CSE 566 Spring 2023

Multiple Sequence Alignment

Instructor: Mingfu Shao

What We've Learned

Problem	Algorithm
Querying P against text T	Z-algo, KMP
Constructing suffix tree	Ukkonen's algorithm
Constructing suffix array	Skew algorithm
Querying q against an suffix array	O(|q| + log|S|)-time algorithm
Constructing/Recovering/Querying BWT	O(|s|)-time algorithms
Constructing local/glocal alignments	O(nm)-time algorithms
Edit distance problem	O(nd)-time, O(n)-space
...	...

Tackling NP-hard Problems

- NP-hard: there does not exist polynomial-time exact/optimal algorithm unless P = NP.
- Still need exact/optimal solution
  - exponential algorithms
- Don't require exact/optimal solution, but some theoretical guarantee
  - Approximation algorithms: $algo(I)/OPT(I) \leq c$, for any I.
  - Randomized algorithms
- Heuristics, often need experimental comparisons

Multiple Sequence Alignment (MSA)

```
S1:ATCGACTACTCGTC    ATC-GACTACTC-GTC
S2:ATCGGACTCGTT      ATCGGA---CTC-GTT
S3:ACGGACACTCGTT     A-CGGAC-ACTC-GTT
S4:ACGGCTACTCAGTT    A-CGG-CTACTCAGTT
```

- Widely used to find common patterns, conserved regions, comparative analysis, etc.
- Input: sequences S1, S2, ..., Sm
- Output: an MSA M, such the cost of M is minimized.
- Different cost functions lead to different formulations.

The First Formulation

- Assume that columns of M are independent
- $cost(M) = \sum_c cost(M_c)$, where M_c is the c-th column of M
- This cost function generalizes the unit gap cost of 2 sequences to m sequences
- This formulation is NP-hard.

```
A T C - G A C T A C T C - G T C
A T C G G A - - - C T C - G T T
A - C G G A C - A C T C - G T T
A - C G G - C T A C T C A G T T
```

Dynamic Programming Algorithm

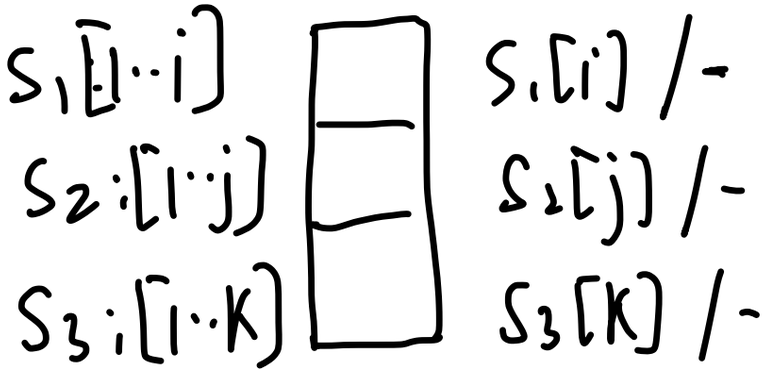
- Take m=3 as example. Define OPT(i, j, k) as the minimized cost between S1[1..i], S2[1..j], and S3[1..k]

$O(n^3)$

- Consider all possibilities of the last column

$$OPT(i,j,k) = \min \begin{cases} cost(S_1[i], S_2[j], S_3[k]) + OPT(i-1, j-1, k-1) \\ cost(S_1[i], S_2[j], -) + OPT(i-1, j-1, k) \\ cost(S_1[i], -, S_3[k]) + OPT(i-1, j, k-1) \\ cost(S_1[i], -, -) + OPT(i-1, j, k) \\ cost(-, S_2[j], S_3[k]) + OPT(i, j-1, k-1) \\ cost(-, S_2[j], -) + OPT(i, j-1, k) \\ cost(-, -, S_3[k]) + OPT(i, j, k-1) \end{cases}$$

last column



Dynamic Programming Algorithm

- Generalizing to m sequence:
  - #(subproblems) = n^m, where assuming $|S_i| = n$
  - Solving each subproblem takes $O(2^m)$ cases,
    - Assume cost of a column can be computed in O(m) time.
- Total running time: $O(m \cdot n^m \cdot 2^m)$
- Only affordable for very small m.

$m=4 \Rightarrow O(4 \cdot n^4 \cdot 2^4)$

The Second Formulation

- An MSA M implies all pairwise alignments.

MSA M

```
S1 : ATC-GACTACTC-GTC
S2 : ATCGGA---CTC-GTT
S3 : A-CGGAC-ACTC-GTT
S4 : A-CGG-CTACTCAGTT
```

$$Cost(M) = M_{12} + M_{13} + M_{23} + M_{14} + M_{24} + M_{34}$$

```
S2 : ATCGGA--CTCGTT
S3 : A-CGGACACTCGTT
```

M_{23}

```
S2 : ATCGGA---CTC-GTT
S4 : A-CGG-CTACTCAGTT
```

M_{24}

- The implied pairwise alignment may not be optimal.

The Second Formulation

- Assume the cost function for alignment between two sequences is given (say, unit gap cost, affine gap cost, etc)
- Assume that this cost function is a metric, i.e., satisfying the triangle inequality: $cost(M_{ij}) \leq cost(M_{ik}) + cost(M_{kj})$, $\forall k$.
- Sum-of-pairs cost: $cost(M) = \sum_{i<j} cost(M_{i,j})$
- We are able to design a 2-approximation algorithm (STAR):

$$\frac{STAR(I)}{OPT(I)} = \frac{\sum_{i<j} cost(M_{i,j})}{\sum_{i<j} cost(M^*_{i,j})} \leq 2$$

M

OPT: M^*

STAR Algorithm

$O(m^2 \cdot n^2)$

- Step 1: build all optimal pairwise alignments; let $M(S_i, S_j)$ be the alignment and let $cost(S_i, S_j)$ be the cost.
- Step 2: find the sequence that is the closest to others, i.e., S_x such that $\sum_{i \neq x} cost(S_i, S_x)$ is minimized.
- Step 3: merge $M(S_1, S_x), M(S_2, S_x), \cdots, M(S_m, S_x)$ into an MSA, called M.

Merging $M(S_i, S_x), 1 \leq i \leq m$

- Keep all columns; add "-" as needed

```
S1:ATC-GACTACTCGTC
S2:ATCGGA---CTCGTT
```

S_x = S2) MSA

```
S2:ATCGGA---CTCGTT
S3:A-CGGACACTCGTT
```

```
S2:ATCGGA---CTC-GTT
S4:A-CGG-CTACTCAGTT
```

```
S1:ATC-GACTACTCGTC
S2:ATCGGA---CTCGTT
S3:A-CGGAC-ACTCGTT
```

```
S1:ATC-GACTACTC-GTC
S2:ATCGGA---CTC-GTT
S3:A-CGGAC-ACTC-GTT
S4:A-CGG-CTACTCAGTT
```

Proof of Approximation Ratio

$$2 \cdot cost(M) = 2 \cdot \sum_{i<j} cost(M_{i,j}) = \sum_{i,j} cost(M_{i,j})$$

$$\leq \sum_{i,j} (cost(M_{i,x}) + cost(M_{x,j}))$$

triangle inequality

step 3

$$= \sum_{i,j} (cost(S_i, S_x) + cost(S_x, S_j))$$

$$= \sum_{i,j} cost(S_i, S_x) + \sum_{i,j} cost(S_x, S_j)$$

$$= 2m \cdot \sum_i cost(S_i, S_x)$$

$\sum_j [\sum_i cost(S_i, S_x)]$

$$2 \cdot cost(M^*) = 2 \cdot \sum_{i<j} cost(M^*_{i,j}) = \sum_{i,j} cost(M^*_{i,j})$$

$$\geq \sum_{i,j} cost(S_i, S_j)$$

$$= \sum_i cost(S_i, S_1) + \sum_i cost(S_i, S_2) + \cdots + \sum_i cost(S_i, S_m)$$

$$\geq m \cdot \sum_i cost(S_i, S_x)$$

step 2

Transform MSA to Profile

n columns

```
A T C - G A C T A
A T C G G A - - -
A - C G G A C - A
A - C G G - C T A
A T - G T - C G -
```

m seqs

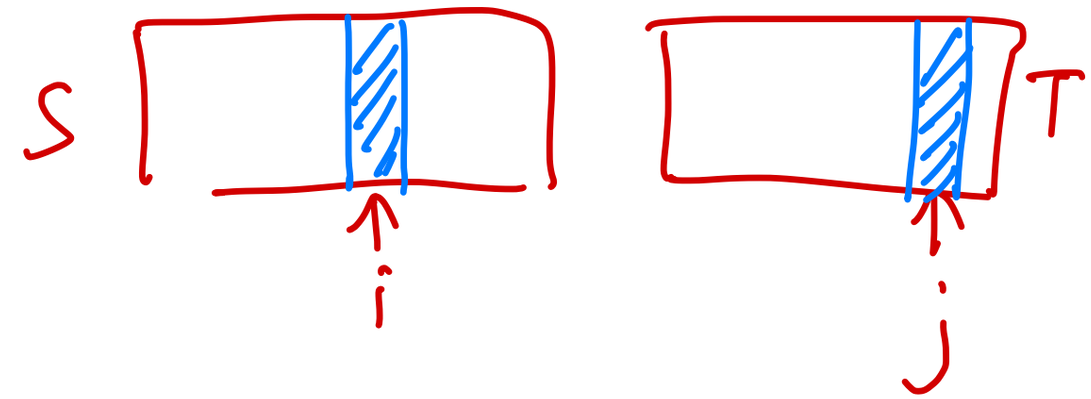
MSA

m x n

A	1	0	0	0	0	0.6	0	0	0.6
C	0	0	0.8	0	0	0	0.8	0	0
G	0	0	0	0.8	0.8	0	0	0.2	0
T	0	0.6	0	0	0.2	0	0	0.4	0
-	0	0.4	0.2	0.2	0	0.4	0.2	0.4	0.4

5 x n

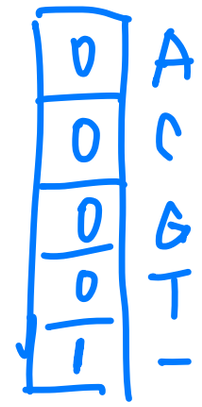
Aligning Two Profiles



- Input: two profiles S and T
- DP algorithm:

$$OPT(i,j) = \max \begin{cases} OPT(i-1,j-1) + score(S[i], T[j]) \\ OPT(i-1,j) + score(S[i], -) \\ OPT(i,j-1) + score(-, T[j]) \end{cases}$$

$$score(S[i], T[j]) = \sum_{c \in \Sigma} S[i][c] \cdot T[i][c]$$



- More sophisticated approaches exist, such as profile HMM; see more in "Biological sequence analysis", by R. Durbin, et al.

Progressive Approach for MSA

- Step 1: build all pairwise alignments to get the cost matrix.
- Step 2: build a guide tree (clustering, phylogeny, etc).
- Step 3: progressively align two profiles following the tree.