Sekvenciranje

- Današnja tehnologija ne omogućuje precizno „čitanje” cijelog genoma odjednom
- Veličina genoma:
  - Virus ~ 50.000 baza
  - Bakterija ~ 5.000.000 baza
  - Kvasac ~ 10.000.000 baza
  - Ptice ~ 1.000.000.000 baza
  - Čovjek ~ 3.000.000.000 baza
    - Jedan kromosom ~ 50.000.000 – 250.000.000 baza
  - Neke biljke čak preko 100.000.000.000 baza

Sekvenciranje - tehnologije

- Prva generacija
  - Sangerovo sekvenciranje
  - Očitanja srednje duljine, precizno
- Druga generacija (NGS – *next generation sequencing*)
  - Illumina, Roche, Ion Torrent …
  - **Očitanja male duljine, tipično 100-200 baza, precizno (~ 1-3%)**
- Treća generacija
  - PacBio, ONT – Oxford Nanopore Technologies
  - **Očitanja velike duljine (deseci tisuća baza) s velikom greškom (~ 5-30%), nova precizna duga očitanja (PacBio Hi-Fi)**

Dinamičko programiranje

- Poravnanje očitanja sa referencom
- Primjer:
  - Escherichia coli – 4,65 Mbp
  - Očitanje – 100 kbp
  - Jedna ćelija u tablici dinamičkog programiranja: 5 byte-ova
  - 2,3 TB memorije
- Preklapanje dva očitanja
- Primjer:
  - Dva očitanja po 50 kbp
  - Jedna ćelija u tablici dinamičkog programiranja: 5 byte-ova
  - 12,5 GB memorije – potencijalno također previše
  - Treba izračunati 2,5 milijardi ćelija – potencijalno presporo

Bioinformatika

2020/2021
Minimizeri

Minimizeri

- Umjesto cijele sekvence, želimo uspoređivati samo dijelove
  - „seed and extend” – uspoređuju se kraće sekvence duljine k – k-mer
  - sljedeće predavanje – BLAST
  - kako odabrati koje k-mere uspoređivati?

- Svakih *w* k-mera odaberemo jedan
  - Osjetljivo na umetanja i brisanja

- Svakih *w* k-mera odaberemo jedan, koji je abecedno minimalan
  - Upravo to je minimizer!

Minimizeri

- k – veličina minimizera (podniza koji pamtimo)
- w – broj uzastopnih k-mera koje gledamo
- w uzastopnih k-mera pokrivaju niz od l = w+k-1 znakova – prozor (engl. window)

- Primjer:

Pozicija	1	2	3	4	5	6	7	1	2	3	4	5	6	7	8	9	10	11	12
Niz	2	3	1	0	3	4	3	4	2	6	4	7	2	8	1	4	7	5	1
k-meri, minimizer je označen **masnim** slovima	2	3	1					4	2	6	4	7	2	8					
		3	1	0					**2**	**6**	**4**	**7**	**2**	**8**	**1**				
			1	0	3					6	4	7	2	8	1	4			
				0	**3**	**4**					4	7	2	8	1	4	7		
					3	4	3					7	2	8	1	4	7	5	
													2	8	1	4	7	5	1
	k = 3, w = 5, l = 7							k = 7, w = 6, l = 12											

Preuzeto iz: Roberts at. al. Reducing storage requirements for biological sequence comparison, Bioinformatics 2004.

Minimizeri

- Ako dvije sekvence imaju zajednički podniz duljine w+k-1, imaju i zajednički minimizer
- Ako imamo skup sekvenci za koje želimo odrediti minimizere, opisujemo ih trojkom (s, i, p)
  - s – sekvenca minimizera
  - i – indeks sekvence u kojoj je minimizer pronađen
  - p – pozicija u sekvenci na kojoj se minimizer nalazi
- Kažemo je trojka (s, i, p) (w, k)-minimizer na niz T_i, ako je minimizer za barem jedan prozor u nizu T_i
- Kako tražimo minimizere na cijelom nizu?
  - Odaberemo w i k
  - Gledamo prozor od l = w+k-1 znakova, koji pomičemo po jedna znak
  - Računamo minimizer za prozor
  - Ako je to novi minimizer za taj niz, dodajemo ga u listu minimizera

Minimizeri

- Primjer:
  - Tražimo sve (4,3)-minimizere za zadani niz

Pozicija	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Niz	2	3	1	0	3	2	1	0	1	2	3	3	1	0	1
Prozor u kojem je minimizer označen **masnim** slovima	2	3	1	**0**	**3**	**2**									
		3	1	**0**	**3**	**2**	1								
			1	**0**	**3**	**2**	1	0							
				0	**3**	**2**	1	0	1						
					3	2	1	**0**	**1**	**2**					
						2	1	**0**	**1**	**2**	3				
							1	**0**	**1**	**2**	3	3			
								0	**1**	**2**	3	3	1		
									1	**2**	**3**	3	1	0	
										2	3	3	**1**	**0**	**1**

Preuzeto iz: Roberts at. al. Reducing storage requirements for biological sequence comparison, Bioinformatics 2004.

Minimizeri

- Primjer:
- Tražimo sve (4,3)-minimizere za zadani niz
- Vidimo da svi znakovi početnog niza nisu sadržani u minimizerima (pozicije 1,2,3,7 i 12)
- Iako to nije uvijek nužno, može biti važno za neke primjene

- Ako postavimo da je $w \leq k$, neće biti rupa među minimizerima

Pozicija	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Niz	2	3	1	0	3	2	1	0	1	2	3	3	1	0	1
Prozor u kojem je minimizer označen **masnim** slovima	2	3	1	**0**	**3**	**2**									
		3	1	**0**	**3**	**2**	1								
			1	**0**	**3**	**2**	1	0							
				0	**3**	**2**	1	0	1						
					3	2	1	**0**	**1**	**2**					
						2	1	**0**	**1**	**2**	3				
							1	**0**	**1**	**2**	3	3			
								0	**1**	**2**	3	3	1		
									1	**2**	**3**	3	1	0	
										2	3	3	**1**	**0**	**1**

Preuzeto iz: Roberts at. al. Reducing storage requirements for biological sequence comparison, Bioinformatics 2004.

Minimizeri

- Primjer:
  - Tražimo sve (**3,3**)-minimizere za zadani niz

Pozicija	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Niz	2	3	1	0	3	2	1	0	1	2	3	3	1	0	1
Prozor u kojem je minimizer označen **masnim** slovima	2	3	**1**	**0**	**3**										
		3	1	**0**	**3**	**2**									
			1	**0**	**3**	**2**	1								
				0	**3**	**2**	1	0							
					3	2	**1**	**0**	**1**						
						2	1	**0**	**1**	**2**					
							1	**0**	**1**	**2**	3				
								0	**1**	**2**	3	3			
									1	**2**	**3**	3	1		
										2	**3**	**3**	1	0	
											3	3	**1**	**0**	**1**

Preuzeto iz: Roberts at. al. Reducing storage requirements for biological sequence comparison, Bioinformatics 2004.

Minimizeri

- Primjer:
- Tražimo sve (**3,3**)-minimizere za zadani niz
- Više nema rupa između minimizera, ali i dalje imamo rupe na početku (mogu biti i na kraju)
- Računamo krajnje minimizere na početku i kraju niza

Pozicija	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Niz	2	3	1	0	3	2	1	0	1	2	3	3	1	0	1
Prozor u kojem je minimizer označen **masnim** slovima	2	3	**1**	**0**	**3**										
		3	1	**0**	**3**	**2**									
			1	**0**	**3**	**2**	1								
				0	**3**	**2**	1	0							
					3	2	**1**	**0**	**1**						
						2	1	**0**	**1**	**2**					
							1	**0**	**1**	**2**	3				
								0	**1**	**2**	3	3			
									1	**2**	**3**	3	1		
										2	**3**	**3**	1	0	
											3	3	**1**	**0**	**1**

Preuzeto iz: Roberts at. al. Reducing storage requirements for biological sequence comparison, Bioinformatics 2004.

Minimizeri

- Primjer:
- Tražimo sve (**3,3**)-minimizere za zadani niz
- Više nema rupa između minimizera, ali i dalje imamo rupe na početku (mogu biti i na kraju)
- Računamo krajnje minimizere na početku i kraju niza
- Računamo (u,k)-minimizere za svaki u < w, na početku i na kraju niza
- w = 3, pa u poprima vrijednosti 1 i 2
- Prozor u ovom slučaju uvijek počinje na početku niza ili završava na kraju niza

Pozicija	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Niz	2	3	1	0	3	2	1	0	1	2	3	3	1	0	1
Prozor u kojem je minimizer označen **masnim** slovima	**2**	**3**	**1**												
	2	**3**	**1**	0											
												3	**1**	**0**	**1**
													1	**0**	**1**

Preuzeto iz: Roberts at. al. Reducing storage requirements for biological sequence comparison, Bioinformatics 2004.

Minimizeri

- Primjer: Tražimo sve (**3,3**)-minimizere za zadani niz

Pozicija	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Niz	2	3	1	0	3	2	1	0	1	2	3	3	1	0	1
	2	**3**	**1**												
	2	**3**	**1**	0											
	2	3	**1**	**0**	**3**										
		3	1	**0**	**3**	**2**									
			1	**0**	**3**	**2**	1								
				0	**3**	**2**	1	0							
					3	2	**1**	**0**	**1**						
						2	1	**0**	**1**	**2**					
							1	**0**	**1**	**2**	3				
								0	**1**	**2**	3	3			
									1	**2**	**3**	3	1		
										2	**3**	**3**	1	0	
											3	3	**1**	**0**	**1**
												3	**1**	**0**	**1**
													1	**0**	**1**

Prozor u kojem je minimizer označen **masnim** slovima

Preuzeto iz: Roberts at. al. Reducing storage requirements for biological sequence comparison, Bioinformatics 2004.

Minimizeri

- Primjer: Tražimo sve (**3,3**)-minimizere za zadani niz

Pozicija	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Niz	2	3	1	0	3	2	1	0	1	2	3	3	1	0	1
	2	**3**	**1**												
	2	**3**	**1**	0											
	2	3	**1**	**0**	**3**										
		3	1	**0**	**3**	**2**									
			1	**0**	**3**	**2**	1								
				0	**3**	**2**	1	0							
					3	2	**1**	**0**	**1**						
						2	1	**0**	**1**	**2**					
							1	**0**	**1**	**2**	3				
								0	**1**	**2**	3	3			
									1	**2**	**3**	3	1		
										2	**3**	**3**	1	0	
											3	3	**1**	**0**	**1**
												3	**1**	**0**	**1**
													1	**0**	**1**

Prozor u kojem je minimizer označen **masnim** slovima

Preuzeto iz: Roberts at. al. Reducing storage requirements for biological sequence comparison, Bioinformatics 2004.

Minimizeri

- Primjer: Tražimo sve (**3,3**)-minimizere za zadani niz

Pozicija	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Niz	2	3	1	0	3	2	1	0	1	2	3	3	1	0	1
	2	**3**	**1**												
	2	**3**	**1**	0											
	2	3	**1**	**0**	**3**										
		3	1	**0**	**3**	**2**									
			1	**0**	**3**	**2**	1								
				0	**3**	**2**	1	0							
					3	2	**1**	**0**	**1**						
						2	1	**0**	**1**	**2**					
							1	**0**	**1**	**2**	3				
								0	**1**	**2**	3	3			
									1	**2**	**3**	3	1		
										2	**3**	**3**	1	0	
											3	3	**1**	**0**	**1**
												3	**1**	**0**	**1**
													1	**0**	**1**

Krajnji minimizeri

Prozor u kojem je minimizer označen **masnim** slovima

Unutarnji minimizeri

Krajnji minimizeri

Preuzeto iz: Roberts at. al. Reducing storage requirements for biological sequence comparison, Bioinformatics 2004.

Minimizeri

- Primjer: Tražimo sve (**3,3**)-minimizere za zadani niz
- Popis minimizera (niz, pozicija)

- (231, 1)
- (103, 3)
- (032, 4)
- (101, 7)
- (012, 8)
- (123, 9)
- (233, 10)
- (101, 13)

Pozicija	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Niz	2	3	1	0	3	2	1	0	1	2	3	3	1	0	1
	2	**3**	**1**												
	2	**3**	**1**	0											
	2	3	**1**	**0**	**3**										
		3	1	**0**	**3**	**2**									
			1	**0**	**3**	**2**	1								
				0	**3**	**2**	1	0							
					3	2	**1**	**0**	**1**						
						2	1	**0**	**1**	**2**					
							1	**0**	**1**	**2**	3				
								0	**1**	**2**	3	3			
									1	**2**	**3**	3	1		
										2	**3**	**3**	1	0	
											3	3	**1**	**0**	**1**
												3	**1**	**0**	**1**
													1	**0**	**1**

Prozor u kojem je minimizer označen **masnim** slovima

Preuzeto iz: Roberts at. al. Reducing storage requirements for biological sequence comparison, Bioinformatics 2004.

Minimizeri

- Minimizeri se mogu primijeniti na usporedbu bilo kakvih nizova
- Primjena u bioinformatici:
  - Kako odabrati k i w?
    - Neki alati koriste k = 15, w = 5
  - Koje vrijednosti dodijeliti pojedinim nukleotidima i amino-kiselinama (može se raditi i s proteinima)?
    - U primjeru smo radili s brojevima, ali u bioinformatici radimo sa slovima
    - Možemo koristiti leksikografske vrijednosti
    - Možemo manje vrijednosti dodijeliti slovima koja se pojavljuju rjeđe – veća vjerojatnost da će rjeđi k-meri biti minimizeri
    - Poly-A tail – poseban problem kod RNA
  - Problem minimizera koji se često pojavljuju
    - Najčešće se izbacuju
  - DNA ima dva lanca
    - Svako očitanje uspoređujemo dva puta, jednom za glavni lanac, jednom za reverzni komplement
    - Računamo minimizere za oba slučaja
    - Kod mapiranja očitanja na referencu, reverzni komplement računamo ili za očitanja ili za referencu (nema potrebe za oboje)
- Kako iskoristiti minimizere za usporedbu nizova?

Minimizeri

- Kako iskoristiti minimizere za usporedbu nizova?

- 1. varijanta
  - Izračunati minimizere za prvi niz
  - Izračunati minimizere za drugi niz
  - *Longest common subsequence problem*
    - Nešto slično smo već računali za sličnost nizova dinamičkim programiranjem

- 2. varijanta
  - Izračunati minimizere za prvi niz
  - Izračunati minimizere za drugi niz
  - Usporediti dvije grupe minimizera i pronaći poklapanja (*engl. match*)
  - Sortirati poklapanja prema poziciji u prvom nizu
  - *Longest increasing subsequence problem* nad pozicijama u drugom nizu

Minimizeri

- Longest common subsequence

„The longest common subsequence (LCS) problem is the problem of finding the longest subsequence common to all sequences in a set of sequences (often just two sequences). It differs from the longest common substring problem: unlike substrings, subsequences are not required to occupy consecutive positions within the original sequences."

- U definiciji se razlikuje riječ **string** i **sequence**, tj. **substring** i **subsequence**
- Najdulji zajednički podniz, gdje se podniz dobije od početnog niza brisanjem proizvoljnog broja elemenata!

- Ovo je slično kao poravnanje i rješava se na sličan način, dinamičkim programiranjem
- Kao da računamo sličnost, bez kažnjavanja zamjena i praznina

Minimizeri

- Longest common subsequence
- Primjer: uzmimo nizove X = XMJYAUZ i Y = MZJAWXU
  - Možemo od oka vidjeti da je najduži zajednički podniz MJAU
- Računamo tablicu dinamičkog programiranja:

- Prvi redi i stupac postavimo na 0
- Računamo elemente:

Ako je X[i] = Y[j] onda

F(i,j) = F(i-1,j-1) + 1

Inače

F(i,j) = max(F(i-1,j), F(i, j-1))

		M	Z	J	A	W	X	U
	0	0	0	0	0	0	0	0
X	0	0	0	0	0	0	1	1
M	0	1						
J	0							
Y	0							
A	0							
U	0							
Z	0							

Preuzeto iz: https://www.techiedelight.com/longest-common-subsequence/

Minimizeri

- Longest common subsequence
- Primjer: uzmimo nizove X = XMJYAUZ i Y = MZJAWXU
  - Možemo od oka vidjeti da je najduži zajednički podniz MJAU
- Računamo tablicu dinamičkog programiranja:

- Računamo elemente i pamtimo na temelju koje ćelije smo ih izračunali

		M	Z	J	A	W	X	U
	0	0	0	0	0	0	0	0
X	0	0	0	0	0	0	1	1
M	0	1	1	1	1	1	1	1
J	0	1	1	2	2	2	2	2
Y	0	1	1	2	2	2	2	2
A	0	1	1	2	3	3	3	3
U	0	1	1	2	3	3	3	4
Z	0	1	2	2	3	3	3	4

Preuzeto iz: https://www.techiedelight.com/longest-common-subsequence/

Minimizeri

- Longest common subsequence
- Primjer: uzmimo nizove X = XMJYAUZ i Y = MZJAWXU
  - Možemo od oka vidjeti da je najduži zajednički podniz MJAU
- Računamo tablicu dinamičkog programiranja:

- Zadnja ćelija sadrži duljinu najduljeg zajedničkog podniza
- Povratkom unatrag možemo ga odrediti
- Ćelije kod kojih su znakovi u oba niza jednaki, odnosno one do kojih smo došli „ukoso”
- Najdulji zajednički podniz:
  MJAU

- Složenost: O(mn)

		M	Z	J	A	W	X	U
	0	0	0	0	0	0	0	0
X	0	0	0	0	0	0	1	1
M	0	1	1	1	1	1	1	1
J	0	1	1	2	2	2	2	2
Y	0	1	1	2	2	2	2	2
A	0	1	1	2	3	3	3	3
U	0	1	1	2	3	3	3	4
Z	0	1	2	2	3	3	3	4

Preuzeto iz: https://www.techiedelight.com/longest-common-subsequence/

Minimizeri

- Longest increasing subsequence

„Longest increasing subsequence problem is to find a subsequence of a given sequence in which the subsequence's elements are in sorted order, lowest to highest, and in which the subsequence is as long as possible. This subsequence is not necessarily contiguous, or unique.”

- Najdulji rastući podniz, gdje se podniz dobije od početnog niza brisanjem proizvoljnog broja elemenata!

Primjer:

za niz [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]

Neki od najduljih rastućih podnizova su (duljina = 6):

[0, 2, 6, 9, 11, 15]

[0, 4, 6, 9, 11, 15]

Preuzeto iz: https://www.geeksforgeeks.org/longest-increasing-subsequence-dp-3/

Minimizeri

- Longest increasing subsequence
- Za početak računamo samo duljinu najduljeg rastućeg podniza
- LIS(i) – duljina najduljeg rastućeg podniza za prvih i elemenata niza
- Možemo to napraviti rekurzivno
  - Neka je niz[0 … n-1] ulazni niz i neka je L(i) duljina najduljeg rastućeg podniza koji završava na poziciji i, takvog da je niz[i] zadnji element tog podniza
  - L(i) možemo izračunati na sljedeći način:

    L(i) = max(L(j)) + 1, za svaki j < i za koji vrijedi niz[i] > niz[j]

    L(i) = 1, ako ne postoji j < i, za koji vrijedi niz[i] > niz[j]

- Da bi izračunali duljinu najduljeg rastućeg niza za zadano polje, moramo naći maksimalnu vrijednost funkcije L(i) za sve i < n

Preuzeto iz: https://www.geeksforgeeks.org/longest-increasing-subsequence-dp-3/

Minimizeri

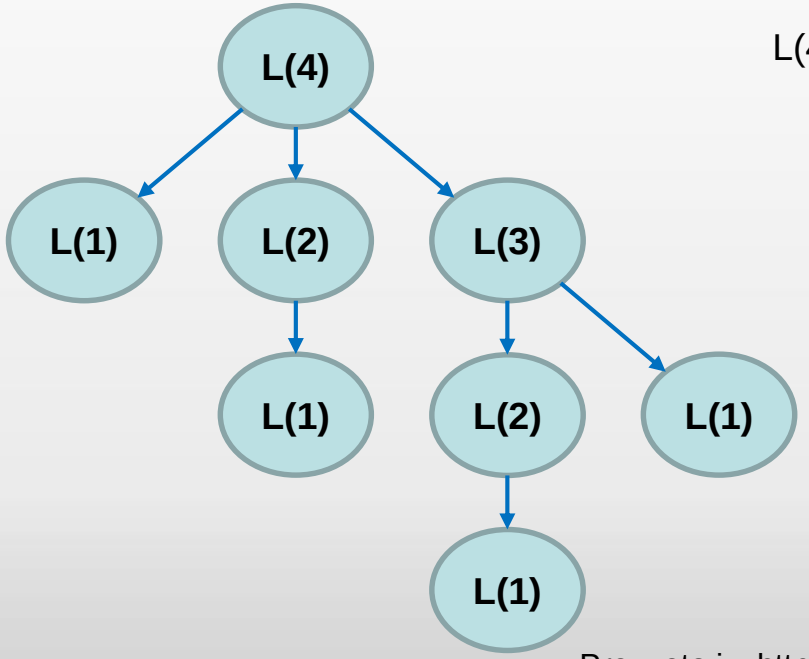
– L(i) možemo izračunati na sljedeći način:

L(i) = max(L(j)) + 1, za svaki j < i za koji vrijedi niz[i] > niz[j]

L(i) = 1, ako ne postoji j < i, za koji vrijedi niz[i] > niz[j]

- Primjer:

niz = [3, 10, 2, 11]



L(4) = 1 + max(L(3), L(2), L(1)), niz[4] > niz[3] = **3**
niz[4] > niz[2]
niz[4] > niz[1]

L(3) = 1, niz[3] < niz[2] = **1**
niz[3] < niz[1]

L(2) = 1 + max(L(1)), niz[2] > niz[1] = **2**

L(1) = 1 = **1**

Preuzeto iz: https://www.geeksforgeeks.org/longest-increasing-subsequence-dp-3/

Minimizeri

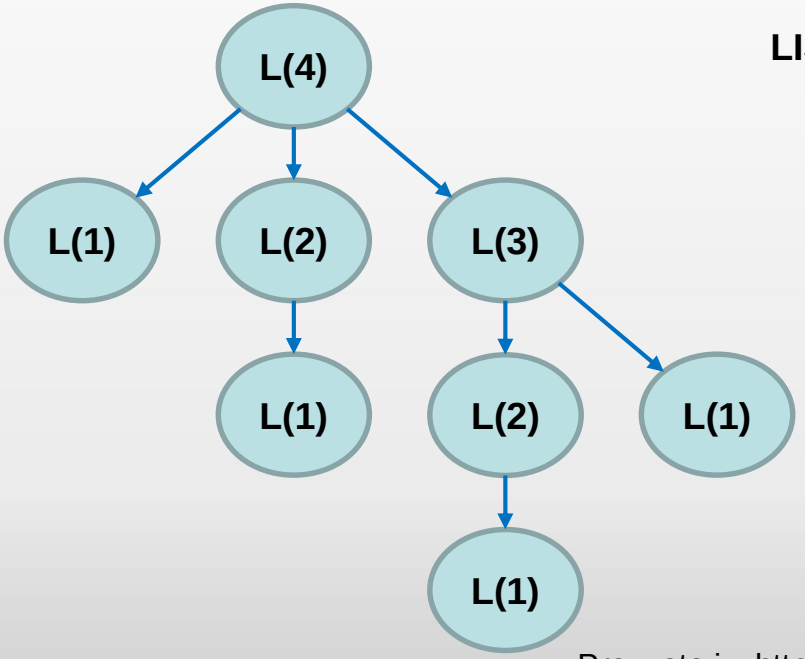
- L(i) možemo izračunati na sljedeći način:

  L(i) = max(L(j)) + 1, za svaki j < i za koji vrijedi niz[i] > niz[j]

  L(i) = 1, ako ne postoji j < i, za koji vrijedi niz[i] > niz[j]

- Primjer:

  niz = [3, 10, 2, 11]



LIS(niz) = max(L(1), L(2), L(3), L(4)) = **3**

Kao i kod primjera s Fibonaccijevim brojevima, vidimo da iste vrijednosti računamo više puta.

Možemo koristiti polje u koje ćemo pohranjivati vrijednosti L(i) te ih računati samo jednom!

Preuzeto iz: https://www.geeksforgeeks.org/longest-increasing-subsequence-dp-3/

Minimizeri

Vrijednosti L(i) računamo redom s lijeva na desno i pohranjujemo ih u polje te koristimo za računanje narednih koraka algoritma.

Primjer:

niz = [3, 10, 2, 11]

	Pomoćno polje
0. niz = [3, 10, 2, 11]	LI = [1, 1, 1, 1]
1. niz = [**3**, 10, 2, 11], LI(1) = 1	LI = [1, 1, 1, 1]
2. niz = [3, **10**, 2, 11], niz[2] > niz[1], LI[2] = max(LI[2], LI[1] + 1)	LI = [1, 2, 1, 1]
3. niz = [3, 10, **2**, 11], niz[3] < niz[1]	LI = [1, 2, 1, 1]
4. niz = [3, 10, **2**, 11], niz[3] < niz[2]	LI = [1, 2, 1, 1]
5. niz = [3, 10, 2, **11**], niz[4] > niz[1], LI[4] = max(LI[4], LI[1] + 1)	LI = [1, 2, 1, 2]
6. niz = [3, 10, 2, **11**], niz[4] > niz[2], LI[4] = max(LI[4], LI[2] + 1)	LI = [1, 2, 1, 3]
7. niz = [3, 10, 2, **11**], niz[4] > niz[3], LI[4] = max(LI[4], LI[3] + 1)	LI = [1, 2, 1, 3]

Preuzeto iz: https://www.geeksforgeeks.org/longest-increasing-subsequence-dp-3/

Minimizeri

```
/* Dynamic Programming C++ implementation of LIS problem */
#include<bits/stdc++.h>
using namespace std;

/* lis() returns the length of the longest
 increasing subsequence in arr[] of size n */
int lis( int arr[], int n )
{
  int lis[n];
   lis[0] = 1;

  /* Compute optimized LIS values in bottom up manner */
  for (int i = 1; i < n; i++ )
  {
    lis[i] = 1;
    for (int j = 0; j < i; j++ )
      if ( arr[i] > arr[j] && lis[i] < lis[j] + 1)
        lis[i] = lis[j] + 1;
  }

  // Return maximum value in lis[]
  return *max_element(lis, lis+n);
}
```

Preuzeto iz: https://www.geeksforgeeks.org/longest-increasing-subsequence-dp-3/

Minimizeri

- Longest increasing subsequence
- **Za početak računamo samo duljinu najduljeg rastućeg podniza**
- Kako vratiti cijelu sekvencu, a ne samo duljinu?

- Osim polja udaljenosti u polju LI, potrebno je pamtiti i najdulje rastuće nizove!

- Složenost: $O(nm) = O(n^2)$

- Postoji algoritam sa složenošću O(nlogn)! (možda sljedeće godine ☺)

Minimizeri

- Primjer:

niz1 = GTCATGCACGTTCACCATGG

[2, 3, 1, 0, 3, 2, 1, 0, 1, 2, 3, 3, 1, 0, 1, 1, 0, 3, 2, 2]

niz2 = TGGCACGACAC

[3, 2, 2, 1, 0, 1, 2, 0, 1, 0, 1]

- Koristimo (3,5)-minimizere

Potencijalno poravnanje (od oka):

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

Minimizeri:

```
[2, 3, 1, 0, 3, 2, -, 1, 0, 1, 2, 3, 3, 1, 0, 1, 1, 0, 3, 2, 2]
[           3, 2, 2, 1, 0, 1, 2, 0, -, 1, 0, 1]
```

Minimizeri

niz1 = GTCATGCACGTTCACCATGG

minimizeri:

(231, 1) ?

(103, 3)

(032, 4)

(101, 7)

(012, 8)

(123, 9)

(233, 10)

(101, 13)

(011, 14)

(032, 17)

(322, 18) ?

Pozicija	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Niz	2	3	1	0	3	2	1	0	1	2	3	3	1	0	1	1	0	3	2	2
	2	**3**	**1**																	
	2	**3**	**1**	0																
	2	3	**1**	**0**	**3**															
		3	1	**0**	**3**	**2**														
			1	**0**	**3**	**2**	1													
				0	**3**	**2**	1	0												
					3	2	**1**	**0**	**1**											
						2	1	**0**	**1**	**2**										
Prozor u kojem je minimizer označen **masnim** slovima							1	**0**	**1**	**2**	3									
								0	**1**	**2**	3	3								
									1	**2**	**3**	3	1							
										2	**3**	**3**	1	0						
											3	3	**1**	**0**	**1**					
												3	1	**0**	**1**	**1**				
													1	**0**	**1**	**1**	0			
														0	**1**	**1**	0	3		
															1	1	**0**	**3**	**2**	
																1	**0**	**3**	**2**	0
																	0	**3**	**2**	0
																		3	**2**	**2**

Minimizeri

niz2 = TGGCACGATAC

[3, 2, 2, 1, 0, 1, 2, 0, 1, 0, 1]

minimizeri:

(322, 1)

(221, 2)

(210, 3)

(101, 4)

(012, 5)

(010, 8)

(101, 9)

Pozicija	1	2	3	4	5	6	7	8	9	10	11
Niz	3	2	2	1	0	1	2	0	1	0	1
Prozor u kojem je minimizer označen **masnim** slovima	**3**	**2**	**2**								
	3	**2**	**2**	**1**							
	3	2	**2**	**1**	**0**						
		2	2	**1**	**0**	**1**					
			2	1	**0**	**1**	**2**				
				1	**0**	**1**	**2**	0			
					0	**1**	**2**	0	1		
						1	2	**0**	**1**	**0**	
							2	**0**	**1**	**0**	1
								0	**1**	**0**	1
									1	**0**	**1**

Minimizeri

niz1 = GTCATGCACGTTCACCATGG

minimizeri:
(231, 1)
(103, 3)
(032, 4)
(101, 7)
(012, 8)
(123, 9)
(233, 10)
(101, 13)
(011, 14)
(032, 17)
(322, 18)

niz2 = TGGCACGATAC

minimizeri:
(322, 1)
(221, 2)
(210, 3)
(101, 4)
(012, 5)
(010, 8)
(101, 9)

Minimizeri

- Kako iskoristiti minimizere za usporedbu nizova?

- 1. varijanta
  - **Izračunati minimizere za prvi niz** — ***O(n)***
  - **Izračunati minimizere za drugi niz** — ***O(n)***
  - *Longest common subsequence problem*
    - Nešto slično smo već računali za sličnost nizova dinamičkim programiranjem — ***$O(n^2)$***

- 2. varijanta
  - **Izračunati minimizere za prvi niz** — ***O(n)***
  - **Izračunati minimizere za drugi niz** — ***O(n)***
  - Usporediti dvije grupe minimizera i pronaći poklapanja (*engl. match*) — ***O(nlogn)***
  - Sortirati poklapanja prema poziciji u prvom nizu — ***O(nlogn)***
  - *Longest increasing subsequence problem* nad pozicijama u drugom nizu — ***O(nlogn)***

Minimizeri

1. Varijanta: Longest common subsequence

			niz1											
			1	3	4	7	8	9	10	13	14	17	18	pos
			231	103	032	101	012	123	233	101	011	032	322	mini-mizer
		0	0	0	0	0	0	0	0	0	0	0	0	
1	322	0	0	0	0	0	0	0	0	0	0	0	0	
2	221	0	0	0	0	0	0	0	0	0	0	0	0	
3	210	0	0	0	0	0	0	0	0	0	0	0	0	
4	**101**	0	0	0	0	1	1	1	1	1	1	1	1	
5	**012**	0	0	0	0	1	2	2	2	2	2	2	2	
8	010	0	0	0	0	1	2	2	2	2	2	2	2	
9	**101**	0	0	0	0	1	2	2	2	3	3	3	3	
pos	mini-mizer													

niz2

Minimizeri

Tablica dinamičkog programiranja

Najdulji zajednički niz minimizera = 3

Početak poravnanja: (101, 7, 4)

Kraj poravnanja: (101, 13, 9)

Poravnanje: niz1 se poravnava s niz2[4:15]

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

			niz1												
			1	3	4	7	8	9	10	13	14	17	18	pos	
			231	103	032	101	012	123	233	101	011	032	322	mini-mizer	
		0	0	0	0	0	0	0	0	0	0	0	0		
1	322	0	0	0	0	0	0	0	0	0	0	0	0		
2	221	0	0	0	0	0	0	0	0	0	0	0	0		
3	210	0	0	0	0	0	0	0	0	0	0	0	0		
4	101	0	0	0	0	1	1	1	1	1	1	1	1		
5	012	0	0	0	0	1	2	2	2	2	2	2	2		
8	010	0	0	0	0	1	2	2	2	2	2	2	2		
9	101	0	0	0	0	1	2	2	2	3	3	3	3		
pos	mini-mizer														

niz2

Minimizeri

2. Varijanta
   - Usporediti dvije grupe minimizera i pronaći poklapanja (*engl. match*)
   - Sortirati poklapanja prema poziciji u prvom nizu
   - *Longest increasing subsequence problem* nad pozicijama u drugom nizu

Poklapanja (minimizer, pozicija u nizu 1, pozicija u nizu 2):

(101, 7, 4)
(101, 7, 9)
(101, 13, 4)
(101, 13, 9)
(012, 8, 5)
(322, 18, 1)

(101, 7, 4)
(101, 7, 9)
(012, 8, 5)
(101, 13, 4)
(101, 13, 9)
(322, 18, 1)

niz1
(231, 1)
(103, 3)
(032, 4)
(101, 7)
(012, 8)
(123, 9)
(233, 10)
(101, 13)
(011, 14)
(032, 17)
(322, 18)

niz2
(322, 1)
(221, 2)
(210, 3)
(101, 4)
(012, 5)
(010, 8)
(101, 9)

Minimizeri

2. Varijanta

– *Longest increasing subsequence problem* nad pozicijama u drugom nizu

Inicijalizacija:

[(101, 7, **4**), (101, 7, **9**), (012, 8, **5**), (101, 13, **4**), (101, 13, **9**), (322, 18, **1**)]

LI = [1, 1, 1, 1, 1, 1]

LIS[1]
(101, 7, 4)

1. korak

[(101, 7, **4**), (101, 7, **9**), (012, 8, **5**), (101, 13, **4**), (101, 13, **9**), (322, 18, **1**)]

LI = [1, **1**, 1, 1, 1, 1] LI[2] = 1, ne može se nastaviti na LI[0], ista pozicija

LIS[2]
(101, 7, 9)

2. korak

[(101, 7, **4**), (101, 7, **9**), (012, 8, **5**), (101, 13, **4**), (101, 13, **9**), (322, 18, **1**)]

LI = [1, 1, **2**, 1, 1, 1] LI[3] = 2 (**5 > 4**) (LI[1] + 1)

LIS[3]
(101, 7, 4)
(012, 8, 5)

3. korak

[(101, 7, **4**), (101, 7, **9**), (012, 8, **5**), (101, 13, **4**), (101, 13, **9**), (322, 18, **1**)]

LI = [1, 1, 2, **1**, 1, 1] LI[4] = 1 (**4** ≤ **4, 5, 9**)

LIS[4]
(101, 13, 4)

Minimizeri

2. Varijanta
   - *Longest increasing subsequence problem* nad pozicijama u drugom nizu

3. korak

   [(101, 7, **4**), (101, 7, **9**), (012, 8, **5**), (101, 13, **4**), (101, 13, **9**), (322, 18, **1**)]

   LI = [1, 1**,** 2, **1**, 1, 1] LI[4] = 1 (**4 < 5, 9**)

LIS[4]
(101, 13, 4)

4. korak

   [(101, 7, **4**), (101, 7, **9**), (012, 8, **5**), (101, 13, **4**), (101, 13, **9**), (322, 18, **1**)]

   LI = [1, 1**,** 2, 1, **3**, 1] LI[5] = 3 (**9 > 5**) (LI[3] + 1)

LIS[5]
(101, 7, 4)
(012, 8, 5)
(101, 13, 9)

5. korak

   [(101, 7, **4**), (101, 7, **9**), (012, 8, **5**), (101, 13, **4**), (101, 13, **9**), (322, 18, **1**)]

   LI = [1, 1**,** 2, 1, 3, **1**] LI[6] = 1 (1 **< 4, 9, 5**)

LIS[6]
(322, 18, 1)

Minimizeri

2. Varijanta
   - *Longest increasing subsequence problem* nad pozicijama u drugom nizu

- Najveći LI je LI[5] = 3
- Najdulji rastući niz je LIS[5] = 4, 5 ,9 koji odgovara minimizerima:

LIS[5]
(101, 7, 4)
(012, 8, 5)
(101, 13, 9)

- Isto kao i kod varijante 1 – očekivano
  - Potencijalno brže

Minimizeri

- Praktična implementacija
  - Kako odabrati k i w?
    - Neki alati koriste k = 15, w = 5
  - Koje vrijednosti dodijeliti pojedinim nukleotidima i amino-kiselinama (može se raditi i s proteinima)?
    - U primjeru smo radili s brojevima, ali u bioinformatici radimo sa slovima
    - Možemo koristiti leksikografske vrijednosti
    - Možemo manje vrijednosti dodijeliti slovima koja se pojavljuju rjeđe – veća vjerojatnost da će rjeđi k-meri biti minimizeri
    - Poly-A tail – poseban problem kod RNA
  - Problem minimizera koji se često pojavljuju
    - Najčešće se izbacuju
  - DNA ima dva lanca
    - Svako očitanje uspoređujemo dva puta, jednom za glavni lanac, jednom za reverzni komplement
    - Računamo minimizere za oba slučaja
    - Kod mapiranja očitanja na referencu, reverzni komplement računamo ili za očitanja ili za referencu (nema potrebe za oboje)
    - **Sortiramo poklapanja minimizera i po oznaci lanca!**
  - **Prevelika razlika udaljenosti kod poklapanja minimizera, između minimizera na jednom i drugom nizu – LIS se prekida – ovisi o primjeni!**