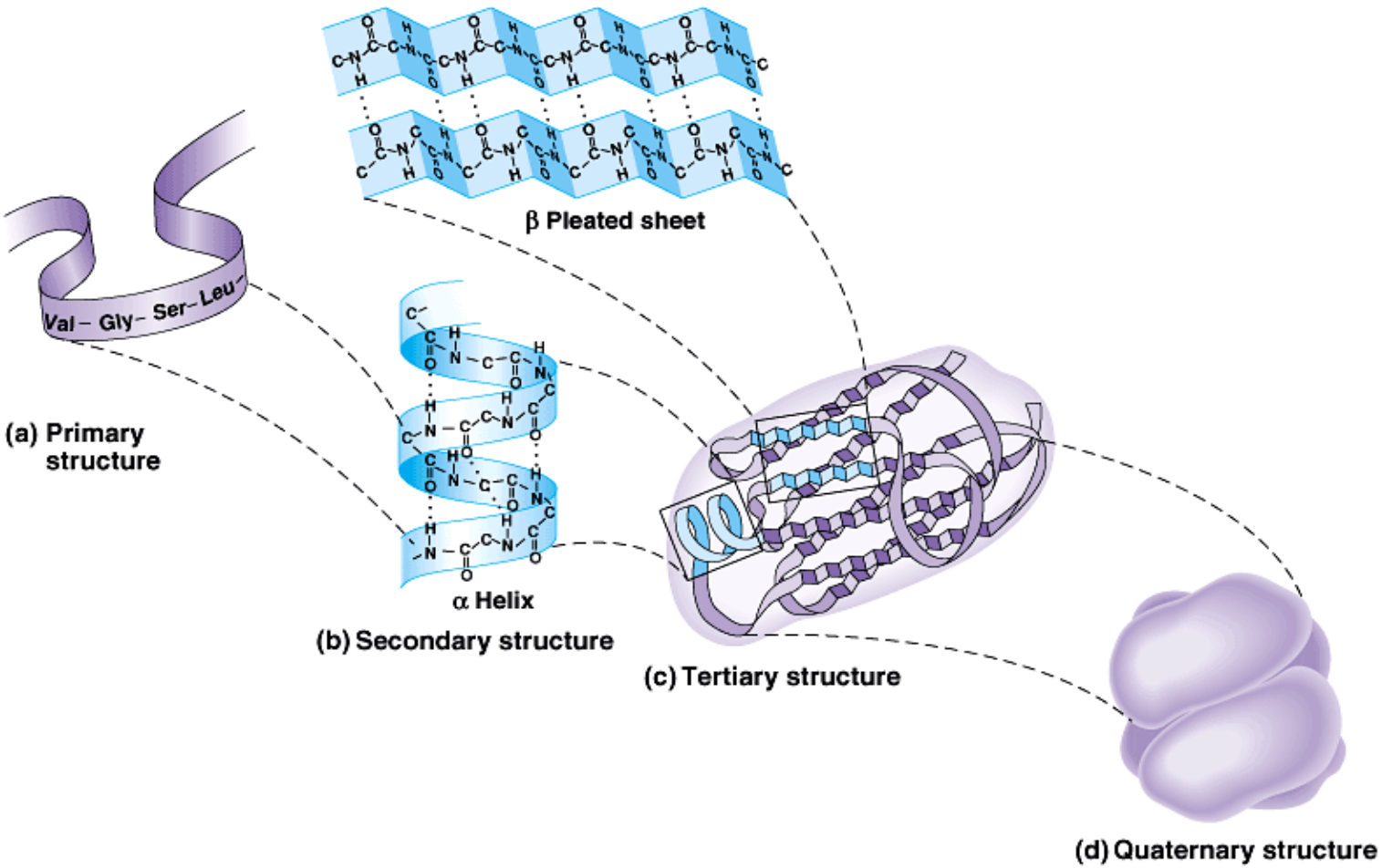
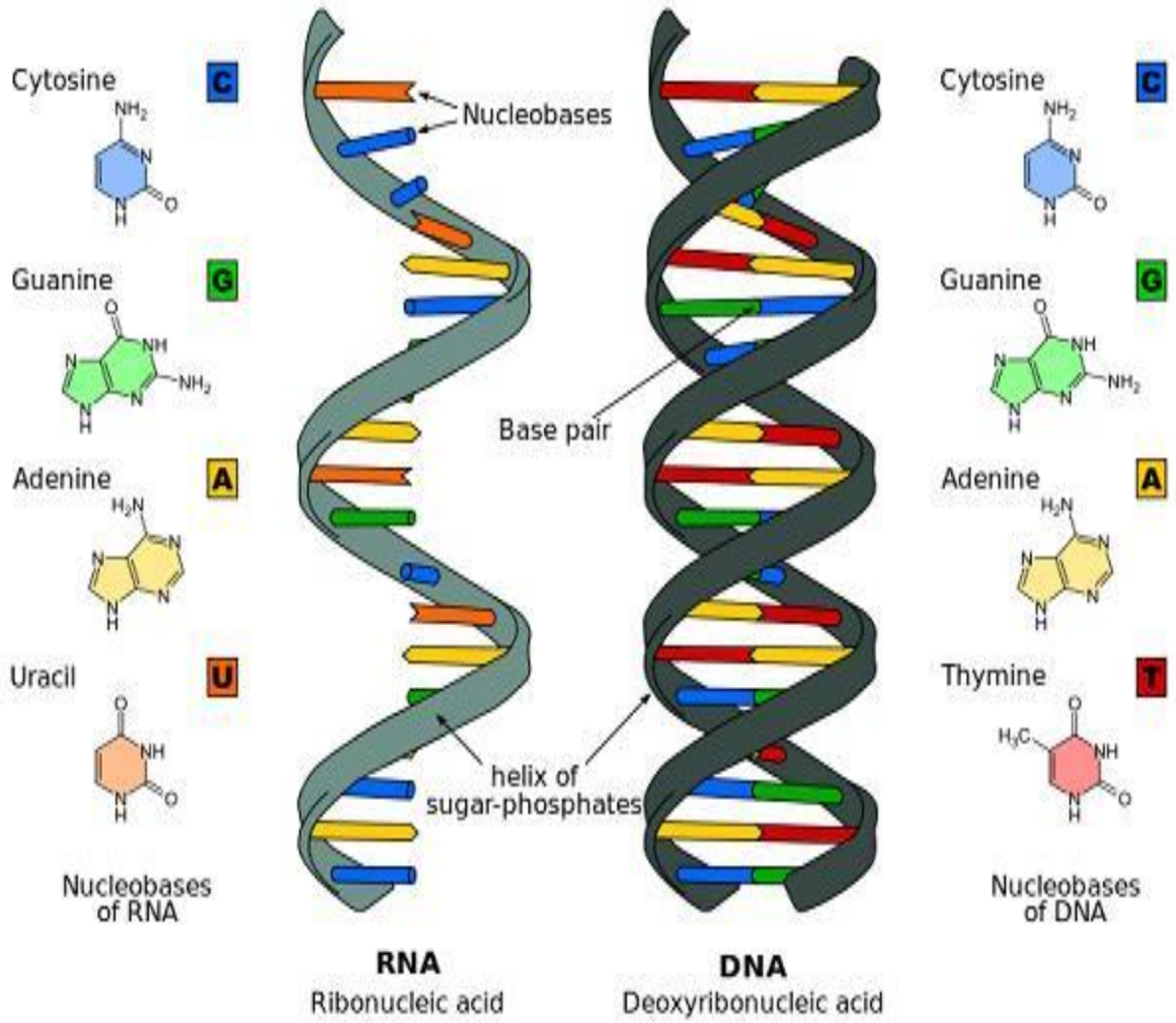
Nucleic Acid Secondary Structure

Structure of protein

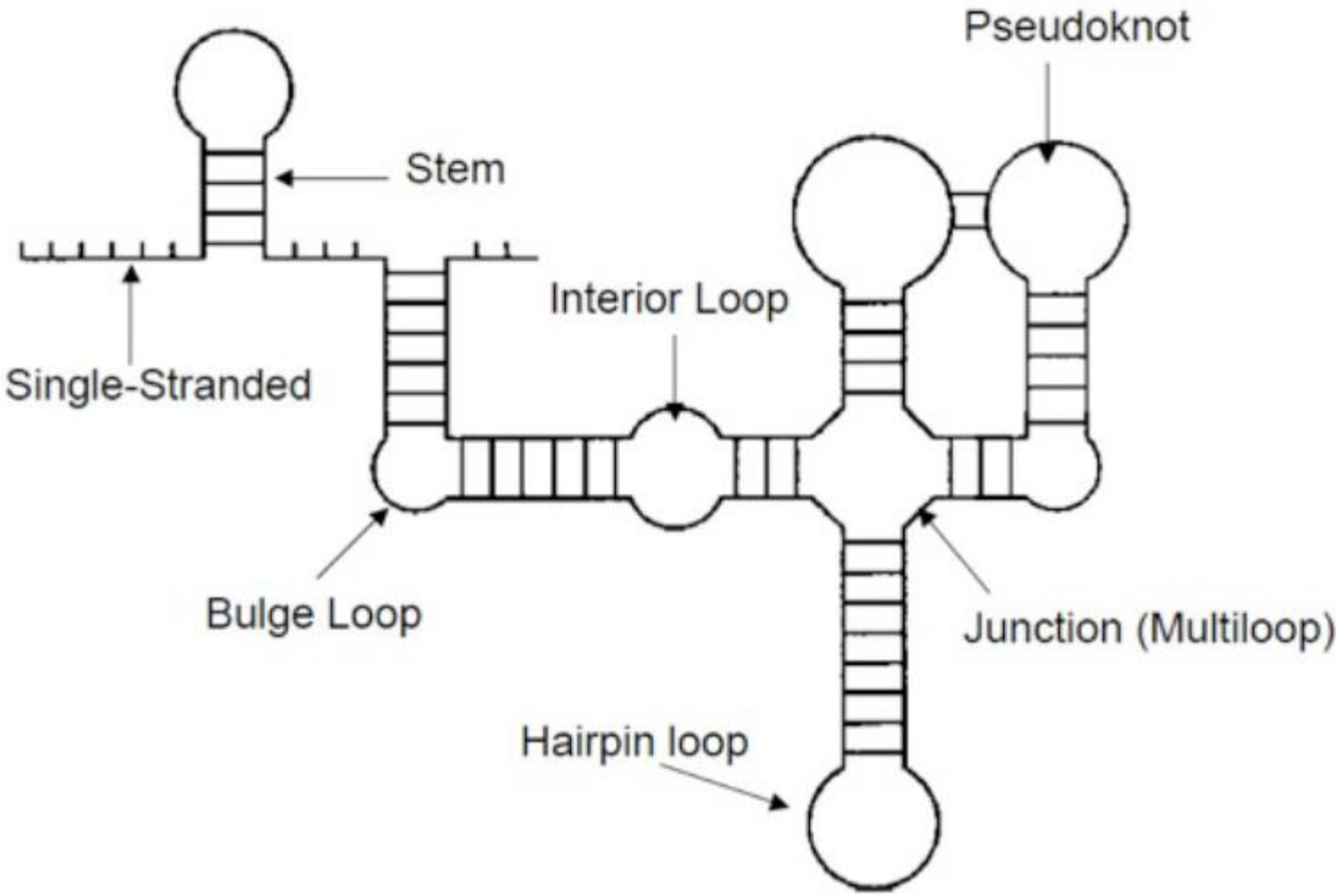


- The secondary structure of a nucleic acid molecule refers to the base paring interactions within a single molecule or set of interacting molecules.
- DNA molecules exist predominantly in the double stranded form.
- On the other hand, RNA molecules exist mostly as single strands.
- However, the single strand of RNA can twist and fold back on itself to form a well defined three dimensional structure.

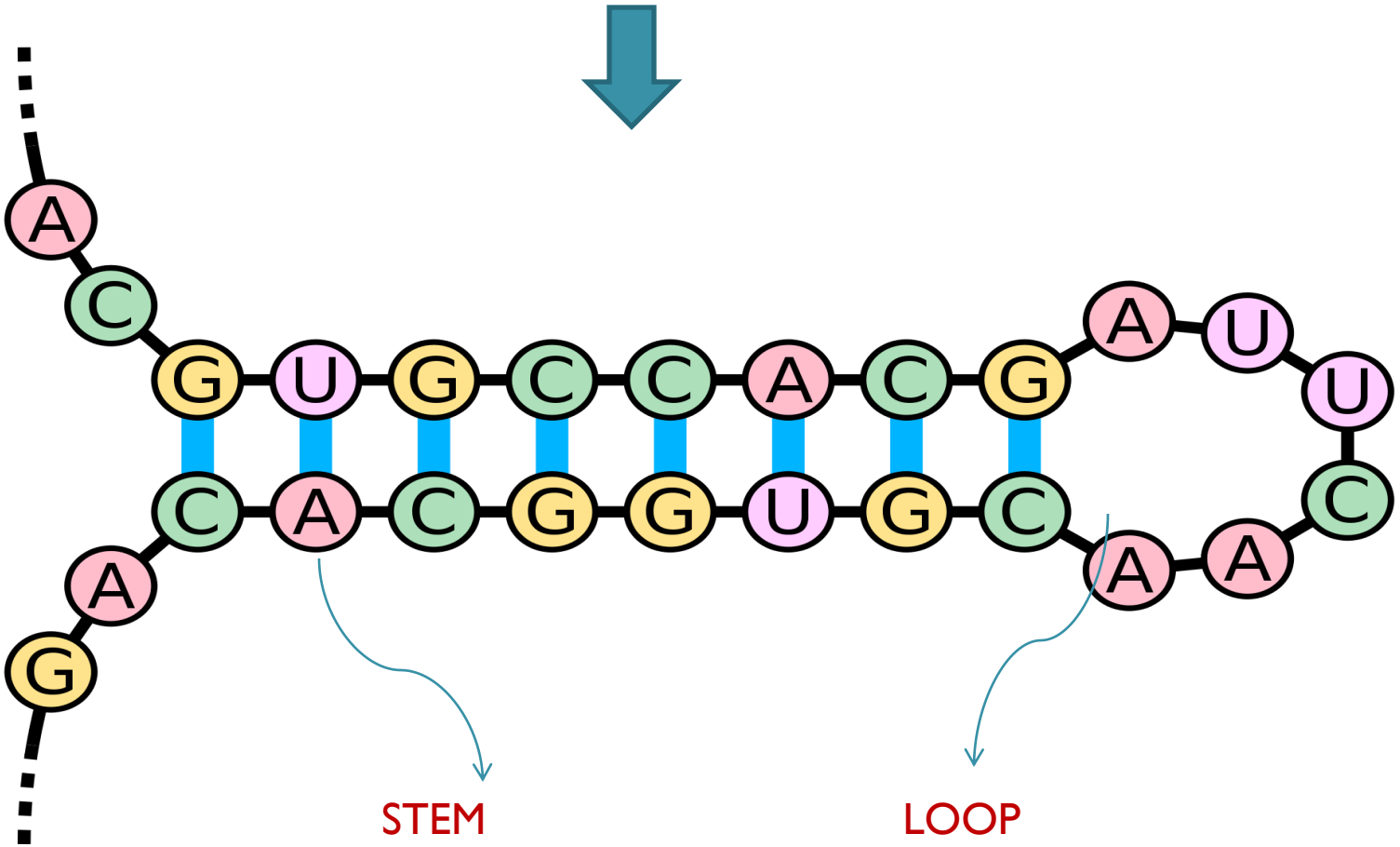

Cytosine
C
Guanine
G
Adenine
A
Uracil
U
Nucleobases of RNA
Nucleobases
Base pair
helix of sugar-phosphates
RNA
Ribonucleic acid
DNA
Deoxyribonucleic acid
Cytosine
C
Guanine
G
Adenine
A
Thymine
T
Nucleobases of DNA

- Perhaps the most common structure that RNA molecules fold into is called Stem Loop.
- This structure is formed when the ribonucleic acid folds onto itself to form a double helical structure that consist of two complementary sequences.
- It also commonly contains mismatched nucleotides that bulge out of the structure.

- Secondary structure elements



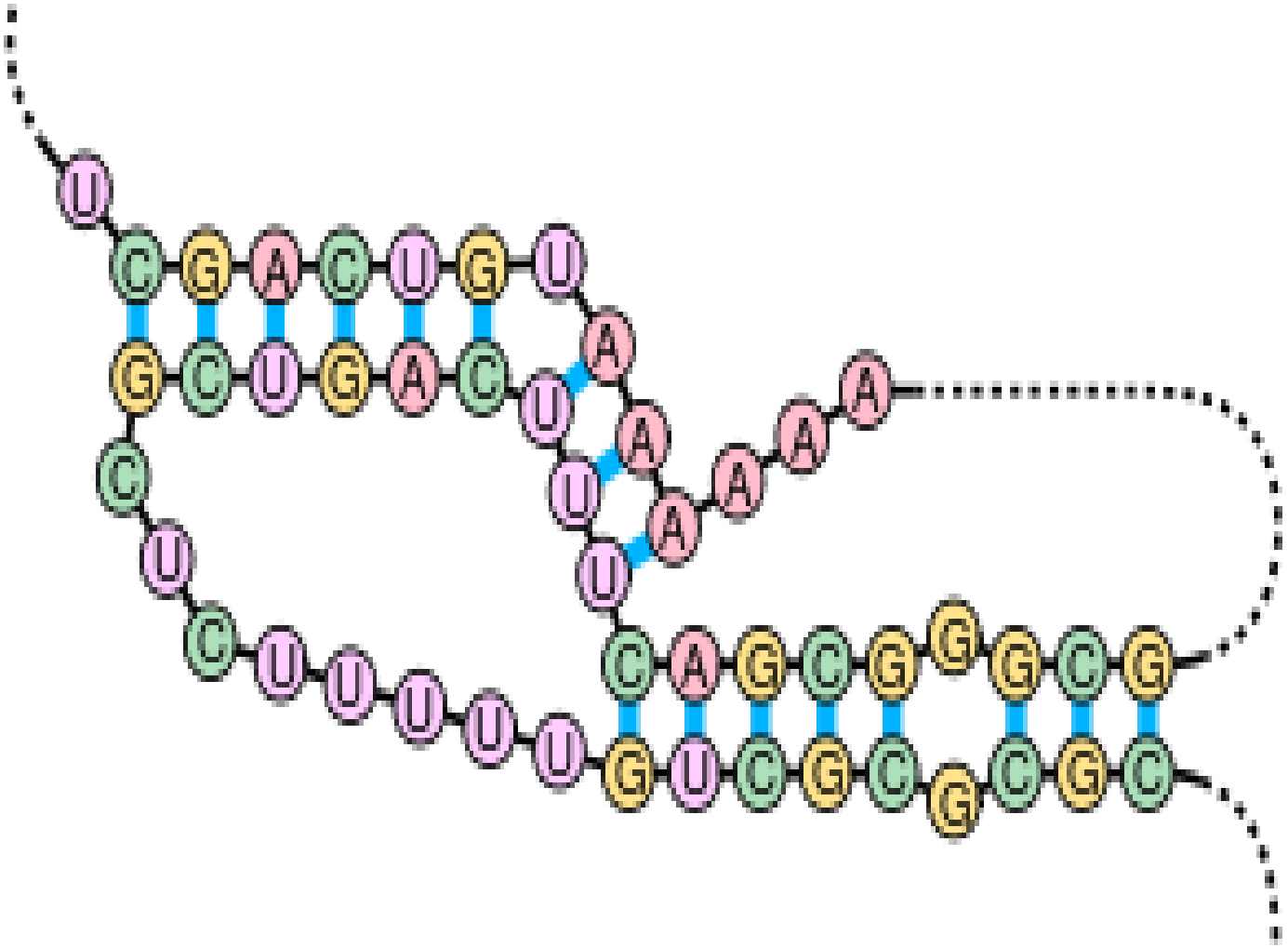
5' A C G U G C C A C G A U U C A A C G U G G C A C A G 3'



Pseudoknots

- A pseudoknot is a nucleic acid secondary structure containing at least two stem-loop structures in which half of one stem is intercalated between the two halves of another stem.

- Pseudoknots fold into knot-shaped three-dimensional conformations but are not true topological knots.



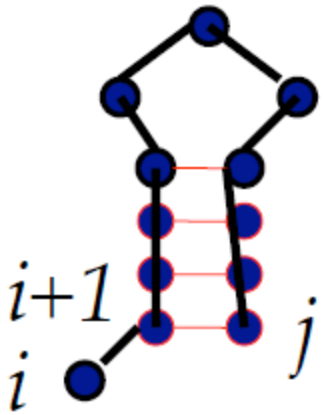
Secondary structure prediction

- Nucleic acid structure prediction is a computational method to determine nucleic acid secondary and tertiary structure from its sequence.
- Secondary structure can be predicted from a single or from several nucleic acid sequences.
- Tertiary structure can be predicted from the sequence, or by comparative modeling.
- The problem of predicting nucleic acid secondary structure is dependent mainly on base pairing and base stacking interactions; many molecules have several possible three-dimensional structures.

Nussinov's algorithm: idea

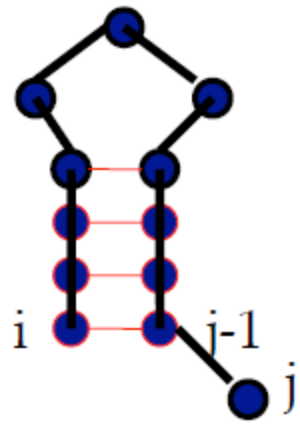
There are only four possible ways of getting the best structure for subsequence (i,j) from the best structures of the smaller subsequences

1) Add unpaired position i onto best structure for subsequence (i+1,j)

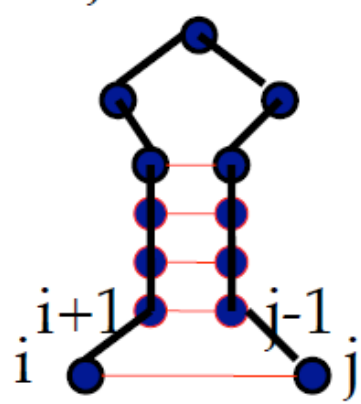


Nussinov's algorithm: idea

2) Add unpaired position j onto best structure for subsequence (i,j-1)

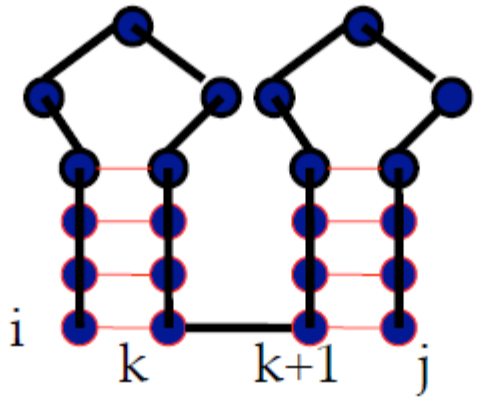


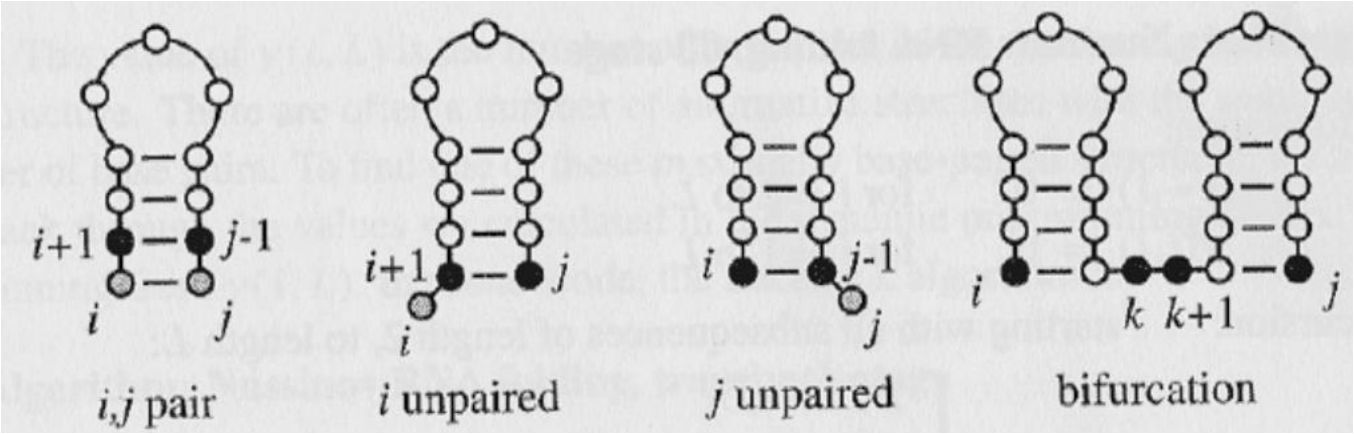
3) Add (i,j) pair onto best structure for subsequence (i+1,j-1)



Nussinov's algorithm: idea

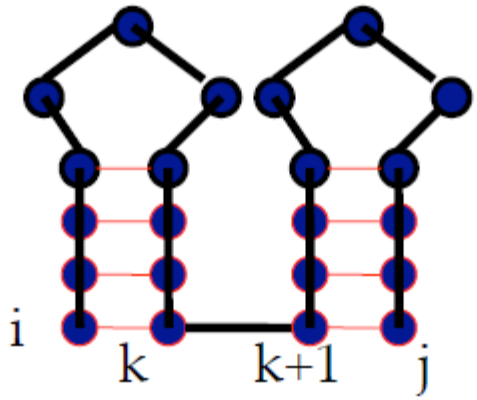
4) Combine two optimal substructures (i, k) and (k+1, j)

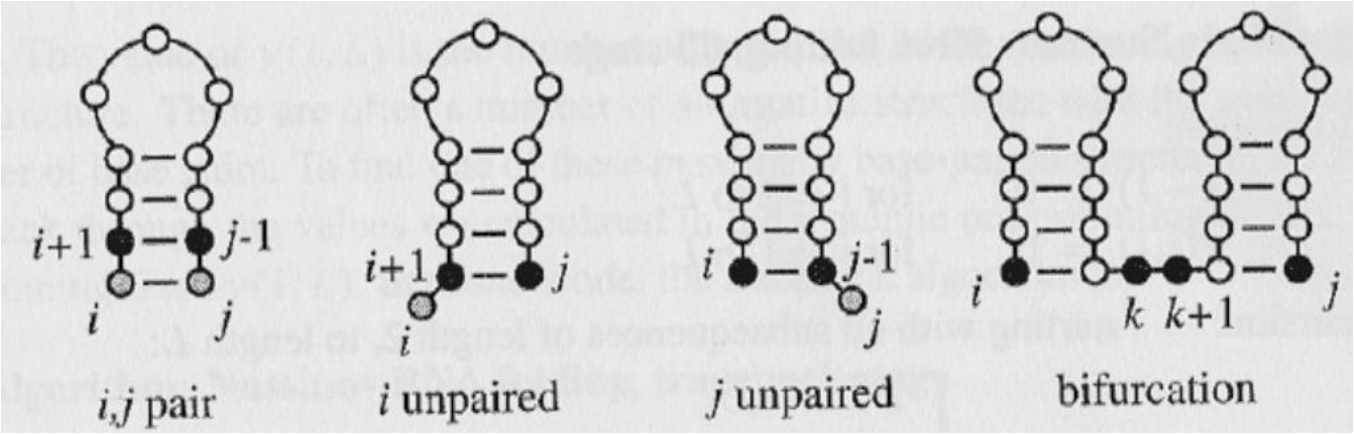




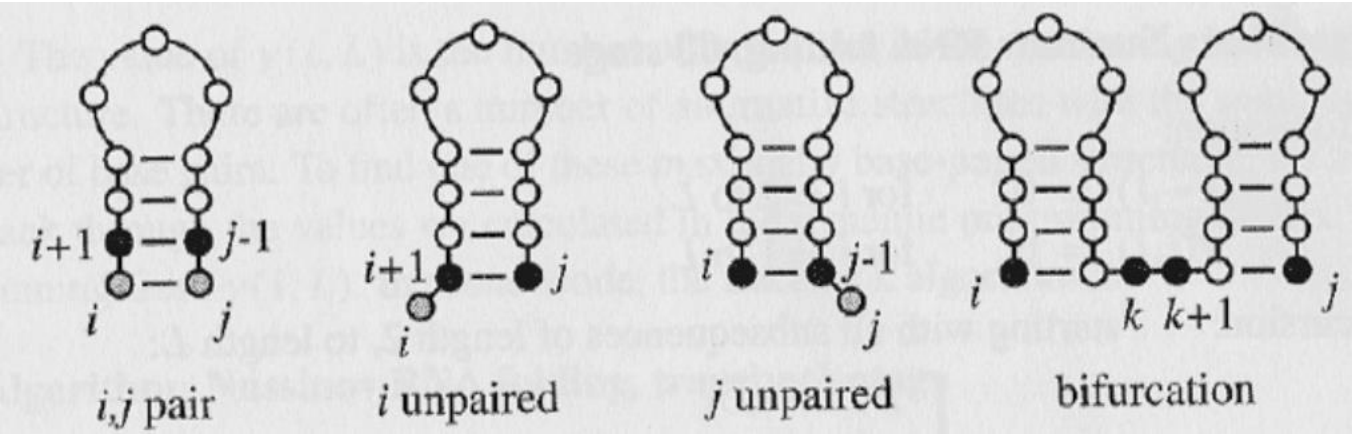
Nussinov's algorithm: idea

4) Combine two optimal substructures (i, k) and (k+1, j)





Nussinov's algorithm: idea



$$S(i,j) = max \begin{cases} S(i+1,j-1) + w(i,j) \\ S(i+1,j) \\ S(i,j-1) \\ max_{i<k<j} S(i,k) + S(k+1,j) \end{cases}$$

Initialization

- $S(i,i\text{-}1) = 0$
- $S(i,i) = 0$

➢Ruth Nussinov and co-workers who used the dynamic programming method for maximizing the number of base-pairs.

Nussinov folding algorithm (I)

- Let S[1..n] be the RNA sequence
- Let V(i,j) be the maximum number of base pairs in S[i..j].
- Base case:
  - V(i,i)=0 since the sequence has only one base!
  - V(i+1,i)= 0 since the sequence is empty!

Nussinov folding algorithm (II)

- When $i<j$, we have four cases:
  1. No base pair attached to j
     - $V(i, j) = V(i, j-1)$
  2. No base pair attached to i
     - $V(i,j) = V(i+1, j)$
  3. (i, j) form a base pair
     - $V(i, j) = V(i+1, j-1) + \delta(S[i], S[j])$
       where $\delta(x, y)=1$ if $(x,y)\in\{(a,u), (u,a), (c,g), (g,c), (g,u), (u,g)\}$; and 0, otherwise
  4. Both I and j attached to some base pairs both (i,j) is not a base pair
     - $V(i, j) = \max_{i\leq k<j}\{V(i,k)+V(k+1,j)\}$

Traceback

Input: Matrix S and positions i, j.
Output: Secondary structure maximizing the number of base pairs.
Initial call: traceback(i = 1, j = L).

```
if i < j then
   if S(i, j) = S(i + 1, j) then                          // case (1)
      traceback(i + 1, j)
   else if S(i, j) = S(i, j – 1) then                     // case (2)
      traceback(i, j – 1)
   else if S(i, j) = S(i + 1, j – 1) + w(i, j) then  // case (3)
      print base pair (i, j)
      traceback(i + 1, j – 1)
   else for k = i + 1 to j – 1 do                         // case (4)
      if S(i, j) = S(i, k) + S(k + 1, j) then
         traceback(i, k)
         traceback(k + 1, j)
         break
end
```

Example: base case

- S[1..7]=ACCAGCU

$$S(i,j) = max \begin{cases} S(i+1,j-1) + w(i,j) \\ S(i+1,j) \\ S(i,j-1) \\ max_{i<k<j} S(i,k) + S(k+1,j) \end{cases}$$

- Initialization
  - $S(i,i\text{-}1) = 0$
  - $S(i,i) = 0$

	1	2	3	4	5	6	7
1	0						
2	0	0					
3		0	0				
4			0	0			
5				0	0		
6					0	0	
7						0	0

Example: recursive case (I)

- S[1..7]=ACCAGCU

$$S(i,j) = max \begin{cases} S(i+1,j-1)+w(i,j) \\ S(i+1,j) \\ S(i,j-1) \\ max_{i<k<j} S(i,k)+S(k+1,j) \end{cases}$$

V (3,5)=max number of base pairs in S[3..5].

By the recursive formula, max

V(3,5)=max{V(4,4)+δ(S[3],S[5]), $\underset{3\le k<5}{V(3,k)}$+V(k+1,5)} =

max{V(4,4)+1, V(3,3)+V(4,5),

V(3,4)+V(5,5)} = 1

C	1	2	3	4	5	6	7
1	0	0	0				
2	0	0	0	0			
3		0	0	0			
4			0	0	0		
5				0	0	1	
6					0	0	0
7						0	0

Example: recursive case (II)

- S[1..7]=ACCAGCU

$$S(i,j) = max \begin{cases} S(i+1,j-1)+w(i,j) \\ S(i+1,j) \\ S(i,j-1) \\ max_{i<k<j} S(i,k)+S(k+1,j) \end{cases}$$

V (4,7)=max number of base pairs in S[4..6].

By the recursive formula, max

$V(4,7)=\max_{4\le k<7}\{V(5,6)+\delta(S[4],S[7]), V(4,k)+V(k+1,7)\} =$

$\max\{V(5,6)+1, V(4,4)+V(5,7), V(4,5)+V(6,7), V(4,6)+V(7,7)\} = 2$

C	1	2	3	4	5	6	7
1	0	0	0	0			
2	0	0	0	0	1		
3		0	0	0	1	1	
4			0	0	0	1	
5				0	0	1	1
6					0	0	0
7						0	0

Example: recursive case (III)

- S[1..7]=ACCAGCU

C	1	2	3	4	5	6	7
1	0	0	0	0	1	1	2
2	0	0	0	0	1	1	2
3		0	0	0	1	1	2
4			0	0	0	1	2
5				0	0	1	1
6					0	0	0
7						0	0

Nussinov folding algorithm (IV)

- Time analysis:
  - We need to fill-in $O(n^2)$ $V(i,j)$ entries
  - Each $V(i,j)$ entry can be computed in $O(n)$ time.
  - Thus, Nussinov algorithm can be solved in $O(n^3)$ time.

THANK YOU