Aspetti algoritmici di base nell'analisi di dati NGS

Basic Algorithmic Aspects of NGS data analysis

NGS Milan Meeting, 17 luglio 2012

Gianluca Della Vedova
Università degli Studi di Milano-Bicocca
gianluca.dellavedova@unimib.it

Efficient Algorithms and NGS

Perfect match

Lots of data

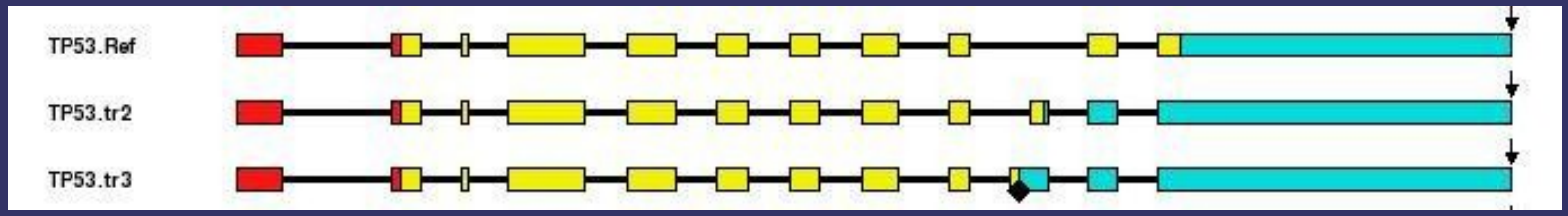
- **short** fragments
- Different error distribution
  - Too many errors

You cannot be too fast

You cannot be too space efficient

NGS in Transcriptomics

- **Genome Annotation (with RNA-seq data)**
  - splice junctions (exon-intron boundaries)
  - reconstruction of full-length isoforms
  - expression level of known isoforms
  - validation of predicted isoforms



http://t.caspur.it/ASPicDB/index.php

The Most Basic Problem: Alignment

- Sequence comparison
- Pattern matching
- Strategy:
  - Find *seeds* (short perfect matches)
  - Filter seeds
  - Extend seeds

NGS Alignment

INPUT: a set R of NGS data (genomic reads or RNA-seq reads) and a reference sequence

OUTPUT: for each read r in R, find the reference regions very similar to r

APPROACH: align each read to the reference sequence

Basic problem: Alignment

- Gapped/ungapped alignment of a **huge amount** of reads to a **long** reference sequence (genome)
- **Goal**: speed and accuracy
- **Strategy**: find seeds → time-space efficient data structures for indexing reads and/or reference

Index

- Index = data structure for pattern matching
  - Seed Hash Table
  - Suffix Tree
  - Suffix Array
  - Burrows-Wheeler Transform (BWT) and FM-index

Seed Hash Table

S = acgtgacgtg

acgt	1,6
cgtg	2,7
gtga	3
tgac	4
gacg	5

Spaced seed (1101):

ac*t, cg*g,

gt*a, tg*c,

ga*g

Human Genome >12GB

SOAP, MAQ

Suffix Tree

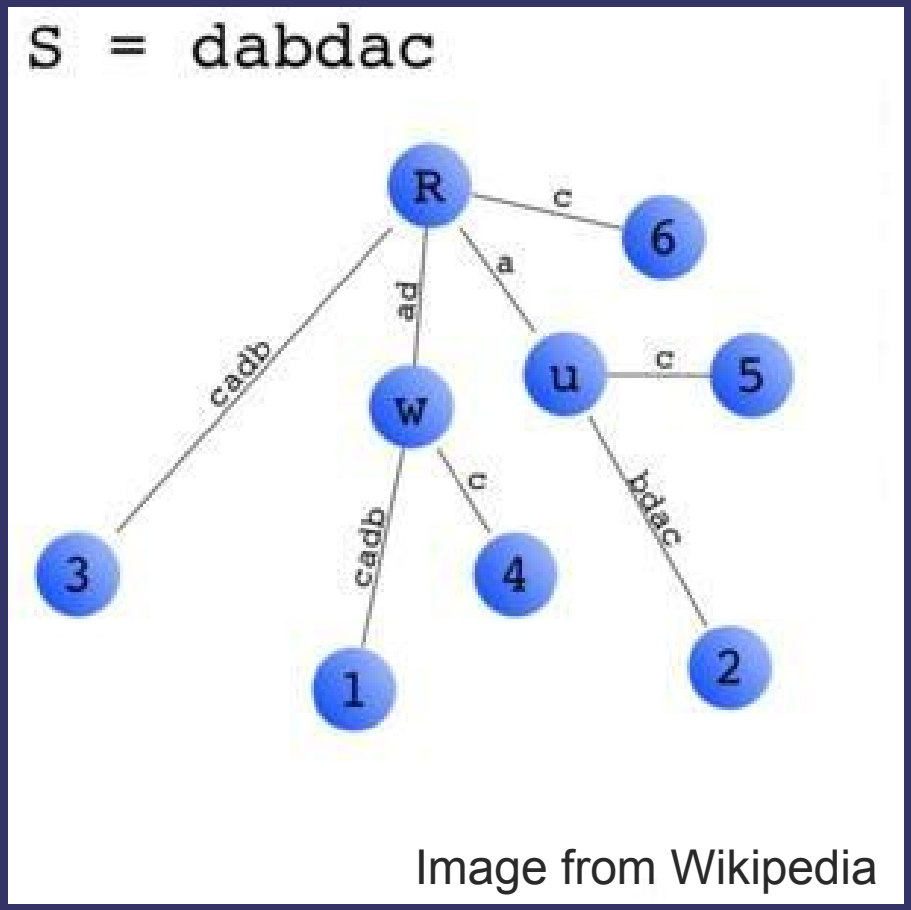


Image from Wikipedia

Human Genome >35GB

MUMmer (not for NGS)

Suffix array

S = acgtgacgtg$

acgtgacgtg$	1
cgtgacgtg$	2
gtgacgtg$	3
tgacgtg$	4
gacgtg$	5
acgtg$	6
cgtg$	7
gtg$	8
tg$	9
g$	10
$	11

	SA
$	11
acgtg$	6
acgtgacgtg$	1
cgtg$	7
cgtgacgtg$	2
g$	10
gacgtg$	5
gtg$	8
gtgacgtg$	3
tg$	9
tgacgtg$	4

Human Genome ~12GB

Vmatch, Segemehl

Burrows-Wheeler Transform (BWT)

```
T = acgtgacgtg$
```

Burrows-Wheeler Transform (BWT)

T = acgtgacgtg$

acgtgacgtg$
cgtgacgtg$a
gtgacgtg$ac
tgacgtg$acg
gacgtg$acgt
acgtg$acgtg
cgtg$acgtga
gtg$acgtgac
tg$acgtgacg
g$acgtgacgt
$acgtgacgtg

Burrows-Wheeler Transform (BWT)

T = acgtgacgtg$

acgtgacgtg$	$acgtgacgtg
cgtgacgtg$a	acgtg$acgtg
gtgacgtg$ac	acgtgacgtg$
tgacgtg$acg	cgtg$acgtga
gacgtg$acgt	cgtgacgtg$a
acgtg$acgtg	g$acgtgacgt
cgtg$acgtga	gacgtg$acgt
gtg$acgtgac	gtg$acgtgac
tg$acgtgacg	gtgacgtg$ac
g$acgtgacgt	tg$acgtgacg
$acgtgacgtg	tgacgtg$acg

Burrows-Wheeler Transform (BWT)

T = acgtgacgtg$

	BWT
acgtgacgtg$	$acgtgacgtg
cgtgacgtg$a	acgtg$acgtg
gtgacgtg$ac	acgtgacgtg$
tgacgtg$acg	cgtg$acgtga
gacgtg$acgt	cgtgacgtg$a
acgtg$acgtg	g$acgtgacgt
cgtg$acgtga	gacgtg$acgt
gtg$acgtgac	gtg$acgtgac
tg$acgtgacg	gtgacgtg$ac
g$acgtgacgt	tg$acgtgacg
$acgtgacgtg	tgacgtg$acg

Burrows-Wheeler Transform (BWT)

T = acgtgacgtg$

acgtgacgtg$	$acgtgacgtg
cgtgacgtg$a	acgtg$acgtg
gtgacgtg$ac	acgtgacgtg$
tgacgtg$acg	cgtg$acgtga
gacgtg$acgt	cgtgacgtg$a
acgtg$acgtg	g$acgtgacgt
cgtg$acgtga	gacgtg$acgt
gtg$acgtgac	gtg$acgtgac
tg$acgtgacg	gtgacgtg$ac
g$acgtgacgt	tg$acgtgacg
$acgtgacgtg	tgacgtg$acg

F BWT

Inverting BWT

BWT
g
g
$
a
a
t
t
c
c
g
g

Inverting BWT

F
$
a
a
c
c
g
g
g
g
t
t

BWT
g
g
$
a
a
t
t
c
c
g
g

Inverting BWT

F
$
a
a
c
c
g
g
g
g
t
t

BWT
g
g
$
a
a
t
t
c
c
g
g

BWT	F
g	$
g	a
$	a
a	c
a	c
t	g
t	g
c	g
c	g
g	t
g	t

Inverting BWT

F
$
a
a
c
c
g
g
g
g
t
t

BWT
g
g
$
a
a
t
t
c
c
g
g

BWT	F
g	$
g	a
$	a
a	c
a	c
t	g
t	g
c	g
c	g
g	t
g	t

F2
$a
ac
ac
cg
cg
g$
ga
gt
gt
tg
tg

Inverting BWT

F
$
a
a
c
c
g
g
g
g
t
t

BWT
g
g
$
a
a
t
t
c
c
g
g

BWT	F
g	$
g	a
$	a
a	c
a	c
t	g
t	g
c	g
c	g
g	t
g	t

BWT	F2
g	$a
g	ac
$	ac
a	cg
a	cg
t	g$
t	ga
c	gt
c	gt
g	tg
g	tg

BWT: pattern searching

S = acgtgacgtg

P = acg

BWT
$acgtgacgtg
acgtg$acgtg
acgtgacgtg$
cgtg$acgtga
cgtgacgtg$a
g$acgtgacgt
gacgtg$acgt
gtg$acgtgac
gtgacgtg$ac
tg$acgtgacg
tgacgtg$acg

Burrows-Wheeler Transform (BWT) vs Suffix Array

	SA
$	11
acgtg$	6
acgtgacgtg$	1
cgtg$	7
cgtgacgtg$	2
g$	10
gacgtg$	5
gtg$	8
gtgacgtg$	3
tg$	9
tgacgtg$	4

	BWT
$acgtgacgtg	g
acgtg$acgtg	g
acgtgacgtg$	$
cgtg$acgtga	a
cgtgacgtg$a	a
g$acgtgacgt	t
gacgtg$acgt	t
gtg$acgtgac	c
gtgacgtg$ac	c
tg$acgtgacg	g
tgacgtg$acg	g

FM-index

- Compressed genome
- Compressed information for pattern matching
  - C(x) = number of characters of S smaller than x
  - Occ(x,i) = occurrence of x in BWT[1:i]
  - SA(T) = suffix array

Proprietà:

begin(aP) = C(a) + Occ(a,begin(P)-1)
end(aP) = C(a) + Occ(a,end(P))-1
BWT[i] = T[SA(i) – 1]

BWT e FM-index

- Bowtie
  - ungapped alignment (no indels)
  - human genome in 1.3 GB
- BWA
  - gapped alignment (with indels)
- SOAP2
- Bowtie2 (gapped alignment)

Thanks

!

Research group www.algolab.eu

- Paola Bonizzoni
- Gianluca Della Vedova
- Raffaella Rizzi
- Yuri Pirola (Post-doc)
- Stefano Beretta (Ph.D. student)
- And more...

References

- Indexing compressed text. Paolo Ferragina, Giovanni Manzini. http://doi.acm.org/10.1145/1082036.1082039
- A block sorting lossless data compression algorithm. M. Burrows and D. Wheeler. http://www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-124.pdf
- Fast and accurate short read alignment with Burrows–Wheeler transform. Heng Li and Richard Durbin, doi: 10.1093/bioinformatics/btp324
- Exploring single-sample SNP and INDEL calling with whole-genome de novo assembly. Heng Li. doi: 10.1093/bioinformatics/bts280
- Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Ben Langmead, Cole Trapnell, Mihai Pop and Steven L Salzberg. doi:10.1186/gb-2009-10-3-r25