858W NOTES 03.09.2010

1-Exam expectations
2-Project proposals are due 03.26.2010
 half page to 1 page description of project
3-Extra class 04.02.2010 @ 2 pm 2 hour class
 Everyone will talk about project proposals that will probably include
  Motivation
  What do you plan to do?
  What is your expected outcome?

To sequence 20 million pairs with average length of sequence 60 bp takes approximately 8 hours.

For example bacteria has average 5 million pairs.Sequencing one time takes 5.5 hours.Assume we repeat 16 times,it takes more than overnight.Then it takes almost 1 week to analyze the data.

Bowtie tells us where sequence matches and also mismatches in genome.Index sequences in a way(cluster) that ask question "Does this group match?" instead of "Does this sequence match?".This speeds up Bowtie.

If we can tolerate some error,then we can do inexact matching.Cluster 1.5 million of pairs less than an hour.At least 100 to 1000 samples for good clustering.

In 80's and 90's.There was less sequence and human does the sequencing job.However now,solving in less memory is more important which was not that important in 80's.

Search On BWT For A String Review

Assume we take rotations of BANANA$ starting from each index.Since there are n indexes, there will be n different right-to-left rotations.Some prefixes of these rotations will be suffixes of original string BANANA$.If we sort these and get the rightmost character from each row,we get BWT of the string.

BANANA$		$banana	BWT is	A
ANANA$		A$banan		N
NANA$	->SORT	ANA$ban		N
ANA$		ANANA$b		B
NA$		BANANA$		$
A$		NAS$bana		A
$		NANA$ba		A

We want to search for string ANA in that string.We go from right-to-left.Remember we're not saving all the arrays above.We only store BWT which is last column and the first column.

1-Find all A's on the first column.(It is between second and fourth rows).Then we find the part of that rows which has next character of string which is N in their last columns.

This is rows from second to third rows both of them being inclusive.We currently found all substrings of the form *A.

2-Then find the corresponding N's in the first column which are sixth and seventh rows.Then we find subpart of that rows which has next character A.This is the same interval(sixth to seventh rows both being inclusive).This means we found all substrings of the form *NA.

3-Then find the corresponding A's in the first column.which are third and fourth rows.Since there is not any more character to check,we are done and find two substrings that has ANA which are the prefixes of the suffixex in the third and fourth rows of Suffix Array.

One important thing will be determine intervals.We can form the data structure shown below.Our alphabet has 3 characters and for each row we keep the number of times it has been seen in BWT of string up to that row for each character.For instance,

	A	B	N
A	0	0	0
N	1	0	0
N	1	0	1
B	1	0	2
$	1	1	2
A	1	1	2
A	2	1	2
	3	1	2

This will take n.∑ .logn space where ∑ is the alphabet size.

However,this takes a lot of space if we do this for human genome. Instead of keeping all of them,we keep some of them as in bucket sort.Assume bucket size is b.Then number of buckets is n/b and we will need
(n.∑ .logn)/b space.For instance,we keep only first,fourth row and eighth rows' values.

If we compare space and running time properties of

1-Searching in a suffix array,
2-Searching in BWT without buckets
3-Searching in BWT with buckets

	Space	Running Time
Suffix Array	nlogn	P+logn
BWT without bucket	n.∑ .logn	P
BWT with bucket	(n.∑ .logn)/b	P.b

where n is the text length and P is the length of the pattern we search.

There is also sparse suffix arrays which only store some of the suffixes in a way similar to the bucket idea.

Bowtie does not implement compression.This can be added to it.

ALIGNMENT REVIEW

```
1-  AGAC_ _ _ TT     3gaps+1substitution
     AGACTGTCT

2-   AGACT_ T _ _     3 gaps, no substitution
     AGACTGTCT
```

Which one is better in biological sense?

In biology,gaps are rare but once there is one,it is highly possible that a gap is followed by other gaps.For example in DNA,there are exons which are active in MRna synthesis and introns which are not.DNA is generally made up of

exons-introns-exons-intron-exons

If we try to match MRna's to regions in DNA,introns will correspond to gaps in this matching since they are not active in protein synthesis.

Therefore,a model which lets consecutive 3 gaps(3 gaps in a row) as in example 1 above makes more sense than the second one.Therefore,classical dynamic programming should be modified to handle that case.

Affine gap penalty = $g_{open} + kg_{extend}$

where = $g_{open} > g_{extend}$

To solve this problem,we will need 3 more arrays.

E(i,j) is score of best alignment of s1[1..i] and s2[1…j] ending with a gap in s1.j'th character of s2 is aligned with a gap in s1.

F(i,j) is same as E(i,j) except gap is now in s2.

G(i,j) is the one in with there is no gap so i'th character of s1 is aligned with j'th character of s2.

$V(i,j)=\max(E(i,j),F(i,j),G(i,j))$

and

$E(i,j)=\max\ ((E(i,j-1)+ g_{extend}), (V(i,j-1)+ g_{open} + g_{extend}))$

$F(i,j)=\max\ ((F(i-1,j)+ g_{extend}), (V(i-1,j)+ g_{open} + g_{extend}))$

$G(i,j)=V(i-1,j-1)+Score(s1(i),s2(j))$

G(i,j) can be merged into V so total number of tables will decrease to 3.

As you noticed,formulas are defined for max so g_{open} and g_{extend} are both negative.However,the formula can easily be changed to a form that is for min.

In classical inexact alignment formulation,we can depending it came from (i-1,j),(i,j-1) or (i-1,j-1).Here we can understand this by checking the E(i,j),F(i,j),G(i,j) values.For instance if V(i,j)=E(i,j) then there does exists a gap in s1.