RNA sequence-structure alignment

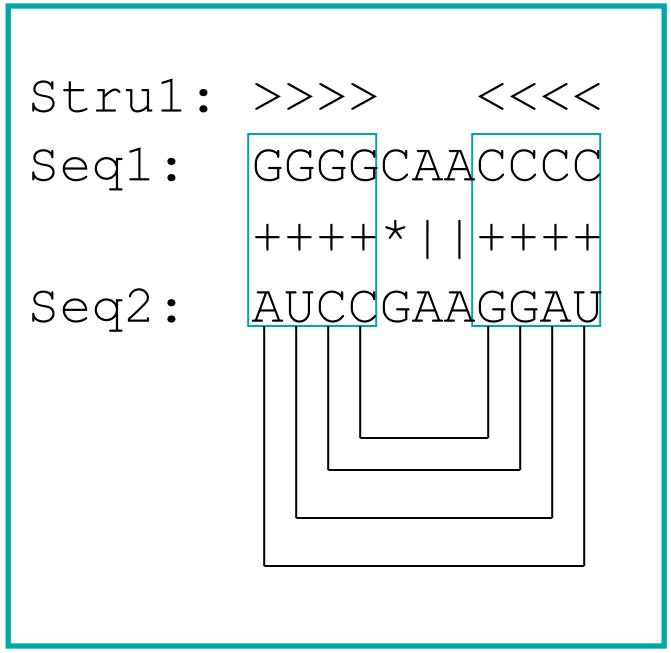
- RNA alignments:
  - sequence-sequence alignment.
  - sequence-structure alignment.
    - It's most useful for ncRNA finding.
  - structure-structure alignment.

What is an alignment?

- Alignment of two sequences

```
ACCCG-UUAAU
|  |||  ||*||
A-CCGUUUCAU
```

- Sequence-structure alignment:
  - One RNA sequence with its known secondary structure.
  - One just sequence
  - Take into account both structure similarity and sequence similarity.



RNA sequence-structure alignment

- Given two RNA sequence $s[1\ldots n]$ and $t[1\ldots m]$, and s has a known secondary structure S, where (i,j) in S implies $s[i]$ is basepaired with $s[j]$.
- Score for the alignment is the sum of scores γ of each column plus the sum of scores δ of each basepair in S.
- Scores for each column:
  - $\gamma(a,b)$ for all a, b *in* $\{A, C, G, U, -\}$.
- Scores for two basepair columns:
  - $\delta(a, b, c, d)$ for all $a, b, c,$ and d *in* $\{A, C, G, U\}$.
  - For example $\delta(a, b, c, d) = 1$ if (a,b) *in* S and c and d form basepair, otherwise $-\infty$.

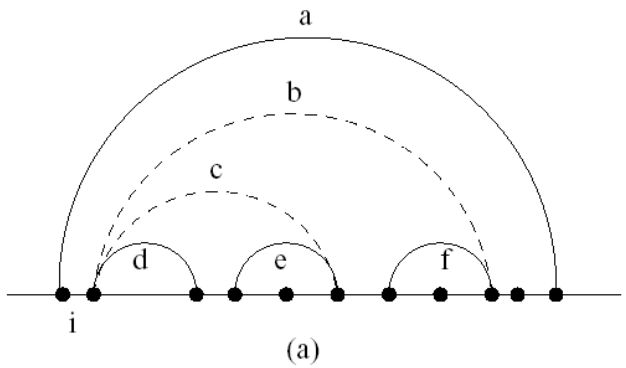
Problem 3: [**RNA sequence-structure alignment problem**] Given two RNA sequence $s[1..n]$ and $t[1...m]$, where s has a known secondary structure without pseudoknots, compute the highest scoring alignment.

Analysis: alignment solutions

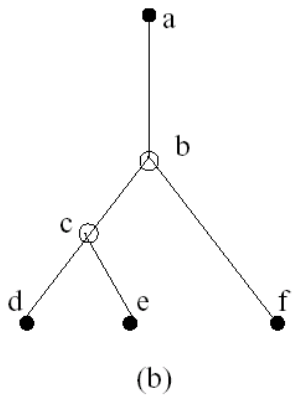
- Similar to Nussinov's base pair maximization problem's solution, the obvious dynamic programming solution is in $O(n^6)$ time.
- Bafna et al. (1995) improved the dynamic programming algorithm to a running time of $O(n^4)$ by using binarized tree for structure *S*.
- We further improve the algorithm by building a binarized tree for whole sequence $s[1 \ldots n]$ with its structure, and reduce the computation time into $O(n^3 m_1)$, where m_1 is the number of the branches in the tree (typically very small constant).

The method of Bafna et al. (1995)

- Adding spurious (dashed) edges to S, so that each node in S' has at most 2 children.
- We use the dashed edges (void nodes) to fix certain interval doing recursions, when we need find the branch points.
- It is in $O(n^2m^2+nm^3)$ time.



(a)



(b)

Binarizing an RNA structure tree (Bafan et al. 1995)

Procedure *Binarize*(i, j)
(* Assume that $(i, j) \in S$ has k children $\{(i_1, j_1), \ldots, (i_k, j_k)\}$ *)
begin
    for $1 \leq u < k$ do
        Binarize(i_u, j_u)
        $S' = S' \cup \{(i_1, j_u)\}$
        if $(u > 1)$
            parent$((i_1, j_{u-1})) = (i_1, j_u)$
            parent$((i_u, j_u)) = (i_1, j_u)$
    parent$(i_1, j_k) = (i, j)$
end

Alignment Algorithm in $O(m^2n^2+nm^3)$ (bafna et al. 1995)

Procedure *InferStructure*()
begin
for intervals (i_1, j_1), $1 \leq i_1 < j_1 \leq n$
and intervals (i_2, j_2), $1 \leq i_2 < j_2 \leq m$
(* Assume that the intervals are examined in lexicographically increasing order of widths*)

$$Align[i_1, j_1, i_2, j_2] = \max \begin{cases} Align[i_1+1, j_1, i_2, j_2] + \gamma(s[i_1], '-') \\ Align[i_1, j_1, i_2+1, j_2] + \gamma('-', t[i_2]) \\ Align[i_1+1, j_1, i_2+1, j_2] + \gamma(s[i_1], t[i_2]) \\ Align[i_1, j_1-1, i_2, j_2] + \gamma(s[j_1], '-') \\ Align[i_1, j_1, i_2, j_2-1] + \gamma('-', t[j_2]) \\ Align[i_1, j_1-1, i_2, j_2-1] + \gamma(s[j_1], t[j_2]) \end{cases}$$

if $(i_1, j_1) \in S$ and
$t[i_2]$ and $t[j_2]$ are complementary base-pairs

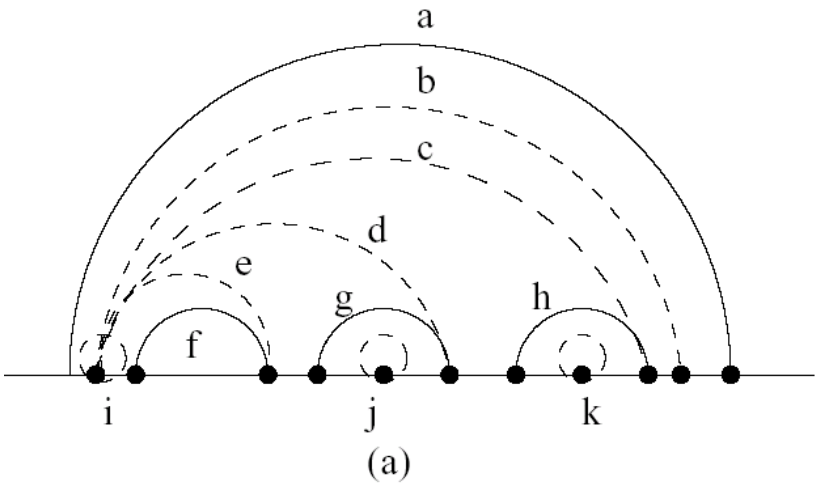
$$Align[i_1, j_1, i_2, j_2] = \max \begin{cases} Align[i_1, j_1, i_2, j_2], \\ \delta(i_1, j_1, i_2, j_2) + \gamma(s[i_1], t[i_2]) \\ +\gamma(s[j_1], t[j_2]) + Align[i_1+1, j_1-1, i_2+1, j_2-1] \end{cases}$$

else if $(i_1, j_1) \in S' - S$ and
$(k, j_1)) = rightchild(i_1, j_1)$

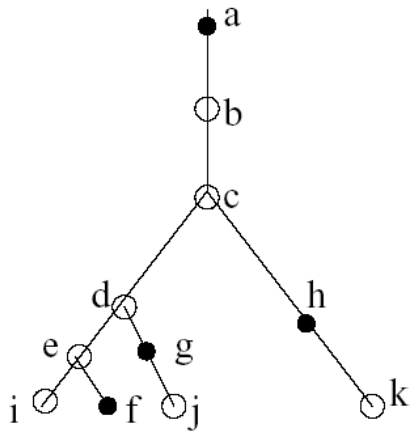
$$Align[i_1, j_1, i_2, j_2] = \max \begin{cases} Align[i_1, j_1, i_2, j_2], \\ \max_{i_2<l<j_2} \{Align[i_1, k-1, i_2, l-1] + Align[k, j_1, l, j_2]\} \end{cases}$$

Extend the binary tree into the whole sequence s[1…n]

- Solid nodes represent the basepairs.
- Dotted nodes represent either branch point or unpaired bases.
- Now the s[1…n] with its structure is changed into a binary tree.
- The dynamic programming compares each nodes against an interval.
- It is in $O(n^3 m_1)$ time. (m_1 the number of branch points)



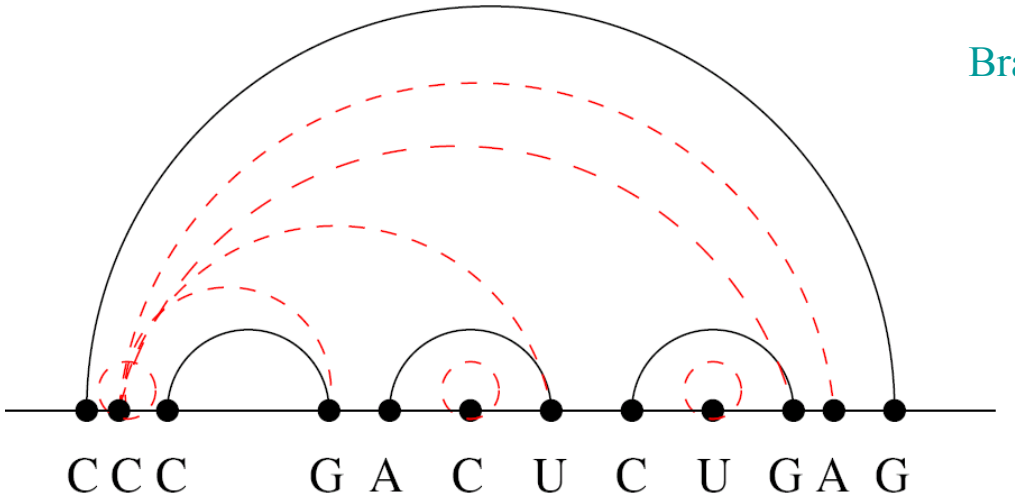
(a)

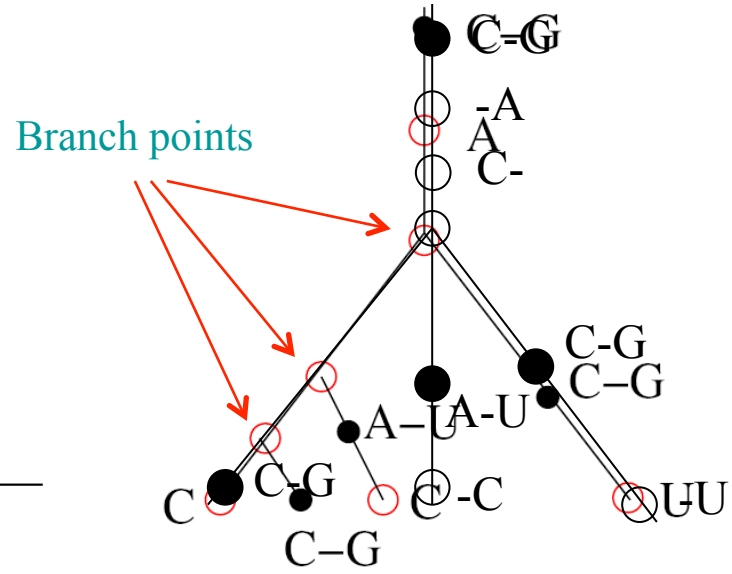


(b)

Using a binary tree to represent an RNA

- Solid nodes represent the basepairs.
- Void nodes represent either branch point or unpaired bases.





The dynamic programming aligns each subtree rooted at node *v* against an subinterval *(i,j)*

If root *v* is a solid node (first and last bases of the subsequence represented by the subtree are paired to each other):

If *v* is a void node (the subsequence's last base is unpaired):

If *v* is a void node (it is a branch point for multi-loop.):

procedure alignRNA

(* S is the set of base-pairs in RNA structure of s. S' is the augmented set. *)

for all intervals (i,j), $1 \le i < j \le n$, all nodes $v \in S'$

if $v \in S$

$$A[i,j,v] = \max \begin{cases} A[i+1, j-1, \text{child}(v)] + \gamma(s[l_v], t[i]) + \gamma(s[r_v], t[j]) \\ \quad +\delta(t[i], t[j], s[l_v], s[r_v]), \\ A[i, j-1, v] + \gamma('-', t[j]), \\ A[i+1, j, v] + \gamma('-', t[i]), \\ A[i+1, j, \text{child}[v]] + \gamma(s[l_v], t[i]) + \gamma(s[r_v], '-'), \\ A[i, j-1, \text{child}[v]] + \gamma(s[l_v], '-') + \gamma(s[r_v], t[j]), \\ A[i, j, \text{child}[v]] + \gamma(s[l_v], '-') + \gamma(s[r_v], '-'), \end{cases}$$

else if $v \in S' - S$, and v has one child

$$A[i,j,v] = \max \begin{cases} A[i, j-1, \text{child}[v]] + \gamma(s[r_v], t[j]), \\ A[i, j, \text{child}[v]] + \gamma(s[r_v], '-'), \\ A[i, j-1, v] + \gamma('-', t[j]), \\ A[i+1, j, v] + \gamma('-', t[i]), \end{cases}$$

else if $v \in S' - S$, and v has two children

$A[i,j,v] = \max_{i \le k \le j} \{A[i, k-1, \text{left\_child}[v]] + A[k, j, \text{right\_child}[v]]\}$

end if

Modifications of the formulation

- Banded alignment. [$O(n^2\delta_n)$]

- Complicated scoring functions:
  - Affine gap penalties in both stacks and loops.
  - Learning scoring matrix from handmade alignments.