Projekti na predmetu Bioinformatika, 2016./2017.

Pored naziva svakoga projekta navedeni su inicijali predavača koji je projekt osmislio, izv. prof. dr. sc. Mile Šikić (MŠ) ili doc. dr. sc. Mirjana Domazet-Lošo (MDL), te se njima možete javiti za sva pitanja i detaljnije upute za pojedine projekte.

Projekti za 40 bodova

- broj članova tima: 4-6
- implementacija obvezno treba biti u programskim jezicima C i Python te drugim jezicima po izboru
- dopuštene su dvije različite implementacije u istom programskom jeziku, ali razlike u implementaciji trebaju biti vidljive
- u projektnu dokumentaciju uključiti opis algoritma i usporedbu performansi svih implementaciji (vrijeme izvođenja, utrošak memorije, točnost)
- ulazni formati podataka: FASTA i običan tekst (bilo koji niz znakova)
    - ako program kao neposredne ulazne podatke zahtijeva podatke koji nisu u FASTA formatu, onda je potrebno podatke pretprocesirati; pri tome je dozvoljeno koristiti postojeće knjižnice ili programe
- za svaki dan zakašnjenja umanjuje se konačan broj bodova za 3 boda

	Broj bodova
Program • ako program ne radi ispravno na testnim podatcima prilikom demonstracije umanjuje se konačan broj bodova za 5 bodova • prepravke je potrebno napraviti u roku od 2 dana	20
Testiranje na sintetskim podatcima 10^2-10^6 znakova • svi rezultati moraju biti u dokumentaciji – prikazani u tablici i/ili grafu (paralelno za sve implementacije)	3
Testiranje na stvarnim podatcima (*Escherichia coli*) • svi rezultati moraju biti u dokumentaciji – prikazani u tablici i/ili grafu (paralelno za sve implementacije)	3
Dokumentacija (zajedno za sve članove) • opis algoritma i vizualizacija na jednostavnom primjeru • obvezno navesti popis literature i navesti izvore unutar teksta • za svaki algoritam napraviti analizu točnosti, vremena izvođenja i utroška memorije za različite testne slučaje	8
Prezentacija • oduzimaju se bodovi, ako je prezentacija dulja od predviđenoga vremena	6

Popis tema s poveznicama na literaturu:

(1) **Izgradnja sufiksnog polja korištenjem Nong-Zhang-Chanovog algoritma SA-DS (Nong et al. 2009; 2011)** (MDL)

G. Nong, S. Zhang and W. H. Chan, Two Efficient Algorithms for Linear Time Suffix Array Construction, IEEE Transactions on Computers, Vol. 60, No. 10, Oct. 2011. [draft, SAIS code]

https://code.google.com/p/ge-nong/

(2) **Usporedba heurističkih programa za određivanje višestrukog poravnanja sljedova (eng. *multiple sequence alignment*)** (MDL)

Svaki član tima treba proučiti i testirati (barem) jedan program, a zatim se uspoređuju vremena izvođenja i memorijski zahtjevi svih odabranih programa, njihova ograničenja i prednosti. Naravno,
svaki član treba odabrati program, koji je različit od programa koje su odabrali ostali članova tima.

http://www.ebi.ac.uk/Tools/msa/

https://en.wikipedia.org/wiki/Multiple_sequence_alignment

A Thompson et al. 2011. Comprehensive Benchmark Study of Multiple Sequence Alignment Methods: Current Challenges and Future Perspectives

http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0018093

Notredame, 2007. Recent Evolutions of Multiple Sequence Alignment Algorithms

http://www.tcoffee.org/Publications/Pdf/PLOS_review.pdf

(3) **Usporedba programa (alata) za mapiranje očitanja na genom (npr. SOAP2, Bowtie2, Bwamem, Maq, bbmap, SNAP, Graphmap)** (MDL)

Svaki član tima treba proučiti i testirati (barem) jedan program, a zatim se uspoređuju vremena izvođenja i memorijski zahtjevi svih odabranih programa, njihova ograničenja i prednosti. Naravno, svaki član treba odabrati program, koji je različit od programa koje su odabrali ostali članova tima.

(4) **LCSk++** (MŠ)

http://arxiv.org/pdf/1407.2407.pdf

(5) **Hunt Szymanski algorithm** (MŠ)

http://www.cs.bgu.ac.il/~dpaa111/wiki.files/HuntSzymanski.pdf

(6) **Minimazer** (MŠ)

http://www.csri.utoronto.ca/~wayne/research/papers/minimizers.pdf

(7) **Watterman-Eggertov algoritam** (MŠ)

http://dornsife.usc.edu/assets/sites/516/docs/papers/msw_papers/msw-079.pdf

(8) **Testiranje perfomansi za de novo assemblere** (MŠ).

Svaki član tima treba odabrati po jedan de novo assembler (SGA, Spades, Abyss, IDBA i ALLPaths-LG) i analizirati njegove performanse za Illumina i PacBio tip podataka (koristiti wgsim i pbsim alate). Kao genom uzeti genome neke bakterije http://bacteria.ensembl.org/index.html (npr. Escherichia coli). Testirati vrijeme izvođenja, N25, N50, N70 najveću duljinu kontiga, prosječnu duljinu kontiga, ukupan broj nukleotida u svim kontizima te napraviti pokrivenost referentnoga genoma bakterije sa skafoldima i kontizima koristeci MUMmer, LASTZ ili BLASR alat. Svi rezultati trebaju biti prikazano zasebno po assemblerima i skupa da se mogu usporediti.

Projekti za 100 bodova

- broj članova tima: 1-3
- implementacija: C/C++
- opis algoritma, implementacije i testiranje
- dozvoljeno je korištenje pomoćnih knjižnica u zadatcima gdje je tako navedeno, a za ostale situacije možete se dogovoriti s nastavnikom koji je zadao temu
- za svaki dan zakašnjenja umanjuje se konačan broj bodova za 3 boda

Napomene:

Projekti za 100 bodova su teški - dobro pročitajte prije nego odaberete ☺

Prepisivanje gotovih rješenja automatski nije dozvoljeno i u tome slučaju projekt dobiva 0 bodova.

Bodovanje (MDL; zadatci (1)-(8))

	Broj bodova
Program • ako program ne radi ispravno na testnim podatcima prilikom demonstracije umanjuje se konačan broj bodova za 10 bodova • prepravke napraviti u roku od 2 dana performanse programa (vrijeme izvođenja i utrošak memorije) a. ako se program uspoređuje sa studentskim rješenjem od prošle godine, vrijeme izvođenja i utrošak memorije implementacije ne smiju biti lošiji od 10% u odnosu na navedenu referencu b. ako se program uspoređuje s objavljenim rješenjem, vrijeme izvođenja i utrošak memorije implementacije ne smiju biti lošiji od 70% u odnosu na navedenu referencu • oduzima se 10 bodova, ako je za (a) odstupanje do 20%, odnosno do 100% za (b)	60

• oduzima se 15 bodova, ako je za (a) odstupanje veće od 20%, odnosno veće od 100% za (b)	
Testiranje na sintetskim podatcima 10^2-10^6 znakova • svi rezultati moraju biti u dokumentaciji – prikazani u tablici i/ili grafu	10
Testiranje na stvarnim podatcima (*Escherichia coli*) • svi rezultati moraju biti u dokumentaciji – prikazani u tablici i/ili grafu	10
Dokumentacija • opis algoritma i vizualizacija na jednostavnom primjeru • obvezno navesti popis literature i navesti izvore unutar teksta • za svaki algoritam napraviti analizu točnosti, vremena izvođenja i utroška memorije za različite testne slučaje	15
Prezentacija • oduzimaju se bodovi, ako je prezentacija dulja od predviđenoga vremena	5

(1) **Računanje najduljeg zajedničkog prefiksa temeljeno na BWT** (MDL)

Timo Beller, Simon Gog, Enno Ohlebusch, Thomas Schnattinger: Computing the longest common prefix array based on the Burrows–Wheeler transform, 2013.

Timo Beller, Simon Gog, Enno Ohlebusch, Thomas Schnattinger: Computing the Longest Common Prefix Array Based on the Burrows-Wheeler Transform – Slides 2013.http://www.csse.monash.edu.au/~gfarr/research/slides/Beller.pdf

U izradi programa:

- implementirati algoritme 1 i 2 iz rada Beller et al. 2013.
- smije se koristiti gotova knjižnica za izgradnju suf. polja (npr., sais, sais-lite, itd.)
- testirati s gotovom knjižnicom sa stablom valića te napraviti svoju implementaciju stabla valića, odnosno funkcije rang (engl. *rank*)

Usporediti s rezultatima Ivane Vanjak, Darie Bužić i Filipa Kozjaka C+; https://github.com/wissil/LCP_BWT

(2) **Izgradnja binarnog stabla valića kao RRR strukture** (MDL) (stablo valića; eng. *wavelet tree*)

http://en.wikipedia.org/wiki/Wavelet_Tree

http://alexbowe.com/wavelet-trees/

http://www.dcc.uchile.cl/~gnavarro/ps/cpm12.pdf

Alexander Bowe, 2010. Multiary Wavelet Trees in Practice (Honours Thesis)

http://alexbowe.com/rrr/

Usporediti s rezultatima Denisa Čaušević̀a (0036462803) i Hajrudina Ćoralića (0036469477), C++

https://github.com/Vaan5/Bioinformatics-Construction-of-binary-wavelet-trees-using-RRR-structure

(3) RBTree + rang/odaberi operacije (eng. *rank/select*) (MDL)

Rodrigo González, Gonzalo Navarro. Rank/select on dynamic compressed sequences and applications , Theoretical Computer Science 410 , 2009 pages 4414-4422,
https://www.dcc.uchile.cl/~gnavarro/algoritmos/ps/tcs08.pdf
Navarro, Gonzalo. Wavelet trees for all, Journal of Discrete Algorithms 25 2014 pages 2-20.
https://www.dcc.uchile.cl/~gnavarro/ps/cpm12.pdf

Usporediti s rezultatima Stjepana Livačića, Valentina Perovića i Marte Poštenjak C+;
https://github.com/vp4655/bioinformatics

(4) **Izgradnja sufiksnog polja korištenjem SACA-K algoritma** (Nong, 2013) (MDL)

G. Nong, Practical Linear Time O(1) Workspace Suffix Sorting for Constant Alphabets, ACM Transactions on Information Systems, Vol. 31, No. 3, Jul. 2013. [draft, code](The presented algorithm, called SACAK, was previously called OSACA and described here: An Optimal Suffix Array Construction Algorithm, Technical Report, Department of Computer Science, Sun Yatsen University, 2011.)

Usporediti s originalnom implementacijom: https://ge-nong.googlecode.com/files/saca-k-tois.pdf

(5) essaMEM: finding maximal exact matches using enhanced sparse suffix arrays (Vyverman et al. 2013)

Michaël Vyverman, Bernard De Baets, Veerle Fack, and Peter Dawyndt. essaMEM: finding maximal exact matches using enhanced sparse suffix arrays. Bioinformatics (2013) 29 (6): 802-804
http://bioinformatics.oxfordjournals.org/content/29/6/802.full.pdf+html

U izradi programa:

- smije se koristiti gotova knjižnica za izgradnju suf. polja (npr., sais, sais-lite, itd.)

Usporediti s originalnom implementacijom: https://github.com/readmapping/essaMEM

(6) A Linear-Time Burrows-Wheeler Transform Using Induced Sorting (Okanohara and Sadakane, 2009);
https://www.researchgate.net/publication/221580028_A_Linear-Time_Burrows-Wheeler_Transform_Using_Induced_Sorting

- Directly compute BWT from the input

U izradi programa:

- smije se koristiti gotova knjižnica za izgradnju suf. polja (npr., sais, sais-lite, itd.)

Usporediti s originalnom implementacijom: http://researchmap.jp/muuw41s7s-1587/#_1587 (dbwt)

(7) **A representation of a compressed de Bruijn graph for pan-genome analysis that enables search** (Beller and Ohlebusch, 2016) https://almob.biomedcentral.com/articles/10.1186/s13015-016-0083-7

- Implementirati algoritme A1 i A2

U izradi programa:

- smije se koristiti gotova knjižnica sdsl
- smije se koristiti gotova knjižnica za izgradnju suf. polja (npr., sais, sais-lite, itd.)

Usporediti s originalnom implementacijom: https://www.uni-ulm.de/in/theo/research/seqana.html

(8) Traženje podudarnih nizova uz *k* različitih znakova

Marius Nicolae and Sanguthevar Rajasekaran, On string matching with k mismatches, 2013. https://arxiv.org/pdf/1307.1406v1.pdf (Algoritam 6, varijanta s naivnim pretprocesiranjem i varijanta s pretprocesiranjem uz korištenje sufisknog polja)

U izradi programa:

- smije se koristiti gotova knjižnica za izgradnju suf. polja (npr., sais, sais-lite, itd.) i lcp polja

Usporediti s originalnom implementacijom: http://www.engr.uconn.edu/~man09004/kmis.zip.

(9) Faza razmještaja u OLC paradigmi sastavljanja genoma (MŠ)

Ulaz: skup očitanja u FASTA formatu, skup preklapanja svih očitanja međusobno (MHAP ili PAF format, trebate generirati sami pomoću minimap-a (https://github.com/lh3/minimap) ili GraphMap-a (https://github.com/isovic/graphmap) ili oba zasebno za bolju evaluaciju vaših rješenja).

Cilj: Izgraditi vrstu grafa preklapanja te iz njega rekonstruirati sekvencirani genom.

Izlaz: kontizi (u najboljem slučaju 1 po kromosomu) u FASTA formatu.

Evaluacija: koristiti alat Gepard (https://github.com/univieCUBE/gepard) za provjeru jesu li kontizi dobro sastavljeni. Za vizualizaciju svojih grafova preklapanja možete koristiti Cytoscape (http://www.cytoscape.org/). Za testne skupove javiti se nastavniku.

Bodovanje:

	Broj bodova

Program
 	
• ako program ne radi ispravno na testnim podatcima prilikom demonstracije umanjuje se konačan broj bodova za 10 bodova (prepravke napraviti u roku od 2 dana)
• vremensko ograničenje od 2h na 1 dretvi, u protivnom se oduzima 5 bodova
• memorijsko ograničenje od 16 GB RAM-a, u protivnom se oduzima 5 bodova
• točnost rezultata:
  ○ za više od 1 kontiga ili razlika u duljini koja je veća od 5% u odnosu na referencu, oduzima se 10 bodova
  ○ za više od 5 kontiga ili razlika u duljini koja je veća od 10% u odnosu na referencu, oduzima se 25 bodova
	80

Dokumentacija • opis algoritma i vizualizacija na jednostavnom primjeru • obavezno navesti popis literature te navesti izvore unutar teksta • napraviti usporedbu točnosti, vremena izvođenja i utroška memorije vaše implementacije i izvorne	15

Prezentacija • oduzimaju se bodovi, ako je prezentacija dulja od predviđenoga vremena	5

Metode (bira se jedna! Do tri tima po temi):

1. Faza razmještaja implementirana u asembleru Miniasm koji koristi **neispravljena** očitanja
    a. Implementirati assembly graph te metode pojednostavljenja istog
        i. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences (http://tinyurl.com/gm4hvzf)
2. Faza razmještaja implementirana u asembleru Canu koji koristi **ispravljena** očitanja
    a. Implementirati best overlap graph (Bogart)
        i. Canu: scalable and accurate long-read assembly via adaptive k-mer weigting and repeat separation (http://tinyurl.com/jx63ssn)

(10) Faza konsenzusa u OLC paradigmi sastavljanja genoma (MŠ)

Ulaz: kontizi u FASTA formatu (85% točnosti kada se usporede s referencom), skup očitanja u FASTQ formatu, skup mapiranja očitanja na kontige (MHAP ili PAF format, trebate generirati sami pomoću minimap-a (https://github.com/lh3/minimap) ili GraphMap-a (https://github.com/isovic/graphmap ili oba zasebno za bolju evaluaciju vaših rješenja).

Cilj: Pomoću skupa početnih očitanja povećati točnost kontiga dobivenih iz faze razmještaja.

Izlaz: popravljeni kontizi u FASTA formatu.

Evaluacija: koristiti alat DnaDiff iz mummer paketa (sudo apt-get install mummer). Za testne skupove javiti se nastavniku.

Bodovanje:

	Broj bodova

Program
• ako program ne radi ispravno na testnim podatcima prilikom demonstracije umanjuje se konačan broj bodova za 10 bodova (prepravke napraviti u roku od 2 dana)
• vremensko ograničenje od 2h na 1 dretvi, u protivnom se oduzima 5 bodova
• memorijsko ograničenje od 16 GB RAM-a, u protivnom se oduzima 5 bodova
• točnost rezultata:
  ○ za poboljšanje manje od 10% točnosti oduzima se 10 bodova
  ○ za poboljšanje manje od 5% točnosti oduzima se 25 bodova
	80

Dokumentacija • opis algoritma i vizualizacija na jednostavnom primjeru • obavezno navesti popis literature te navesti izvore unutar teksta • napraviti usporedbu točnosti, vremena izvođenja i utroška memorije vaše implementacije i izvorne	15
Prezentacija • oduzimaju se bodovi, ako je prezentacija dulja od predviđenoga vremena	5

Metode (bira se jedna! Do tri tima po temi):

1. Faza konsenzusa implementirana u modulu Racon
    a. Implementirati POA algoritam

        i. Multiple sequence alignment using partial order graphs (http://tinyurl.com/hxxco8u).
        ii. Generating consensus sequences from partial order multiple sequence alignment graphs (http://tinyurl.com/h7j5zpw).
    b. Implementirati konsenzus modul pomoću a dijela.
        i. Fast and accurate de novo genome assembly from long uncorrected reads (http://tinyurl.com/jgt5r6l).
2. Faza konsenzusa implementirana u modulu Sparc
    a. Implementirati sparse k-mer graph
        i. Sparc: a sparsity-based consensus algorithm for long erroneous sequencing reads (http://tinyurl.com/zqjcd9q).
    b. Implementirati konsenzus modul pomoću a dijela.
        i. Referenca jednaka onoj pod a.