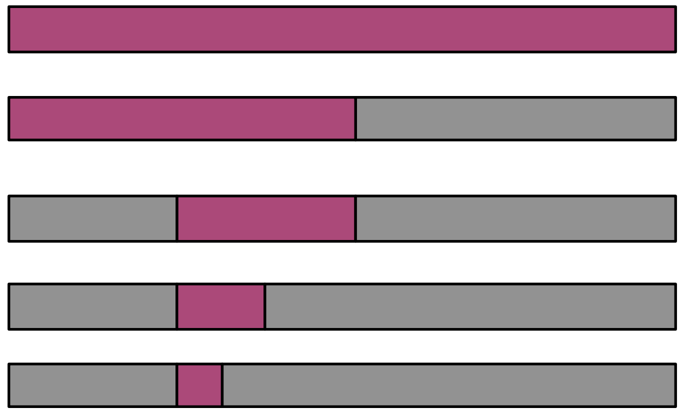
n! grows even faster

Sorting and complexity



To **find** an item in a sorted list of N items takes $\log_2(N)$ steps (“*binary search*” algorithm)

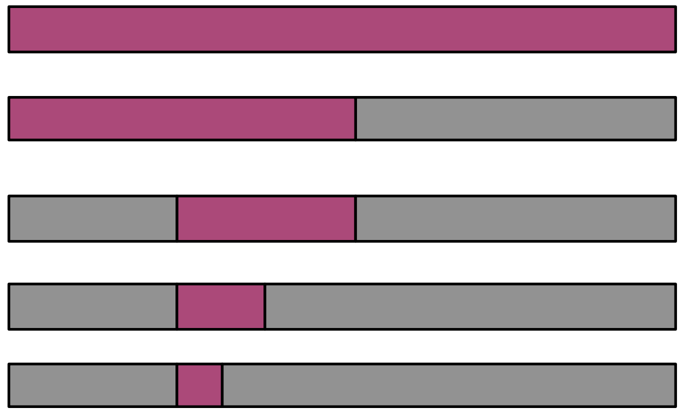
Sorting and complexity



To **find** an item in a sorted list of N items takes $\log_2(N)$ steps (*"binary search"* algorithm)

Similarly, to find the correct place to **insert** a new item takes $\log_2(N)$ steps

Sorting and complexity

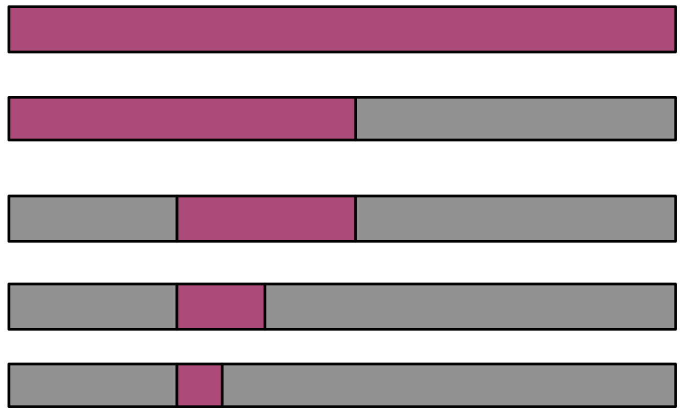


To **find** an item in a sorted list of N items takes $\log_2(N)$ steps (*"binary search"* algorithm)

Similarly, to find the correct place to **insert** a new item takes $\log_2(N)$ steps

To place N items correctly takes of order $N*\log_2(N)$ steps

Sorting and complexity



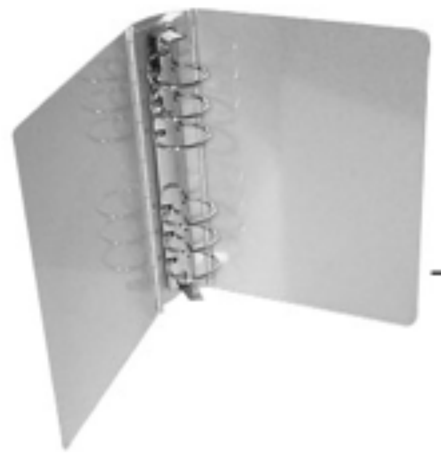
To **find** an item in a sorted list of N items takes $\log_2(N)$ steps (*"binary search"* algorithm)

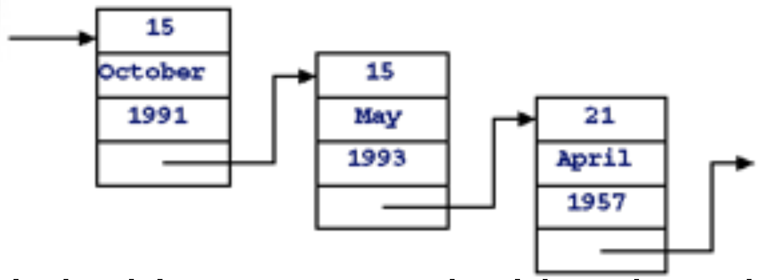
Similarly, to find the correct place to **insert** a new item takes $\log_2(N)$ steps

To place N items correctly takes of order $N*\log_2(N)$ steps

To **sort** N items takes of order $N*\log_2(N)$ steps

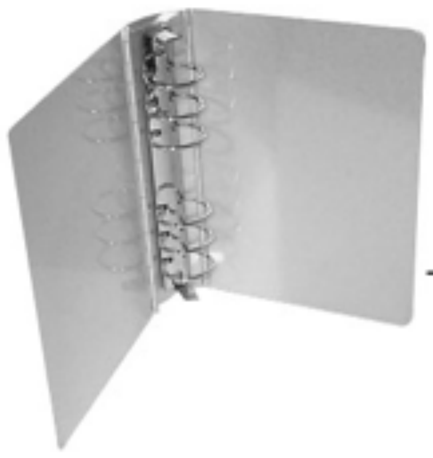
Containers and complexity

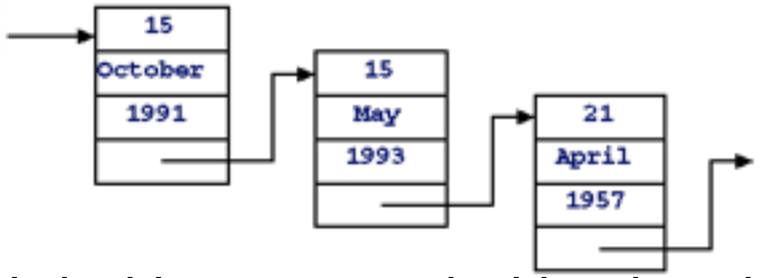




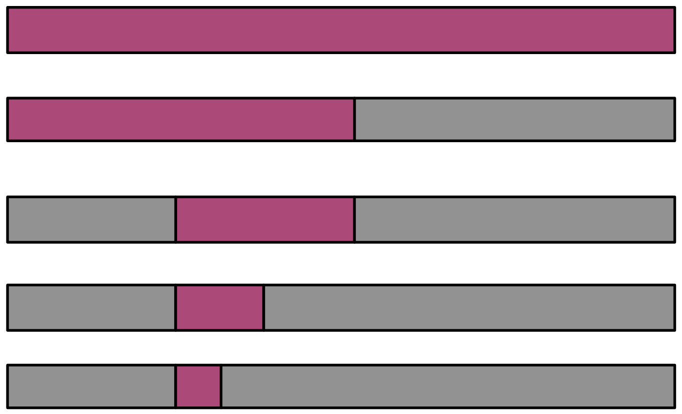
A linked-list is easy to build and modify,
but takes O(N) time to scan N items

Containers and complexity



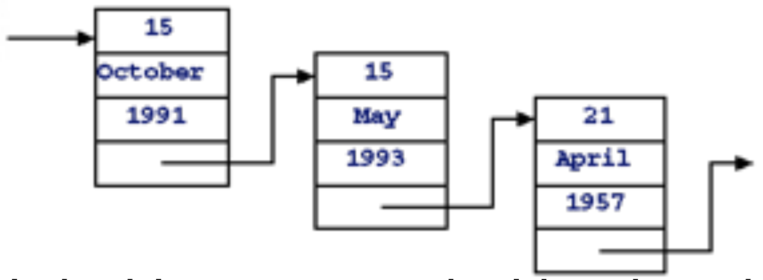


A linked-list is easy to build and modify,
but takes O(N) time to scan N items

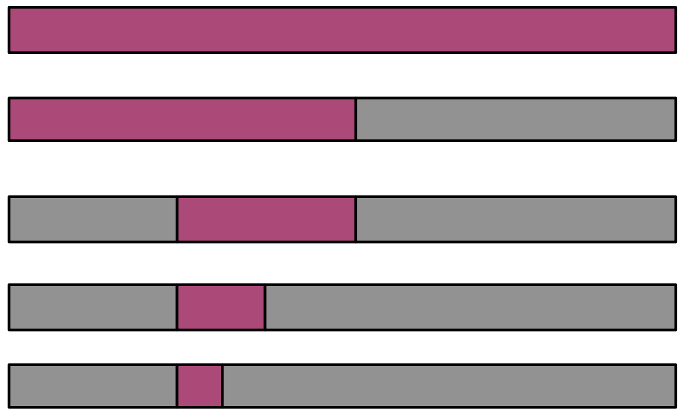


Arrays (c.f. Python lists) are constant to access,
O(log N) to search (if pre-sorted),
O(N log N) to sort, and O(N) to insert/delete

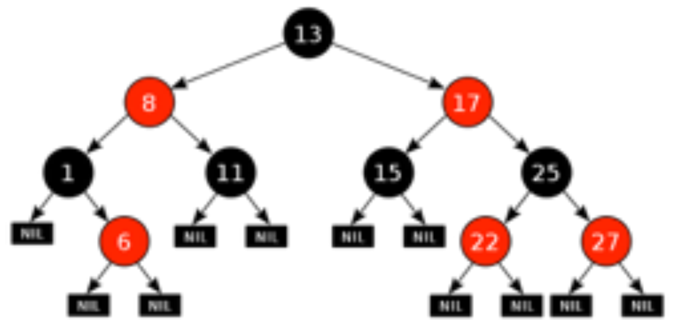
Containers and complexity



A linked-list is easy to build and modify,
but takes O(N) time to scan N items

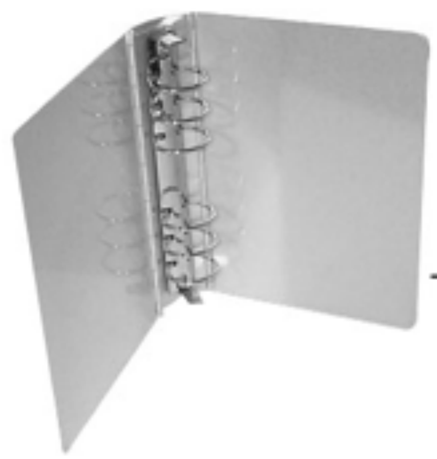


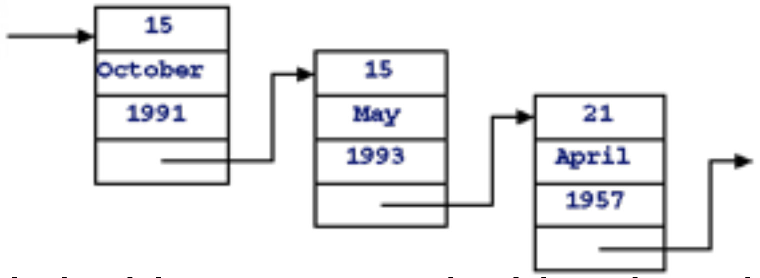
Arrays (c.f. Python lists) are constant to access,
O(log N) to search (if pre-sorted),
O(N log N) to sort, and O(N) to insert/delete



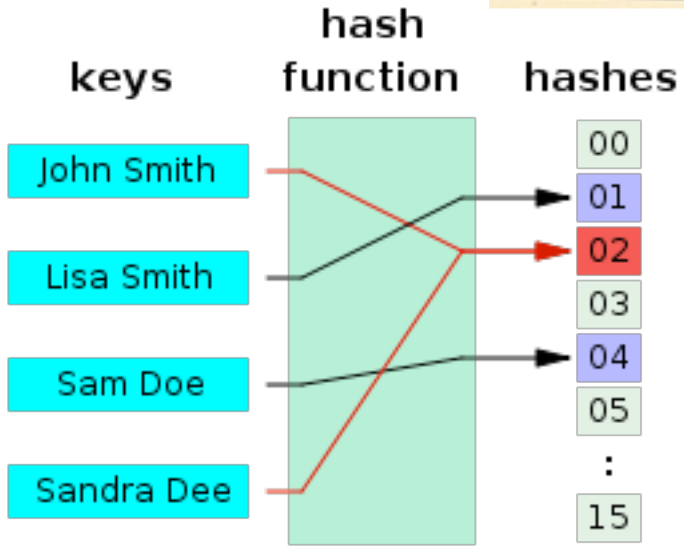
Balanced trees are O(logN) to
search *and* modify, but a little
slower in practice than hashtables

Containers and complexity





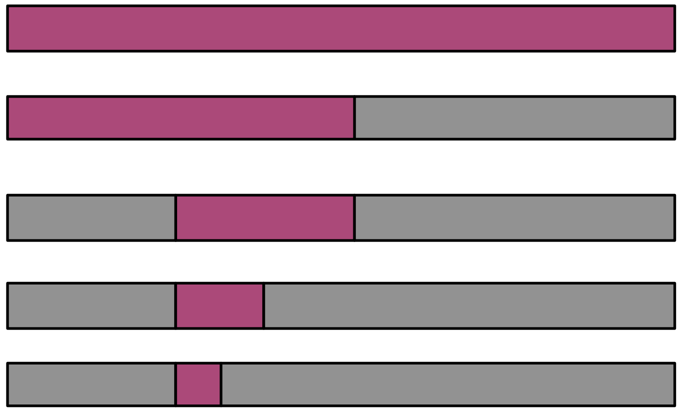
A linked-list is easy to build and modify, but takes O(N) time to scan N items



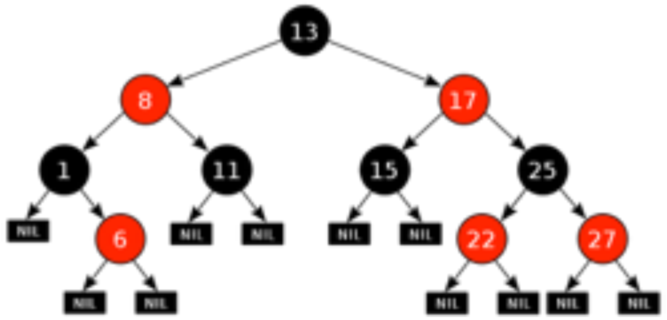
Hashtables (c.f. Python dictionaries) have O(N) worst-case behavior for many operations, but typical case is often better, e.g. O(1)

Summer Fun List
catch fireflies make ice cream make
birdseed wreath scavenger hunt make
hummingbird feeder swim in the ocean release
butterflies make sheet tent rock creatures
leaf prints nature walk tie dye library trip
Greenbear lake paintings hula hoop movie
ice cream at creamery kickball party camp
grandma roast marshmallows freeze tag
play in sprinkler oreo popsicles water
rockets water balloons slip n slide rainy
day movie dream catchers play at park
Sturbridge Village kites obstacle course
lay under stars picnic I Spy pajama
day make smores potato paint write a
poem mini golf bowling feed calves
ocean treasure jars eat home grown veggies
make a CD make a video handprint art
Make a Splash!

TO Do:
- Cast NY lead Raptured
- Take pics for comic
- Write comic scripts
- Plan "50 Kisses" w/Sarah
- Apply for funding - Raptured
- launch food show + pitch format
- finish Raptured scripts
- create timeline for next year
- brainstorm new ideas
  Lchannel 4
- New draft GRIM
- Recce for MOBR



Arrays (c.f. Python lists) are constant to access, O(log N) to search (if pre-sorted), O(N log N) to sort, and O(N) to insert/delete



Balanced trees are O(logN) to search *and* modify, but a little slower in practice than hashtables

Nussinov complexity

- Memory usage

$$f(L) = \sum_{i=1}^{L}\sum_{j=i}^{L} \text{sizeof(int)} = \frac{1}{2}L(L+1) * \text{sizeof(int)}$$

$$g(L) = L^2$$

- Time usage

$$f(L) = \sum_{i=1}^{L}\sum_{j=i}^{L}\sum_{k=i}^{j} \text{(time of innermost step)}$$

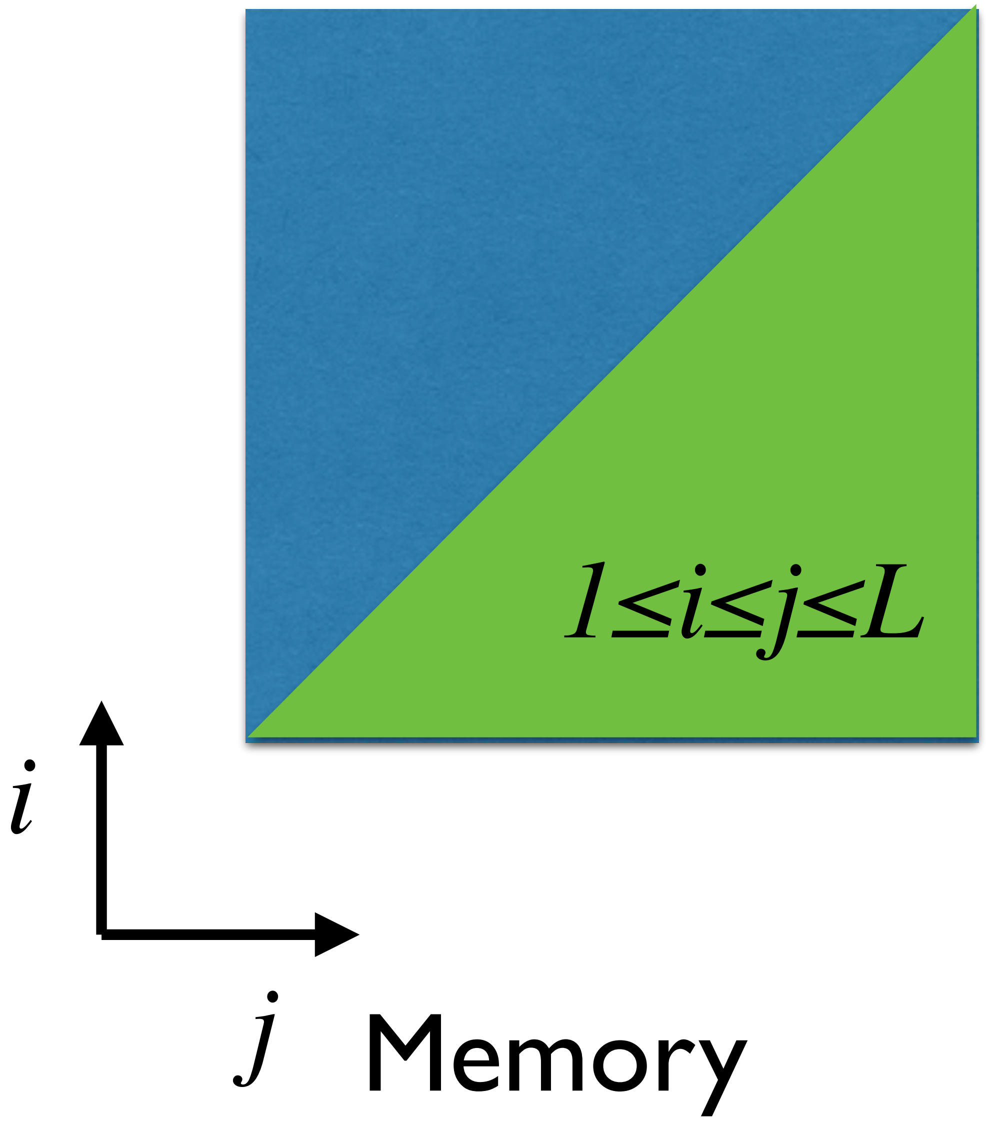
$$g(L) = L^3$$

$$N_{ij} = \max\begin{pmatrix} N_{i+1,j-1} + \delta_{WC}(X_i, X_j) \\ N_{i+1,j} \\ N_{i,j-1} \\ \max_{i<k<j}(N_{ik} + N_{k+1,j}) \end{pmatrix} (\text{if } j > i)$$

$$N_{ii} = 0$$

Nussinov complexity (geometric argument)

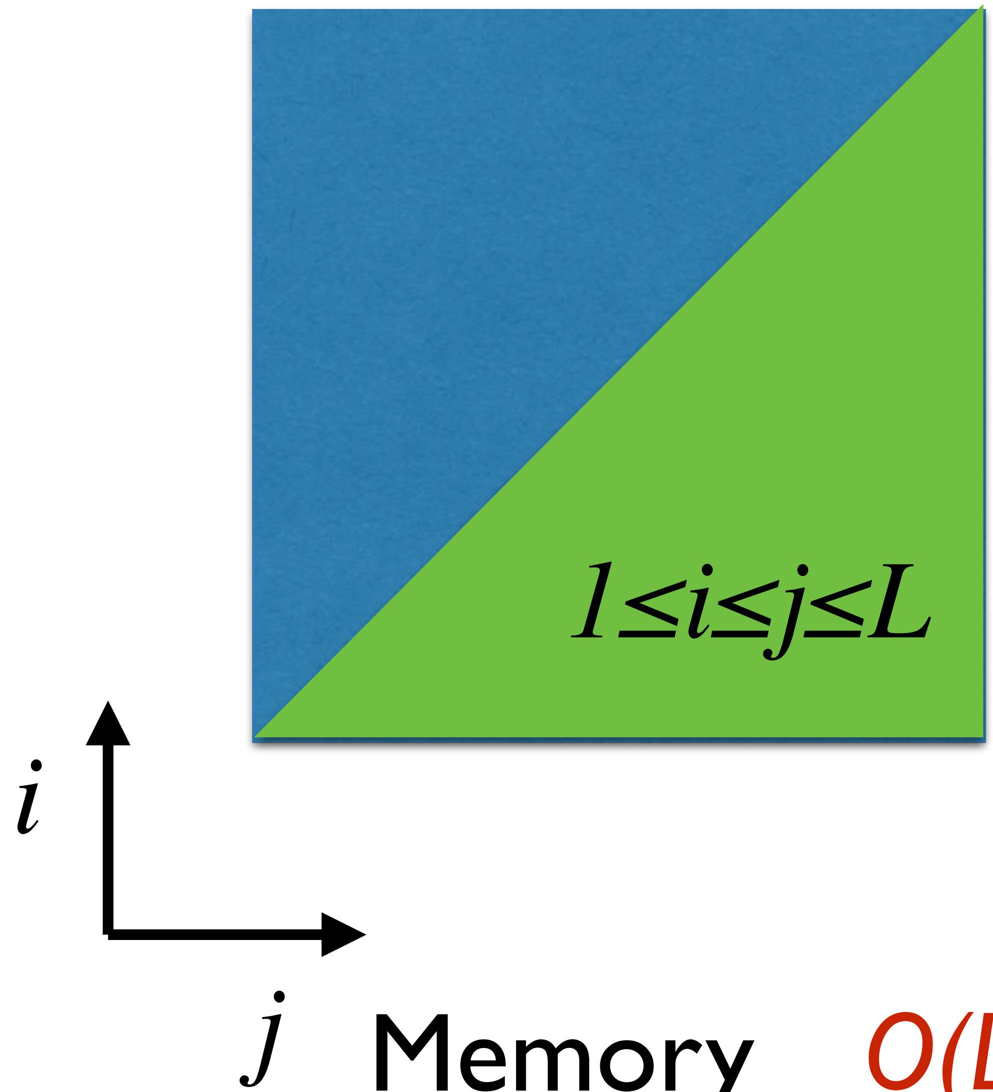
$$N_{ij} = \max \begin{pmatrix} N_{i+1,j-1} + \delta_{WC}(X_i, X_j) \\ N_{i+1,j} \\ N_{i,j-1} \\ \max_{i<k<j}(N_{ik} + N_{k+1,j}) \end{pmatrix}$$



Memory

Nussinov complexity (geometric argument)

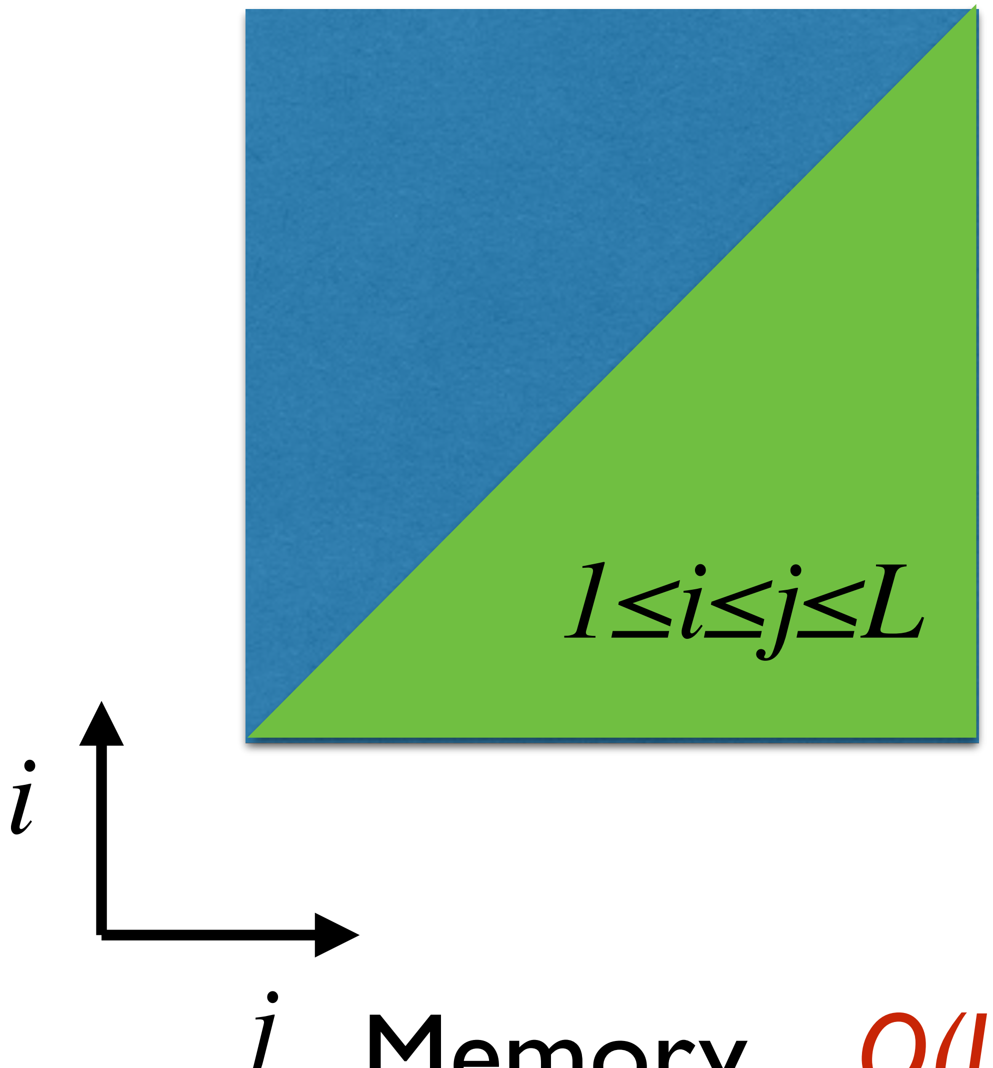
$$N_{ij} = \max \begin{pmatrix} N_{i+1,j-1} + \delta_{WC}(X_i, X_j) \\ N_{i+1,j} \\ N_{i,j-1} \\ \max_{i<k<j}\left(N_{ik} + N_{k+1,j}\right) \end{pmatrix}$$



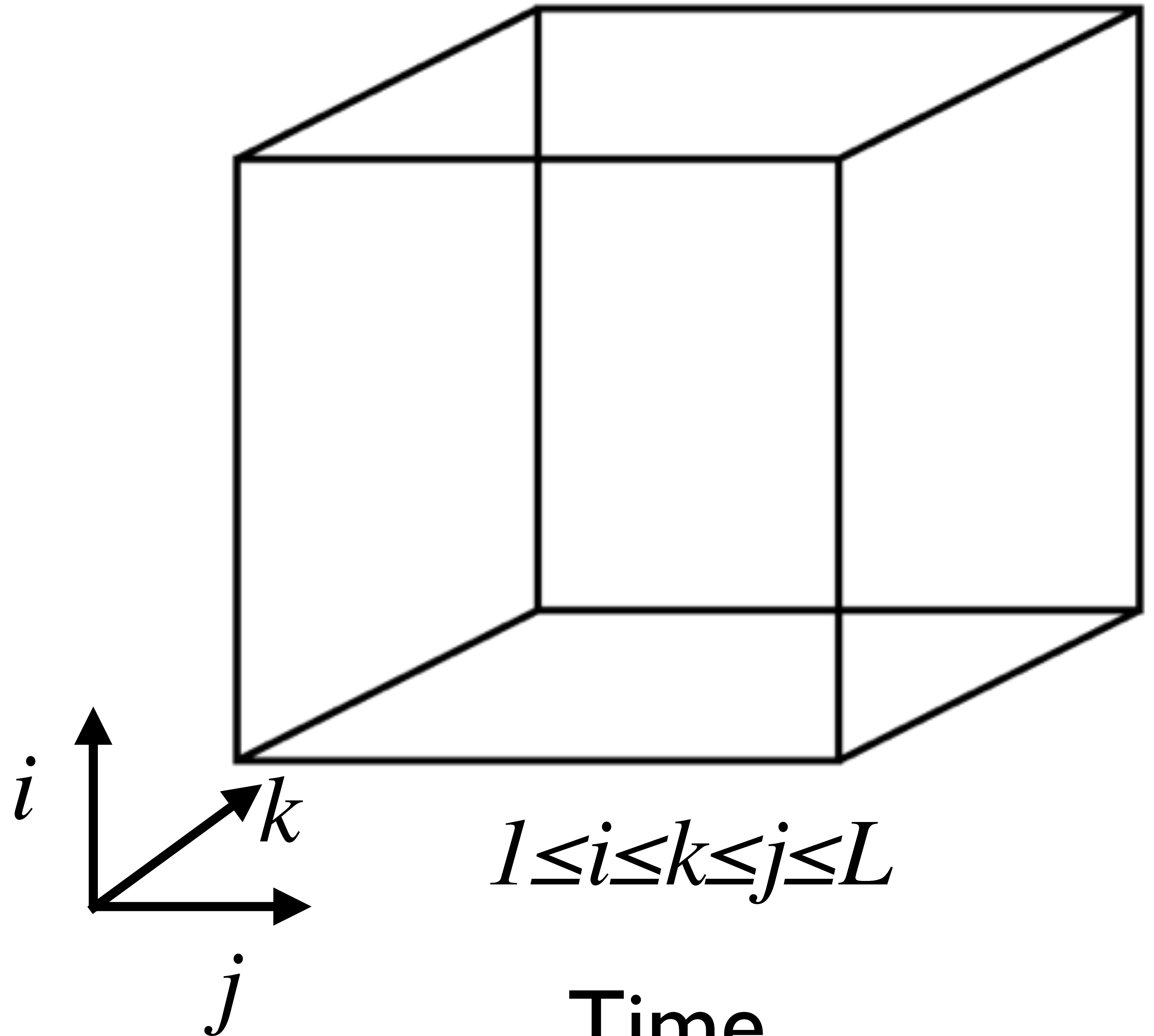
Memory $O(L^2)$

Nussinov complexity (geometric argument)

$$N_{ij} = \max \begin{pmatrix} N_{i+1,j-1} + \delta_{WC}(X_i, X_j) \\ N_{i+1,j} \\ N_{i,j-1} \\ \max_{i<k<j}(N_{ik} + N_{k+1,j}) \end{pmatrix}$$



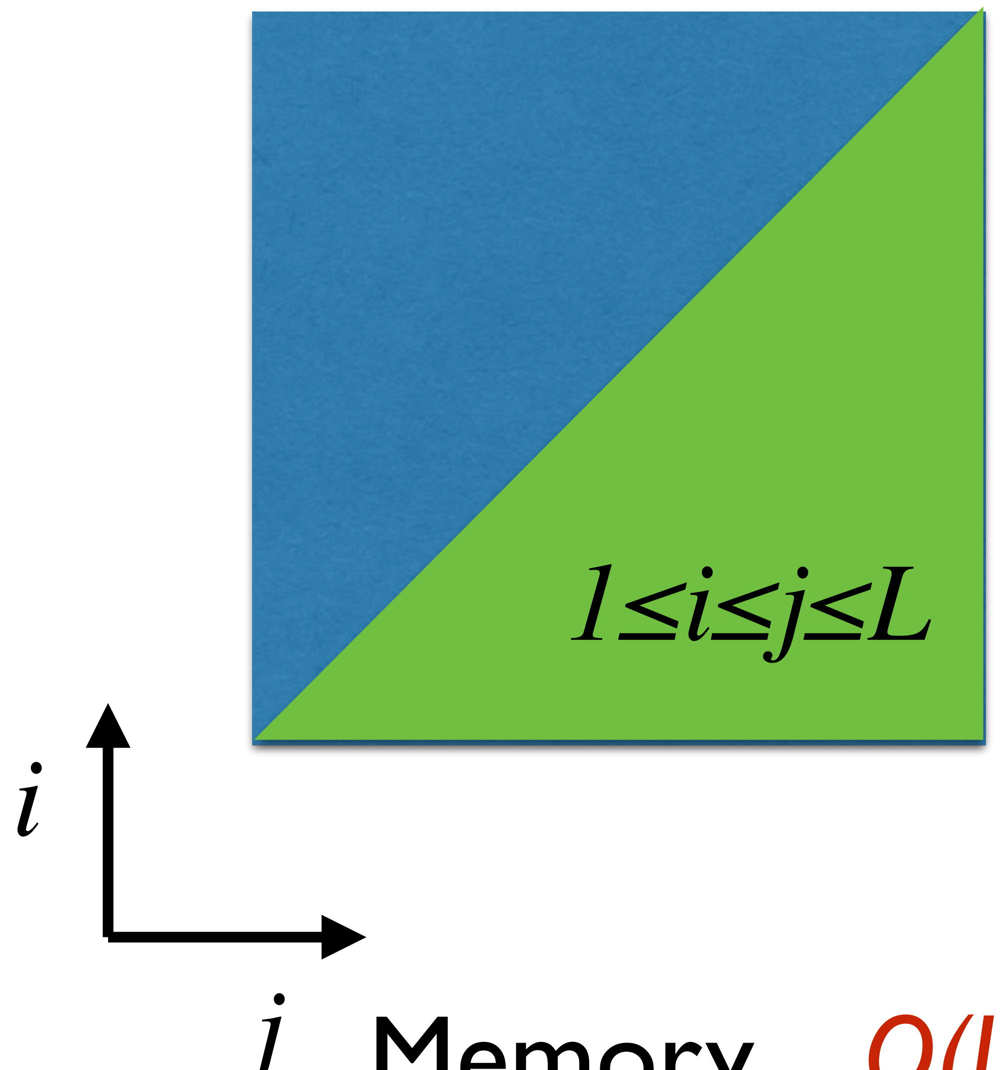
Memory $O(L^2)$



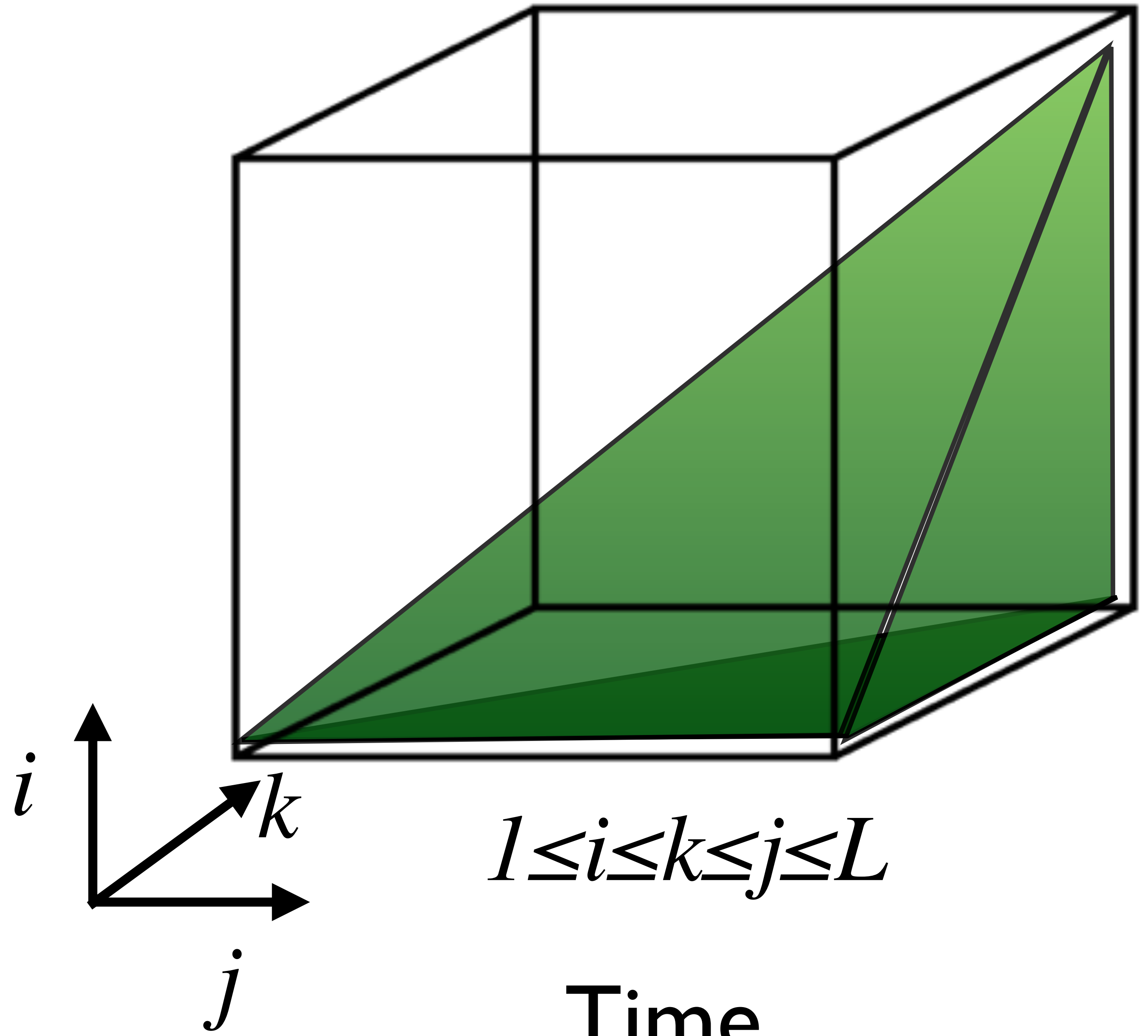
Time

Nussinov complexity (geometric argument)

$$N_{ij} = \max \begin{pmatrix} N_{i+1,j-1} + \delta_{WC}(X_i, X_j) \\ N_{i+1,j} \\ N_{i,j-1} \\ \max_{i<k<j}(N_{ik} + N_{k+1,j}) \end{pmatrix}$$



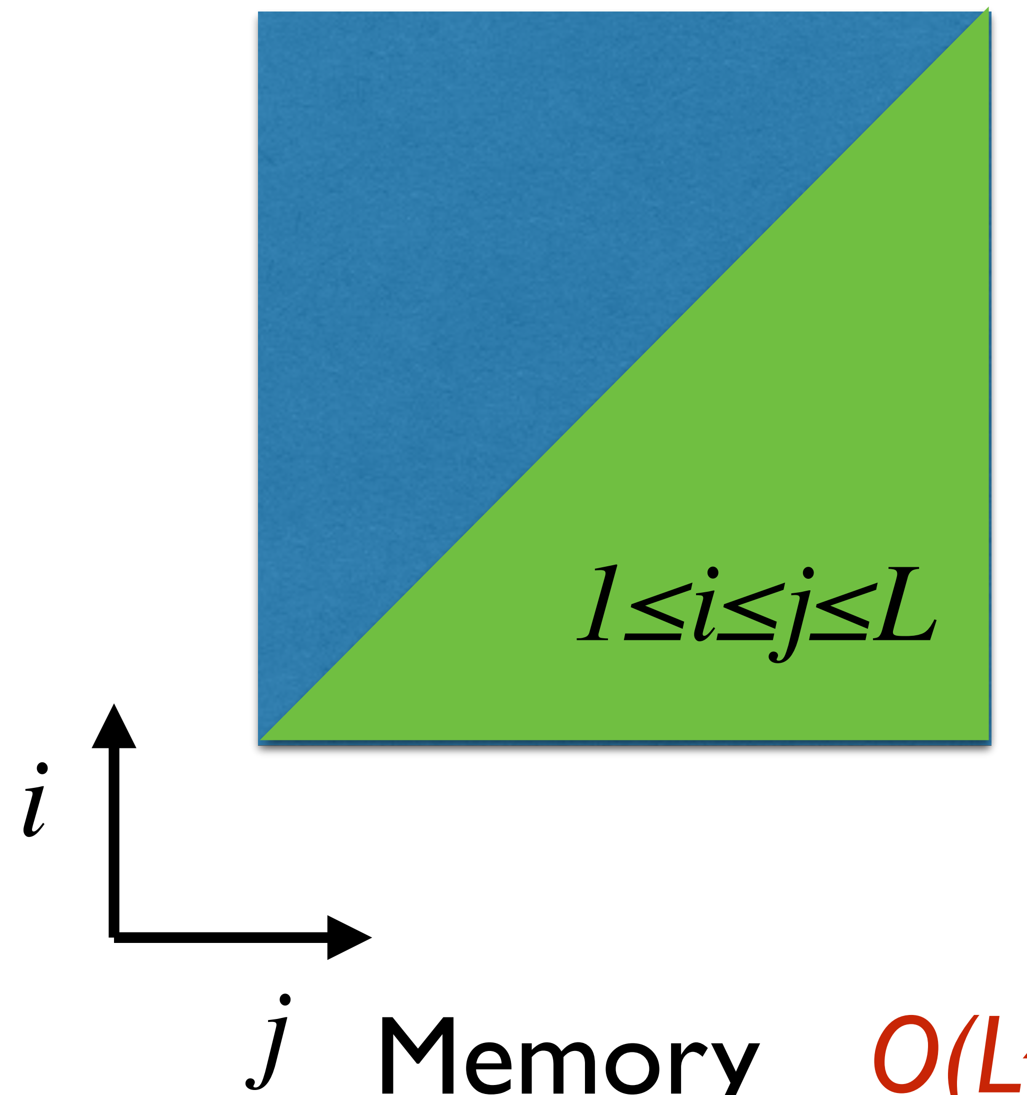
Memory $O(L^2)$



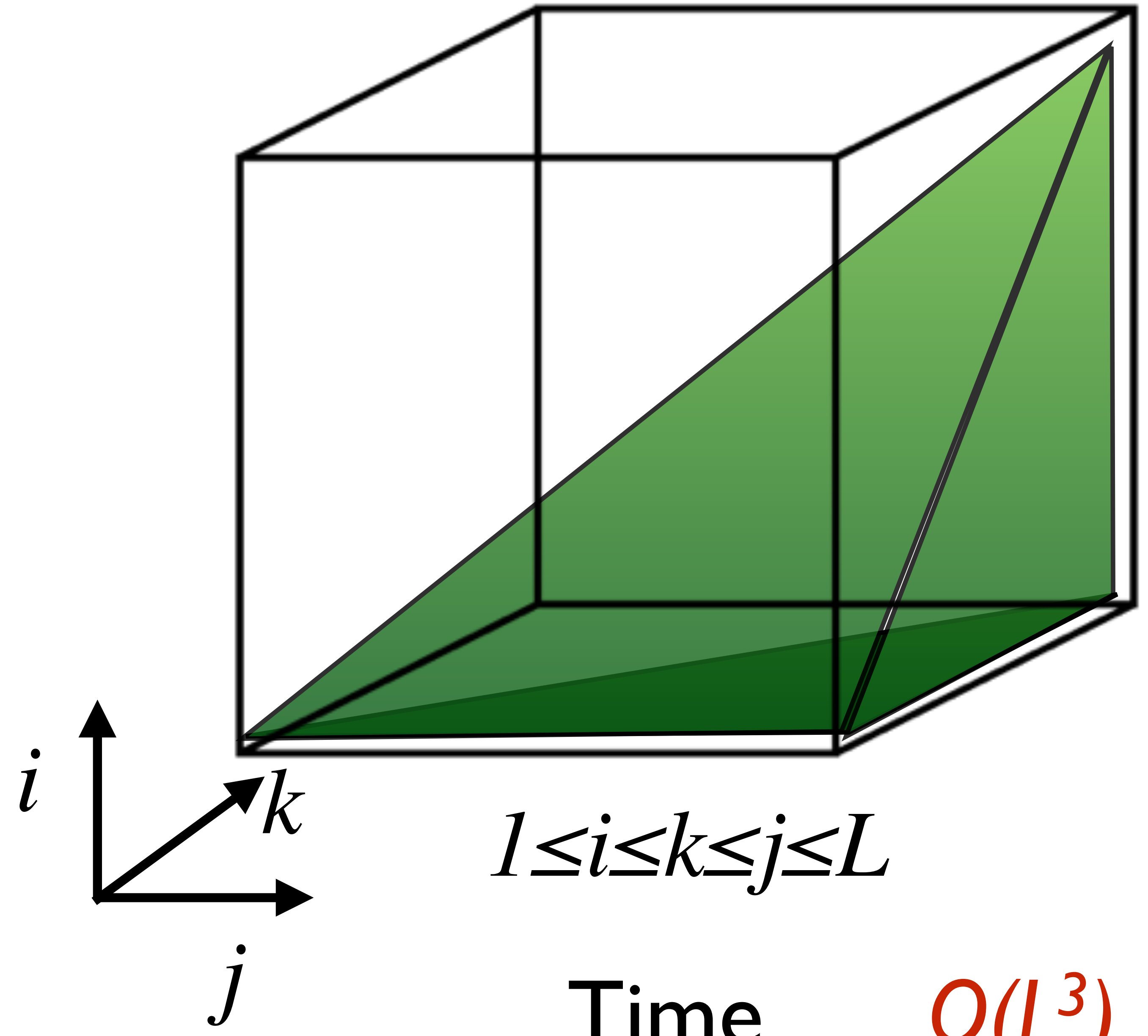
Time

Nussinov complexity (geometric argument)

$$N_{ij} = \max \begin{pmatrix} N_{i+1,j-1} + \delta_{WC}(X_i, X_j) \\ N_{i+1,j} \\ N_{i,j-1} \\ \max_{i<k<j}(N_{ik} + N_{k+1,j}) \end{pmatrix}$$



Memory $O(L^2)$

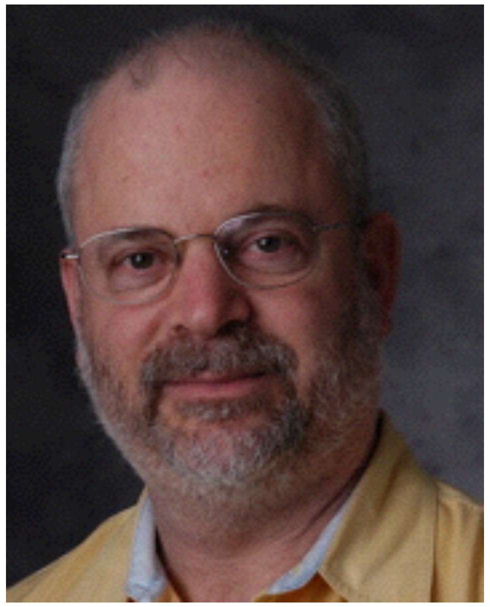


Time $O(L^3)$

Towards Zuker's algorithm

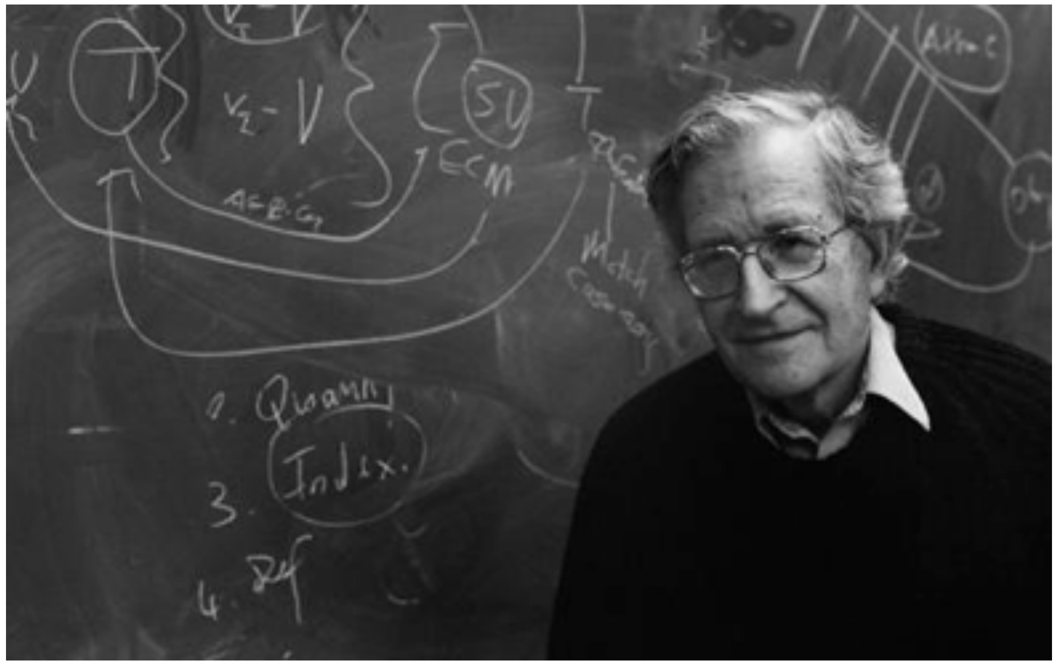
- How would you modify the Nussinov recursion to...
  - allow wobble base-pairs?
  - calculate the energy of all base-pairs in the structure, rather than just the number of base-pairs?
  - prevent loops of length <3?
  - incorporate loop energies?
  - incorporate stacking energies?

$$N_{ij} = \max \begin{pmatrix} N_{i+1,j-1} + \delta_{WC}(X_i, X_j) \\ N_{i+1,j} \\ N_{i,j-1} \\ \max_{i<k<j}(N_{ik} + N_{k+1,j}) \end{pmatrix} \text{(if } j > i\text{)}$$
$$N_{ii} = 0$$


Michael Zuker

Towards Zuker's algorithm

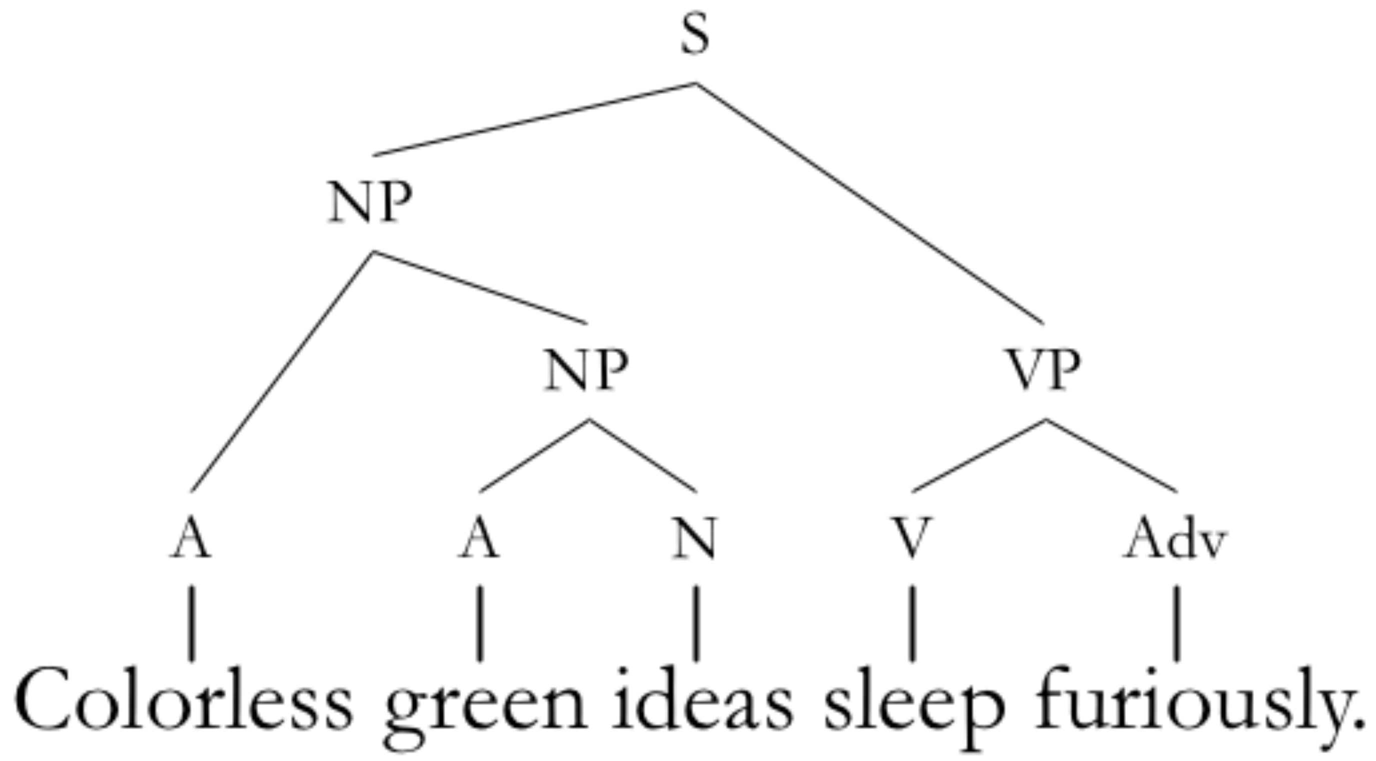
- How would you modify the Nussinov recursion to...
  - allow wobble base-pairs?
  - calculate the <u>energy</u> of all base-pairs in the structure, rather than just the <u>number</u> of base-pairs?
  - prevent loops of length <3?
  - incorporate loop energies?
  - incorporate stacking energies?



One approach is to use Chomsky Grammars

Chomsky grammars

- A Chomsky grammar is an abstract recipe for generating a string
- The process of generating the string reflects the structure underlying the string



Chomsky grammars

- A **transformational grammar** consists of
  - A set of grammar **symbols**, including
    - **Nonterminal symbols,**
      one of which is labeled as the "Start" symbol
    - **Terminal symbols**
  - A set of **production rules** of the form $A \rightarrow B$ where
    - A is a string of nonterminals
    - B is a string of symbols (terms and/or nonterms)
- In an attributed grammar, the rules are associated with attributes of some kind (scores, weights, energies, lines of code, ...)
- Context-free grammars are useful models for RNA structure
- More on grammars later in the semester

Chomsky grammars

- A **transformational grammar** consists of
  - A set of grammar **symbols**, including
    - **Nonterminal symbols,**
      one of which is labeled as the "Start" symbol {S}
    - **Terminal symbols**
  - A set of **production rules** of the form $A \rightarrow B$ where
    - A is a string of nonterminals
    - B is a string of symbols (terms and/or nonterms)
- In an attributed grammar, the rules are associated with attributes of some kind (scores, weights, energies, lines of code, ...)
- Context-free grammars are useful models for RNA structure
- More on grammars later in the semester

Chomsky grammars

- A **transformational grammar** consists of
    - A set of grammar **symbols**, including
        - **Nonterminal symbols,**
          one of which is labeled as the "Start" symbol — {S}
        - **Terminal symbols** — {A,C,G,U}
    - A set of **production rules** of the form *A*→*B* where
        - *A* is a string of nonterminals
        - *B* is a string of symbols (terms and/or nonterms)
- In an attributed grammar, the rules are associated with attributes of some kind (scores, weights, energies, lines of code, ...)
- Context-free grammars are useful models for RNA structure
- More on grammars later in the semester

Chomsky grammars

- A **transformational grammar** consists of
  - A set of grammar **symbols**, including
    - **Nonterminal symbols,** one of which is labeled as the "Start" symbol — {S}
    - **Terminal symbols** — {A,C,G,U}
  - A set of **production rules** of the form $A \rightarrow B$ where
    - A is a string of nonterminals
    - B is a string of symbols (terms and/or nonterms)
- In an attributed grammar, the rules are associated with attributes of some kind (scores, weights, energies, lines of code, ...)
- Context-free grammars are useful models for RNA structure
- More on grammars later in the semester

$S \rightarrow ASU$
$S \rightarrow CSG$
$S \rightarrow GSC$
$S \rightarrow USA$
$S \rightarrow SS$
$S \rightarrow A$
$S \rightarrow C$
$S \rightarrow G$
$S \rightarrow U$

Chomsky grammars

S

{S}

{A,C,G,U}

$S \rightarrow ASU$
$S \rightarrow CSG$
$S \rightarrow GSC$
$S \rightarrow USA$
$S \rightarrow SS$
$S \rightarrow A$
$S \rightarrow C$
$S \rightarrow G$
$S \rightarrow U$

Chomsky grammars

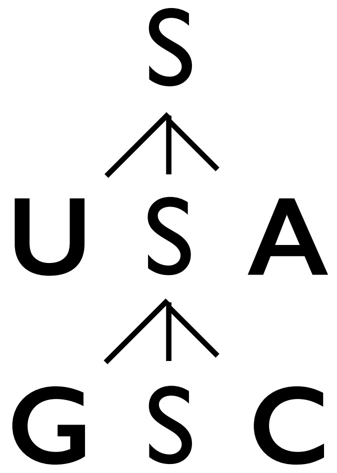
$$\begin{tikzcd}[column sep=small] & S \arrow[dl, no head] \arrow[d, no head] \arrow[dr, no head] & \\ \mathbf{U} & S & \mathbf{A} \end{tikzcd}$$

{S}

{A,C,G,U}

S→ASU
S→CSG
S→GSC
S→USA
S→SS
S→A
S→C
S→G
S→U

Chomsky grammars

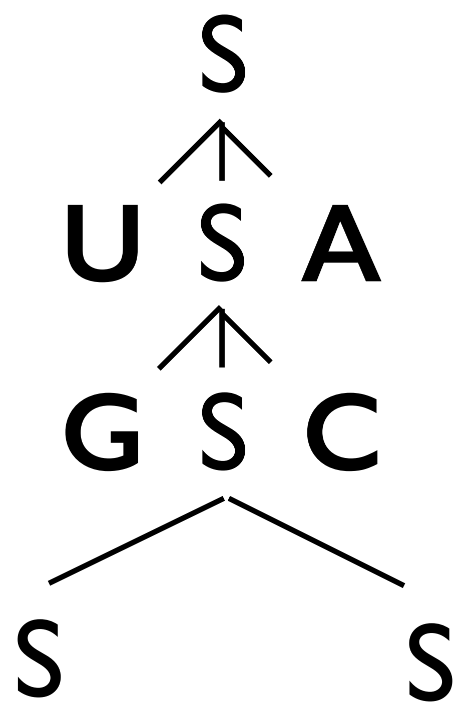


{S}

{A,C,G,U}

S→*ASU*
S→*CSG*
S→*GSC*
S→*USA*
S→*SS*
S→*A*
S→*C*
S→*G*
S→*U*

Chomsky grammars

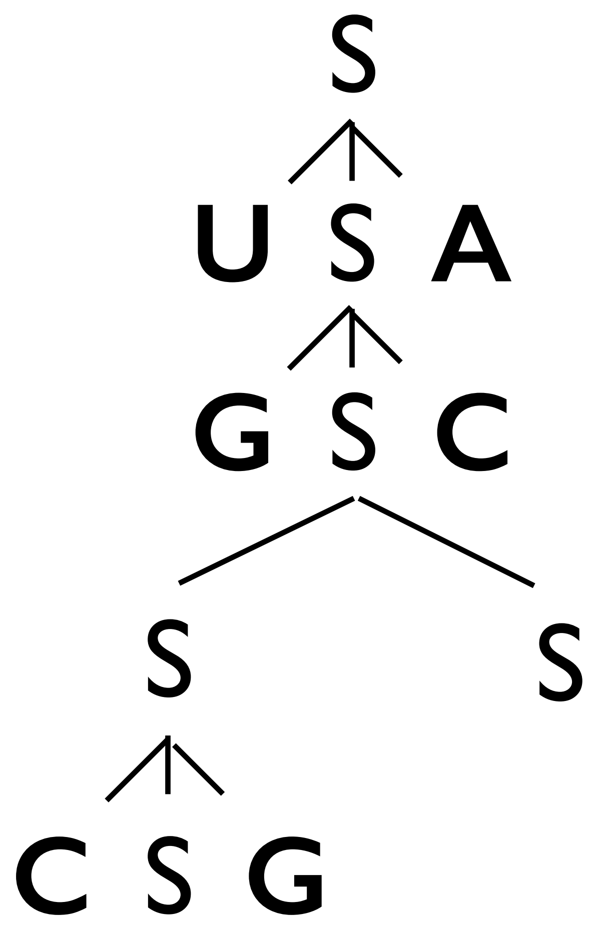


{S}

{A,C,G,U}

S→ASU
S→CSG
S→GSC
S→USA
S→SS
S→A
S→C
S→G
S→U

Chomsky grammars

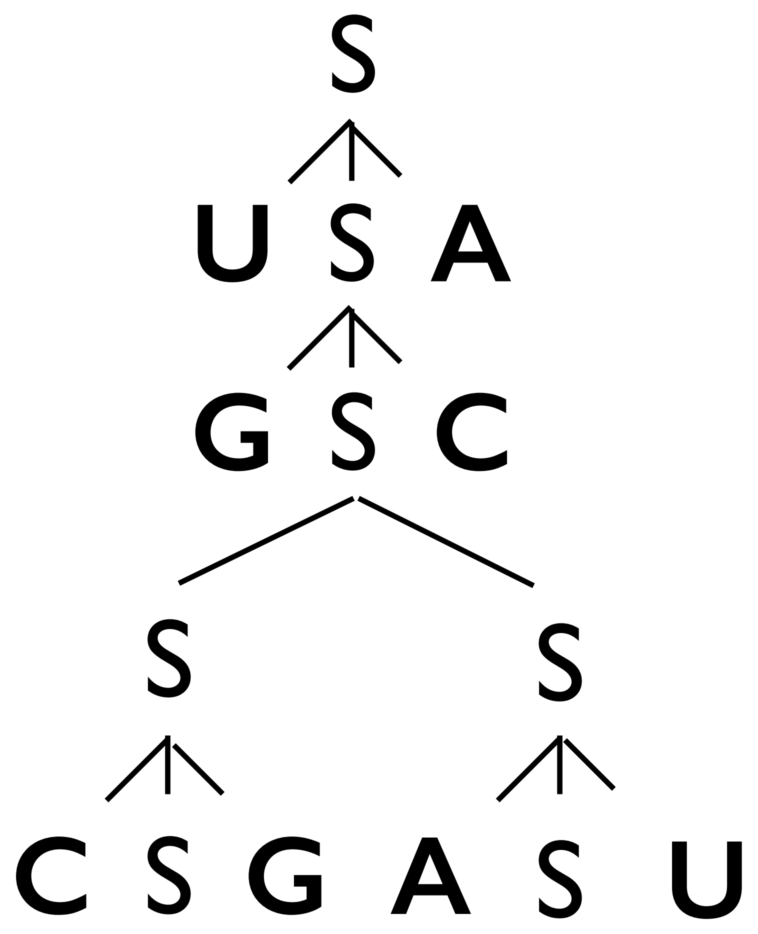


{S}

{A,C,G,U}

S→ASU
S→CSG
S→GSC
S→USA
S→SS
S→A
S→C
S→G
S→U

Chomsky grammars

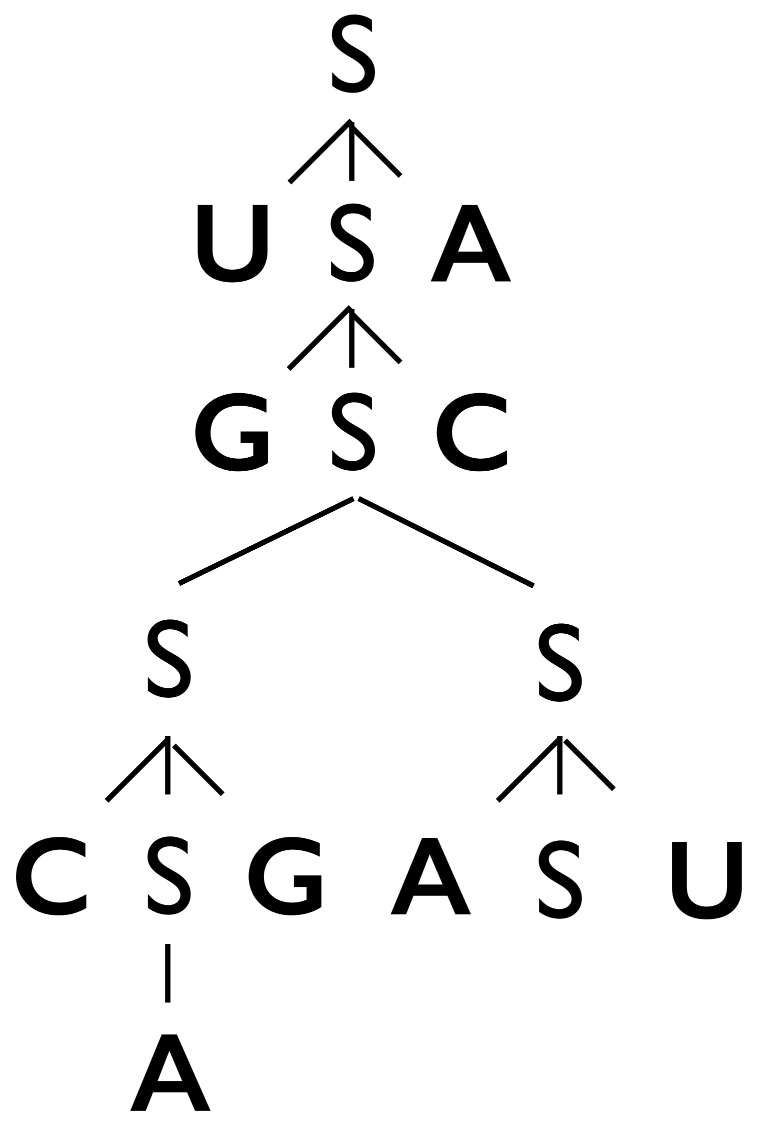


{S}

{A,C,G,U}

S→ASU
S→CSG
S→GSC
S→USA
S→SS
S→A
S→C
S→G
S→U

Chomsky grammars

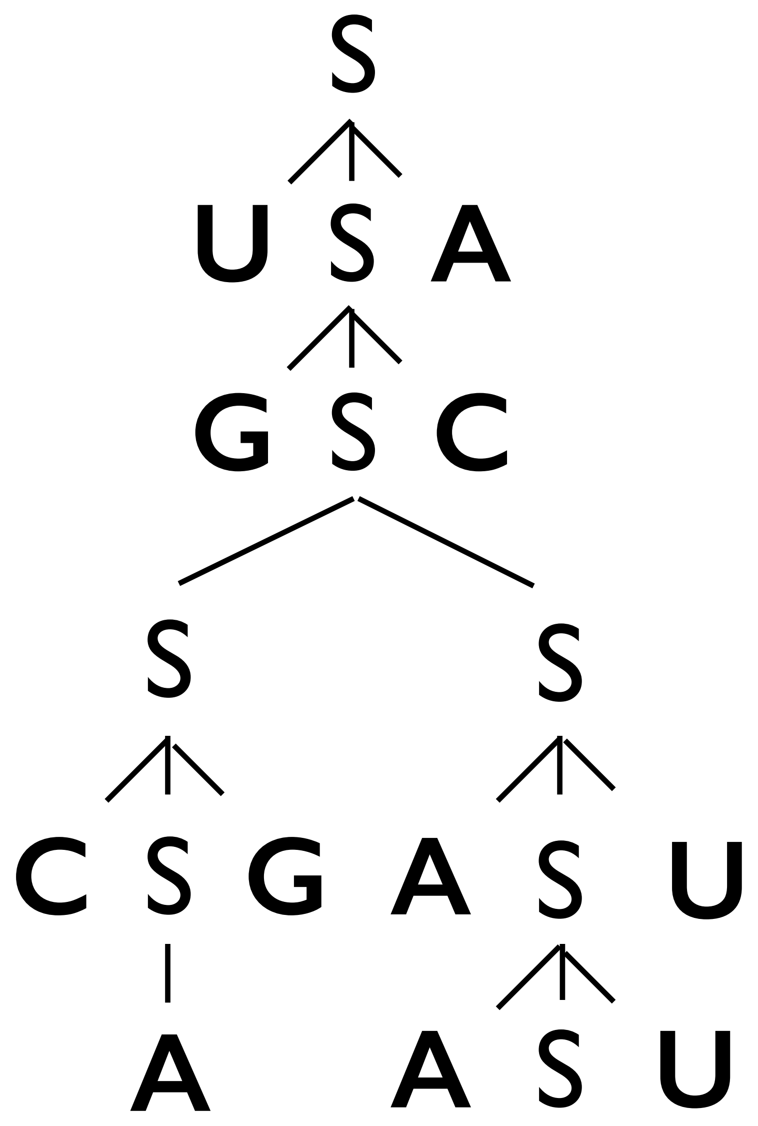


{S}

{A,C,G,U}

S→ASU
S→CSG
S→GSC
S→USA
S→SS
S→A
S→C
S→G
S→U

Chomsky grammars

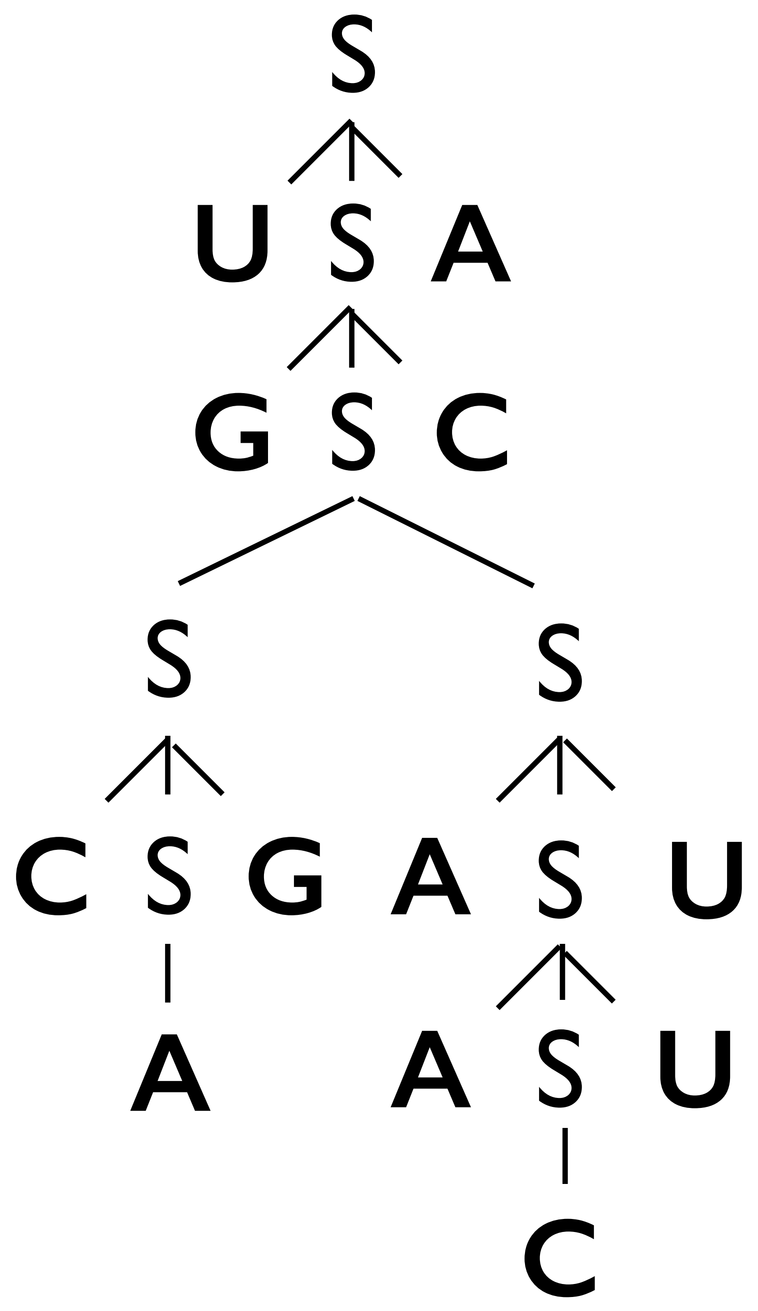


{S}

{A,C,G,U}

S→ASU
S→CSG
S→GSC
S→USA
S→SS
S→A
S→C
S→G
S→U

Chomsky grammars

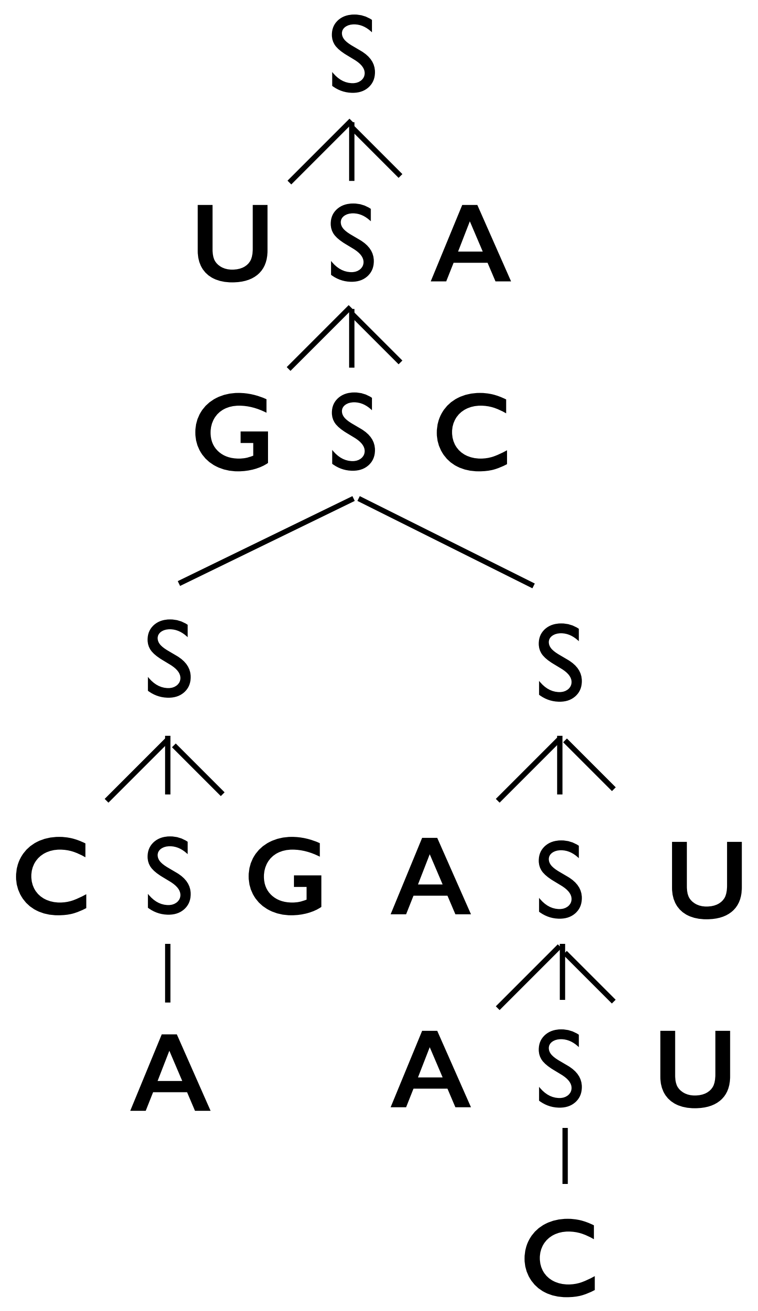


{S}

{A,C,G,U}

S→ASU
S→CSG
S→GSC
S→USA
S→SS
S→A
S→C
S→G
S→U

Chomsky grammars



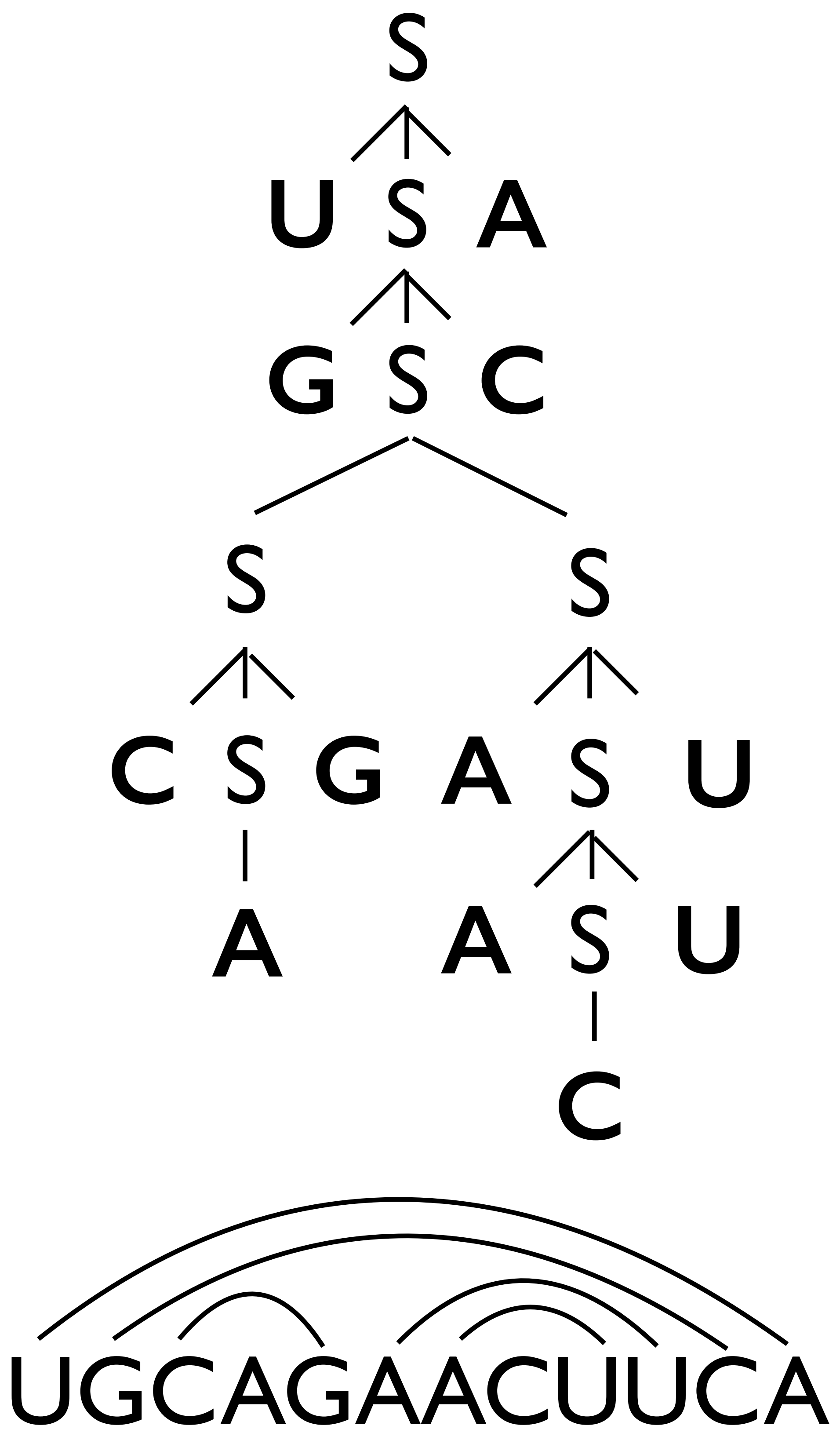
{S}

{A,C,G,U}

S→ASU
S→CSG
S→GSC
S→USA
S→SS
S→A
S→C
S→G
S→U

UGCAGAACUUCA

Chomsky grammars



{S}

{A,C,G,U}

S→ASU
S→CSG
S→GSC
S→USA
S→SS
S→A
S→C
S→G
S→U

The partition function

$$Z(X) = \sum_{S \in \mathbb{S}(X)} e^{-\beta E_S} = \sum_{S \in \mathbb{S}(X)} \prod_{(i,j) \in \mathbb{B}(S)} \exp(-\beta E(X_i, X_j))$$

$$\beta = \frac{1}{k_B T}$$

$$E_S = \text{energy of structure } S = \sum_{(i,j) \in \mathbb{B}(S)} E(X_i, X_j)$$

$$E(a,b) = \text{energy of basepair } (a,b)$$

$$\mathbb{S}(X) = \text{set of possible structures of sequence } X$$

$$\mathbb{B}(S) = \text{set of basepair coords in structure } S$$

The partition function

$$Z(X) = \sum_{S \in \mathbb{S}(X)} e^{-\beta E_S} = \sum_{S \in \mathbb{S}(X)} \prod_{(i,j) \in \mathbb{B}(S)} \exp(-\beta E(X_i, X_j))$$

$$\beta = \frac{1}{k_B T}$$

$$E_S = \text{energy of structure } S = \sum_{(i,j) \in \mathbb{B}(S)} E(X_i, X_j)$$

$$E(a,b) = \text{energy of basepair } (a,b)$$

$$\mathbb{S}(X) = \text{set of possible structures of sequence } X$$

$$\mathbb{B}(S) = \text{set of basepair coords in structure } S$$

$$\Pr(\text{structure } S \mid \text{sequence } X) = \frac{e^{-\beta E_S}}{Z}$$

The partition function

$$Z(X) = \sum_{S \in \mathbb{S}(X)} e^{-\beta E_S} = \sum_{S \in \mathbb{S}(X)} \prod_{(i,j) \in \mathbb{B}(S)} \exp(-\beta E(X_i, X_j))$$

$$\beta = \frac{1}{k_B T}$$

$$E_S = \text{energy of structure } S = \sum_{(i,j) \in \mathbb{B}(S)} E(X_i, X_j)$$

$$E(a,b) = \text{energy of basepair } (a,b)$$

$$\mathbb{S}(X) = \text{set of possible structures of sequence } X$$

$$\mathbb{B}(S) = \text{set of basepair coords in structure } S$$

$$\Pr(\text{structure } S \mid \text{sequence } X) = \frac{e^{-\beta E_S}}{Z}$$

$$\Pr(\text{basepair } i,j \mid \text{sequence } X) = \frac{Z_{ij}}{Z} = \frac{\sum_{S \in \mathbb{S}(X),\, (i,j) \in \mathbb{B}(S)} e^{-\beta E_S}}{Z(X)}$$

$$\mathbb{B}(S) = \text{set of basepair coords in structure } S$$

Relation to Nussinov

$$N(X) = \max_{S \in \mathbb{S}(X)} N(S) = \max_{S \in \mathbb{S}(X)} \sum_{(i,j) \in \mathbb{B}(S)} \delta_{WC}(X_i, X_j)$$

$$Z(X) = \sum_{S \in \mathbb{S}(X)} e^{-\beta E_S} = \sum_{S \in \mathbb{S}(X)} \prod_{(i,j) \in \mathbb{B}(S)} \exp(-\beta E(X_i, X_j))$$

$$\beta = \frac{1}{k_B T}$$

$$E_S = \text{energy of structure } S = \sum_{(i,j) \in \mathbb{B}(S)} E(X_i, X_j)$$

$$E(a,b) = \text{energy of basepair } (a,b)$$

$$\mathbb{S}(X) = \text{set of possible structures of sequence } X$$

$$\mathbb{B}(S) = \text{set of basepair coords in structure } S$$

Relation to Nussinov

$$N(X) = \max_{S \in \mathbb{S}(X)} N(S) = \max_{S \in \mathbb{S}(X)} \sum_{(i,j) \in \mathbb{B}(S)} \delta_{WC}(X_i, X_j)$$

$$Z(X) = \sum_{S \in \mathbb{S}(X)} e^{-\beta E_S} = \sum_{S \in \mathbb{S}(X)} \prod_{(i,j) \in \mathbb{B}(S)} \exp(-\beta E(X_i, X_j))$$

$$\beta = \frac{1}{k_B T}$$

$$E_S = \text{energy of structure } S = \sum_{(i,j) \in \mathbb{B}(S)} E(X_i, X_j)$$

$$E(a,b) = \text{energy of basepair } (a,b)$$

$$\mathbb{S}(X) = \text{set of possible structures of sequence } X$$

$$\mathbb{B}(S) = \text{set of basepair coords in structure } S$$

Relation to Nussinov

$$N(X) = \max_{S \in \mathbb{S}(X)} N(S) = \max_{S \in \mathbb{S}(X)} \sum_{(i,j) \in \mathbb{B}(S)} \delta_{WC}(X_i, X_j)$$

$$Z(X) = \sum_{S \in \mathbb{S}(X)} e^{-\beta E_S} = \sum_{S \in \mathbb{S}(X)} \prod_{(i,j) \in \mathbb{B}(S)} \exp(-\beta E(X_i, X_j))$$

$$\beta = \frac{1}{k_B T}$$

$$E_S = \text{energy of structure } S = \sum_{(i,j) \in \mathbb{B}(S)} E(X_i, X_j)$$

$$E(a,b) = \text{energy of basepair } (a,b)$$

$$\mathbb{S}(X) = \text{set of possible structures of sequence } X$$

$$\mathbb{B}(S) = \text{set of basepair coords in structure } S$$

Relation to Nussinov

$$N(X) = \max_{S \in \mathbb{S}(X)} N(S) = \max_{S \in \mathbb{S}(X)} \sum_{(i,j) \in \mathbb{B}(S)} \delta_{WC}(X_i, X_j)$$

$$Z(X) = \sum_{S \in \mathbb{S}(X)} e^{-\beta E_S} = \sum_{S \in \mathbb{S}(X)} \prod_{(i,j) \in \mathbb{B}(S)} \exp(-\beta E(X_i, X_j))$$

$$\beta = \frac{1}{k_B T}$$

$$E_S = \text{energy of structure } S = \sum_{(i,j) \in \mathbb{B}(S)} E(X_i, X_j)$$

$$E(a,b) = \text{energy of basepair } (a,b)$$

$$\mathbb{S}(X) = \text{set of possible structures of sequence } X$$

$$\mathbb{B}(S) = \text{set of basepair coords in structure } S$$

McCaskill algorithm

- Very similar structure to Nussinov
  - Two-dimensional DP array (i,j)
  - Three nested loops (i,j,k)

McCaskill algorithm

- Very similar structure to Nussinov
  - Two-dimensional DP array (i,j)
  - Three nested loops (i,j,k)

$$\Pr(\text{basepair } i,j \mid \text{sequence } X) = \frac{Z_{ij}}{Z} = \frac{\sum_{S \in \mathbb{S}(X),\,(i,j) \in \mathbb{B}(S)} e^{-\beta E_S}}{Z(X)}$$

$$\mathbb{B}(S) = \text{set of basepair coords in structure } S$$

McCaskill algorithm

- Very similar structure to Nussinov
  - Two-dimensional DP array (i,j)
  - Three nested loops (i,j,k)

$$\Pr(\text{basepair } i,j \mid \text{sequence } X) = \frac{Z_{ij}}{Z} = \frac{\boxed{\displaystyle\sum_{S \in \mathbb{S}(X),\,(i,j) \in \mathbb{B}(S)} e^{-\beta E_S}}}{Z(X)}$$

$$\mathbb{B}(S) = \text{set of basepair coords in structure } S$$

Z_{ij} can be factored into three terms:
Inside, (i,j), Outside

McCaskill algorithm

ACCACCUGCACAGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGUGUGCAGG

Z_{ij} can be factored into three terms:

Inside, (i,j), Outside

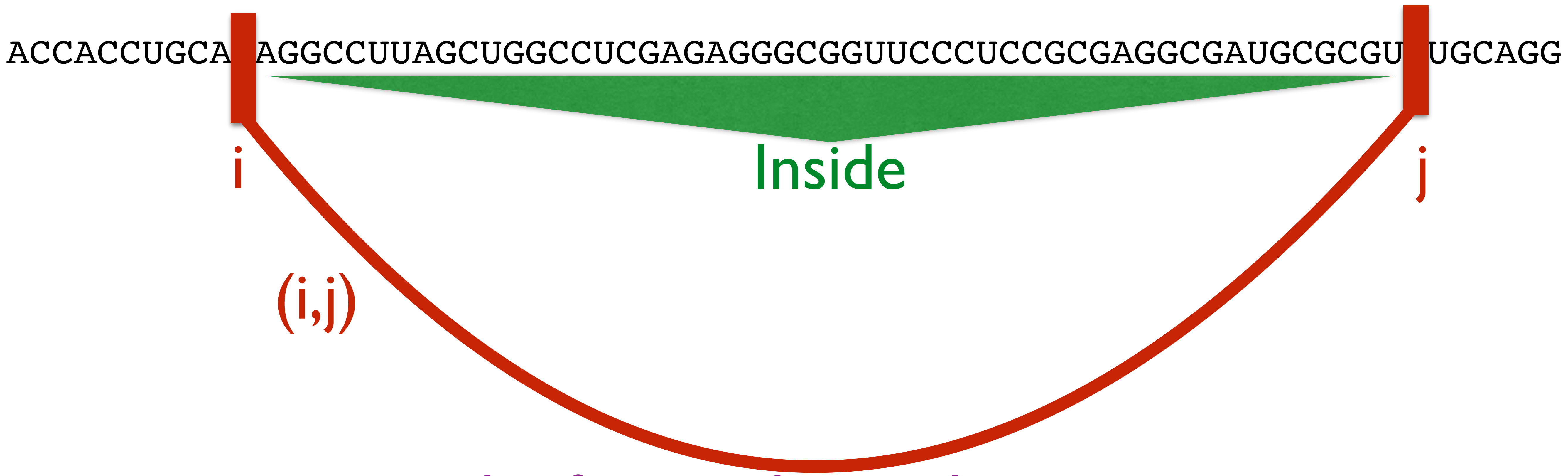
McCaskill algorithm

ACCACCUGCACAGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGUGUGCAGG

Z_{ij} can be factored into three terms:

Inside, (i,j), Outside

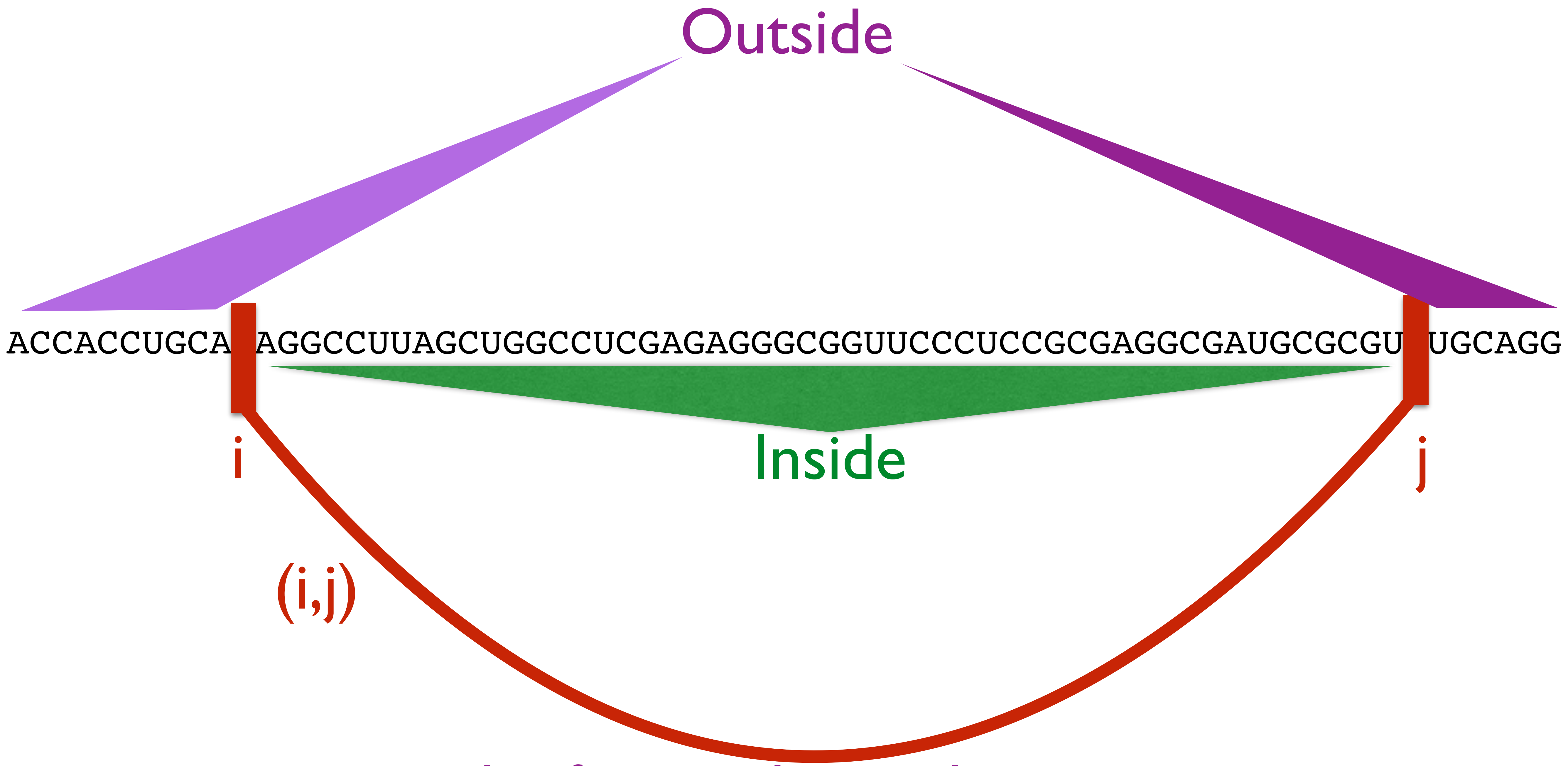
McCaskill algorithm


ACCACCUGCA AGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGU UGCAGG
i
j
Inside
(i,j)

Z_{ij} can be factored into three terms:

Inside, (i,j), Outside

McCaskill algorithm

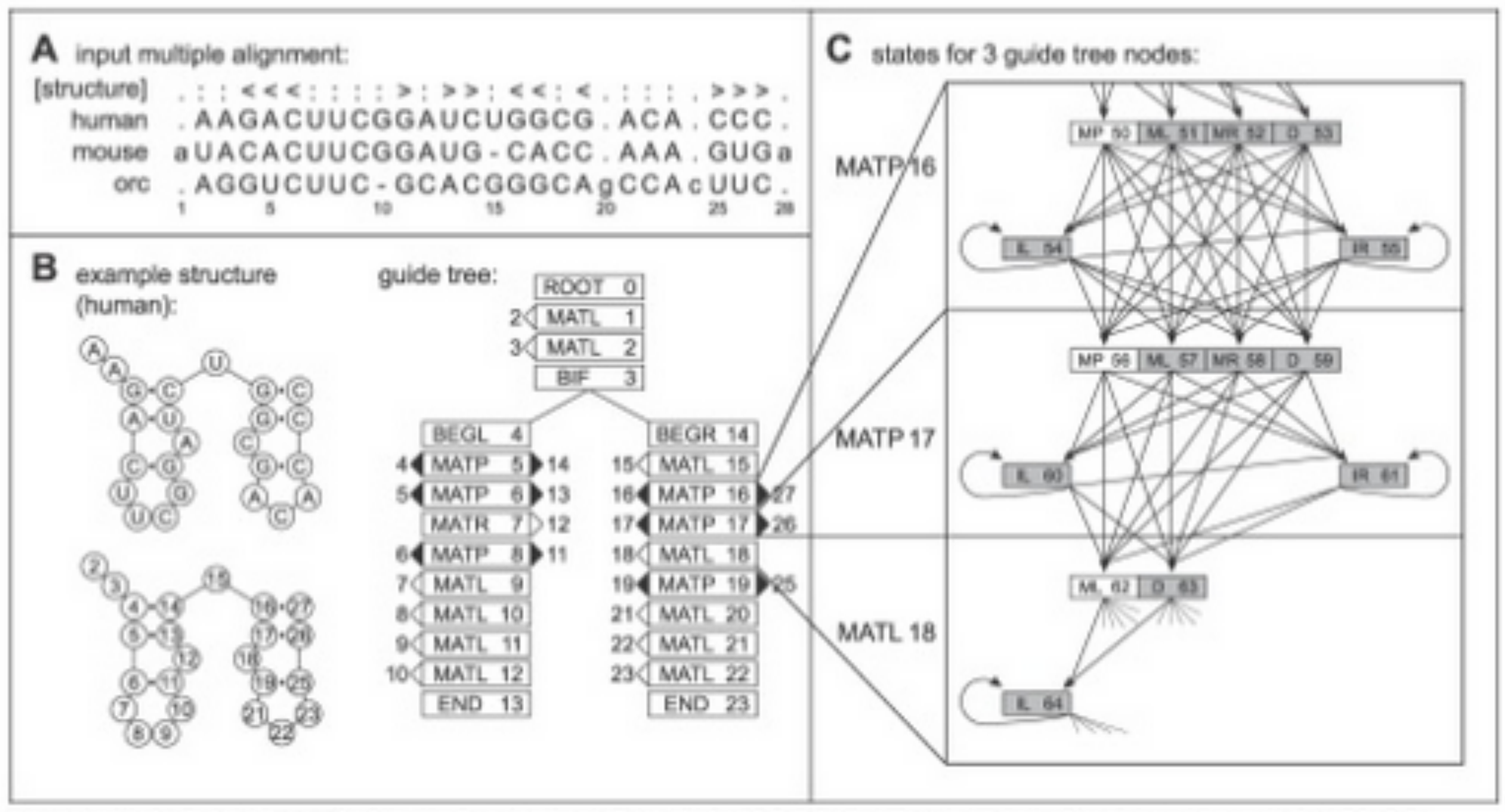

Outside
ACCACCUGCA AGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGU UGCAGG
i
Inside
j
(i,j)

Z_{ij} can be factored into three terms:

Inside, (i,j), Outside

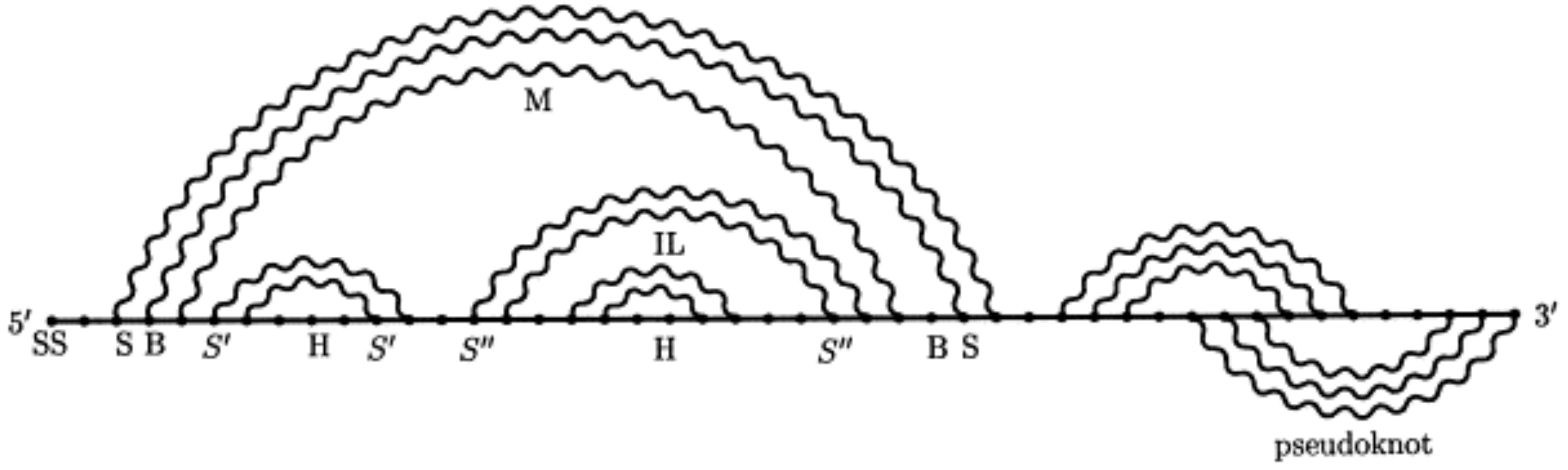
RNA family "profiles"

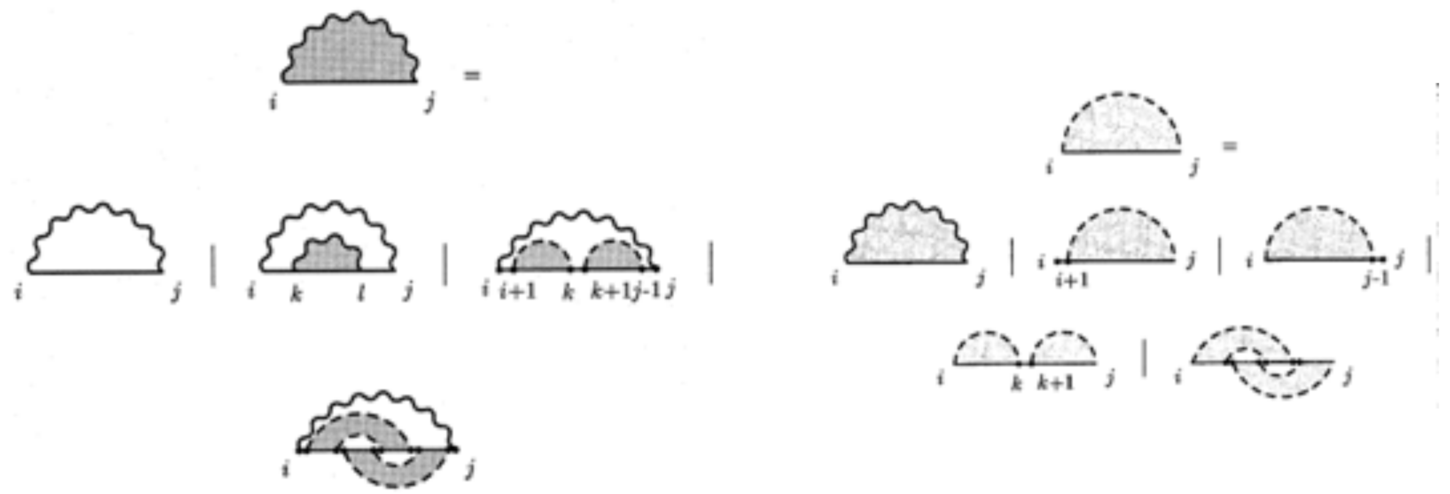
- The Infernal search tool, at the core of the RFam database, uses grammatical profiles whose underlying structure reflects the parse tree

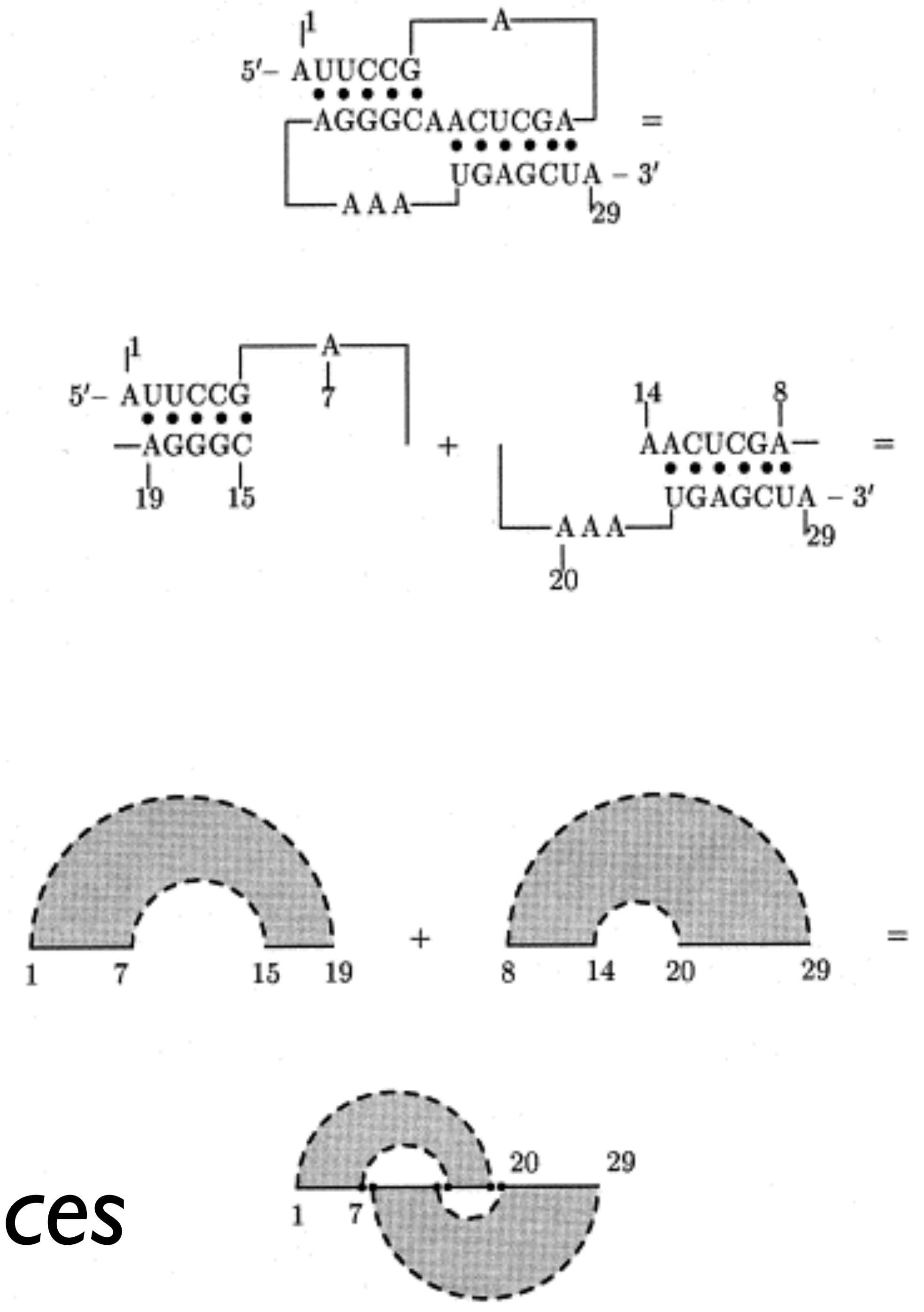


Nawrocki, Kolbe & Eddy, 2009

Pseudoknot extensions







*Key idea: iterate over **pairs** of subsequences*

$O(L^4)$ memory, $O(L^6)$ time

"Tree-Adjoining Grammars"

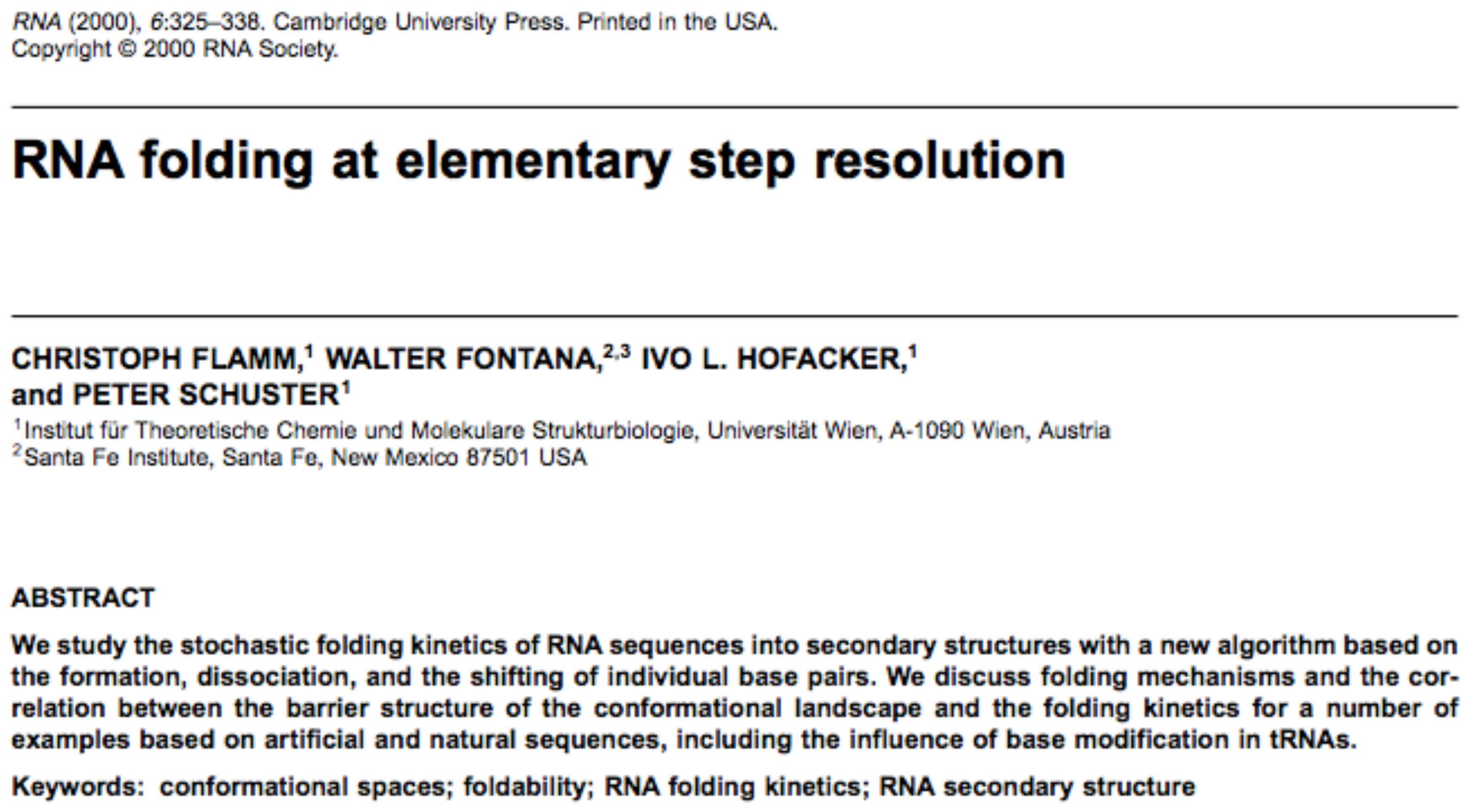
Rivas & Eddy, 1999

Chiang, Joshi & Searls, 2006

RNA folding kinetics

- General idea is to simulate diffusion in RNA structure space
- Structure is coarse-grained: usually represented as a set of basepairs (secondary structure)
    - Sometimes specify coordinates on a lattice
- Structure is progressively modified via a set of randomly sampled “moves” - a **Monte Carlo method**
- Can measure folding distance from structure A to structure B by starting many simulation runs from A and observing, on average, how many “moves” to reach B

kinfold



RNA (2000), *6*:325–338. Cambridge University Press. Printed in the USA.
Copyright © 2000 RNA Society.

RNA folding at elementary step resolution

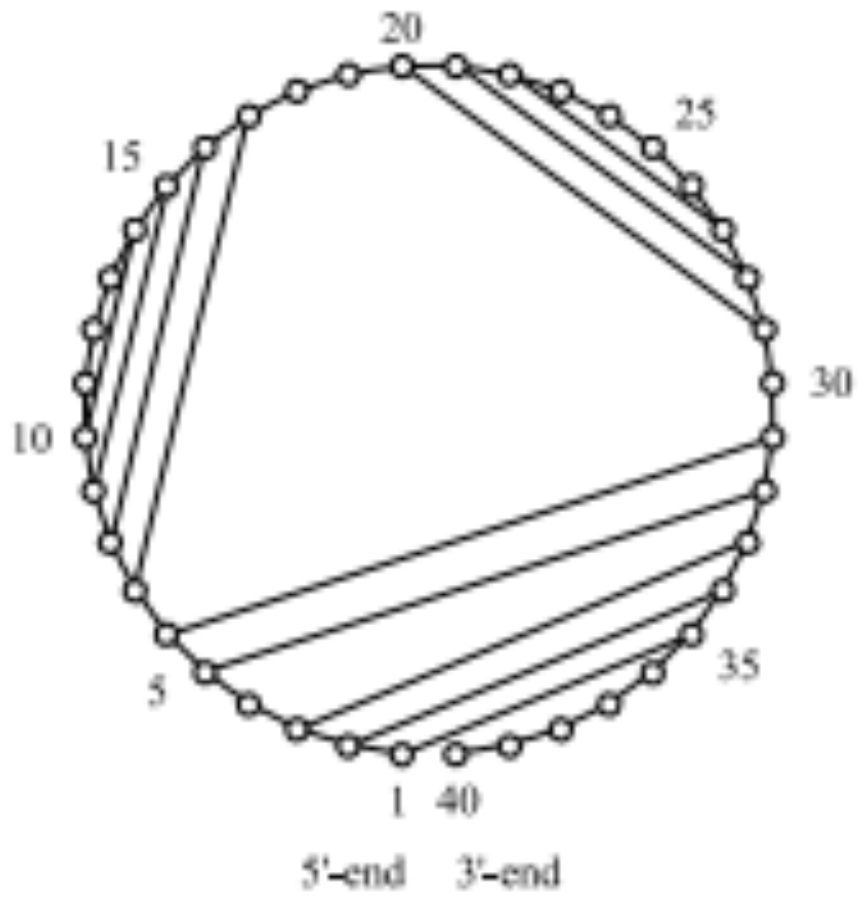
CHRISTOPH FLAMM,[1] WALTER FONTANA,[2,3] IVO L. HOFACKER,[1]
and PETER SCHUSTER[1]
[1]Institut für Theoretische Chemie und Molekulare Strukturbiologie, Universität Wien, A-1090 Wien, Austria
[2]Santa Fe Institute, Santa Fe, New Mexico 87501 USA

ABSTRACT

We study the stochastic folding kinetics of RNA sequences into secondary structures with a new algorithm based on the formation, dissociation, and the shifting of individual base pairs. We discuss folding mechanisms and the correlation between the barrier structure of the conformational landscape and the folding kinetics for a number of examples based on artificial and natural sequences, including the influence of base modification in tRNAs.

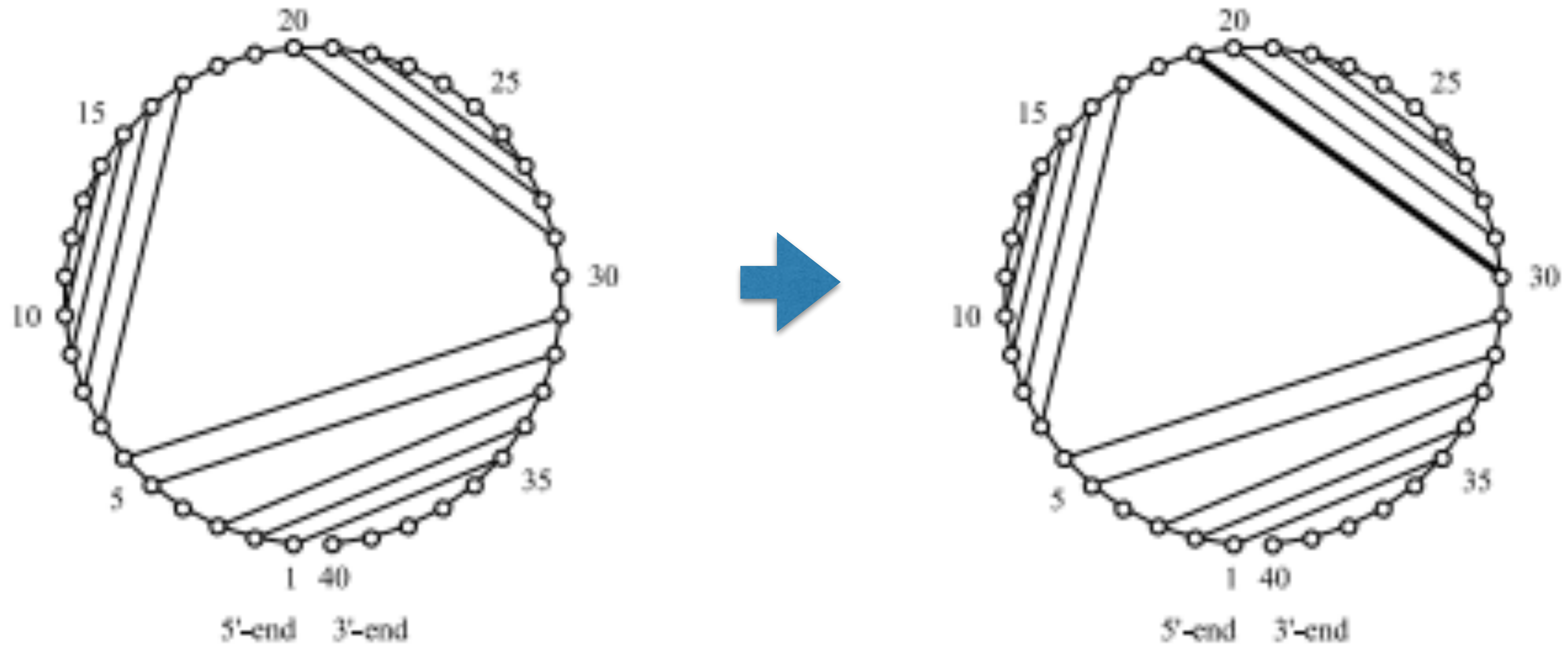
Keywords: conformational spaces; foldability; RNA folding kinetics; RNA secondary structure

Circular representation



```
(((•((((((•••))))••(((••••)))•)))))•••••
```

Base pair formation



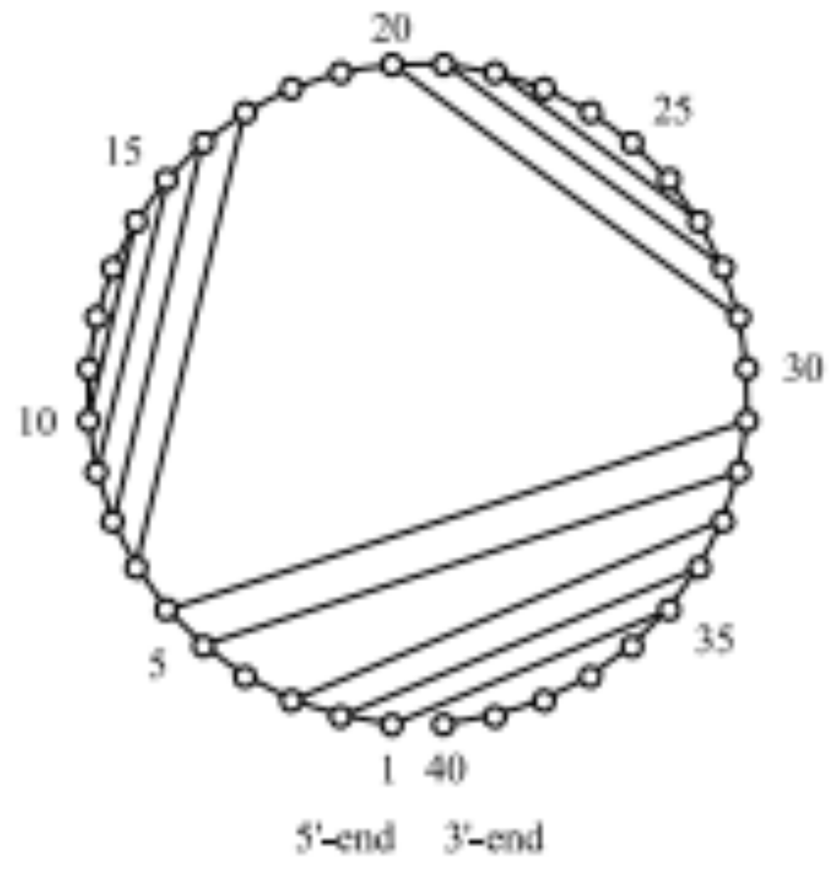
(((•((((((•••))))••(((••••)))•)))))•••••

(((•((((((•••))))••(((••••)))•)))))•••••

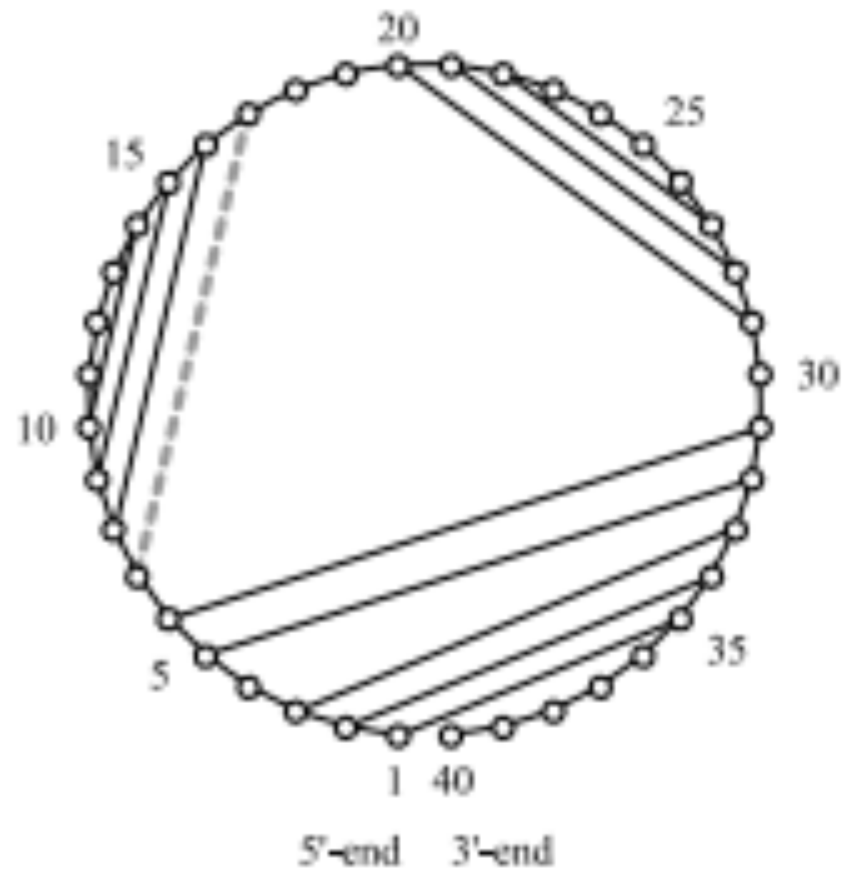
↓

(((•((((((•••))))•((((••••))))))))))•••••

Base pair removal



(((•((((((•••))))••(((••••)))•)))))•••••

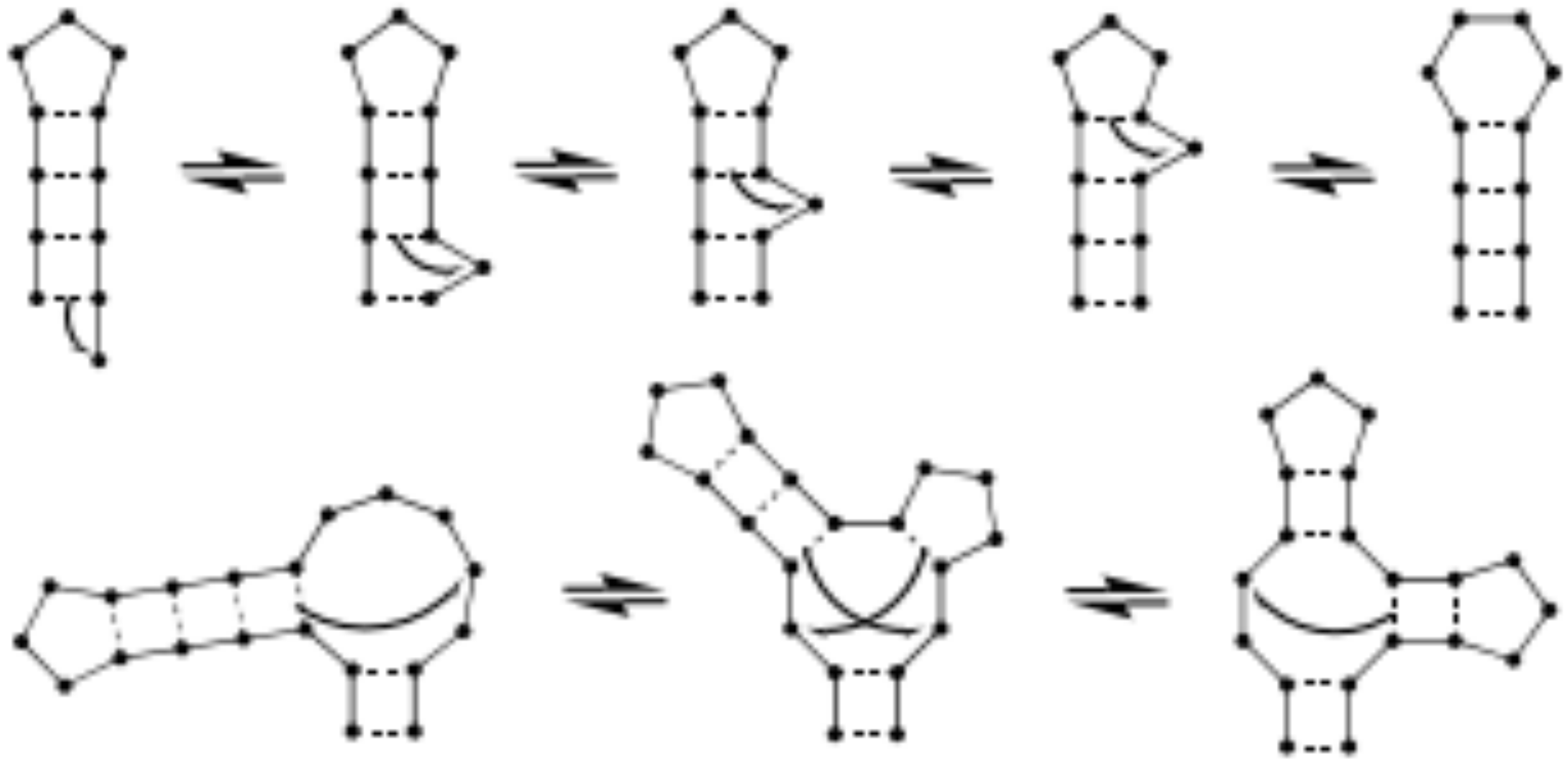


(((•((((((•••))))••(((••••)))•)))))•••••

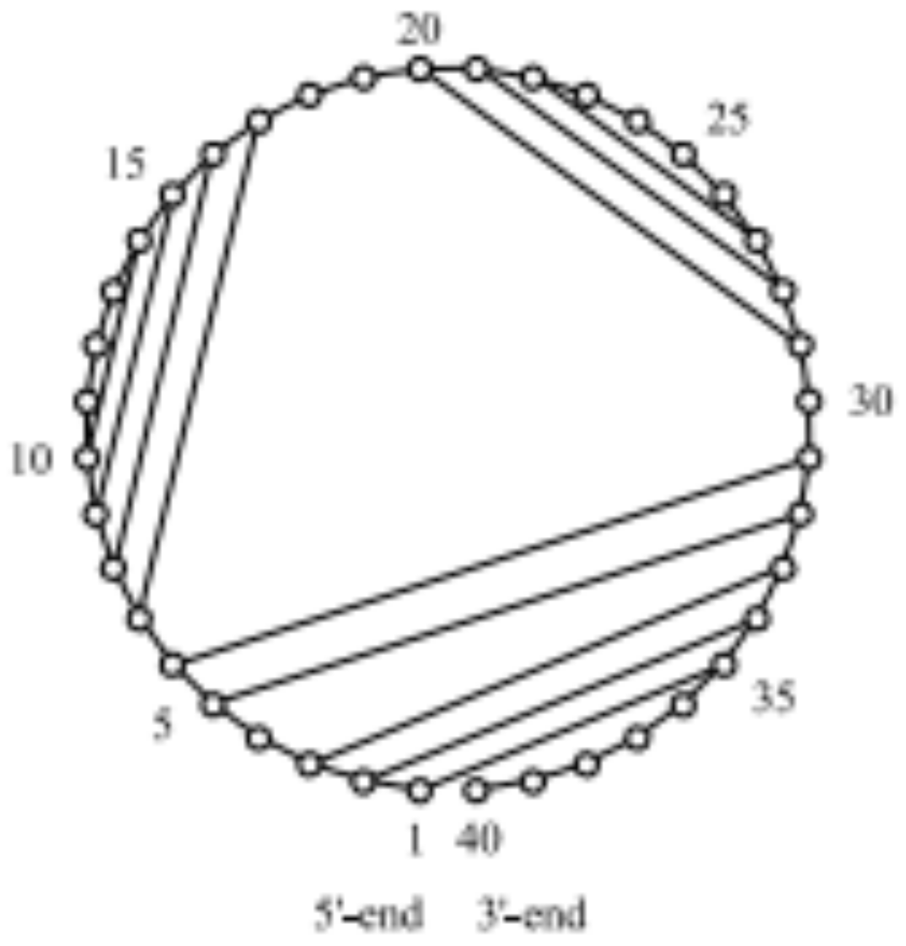
↓

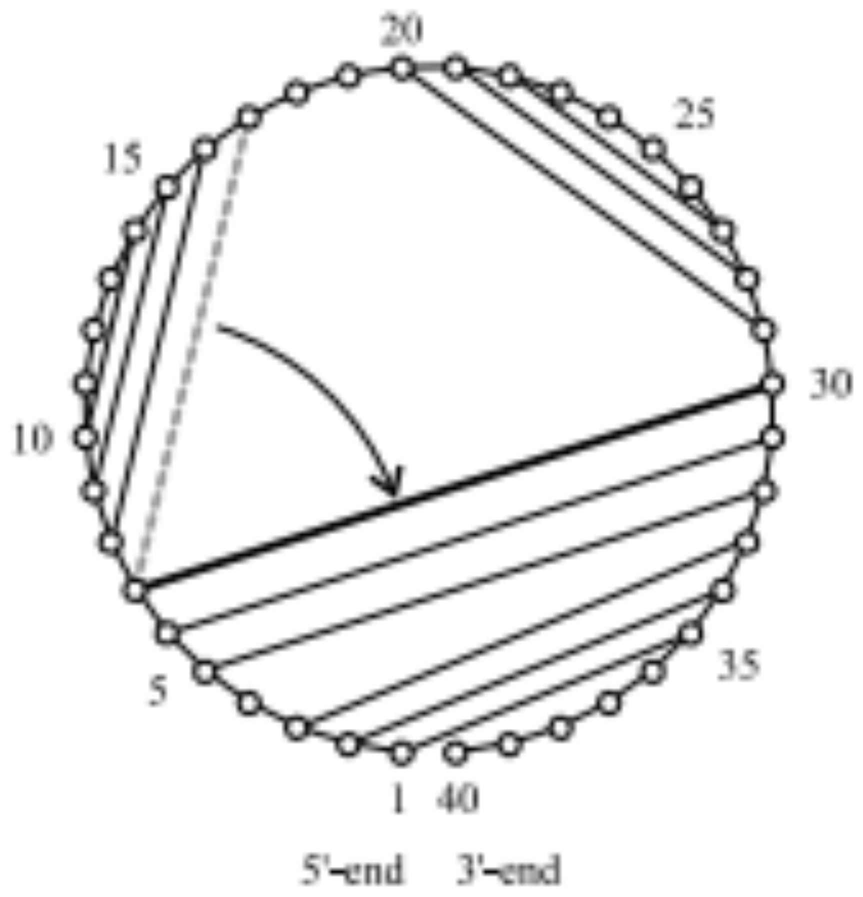
(((•((•(((•••)))•••(((••••)))•)))))•••••

Defect diffusion and helix morphing



Base pair shift (#1)





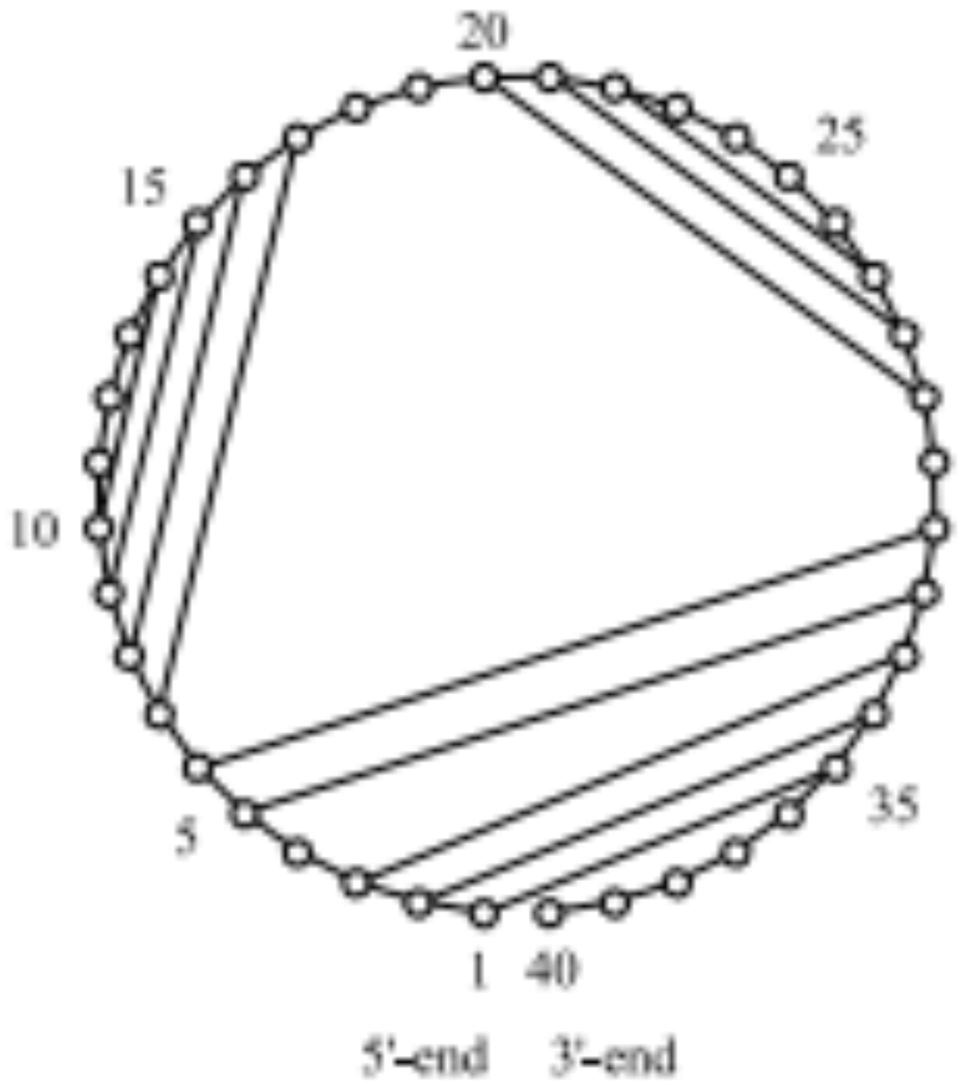
(((•((((((•••))))••(((••••)))•)))))•••••

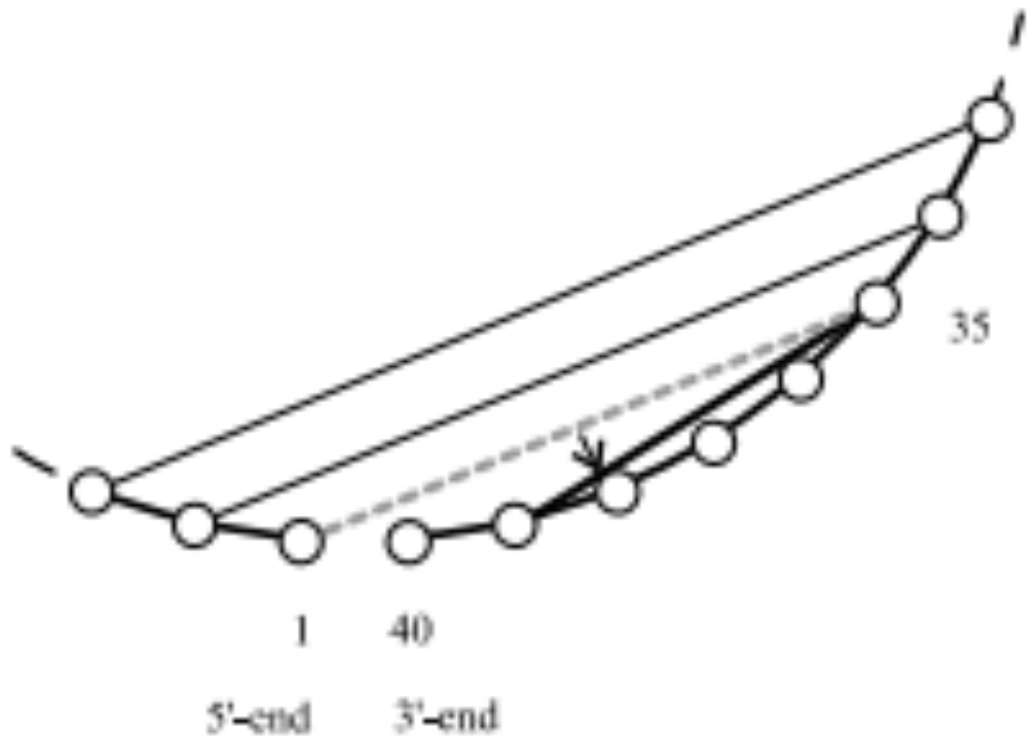
(((•((((((•••))))••(((••••)))•)))))•••••

↓

(((•((((((•••)))•••(((••••)))))))))•••••

Base pair shift (#2)





(((•((((((•••))))••(((••••)))•)))))•••••

(((•((((((•••))))••(((••••)))•)))))•••••

↓

•((•((((((•••))))••(((••••)))•))))(•••)•

Thermodynamic simulations

Rates of "moves" between states i & j
must satisfy "detailed balance"

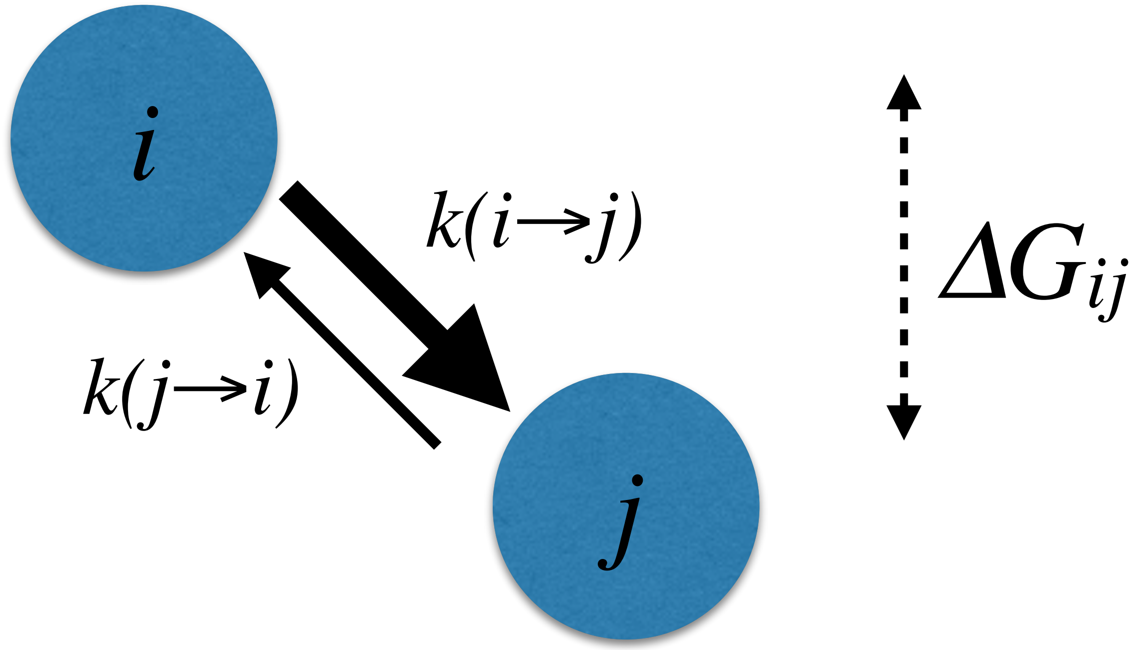
$$\frac{k_{ij}}{k_{ji}} = \exp(-\Delta G_{ij}/RT) = \exp\left(-(\Delta G_j^0 - \Delta G_i^0)/RT\right),$$

Thermodynamic simulations

Rates of "moves" between states i & j
must satisfy "detailed balance"

$$\frac{k_{ij}}{k_{ji}} = \exp(-\Delta G_{ij}/RT) = \exp\left(-(\Delta G_j^0 - \Delta G_i^0)/RT\right),$$

At equilibrium, net flux between any two states is zero



$$P(i)k(i \rightarrow j) = P(j)k(j \rightarrow i)$$

$$P(i) \propto \exp(-\Delta G_i/RT)$$

Thermodynamic simulations

Rates of "moves" between states i & j must satisfy "detailed balance"

$$\frac{k_{ij}}{k_{ji}} = \exp(-\Delta G_{ij}/RT) = \exp\left(-(\Delta G_j^0 - \Delta G_i^0)/RT\right),$$

e.g. (Metropolis & Hasting, 1953)

$$k_{ij} = \begin{cases} \exp(-\Delta G_{ij}/RT) & \text{if } \Delta G_j^0 > \Delta G_i^0, \\ 1 & \text{if } \Delta G_j^0 < \Delta G_i^0, \end{cases}$$

Thermodynamic simulations

Rates of "moves" between states i & j must satisfy "detailed balance"

$$\frac{k_{ij}}{k_{ji}} = \exp(-\Delta G_{ij}/RT) = \exp\left(-(\Delta G_j^0 - \Delta G_i^0)/RT\right),$$

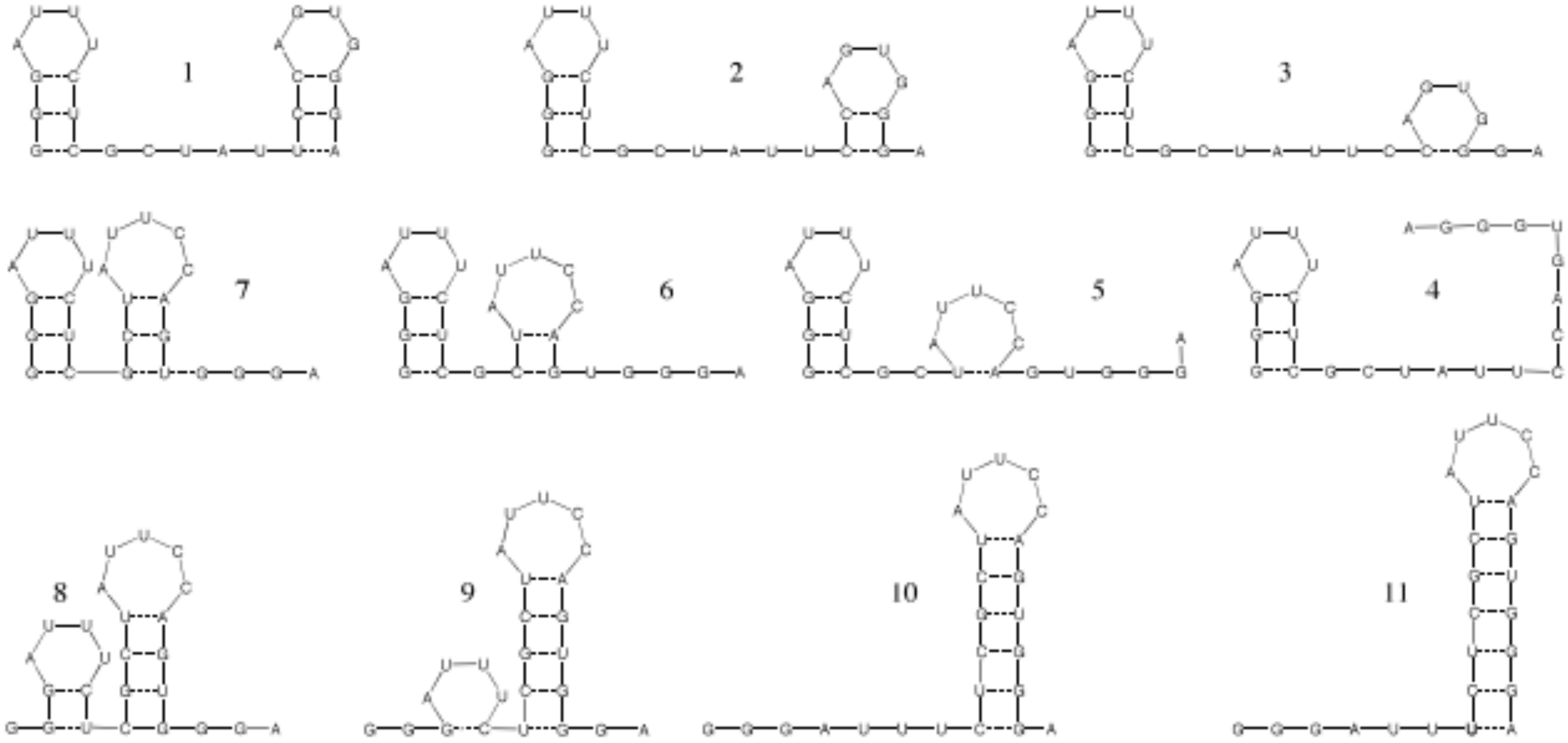
e.g. (Metropolis & Hasting, 1953)

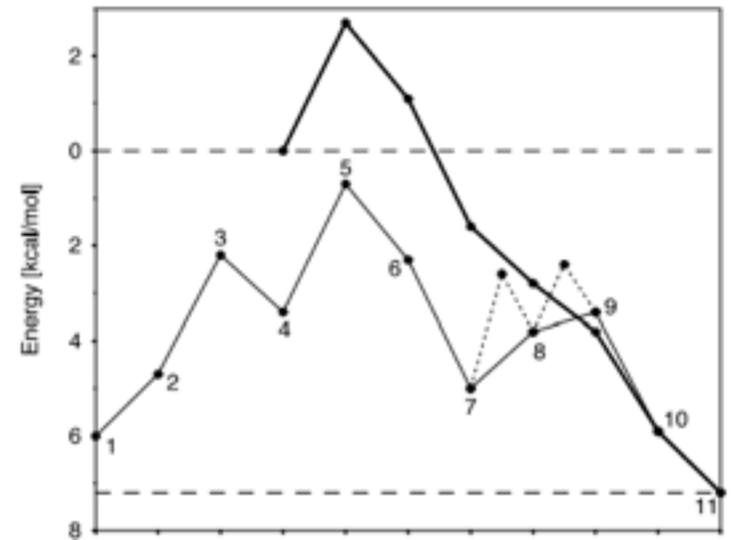
$$k_{ij} = \begin{cases} \exp(-\Delta G_{ij}/RT) & \text{if } \Delta G_j^0 > \Delta G_i^0, \\ 1 & \text{if } \Delta G_j^0 < \Delta G_i^0, \end{cases}$$

or (Kawasaki, 1996)

$$k_{ij} = \exp(-\Delta G_{ij}/2RT).$$

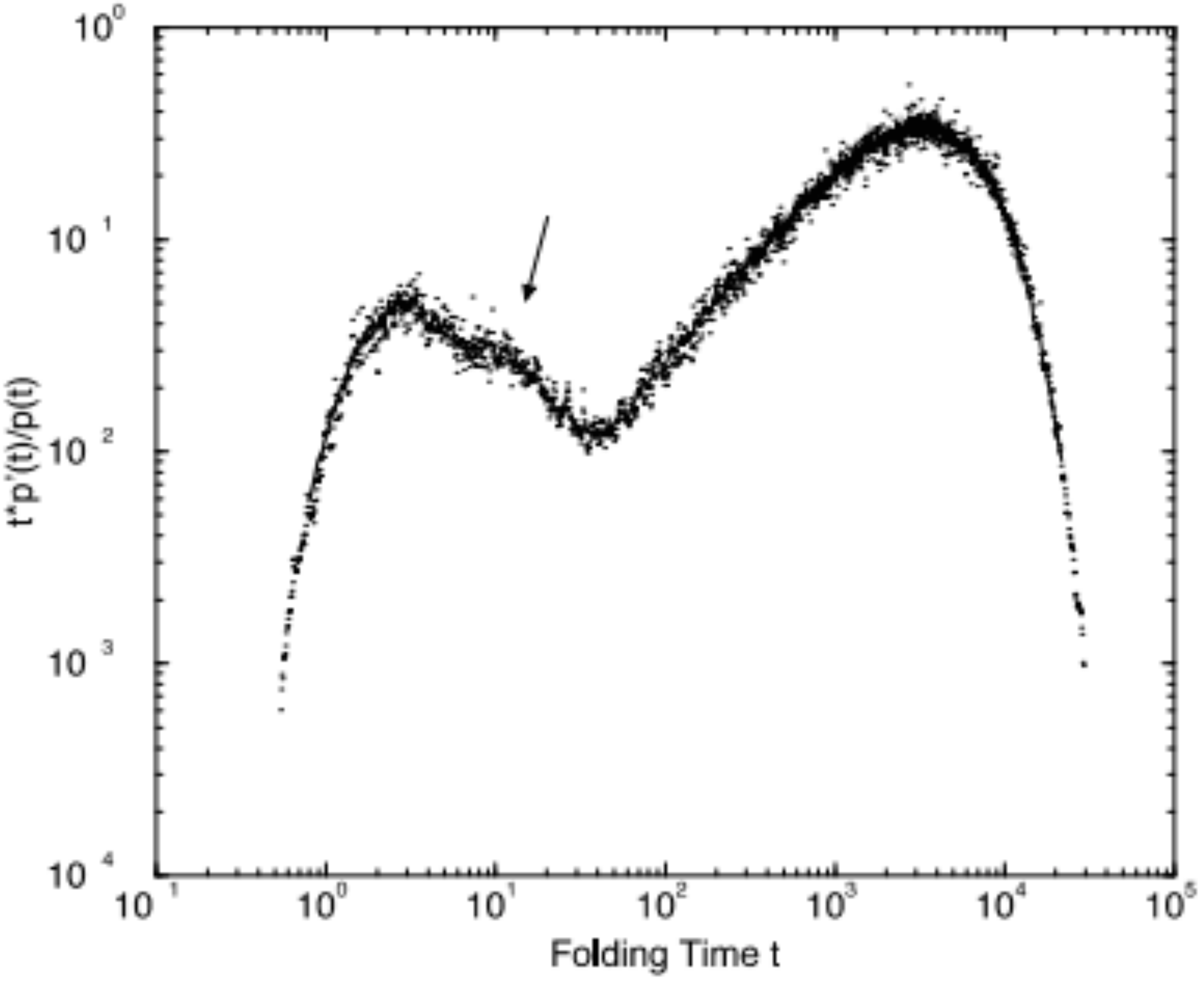
Escape from conformational trap





Dashed lines indicate energy barriers in the absence of shift moves. Bold line corresponds to fast folding path following simple zipper from open chain

Distribution of folding times



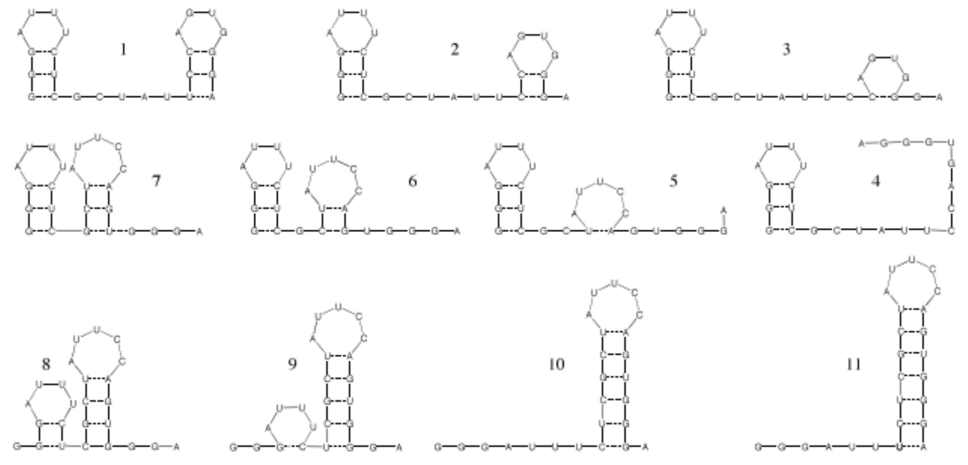
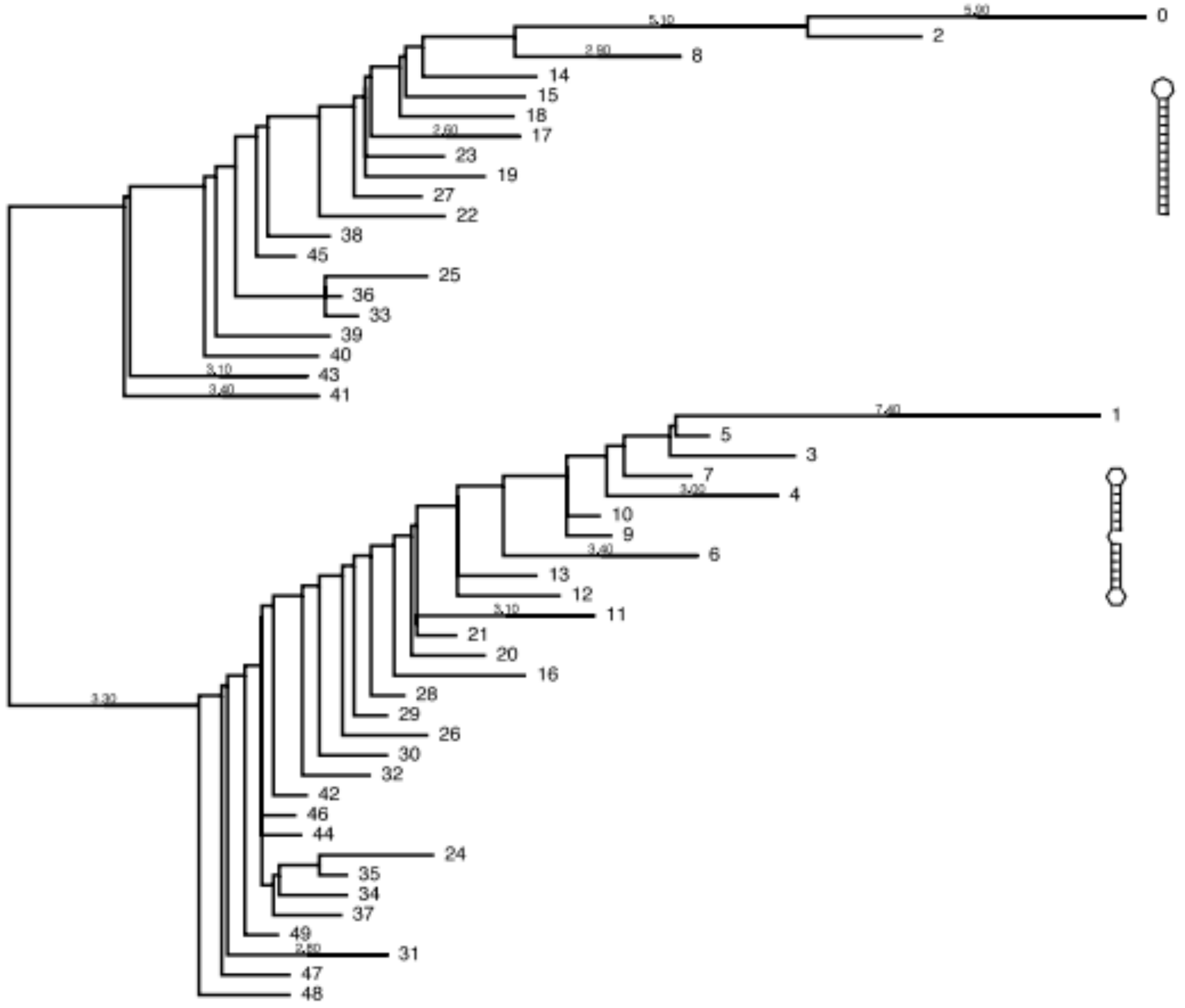
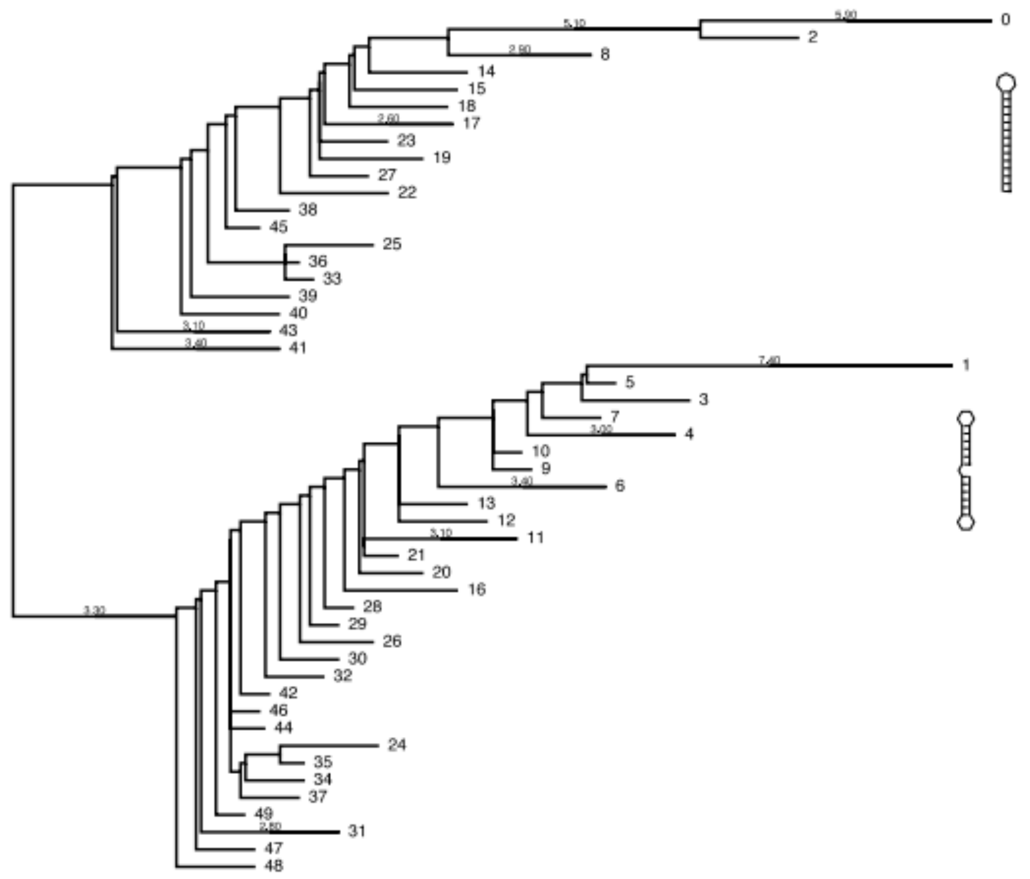


FIGURE 7. The folding characteristic of a small RNA. The folding characteristic $\chi(t)$ of the sequence I = (GGGAUUUCUCGCUAUUC CAGUGGGA) is plotted on a logarithmic time scale. The points correspond to individual trajectories and show some scatter that has been smoothed by a moving average in the solid line. The curve shows two distinct humps corresponding to different folding paths (direct or via S_1). A less prominent pathway appears as a shoulder on the right flank of the first hump (arrow).

Two competing structures



Two competing structures



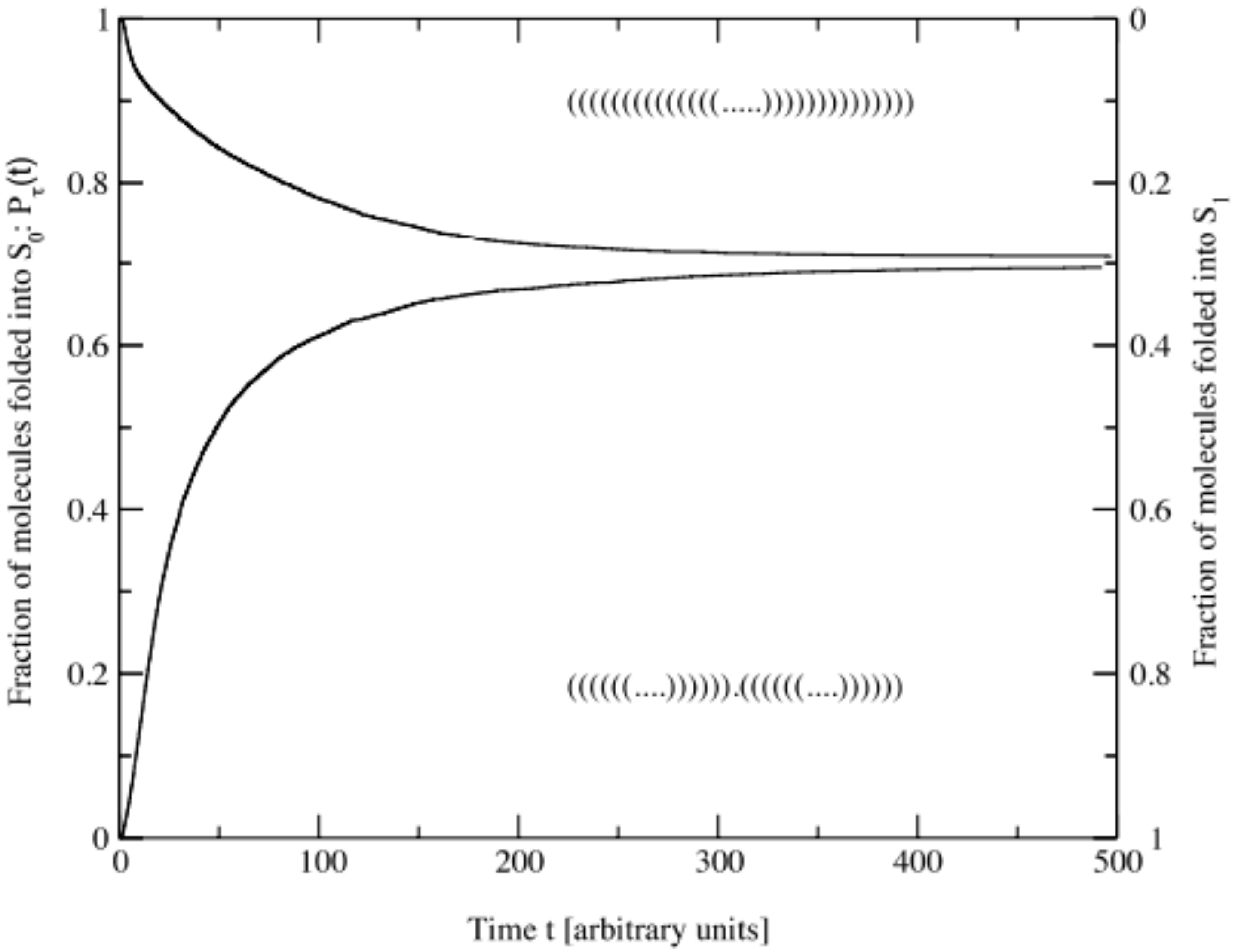


FIGURE 9. The fraction of folded molecules. The two curves show the abundance of conformations, S_0 (left ordinate), and S_1 (right ordinate, upside down). As time progresses, the two conformations of low free energy, S_0 and S_1, increase in frequency at the expense of all other conformations. The final ratio of S_0/S_1 is close to 1:2 in agreement with the number of nucleation centers in the two conformations.

Further work in RNA kinetics

- Isambert, 2003: **kinefold**
  Models RNA helices as stiff rods and single-stranded regions as polymer springs, integrates out kinetic traps
- Senter & Clote, 2015: **Hermes**
  Uses Fast Fourier Transform and matrix algebra to compute mean first passage times between structures
- Jost & Everaers, 2010
  Lattice-based model

…<u>not</u> a complete list!

RNA inverse folding

- Problem: given a target structure, find an RNA sequence whose minimum free energy structure is [close to] that structure

```
....<<<<<<<<<<<<......>>>>>...<<<<<.....>>>>><<<<........>>>>.>>>>>>>
```

RNA inverse folding

- Problem: given a target structure, find an RNA sequence whose minimum free energy structure is [close to] that structure

```
....<<<<<<<<<<<<......>>>>>...<<<<<.....>>>>><<<<........>>>>.>>>>>>>
```

$$\downarrow$$

```
ACCACCUGCUCAGGCCUUAGCUGGCCUCGAGAGGGCGGUUCCCUCCGCGAGGCGAUGCGCGUGUGCAGG
```

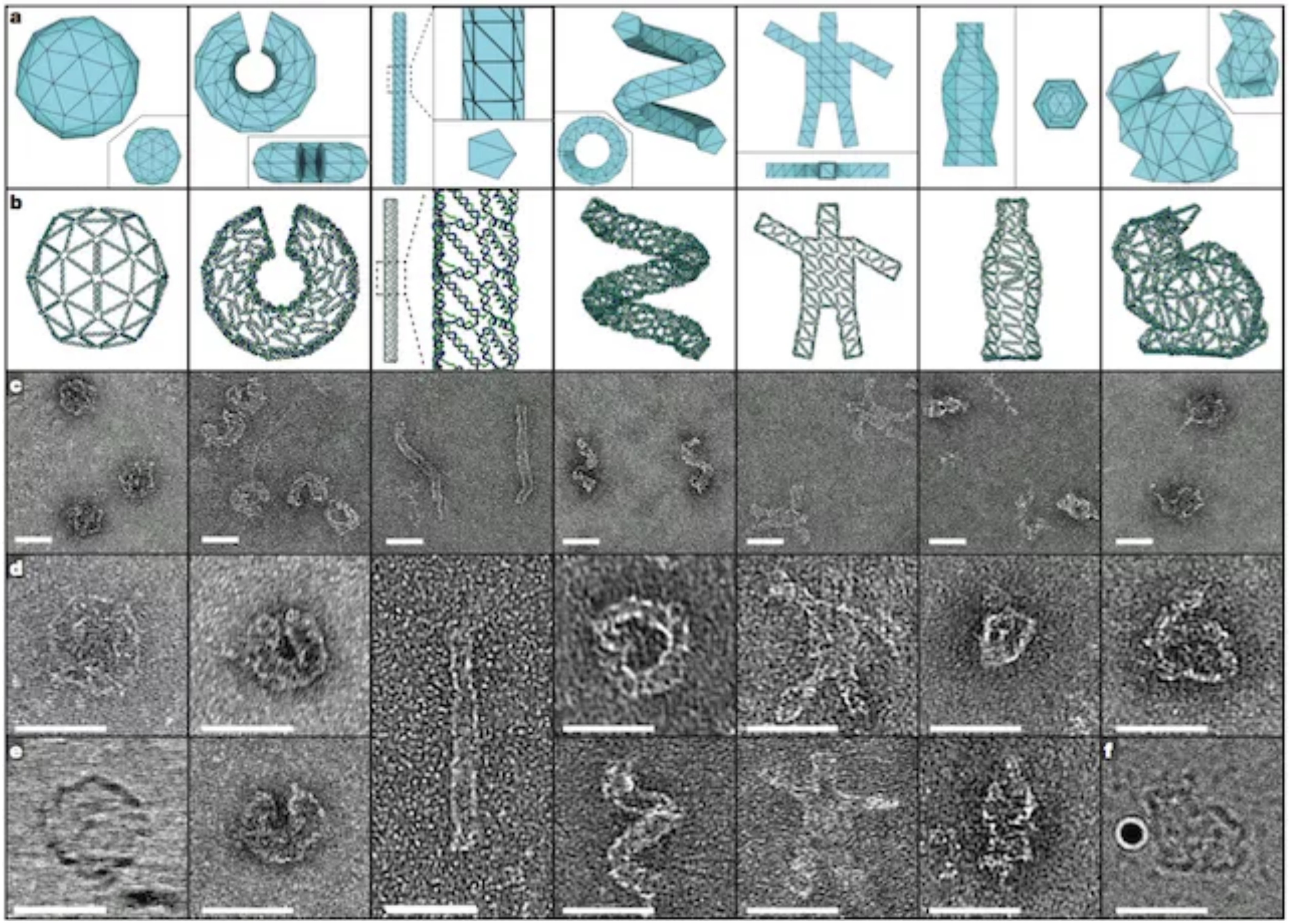
RNA inverse folding

- RNAinverse
  (Hofacker et al, Fast folding and comparison of RNA secondary structures, 1994)
- RNA-SSD
  (Andronescu et al, A new algorithm for RNA secondary structure design, 2004)
- INFO-RNA
  (Busch et al, Info-rna, a fast approach to inverse RNA folding, 2006)
- Inv
  (Gao et al, Inverse folding of RNA pseudoknot structures, 2010)
- MODENA
  (Taneda, MODENA: A multi-objective RNA inverse folding, 2011)
- NUPACK-DESIGN
  (Zadeh et al, Nucleic acid sequence design via efficient ensemble defect optimization, 2011)
- RNAiFold
  (Garcia-Martin et al, RNAiFOLD: a constraint programming algorithm for RNA inverse folding and molecular design, 2013)

Nucleic acid complexes

- Mathews, Burkard, Freier, Wyatt, Turner. 1999.
  Predicting oligonucleotide affinity to nucleic acid targets.
- Zhang, Chen. 2001.
  A three-dimensional statistical mechanical model of folding double-stranded chain molecules.
- Andronescu, Zhang, Condon. 2005.
  Secondary structure prediction of interacting RNA molecules.
- Dirks, Bois, Schaeffer, Winfree, Pierce. 2007.
  Thermodynamic analysis of interacting nucleic acid strands.
- Muckstein, Tafer, Hackermuller, Bernhart, Stadler, Hofacker. 2006.
  Thermodynamics of RNA–RNA binding.
- Mneimneh, Ahmed. 2015.
  Multiple RNA interaction: beyond two.

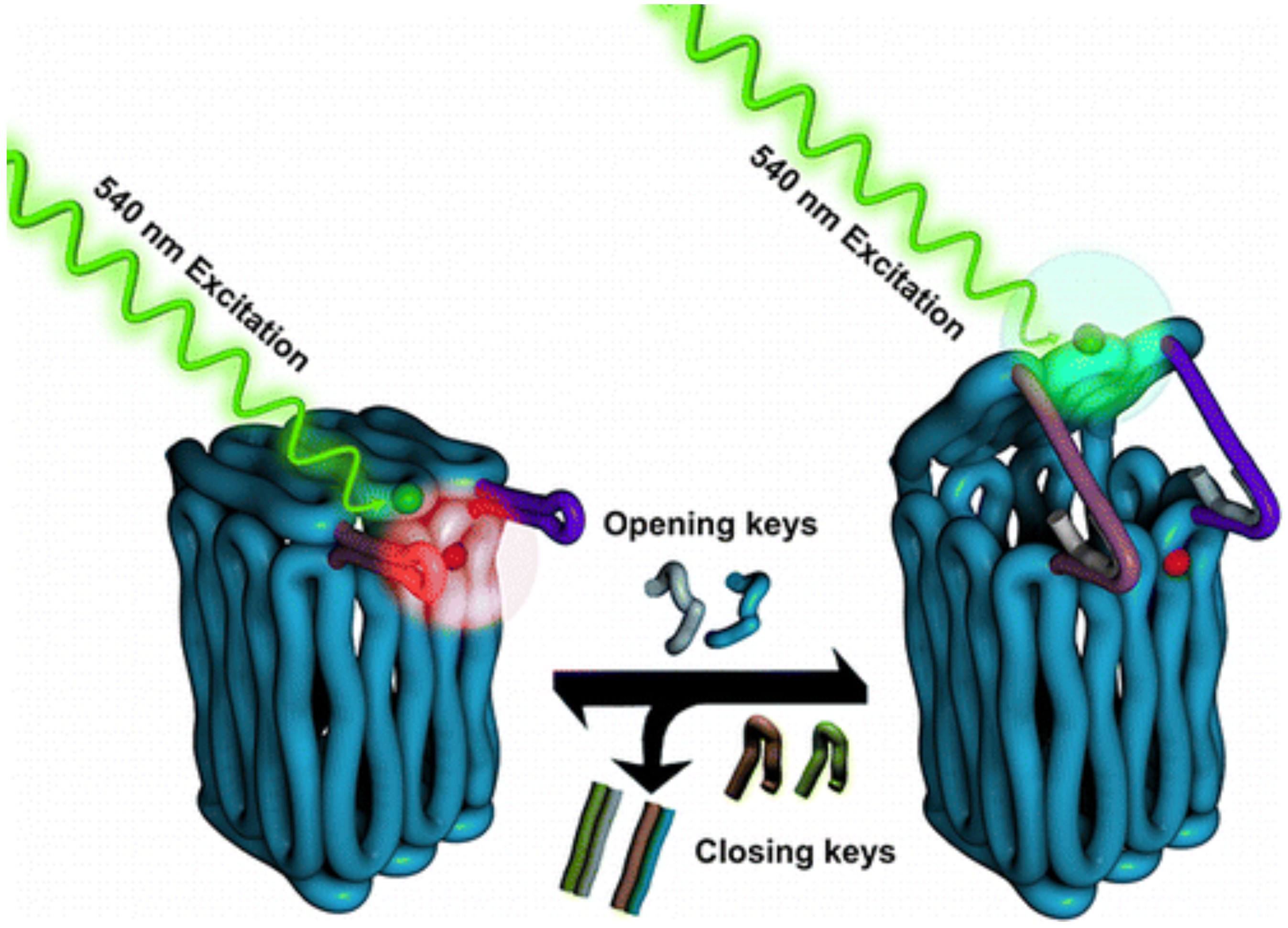
DNA origami



Högberg et al, 2015
DNA rendering of polyhedral meshes at the nanoscale

Why DNA origami?

- Self-assembly
- Addressable surfaces
- Molecular programming
- DNA nanorobots…



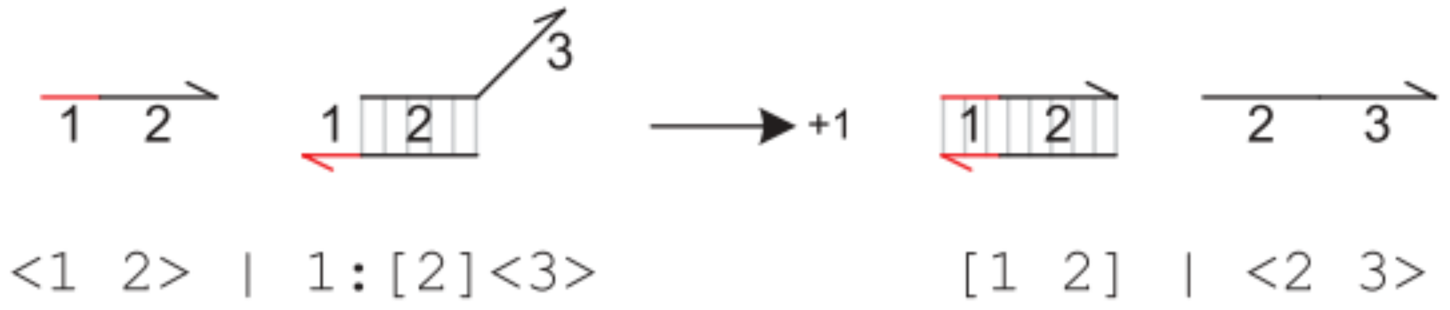
Switchable DNA structures gated by nucleic acid logic (Zadegan, Jepsen et al, 2012; Douglas, Bachelet & Church, 2012)

DNA nanorobot & circuit design

- Phillips and Cardelli, 2009
  A programming language for composable DNA circuits

- Jabbari et al, 2015
  Computational approaches to nucleic acid origami

- Kahramanoğulları and Cardelli, 2015
  Gener: a minimal programming module for chemical controllers based on DNA strand displacement



Summary: DNA/RNA design tools

- Structure prediction ("RNA folding") typically uses **dynamic programming** (Nussinov, grammars, etc)
- Kinetics simulations use **Monte Carlo methods**
- Sequence design ("inverse folding") uses various **optimization methods** (greedy search, stochastic search, constraint programming…)
- Methods for nucleic acid complexes are relatively new… but highly relevant to bioeng/nanotech