Branch Prediction Analysis of Morris-Pratt and Knuth-Morris-Pratt Algorithms

Carine Pivoteau

with Cyril Nicaud and Stéphane Vialette

LIGM - Université Gustave Eiffel

CPM - June 2025

Model for analysis of algorithms

Classical (theoretical)

- Worst case
- Unit cost operations (comparisons, accesses, arithmetic, ...)
- Sequential execution

Model for analysis of algorithms

Classical (theoretical) vs. Reality (execution)

- Worst case
- Unit cost operations (comparisons, accesses, arithmetic, ...)
- Sequential execution

Model for analysis of algorithms

Classical (theoretical) vs. Reality (execution)

- Worst case

  ▷ Average case can be quite different from the worst, it depends a lot on the data

- Unit cost operations (comparisons, accesses, arithmetic, ...)

- Sequential execution

Model for analysis of algorithms

Classical (theoretical) vs. Reality (execution)

- Worst case

  ▷ Average case can be quite different from the worst, it depends a lot on the data

- Unit cost operations (comparisons, accesses, arithmetic, ...)

  ▷ The latency of an operation depends heavily on the processor (ALU, cache, ...)

- Sequential execution

Model for analysis of algorithms

Classical (theoretical) vs. Reality (execution)

- Worst case

  ▷ Average case can be quite different from the worst, it depends a lot on the data

- Unit cost operations (comparisons, accesses, arithmetic, ...)

  ▷ The latency of an operation depends heavily on the processor (ALU, cache, ...)

- Sequential execution

  ▷ Instructions are parallelized in different ways (pipeline, SIMD, multi-core processors...)

In general, only the first point can change the complexity, but the last two points can have an big impact on execution time.

Pipeline, hazards, and branch prediction

- (Modern) processors use a pipeline to create instruction-level parallelism.
- Various hazards can stall a pipeline. For instance branching instructions (*e.g.*, `if`, `if-then-else`, `while`, ...) may need the previous one to finish its execution.
- Solution : anticipating the outcome of a branch instruction by using a predictor.

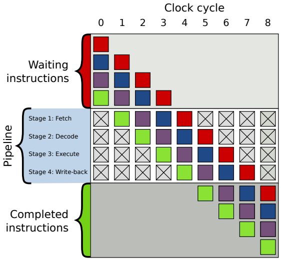


Illustration: Wikipedia

Computer Architecture: A Quantitative Approach (5th ed.), Hennessy & Patterson

Pipeline, hazards, and branch prediction

- (Modern) processors use a pipeline to create instruction-level parallelism.
- Various hazards can stall a pipeline. For instance branching instructions (*e.g.*, `if`, `if-then-else`, `while`, ...) may need the previous one to finish its execution.
- Solution : anticipating the outcome of a branch instruction by using a predictor.

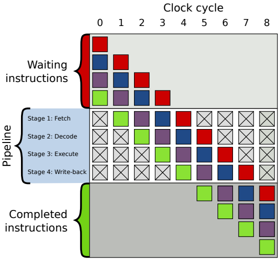


Illustration: Wikipedia

A **dynamic** branch predictor uses runtime information (specifically, the branch execution history) for accuracy.

Computer Architecture: A Quantitative Approach (5th ed.), Hennessy & Patterson

Pipeline, hazards, and branch prediction

- (Modern) processors use a pipeline to create instruction-level parallelism.
- Various hazards can stall a pipeline. For instance branching instructions (*e.g.*, `if`, `if-then-else`, `while`, ...) may need the previous one to finish its execution.
- Solution : anticipating the outcome of a branch instruction by using a predictor.

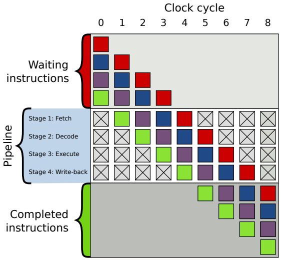


Illustration: Wikipedia

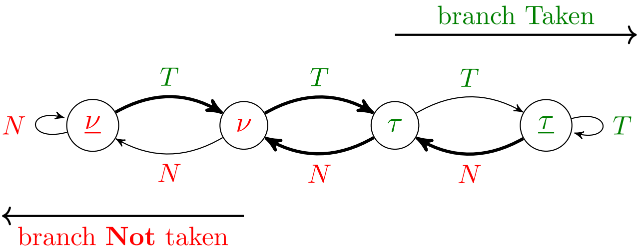
A **dynamic** branch predictor uses runtime information (specifically, the branch execution history) for accuracy.

- **local**: independent history for each conditional jump $\Longrightarrow$ predictions based on the behavior of that specific instruction
- **global**: shared history of all jumps $\Longrightarrow$ capture correlations and improve prediction accuracy
- Since the 2000s, processors use a combination of local and global

Computer Architecture: A Quantitative Approach (5th ed.), Hennessy & Patterson

Our model: classical local branch predictor

2-bit saturated counter:



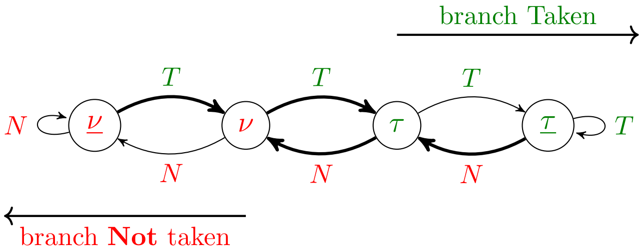
State = prediction:

- $\underline{\nu}$ and ν predicts not taken
- $\underline{\tau}$ and $\underline{\tau}$ predicts taken

Transition = actual outcome: the branch is **T**aken or **N**ot taken

Our model: classical local branch predictor

2-bit saturated counter:



State = prediction:

- $\underline{\nu}$ and ν predicts not taken
- $\underline{\tau}$ and $\underline{\tau}$ predicts taken

Transition = actual outcome: the branch is **T**aken or **N**ot taken

▷ How well does it work during the execution of an algorithm?

Past and present work

- Brodal & Moruz, 2005 : mispredictions and (adaptive) sorting

Tradeoffs Between Branch Mispredictions and Comparisons for Sorting Algorithms

Gerth Stølting Brodal[1,*] and Gabriel Moruz[1]

BRICS[**], Department of Computer Science, University of Aarhus,
IT Parken, Åbogade 34, DK-8200 Århus N, Denmark
{gerth, gabi}@daimi.au.dk

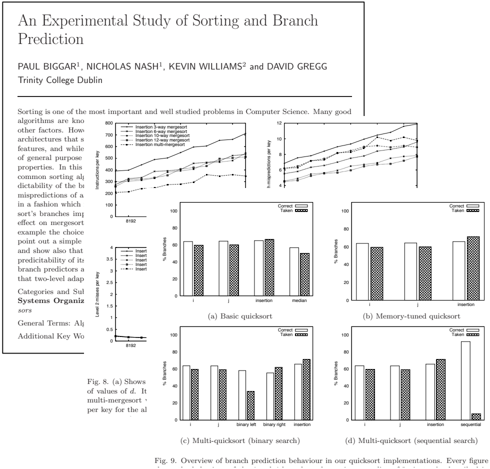
Abstract. Branch mispredictions is an important factor affecting the running time in practice. In this paper we consider tradeoffs between the number of branch mispredictions and the number of comparisons for sorting algorith mispred by adop tions. F rithm p $\Omega(n \log$ of inver by Esti col and mispred

Measure	Comparisons	Branch mispredictions
Dis	$O(dn(1+\log(1+\text{Dis})))$	$\Omega(n\log_d(1+\text{Dis}))$
Exc	$O(dn(1+\text{Exc}\log(1+\text{Exc})))$	$\Omega(n\text{Exc}\log_d(1+\text{Exc}))$
Enc	$O(dn(1+\log(1+\text{Enc})))$	$\Omega(n\log_d(1+\text{Enc}))$
Inv	$O(dn(1+\log(1+\text{Inv}/n)))$	$\Omega(n\log_d(1+\text{Inv}/n))$
Max	$O(dn(1+\log(1+\text{Max})))$	$\Omega(n\log_d(1+\text{Max}))$
Osc	$O(dn(1+\log(1+\text{Osc}/n)))$	$\Omega(n\log_d(1+\text{Osc}/n))$
Reg	$O(dn(1+\log(1+\text{Reg})))$	$\Omega(n\log_d(1+\text{Reg}))$
Rem	$O(dn(1+\text{Rem}\log(1+\text{Rem})))$	$\Omega(n\text{Rem}\log_d(1+\text{Rem}))$
Runs	$O(dn(1+\log(1+\text{Runs})))$	$\Omega(n\log_d(1+\text{Runs}))$
SMS	$O(dn(1+\log(1+\text{SMS})))$	$\Omega(n\log_d(1+\text{SMS}))$
SUS	$O(dn(1+\log(1+\text{SUS})))$	$\Omega(n\log_d(1+\text{SUS}))$

Fig. 4. Lower bounds on the number of branch mispredictions for deterministic comparison based adaptive sorting algorithms for different measures of presortedness, given the upper bounds on the number of comparisons

Past and present work

- Brodal & Moruz, 2005 : mispredictions and (adaptive) sorting
- Biggar *et al*, 2008 : experimental, branch prediction and sorting



Past and present work

- Brodal & Moruz, 2005 : mispredictions and (adaptive) sorting
- Biggar *et al*, 2008 : experimental, branch prediction and sorting
- Kaligosi and Sanders, 2006 : mispredictions and quicksort

How Branch Mispredictions Affect Quicksort

Kanela Kaligosi[1] and Peter Sanders[2]

[1] Max Planck Ir
Saarbrüc
kaligosi@
[2] Universität I
sanders

Abstract. We explain the cou
"good" pivots (close to the medi
not impro
pivot *imp*
count dec
direction
fect of sin
hardware.

Table 1. Number of branch mispredictions

	random pivot	α-skewed pivot
static predictor	$\frac{\ln 2}{2} n \lg n + \mathcal{O}(n)$, $\frac{\ln 2}{2} \approx 0.3466$	$\frac{\alpha}{H(\alpha)} n \lg n + \mathcal{O}(n)$, $\alpha < 1/2$ $\frac{1-\alpha}{H(\alpha)} n \lg n + \mathcal{O}(n)$, $\alpha \geq 1/2$
1-bit predictor	$\frac{2\ln 2}{3} n \lg n + \mathcal{O}(n)$, $\frac{2\ln 2}{3} \approx 0.4621$	$\frac{2\alpha(1-\alpha)}{H(\alpha)} n \lg n + \mathcal{O}(n)$
2-bit predictor	$\frac{28\ln 2}{45} n \lg n + \mathcal{O}(n)$, $\frac{28\ln 2}{45} \approx 0.4313$	$\frac{2\alpha^4 - 4\alpha^3 + \alpha^2 + \alpha}{(1-\alpha(1-\alpha))H(\alpha)} n \lg n + \mathcal{O}(n)$

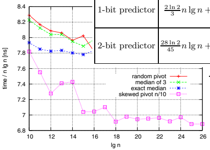


Fig. 3. Time / $n \lg n$ for random pivot, median of 3, exact median, 1/10-skewed pivot

Past and present work

- Brodal & Moruz, 2005 : mispredictions and (adaptive) sorting
- Biggar *et al*, 2008 : experimental, branch prediction and sorting
- Kaligosi and Sanders, 2006 : mispredictions and quicksort
- Martínez, Nebel and Wild, 2014 : mispredictions and quicksort

Analysis of Branch Misses in Quicksort*

Conrado Martínez[†] Markus E. Nebel[‡§] Sebastian Wild[‡]

November 11, 2014

Abstract

The analysis of algorithms mostly relies on counting classic elementary operations like additions, multiplications, comparisons, swaps etc. This approach is of
efficiency. I
ern process
and memor
running tin
get a reliab
sort: It has
under certa
sically opti
dian of a s
mispredicte
dominating

pivots are chosen from a sample of the input. We conclude that the difference in branch misses is too small to explain the superiority of the dual-pivot algorithm.

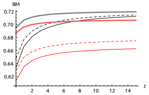


Figure 5: Branch mispredictions, as a function of t, in CQS (black) and YQS (red) with 1-bit branch prediction (fat), 2-bit saturating counter (thin solid) and 2-bit flip-consecutive (dashed) using symmetric sampling: $\mathbf{t}_{\text{CQS}} = (3t+2, 3t+2)$ and $\mathbf{t}_{\text{YQS}} = (2t+1, 2t+1, 2t+1)$

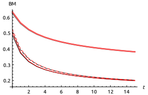


Figure 6: Branch mispredictions, as a function of t, in CQS (black) and YQS (red) with 1-bit (fat), 2-bit sc (thin solid) and 2-bit fc (dashed) predictors, using extremely skewed sampling: $\mathbf{t}_{\text{CQS}} = (0, 6t+4)$ and $\mathbf{t}_{\text{YQS}} = (0, 6t+3, 0)$

Past and present work

- Brodal & Moruz, 2005 : mispredictions and (adaptive) sorting
- Biggar *et al*, 2008 : experimental, branch prediction and sorting
- Kaligosi and Sanders, 2006 : mispredictions and quicksort
- Martínez, Nebel and Wild, 2014 : mispredictions and quicksort
- Auger, Nicaud, Pivoteau 2016 : average (trade-off) analysis for min/max, exponentiation and binary search

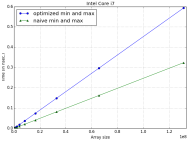
Good Predictions Are Worth a Few Comparisons

Nicolas Auger, Cyril Nicaud, and Carine Pivoteau

Université Paris-Est, LIGM (UMR 8049), F77454 Marne-la-Vallée, France

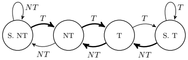
Abstract

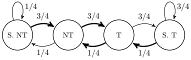
Most modern processors are heavily parallelized and use predictor
conditional branches, in order to avoid costly stalls in their pipelin
friendly versions of two classical algorithms: exponentiation by squa
sorted array. These variants result in less mispredictions on average,
number of operations. These theoretical results are supported by e
that our algorithms perform significantly better than the standard on



1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms a

Keywords and phrases branch misses, binary search, exponentiation by squaring, Markov chains





Past and present work

- Brodal & Moruz, 2005 : mispredictions and (adaptive) sorting
- Biggar *et al*, 2008 : experimental, branch prediction and sorting
- Kaligosi and Sanders, 2006 : mispredictions and quicksort
- Martínez, Nebel and Wild, 2014 : mispredictions and quicksort
- Auger, Nicaud, Pivoteau 2016 : average (trade-off) analysis for min/max, exponentiation and binary search

These algorithms heavily relies on branch instructions, but most of them are independent. What happens when they are correlated ?

Past and present work

- Brodal & Moruz, 2005 : mispredictions and (adaptive) sorting
- Biggar *et al*, 2008 : experimental, branch prediction and sorting
- Kaligosi and Sanders, 2006 : mispredictions and quicksort
- Martínez, Nebel and Wild, 2014 : mispredictions and quicksort
- Auger, Nicaud, Pivoteau 2016 : average (trade-off) analysis for min/max, exponentiation and binary search

These algorithms heavily relies on branch instructions, but most of them are independent. What happens when they are correlated ?

This happens in pattern matching algorithms, *e.g.*, **MP** and **KMP**.

- How can we study the **impact of branch prediction** on them ?
- Can we observe it on the **execution time** ?

Note: the classical average analysis of the number of comparisons for a random text and a random pattern was done by M. Regnier an W. Szpankowski in the 90s

W = text of length n , X = pattern of length m,

```
i, j, nb ← 0, 0, 0
while j < n do
    while i ≥ 0 and X[i] ≠ W[j] do
        i ← B[i]
    i, j ← i + 1, j + 1
    if i = m then
        i ← B[i]
        nb ← nb + 1
return nb
```

B = pre-computed border table of X:

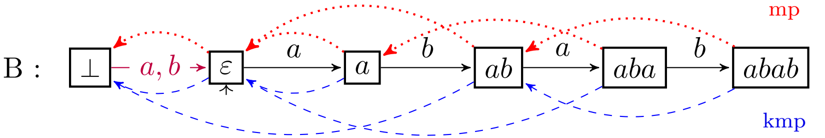
- mp : $B[i]$ = size of longest border u of $X[0..i-1]$
- kmp: $B[i]$ = size of longest border u of $X[0..i-1]$
  + u followed by $X[i]$ is not a prefix of X

W = text of length n , X = pattern of length m,

```
i, j, nb ← 0, 0, 0
while j < n do
    while i ≥ 0 and X[i] ≠ W[j] do
        i ← B[i]
    i, j ← i + 1, j + 1
    if i = m then
        i ← B[i]
        nb ← nb + 1
return nb
```

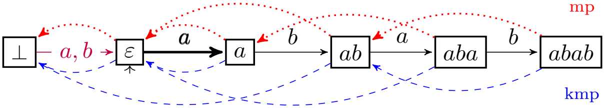
X = ababb

B =	-1	0	0	1	2	mp
B =	-1	0	-1	0	2	kmp



Branches in MP and KMP

while $j < n$ **do**
- **while** $i \geq 0$ **and** $X[i] \neq W[j]$ **do**
  - $i \leftarrow B[i]$
- $i, j \leftarrow i + 1, j + 1$
- **if** $i = m$ **then**
  - $i \leftarrow B[i]$
  - $nb \leftarrow nb + 1$



W

a	a	b	a	b	a	b	b	a	b	a	a	b	a	...

↑

X

a	b	a	b	b

while $j < n$ **do**

  while $i \geq 0$ **and** $X[i] \neq W[j]$ **do**

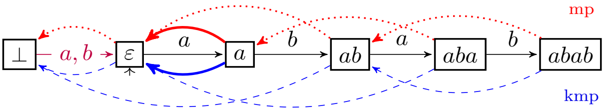
    $i \leftarrow B[i]$

  $i, j \leftarrow i + 1, j + 1$

  if $i = m$ **then**

    $i \leftarrow B[i]$

    $nb \leftarrow nb + 1$



W	a	a	b	a	b	a	b	b	a	b	a	a	b	a	...
		↑													
X	a	b	a	b	b										

Branches in MP and KMP

while $j < n$ **do**

  while $i \geq 0$ **and** $X[i] \neq W[j]$ **do**

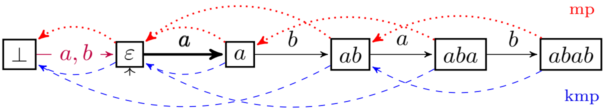
    $i \leftarrow B[i]$

  $i, j \leftarrow i + 1, j + 1$

  if $i = m$ **then**

    $i \leftarrow B[i]$

    $nb \leftarrow nb + 1$



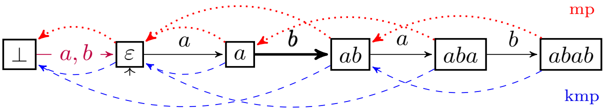
W	a	a	b	a	b	a	b	b	a	b	a	a	b	a	…

↑

X	a	b	a	b	b

Branches in MP and KMP

while $j < n$ **do**
- **while** $i \geq 0$ **and** $X[i] \neq W[j]$ **do**
  - $i \leftarrow B[i]$
- $i, j \leftarrow i + 1, j + 1$
- **if** $i = m$ **then**
  - $i \leftarrow B[i]$
  - $nb \leftarrow nb + 1$

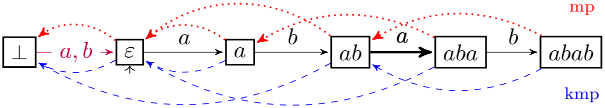


W															
	a	a	b	a	b	a	b	b	a	b	a	a	b	a	…

X						
		a	b	a	b	b

Branches in MP and KMP

while $j < n$ **do**
  while $i \geq 0$ **and** $X[i] \neq W[j]$ **do**
    $i \leftarrow B[i]$
  $i, j \leftarrow i + 1, j + 1$
  if $i = m$ **then**
    $i \leftarrow B[i]$
    $nb \leftarrow nb + 1$

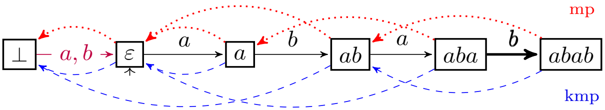


W	a	a	b	a	b	a	b	b	a	b	a	a	b	a	...

↑ (under the 4th cell of W)

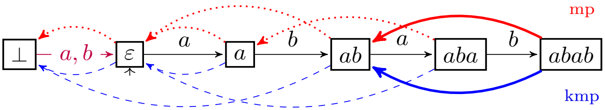
X	a	b	a	b	b

while $j < n$ **do**
  while $i \geq 0$ **and** $X[i] \neq W[j]$ **do**
    $i \leftarrow B[i]$
  $i, j \leftarrow i + 1, j + 1$
  if $i = m$ **then**
    $i \leftarrow B[i]$
    $nb \leftarrow nb + 1$



W	a	a	b	a	b	a	b	b	a	b	a	a	b	a	...
					↑										
X		a	b	a	b	b									

while $j < n$ **do**
- **while** $i \geq 0$ **and** $X[i] \neq W[j]$ **do**
  - $i \leftarrow B[i]$
- $i, j \leftarrow i+1, j+1$
- **if** $i = m$ **then**
  - $i \leftarrow B[i]$
  - $nb \leftarrow nb + 1$



W															
	a	a	b	a	b	a	b	b	a	b	a	a	b	a	…

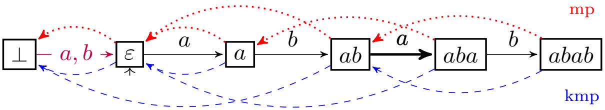
↑ (under the 6th cell of W, the a)

X					
	a	b	a	b	b

Branches in MP and KMP

while $j < n$ **do**

- **while** $i \geq 0$ **and** $X[i] \neq W[j]$ **do**
  - $i \leftarrow B[i]$
- $i, j \leftarrow i + 1, j + 1$
- **if** $i = m$ **then**
  - $i \leftarrow B[i]$
  - $nb \leftarrow nb + 1$



W	a	a	b	a	b	a	b	b	a	b	a	a	b	a	...

↑ (under the 6th cell of W)

X	a	b	a	b	b

Branches in MP and KMP

while $j < n$ **do**

  while $i \geq 0$ **and** $X[i] \neq W[j]$ **do**

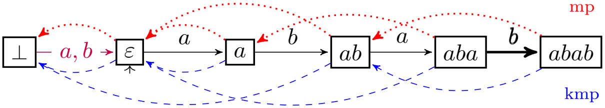
    $i \leftarrow B[i]$

  $i, j \leftarrow i+1, j+1$

  if $i = m$ **then**

    $i \leftarrow B[i]$

    $nb \leftarrow nb + 1$



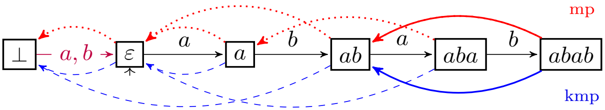
W	a	a	b	a	b	a	b	b	a	b	a	a	b	a	…

↑

X	a	b	a	b	b

Branches in MP and KMP

while $j < n$ **do**
  while $i \geq 0$ **and** $X[i] \neq W[j]$ **do**
    $i \leftarrow B[i]$
  $i, j \leftarrow i+1, j+1$
  if $i = m$ **then**
    $i \leftarrow B[i]$
    $nb \leftarrow nb + 1$



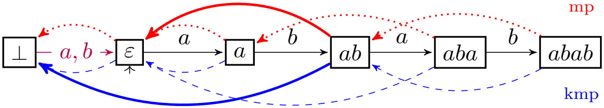
W	a	a	b	a	b	a	b	b	a	b	a	a	b	a ...

↑

X	a	b	a	b	b

Branches in MP and KMP

```
while j < n do
    while i ≥ 0 and X[i] ≠ W[j] do
        i ← B[i]
    i, j ← i + 1, j + 1
    if i = m then
        i ← B[i]
        nb ← nb + 1
```



W

a	a	b	a	b	a	b	b	a	b	a	a	b	a	…

X

a	b	a	b	b

Branches in MP and KMP

while $j < n$ **do**

  while $i \geq 0$ **and** $X[i] \neq W[j]$ **do**

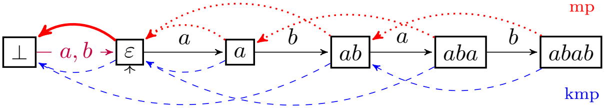
    $i \leftarrow B[i]$

  $i, j \leftarrow i + 1, j + 1$

  if $i = m$ **then**

    $i \leftarrow B[i]$

    $nb \leftarrow nb + 1$



W	a	a	b	a	b	a	b	b	a	b	a	a	b	a	…

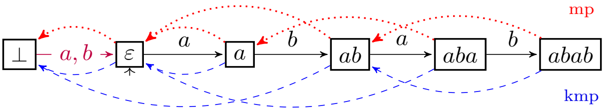
↑

X	a	b	a	b	b

kmp: skipped

Branches in MP and KMP

while $j < n$ **do**
    while $i \geq 0$ **and** $X[i] \neq W[j]$ **do**
        $i \leftarrow B[i]$
    $i, j \leftarrow i+1, j+1$
    if $i = m$ **then**
        $i \leftarrow B[i]$
        $nb \leftarrow nb + 1$



W	a	a	b	a	b	a	b	b	a	b	a	a	b	a	...
								↑							

X

Branches in MP and KMP

while $j < n$ **do**

  while $i \geq 0$ **and** $X[i] \neq W[j]$ $X[i] \neq W[j]$ **do**

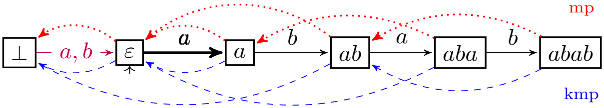
    $i \leftarrow B[i]$

  $i, j \leftarrow i+1, j+1$

  if $i = m$ **then**

    $i \leftarrow B[i]$

    $nb \leftarrow nb + 1$



W	a	a	b	a	b	a	b	b	a	b	a	a	b	a	…

↑ (under the 9th cell of W)

X	a	b	a	b	b

Branches in MP and KMP

while $j < n$ **do**

  while $i \geq 0$ **and** $X[i] \neq W[j]$ **do**

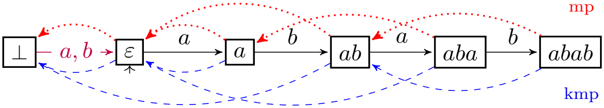
    $i \leftarrow B[i]$

  $i, j \leftarrow i + 1, j + 1$

  if $i = m$ **then**

    $i \leftarrow B[i]$

    $nb \leftarrow nb + 1$



W	a	a	b	a	b	a	b	b	a	b	a	a	b	a	...

↑ (under the 10th cell, b)

X ... ...

Expected number of mispredictions

Local predictor for MP/KPM on a random text

```
while j < n do          ← super easy: at most 3
    while i ≥ 0 and X[i] ≠ W[j] do
        i ← B[i]
    i, j ← i + 1, j + 1
    if i = m then       ← almost easy: ∼ nb. of occurrences of X
        i ← B[i]
        nb ← nb + 1
```

Expected number of mispredictions

Local predictor for MP/KPM on a random text

```
while j < n do
    while i ≥ 0 and X[i] ≠ W[j] do        ← this talk
        i ← B[i]
    i, j ← i + 1, j + 1
    if i = m then
        i ← B[i]
        nb ← nb + 1
```

Analysis of the mispredictions caused by letter comparisons:

- depends on the pattern X
- probability measure on A such that for all $\alpha \in A$, $0 < \pi(\alpha) < 1$
- transducer for the (mis)predictions + Markov chain

Expected number of mispredictions

Local predictor for MP/KPM on a random text

```
while j < n do
    while i ≥ 0 and X[i] ≠ W[j] do        ← this talk
        i ← B[i]
    i, j ← i + 1, j + 1
    if i = m then
        i ← B[i]
        nb ← nb + 1
```

Analysis of the mispredictions caused by letter comparisons:

- depends on the pattern X
- probability measure on A such that for all $\alpha \in A$, $0 < \pi(\alpha) < 1$
- transducer for the (mis)predictions + Markov chain
- same kind of ideas for $i \geq 0$

Results: number of mispredictions per symbol

Asymptotic expected number of mispredictions per symbol in KMP with $\Sigma = \{a, b\}$ and $p := \pi(a) = 1 - \pi(b)$.

X	`i = m`	`i >= 0`	`X[i] != T[j]`
aa	too large	$1-p$	$\frac{p(1-p)}{1-2p+2p^2}$
ab	$p(1-p)$	$(1-p)^2$	$\frac{p(3-7p+7p^2-2p^3)}{1-p+2p^2-p^3}$
aaa	$p^3(1-p)(1+p)^2$	$1-p$	$\frac{p(1-p)}{1-2p+2p^2}$
aab	$p^2\,(1-p)$	$(1-p)^2(1+p)$	$\frac{p(1-2p^2-p^3+5p^4-3p^5+p^6)}{1-2p+3p^2-2p^3+p^4}$
aba	$p^2\,(1-p)$	$(1-p)^2$	$\frac{p(3-7p+7p^2-2p^3)}{1-p+2p^2-p^3}$
abb	$p(1-p)^2$	$(1-p)^3$	$p(4-13p+21p^2-16p^3+6p^4-p^5)$

Last column for $x = abab$:

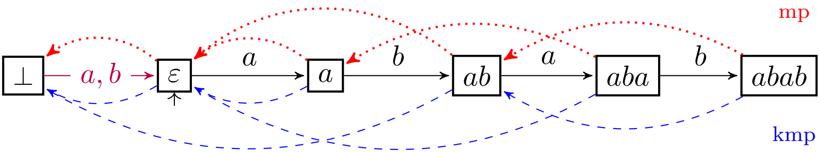
$$\frac{\pi_a(-\pi_a^3\pi_b + 2\pi_a^2\pi_b^3 + 4\pi_a^2\pi_b^2 + 3\pi_a^2\pi_b + \pi_a^2 - 5\pi_a\pi_b^2 - 4\pi_a\pi_b - 2\pi_a + 2\pi_b + 1)}{(1-\pi_a)(\pi_a^2\pi_b^2 + \pi_a^2\pi_b - \pi_a\pi_b - \pi_a + 1)}$$

Results: number of mispredictions per symbol

Asymptotic expected number of mispredictions per input symbol in a random text, with **uniform** distribution over alphabets of size 2 or 4.

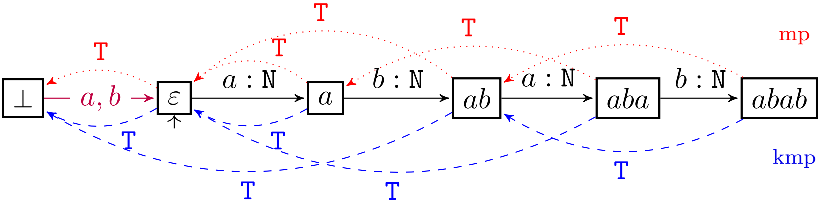
	$\lvert A\rvert = 2$					$\lvert A\rvert = 4$				
X	`i=m`	`i>=0`	algo	`X[i]!=T[j]`	Total	`i=m`	`i>=0`	algo	`X[i]!=T[j]`	Total
aa	0.283	0.5	mp	0.571	1.353	0.073	0.75	mp	0.295	1.117
			kmp	0.5	1.283			kmp	0.3	1.123
ab	0.25	0.25	both	0.571	1.321	0.062	0.688	both	0.375	1.186
aaa	0.14	0.5	mp	0.563	1.202	0.018	0.75	mp	0.293	1.06
			kmp	0.5	1.14			kmp	0.3	1.068
aab	0.125	0.375	mp	0.605	1.23	0.015	0.734	mp	0.322	1.086
			kmp	0.542	1.166			kmp	0.322	1.086
aba	0.125	0.25	mp	0.708	1.083	0.015	0.688	mp	0.367	1.068
			kmp	0.571	0.946			kmp	0.375	1.076
abb	0.125	0.125	both	0.547	0.921	0.015	0.672	both	0.397	1.098

while $j < n$ **do**
  while $i \geq 0$ **and** $X[i] \neq W[j]$ **do**
    $i \leftarrow B[i]$
  $i, j \leftarrow i+1, j+1$
  if $i = m$ **then**
    $i \leftarrow B[i]$
    $nb \leftarrow nb + 1$



Analysis of letter comparisons

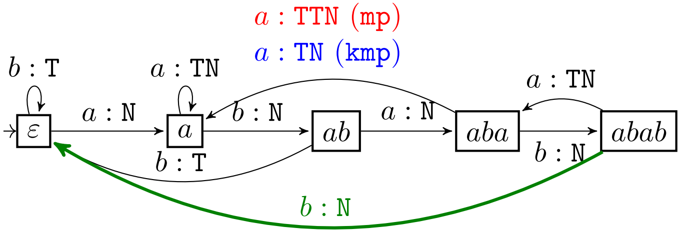
while $j < n$ **do**
  while $i \geq 0$ **and** $X[i] \neq W[j]$ **do**
    $i \leftarrow B[i]$
  $i, j \leftarrow i + 1, j + 1$
  if $i = m$ **then**
    $i \leftarrow B[i]$
    $nb \leftarrow nb + 1$



Transducer following if the branch is taken or not.

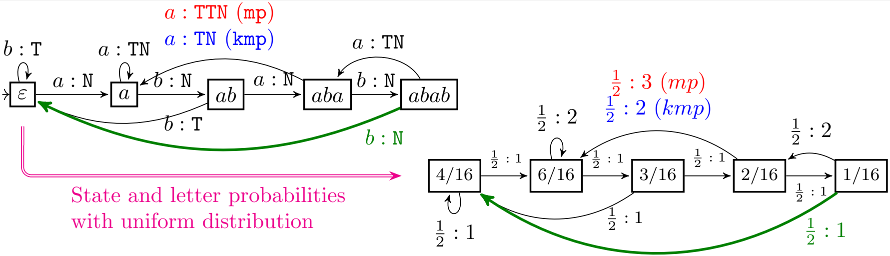
Analysis of letter comparisons

while $j < n$ **do**
  while $i \geq 0$ **and** $X[i] \neq W[j]$ **do**
    $i \leftarrow B[i]$
  $i, j \leftarrow i+1, j+1$
  if $i = m$ **then**
    $i \leftarrow B[i]$
    $nb \leftarrow nb + 1$



Transducer following the branches for each letter read in W.

Number of comparisons (`X[i]!=T[j]`)



Lemma (proba. of being in state u after reading j letters of W)

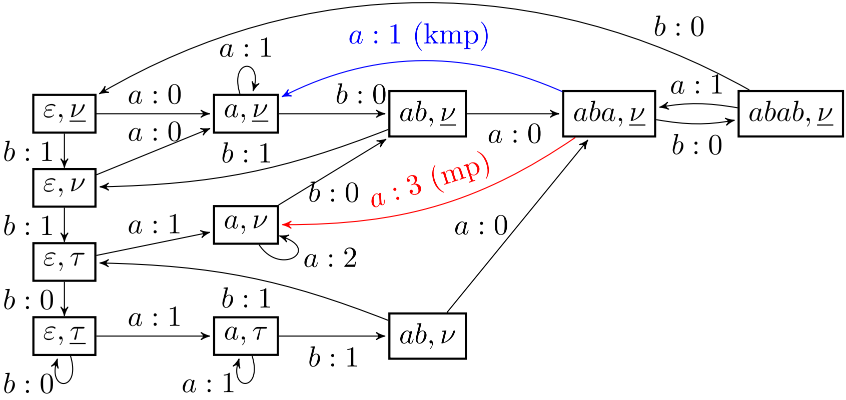
$$p_X(j,u) = p_X(u) := \pi(u) - \sum_{\substack{state\ v \in Q_X \\ bord(v)=u}} \pi(v)$$

Proposition

The expected number of letter **comparisons** *performed by (K)MP on a random text of length n and a pattern X is asymptotically equivalent to $C_X \cdot n$ as $n \to \infty$, where*

$$C_X = \sum_{u \in Q_X} p_X(u) \sum_{a \in A} \pi(a) \cdot \left| output\left(u \xrightarrow{a}\right) \right|, \text{ and } 1 \le C_X \le 2.$$

Number of mispredictions (`X[i]!=T[j]`)

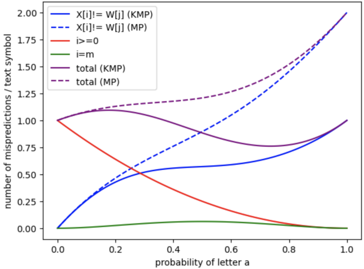


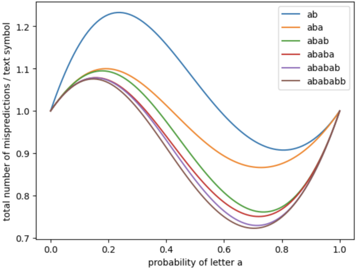
Proposition

The expected number of **mispredictions** *caused by letter comparisons in KMP on a random text of length* n *and a pattern* X*, is asymptotically equivalent to* $L_X \cdot n$*, with*

$$L_X = \sum_{u \in Q_X} \sum_{\lambda \in \{\underline{\nu}, \nu, \tau, \underline{\tau}\}} \pi_0(u, \lambda) \times \sum_{\alpha \in A} \pi(\alpha) \cdot output((u, \lambda) \xrightarrow{\alpha})$$

$X = abab$

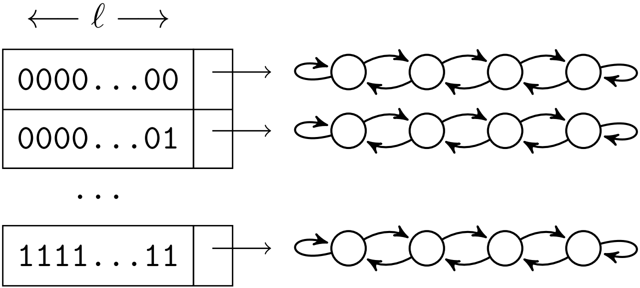




- **Today:** a first theoretical exploration of pattern matching algorithms, considering local branch prediction.
- **Future:** analysis of **global predictors** to capture correlations. In our simulations, the actual number of mispredictions is roughly divided by $|A|$ in practice.
- **Other possible direction:** enhanced probabilistic distributions for texts, other than memoryless sources (*e.g.* Markovian sources should be manageable within our model).

Example of global (or mixed) predictor

- History of the ℓ last branches of a whole program
- Each possible global history is associated with a 2-bit saturated counter



Computer Architecture: A Quantitative Approach (5th ed.), Hennessy & Patterson