EXERCISE 5: RESTRICTION MAPS

Programming in Bioinformatics

The Double-Digest Problem

The DNA molecule is completely digested by enzyme A, enzyme B, and a combination of both enzymes. The input to the problem consists of three multisets of fragments generated by digestion sorted by increasing size:

$\Delta X_A = \{a_1, a_2, ..., a_m\}$ - multiset of fragments generated by digestion by enzyme A

$\Delta X_B = \{b_1, b_2, ..., b_n\}$ - multiset of fragments generated by digestion by enzyme B

$\Delta X_{AB} = \{c_1, c_2, ..., c_{m+n-1}\}$ - multiset of fragments generated by digestion by enzymes A + B

The task is to find the positions for cleavage by A and B such that the multiset of fragments ΔX_{AB} is generated.

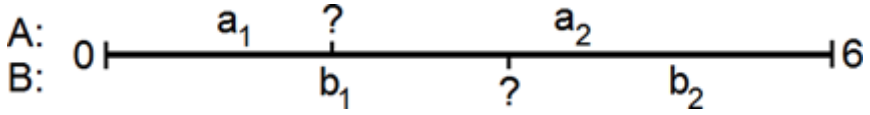
Example 1

$\Delta X_A = \{2, 4\}$

$\Delta X_B = \{1, 5\}$

$\Delta X_{AB} = \{1, 1, 4\}$

Cleavage by enzymes A and B produces only two fragments, meaning that DNA contains one restriction site for each of the enzymes, and the length of the molecule is 6 (the sum of fragment lengths).



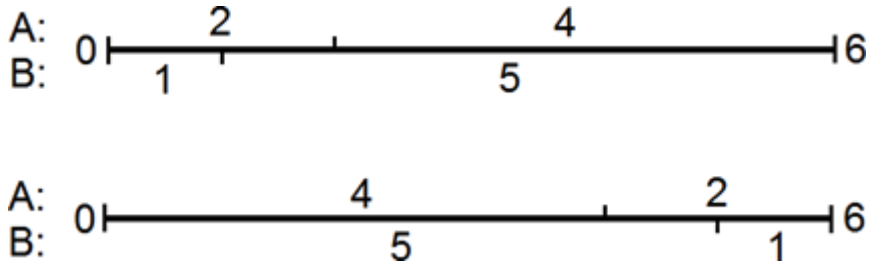
Simultaneous cleavage by both enzymes produces three fragments. It is necessary to determine the locations of the restriction sites on the DNA molecule. The number of fragments *m* for enzyme A and *n* for B determines the number of combinations of restriction site positions.

Specifically, for this example: *m*! * *n*! = 2! * 2! = 4 combinations of fragment positions:

2 4	4 2	2 4	4 2
1 5	1 5	5 1	5 1

Which of these combinations generates multiset X_{AB}?

Correct solution: X_A = {2}, X_B = {1}



Reversion of one solution creates another correct solution: X_A = {4}, X_B = {5}

Enzyme A cleaves DNA at position 2, and enzyme B at position 1, or at positions 4 and 5.

Example 2

ΔX_A = {2, 3, 5, 10}

ΔX_B = {3, 7, 10}

ΔX_{AB} = {1, 2, 2, 5, 5, 5}

Enzyme A has 3 restriction sites on the DNA, enzyme B has 2 sites, and the DNA is 20 bp long. It is necessary to verify 4! * 3! = 144 combinations of positions.

The exhaustive search (brute-force) algorithm checks all possible position combinations, for example in the following manner:

1) fragment arrangements

   Take multisets of fragments ΔX_A and ΔX_B and create an arrangement of fragments for both enzymes, e.i. create a combination of fragments to be verified.

   fragments for digestion by enzyme A: `3, 5, 2, 10`
   fragments for digestion by enzyme B: `7, 3, 10`

2) position maps

   Create a map of positions starting at 0, successive positions are calculated by cumulative sum of fragments sizes from fragment arrangement.

fragment arrangement		position maps
A: `3, 5, 2, 10`	`0 + 3, 3 + 5, 8 + 2, 10 + 10`	`0, 3, 8, 10, 20`
B: `7, 3, 10`	`0 + 7, 7 + 3, 10 + 10`	`0, 7, 10, 20`

3) combined positions

   Merge position maps for both enzymes and keep only unique values.

position maps	combined positions
A: `0, 3, 8, 10, 20`	`0, 3, 7, 8, 10, 20`
B: `0, 7, 10, 20`	

4) successive differences

   Calculate successive differences by subtraction of adjacent combined positions values

combined positions
`0, 3, 7, 8, 10, 20`

successive differences
`3 - 0, 7 - 3, 8 - 7, 10 - 8, 20 - 10`
`3, 4, 1, 2, 10`

5) sorted successive differences
   Sort successive differences in ascending order.

successive differences	sorted successive differences
`3, 4, 1, 2, 10`	`1, 2, 3, 4, 10`

6) compare sorted successive differences with ΔX_{AB}
   Fragment arrangements for enzymes A and B are arranged correctly if sorted successive differences are equal to fragments of double digestion (ΔX_{AB}).

   ΔX_{AB} = {1, 2, 2, 5, 5, 5} ≠ {1, 2, 3, 4, 10}

   In this case, current positions of restriction sites are not the correct solution because created fragment sizes are not equal ΔX_{AB}.

Another combination of fragments:

fragment arrangements	position maps	combined positions
A: `2, 3, 10, 5`	`0, 2, 5, 15, 20`	`0 2 3 5 10 15 20`
B: `3, 7, 10`	`0, 3, 10, 20`	

successive differences
`2 − 0, 3 − 2, 5 − 3, 10 − 5, 15 − 10, 20 − 15 = 2, 1, 2, 5, 5, 5`
sorted successive differences
`1, 2, 2, 5, 5, 5 =` ΔX_{AB} `= {1, 2, 2, 5, 5, 5}`

Thus, the restriction sites for enzyme A are {2, 5, 15} and for enzyme B {3, 10}, and additional solutions are their reversals {5, 15, 18} and {10, 17}. Reversals are formed by reversing the order of the fragment arrangements and creating a position map.

This approach of testing all possible combinations of fragment arrangements will find all possible positions of restriction sites. However, the algorithm is enormously computationally intensive.

Task 1

- In R, create a function for the brute-force DDP algorithm for one possible arrangement of fragments.
- Modify it to work with all possible arrangements of fragments.

The Partial Digest Problem

The DNA molecule is digested incompletely by one restriction enzyme. The input for this problem is a multiset of fragments generated by incomplete (partial) digestion $\Delta X = \{x_1, x_2, \ldots, x_m\}$.
The task is to find a set of positions X from which the multiset of fragments was generated: $X = \{x_1 = 0, x_2, \ldots, x_n\}$, where $\Delta X = \{x_j - x_i : 1 \le i < j \le n\}$ and n is the length of the molecule.

The number of restriction sites N can be estimated from equation $m = \frac{N(N-1)}{2}$, where m is the number of fragments in the multiset ΔX.

Example 1

$\Delta X = \{2, 2, 3, 3, 4, 5, 6, 7, 8, 10\}$

The size of ΔX is 10; therefore, according to $10 = \frac{N(N-1)}{2} \rightarrow N = 5$, we are looking for $X = \{x_1, x_2, x_3, x_4, x_5\}$. The vector of lengths ΔX was generated according to this table:

	x_1	x_2	x_3	x_4	x_5
x_1		$x_2 - x_1$	$x_3 - x_1$	$x_4 - x_1$	$x_5 - x_1$
x_2			$x_3 - x_2$	$x_4 - x_2$	$x_5 - x_2$
x_3				$x_4 - x_3$	$x_5 - x_3$
x_4					$x_5 - x_4$
x_5					

1) We find the maximum in ΔX. The maximum element must, assuming that the vector X is sorted in ascending order, correspond to the element $x_5 - x_1$ from the table.
   max = 10 → $x_5 - x_1 = 10$, when $x_1 = 0$, then $x_5 = 10$, therefore $X = \{0, x_2, x_3, x_4, 10\}$.
   Remove the element $x_5 - x_1$ from ΔX and get the new $\Delta X = \{2, 2, 3, 3, 4, 5, 6, 7, 8\}$.

2) max = 8 → We have two possibilities: $x_4 - x_1 = 8, x_4 = 8$ $x_5 - x_2 = 8, x_2 = 2$.
   Since both options have the same distance for the first or last value of X, we can choose either of them.
   Choose $x_2 = 2$, then new $X = \{0, 2, x_3, x_4, 10\}$.
   Remove elements with values: $x_2 - x_1 = 2$, $x_5 - x_2 = 8$, then new $\Delta X = \{2, 3, 3, 4, 5, 6, 7\}$.

3) max = 7 → There are two possibilities: $x_4 - x_1 = 7, x_4 = 7$ $x_5 - x_3 = 7, x_3 = 3$.
   If $x_3 = 3$, then $x_3 - x_2 = 1$, but 1 is not in ΔX, so we choose $x_4 = 7$,
   then new $X = \{0, 2, x_3, 7, 10\}$.
   Remove elements with values: $x_5 - x_4 = 3$, $x_4 - x_1 = 7$, and $x_4 - x_2 = 5$,
   then new $\Delta X = \{2, 3, 4, 6\}$.

4) max = 6 → There are two possibilities left: $x_3 - x_1 = 6, x_3 = 6$ $x_5 - x_3 = 6, x_3 = 4$.
   If $x_3 = 6$, then $x_4 - x_3 = 1$, which is not in ΔX, so we choose $x_3 = 4$.

Result: $X = \{0, 2, 4, 7, 10\}$

We perform a correctness check. We calculate all possible distance combinations between elements, and after sorting, they must match the given multiset.
ΔX = {2, 4, 7, 10, 2, 5, 8, 3, 6, 3} = {2, 2, 3, 3, 4, 5, 6, 7, 8, 10}.

More examples

1) ΔX = {1, 2, 2, 3, 4, 4, 5, 6, 7, 8}
2) ΔX = {2, 2, 2, 4, 4, 4, 6, 6, 8, 10}
3) ΔX = {1, 2, 2, 2, 2, 3, 4, 4, 4, 5, 6, 6, 7, 8, 9}

Task 2

- In R, implement a recursive algorithm for the Partial Digest Problem (PDP) according to the following pseudocode.

```
PartialDigestProblem(deltaX)
1 width ← the maximum element from deltaX
2 delete the maximum element from deltaX
3 X ← {0, width}
4 Place(deltaX, X, width)
```

```
Place(deltaX, X, width)
1   if deltaX is empty
2     output X
3     return
4   y ← the maximum element from deltaX
5   if Δ(y, X) ⊆ deltaX
6     add y to X and remove the lengths Δ(y, X) from deltaX
7     Place(deltaX, X, width)
8     remove y from X and add the lengths Δ(y, X) to deltaX
9   if Δ(width - y, X) ⊆ deltaX
10    add width - y to X and remove the lengths Δ(width - y, X) from deltaX
11    Place(deltaX, X, width)
12    remove width - y from X and add the lengths Δ(width - y, X) to deltaX
13  return
```

Notes:

- `deltaX` = ΔX = multiset of fragments lengths
- `Δ(y, X)` is a multiset of lengths between value y and all values in X.
- Hint: Create additional function `Remove()`, which removes from `deltaX` all used lengths `Δ(y, X)`.