CSE 417: Algorithms and Computational Complexity

Winter 2007
W. L. Ruzzo
Dynamic Programming, II
RNA Folding
The Double Helix

As shown, the two strands coil about each other in a fashion such that all the bases project inward toward the helix axis. The two strands are held together by hydrogen bonds (pink rods) linking each base projecting from one backbone to its so-called complementary base projecting from the other backbone. The base A always bonds to T (A and T are comple-

Shown in (b) is an uncoiled fragment of (a) three complementary base pair. According to the chemist's viewpoint, each strand is a polymer made up of four re-

called deoxyribonucleotides.

http://www.rcsb.org/pdb/explore.do?structureId=1GAT
The “Central Dogma” of Molecular Biology

DNA → RNA → Protein
Non-coding RNA

• Messenger RNA - codes for proteins
• Non-coding RNA - all the rest
  – Before, say, mid 1990’s, 1-2 dozen known
    (critically important, but narrow roles: e.g. ribosomal and transfer RNA, splicing, SRP)
• Since mid 90’s dramatic discoveries
  • Regulation, transport, stability/degradation
  • E.g. “microRNA”: hundreds in humans
  • E.g. “riboswitches”: thousands in bacteria
DNA structure: dull

...ACCGCTAGATG...

...TGGCGATCTAC...
RNA
Structure: Rich

• RNA’s fold, and function
• Nature uses what works
RNA
Secondary Structure:

Not everything, but important, easier than 3d
Why is structure important?

• For protein-coding, similarity in sequence is a powerful tool for finding related sequences
  – e.g. “hemoglobin” is easily recognized in all vertebrates

• For non-coding RNA, many different sequences have the same structure, and structure is most important for function.
  – So, using structure plus sequence, can find related sequences at much greater evolutionary distances
Q: What's so hard?

A: Structure often more important than sequence
6S mimics an open promoter

Barrick et al. RNA 2005
Trotochaud et al. NSMB 2005
Willkomm et al. NAR 2005
Chloroflexus aurantiacus
Geobacter metallireducens
Geobacter sulphurreducens
Symbiobacterium thermophilum

δ-Proteobacteria

γ-Proteobacteria

β-Proteobacteria

α-Proteobacteria

ε-Proteobacteria

Spirochaetes
Chlamydiae

Actinobacteria (high GC)

Cyanobacteria

Firmicutes (low GC)
“Central Dogma”

= “Central Chicken & Egg”?

Was there once an “RNA World”?

DNA → RNA → Protein

DNA (chromosome)

RNA (messenger)

cell

Protein
6.5 RNA Secondary Structure

Nussinov’s Algorithm
RNA Secondary Structure

RNA. String $B = b_1b_2\ldots b_n$ over alphabet \{A, C, G, U\}.

Secondary structure. RNA is usually single-stranded, and tends to loop back and form base pairs with itself. This structure is essential for understanding behavior of molecule.

Ex: GUCGAUUGAGCGAAUGUAACAACGUGGCCUACGCGGAGA

complementary base pairs: A-U, C-G
RNA Secondary Structure (somewhat oversimplified)

Secondary structure. A set of pairs $S = \{ (b_i, b_j) \}$ that satisfy:

- [Watson-Crick.] $S$ is a matching, i.e. each base pairs with at most one other, and each pair in $S$ is a Watson-Crick pair: A-U, U-A, C-G, or G-C.

- [No sharp turns.] The ends of each pair are separated by at least 4 intervening bases. If $(b_i, b_j) \in S$, then $i < j - 4$.

- [Non-crossing.] If $(b_i, b_j)$ and $(b_k, b_l)$ are two pairs in $S$, then we cannot have $i < k < j < l$. (Violation of this is called a pseudoknot.)

Free energy. Usual hypothesis is that an RNA molecule will form the secondary structure with the optimum total free energy.

Goal. Given an RNA molecule $B = b_1b_2...b_n$, find a secondary structure $S$ that maximizes the number of base pairs.
RNA Secondary Structure: Examples

Examples.

- **base pair**
- **sharp turn**
- **ok**
- **crossing**
RNA Secondary Structure: Subproblems

First attempt. \( OPT[j] = \) maximum number of base pairs in a secondary structure of the substring \( b_1b_2...b_j \).

Difficulty. Results in two sub-problems.

- Finding secondary structure in: \( b_1b_2...b_{t-1} \).
- Finding secondary structure in: \( b_{t+1}b_{t+2}...b_{j-1} \).

\( OPT(t-1) \)  
\( \text{not OPT of anything; need more sub-problems} \)
Dynamic Programming Over Intervals: (R. Nussinov’s algorithm)

Notation. \( \text{OPT}[i, j] = \) maximum number of base pairs in a secondary structure of the substring \( b_i b_{i+1} \ldots b_j \).

- **Case 1.** If \( i \geq j - 4 \).
  - \( \text{OPT}[i, j] = 0 \) by no-sharp turns condition.

- **Case 2.** Base \( b_j \) is not involved in a pair.
  - \( \text{OPT}[i, j] = \text{OPT}[i, j-1] \)

- **Case 3.** Base \( b_j \) pairs with \( b_t \) for some \( i \leq t < j - 4 \).
  - non-crossing constraint decouples resulting sub-problems
  - \( \text{OPT}[i, j] = 1 + \max_t \{ \text{OPT}[i, t-1] + \text{OPT}[t+1, j-1] \} \)
    
    take max over \( t \) such that \( i \leq t < j-4 \) and \( b_t \) and \( b_j \) are Watson-Crick complements

**Key point:** Either last base is unpaired (case 1,2) or paired (case 3)

**Remark.** Same core idea in CKY algorithm to parse context-free grammars.
Bottom Up Dynamic Programming Over Intervals

Q. What order to solve the sub-problems?
A. Do shortest intervals first.

```
RNA(b_1, ..., b_n) {
    for k = 5, 6, ..., n-1
        for i = 1, 2, ..., n-k
            j = i + k
            Compute OPT[i, j]
        return OPT[1, n] using recurrence
}
```

Running time. \( O(n^3) \).
E.g.:
OPT[6,16] = 2:
GUUGCAAUGUC
((....)....)

E.g.:
OPT[1,6] = 1:
CUCCGG
((.....))

CUCCGGGUUGUC
((.....))
Computing one cell: $\text{OPT}[2,18] = ?$

$$\text{OPT}(i,j) = \begin{cases} 
0 & \text{if } i \geq j - 4 \\
\max \left\{ \text{OPT}[i,j-1], 1 + \max_t (\text{OPT}[i,t-1] + \text{OPT}[t+1,j-1]) \right\} & \text{otherwise}
\end{cases}$$

Case 1:

$2 \geq 18 - 4$? no.

Case 2:

$B_{18}$ unpaired?

Always a possibility; then $\text{OPT}[2,18] \geq 3$
Computing one cell: \( \text{OPT}[2,18] = ? \)

\[
\text{OPT}(i, j) = \begin{cases} 
0 & \text{if } i \geq j - 4 \\
\max \left\{ \text{OPT}[i, j - 1] \right\} & \text{otherwise}
\end{cases}
\]

Case 3, \( 2 \leq t < 18 - 4 \): 
\( t = 2 \): no pair
Computing one cell: OPT[2, 18] = ?

\[ OPT(i, j) = \begin{cases} 
0 & \text{if } i \geq j - 4 \\
\max \left\{ \begin{array}{l}
\text{OPT}[i, j - 1] \\
1 + \max_t (\text{OPT}[i, t - 1] + \text{OPT}[t + 1, j - 1])
\end{array} \right. & \text{otherwise}
\end{cases} \]

Case 3, \(2 \leq t < 18-4:\)
\(t = 3: \) no pair
Computing one cell: OPT[2,18] = ?

Case 3, $2 \leq t < 18-4$:
- $t = 4$: yes pair
- $\text{OPT}[2,18] \geq 1 + 0 + 3$

GGAAAACCCAAAGGGGU
\cdots((\cdots))

OPT$(i,j) = \begin{cases} 
0 & \text{if } i \geq j - 4 \\
\max \left\{ \begin{array}{l}
\text{OPT}[i, j - 1] \\
1 + \max_t \left( \text{OPT}[i,t - 1] + \text{OPT}[t + 1, j - 1] \right)
\end{array} \right. & \text{otherwise}
\end{cases}$
Computing one cell: $\text{OPT}[2,18] = ?$

Case 3, $2 \leq t < 18 - 4$:
- $t = 5$: yes pair
- $\text{OPT}[2,18] \geq 1 + 0 + 3$

$$\text{OPT}(i, j) = \begin{cases} 0 & \text{if } i \leq j - 4 \\ \max \left\{ \text{OPT}[i, j - 1], 1 + \max_t (\text{OPT}[i, t - 1] + \text{OPT}[t + 1, j - 1]) \right\} & \text{otherwise} \end{cases}$$
Computing one cell: $\text{OPT}[2,18] = ?$

\[
\text{OPT}(i, j) = \begin{cases} 
0 & \text{if } i \geq j - 4 \\
\max \left\{ \text{OPT}[i, j - 1], 1 + \max_t (\text{OPT}[i, t - 1] + \text{OPT}[t + 1, j - 1]) \right\} & \text{otherwise}
\end{cases}
\]

Case 3, $2 \leq t < 18 - 4$: 
\( t = 6: \) yes pair

\( \text{OPT}[2, 18] \geq 1 + 0 + 3 \)

GGAAACCCAAAGGGGU

\( \ldots \ldots (\ldots (\ldots (\ldots )))) \)
Computing one cell: $\text{OPT}[2,18] = \, ?$

\[
\text{OPT}(i, j) = \begin{cases} 
0 & \text{if } i \geq j - 4 \\
\max \left\{ \text{OPT}[i, j - 1], 1 + \max_t (\text{OPT}[i, t - 1] + \text{OPT}[t + 1, j - 1]) \right\} & \text{otherwise}
\end{cases}
\]

Case 3, $2 \leq t < 18 - 4$:
- $t = 7$: yes pair
- $\text{OPT}[2,18] \geq 1 + 0 + 3$

GGAAACCCAAAGGGGU

\[\cdots \cdot \cdot \cdot (((((\cdots))) )\right)\]
Computing one cell: $\text{OPT}[2,18] = ?$

$$
\begin{align*}
\text{OPT}(i,j) &= \begin{cases} 
0 & \text{if } i \geq j - 4 \\
\max \left\{ \text{OPT}[i,j - 1] \right\} & \text{if } i < j - 4 \\
1 + \max_t (\text{OPT}[i,t - 1] + \text{OPT}[t + 1, j - 1]) & \text{if } i < j - 4 
\end{cases}
\end{align*}
$$

Case 3, $2 \leq t < 18 - 4$:
$t = 8$: no pair
Computing one cell: $\text{OPT}[2, 18] = \text{?}$. 

$\text{OPT}(i, j) = \begin{cases} 
0 & \text{if } i \geq j - 4 \\
\max \left\{ \text{OPT}[i, j - 1], 1 + \max_t (\text{OPT}[i, t - 1] + \text{OPT}[t + 1, j - 1]) \right\} & \text{otherwise}
\end{cases}$
Computing one cell: \( \text{OPT}[2,18] = 4 \)

Overall, Max = 4 several ways, e.g.: 

\text{GGAAAAACCCAAAGGGGU} \dots (\ldots(((\ldots))))

Tree shows trace back: 
square = case 3
octagon = case 1

\[
\text{OPT}(i, j) = \begin{cases} 
0 & \text{if } i \geq j - 4 \\
\max \left\{ \text{OPT}[i, j - 1], 1 + \max_t (\text{OPT}[i, t - 1] + \text{OPT}[t + 1, j - 1]) \right\} & \text{otherwise}
\end{cases}
\]
Another Trace Back Example

E.g.: OPT[1,16] = 3:
CUCCGGUUGCAAUGUC
((.(.....).)...)...

OPT(i, j) =
0 if i ≥ j - 4
\[
\max\left\{ \begin{array}{l}
0 \\
\text{OPT}[i, j-1] \\
1 + \max_t (\text{OPT}[i, t-1] + \text{OPT}[t+1, j-1])
\end{array} \right. \\
\] otherwise