FEMTO: Fast Search of Large Sequence Collections

Michael Ferguson

CPM 2012

FEMTO?

FM-index for
External
Memory with
Throughput
Optimizations

- an FM-index* with minor theoretical improvements
- indexing and search software for large collections
- assume not enough RAM to hold data or index

*FM-index from Ferragina and Manzini, 2005

Outline

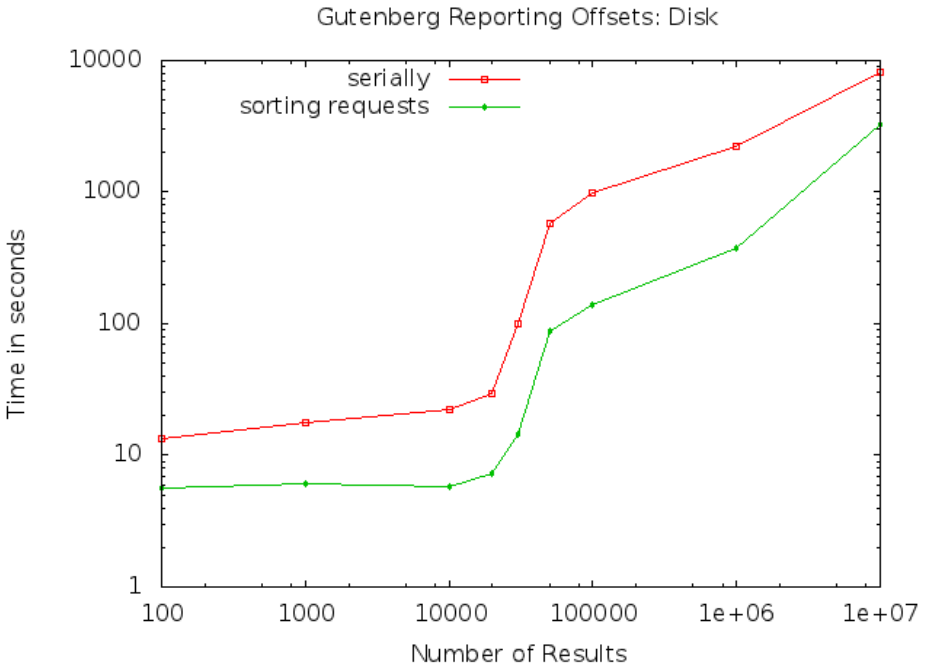
- Improvements to the FM-index for Disks
- || External Memory Suffix Array Construction
- Large Index Experiments
- Regular Expression search in an FM-index

Some Improvements to FM-index for external memory

(more in paper)

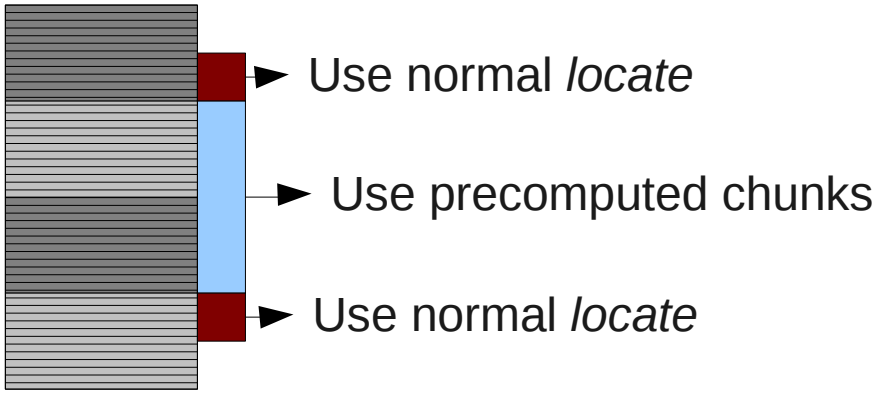
Improving Disk Throughput

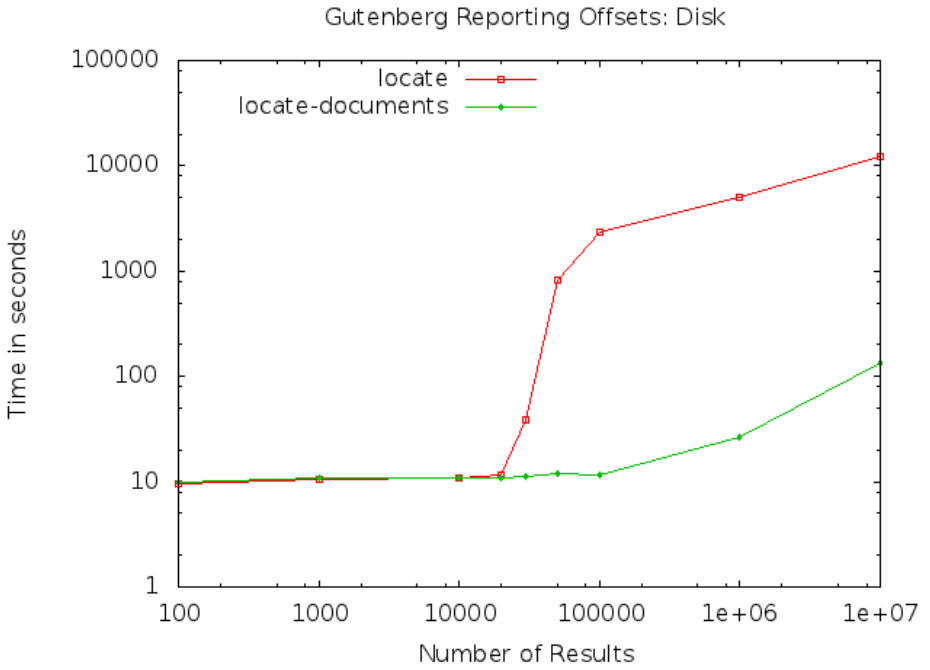
- Represent FM-index queries, such as Occ(L[i],i) as a *block request*
- Service *block requests* in sorted order
- FEMTO keeps these in a tree structure



Improving Document Search

- Divide index into chunks of *h* rows
- For each chunk, store a compressed list of matching document numbers
- Use these lists to speed up requests for documents matching many rows

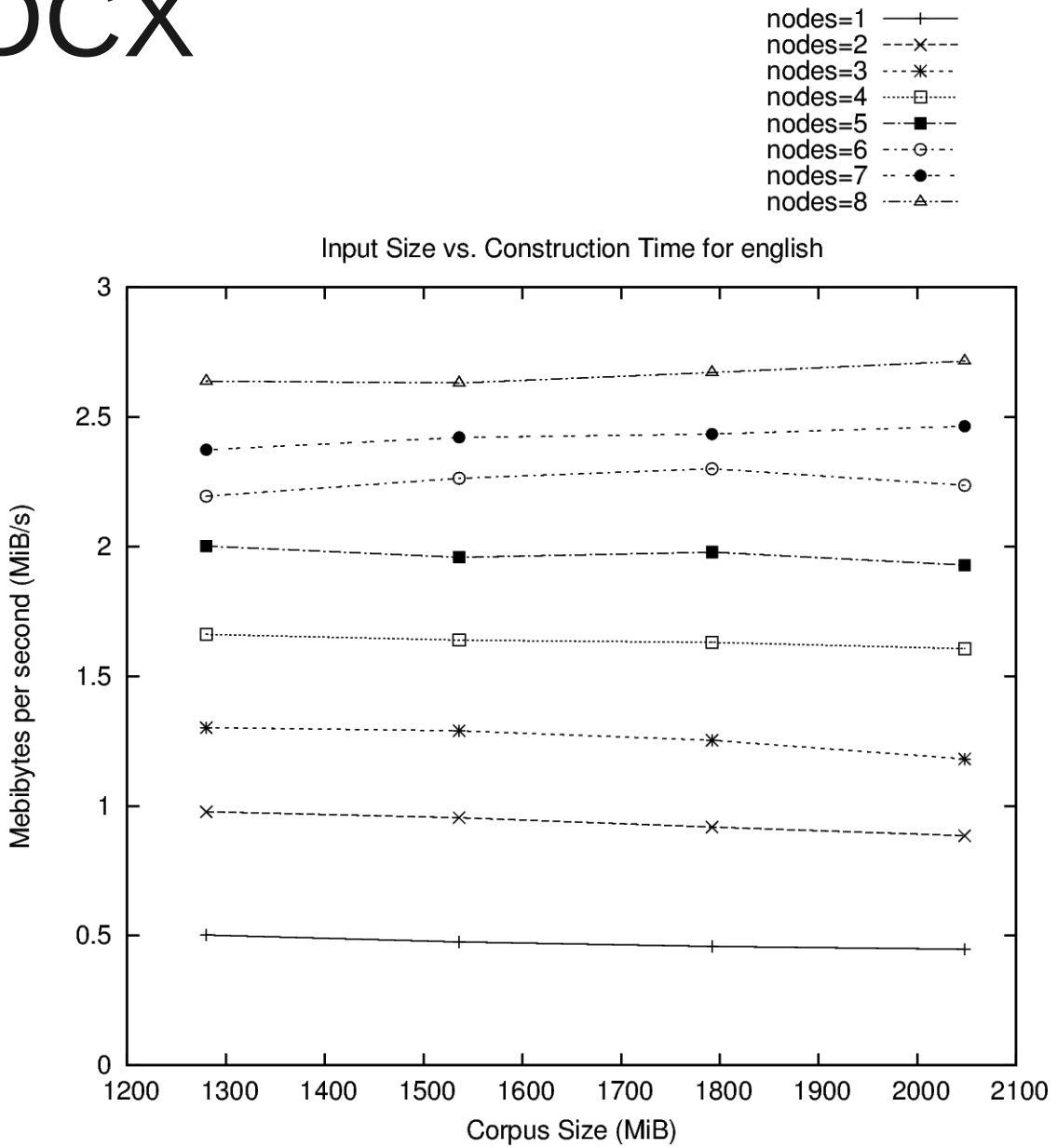




Parallel, External Memory Suffix Array Construction

DCX

- Difference Cover* Algorithm with varying parameter
- Parameter 7 or 13 work well for external memory
- Parallel implementation with C++ and MPI
- Observed *O(n)* running time and some parallel speedup



*Difference Cover algorithm from Kärkkäinen, Sanders, Burkhardt 2006

Large Index Experiments

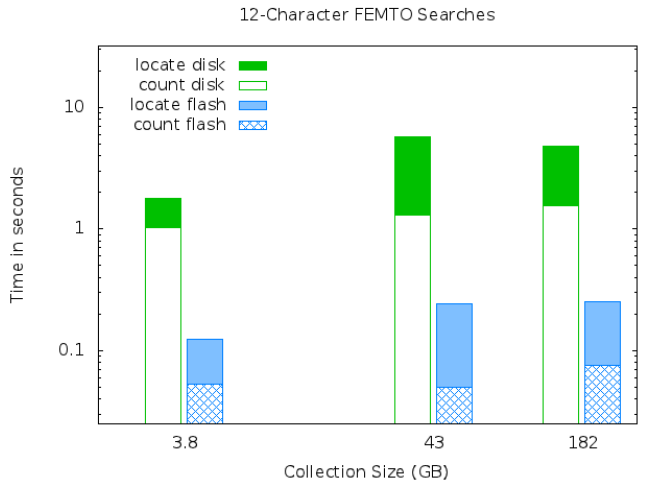
Collection and Index Information

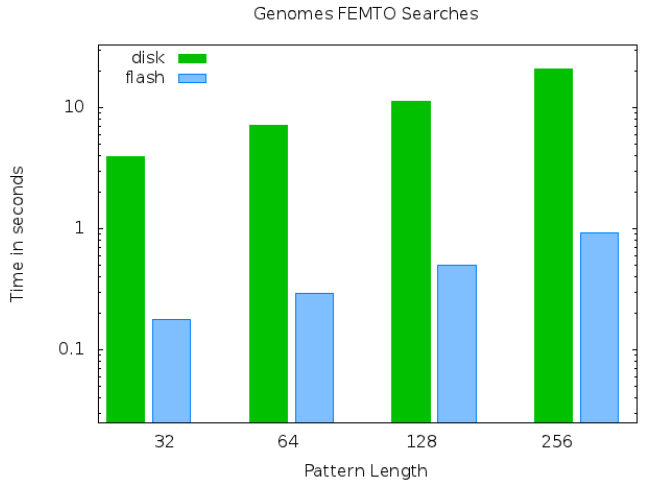
- Lucene is about 20x faster at building an index
- ... but an FM-index is more powerful
- Genomes is all sequence files from ftp.ncbi.nih.gov/genomes
- 182 GB is largest FM-index we know of

		Build Time	Index size
Bacteria – 3.76 GB – 1896 docs			
	Bowtie2	16940 s	5.26 GB
	FEMTO	7572 s	2.34 GB
Gutenberg – 42.93 GB – 94471 docs			
	Lucene	5133 s	14.84 GB
	FEMTO	108025 s	40.87 GB
Genomes – 182.43 GB – 23242528 docs			
	FEMTO	158434 s (7 nodes)	296.68 GB

Searching Large Collections

- Experiments were with index not yet in main memory
- Search times in seconds
- ... even for collections much larger than main memory





12-character searches return 10 results; longer patterns return 1 or 2

Regular Expression Search for FM-indexes

Regexp Search: the Idea

- Maintain a data structure mapping ranges of rows to sets of active states in an NFA
- Pop a mapping from the data structure
- Add new mappings by computing the new range of rows for characters with transitions from an active NFA state

Regexp Search: Pseudocode

simulate_nfa :
  add the entry ([0, n-1] -> the set of start states) to mapping
  while(the mapping is not empty) :
    pop an entry ([first, last] -> nfa_states) from the mapping
    for every character ch reachable from nfa_states :
      new_first = C[ch] + Occ(ch, first - 1)
      new_last = C[ch] + Occ(ch, last) - 1
      new_states=states reachable from nfa_states after reading ch
      if new_first <= *new_last,* add
        ([new_first, new_last] -> new_states) to the mapping,
        reporting a match if a final state is set

Example FM-index

C[$]=0
C[i]=1
C[m]=5
C[p]=6
C[s]=8

row	suffix	L
0	$mississippi	i
1	i$mississipp	p
2	ippi$mississ	s
3	issippi$miss	s
4	ississippi$m	m
5	mississippi$	$
6	pi$mississip	p
7	ppi$mississi	i
8	sippi$missis	s
9	sissippi$mis	s
10	ssippi$missi	i
11	ssissippi$mi	i

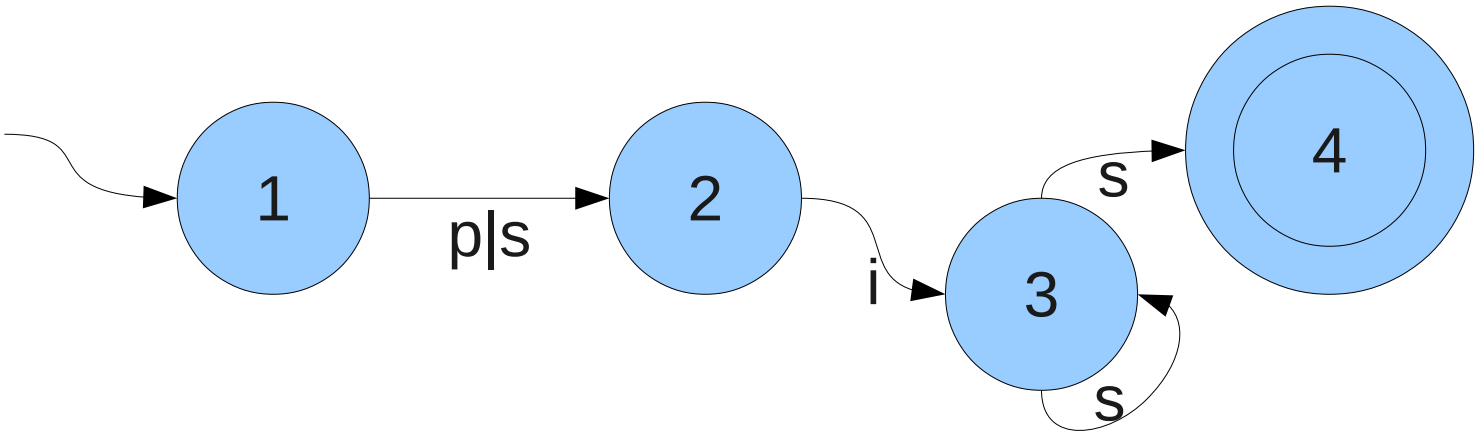
- FM-index of mississippi$
- C[x] = # times characters < x occur in input
- L column is BWT of input
- Occs(x,r) = # times character x occurs in L column at or before row r

Example Regexp

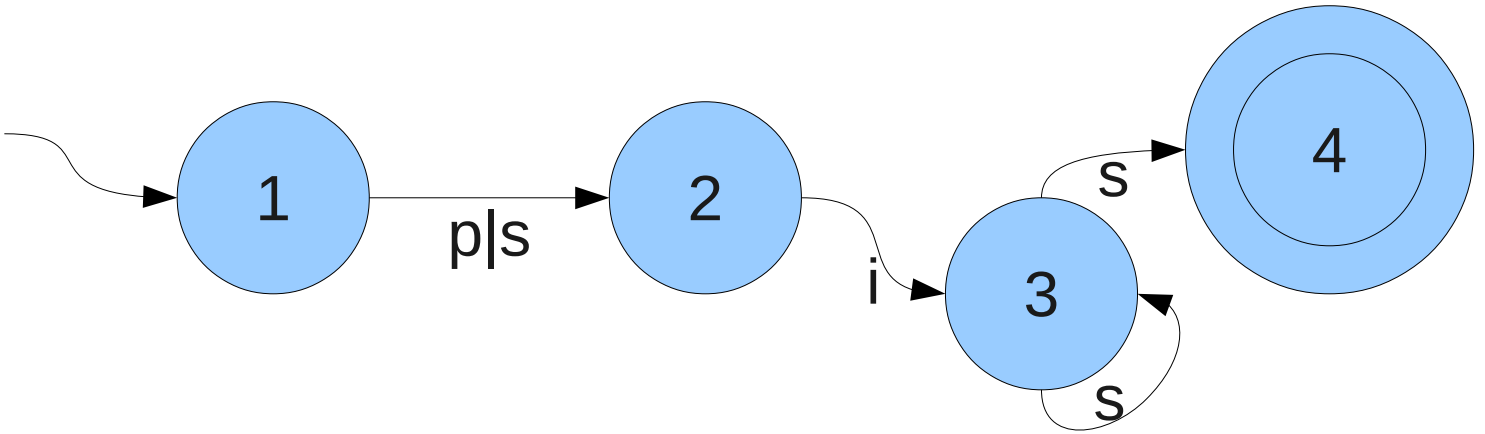
C[$]=0
C[i]=1
C[m]=5
C[p]=6
C[s]=8

row	suffix	L
0	$mississippi	i
1	i$mississipp	p
2	ippi$mississ	s
3	issippi$miss	s
4	ississippi$m	m
5	mississippi$	$
6	pi$mississip	p
7	ppi$mississi	i
8	sippi$missis	s
9	sissippi$mis	s
10	ssippi$missi	i
11	ssissippi$mi	i

- We will search for ss*i(p|s)
- First, reverse the regular expression: (p|s)is*s
- Then, compute the NFA
- 1 start state, 4 final state



C[$]=0
C[i]=1
C[m]=5
C[p]=6
C[s]=8

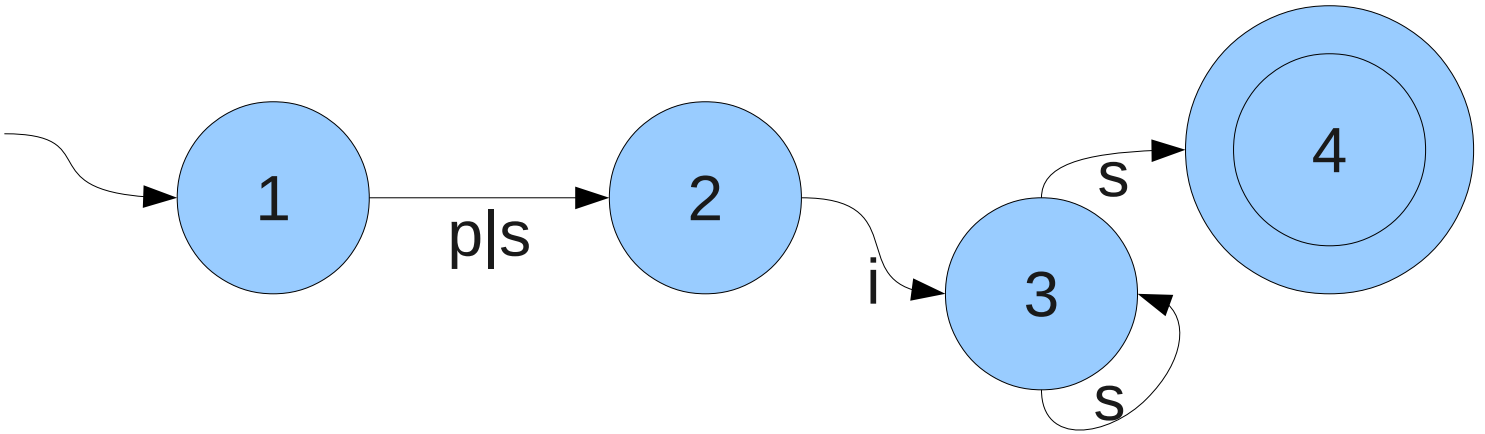


row	suffix	L	
0	$mississippi	i	
1	i$mississipp	p	
2	ippi$mississ	s	
3	issippi$miss	s	
4	ississippi$m	m	
5	mississippi$	$	{1}
6	pi$mississip	p	
7	ppi$mississi	i	
8	sippi$missis	s	
9	sissippi$mis	s	
10	ssippi$missi	i	
11	ssissippi$mi	i	

add the entry ([0, n-1] -> the set of start states) to mapping:

`[0,11]->{1}`

C[$]=0
C[i]=1
C[m]=5
C[p]=6
C[s]=8



row	suffix	L	
0	$mississippi	i	
1	i$mississipp	p	
2	ippi$mississ	s	
3	issippi$miss	s	
4	ississippi$m	m	
5	mississippi$	$	{1}
6	pi$mississip	p	{2}
7	ppi$mississi	i	
8	sippi$missis	s	{2}
9	sissippi$mis	s	
10	ssippi$missi	i	
11	ssissippi$mi	i	

Pop `[0,11]->{1}`

consider p

`[C[p]+Occ(p,-1),C[p]+Occ(p,11)-1]->{2}`

`[6,7]->{2}`

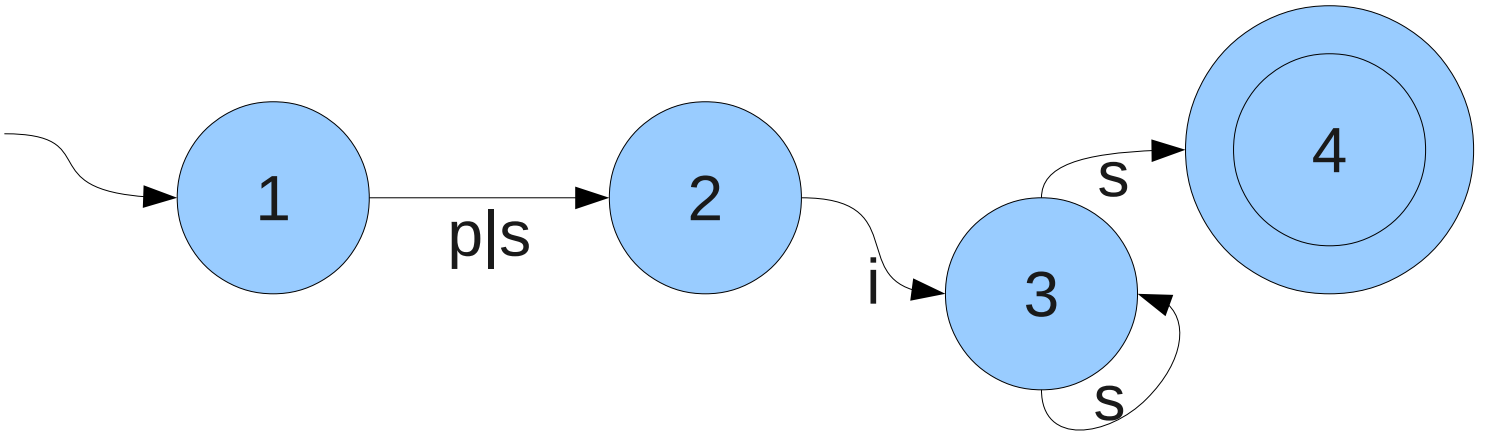
consider s

`[C[s]+Occ(s,-1),C[s]+Occ(s,11)-1]->{2}`

`[8,11]->{2}`

new range =
[C[ch] + Occ(ch, first-1),C[ch] + Occ(ch, last)-1]

C[$]=0
C[i]=1
C[m]=5
C[p]=6
C[s]=8



row	suffix	L
0	$mississippi	i
1	i$mississipp	p
2	ippi$mississ	s
3	issippi$miss	s
4	ississippi$m	m
5	mississippi$	$
6	pi$mississip	p
7	ppi$mississi	i
8	sippi$missis	s
9	sissippi$mis	s
10	ssippi$missi	i
11	ssissippi$mi	i

{3}

{2}

{2}

Pop [6,7]->{2}

consider i

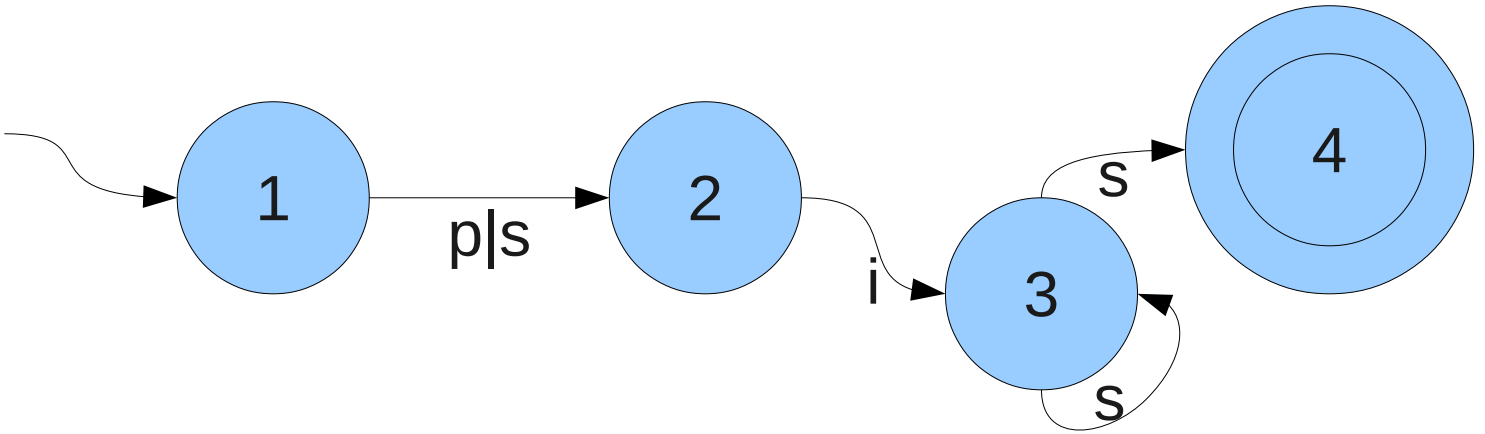
[C[i]+Occ(i,5),C[i]+Occ(i,7)-1]->{3}

[2,2]->{3}

new range =

[C[ch] + Occ(ch, first-1),C[ch] + Occ(ch, last)-1]

C[$]=0
C[i]=1
C[m]=5
C[p]=6
C[s]=8



row	suffix	L
0	$mississippi	i
1	i$mississipp	p
2	ippi$mississ	s
3	issippi$miss	s
4	ississippi$m	m
5	mississippi$	$
6	pi$mississip	p
7	ppi$mississi	i
8	sippi$missis	s
9	sissippi$mis	s
10	ssippi$missi	i
11	ssissippi$mi	i

{3}

{3,4}

{2}

Pop [2,2]->{3}

consider s

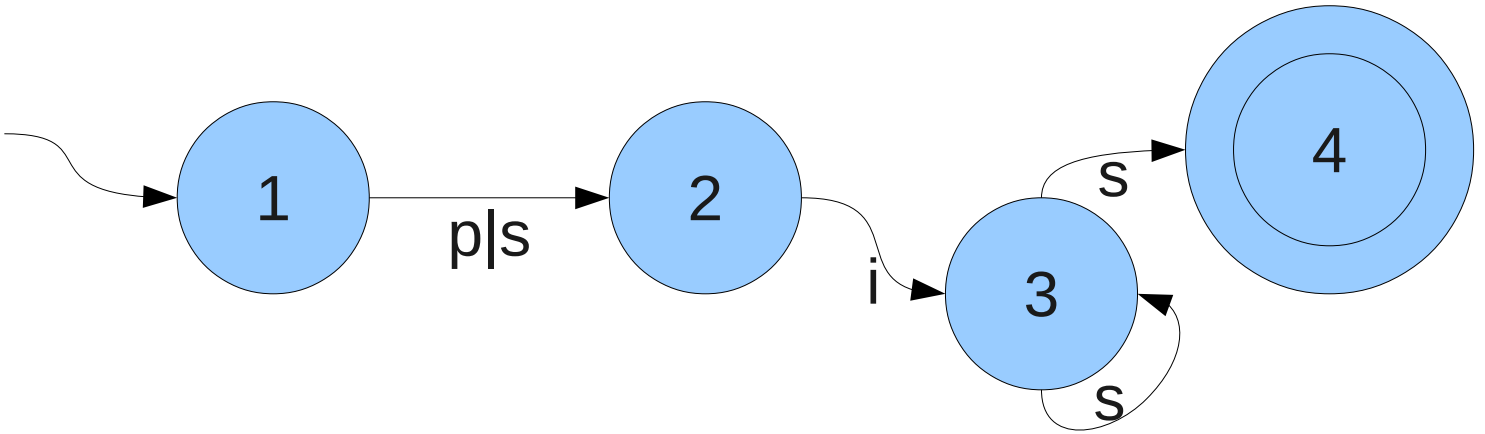
[C[s]+Occ(s,1),C[s]+Occ(s,2)-1]->{3,4}

[8,8]->{3,4}

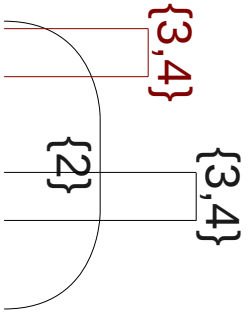
report [8,8]

Results: [8,8]

C[$]=0
C[i]=1
C[m]=5
C[p]=6
C[s]=8



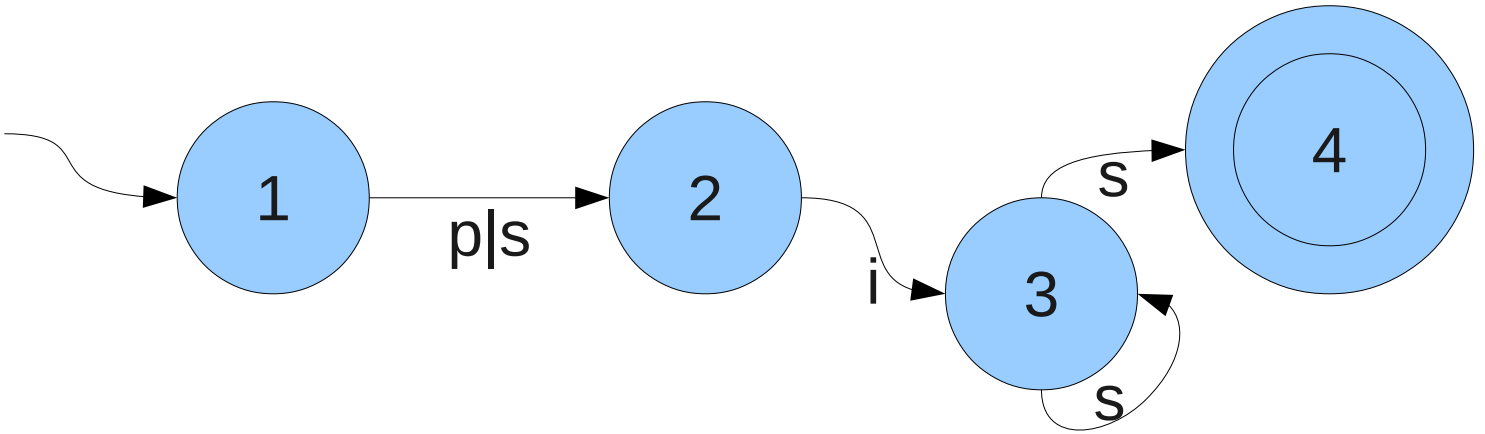
row	suffix	L
0	$mississippi	i
1	i$mississipp	p
2	ippi$mississ	s
3	issippi$miss	s
4	ississippi$m	m
5	mississippi$	$
6	pi$mississip	p
7	ppi$mississi	i
8	sippi$missis	s
9	sissippi$mis	s
10	ssippi$missi	i
11	ssissippi$mi	i



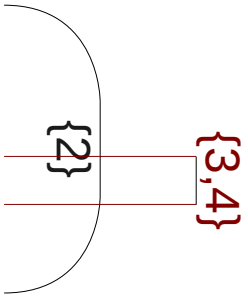
Pop [8,8]->{3,4}
consider s
[C[s]+Occ(s,7),C[s]+Occ(s,8)-1]->{3,4}
[10,10]->{3,4}
report [10,10]

Results: [8,8] [10,10]

C[$]=0
C[i]=1
C[m]=5
C[p]=6
C[s]=8



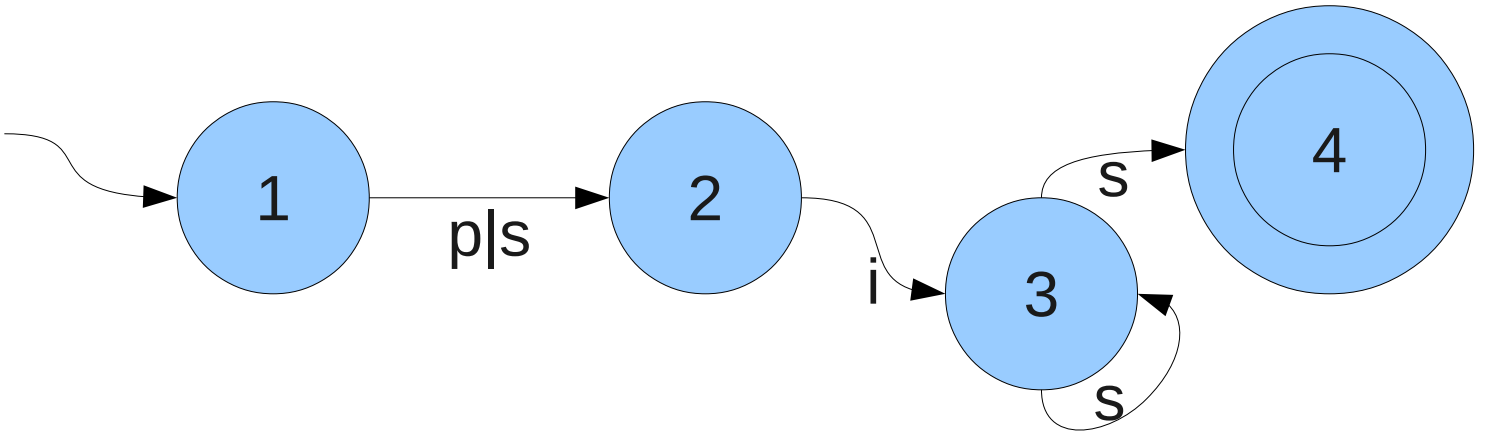
row	suffix	L
0	$mississippi	i
1	i$mississipp	p
2	ippi$mississ	s
3	issippi$miss	s
4	ississippi$m	m
5	mississippi$	$
6	pi$mississip	p
7	ppi$mississi	i
8	sippi$missis	s
9	sissippi$mis	s
10	ssippi$missi	i
11	ssissippi$mi	i



Pop `[10,10]->{3,4}`
consider s
`[C[s]+Occ(s,9),C[s]+Occ(s,10)-1]->{3,4}`
`[12,11]->{3,4}`
invalid range - do not add to mapping or report a result

Results: [8,8] [10,10]

C[$]=0
C[i]=1
C[m]=5
C[p]=6
C[s]=8



row	suffix	L
0	$mississippi	i
1	i$mississipp	p
2	ippi$mississ	s
3	issippi$miss	s
4	ississippi$m	m
5	mississippi$	$
6	pi$mississip	p
7	ppi$mississi	i
8	sippi$missis	s
9	sissippi$mis	s
10	ssippi$missi	i
11	ssissippi$mi	i

{3}

{2}

Pop [8,11]->{2}

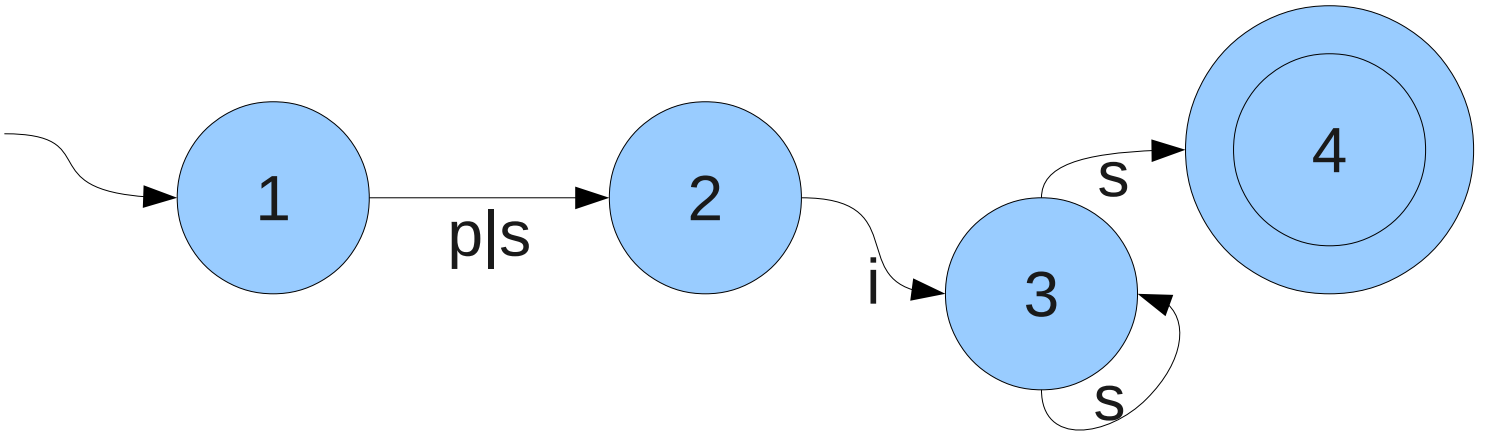
consider i

[C[i]+Occ(i,7),C[i]+Occ(i,11)-1]->{3}

[3,4]->{3}

Results: [8,8] [10,10]

C[$]=0
C[i]=1
C[m]=5
C[p]=6
C[s]=8



row	suffix	L
0	$mississippi	i
1	i$mississipp	p
2	ippi$mississ	s
3	issippi$miss	s
4	ississippi$m	m
5	mississippi$	$
6	pi$mississip	p
7	ppi$mississi	i
8	sippi$missis	s
9	sissippi$mis	s
10	ssippi$missi	i
11	ssissippi$mi	i

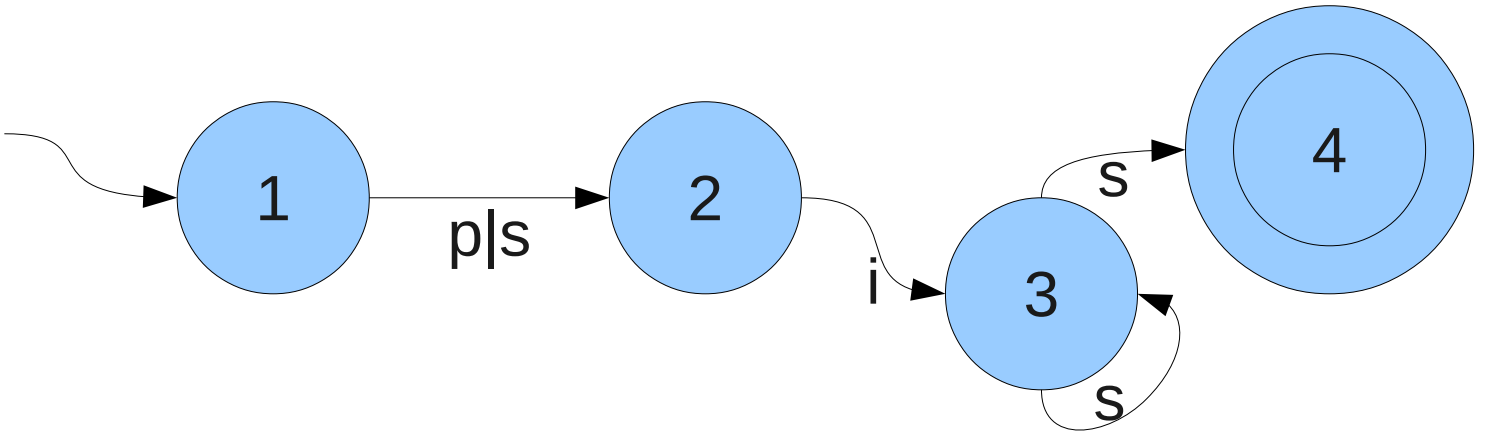
{3}

{3,4}

Pop [3,4]->{3}
consider s
[C[s]+Occ(s,2),C[s]+Occ(s,4)-1]->{3,4}
[9,9]->{3,4}
report [9,9]

Results: [8,8] [10,10] [9,9]

C[$]=0
C[i]=1
C[m]=5
C[p]=6
C[s]=8



row	suffix	L
0	$mississippi	i
1	i$mississipp	p
2	ippi$mississ	s
3	issippi$miss	s
4	ississippi$m	m
5	mississippi$	$
6	pi$mississip	p
7	ppi$mississi	i
8	sippi$missis	s
9	sissippi$mis	s
10	ssippi$missi	i
11	ssissippi$mi	i

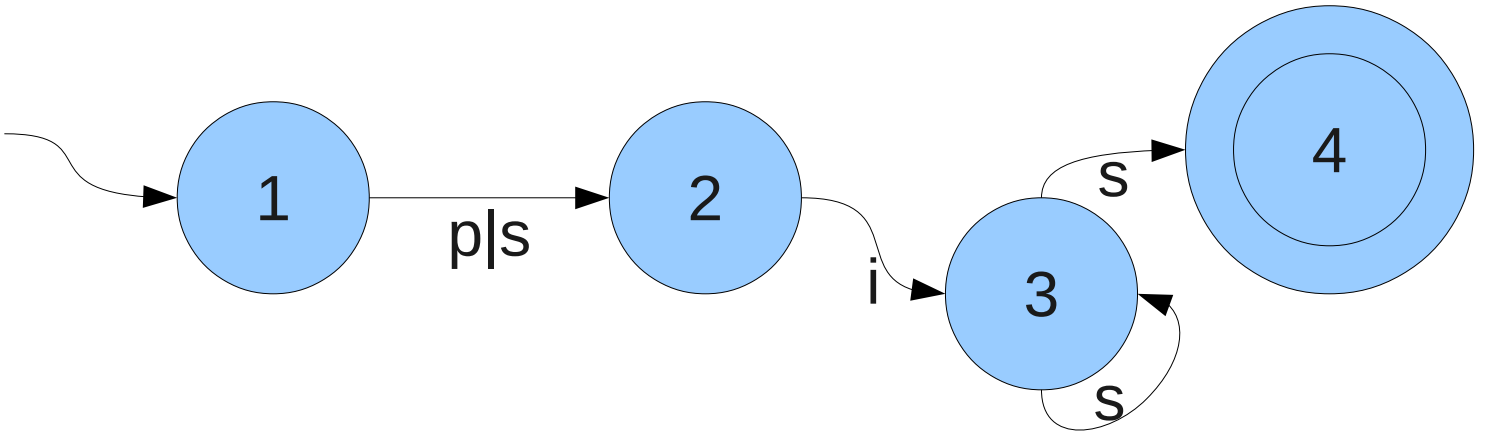
{3,4}

{3,4}

Pop [9,9]->{3,4}
consider s
[C[s]+Occ(s,8),C[s]+Occ(s,9)-1]->{3,4}
[11,11]->{3,4}
report [11,11]

Results: [8,8] [10,10] [9,9] [11,11]

C[$]=0
C[i]=1
C[m]=5
C[p]=6
C[s]=8



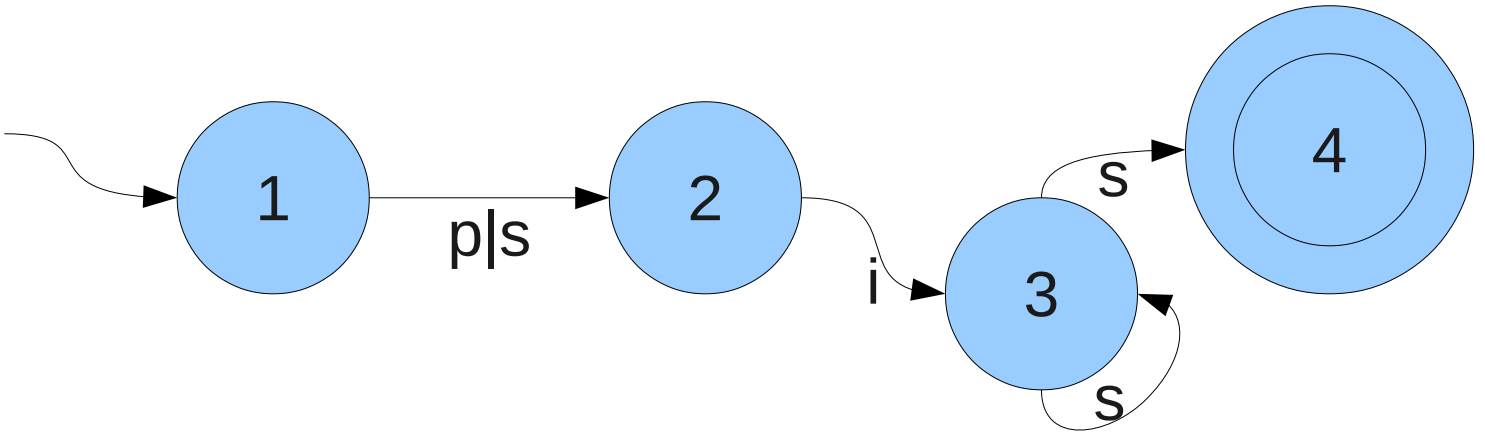
row	suffix	L
0	$mississippi	i
1	i$mississipp	p
2	ippi$mississ	s
3	issippi$miss	s
4	ississippi$m	m
5	mississippi$	$
6	pi$mississip	p
7	ppi$mississi	i
8	sippi$missis	s
9	sissippi$mis	s
10	ssippi$missi	i
11	ssissippi$mi	i

{3,4}

Pop [11,11]->{3,4}
consider s
[C[s]+Occ(s,10),C[s]+Occ(s,11)-1]->{3,4}
[12,11]->{3,4}
invalid range - do not add to mapping or report a result

Results: [8,8] [10,10] [9,9] [11,11]

C[$]=0
C[i]=1
C[m]=5
C[p]=6
C[s]=8



row	suffix	L
0	$mississippi	i
1	i$mississipp	p
2	ippi$mississ	s
3	issippi$miss	s
4	ississippi$m	m
5	mississippi$	$
6	pi$mississip	p
7	ppi$mississi	i
8	sippi$missis	s
9	sissippi$mis	s
10	ssippi$missi	i
11	ssissippi$mi	i

Results: [8,8] [10,10] [9,9] [11,11]

So matches for
ss*i(p|s)
are rows 8,9,10,11:
sip
sis
ssip
ssis

FEMTO Availability

- Not yet released
- Looking for partners to improve, maintain, and release FEMTO
- Please contact me (mferguson at ltsnet.net) if you are interested

FEMTO Demonstrates

- large FM-index scaling
- techniques for improving external memory FM-index
- regular expression search over FM-index

Future Work

- Faster, more scalable suffix array construction
- Is it practical to add to an existing index?
- Complexity analysis of regular expression search