СОФИЙСКИ УНИВЕРСИТЕТ "СВ. КЛИМЕНТ ОХРИДСКИ"
ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

Алгоритми на молекулярната биология

Л Е К Ц И И

към курса по "Молекулярна биология и алгоритми"

за студенти от магистърска програма

"Био-медицинска информатика"

ст. н. с. II ст. д-р Маргарита Камбурова
Българска Академия на Науките,
Институт по Микробиология, с-я Екстремофилни бактерии

ст. н. с. I ст. дмн Светослав Марков
Българска Академия на Науките,
Институт по Математика и Информатика, с-я Биоматематика

София, 2008

В този материал се обсъждат въпроси от конспекта, отнасящи се до някои основни алгоритми за сравняване на биологични секвенции.

13. Алгоритми

13.1. Примери за алгоритми. Алгоритъм на Евклид. Псевдоалгоритмичен език (псевдокод). Рекурсивни и итеративни алгоритми.

13.2. Алгоритъм на Хорнер.

13.3. Алгоритъм на Фибоначи.

13.4. Алгоритъм за ресто с минимален брой монети.

14. Алгоритми за подреждане на био-секвенции.

14.1. Сравняване на биологични секвенции. Редактиране на биосеквенции. Редакционни операции и редакционно разстояние.

14.2. Подреждане на две био-секвенции. Матрица на субституциите.

14.3. Алгоритъм за глобално подреждане на био-секвенции.

14.4. Примерни задачи.

14.5. Алгоритъм за локално подреждане на био-секвенции. Алгоритми за подреждане при свободни краища и дупки.

14.6. Многократно подреждане. Понятие за филогенетично дърво.

13.1. **Примери за алгоритми. Алгоритъм на Евклид. Псевдоалгоритмичен език (псевдокод). Рекурсивни и итеративни алгоритми.**

Псевдоалгоритмичен език, псевдокод: нещо напомнящо C, Fortran или Pascal

Алгоритъм на Евклид [1] за намиране на най-голямо общо кратно (делител) (НОК, gcd)

Дадени са две естествени числа a и b. Ако b е нула, то a е НОК. Ако не, повтори процеса използвайки съответно числата b и $a \bmod b$. С $a \bmod b$ означаваме остатъка от деленето на числата a и b, например: $15 \bmod 9 = 6$, $24 \bmod 7 = 3$, $7 \bmod 24 = 7$.

Като използваме рекурсия, алгоритъма се записва така:

function gcd(a, b)
    if $b = 0$ return a
    else return gcd(b, $a \bmod b$)

Като се използват итерации (което е по-ефективно при някои компилатори):

function gcd(a, b)
    while $b \neq 0$
        $t := b$
        $b := a \bmod b$
        $a := t$
    end-while
    return a

Традиционен алгоритъм. Както е даден от Евклид, алгоритъма се записва така:

function gcd(a, b)
    while $a \neq b$
        if $a > b$
            $a := a - b$
        else
            $b := b - a$
    return a

Пример. Да намерим gcd на 1071 и 1029, което е 21. Припомняме, че “mod” означава “остатъкът след делене”

С рекурсивния алгоритъм имаме:

gcd(1071, 1029)	Начални стойности на аргументите
= gcd(1029, 42)	Вторият аргумент е 1071 mod 1029
= gcd(42, 21)	Вторият аргумент е 1029 mod 42
= gcd(21, 0)	Вторият аргумент е 42 mod 21
= 21	Понеже b=0, резултатът е a

С итеративния алгоритъм:

1071 1029 Стъпка 1: Начални стойности на аргументите

1029 42 Стъпка 2: Остатъкът от деленето на 1071 на 1029 е 42, което поставяме вдясно, а делителят 1029 поставяме вляво

42 21 Стъпка 3: Повтаряме цикъла, разделяйки 1029 на 42, и получавайки 21 като остатък.

21 0 Стъпка 4: Повтаряме цикъла, като делим 42 на 21, получавайки 0 за остатък, и алгоритъма спира. Числото вляво, т. е. 21, е търсеното gcd.

13.2. **Алгоритъм на Хорнер** [2]–[4]

Даден е полиномът

$$P(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + ... + a_{n-1}x^{n-1} + a_nx^n$$

където $a_0, a_1, a_2, ..., a_n$ са реални числа. Искаме да пресметнем $P(x_0)$ и $P'(x_0)$ в дадена точка x_0.

За тази цел полагаме

$b_n := a_n$
$b_{n-1} := a_{n-1} + b_nx_0$
$b_{n-2} := a_{n-2} + b_{n-1}x_0$
.
.
.
$b_0 := a_0 + b_1x_0$

Тогава b_0 и b_1 са търсените стойности $P(x_0)$, т. е. $b_0 = P(x_0)$, $b_1 = P'(x_0)$.

За да се убедим в това да забележим, че полиномът може да се запише във вида:

$$P(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + ... + x(a_{n-1} + a_nx)...)))$$

Псевдокод за алгоритъма на Хорнер:

Стъпка 1 $p := a_n$ и $q := 0$
Стъпка 2 do 3 and 4 for $i = n - 1, n - 2, ..., 0$
Стъпка 3 $q := x * q + p$
Стъпка 4 $p := x * p + a_i$
Стъпка 5 Return p, q

Имаме $P(x_0) = p$ и $P'(x_0) = q$

Задача. Запишете алгоритъма на Хорнер с рекурсия.

13.3. Алгоритъм на Фибоначи. [5]

Задача. Да се опише развитието на популация зайци с родоначалници двойка млади зайци (мъжки и женски) при следните предположения:
— всяка двойка зайци ражда две зайчета всяка година;
— децата-зайци имат свои деца след като станат на две години;

— зайците не умират.

Решение. Означаваме с n годината, а с $F(n)$ броя на двойките зайци през n-тата година. Имаме:

$F(1) = 1$ – почва се с една двойка първата година;

$F(2) = 1$ – зайците са още малки на втората година и нямат още деца;

$F(3) = 2$ – на третата година има нова двойка зайчета;

$F(4) = 3$ – на четвъртата година има нова двойка;

$F(5) = 5$ – на петата година има двойка внучета-зайци, и т. н.

Общата формула е:

$$F(n) = F(n-1) + F(n-2)$$

:

наистина, всяка нова година броя на зайците $(F(n-1))$от предната година се увеличава с броят им от преди две години $F(n-2)$. Да се напише алгоритъм за пресмятане на $F(n)$.

13.4. Алгоритъм за ресто с минимален брой монети. [6]

14. Алгоритми за подреждане на био-секвенции.

14.1. Сравняване на биологични секвенции. Редактиране на биосеквенции. Редакционни операции и редакционно разстояние.

Редакционни операции:

— вмъкване (инсерция)

— изтриване (делеция)

— заместване (субституция)

Вмъкване и изтриване са обратни една на друга — индел операция. Наистина можем да считаме, че втората секвенция в подреждането:

$$\begin{array}{cccccccc} a & b & c & d & f & g & e \\ a\,' & b\,' & c\,' & -\,' & f\,' & g\,' & e\,' \end{array} \tag{1}$$

е получена от първата чрез изтриване на буквата d. Но също така можем да считаме, че първата секвенция в подреждането:

$$\begin{array}{cccccccc} a & b & c & e & d & f & g \\ a\,' & b\,' & c\,' & -\,' & d\,' & f\,' & d\,' \end{array} \tag{2}$$

е получено чрез вмъкване на буквата e.

14.2. Подреждане на две био-секвенции. Матрица на субституциите.

Биосеквенция (накратко: секвенция) е последователност от букви, принадлежащи на дадена азбука. На практика азбуката се състои или от четири букви (съответстващи на базите на нуклеиновете киселини) или от 20 букви (съответстващи на аминокиселините в белтъците.

Биосеквенциите могат да се "разтягат" като се вмкват индели между буквите. От биосеквенциите могат да се взимат подсеквенции – това са подмножества при които не се позволява разместване на буквите, а само изтриване на букви и вмъкване на индели.

Казваме, че една био-секвенция $S_1\ S_2\ \ldots\ S_n$ е разтеглена (разредена) до дължина $k \geq n$, ако в нея са добавени (вмъкнати) $n-k$ интервала пред, след или между буквите.

Дадени са две био-секвенции:

$$S_1\ S_2\ \ldots\ S_n, \qquad T_1\ T_2\ \ldots\ T_m, \qquad n, m \geq 0. \tag{3}$$

Нека редиците (3) са разтеглени (с помощта на индели) до една и съща дължина $k \geq n$, $k \geq m$. Да разположим така разтегнатите редици една под друга в матрица от два реда и k стълба. Ако всички стълбове в матрицата имат една от следните три форми

$$\begin{matrix} S_i \\ T_j \end{matrix}\ , \qquad \begin{matrix} S_i \\ - \end{matrix}\ , \qquad \begin{matrix} - \\ T_j \end{matrix}\ , \tag{4}$$

казваме, че така получената матрица е едно подреждане (в двуреден строй) на биосеквенциите (3). Съгласно казаното в едно подреждане на редици не трябва да има стълбо от вида $\begin{pmatrix} - \\ - \end{pmatrix}$, но за удобство говорим и за такива стълбове (някои алгоритми изискват двуредната матрица да започва с такъв стълб. На всеки стълб $\begin{pmatrix} S_i \\ T_j \end{pmatrix}$ от подреждането се присвоява *тегло* $\sigma_{ij} = \sigma(S_i, T_j)$, $i = 1, ..., n,\ j = 1, ..., m$. Тегла (наречени още индел-тегла) се задават и на стълбовете от вида $\begin{pmatrix} S_i \\ - \end{pmatrix}$, $\begin{pmatrix} - \\ T_j \end{pmatrix}$, а теглото на стълб от вида $\begin{pmatrix} - \\ - \end{pmatrix}$ се счита за нула, $\sigma(-, -) = 0$. Теглата на всички възможни стълбове се задават с матрица на теглата, наречена

още матрица на субституциите. Тази матрица се изготвя от специалисти по молекулярна биология.

Дадена е матрица на субституциите $\sigma_{ij} = \sigma(S_i, T_j),\ i = 1, ..., n,\ j = 1, ..., m$, както и индел-теглата

$$\sigma(S_i, -) = \sigma(-, T_j) = -d < 0,\ \ \sigma(-, -) = 0.$$

Под *тегло на едно подреждане* разбираме сумата от теглата на всички стълбове в подреждането.

Забележка. Съгласно горната дефиниция, теглото на дупка (няколко последователни индела) с дължина l е сума от теглата на инделите в дупката (т. е. $-ld$). За някои цели тази уговорка може да се промени.

Така на всяко подреждане на редиците (4) съответства някакво тегло. Подреждане, на което съответства най-голямото възможно тегло, наричаме *оптимално*. Възможно е да има две или повече оптимални подреждания.

Въпрос. Кой е възможно най-дългият строй и колко е теглото му?

14.3. Алгоритъм за глобално подреждане на био-секвенции

Задача. Да се намери оптимално подреждане (оптимален строй) на редиците $S_1\ S_2 \ldots S_n$ и $T_1\ T_2 \ldots T_m$.

Решение. (Алгоритъм на Needleman-Wunsch, 1970; модифицирана версия на Goth, 1982)

Означения. За $i \le n,\ j \le m$ означаваме с V^*_{ij} оптималния строй на подредиците

$$S_1\ S_2 \ldots S_i, \qquad T_1\ T_2 \ldots T_j, \tag{5}$$

а теглото на този оптимален строй — с V_{ij}. Ако можем да определим V_{ij} за всеки $i, j,\ 1 \le i \le n,\ 1 \le j \le m$, то V_{nm} е теглото, което търсим, а V^*_{nm} е търсеният оптимален строй.

Означаваме за удобство с V^*_{00} празното подреждане (празен строй) — можем да си го мислим като стълб от вида $(-, -)$, а с V_{00} — теглото на празния строй, което приемаме за равно на 0. Да отбележим, че подрежданията:

$$V^*_{i0} = \{ \begin{matrix} S_1 & S_2 & . & . & . & S_i \\ - & - & . & . & . & - \end{matrix} \},\quad V^*_{0j} = \{ \begin{matrix} - & - & . & . & . & - \\ T_1 & T_2 & . & . & . & T_j \end{matrix} \},$$

имат тегла съответно $V_{i0} = -id$ и $V_{0j} = -jd$ (поради уговорката за теглата на дупките). В частност теглото на стълба $V^*_{10} = (S_1, -)$ е $V_{10} = -d$, толкова е и теглото на стълба $V^*_{01} = (-, T_1)$, т. е. $V_{01} = -d$.

Да намерим подреждане V^*_{11} с оптимално тегло V_{11}. Възможните подреждания на две биосеквенции (5) с по една буква ($i = j = 1$) са:

$$\begin{matrix} S_1 \\ T_1 \end{matrix}\ , \quad \begin{matrix} - & S_1 \\ T_1 & - \end{matrix}\ , \quad \begin{matrix} S_1 & - \\ - & T_1 \end{matrix}\ , \tag{6}$$

Да сравним теглата на горните три възможни строя на секвенциите S_1 и T_1. Търсеният строй V^*_{11} е този измежду строевете (7), който има най-голямо тегло V_{11}. (Теглото на първия стой вземаме от таблицата на субституциите и го сравняваме с числото -2d, което е теглото на втория и третия строй).

Забелязваме, че вторият строй в (7) се получава от строя V^*_{01} с добавяне на стълб $(S_1, -)$, а третият — от V^*_{10} с добавяне на стълб $(-, T_1)$. За униформеност приемаме, че първият строй в (7) се получава от празния строй V^*_{00} чрез добавяне на стълба (S_1, T_1), т. е. мислим си (7) във вида

$$\begin{matrix} - & S_1 \\ - & T_1 \end{matrix}\ , \quad \begin{matrix} - & S_1 \\ T_1 & - \end{matrix}\ , \quad \begin{matrix} S_1 & - \\ - & T_1 \end{matrix}\ , \tag{7}$$

Следователно теглата на строевете (7) могат да се запишат съответно така: $V_{00}+\sigma(S_1, T_1)$, $V_{01}+\sigma(S_1, -) = V_{01}+(-d)$, $V_{10}+\sigma(-, T_1) = V_{10}+(-d)$. Ще определим търсеното тегло V_{11} като намерим кой от трите строя има максимално тегло, т. е.:

$$V_{11} = \max\{V_{00} + \sigma(S_1, T_1),\ V_{01} - d,\ V_{10} - d\}. \tag{8}$$

Като използуваме, че $V_{00} = 0$, $V_{01} = V_{10} = -d$, получаваме:

$$V_{11} = \max\{\sigma(S_1, T_1),\ -d - d,\ -d - d\} = \max\{\sigma(S_1, T_1),\ -2d\}.$$

Конкретната стойност на V_{11} намираме като определим стойностите на $\sigma(S_1, T_1)$ и d от матрицата на субституциите и вземем по-голямата от стойностите $\sigma(S_1, T_1)$ и $-2d$. Строят с така намереното максимално тегло V_{11} означаваме с V^*_{11}. Ако този строй не е единствен, което може да се случи при $\sigma(S_1, T_1) \leq -2d$, то за V^*_{11} вземаме един от възможните строеве с максимално тегло.

Аналогично V_{12}^* е един от възможните строеве на подредиците (5) при $i = 1, j = 2$ (т. е. на подредиците S_1 и T_1T_2). Очевидно това са следните пет строя

$$\begin{array}{cccccccccc} - & S_1 & , & - & - & S_1 & , & S_1 & - & , & - & S_1 & - & , & S_1 & - & - \\ T_1 & T_2 & , & T_2 & T_1 & - & , & T_1 & T_2 & , & T_1 & - & T_2 & , & - & T_1 & T_2 \end{array} . \tag{9}$$

Забелязваме, че всеки един от горните пет строя (9) се получава от някой предходен строй. Така първият строй се получава от V_{01}^* чрез добавяне на стълба (S_1, T_2), вторият строй — от V_{02}^* чрез добавяне на стълба $(S_1, -)$, а последните три строя се получават от строевете (7), участващи в определянето на V_{11}^*, с добавяне на стълба $(-, T_2)$.

Да определим кой от строевете (9) има максимално тегло. Теглото на първия строй е $V_{01} + \sigma(S_1, T_2)$, а вторият има тегло $V_{02} - d$. Що се отнася до останалите три строя, то, както вече споменахме, всички те се получават от строевете (7) с добавяне на стълба $(-, T_2)$. Следователно, от последните три строя максимално тегло има строят, получен от V_{11}^* с добавяне на стълба $(-, T_2)$ и неговото тегло е $V_{11} - d$. Така получихме

$$V_{12} = \max\{V_{01} + \sigma(S_1, T_2),\ V_{02} - d,\ V_{11} - d\}. \tag{10}$$

По същия начин получаваме $V_{13}, V_{14}, ..., V_{1m}, V_{21}, V_{22}, ...$ докато стигнем до търсената стойност V_{nm}. Например,

$$V_{13} = \max\{V_{02} + \sigma(S_1, T_3),\ V_{03} - d,\ V_{12} - d\}, \tag{11}$$

$$V_{21} = \max\{V_{10} + \sigma(S_2, T_1),\ V_{11} - d,\ V_{20} - d\}, \tag{12}$$

$$V_{22} = \max\{V_{11} + \sigma(S_2, T_2),\ V_{12} - d,\ V_{21} - d\}. \tag{13}$$

От формули (8), (10) и (11) се вижда, че общата рекурсивна (рекурентна) формула може да се запише така:

$$V_{ij} = \max\{V_{i-1,j-1} + \sigma(S_i, T_j),\ V_{i-1,j} - d,\ V_{i,j-1} - d\}. \tag{14}$$

Стойността V_{nm} е търсеното оптимално тегло. Пресметнатите тегла V_{ij} разполагаме в таблица с n реда и m стълба. В таблицата е удобно да отбележим със стрелка от коя от трите стойности $V_{i-1,j-1}$, $V_{i-1,j}$, или $V_{i,j-1}$ (от кое квадратче) е получена новата стойност V_{ij}. По този начин можем лесно да възстановим оптималния строй след като попълним таблицата (ако този строй не е единствен, можем да определим всички строеве с оптимално тегло). Този процес наричаме обратно проследяване (backtracing).

Сложността на алгоритъма е $O(mn)$, необходимата памет е $O(mn)$.

14.4. Алгоритъм за локално подреждане на био-секвенции. Алгоритми за подреждане при свободни краища и дупки.

Обща рекурсивна формула:

$$V_{ij} = \max\{0, V_{i-1,j-1} + \sigma(S_i, T_j),\ V_{i-1,j} - d,\ V_{i,j-1} - d\}. \tag{15}$$

Условие:
$\sigma(x, y) > 0$, ако x, y съвпадат,
$\sigma(x, y) \leq 0$ ако x, y не съвпадат (или едната буква е интервал)

14.5. Многократно подреждане. Понятие за филогенетично дърво.

$$\begin{aligned} V_{ijk} = \quad & \max\{V_{i-1,j-1,k-1} + \sigma(S_i, T_j, R_k),\ V_{i-1,j-1,k} - d,\ V_{i-1,j,k-1} - d, \\ & V_{i,j-1,k-1} - d,\ V_{i-1,j,k} - d,\ V_{i,j-1,k} - d,\ V_{i,j,k-1} - d.\}. \end{aligned}$$

Апроксимационни алгоритми
— централна звезда
— консенсусна секвенция

Литература

[1] http://en.wikipedia.org/wiki/Euclidean_algorithm

[2] http://en.wikipedia.org/wiki/Horner_scheme

[3] http://www.physics.utah.edu/_detar/lessons/c++/array/node4.html

[4] http://www.physics.utah.edu/_detar/phycs6720/handouts/horner.txt

[5] http://www.ics.uci.edu/ eppstein/161/960109.html

[6] Neil C. Jones, Pavel A. Pevzner, An Introduction to Bioinformatics Algorithms, A Bradford Book, The MIT Press, 2004.

[7] J. C. Setubal and J. Meidanis. Introduction to Computational Molecular Biology, Boston: PWS Publishing Company, 296pp., 1997

[8] R. Durbin et al. Biological sequence analysis. Probabilistic models of proteins and nucleic acids. CAmbridge University Press, 1998.

[9] List of textbooks in bioinformatics:
http://kdpnw.kdp-baptist.louisville.edu/proteomelab/bookbio.html
http://www.sacs.ucsf.edu/Resources/books.html

Приложение:

Англо-български терминологичен речник

character — буква
space, blank — интервал (брои се за буква), празно място
alphabet — азбука (от букви)
(bio-)sequence — (био-)секвенция, редица, дума
subsequence — подсеквенция, подредица, част от дума
alignment — подреждане (в две или повече редици),
(двуреден, k-реден) строй, престрояване, алиниране
подравняване (по двойки, по k-торки)
alignment row — (построен) ред, един от редовете на строя
(съдържащ вмъкнати интервали)
alignment column — (построен) стълб, всяка двойка или k-торка
букви (или букви и интервали) от строя
разположени виртикално (една под друга)
alignment lenght — дължина на подреждането,
брой на стълбовете в подреждането,
дължината на подредените редици
gap — дупка, два и повече последователни интервала
в един построен ред
insertion — инсерция, вмъкване
deletion — делеция, изтриване
substitution — субституция, заместване
indel — индел, инсерция или делеция
score, weight — точки, тегло
scoring — точкуване, изчисляване на тегло
substitution matrix — матрица на субституциите
local modifications — локални модификации
edit operations — редакционни операции
edit distance — редакционно разстояние
backtracing — обратно проследяване
overlapping sequences — припокриващи се редици
prefix — префикс, представка, начална част от дума
suffix — суфикс, наставка, крайна част от дума