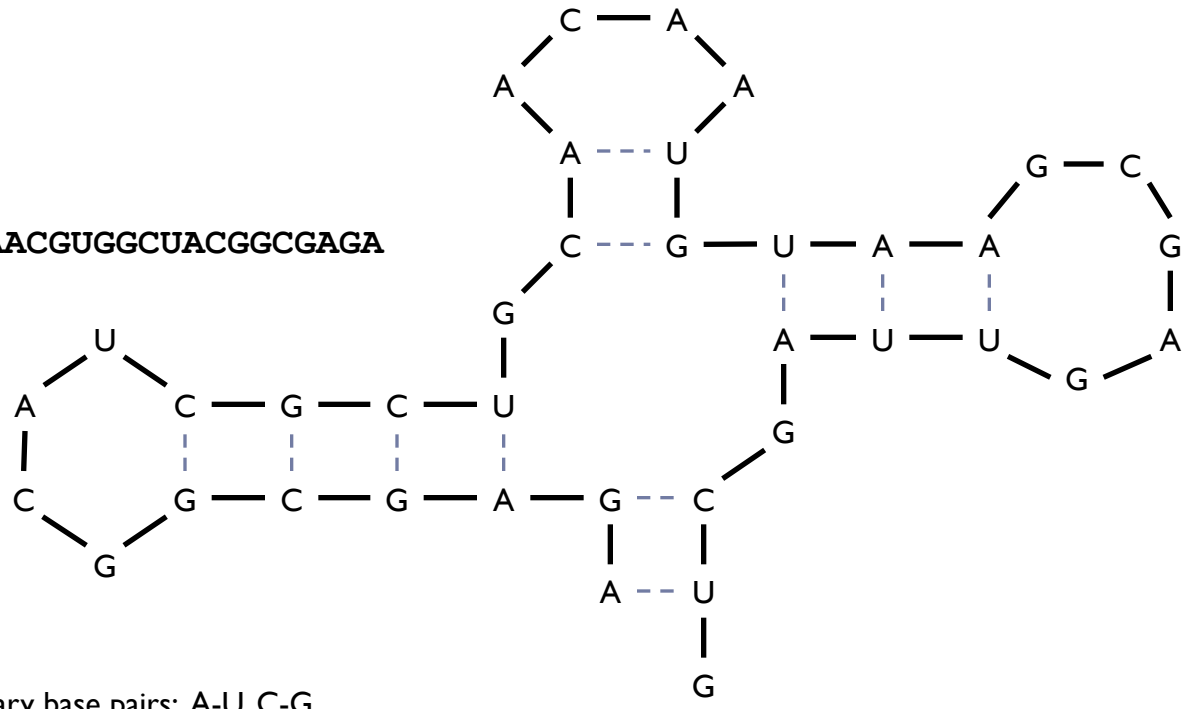
RNA Secondary Structure

Dynamic Programming over intervals

RNA Secondary Structure

- RNA: String $B = b_1 b_2 \ldots b_n$ over alphabet { A, C, G, U }.
- Secondary structure: RNA is single-stranded so it 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

- *Secondary structure*: A set of pairs $S = \{(bi, b_j)\}$ that satisfy:
  - [Watson-Crick.] S is a matching and each pair in S is a Watson-Crick complement: 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 $(bi, b_j) \in S$, then $j - i > 4$
  - [Non-crossing.] If (bi, b_j) and (b_k, b_l) are two pairs in S, then we cannot have $i < k < j < l$

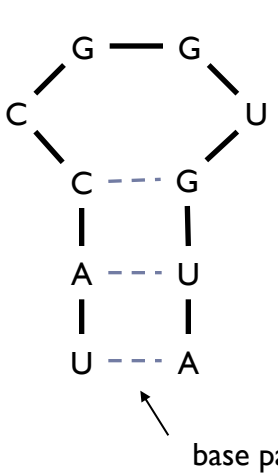
- *Free energy*: Usual hypothesis is that an RNA molecule will form the secondary structure with the minimum total free energy

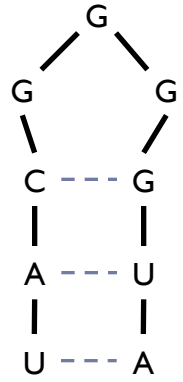
approximated by max number of unpaired bases

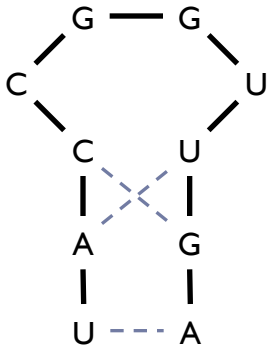
- *Goal*: Given an RNA molecule $B = b_1 b_2 \ldots b_n$, find a secondary structure S that maximizes the number of base pairs

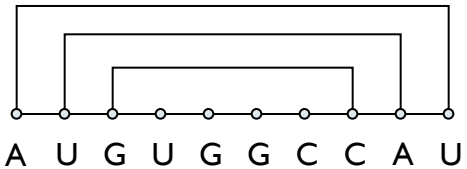
RNA Secondary Structure: Examples

- *Examples*:

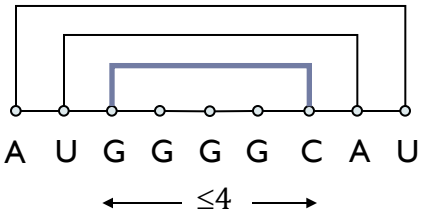




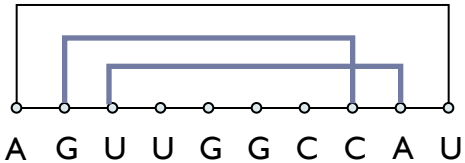




ok



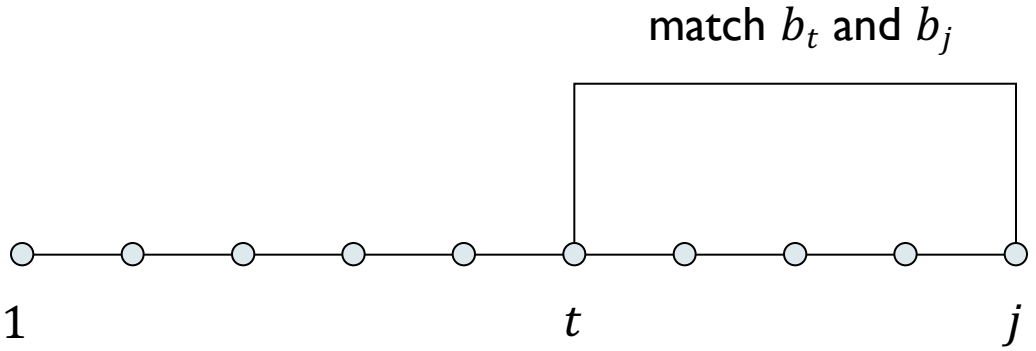
sharp turn



crossing

RNA Secondary Structure: Subproblems

- *First attempt*: $OPT(j)$ = maximum number of base pairs in a secondary structure of the prefix $b_1 b_2 \ldots b_j$



- *Difficulty*: Results in two sub-problems:
  - Finding secondary structure in: $b_1 b_2 \ldots b_{j-1}$ ← $OPT(t-1)$
  - Finding secondary structure in: $b_1 b_2 \ldots b_{t-1}$ and $b_{t+1} b_{t+2}$ … b_{j-1}

need more sub-problems

Dynamic Programming Over Intervals

- *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
    - $OPT(i, j) = 1 + max_t\{OPT(i, t-1) + OPT(t+1, j-1)\}$

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

DP relation: summary

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

↑

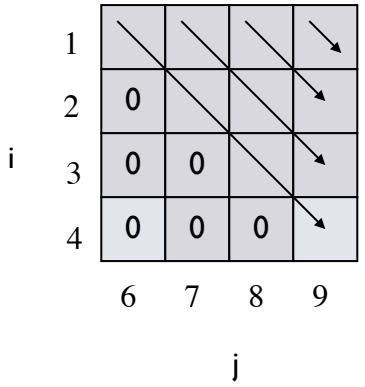
max over t such that $i \leq t < j-4$ and
b_t and b_j are Watson-Crick complements

Bottom Up Dynamic Programming Over Intervals

What order to solve the sub-problems?

Possible order: 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)
                                  using recurrence
    return OPT
}
```



Running time: $O(n^3)$

Exercise

- Which of the following orders of processing subproblems (i, j) allows a correct computation of values $OPT(i, j)$?
  - Increasing lexicographic order by pairs $(-i, j)$ (sorting by decreasing left end)
  - Increasing lexicographic order of pairs $(j - i, i)$ (from shortest to longest)
  - Increasing lexicographic order of pairs (j, i) (sorting by right end)
  - Increasing lexicographic order of pairs (i, j) (sorting by left end)

Exercise

- Which of the following orders of processing subproblems (i, j) allows a correct computation of values $OPT(i, j)$?
  - Increasing lexicographic order by pairs $(-i, j)$ (sorting by decreasing left end)
  - Increasing lexicographic order of pairs $(j - i, i)$ (from shortest to longest)
  - Increasing lexicographic order of pairs (j, i) (sorting by right end)
  - Increasing lexicographic order of pairs (i, j) (sorting by left end)