دانشگاه صنعتی امیرکبیر
(پلیتکنیک تهران)
دانشکده مهندسی کامپیوتر

درس بیوانفورماتیک
تمرین سوم۳

امیرمهدی زرین‌نژاد
۹۷۳۱۰۸۷

نیم‌سال دوم ۱۴۰۰

سوال ۱) سه روش BLAST، FASTA و داینامیک پروگرمینگ را برای جستجو یک توالی در پایگاه داده با هم مقایسه کنید.

برای جستوجوی توالی در پایگاه‌داده ها روش‌های exhaustive داریم و یکسری روش‌های heuristic. در روش‌های exhaustive مانند Dynamic programming(مانند Smith-Waterman) تمامی راه‌حل‌های ممکن را بررسی می‌کنیم. و تضمین می‌شود که بهترین پاسخ(بهترین پاسخ لزوما پاسخ درست نیست) را پیدا کنیم. اما این روش‌ها زمان زیادی نیاز دارند. پس در مسائل بزرگ که محاسبات زیادی دارند مناسب نخواهند بود. اما برای مسائل کوچک می‌توان از آن‌ها استفاده کرد.

پس برای بهبود عملکرد از نظر زمان در مسائل بزرگ، از heuristicهایی استفاده می‌کنند. با استفاده از هیوریستیک‌ها دیگر همه حالت‌ها و احتمالات ممکن بررسی نمی‌شوند بلکه بر یک اساسی، تنها آن‌هایی که محتمل‌تر به نظر می‌رسند بررسی می‌شوند. پس تضمینی بر پیدا کردن بهترین پاسخ وجود ندارد اما می‌تواند بسیار سریع‌تر از روش‌های exhaustive عمل کند با sensitivity و specificity مناسب. (ممکن است کمی صحت نتایج برای افزایش سرعت پایین بیاید و tradeoff بین کارایی و سرعت دارند) (از نظر فضایی هم ممکن است هزینه‌برتر باشند)

در این زمینه الگوریتم‌هایی مانند BLAST و FASTA وجود دارند.

در الگوریتم FASTA به صورت کلمه کلمه با طول (کلمه‌های با طول k) بررسی می‌کنیم. که کندتر از BLAST است اما حساسیت بیش‌تری دارد(در k های کوچک‌تر) پس برای جست‌وجو در رشته‌های کوئری کوتاه مناسب‌تر است.

از طرفی خانواده BLAST برای انواع خاصی از کوئری‌ها بهینه شده‌اند(مانند جستوجو برای پیدا کردن رشته‌های مرتبط اما با فاصله زیاد از یکدیگر) که در برابر FASTA سریع‌تر هستند و accuracy زیادی از دست نمی‌دهند. (در BLAST سرعت بیش‌تر است زیرا کلمات کم‌تر اما مهم‌تر را فیلتر و جستوجو می‌کنیم.)

هم‌چنین روش FASTA فقط یک رشته برمی‌گرداند اما در روش‌های دیگر می‌توان هرتعداد رشته که لازم است برگرداند.

در الگوریتم BLAST، در فاز اولیه exact match انجام نمی‌دهیم و راه حلی که استفاده می‌کنیم و سرعت خوبی هم دارد اینست که ابتدا کلماتی که برای مچ شدن با یک کلمه امتیازشان از یه حدی بیشتر باشد را پیدا می‌کنیم و روی آن‌ها exact match انجام می‌دهیم. اما FASTA از هشینگ استفاده می‌کند و exact matchها را پیدا می کند.

در FASTA به‌جای word(در BLAST)، ktup داریم که طولش معمولا کوتاه تر از کلمات در BLAST است چراکه مستقیما exact match می‌خواهیم بگیریم.

پس به طور کلی سرعت BLAST بیشتر است و بعد FASTA و بعد DP. اما از نظر حافظه شاید بتوان گفت DP حافظه کمتری نیاز دارد و بعد FASTA و بعد BLAST(چراکه هردو dp را در خود دارند و BLAST علاوه برآن، درخت را نیز باید نگهداری کند.)

سوال ۲)

الف) با توجه به جدول زیر تمام کلماتی که با کلمه "QIV" با حداقل امتیاز ۱۰ همتراز میشوند را بیابید.

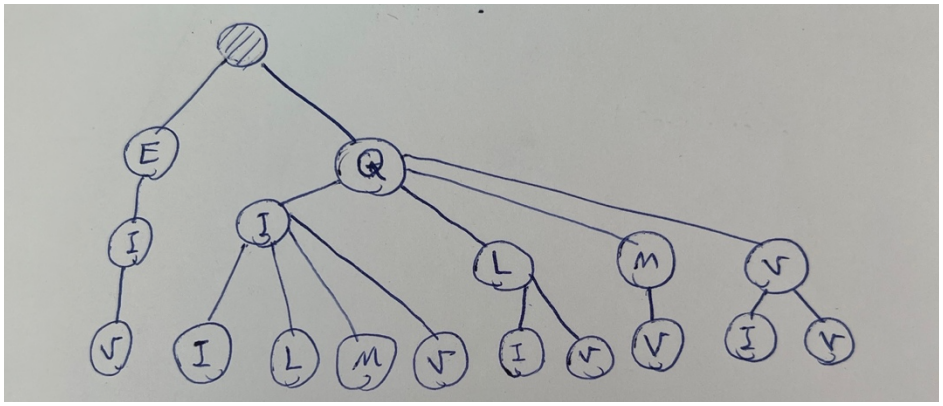
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	4																			
R	-1	5																		
N	-2	0	6																	
D	-2	-2	1	6																
C	0	-3	-3	-3	9															
Q	-1	1	0	0	-3	5														
E	-1	0	0	2	-4	2	5													
G	0	-2	0	-1	-3	-2	-2	6												
H	-2	0	1	-1	-3	0	0	-2	8											
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

QIV: QIV(5+4+4=13),
QII(5+4+3=12) ,
QIL(5+4+1=10) ,
QIM(5+4+1=10),

QVV(5+3+4=12),
QVI(5+3+3=11),
QLV(5+2+4=11),
QLI(5+2+3=10),
QMV(5+1+4=10),

EIV(2+4+4=10)

ب) با توجه به کلمات بدست آمده از قسمت قبل، درخت جستجو کلمات را رسم کنید.



سوال ۳) در هر مورد با استفاده از الگوریتم FASTA، بلندترین زیر رشته مشترک دنباله‌های پایگاه داده را با رشته ورودی (Query) پیدا کنید.

الف)

```
Query: CACGTTGACAT
  Database:
     • ATGACATTCGAA
     • CGATTCGGACA
  ktup = 1
```

Query indexes➔ 1:C, 2:A, 3:C, 4:G, 5:T, 6:T, 7:G, 8:A, 9:C, 10:A, 11:T

Query Hashtable:

A	C	G	T
2	1	4	5
8	3	7	6
10	9		11

Target table 1:

1	2	3	4	5	6	7	8	9	10	11	12
A	T	G	A	C	A	T	T	C	G	A	A
1	3	1	-2	-4	-4	-2	-3	-8	-6	-9	-10
7	4	4	4	-2	2	-1	-2	6	-3	-3	-4
9	9		6	4	4	4	3	0		-1	-2

بلندترین زیررشته مشترک: TGACAT که با شیفت به اندازه ۴ کاراکتر الاین می‌شود.

Target table 2:

1	2	3	4	5	6	7	8	9	10	11
C	G	A	T	T	C	G	G	A	C	A
0	2	-1	1	0	-5	-3	-4	-7	-9	-9
2	5	5	2	1	-3	0	-1	-1	-7	-3
8		7	7	6	3			1	-1	-1

بلندترین زیررشته مشترک: GACA که با شیفت به اندازه ۱ کاراکتر الاین می‌شود.

(بین این‌دو هم بخواهیم مقایسه کنیم، TGACAT بلندتر از GACA است)

ب)

```
Query: GTTACCACG
  Database:
     • TACGTCGT
  ktup = 2
```

Query indexes➔ 1:GT, 2:TT, 3:TA, 4:AC, 5:CC, 6:CA, 7:AC, 8:CG

Query Hashtable:

AC	CA	CC	CG	GT	TA	TT
4	6	5	8	1	3	2
7						

1	2	3	4	5	6	7
TA	AC	CG	GT	TC	CG	GT
2	2	5	-3		2	-6
	5					

بلندترین زیررشته‌های مشترک: TAC (با شیفت به اندازه ۲تا کاراکتر) و ACG (با شیفت به اندازه ۳تا کاراکتر)

سوال ۴) همترازی چندگانه دنباله‌های زیر را با استفاده از الگوریتم Star محاسبه کنید.

- A:ACGCTAAC
- B:TTGCACATC
- C:TCGGTAGATC
- D:TCACTGGAC

برای ماتریس امتیازدهی، از ماتریس DNAfullاستفاده کنید.
https://rosalind.info/glossary/dnafull/
و همچنین برای همترازی‌های دوتایی می‌توانید از لینک زیر استفاده کنید. دقت شود که در مرحله اول باید رشته‌های DNAرا انتخاب کرده و در مرحله دوم ماتریس امتیازدهی DNAfull را انتخاب کنید .
https://www.ebi.ac.uk/Tools/psa/emboss_needle/

ابتدا رشته‌ها را دو به دو الاین می‌کنیم و امتیازاتشان را حساب می‌کنیم:

A, B:

```
EMBOSS_001         1 ----ACGCTAAC      8
                       ||   |.|
EMBOSS_001         1 TTGCAC---ATC      9
```

Score = 5

A, C:

```
EMBOSS_001         1 ACGCTAAC--        8
                     .||.||..
EMBOSS_001         1 TCGGTAGATC       10
```

Score = 4

A, D:

```
EMBOSS_001         1 ACGCTAAC-----     8
                         |.||
EMBOSS_001         1 ----TCACTGGAC     9
```

Score = 11

B, C:

```
EMBOSS_001         1 TT--GCACATC       9
                     |  |.|.|||
EMBOSS_001         1 -TCGGTAGATC      10
```

Score = 11.5

B, D:

```
EMBOSS_001         1 ---TTGCACATC      9
                         .||.||
EMBOSS_001         1 TCACTGGAC---      9
```

Score = 12

C, D:

```
EMBOSS_001         1 TCGGTAGATC-------     10
                     ||
EMBOSS_001         1 --------TCACTGGAC      9
```

Score = 10

حال ماتریس امتیازات را تشکیل می‌دهیم:

	A	B	C	D	
A	-	5	4	11	20
B	5	-	11.5	12	28.5
C	4	11.5	-	10	25.5
D	11	12	10	-	33

باتوجه به اینکه رشته D بیشترین امتیاز را در الاینمنت با دیگر رشته‌ها کسب کرده، به عنوان رشته مرکزی انتخاب می‌شود و مرکز همترازی خواهد بود. سپس رشته‌ها را یک به یک با مرکز الاین می‌کنیم و قبلی‌ها را درهر مرحله بروز می‌کنیم. (قانون once a gap, always a gap را هم اعمال می‌کنیم)
(می‌توانیم به ترتیب نزولی امتیازات کسب شده با رشته مرکزی الاینمنت را انجام دهیم اما حاصل فرقی نمی‌کند.)

برحسب امتیاز: ابتدا D, B سپس با A و نهایتا با C:

D,B➔

```
D: TCACTGGAC---
B: ---TTGCACATC
```

D,B,A➔

```
D: ----TCACTGGAC---
B: -------TTGCACATC
A: ACGCTAAC--------
```

D,B,A,C➔

```
D: ------------TCACTGGAC---
B: ---------------TTGCACATC
A: --------ACGCTAAC--------
C: ----TCGGTAGATC----------
```

از آنجایی که در MSA ستون شامل فقط gap نباید داشته باشیم، ۴ ستون اول را کنار می‌گذاریم. درحقیقت زمان هم‌تراز کردن رشته‌ی C (مرحله آخر) که ۸ گپ به ابتدای D اضافه می‌کند، ۴ گپ از قبل داریم و ۴تا دیگر اضافه می‌کنیم و نه ۸تا

درنتیجه همترازی صحیح برای مرحله آخر(اضافه کردن C):

```
D: --------TCACTGGAC---
B: ----------TTGCACATC
A: ----ACGCTAAC--------
C: TCGGTAGATC----------
```

سوال ۵) روش و عملکرد سه الگوریتم ClustalW، Star و T-coffee را با یکدیگر مقایسه کنید و مزایا و معایب و محدودیت‌های هر کدام را ذکر کنید.

ClustalW:

در پیاده‌سازی این الگوریتم ابتدا بین رشته‌ها دو به دو alignment انجام می‌دهیم ویک ماتریس فاصله تشکیل می‌دهیم. سپس با توجه به این ماتریس و ترتیب امتیازات جفت رشته‌ها به روش neighbor joining، یک guide tree ایجاد می‌کنیم که ترتیب الاین کردن رشته‌ها را در خود دارد. سپس با توجه به این guide tree ایجاد شده، از برگ‌ها به سمت ریشه رشته‌ها را pairwise با یکدیگر تراز می‌کنیم. (خروجی همترازی‌ها مرحله اول را دوباره با هم الاین می‌کنیم و الی آخر)

الگوریتم Clustal الگوریتمی براساس Global alignment است. نتیجتا برای مقایسه‌ی رشته‌هایی با طول خیلی متفاوت مناسب نیست. در این الگوریتم نتیجه‌ی نهایی همترازی به ترتیب همترازی رشته‌ها بایکدیگر وابسته است. و در انتخاب pairwise alignmentهای اولیه خاصیتی حریصانه دارد. در این الگوریتم اگر خطایی در مراحل اولیه رخ دهد قابل تصحیح نیست. در پیاده‌سازی این الگوریتم می‌توان برحسب شرایط و میزان شباهت از ماتریس‌های جانشینی مختلف استفاده کرد که یک مزیت به حساب می‌آید. هم‌چنین گپ پنالتی را می‌توان برایش تنظیم کرد تا کنترل کنیم insertion و deletion های کمتری در نواحی محافظت‌شده درنظر گرفته شود و خارج از آن اجازه رخ دادنش بیش‌تر باشد.

Star:

در الگوریتم star نیز ابتدا بین تمام رشته‌ها pairwise alignment انجام می‌دهیم و امتیازشان را در یک ماتریس امتیازدهی قرارمی‌دهیم. سپس sequence یک رشته مرکزی درنظر می‌گیریم که رشته‌ای است که جمع امتیازات pairwise alignmentاش با دیگر رشته‌ها بیشتر از بقیه باشد و مجموعا امتیاز بیش‌تری کسب کرده باشد. سپس رشته‌های دیگر را (به ترتیب نزولی امتیاز هم‌ترازی‌شان با مرکز) وارد می‌کنیم و با مرکز هم‌تراز می‌کنیم. در طی هم‌ترازی‌ها هم به قانون once a gap, always a gap عمل می‌کنیم و هرگاه گپ جدیدی به center اضافه شد، آن را به رشته‌های الاین شده قبلی و رشته‌های بعدی که وارد می‌شوند اضافه می‌کنیم.

الگوریتم Star هیوریستیک سریعی برای محاسبه‌ی همترازی چندتایی دارد و اگر معیار امتیازدهی ویژگی مثلث حمار را داشته باشد؛ با تخمین خوبی همترازی چندگانه optimal را خروجی خواهد داد.

T-Coffee:

الگوریتم T-Coffee یکی از الگوریتم‌هایی است که برای ارتقای الگوریتم‌های قبلی ارائه شدند. مانند clustal به صورت progressive رشته‌ها را الاین می‌کند و تفاوت اصلی‌اش این است که هردو همترازی دوتایی(pairwise) محلی و سراسری را برای همه‌ی جفت رشته‌های ممکن اعمال می‌کند نتیجتا می‌تواند رشته‌های با طول متفاوت را هم همتراز کند. پس این الگوریتم در الاین کردن رشته‌هایی نسبتا متفاوت از clustal بهتر عمل می‌کند اما در برابر clustal سرعت کم‌تری دارد.