Lec22-alignment-rna-folding

KT 6.5, 6.8

Sunday, October 27, 2024 10:40 PM

String comparison is an essential task in bioinformatics.
- comparing DNA or protein sequences may reveal evolutionary relationships
- useful for taxonomy, but also functional predictions

Problem: Given two strings $A = a_1 a_2 a_3 \cdots a_m$, $B = b_1 b_2 b_3 \cdots b_n$,

measure some similarity or distance b/t A & B.

Many different possible measures of similarity / distance, but let's focus on just one, the (Levenshtein) edit distance.

Edit distance: $d(A, B)$ = minimum number of (single-char) mutations, insertions, deletions to transform A into B.

Ex. apple → (del a) pple → (del p) ple → (del l) pe → (ins a) pea → (ins r) pear

Alternate: pele → (l→a) peae → (e→r) pear (p→e)

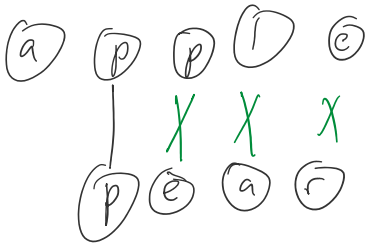
String alignment: Consider the bipartite graph where on one side is all a_i and on the other side all b_j, and edge (a_i, b_j) exists if $a_i = b_j$.

We may add extra edges (a_i, b_j) for $a_i \neq b_j$ at a cost.

Find a bipartite matching such that if (a_{i_1}, b_{j_1}), (a_{i_2}, b_{j_2}) are chosen, then $i_1 < i_2$, then $j_1 < j_2$, (no crossings)

where each non-matched item incurs a gap cost, and minimizing total cost.

Ex



Additionally, we may choose indels to cost a different penalty than mismatches.

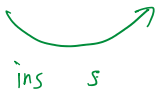
Representing edits as alignments

Ex. mispell, misspell

m i s – p e l l

m i s s p e l l

Ex, mispell, misspell




```
m i s - p e l l
| | |   | | | |
m i s s p e l l
```

1 gap

Spite, suite

```
s p i t e
| x | | |
s u i t e
```

1 mismatch (mm)

```
s p - i t e
|     | | |
s - u i t e
```

2 gaps

principle, princpal

```
prin - ciple
|||| | ||| xx
prinncipal
```

1 gap, 2 mismatch (mm)

```
prin-cip-le
|||| ||| |
prinncipal-
```

3 gaps

Generalized edit distance + alignment cost

Instead of just counting total numbers of edits / matchings, we can assign diff. costs:

Let GAP be cost(-), the cost of an indel or gap.

Let cost(x,y) be the cost of aligning char "x" with "y",
i.e. a substitution cost or mismatch penalty.

Simplest case all costs of mismatches the same.

Goal: compute lowest cost alignment / edit distance

↳ cost of alignment = sum of cost(x,y) for pairs of aligned characters + gap × # unmatched

Algorithm: Consider the last characters of strings

$A = a_1 a_2 \cdots a_m$

$B = b_1 b_2 \cdots b_n$

Either: (1) (a_m, b_n) are matched to each other (either correctly if $a_m = b_n$ or incorrectly with cost (a_m, b_n))

(2) a_m is not matched at all

(3) b_n is not matched at all

~~(4) a_m is matched to some b_j $(j \neq n)$ and b_n is matched to some a_k $(k \neq m)$.~~ because no crossings

Recursion:

$$OPT(i,j) = \min \begin{cases} \text{cost}(a_i, b_j) + OPT(i-1, j-1) \\ GAP + OPT(i-1, j) \\ GAP + OPT(i, j-1) \end{cases}$$

match a_i, b_j

a_i not matched

b_j not matched

don't know actual solution, so try all 3.

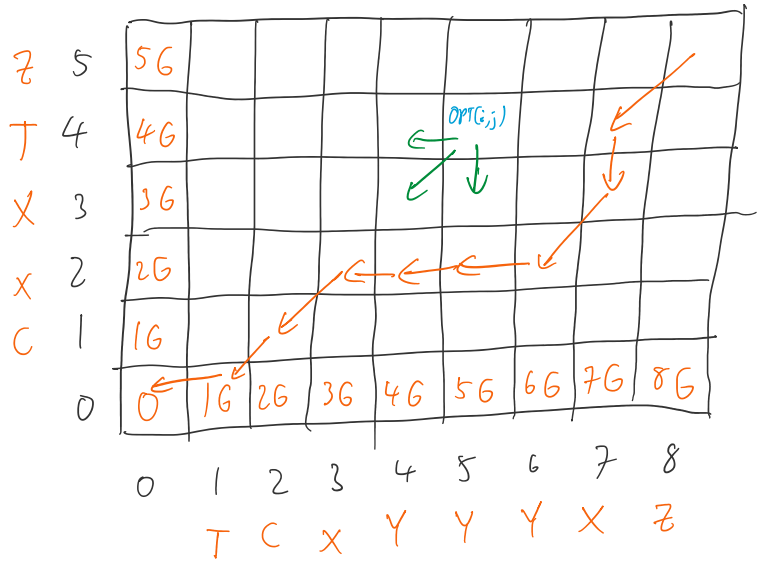
Base case:

$OPT(i, 0) = i \cdot GAP$

$OPT(0, j) = j \cdot GAP$

aligning i characters to 0 char must use i gaps.

DP array



Back track to find actual alignment

move right = add gap to B, char from A

move up = add gap to A, char from B

move diag = choose both chars

A: T C X Y Y Y X – Z

B: – C X – – – X T Z

i

Psentocode:

EditDistance (A, B):

For $i = 1, \dots, m$: $M[i, 0] = i \cdot GAP$

For $j = 1, \dots, n$: $M[0, j] = j \cdot GAP$

For $i = 1 \dots m$:

For $j = 1 \dots n$:

$$M[i,j] = \min \begin{cases} cost(A[i], B[j]) + M[i-1, j-1] \\ GAP + M[i-1, j] \\ GAP + M[i, j-1] \end{cases}$$

Return $M[m,n] = OPT(m,n)$ ↖ edit distance b/t A & B } backtrack to get alignment

Runtime $O(1)$ time to fill in each matrix entry.

$O(mn)$ running time.

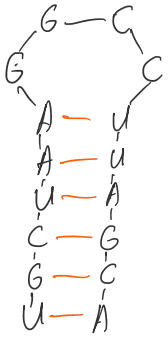
RNA folding

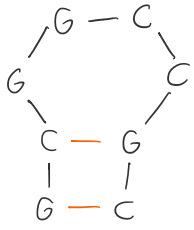
RNA is a single strand that folds up:

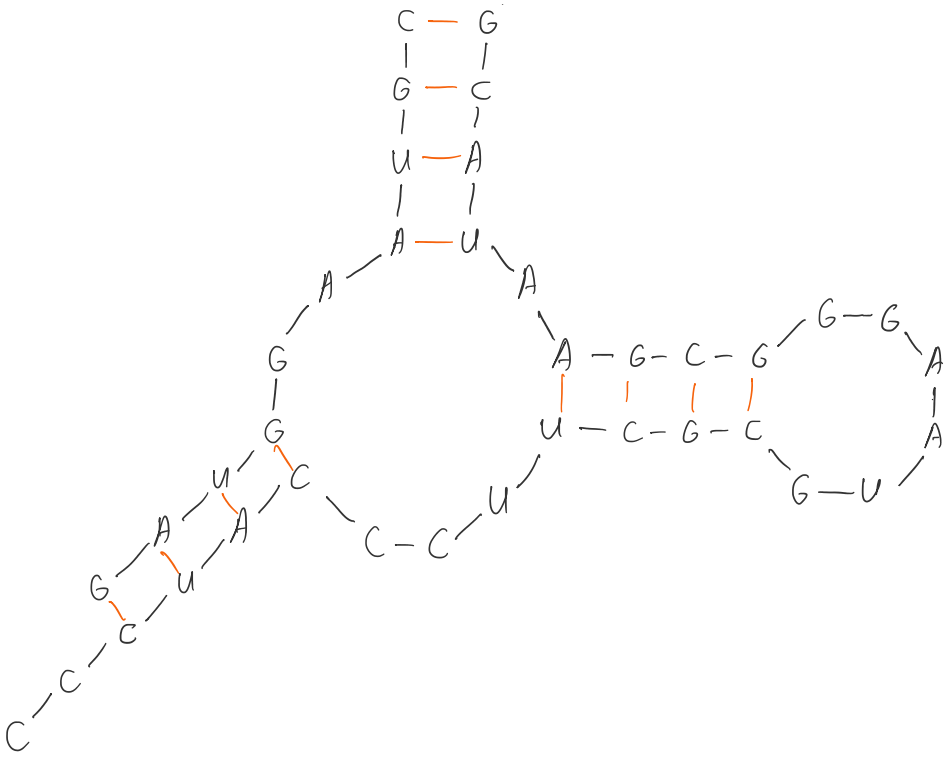
- G & C stick together
- A & U stick together
- Bases closer than 4 cannot pair
- Pairs cannot cross. If (i,j) & (k,l) are paired, $i < k < l < j$. (nested)

$\{A, C, G, U\}$

Goal: Given string $r = b_1 b_2 \cdots b_n$ find largest set of pairs $S = \{(i,j)\}$ satisfying pairing rules.



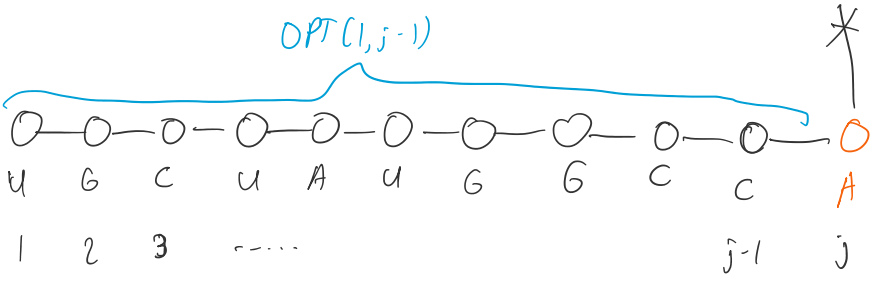




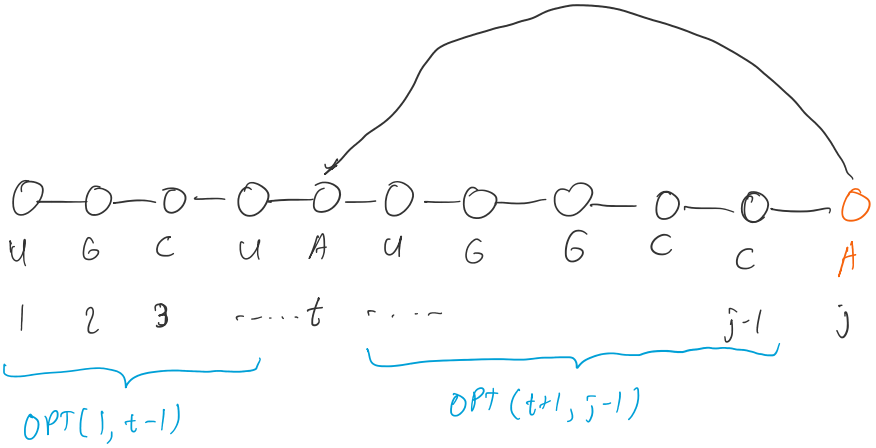
Subproblems:

Consider possibilities for the jth character.

j is not paired



j is paired with some $t \le j-4$



We have a subproblem for every interval (i, j).

$\binom{n}{2} = O(n^2)$ subproblems

Recursion: Base case: If $j - i \le 4$, $OPT(i,j) = 0$

Recursion: If $j - i > 4$:

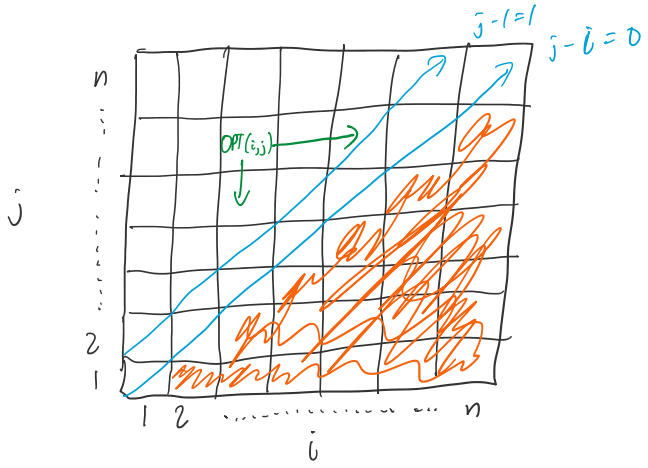
$$OPT(i,j) = \max \begin{cases} OPT(i, j-1) \\ \max_{t = i, \dots, j-4} \{ 1 + OPT(i, t-1) + OPT(t+1, j-1) \} \end{cases}$$

↑ need to try all possible t to pair with j.

What makes a subproblem "simpler"? Size of interval $j - i$.

Need to potentially solve all smaller intervals before bigger one.

DP Matrix Only use half $i < j$



$O(n^2)$ subproblems.

Each takes $O(n)$ time to solve.

$\Rightarrow O(n^3)$ running time.