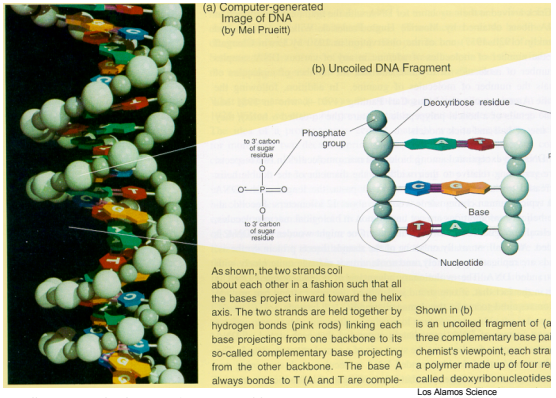
CSE 421: Intro to Algorithms

Summer 2007

W. L. Ruzzo

Dynamic Programming, II
RNA Folding

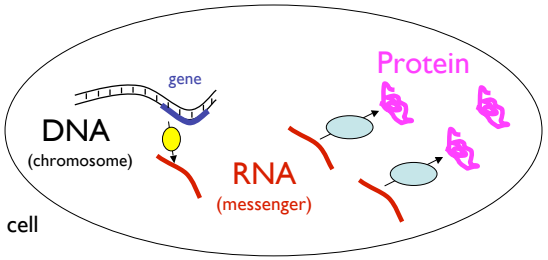
The Double Helix



http://www.rcsb.org/pdb/explore.do?structureId=1GAT

The "Central Dogma" of Molecular Biology

DNA → RNA → Protein

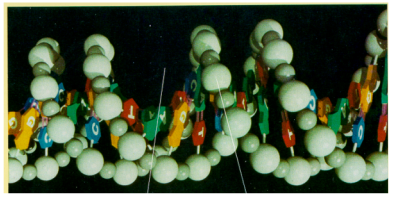


Non-coding RNA

- 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

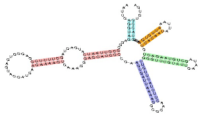
DNA structure: dull

…ACCGCTAGATG…
| | | | | | | | | | |
…TGGCGATCTAC…

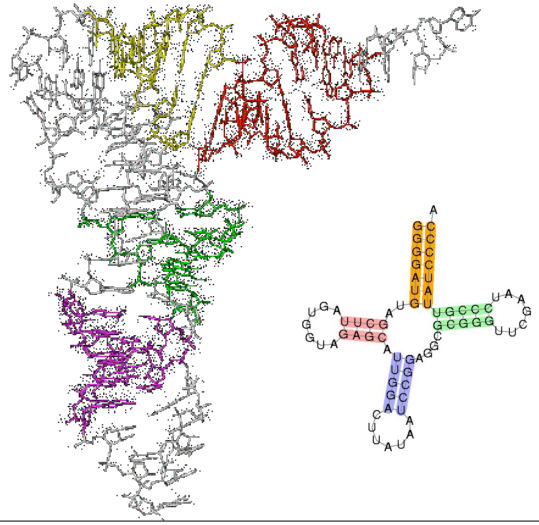


RNA Structure: Rich

- RNA's fold, and function
- Nature uses what works

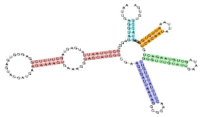


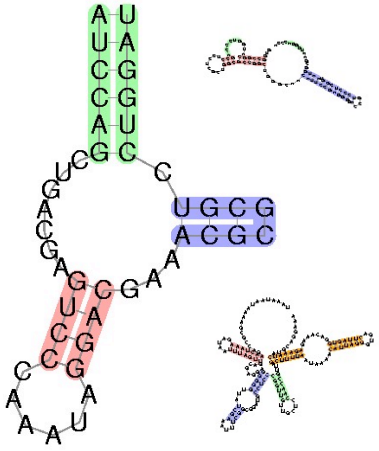
http://www.rcsb.org/pdb/explore.do?structureId=1EVV



RNA Secondary Structure:

Not everything, but important, easier than 3d

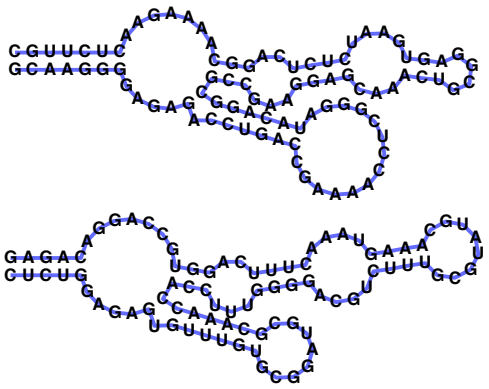




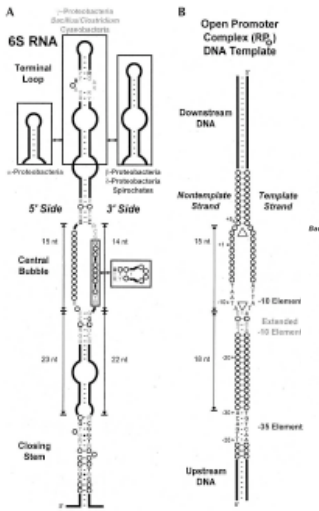
Why is structure important?

- 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

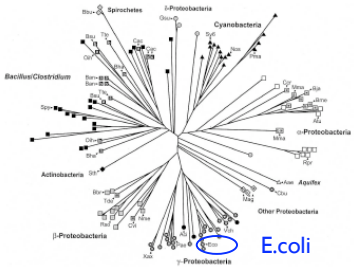
Q: What's so hard?



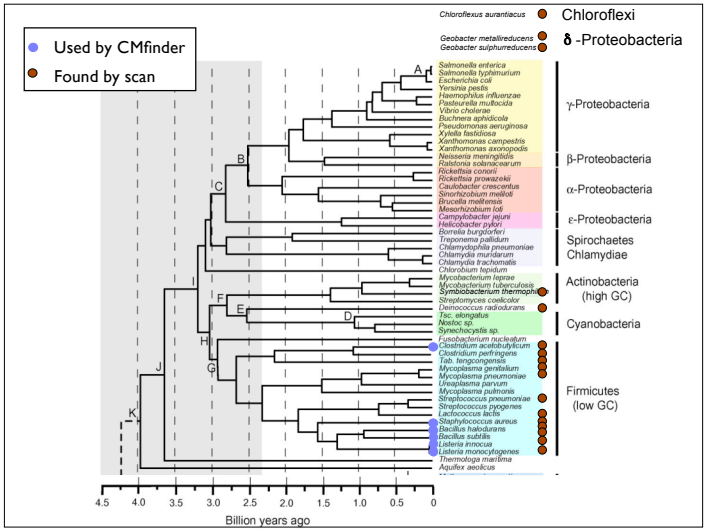
A: Structure often more important than sequence

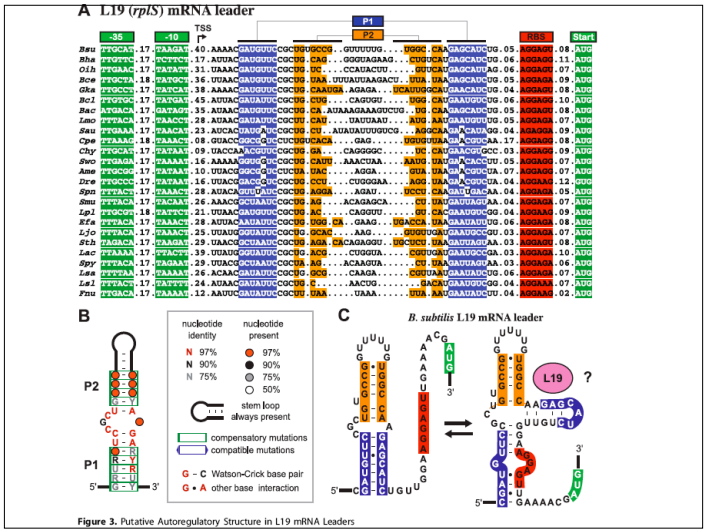


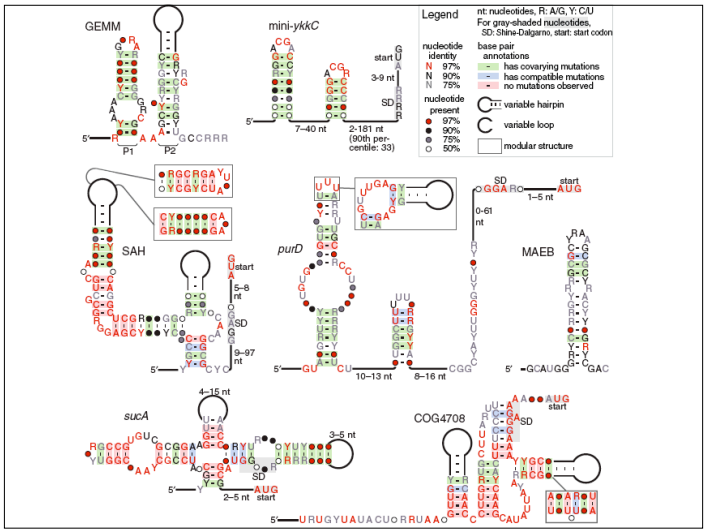
6S mimics an open promoter



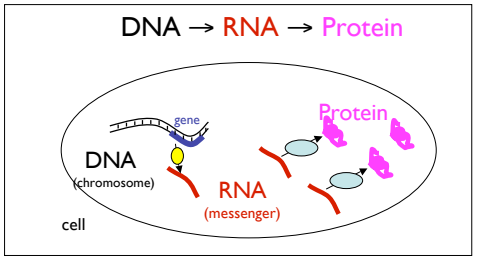
Barrick et al. *RNA* 2005
Trotochaud et al. *NSMB* 2005
Willkomm et al. NAR 2005







"Central Dogma" = "Central Chicken & Egg"?



Was there once an "RNA World"?

6.5 RNA Secondary Structure

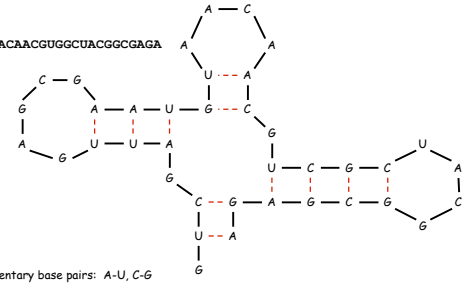
Nussinov's Algorithm

RNA Secondary Structure

RNA. String $B = b_1 b_2 \ldots b_n$ 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: GUCGAUUGAGCGAAUGUAACAACGUGGCUACGGCGAGA



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*.)

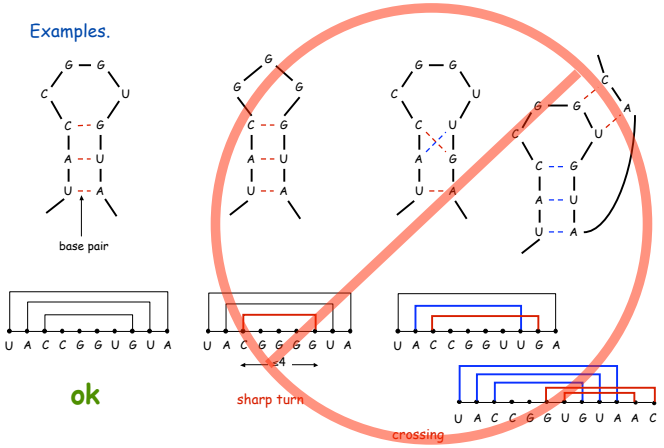
Free energy. Usual hypothesis is that an RNA molecule will form the secondary structure with the optimum total free energy.

approximate by number of base pairs

Goal. Given an RNA molecule $B = b_1 b_2 \ldots b_n$, find a secondary structure S that maximizes the number of base pairs.

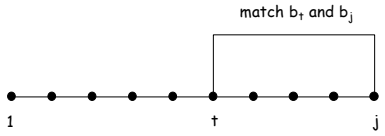
RNA Secondary Structure: Examples

Examples.



RNA Secondary Structure: Subproblems

First attempt. OPT[j] = maximum number of base pairs in a secondary structure of the substring $b_1 b_2 \ldots b_j$.



Difficulty. Results in two sub-problems.

- Finding secondary structure in: $b_1 b_2 \ldots b_{t-1}$. ← OPT(t-1)
- Finding secondary structure in: $b_{t+1} b_{t+2} \ldots b_{j-1}$. ← not OPT of anything: need more sub-problems

Dynamic Programming Over Intervals: (R. Nussinov's algorithm)

Notation. 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 $i \geq j - 4$.
  - OPT[i, j] = 0 by no-sharp turns condition.
- Case 2. Base b_j is not involved in a pair.
  - OPT[i, j] = 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
  - $OPT[i, j] = 1 + \max_t \{ OPT[i, t-1] + OPT[t+1, j-1] \}$

take max over t such that $i \leq t < j-4$ and b_t and b_j are Watson-Crick complements

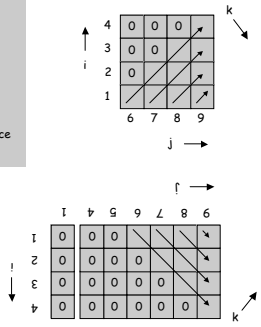
Key point: Either last base is unpaired (case 1,2) or paired (case 3)

Remark. Same core idea in CKY algorithm to parse context-free grammars.

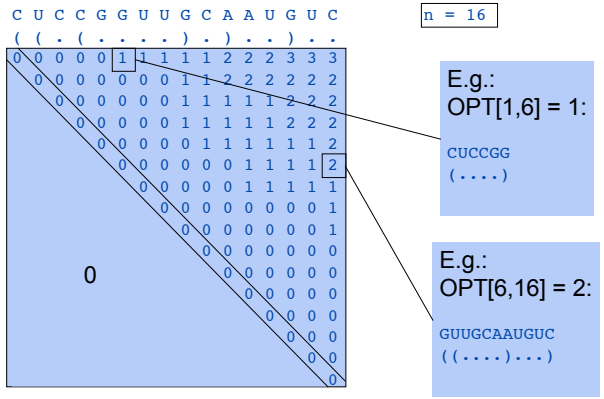
Bottom Up Dynamic Programming Over Intervals

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

```
RNA(b1,…,bn) {
   for k = 5, 6, …, n-1
      for i = 1, 2, …, n-k
         j = i + k
         Compute OPT[i, j]
                          ↖ using recurrence
   return OPT[1, n]
}
```



Running time. $O(n^3)$.



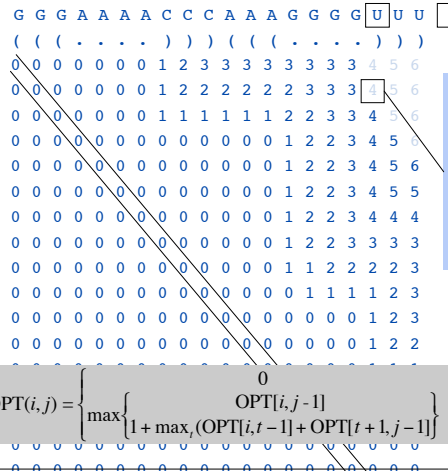
E.g.:
OPT[1,6] = 1:

CUCCGG
(....)

E.g.:
OPT[6,16] = 2:

GUUGCAAUGUC
((....)...)

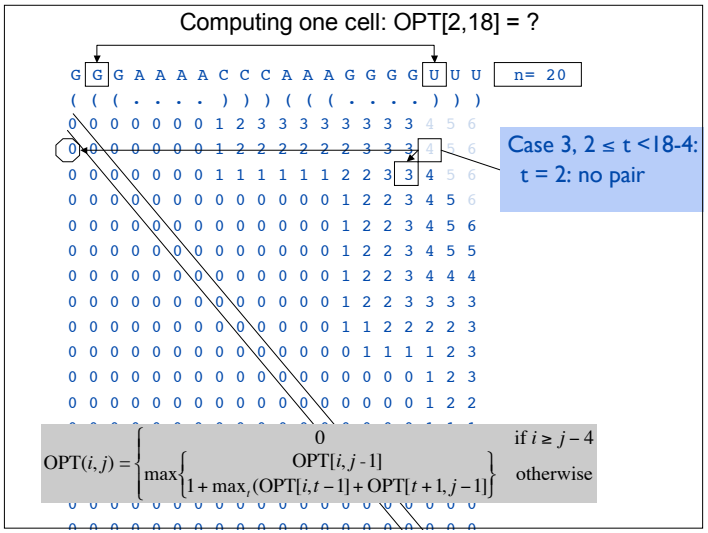
Computing one cell: OPT[2,18] = ?

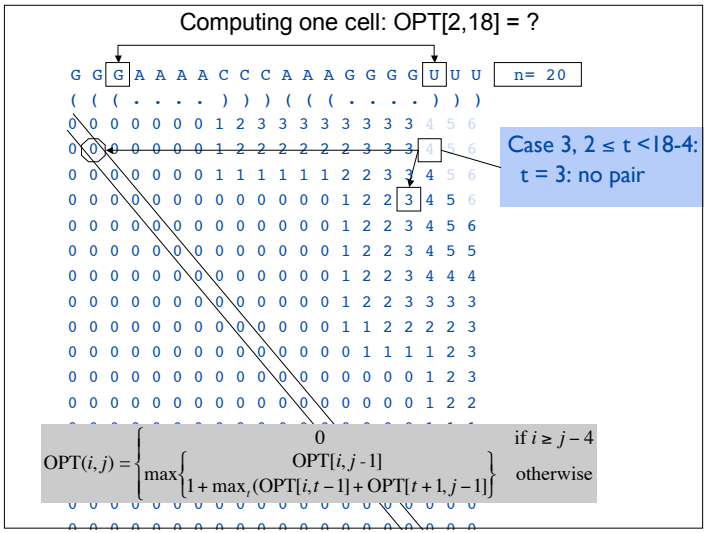


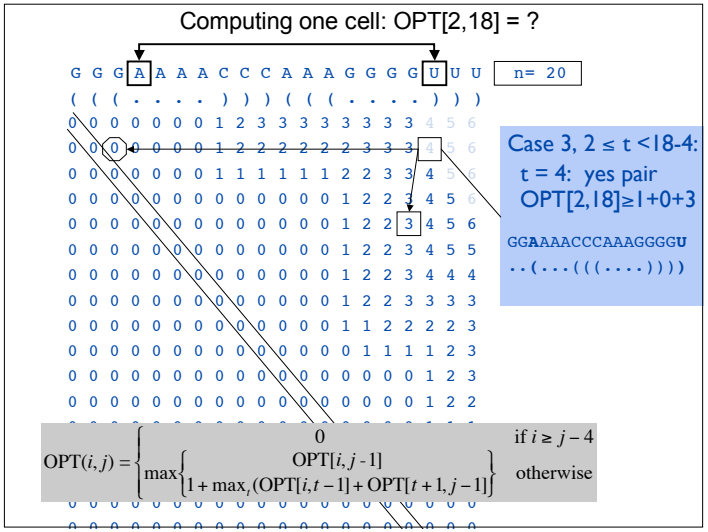
Case 1:
2 ≥ 18-4? no.
Case 2:
B_{18} unpaired?
Always a possibility;
then OPT[2,18] ≥ 3

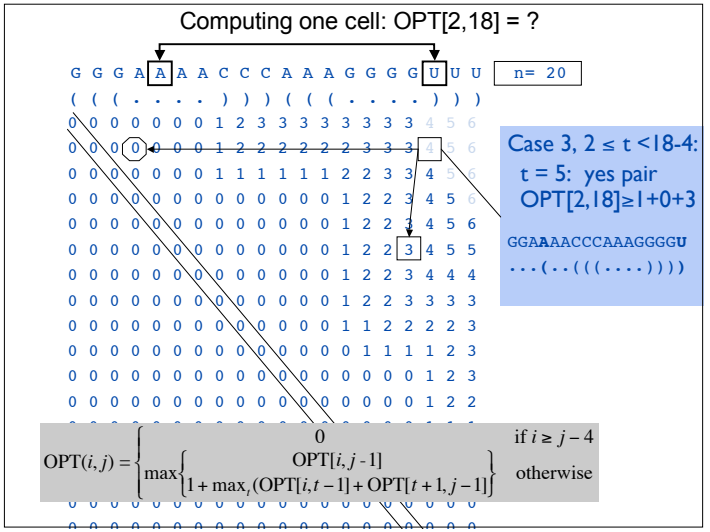
GGAAAACCCAAAGGGGU
((....))(....)...

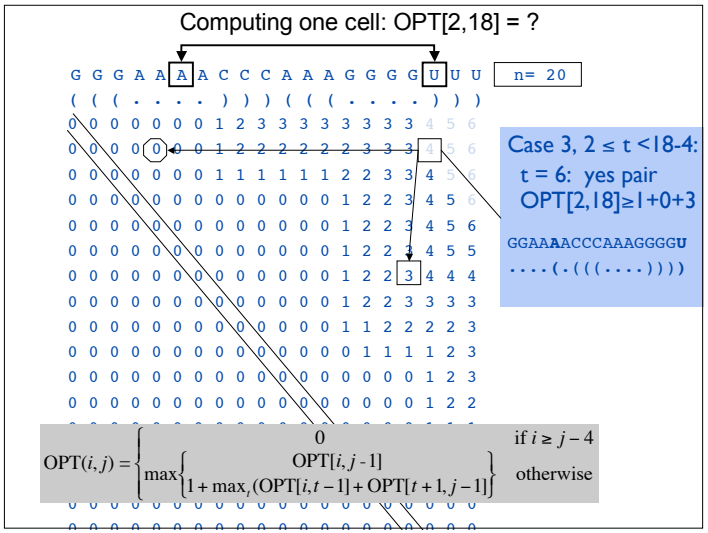
$$OPT(i,j) = \begin{cases} 0 & \text{if } i \geq j-4 \\ \max\left\{ \begin{array}{c} OPT[i,j-1] \\ 1 + \max_t (OPT[i,t-1] + OPT[t+1,j-1]) \end{array} \right\} & \text{otherwise} \end{cases}$$

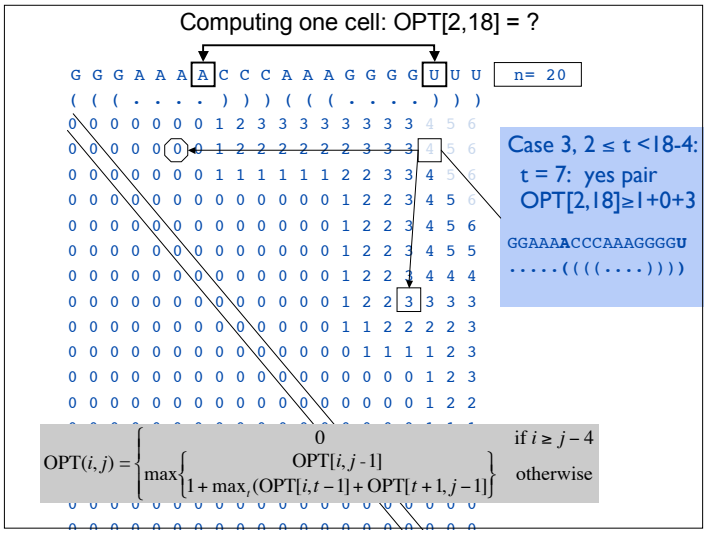

Computing one cell: OPT[2,18] = ?
G G G A A A A C C C A A A G G G G U U U
n= 20
(((. . . .))) (((. . . .)))
Case 3, 2 ≤ t <18-4:
t = 2: no pair
$OPT(i,j) = \begin{cases} 0 & \text{if } i \geq j-4 \\ \max\left\{ \begin{matrix} OPT[i,j-1] \\ 1+\max_t(OPT[i,t-1]+OPT[t+1,j-1]) \end{matrix} \right\} & \text{otherwise} \end{cases}$

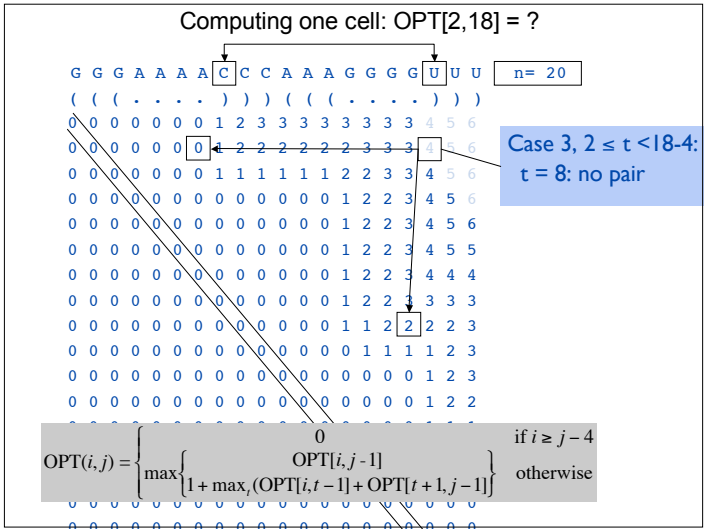

Computing one cell: OPT[2,18] = ?
G G G A A A A C C C A A A G G G G U U U
n= 20
(((. . . .))) (((. . . .)))
Case 3, 2 ≤ t <18-4:
t = 3: no pair
$OPT(i,j) = \begin{cases} 0 & \text{if } i \geq j-4 \\ \max\left\{ \begin{matrix} OPT[i,j-1] \\ 1+\max_t(OPT[i,t-1]+OPT[t+1,j-1]) \end{matrix} \right\} & \text{otherwise} \end{cases}$

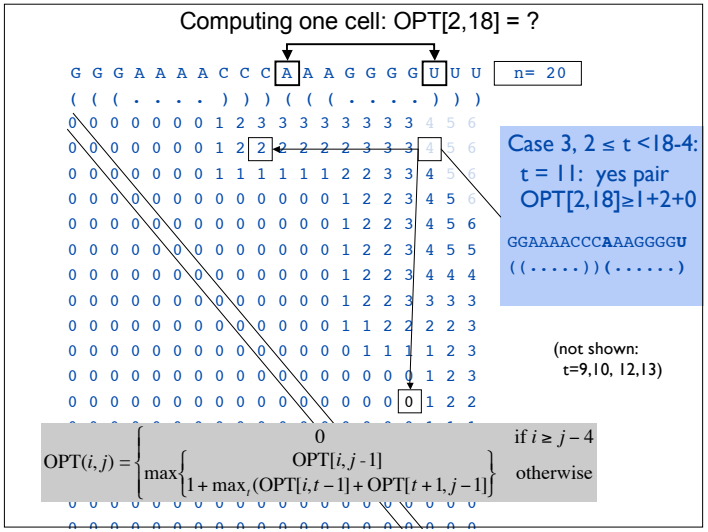

Computing one cell: OPT[2,18] = ?
G G G A A A A C C C A A A G G G G U U U
n= 20
(((. . . .))) (((. . . .)))
Case 3, 2 ≤ t <18-4:
t = 4: yes pair
OPT[2,18]≥1+0+3
GGAAAACCCAAAGGGGU
..(...(((....)))).
$OPT(i,j) = \begin{cases} 0 & \text{if } i \geq j-4 \\ \max\left\{ \begin{matrix} OPT[i,j-1] \\ 1+\max_t(OPT[i,t-1]+OPT[t+1,j-1]) \end{matrix} \right\} & \text{otherwise} \end{cases}$

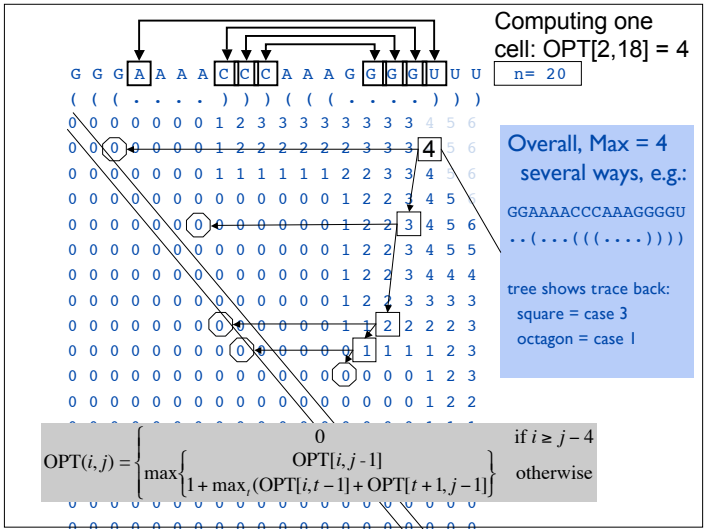

Computing one cell: OPT[2,18] = ?
G G G A A A A C C C A A A G G G G U U U
n= 20
(((. . . .))) (((. . . .)))
Case 3, 2 ≤ t <18-4:
t = 5: yes pair
OPT[2,18]≥1+0+3
GGAAAACCCAAAGGGGU
...(..(((....)))).
$OPT(i,j) = \begin{cases} 0 & \text{if } i \geq j-4 \\ \max\left\{ \begin{matrix} OPT[i,j-1] \\ 1+\max_t(OPT[i,t-1]+OPT[t+1,j-1]) \end{matrix} \right\} & \text{otherwise} \end{cases}$

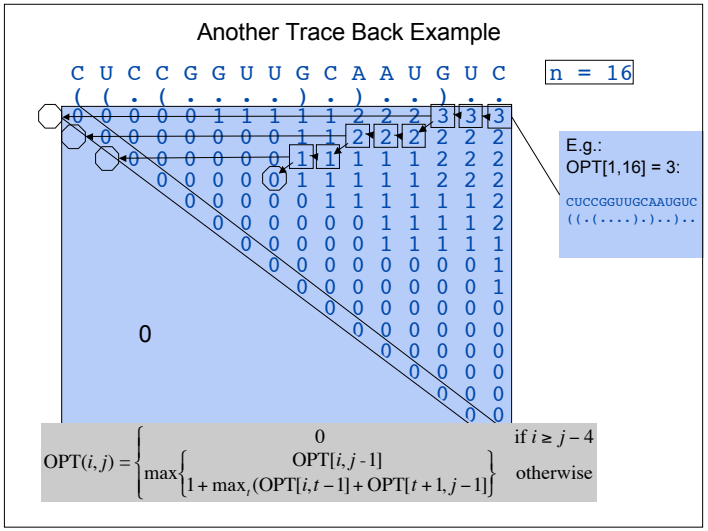

Computing one cell: OPT[2,18] = ?
G G G A A A C C C A A A G G G G U U U
n= 20
(((. . . .))) (((. . . .)))
Case 3, 2 ≤ t <18-4:
t = 6: yes pair
OPT[2,18]≥1+0+3
GGAAAACCCAAAGGGGU
....(.(((....))))
$OPT(i,j) = \begin{cases} 0 & \text{if } i \geq j-4 \\ \max\left\{ \begin{matrix} OPT[i,j-1] \\ 1+\max_t(OPT[i,t-1]+OPT[t+1,j-1]) \end{matrix} \right\} & \text{otherwise} \end{cases}$


Computing one cell: OPT[2,18] = ?
G G G A A A C C C A A A G G G G U U U
n= 20
(((. . . .))) (((. . . .)))
Case 3, 2 ≤ t <18-4:
t = 7: yes pair
OPT[2,18]≥1+0+3
GGAAAACCCAAAGGGGU
.....((((....))))
$OPT(i,j) = \begin{cases} 0 & \text{if } i \geq j-4 \\ \max\left\{ \begin{matrix} OPT[i,j-1] \\ 1+\max_t(OPT[i,t-1]+OPT[t+1,j-1]) \end{matrix} \right\} & \text{otherwise} \end{cases}$


Computing one cell: OPT[2,18] = ?
G G G A A A A C C C A A A G G G G U U U
n= 20
(((. . . .))) (((. . . .)))
Case 3, 2 ≤ t <18-4:
t = 8: no pair
$OPT(i,j) = \begin{cases} 0 & \text{if } i \geq j-4 \\ \max\left\{ \begin{matrix} OPT[i,j-1] \\ 1+\max_t(OPT[i,t-1]+OPT[t+1,j-1]) \end{matrix} \right\} & \text{otherwise} \end{cases}$


Computing one cell: OPT[2,18] = ?
G G G A A A A C C C A A A G G G G U U U
n= 20
(((. . . .))) (((. . . .)))
Case 3, 2 ≤ t <18-4:
t = 11: yes pair
OPT[2,18]≥1+2+0
GGAAAACCCAAAGGGGU
((.....))(.....)
(not shown:
t=9,10, 12,13)
$OPT(i,j) = \begin{cases} 0 & \text{if } i \geq j-4 \\ \max\left\{ \begin{matrix} OPT[i,j-1] \\ 1+\max_t(OPT[i,t-1]+OPT[t+1,j-1]) \end{matrix} \right\} & \text{otherwise} \end{cases}$


Computing one cell: OPT[2,18] = 4
n= 20
Overall, Max = 4 several ways, e.g.:
GGAAAACCCAAAGGGGU
..(...(((....)))).
tree shows trace back:
square = case 3
octagon = case 1
$OPT(i,j) = \begin{cases} 0 & \text{if } i \geq j-4 \\ \max\left\{ \begin{array}{c} OPT[i,j-1] \\ 1+\max_t(OPT[i,t-1]+OPT[t+1,j-1]) \end{array} \right\} & \text{otherwise} \end{cases}$


Another Trace Back Example
n = 16
E.g.:
OPT[1,16] = 3:
CUCCGGUUGCAAUGUC
((.(....).)..)..
0
$OPT(i,j) = \begin{cases} 0 & \text{if } i \geq j-4 \\ \max\left\{ \begin{array}{c} OPT[i,j-1] \\ 1+\max_t(OPT[i,t-1]+OPT[t+1,j-1]) \end{array} \right\} & \text{otherwise} \end{cases}$