Dynamic Programming Algorithms

RNA Secondary Structures

CS 336: Design and Analysis of Algorithms

Bracket Matching

```
for (i in Union({ 1,...,n }, { n,...,2n }))
```

- Most modern integrated development environments (IDE) have a bracket matching feature.
- How to match brackets in a text?

Bracket Matching

```
for (i in Union({ 1,...,n }, { n,...,2n }))
```

- How to match brackets in a text? Use greedy.

```
stack OpenBrackets;
while (not end of document)
    Read next character c;
    if (c is an opening bracket) OpenBrackets.push(c);
    if (c is a closing bracket)
      d = OpenBrackets.pop();
      if c and d match, then output (c,d);
```

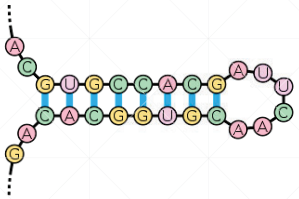
DNA & RNA Molecules

- DNA & RNA molecules are chains of nucleotides carrying the genetic information.
- DNA & RNA molecules or strands are usually represented as strings of letters/bases $\{A, C, G, T\}$ and $\{A, C, G, U\}$.

DNA strand: ACGTGACTAGCAC
RNA strand: ACGUGACUAGCAC

RNA Secondary Structures

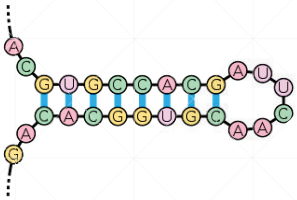
- RNA strands are represented as strings of letters $\{A, C, G, U\}$.
- Secondary structure is a matching between letters/bases.
-



RNA Secondary Structures

- RNA strands are represented as strings of letters $\{A, C, G, U\}$.
- Secondary structure is a matching between letters/bases.
- Rules:
  - Pairs $\{A, U\}$ and $\{C, G\}$ can match.
  - Non-crossing: if (i, j) and (k, l) match, then we cannot have $i < k < j < l$.

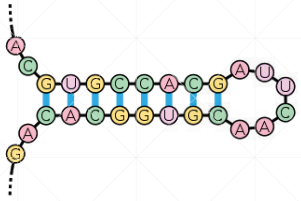
```
ACG xxxxx CGU
[{{ xxxxx }}]
```



RNA Secondary Structures

- RNA strands are represented as strings of letters $\{A, C, G, U\}$.
- Secondary structure is a matching between letters/bases.
- Rules:
  - Pairs $\{A, U\}$ and $\{C, G\}$ can match.
  - Non-crossing: if (i, j) and (k, l) match, then we cannot have $i < k < j < l$.

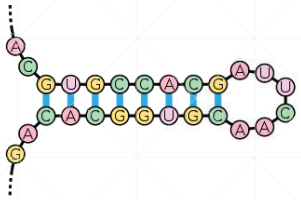
```
ACG xxxxx CGU
[{( xxxxx )}]
```



RNA Secondary Structures

- RNA strands are represented as strings of letters $\{A, C, G, U\}$.
- Secondary structure is a matching between letters/bases.
- Rules:
  - Pairs $\{A, U\}$ and $\{C, G\}$ can match.
  - Non-crossing: if (i, j) and (k, l) match, then we cannot have $i < k < j < l$.

```
ACG xxxxx GCU
[{( xxxxx })]
```



RNA Secondary Structures

RNA strands are strings of letters $\{A, C, G, U\}$.

Rules:

- Pairs $\{A, U\}$ and $\{C, G\}$ can match.
- Non-crossing: if (i, j) and (k, l) match, then we ***cannot*** have $i < k < j < l$.

Goal: Find maximum matching.

Challenge: Each base can be

- opening bracket;
- closing bracket;
- not a bracket.

RNA Secondary Structures

RNA strands are strings of letters $\{A, C, G, U\}$.

Rules:

- Pairs $\{A, U\}$ and $\{C, G\}$ can match.
- Non-crossing: if (i, j) and (k, l) match, then we cannot have $i < k < j < l$.

Goal: Find maximum matching.

Challenge: Each base can be

- opening bracket;
- closing bracket;
- not a bracket.

Can we use greedy?

Attempt #1: Greedy

- **Input:** ACCUGG
- **Greedy:**

Attempt #1: Greedy

- **Input:** ACCUGG
- **Greedy:** (ACCU)
- **Optimal solution:** A{C[CUG]G}

How to design a DP Algorithm?

1. Find one or several subproblems.
2. Express the cost of each subproblem in terms of smaller subproblems.
3. **Think:** What options do we have at every step?

Subproblem of Problem S

Subproblem: Find the maximum matching for the substring

$$S[i:j] = S[i], S[i+1], \dots, S[j]$$

What are our options at each step?

RNA strands are strings of letters $\{A, C, G, U\}$.

Rules:

- Pairs $\{A, U\}$ and $\{C, G\}$ can match.
- Non-crossing: if (i, j) and (k, l) match, then we cannot have $i < k < j < l$.

Given: a letter c should we match it with another letter?

…A…ACGUCGA…

- Match with one of the previous letters.
- Leave it unmatched.

Our Options for the Last Letter

Given: a letter c should we match it with another letter?

…A…ACGU

a. Match with one of the previous letters.

…A…(ACGU)
…(A…ACGU)

b. Don't match this letter.

Our Options for the Last Letter

Given: a letter c should we match it with another letter?

…A…ACGU

a. Match with one of the previous letters.

$\underbrace{\text{…}}_{\substack{\text{1st}\\\text{subproblem}}}$ (A $\underbrace{\text{… ACG}}_{\substack{\text{2nd}\\\text{subproblem}}}$ U)

Our Options for the Last Letter

Subproblem: Find maximum RNA matching in substring $\{i, \dots, j\}$.

$$OptionA = \max\{1 + OPT(i, t-1) + OPT(t+1, j-1)\}$$

where max is taken over $t \in \{i, \dots, j-1\}$ such that s_j match s_t.

i $\underbrace{\text{…}}_{\substack{\text{1st}\\\text{subproblem}}}$ (A t $\underbrace{\text{… ACG}}_{\substack{\text{2nd}\\\text{subproblem}}}$ U) j

Our Options for the Last Letter

Subproblem: Find maximum RNA matching in substring $\{i, \dots, j\}$.

$$OptionB = OPT(i, j-1)$$

DP for RNA Secondary Structures

for all intervals $\{i, \dots, j\}$.
  Compute OptionA.
  Compute OptionB.
  Let $\mathrm{OPT}(i, j)$ = max (OptionA, OptionB).
return $\mathrm{OPT}(1, n)$

DP for RNA Secondary Structures

for all intervals $\{i, \dots, j\}$.
  Compute OptionA.
  Compute OptionB.
  Let $\mathrm{OPT}(i, j)$ = max (OptionA, OptionB).
return $\mathrm{OPT}(1, n)$

What is the right order for intervals/subproblems?

- from small to large (i.e., sort by the length $j - i + 1$)

DP for RNA Secondary Structures

for all intervals $\{i, \dots, j\}$.
  Compute OptionA.
  Compute OptionB.
  Let $\mathrm{OPT}(i,j)$ = max (OptionA, OptionB).
return $\mathrm{OPT}(1,n)$

Running time? $O(n^3)$

$O(n^2)$ subproblems; $O(n)$ to find the best Option A.

More constraints on RNA secondary structure

Rules:

- Pairs $\{A,U\}$ and $\{C,G\}$ can match.
- Non-crossing: if (i,j) and (k,l) match, then we cannot have
$$i < k < j < l$$
- No sharp turns: Positions of matched letters should differ by at least 4.

How to fix the algorithm?

Discussion & Questions