

5. Počítačové cvičení

Restrikční mapování

Double Digest Problem

(Kniha 3, kapitola 4.1 až 4.4)

Molekula DNA je kompletně štěpena enzymem A, enzymem B a kombinací obou enzymů.

Vstupem problému jsou tři vektory vzestupně seřazených fragmentů štěpení:

$\Delta X_A = \{a_1, a_2, \dots, a_m\}$ - vektor fragmentů po štěpení enzymem A,

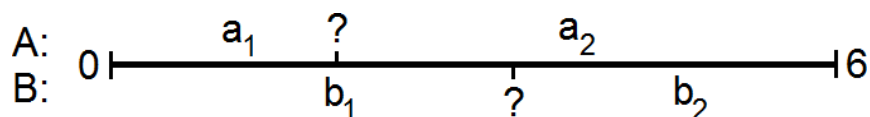
$\Delta X_B = \{b_1, b_2, \dots, b_n\}$ - vektor fragmentů po štěpení enzymem B,

$\Delta X_{AB} = \{c_1, c_2, \dots, c_{m+n-1}\}$ - vektor fragmentů po současném štěpení enzymy A+B.

Úkolem je nalézt pozice štěpení pro oba enzymy.

Triviální příklad: $\Delta X_A = \{2, 4\}$, $\Delta X_B = \{1, 5\}$ a $\Delta X_{AB} = \{1, 1, 4\}$.

Štěpení enzymem A i B produkuje pouze dva fragmenty, což znamená, že DNA obsahuje jedno restrikční místo pro každý z enzymů a délka molekuly je 6 (suma délek fragmentů).



Štěpení oběma enzymy současně produkuje tři fragmenty. Je potřeba určit, kde na molekule DNA se nachází restrikční místa.

Počet fragmentů m pro enzym A a n pro B určuje počet kombinací pozic restrikčních míst.

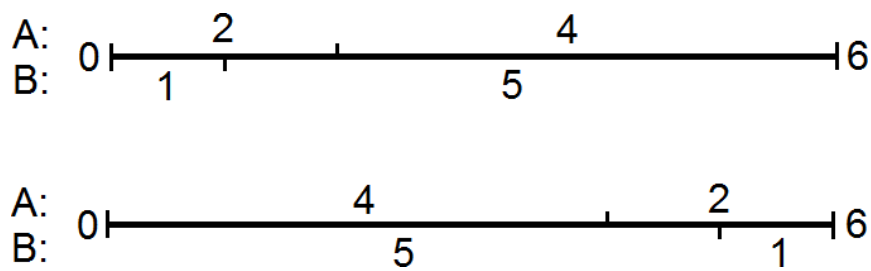
Konkrétně pro příklad: $(m!)*(n!) = (2!)*(2!) = 4$ kombinace pozic fragmentů:

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

Která z těchto kombinací generuje vektor X_{AB} ?

Správné řešení:

(reverze jednoho řešení je dalším řešením)



Enzym A štěpí DNA na pozici 2 a enzym B na pozici 1, nebo na pozici 4 a 5.

Další příklad:

$$\Delta X_A = \{2, 3, 5, 10\}$$

$$\Delta X_B = \{3, 7, 10\}$$

$$\Delta X_{AB} = \{1, 2, 2, 5, 5, 5\}$$

Enzym A má na DNA 3 restrikční místa, enzym B 2 místa, DNA je dlouhá 20. Je potřeba prověřit $(4!)*(3!) = 144$ kombinací pozic.

Exhaustive search (brute-force) algoritmus prověří všechny možnosti pozic, např. tímto způsobem:

uspořádání fragmentů	mapa pozic	sloučené pozice
3 5 2 10	0 3 8 10 20	0 3 7 8 10 20
7 3 10	0 7 10 20	
postupné difference		
$3 - 0, 7 - 3, 8 - 7, 10 - 8, 20 - 10 = 3, 4, 1, 2, 10$		

setřídít na 1, 2, 3, 4, 10 toto ale není správné řešení, neboť se to nerovná X_{AB} .

Mapa pozic se tvoří postupným sčítáním fragmentů a sloučené pozice neobsahují duplikace.

Pro jinou kombinaci:

uspořádání fragmentů	mapa pozic	sloučené pozice
2 3 10 5	0 2 5 15 20	0 2 3 5 10 15 20
3 7 10	0 3 10 20	
postupné difference		
$2 - 0, 3 - 2, 5 - 3, 10 - 5, 15 - 10, 20 - 15 = 2, 1, 2, 5, 5, 5$		

setřídít na 1, 2, 2, 5, 5, 5 a toto se rovná X_{AB} a tudíž restrikční místa pro enzym A jsou {2, 5, 15} a enzym B {3, 10} a dalším řešením je jejich reverze {5, 15, 18} a {10, 17}. Reverze se tvoří obrácením pořadí uspořádání fragmentů a vytvořením mapy pozic.

Tento přístup testování všech možných kombinací uspořádání fragmentů nalezne všechny možné pozice restričních míst. Algoritmus je však enormně výpočetně náročný.

Úkol: V R naprogramujte funkci pro brute-force algoritmus DDP pro jedno možné uspořádání fragmentů. Následně upravte pro všechny možné uspořádání fragmentů.

Partial Digest Problem

Řešený příklad:

Máme vektor délek $\Delta X = \{2, 2, 3, 3, 4, 5, 6, 7, 8, 10\}$. Najděte vektor X , z kterého byl vektor délek vygenerován. $X = \{x_1=0, x_2, \dots, x_n\}$, kde $\Delta X = \{x_j - x_i : 1 \leq i < j \leq n\}$.

Velikost $\Delta X = 10$ tudíž podle $\frac{n(n-1)}{2} = 10 \rightarrow n = 5$, tudíž hledáme $X = \{x_1, x_2, x_3, x_4, x_5\}$. Vektor délek ΔX byl vygenerován podle této tabulky:

	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) Nalezneme maximum v ΔX . Maximální prvek musí za předpokladu, že vektor X je setříděn vzestupně, odpovídat prvku z tabulky $x_5 - x_1$.

$$\max = 10 \quad x_5 - x_1 = 10, \text{ když } x_1 = 0, \text{ pak } x_5 = 10$$

$$\text{tudíž } X = \{0, x_2, x_3, x_4, 10\}.$$

Z ΔX odstraníme prvek $x_5 - x_1$ a dostáváme nové $\Delta X = \{2, 2, 3, 3, 4, 5, 6, 7, 8\}$.

- 2) $\max = 8$ máme možnosti: $x_4 - x_1 = 8, x_4 = 8$
 nebo $x_5 - x_2 = 8, x_2 = 2$
 jelikož jde o zrcadlové prvky (od první i poslední hodnoty X mají stejnou vzdálenost) můžeme vybrat jakoukoli z nich.
 Vezměme, že $x_2 = 2$.

$$\text{Nové } X = \{0, 2, x_3, x_4, 10\}.$$

Z ΔX odstraníme prvky o hodnotách: $x_2 - x_1 = 2, x_5 - x_2 = 8$.

Nové $\Delta X = \{2,3,3,4,5,6,7\}$.

- 3) $\max = 7$ máme možnosti: $x_4 - x_1 = 7, x_4 = 7$
 nebo $x_5 - x_3 = 7, x_3 = 3$
 avšak kdyby $x_3 = 3$, tak by $x_3 - x_2 = 1$, ale 1 není v ΔX , tudíž volíme $x_4 = 7$.

Nové $X = \{0,2,x_3,7,10\}$.

Z ΔX odstraníme prvky o hodnotách: $x_5 - x_4 = 3, x_4 - x_1 = 7$ a $x_4 - x_2 = 5$.

Nové $\Delta X = \{2,3,4,6\}$.

- 4) $\max = 6$ zbyly dvě možnosti $x_3 - x_1 = 6, x_3 = 6$
 nebo $x_5 - x_3 = 6, x_3 = 4$
 kdyby $x_3 = 6$ tak by $x_4 - x_3 = 1$, to ale není v ΔX , takže $x_3 = 4$.

Výsledek: $X = \{0,2,4,7,10\}$

Provedeme kontrolu správnosti řešení. Vypočítáme všechny možnosti vzdáleností mezi prvky a po seřídění musí dát zadaný vektor.

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

Příklady k řešení:

- 1) $\Delta X = \{1,2,2,3,4,4,5,6,7,8\}$
- 2) $\Delta X = \{2,2,2,4,4,4,6,6,8,10\}$
- 3) $\Delta X = \{1,2,2,2,2,3,4,4,4,5,6,6,7,8,9\}$

Úkol: V R implementujte rekurzivní algoritmus pro PDP (Partial Digest Problem) podle následujícího pseudokódu:

PartialDigestProblem(L)

- 1 $width \leftarrow \text{maximální prvek z } L$
- 2 $Delete(width, L)$
- 3 $X \leftarrow \{0, width\}$
- 4 $Place(L, X)$

Place(L,X)

```
1   if L je prázdné
2       output X
3       return
4    $y \leftarrow \text{maximální prvek z } L$ 
5   if  $\Delta(y,X) \supseteq L$ 
6       přidej y do X a odstraň délky  $\Delta(y,X)$  z L
7       Place(L,X)
8       odstraň y z X a přidej délky  $\Delta(y,X)$  do L
9   if  $\Delta(\text{width-}y,X) \supseteq L$ 
10      přidej width-y do X a odstraň délky  $\Delta(\text{width-}y,X)$  z L
11      Place(L,X)
12      odstraň width-y z X a přidej délky  $\Delta(\text{width-}y,X)$  do L
13  return
```

Upřesnění: $L = \Delta X$, $Delete(y,L)$ vymaže hodnotu y z L , $\Delta(y,X)$ je vektor délek mezi hodnotou y a všemi hodnotami X .

Nápověda: Vytvořte externí funkci *Remove()*, která bude z vektoru L odstraňovat použité délky Δ .

Brute-force Motif Search

Ačkoli některé problémy v biologických systémech lze řešit pomocí velmi jednoduchých vyhledávacích algoritmů, velký prohledávací prostor může způsobit exponenciální nárůst doby běhu s velikostí systému. V boji proti tomuto problému je obvykle možné využít porozumění omezením vyhledávacího prostoru k chytrému návrhu algoritmů, které dávají rozumnou dobu běhu v porovnání s velikostí biologických systémů.

V předchozí části cvičení jsme se zabývali dvěma algoritmy pro řešení částečného rozkladu problému (DDP & PDP). Metoda hrubé síly prohledává všechny možné množiny $(n - 2)$

restrikčních míst pro původní řetězec složený ze $\binom{n}{2}$ prvků, dokud se nenajde množina L . Toho se dosáhne pomocí funkcí *place* a *select* k vytvoření řetězců o délce $(n - 2)$. Doba běhu tohoto algoritmu je však $O(W(n-2))$, kde W je délka původního řetězce.

Tím, že si uvědomíme, že největší prvek v L bude délka původního řetězce, a že dalšími největšími prvky v množině budou vzdálenosti od omezení ke koncům původního řetězce, jsme vytvořili nový algoritmus nazvaný **Branch and Bound**. Tím se doba běhu zkrátila na $O(n^2)$.

Naopak hrubou silou (**Brute-force Motif Search**) lze jednoduše iterovat přes všechny $(L_1 - n) \times (L_2 - n) \times \dots \times (L_k - n)$ takových výchozích pozic $\{s_1, s_2, \dots, s_k\}$ a zachovat posloupnost, jejíž profilová matice dává výsledek nejnižší konsenzuální skóre. Protože potřebujeme vyhodnotit na pořadí matic L_k každá s $n \times k$ prvky, roste doba běhu této metody jako $T = O(n_k L_k)$, což je exponenciální v počtu sekvencí DNA, které chceme zkoumat. Proto tato metoda funguje, pokud se porovnává pouze několik sekvencí.

Postup Brute-force Motif Search

1. Funkce **Score**
2. Funkce **NextLeaf**
3. Funkce **BFMotifSearch**
4. Funkce **NextVertex**
5. Funkce **SimpleMotifSearch**

Úkol: V R implementujte Brute-force Motif Search algoritmus. Pseudokódy pro jednotlivé funkce tohoto algoritmu jsou dostupné v Knize 1 – kapitola 4.