L14

4102

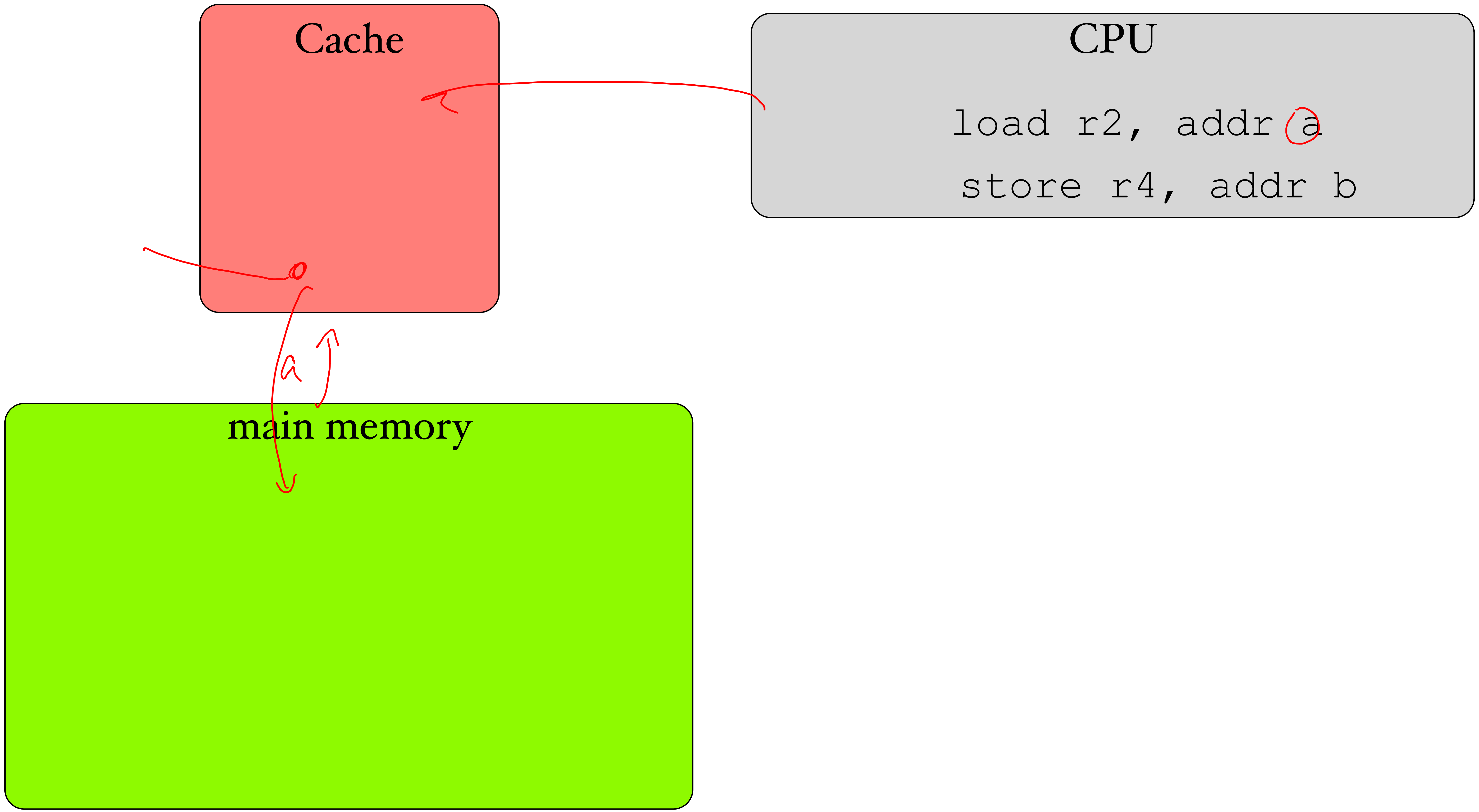
10.10.2013

abhi shelat

Greedy Alg:
Caching,
HUFFMAN CODING

CACHING

CACHE HIT



QUESTION:

How to manage the cache??

Best-case scenario in which the pattern of all memory accesses is known a priori

PROBLEM STATEMENT

input: K, the size of the cache
$d_1, d_2, \ldots, d_m$ memory accesses

output: min # of cache misses

cache is fully associative, line size is 1

BELADY EVICT RULE

"farthest-in-the-future" or FF

If you must evict, evict the element that is accessed the farthest in the future.

EXAMPLE

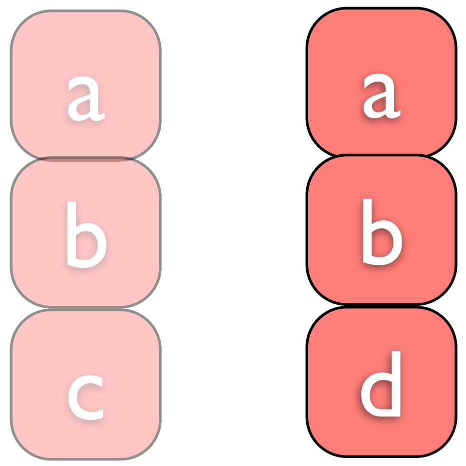
cache

a
b
c

a b c d a d e a d b a e c e a

EXAMPLE

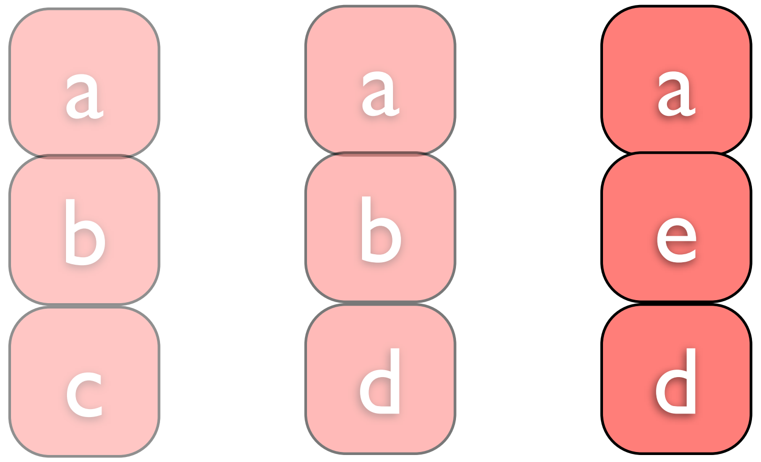
cache



a b c d a d e a d b a e c e a

EXAMPLE

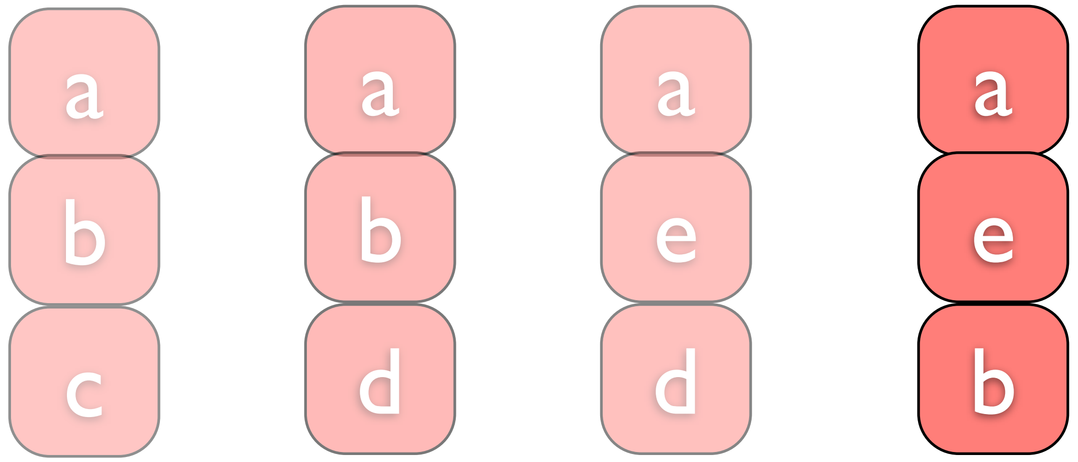
cache



a b c d a d e a d b a e c e a

EXAMPLE

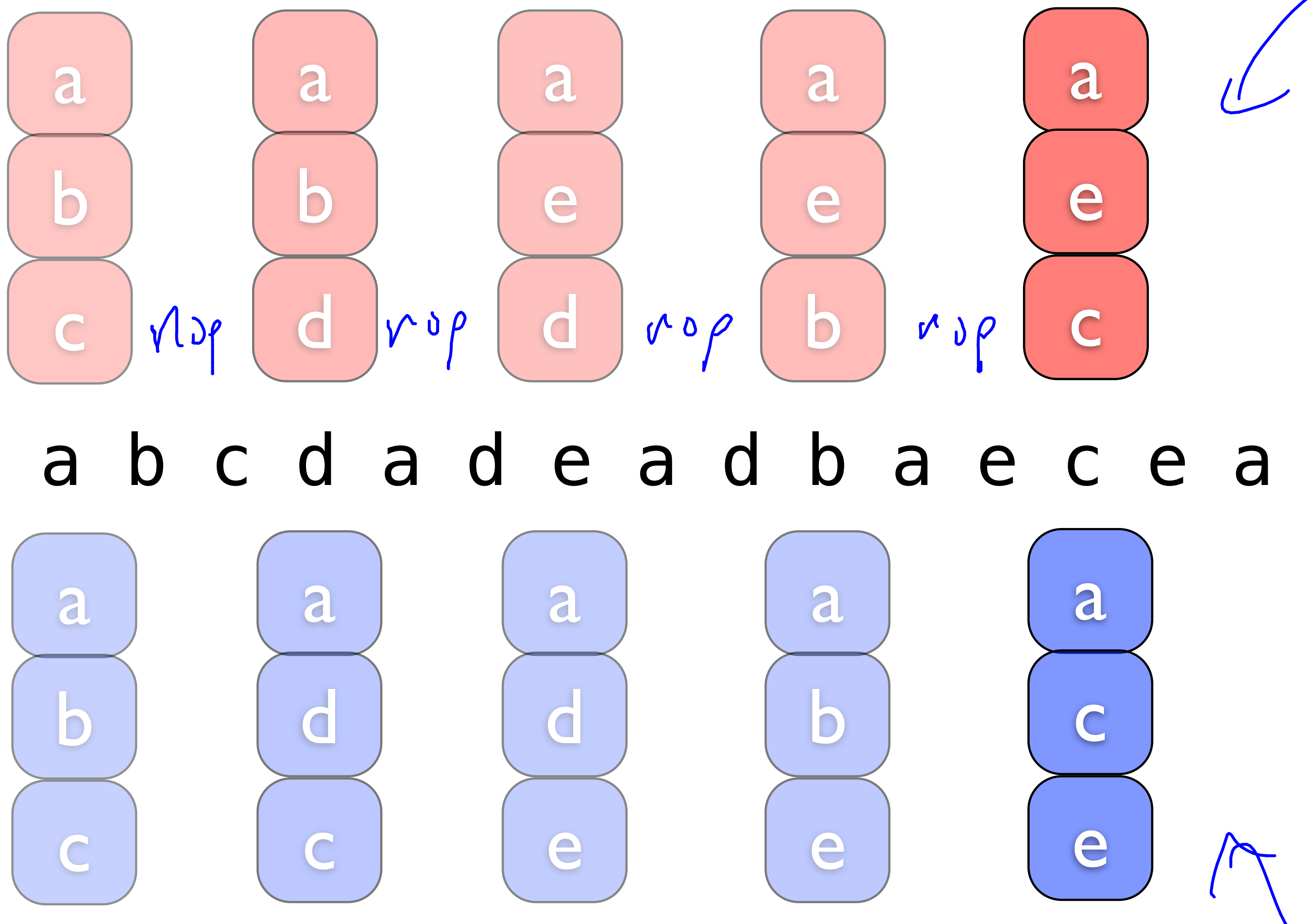
cache



a b c d a d e a d b a e c e a

EXAMPLE

cache



SFF

schedule of operations to the cache that follows SFF

another schedule that has the same # of misses.

SURPRISING THEOREM

The schedule in which we evict the item that is accessed farthest-in-the-future, ie, S_{ff} is optimal.

SCHEDULE

Schedule for access pattern d1,d2,...,dn:

operation on the cache @ each access

"nop" or "evict x for y"

Reduced schedule:

schedule in which "evict x for y" only

occurs when the access is $d_i = y$

for any schedule S, $\text{misses}(\text{Reduced}(S)) \leq \text{misses}(S)$

EXCHANGE LEMMA

If S is a reduced schedule that agrees w/ S_{FF} on the first j ops,

then $\exists$ a reduced schedule S' that agrees w/ S_{FF}

on $j+1$ operations, and

$$\text{misses}(S') \leq \text{misses}(S).$$

Exchange Lemma:

Let S be a reduced sched that agrees with S_{ff} on j items.
There exists a reduced sched S' that agrees on j+1 items
and has the same or fewer # of misses as S.

optimal $\rightsquigarrow$ $S^* \underset{\text{lemma}}{\sim} S_1 \underset{\text{lemma}}{\sim} S_2 \sim \dots \sim S_{ff}$

$j=1$ $\quad$ $j=2$ $\quad$ $j=m$

why do we care about this lemma??

$$\text{misses}(S_{ff}) \leq \text{misses}(S_{opt})$$

Thm: Let S be a reduced sched that agrees with S_{ff} on j items. There exists a reduced sched S' that agrees on j+1 items and has the same # of misses as S.

Proof: Since S and S_{ff} agree on the first j operations, both schedules produce the same cache state.

Let d be the address accessed at time j+1.

State of the cache after J operations under the two schedules.

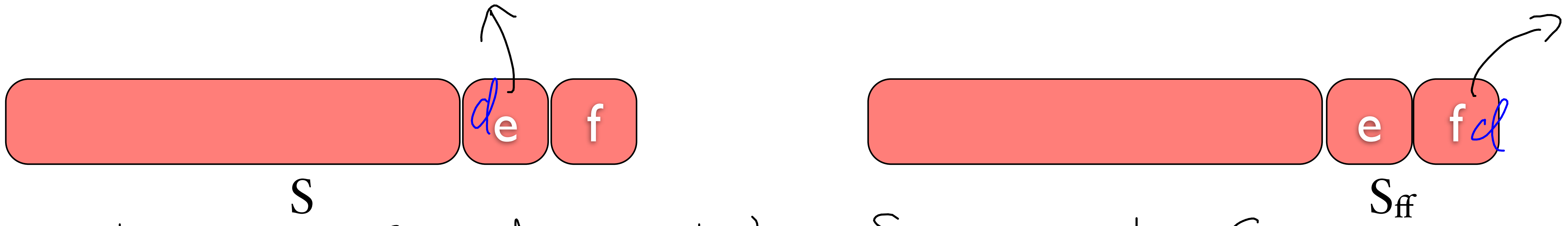
S $\qquad\qquad$ S_{ff}

easy case 1 $d \in$ cache

easy case 2 $d \notin$ cache but both S and S_{ff} "evict e for d"

$\Rightarrow$ Both S + S_{ff} also agree on the first $j+1$ operations.

So set $S' = S$. done.



case 3: $d \notin$ cache. S evicts e but S_{ff} evicts f.

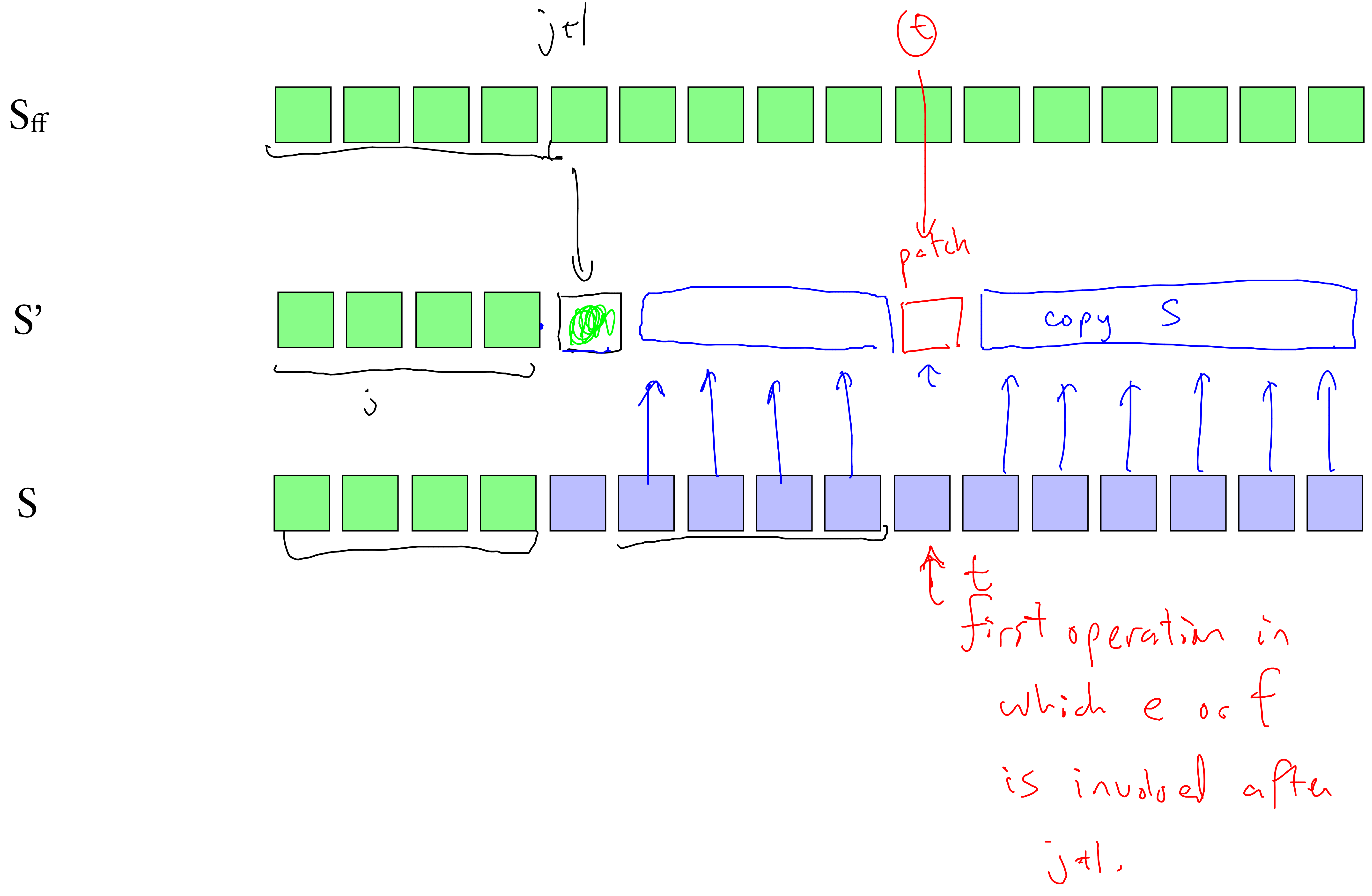
harder

so how can we construct S' in this case??

S' must do what S_{ff} does and

"evict f for d"

THE CONSTRUCTION OF S'



State of cache

Let access t be the first access involving e or f after j+1.

(1) S & S' will be the same @ this point except for $\{d\ f\} \cup \{e, d\}$.

Cases to consider:

t accesses e.

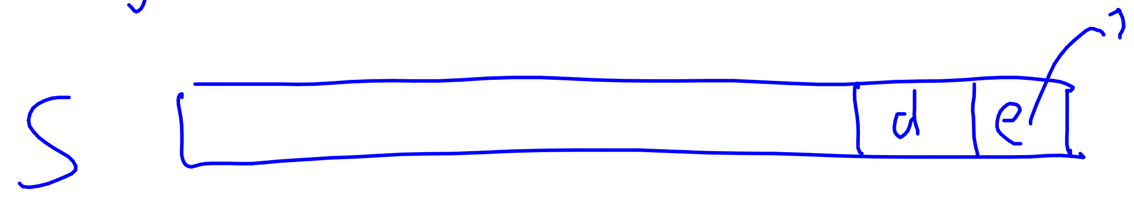
t accesses f.

t accesses $g \neq e \neq f$.

what if t=e ? ✓

① S must evict some element to bring in e.

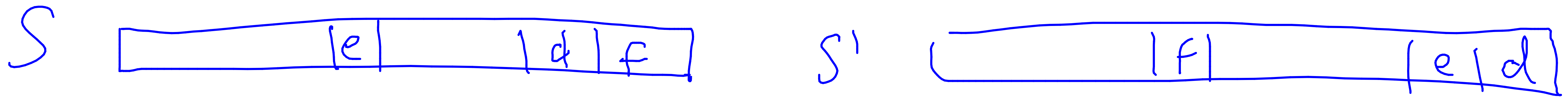
– if S evicts f, then S + S' agree.



$\text{misses}(S') < \text{misses}(S)$

$h \neq f$

– if S "evicts h for e" ⇒ make S' "evict h for f"



$\text{misses}(S') = \text{misses}(S)$

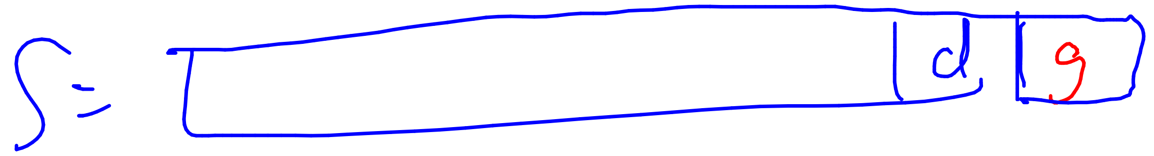
what if t=f ?

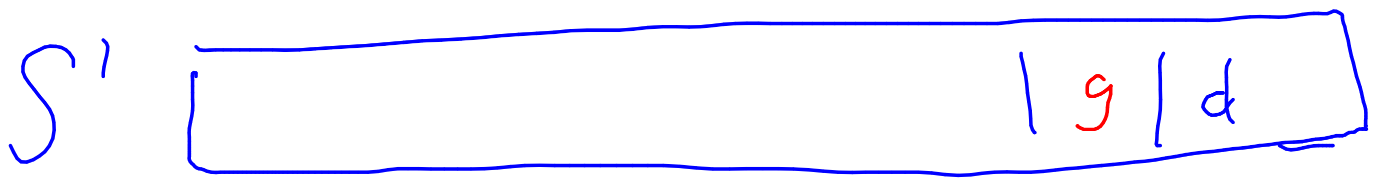
IMPOSSIBLE !!

SFF evicted f instead of e. This means that f had to be accessed after e → farthest in the future

what if t is neither e nor f ?

⇒ S's operation must be "evict f for g"





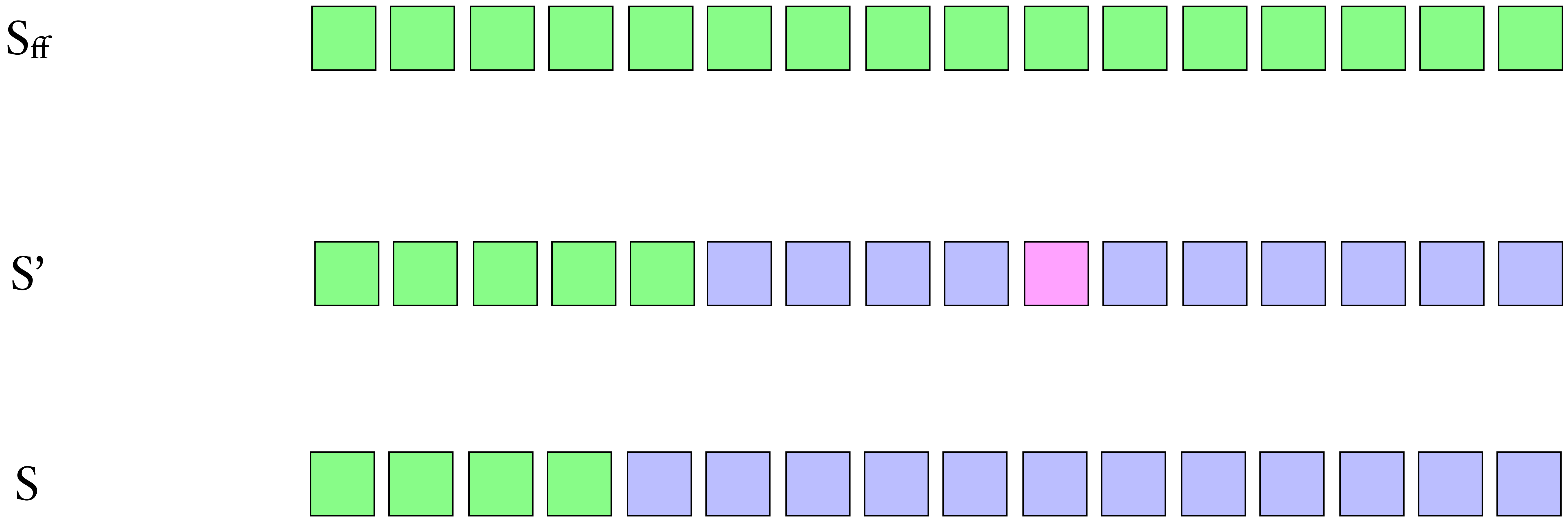
S': "evict e for g"

misses(S) = misses(S')

Finally, we set

S' = Reduce(S') and conclude thm.

WHAT HAVE WE SHOWN

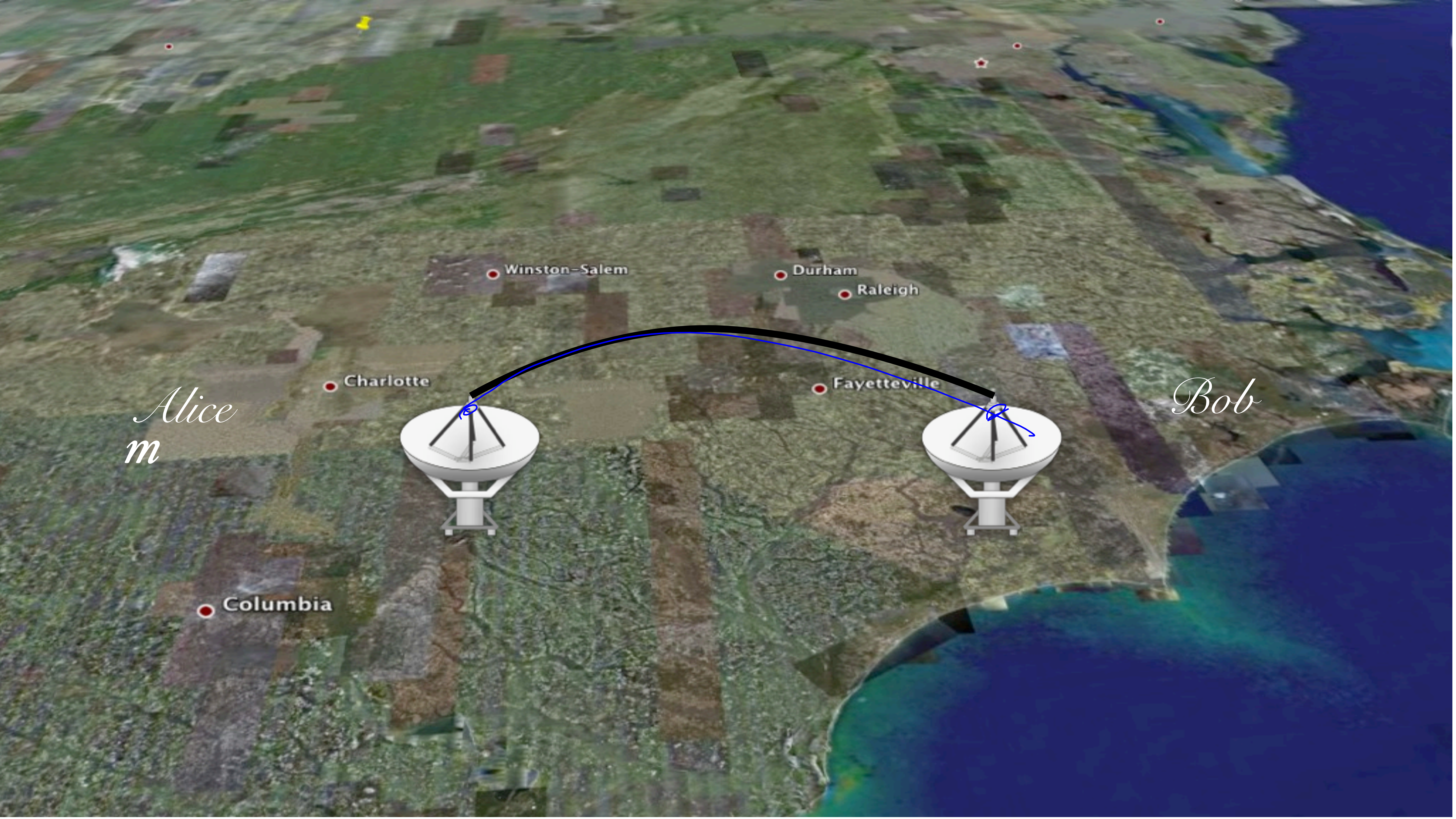


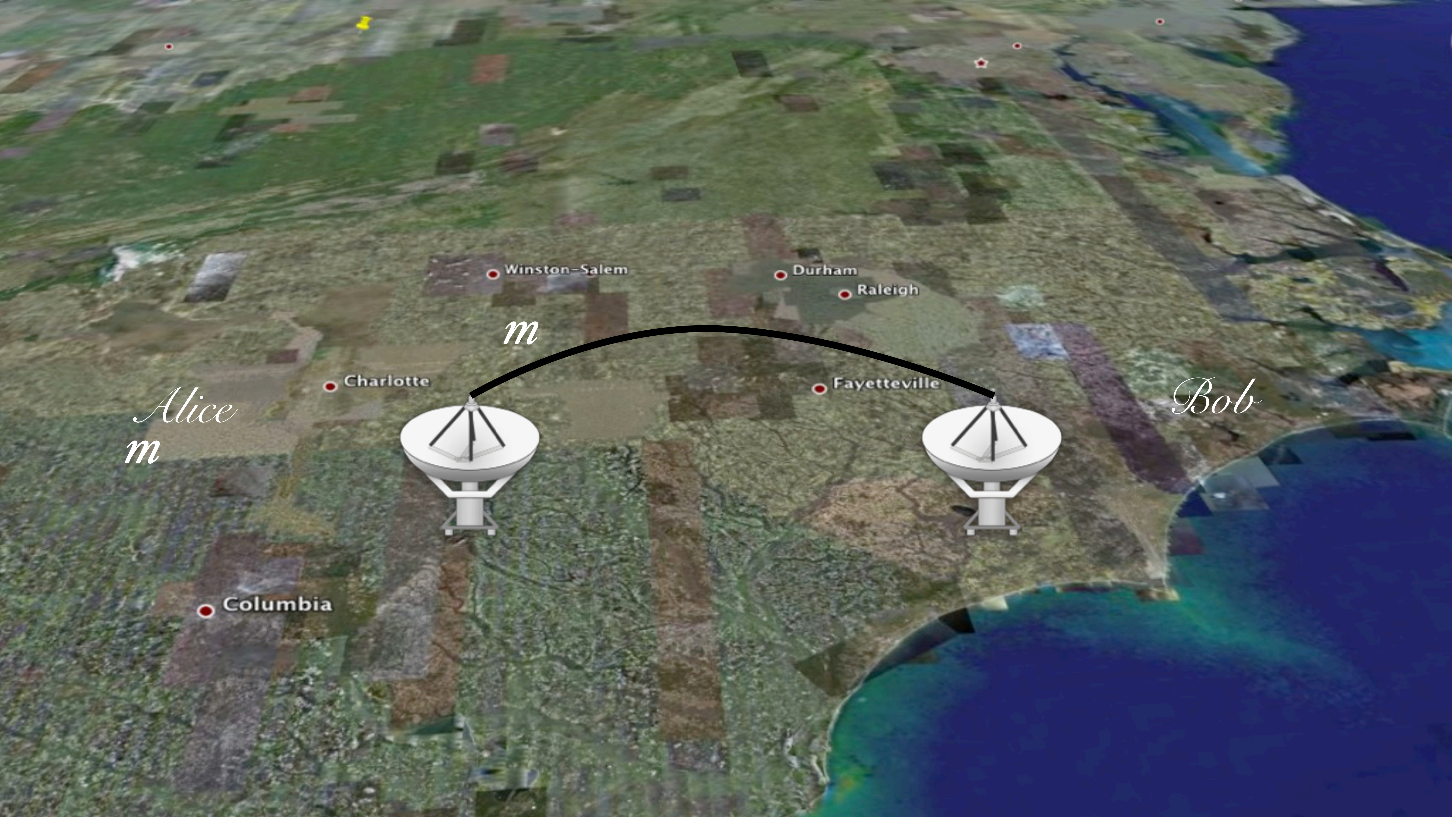
misses(S') ≤ misses(S)

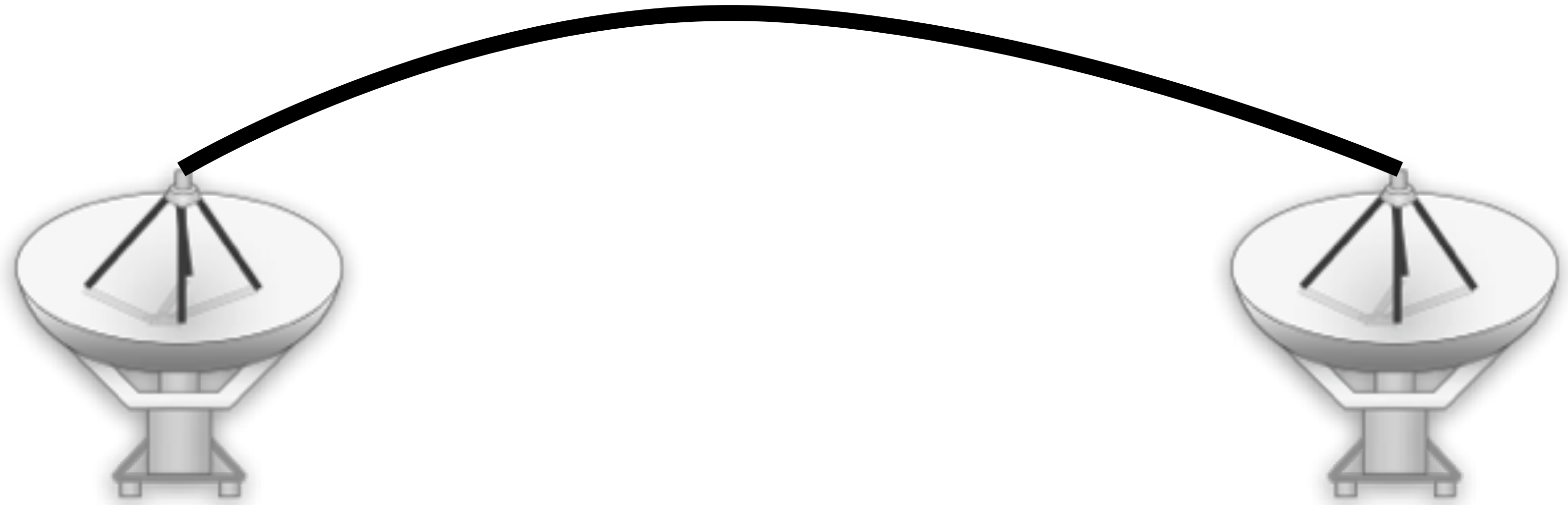
S^* S_{ff}

Huffman Coding

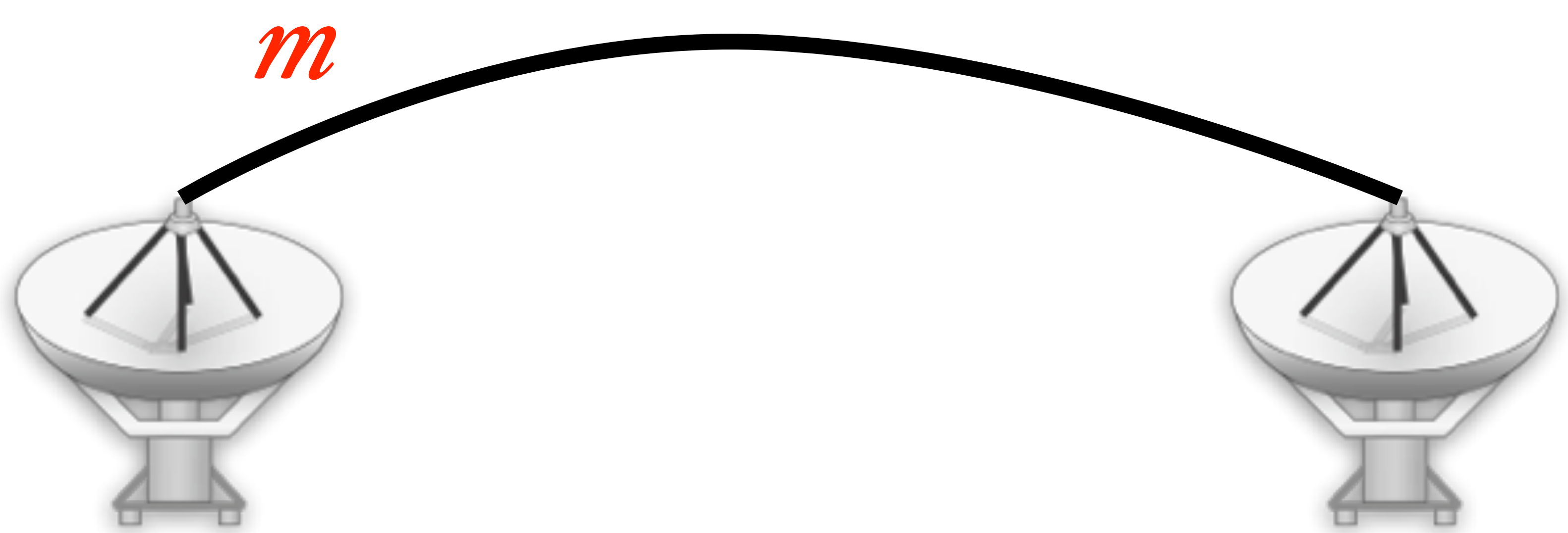
Morse


Winston-Salem
Durham
Raleigh
Charlotte
Fayetteville
Alice
m
Bob
Columbia


Winston-Salem
Durham
Raleigh
Charlotte
Fayetteville
Columbia
Alice
Bob
m
m



In testimony before the committee, Mr. Lew stressed that the Treasury Department would run out of "extraordinary measures" to free up cash in a matter of days. At that point, the country's bills might overwhelm its cash on hand plus any receipts from taxes or other sources, leading to an unprecedented default.Mr. Lew said that Treasury had no workarounds to avoid breaching the debt ceiling. "There is no plan other than raising the debt limit," he said. "The legal issues, even regarding interest and principal on the debt, are complicated."



In testimony before the committee, Mr. Lew stressed that the Treasury Department would run out of “extraordinary measures” to free up cash in a matter of days. At that point, the country’s bills might overwhelm its cash on hand plus any receipts from taxes or other sources, leading to an unprecedented default.Mr. Lew said that Treasury had no workarounds to avoid breaching the debt ceiling. “There is no plan other than raising the debt limit,” he said. “The legal issues, even regarding interest and principal on the debt, are complicated.”

$c \in C$	f_c	T
e:	235	000
i:	200	001
o:	170	010
u:	87	⋮
p:	78	
g:	47	
b:	40	
f:	24	

881

$c \in C$	f_c	T	ℓ_c
e:	235	000	3
i:	200	001	3
o:	170	010	3
u:	87	011	3
p:	78	100	3
g:	47	101	3
b:	40	110	3
f:	24	111	3

881

000 101 101 - ...

cost of sending an 881 char msg would be $881 \cdot 3 = 2643$

Q: Can we do better??

DEF: COST OF AN ENCODING

$$\underbrace{B(T, \{f_c\})}_{\text{encoding}} = \sum_{c \in C} f_c \cdot \ell_c$$

$c \in C$	f_c	T	ℓ_c
e:	235	000	3
i:	200	001	3
o:	170	010	3
u:	87	011	3
p:	78	100	3
g:	47	101	3
b:	40	110	3
f:	24	111	3
	881		

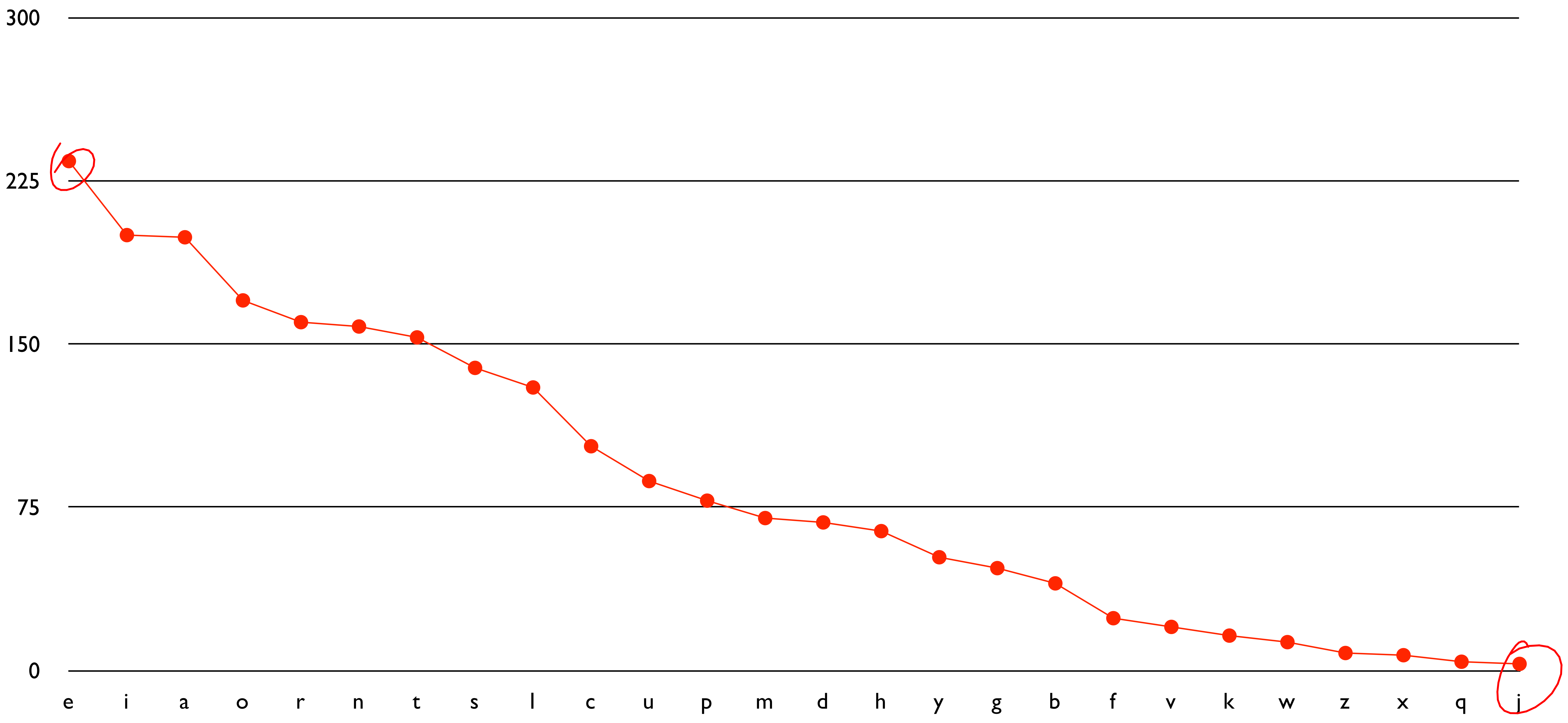
simple

$\ell_c = 3$ for every $c \in C$

alphabet

CHARACTER FREQUENCY

e: 234803
i: 200613
a: 198938
o: 170392
r: 160491
n: 158281
t: 152570
s: 139238
l: 130172
c: 103307
u: 87211
p: 78077
m: 70504
d: 68007
h: 64165
y: 51527
g: 47011
b: 40351
f: 24110
v: 20103
k: 16012
w: 13825
z: 8439
x: 6926
q: 3729
j: 3075



MORSE CODE

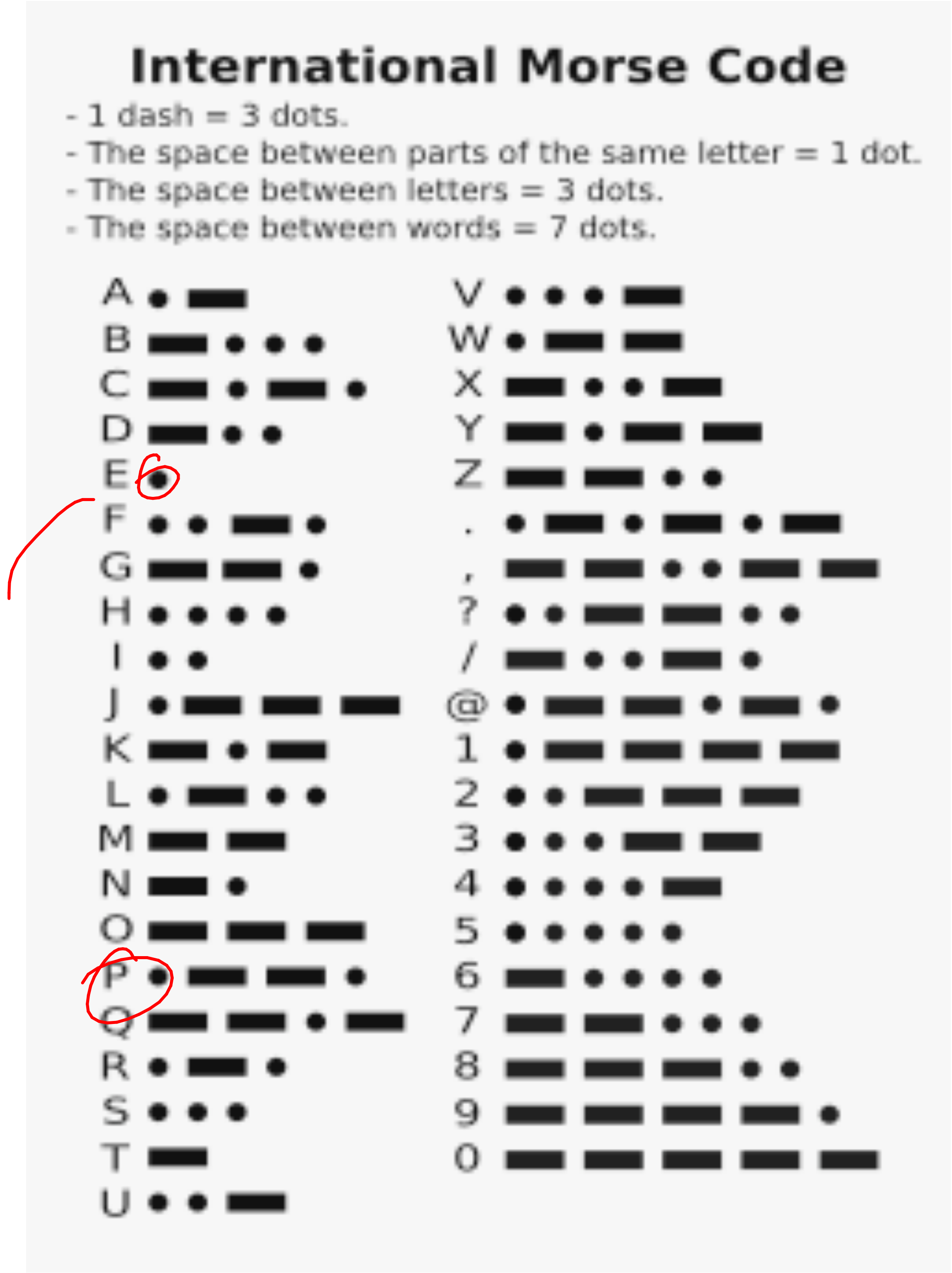
image http://en.wikipedia.org/wiki/Morse_code

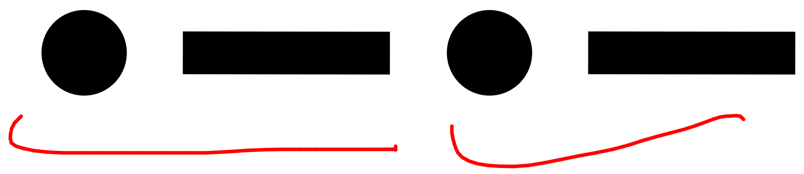
A A

E T E T

R T

MORSE CODE


International Morse Code
- 1 dash = 3 dots.
- The space between parts of the same letter = 1 dot.
- The space between letters = 3 dots.
- The space between words = 7 dots.
A • ▬
B ▬ • • •
C ▬ • ▬ •
D ▬ • •
E •
F • • ▬ •
G ▬ ▬ •
H • • • •
I • •
J • ▬ ▬ ▬
K ▬ • ▬
L • ▬ • •
M ▬ ▬
N ▬ •
O ▬ ▬ ▬
P • ▬ ▬ •
Q ▬ ▬ • ▬
R • ▬ •
S • • •
T ▬
U • • ▬
V • • • ▬
W • ▬ ▬
X ▬ • • ▬
Y ▬ • ▬ ▬
Z ▬ ▬ • •
. • ▬ • ▬ • ▬
, ▬ ▬ • • ▬ ▬
? • • ▬ ▬ • •
/ ▬ • • ▬ •
@ • ▬ ▬ • ▬ •
1 • ▬ ▬ ▬ ▬
2 • • ▬ ▬ ▬
3 • • • ▬ ▬
4 • • • • ▬
5 • • • • •
6 ▬ • • • •
7 ▬ ▬ • • •
8 ▬ ▬ ▬ • •
9 ▬ ▬ ▬ ▬ •
0 ▬ ▬ ▬ ▬ ▬



DEF: PREFIX-FREE CODE

for every $x, y \in C$ $x \neq y$

$c(x)$ is not a prefix of $c(y)$

DEF: PREFIX-FREE CODE

$$\forall x, y \in C, x \neq y \implies \textsc{code}(x) \text{ not a prefix of } \textsc{code}(y)$$

DEF: PREFIX CODE

$$\forall x, y \in C, x \neq y \implies \textsc{code}(x) \text{ not a prefix of } \textsc{code}(y)$$

e:	235	0
i:	200	10
o:	170	110
u:	87	1110
p:	78	11110
g:	47	111110
b:	40	1111110
f:	24	11111110

0 - 111110 .

DECODING A PREFIX CODE

e:	235	0
i:	200	10
o:	170	110
u:	87	1110
p:	78	11110
g:	47	111110
b:	40	1111110
f:	24	11111110

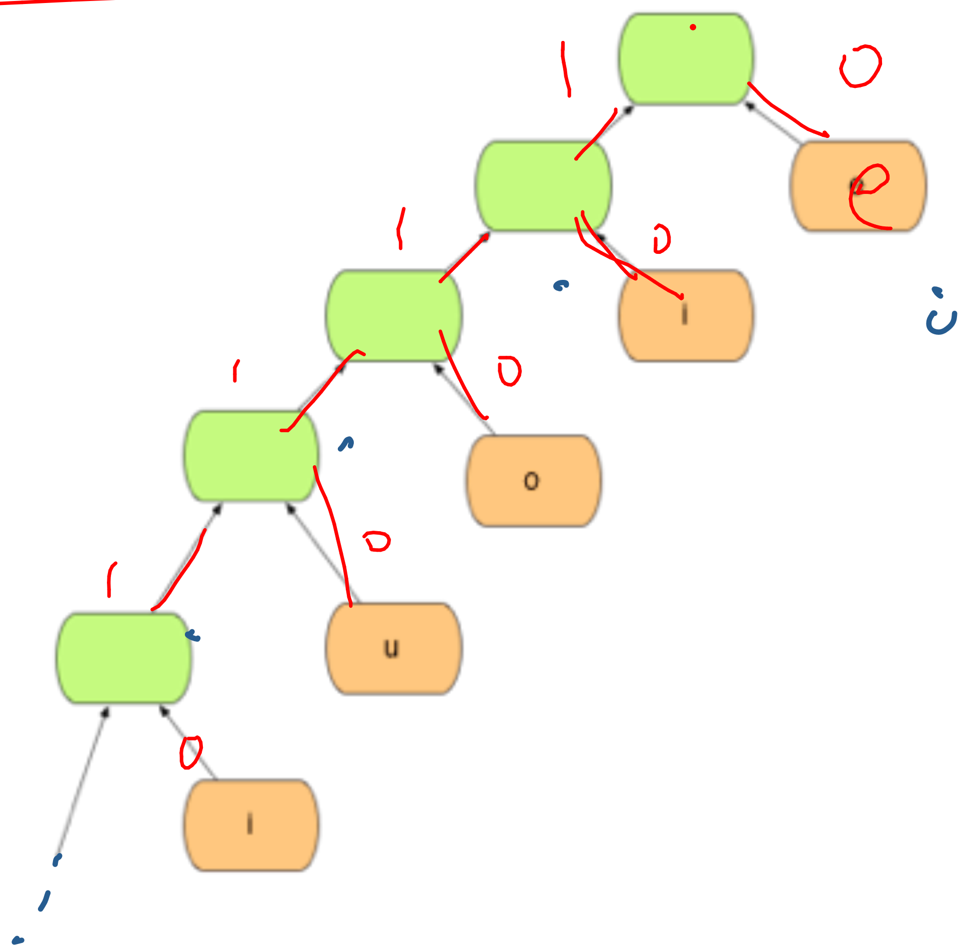
$\underbrace{1111110}\underbrace{10}111110$

Big

CODE TO BINARY TREE

e:	235	0
i:	200	10
o:	170	110
u:	87	1110
p:	78	11110
g:	47	111110
b:	40	1111110
f:	24	11111110

prefix-free code →



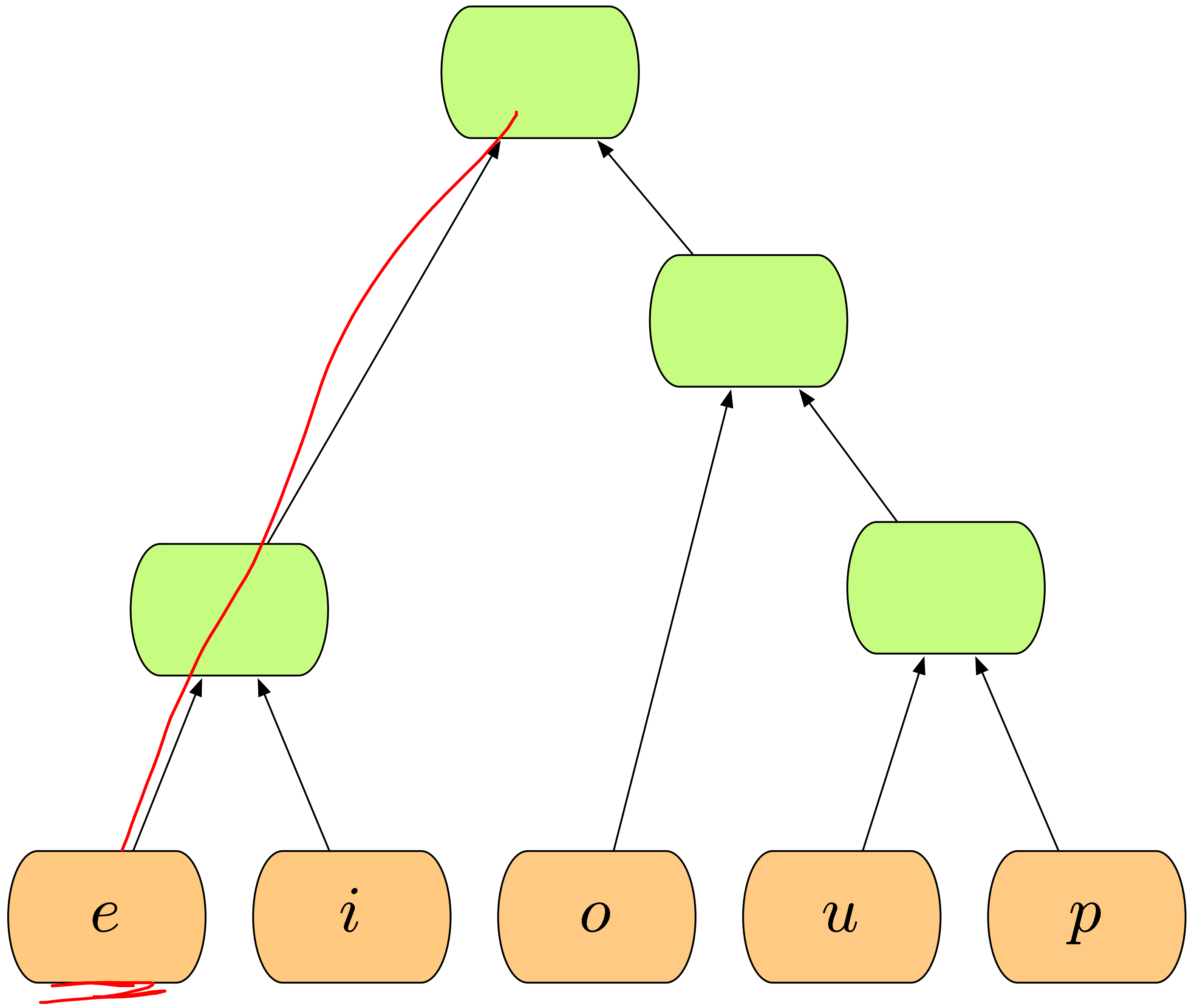
11111101011110

B I

PREFIX CODE

$\Updownarrow$

BINARY TREE



$c \in C$	f_c	T	ℓ_c
e:	235	00	2
i:	200	01	2
o:	170	10	2
u:	87	110	3
p:	78	111	3

USE TREE TO ENCODE MESSAGES

GOAL

GIVEN THE frequencies for an alphabet $\{f_c\}_{c \in C}$

compute the (prefix-free encoding) T that

binary tree

$$\min_{T} \left\{ \underline{B(T, \{f_c\})} \right\}$$

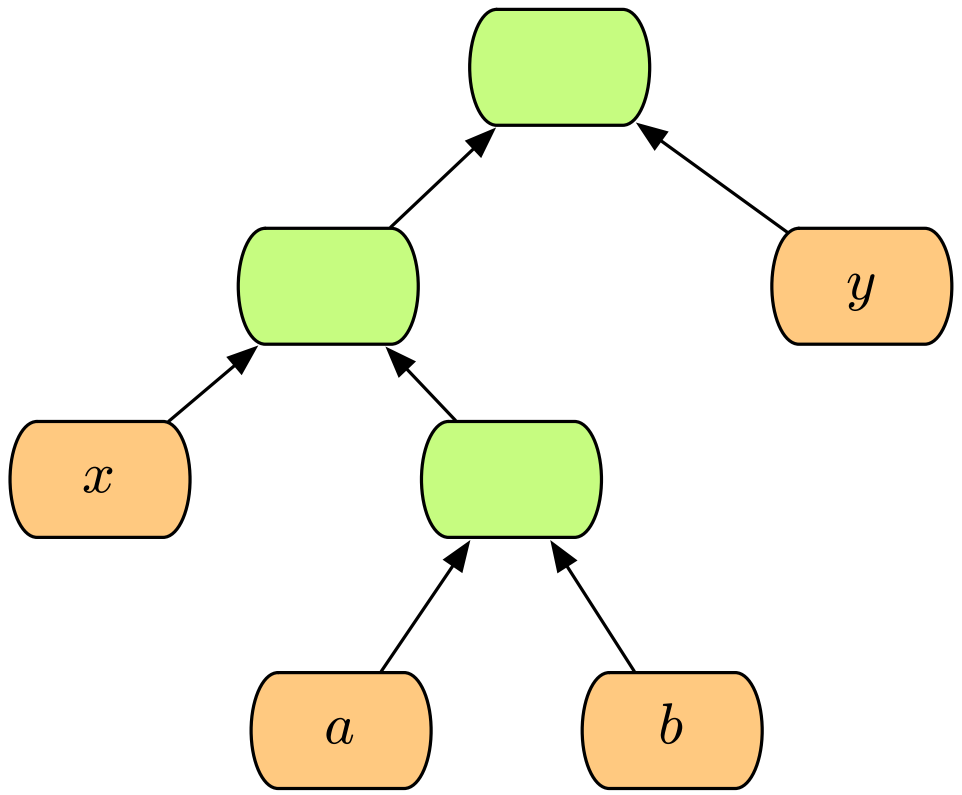
GOAL

GIVEN THE CHARACTER FREQUENCIES $\{f_c\}_{c \in C}$

PRODUCE A PREFIX CODE T WITH SMALLEST COST

$$\min_T B(T, \{f_c\})$$

PROPERTY

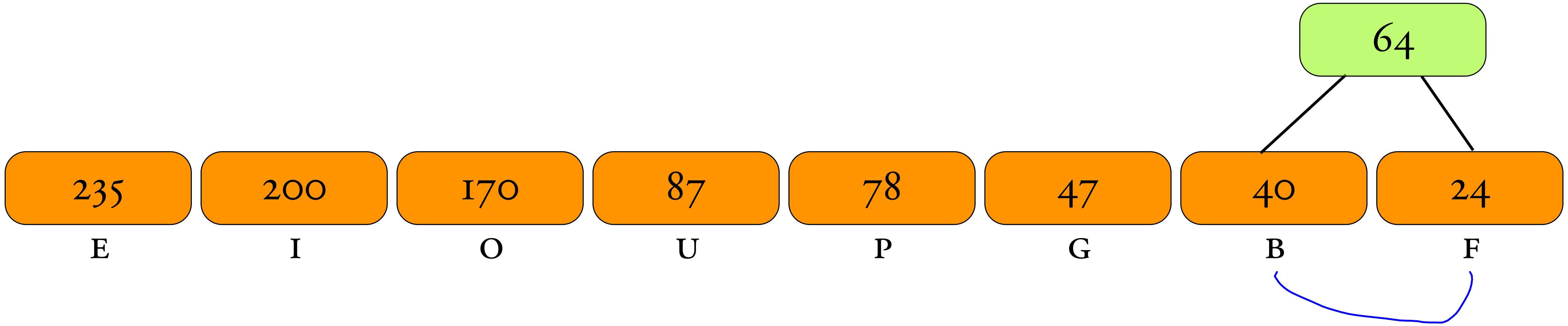


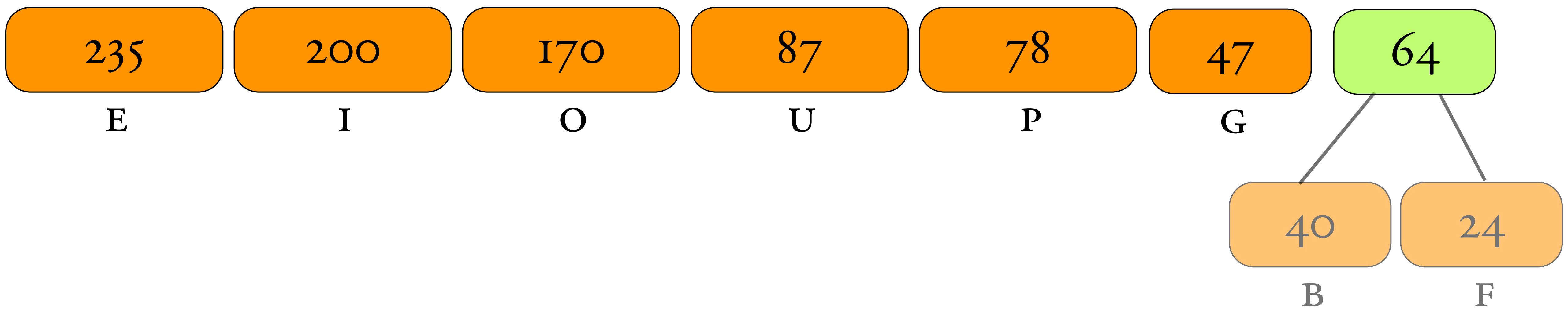
LEMMA:OPTIMAL TREE MUST BE FULL.

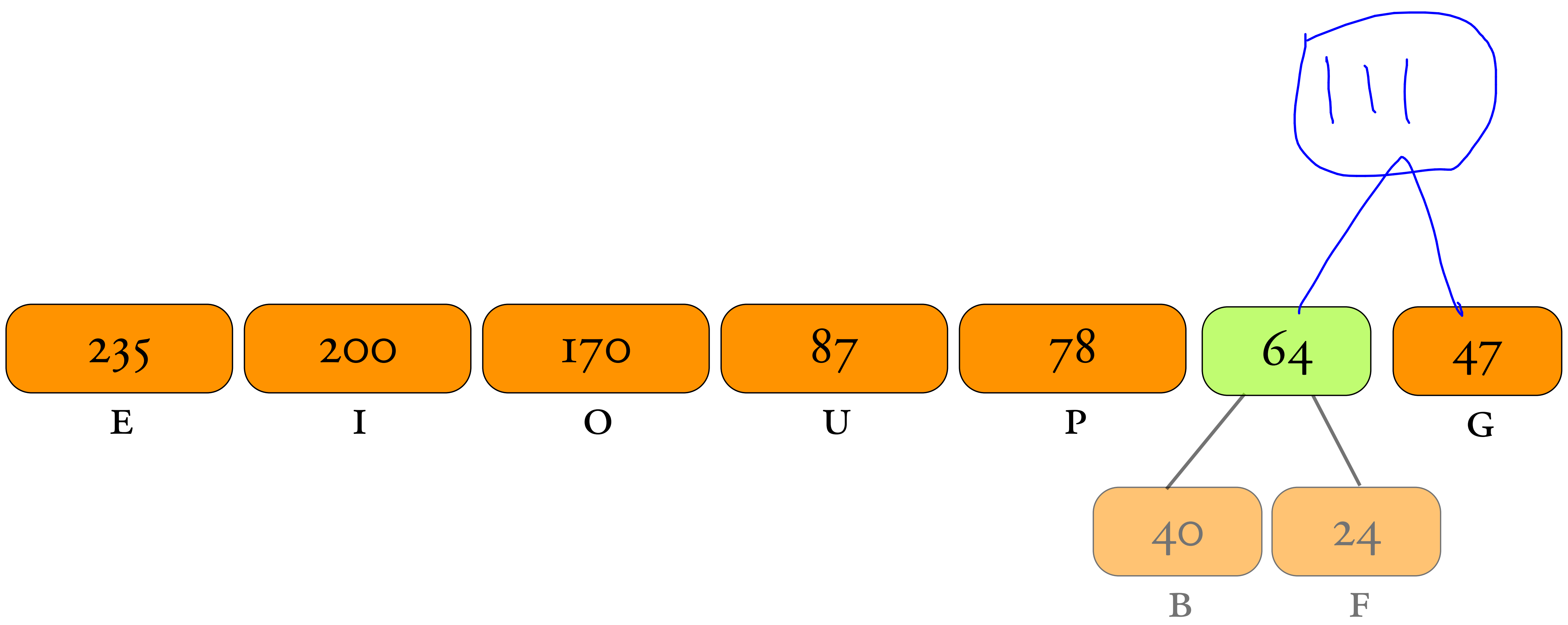
DIVIDE & CONQUER?

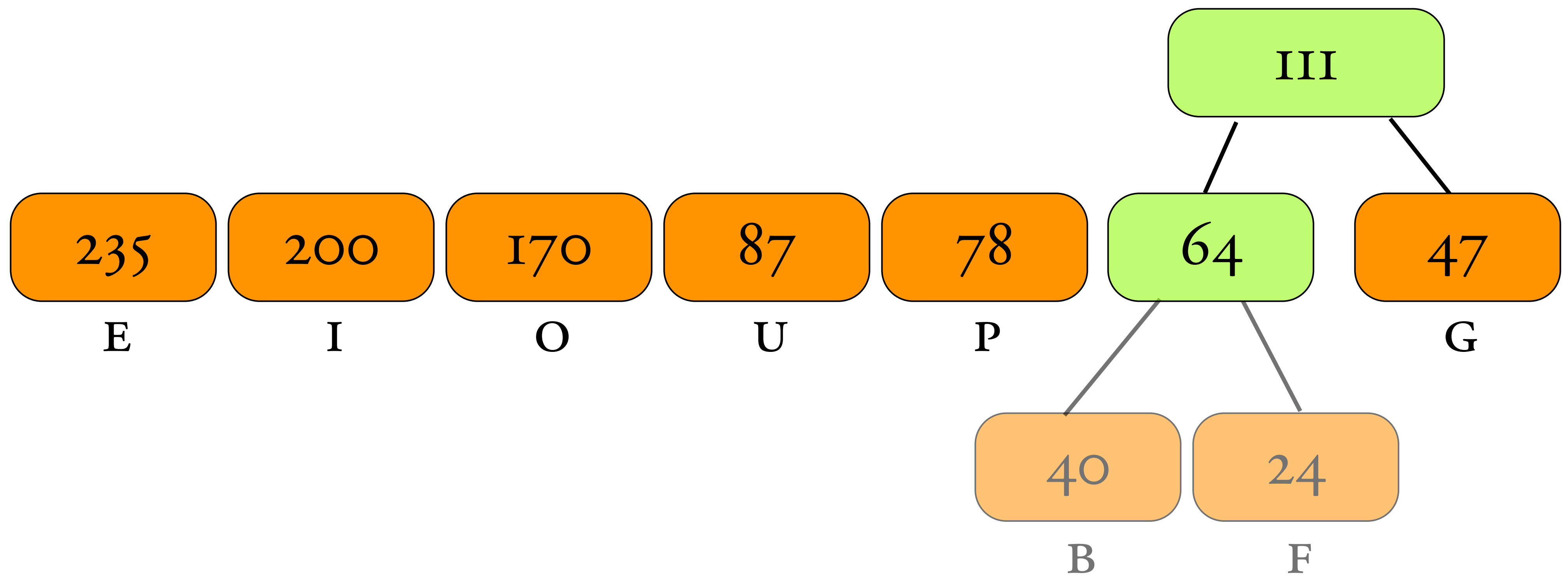
COUNTER-EXAMPLE

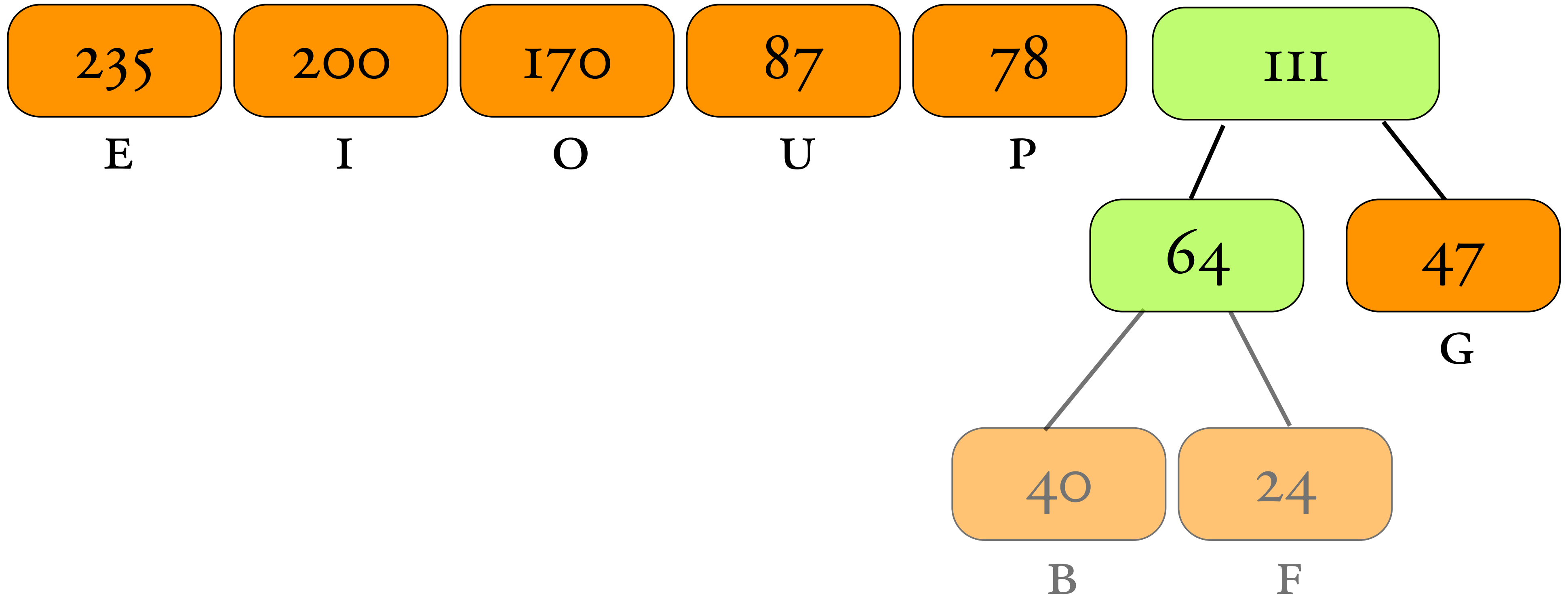
```
e: 32
i: 25
o: 20
u: 18
p: 5
```

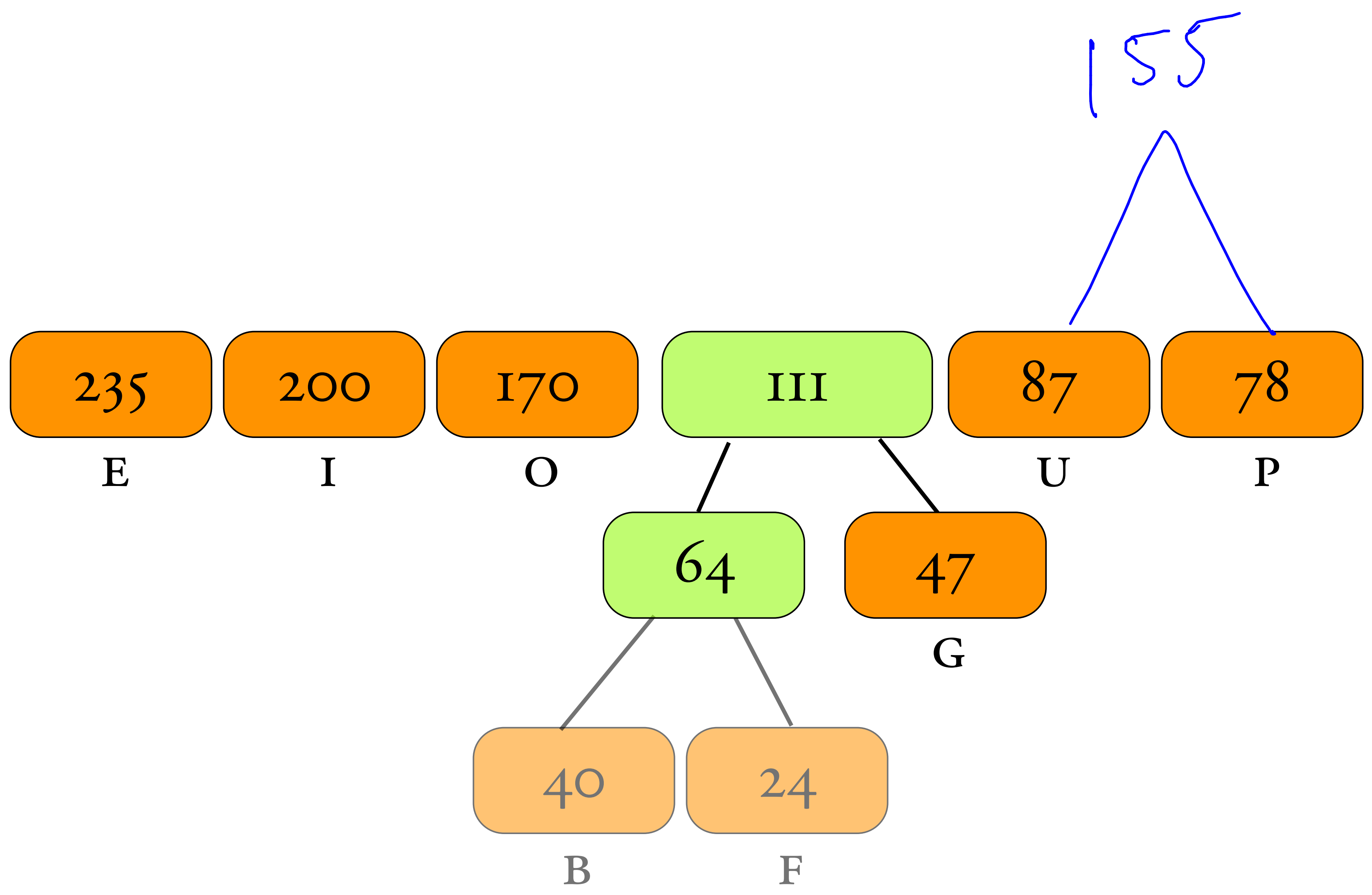

64
235
200
170
87
78
47
40
24
E
I
O
U
P
G
B
F

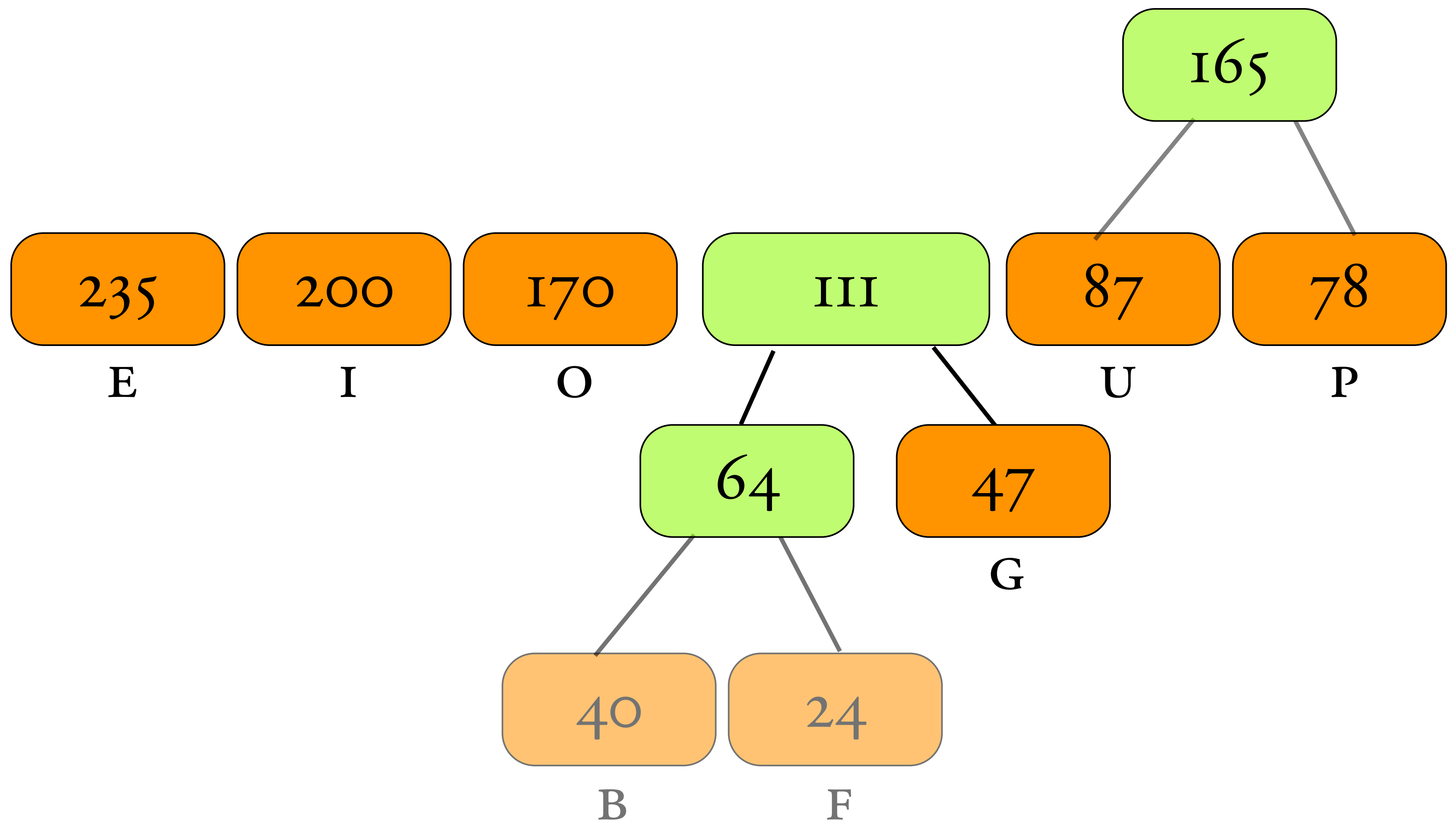

235
E
200
I
170
O
87
U
78
P
47
G
64
40
B
24
F

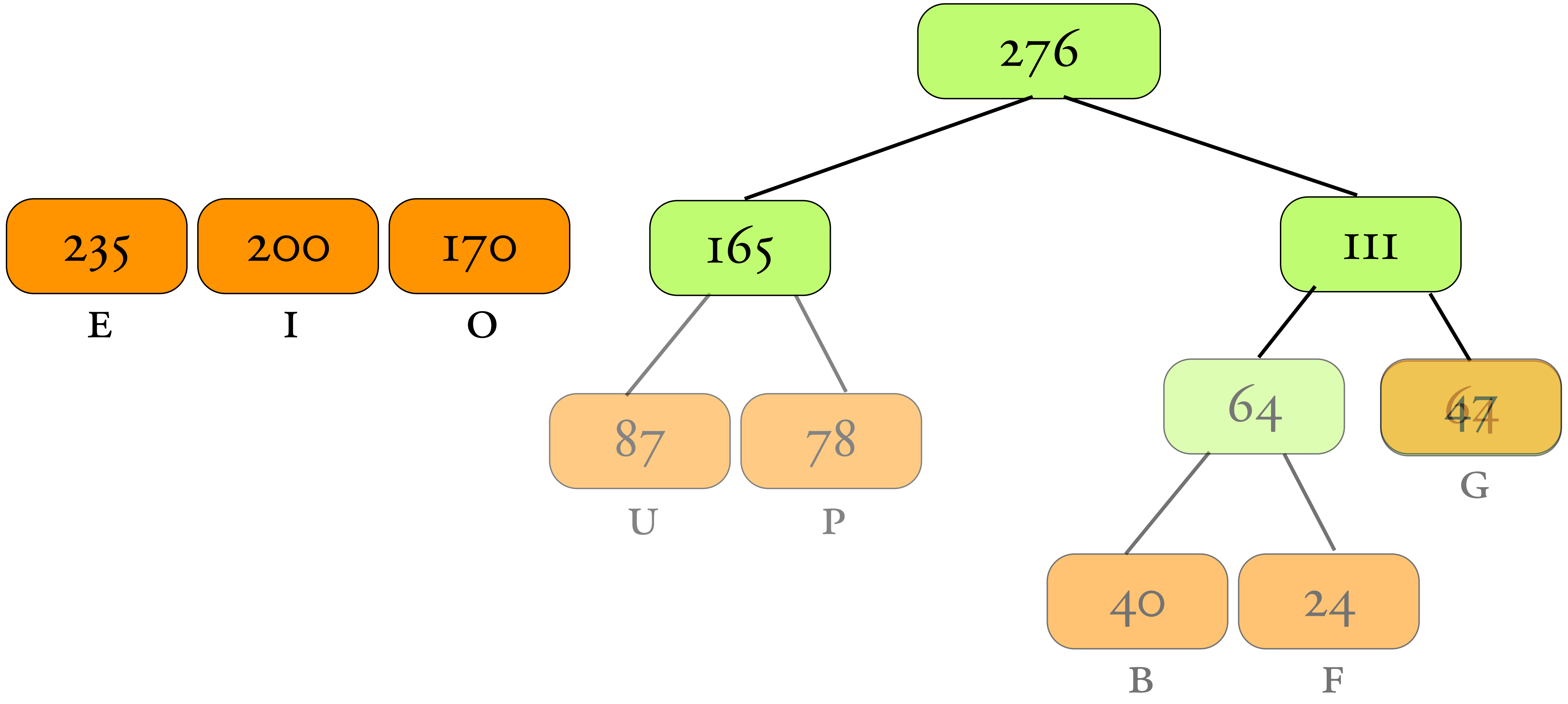

235
E
200
I
170
O
87
U
78
P
64
47
G
40
B
24
F

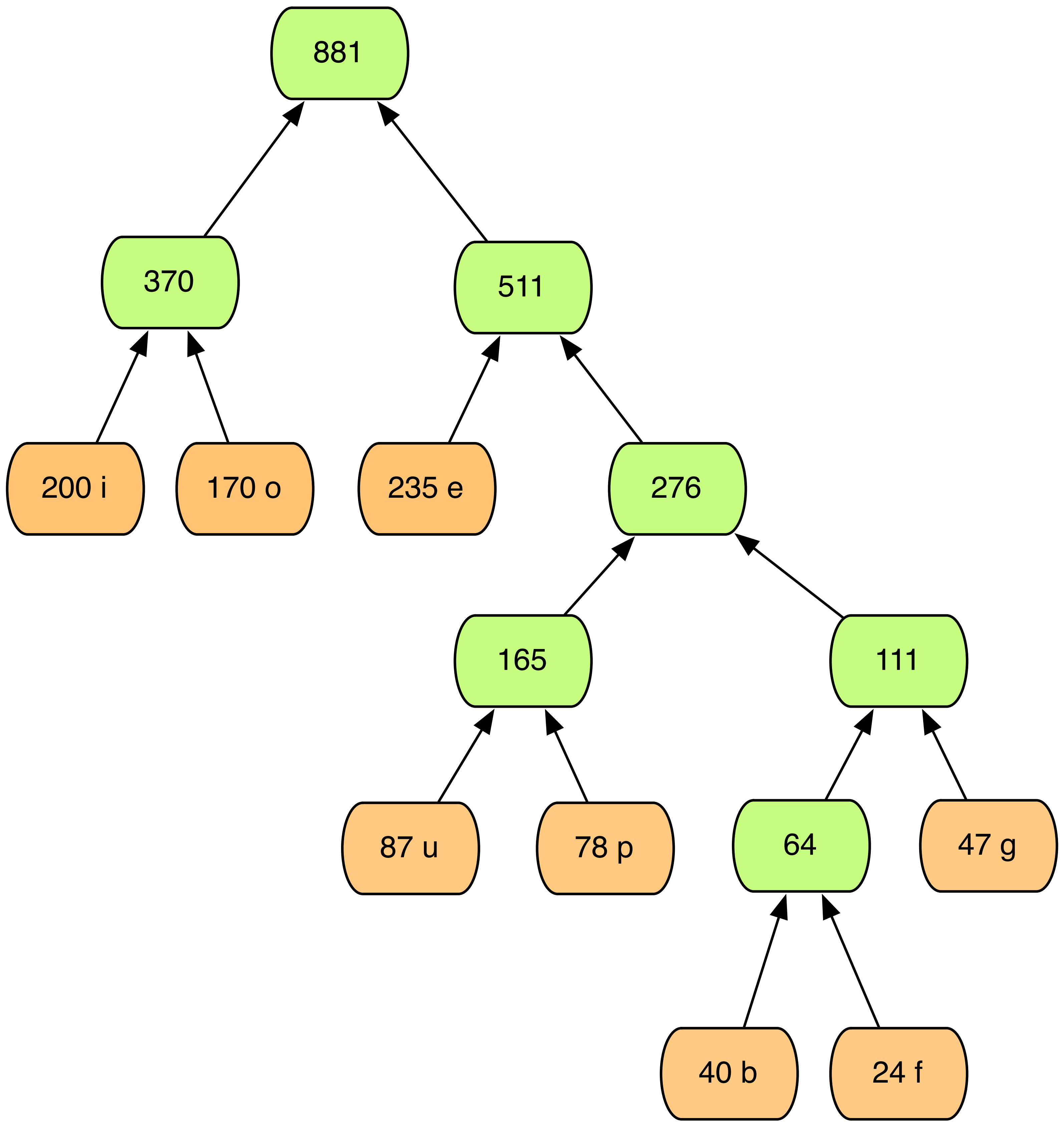

111
235
E
200
I
170
O
87
U
78
P
64
47
G
40
B
24
F

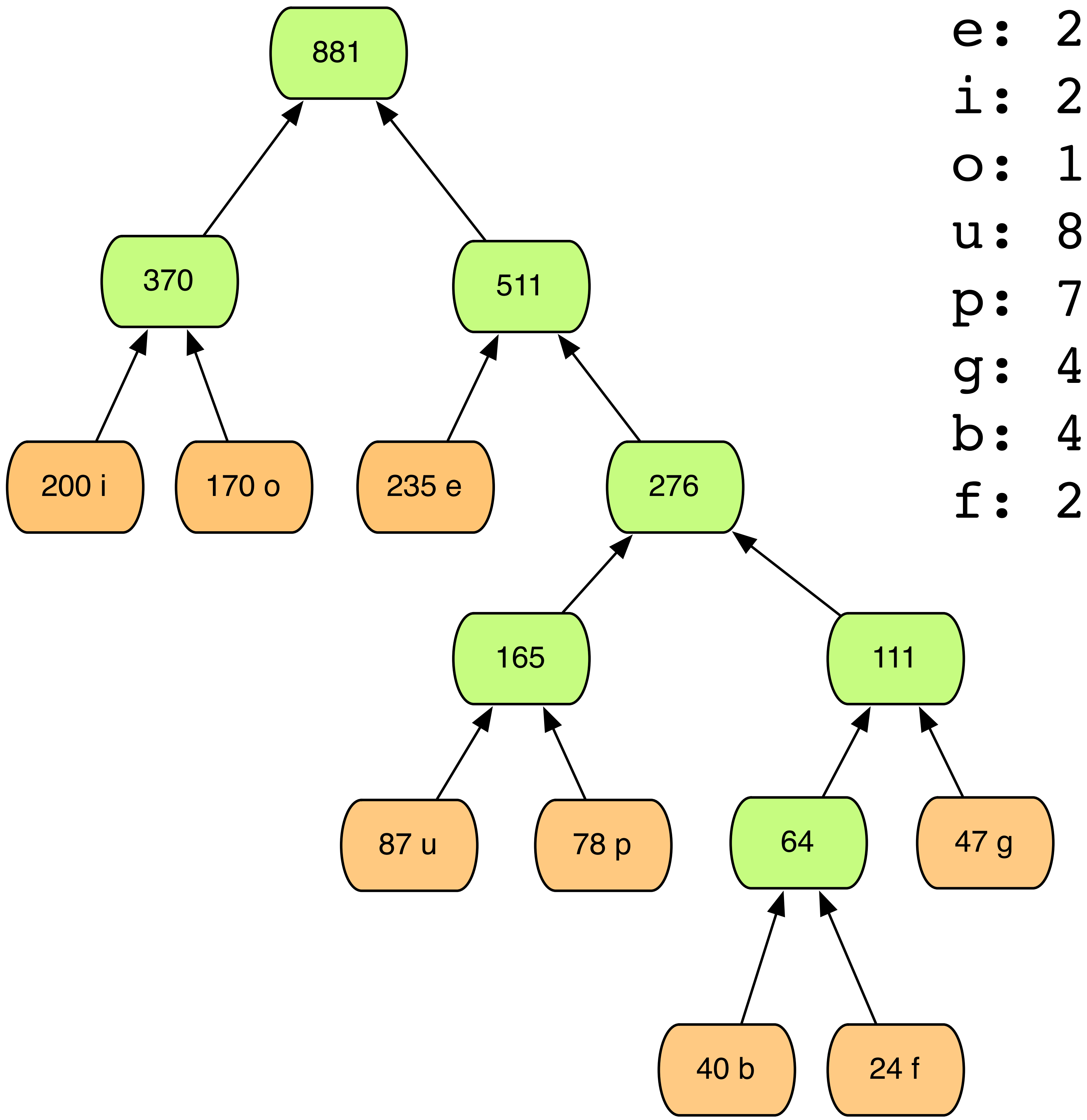

235
E
200
I
170
O
87
U
78
P
111
64
47
G
40
B
24
F


155
235
E
200
I
170
O
111
87
U
78
P
64
47
G
40
B
24
F

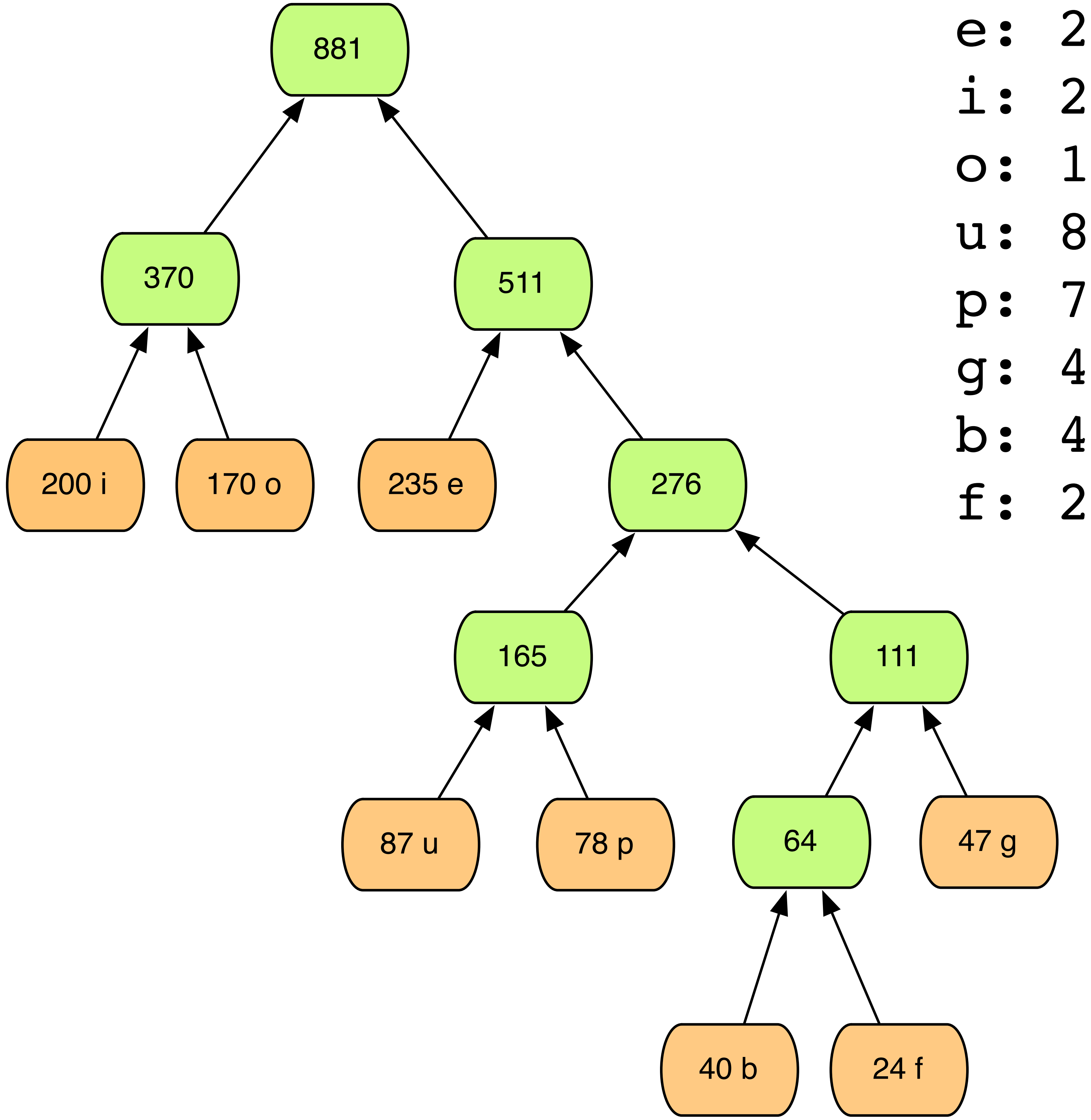

165
235
200
170
111
87
78
E
I
O
U
P
64
47
G
40
24
B
F


235
200
170
E
I
O
276
165
111
87
78
U
P
64
47
G
40
24
B
F


881
370
511
200 i
170 o
235 e
276
165
111
87 u
78 p
64
47 g
40 b
24 f




```
e: 235 01
i: 200 11
o: 170 10
u: 87  0011
p: 78  0010
g: 47  0000
b: 40  00011
f: 24  00010
```



e:	235	01
i:	200	11
o:	170	10
u:	87	0011
p:	78	0010
g:	47	0000
b:	40	00011
f:	24	00010

470
400
340
348
312
188
200
120
2378

versus

2643

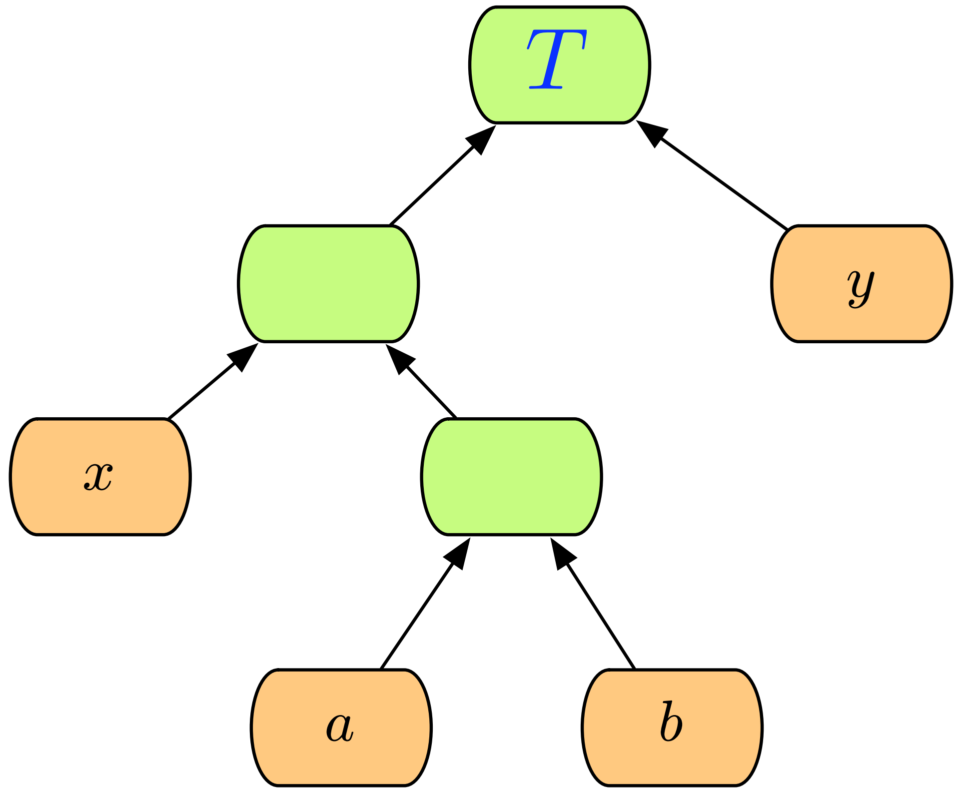
OBJECTIVE

EXCHANGE ARGUMENT

LEMMA:

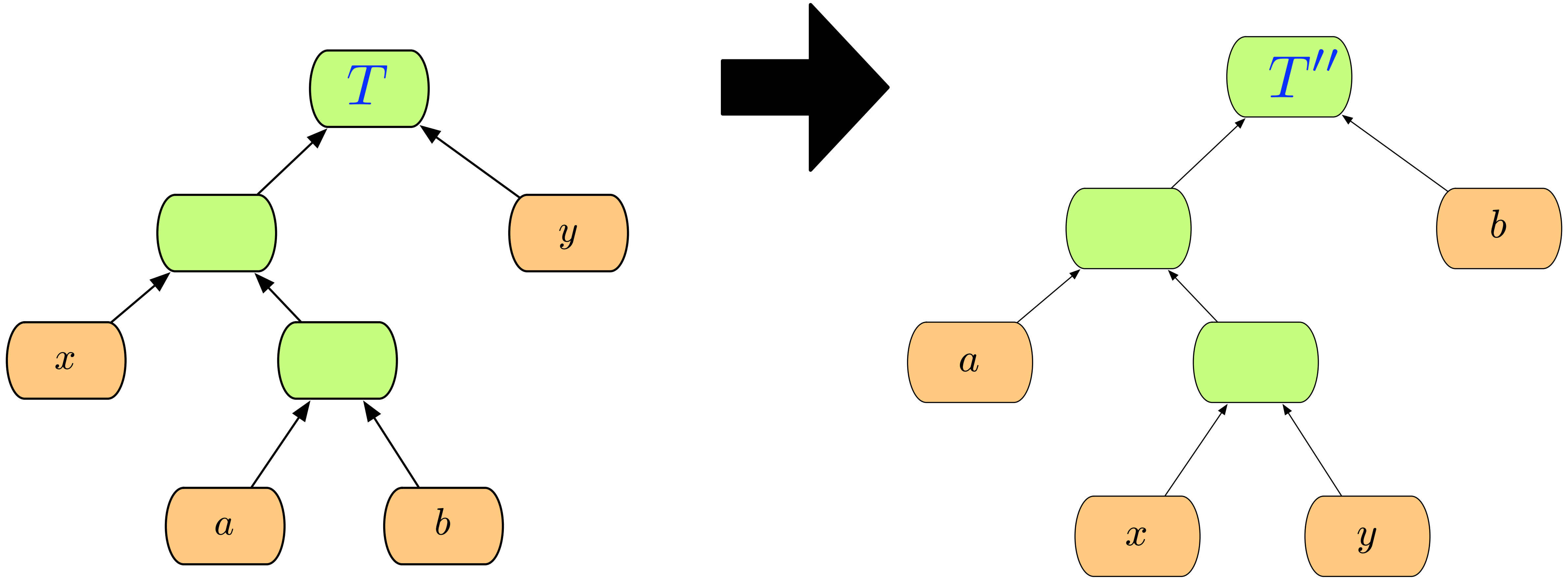
EXCHANGE ARGUMENT

LEMMA: Let $x, y \in C$ be characters with smallest frequencies f_x, f_y. There exists an optimal prefix code T'' for C in which x, y are siblings. That is, the codes for x, y have the same length and only differ in the last bit.



EXCHANGE ARGUMENT

LEMMA: Let $x, y \in C$ be characters with smallest frequencies f_x, f_y. There exists an optimal prefix code T'' for C in which x, y are siblings. That is, the codes for x, y have the same length and only differ in the last bit.



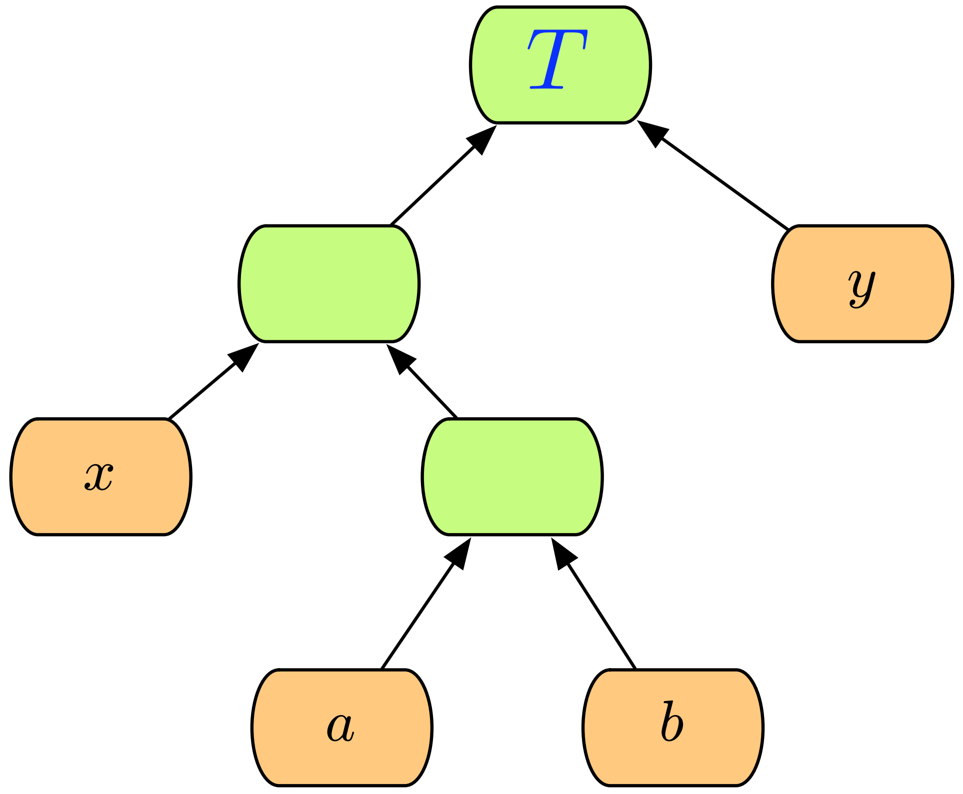
EXCHANGE ARGUMENT

LEMMA: Let $x, y \in C$ be characters with smallest frequencies f_x, f_y. There exists an optimal prefix code T'' for C in which x, y are siblings. That is, the codes for x, y have the same length and only differ in the last bit.

PROOF:

EXCHANGE ARGUMENT

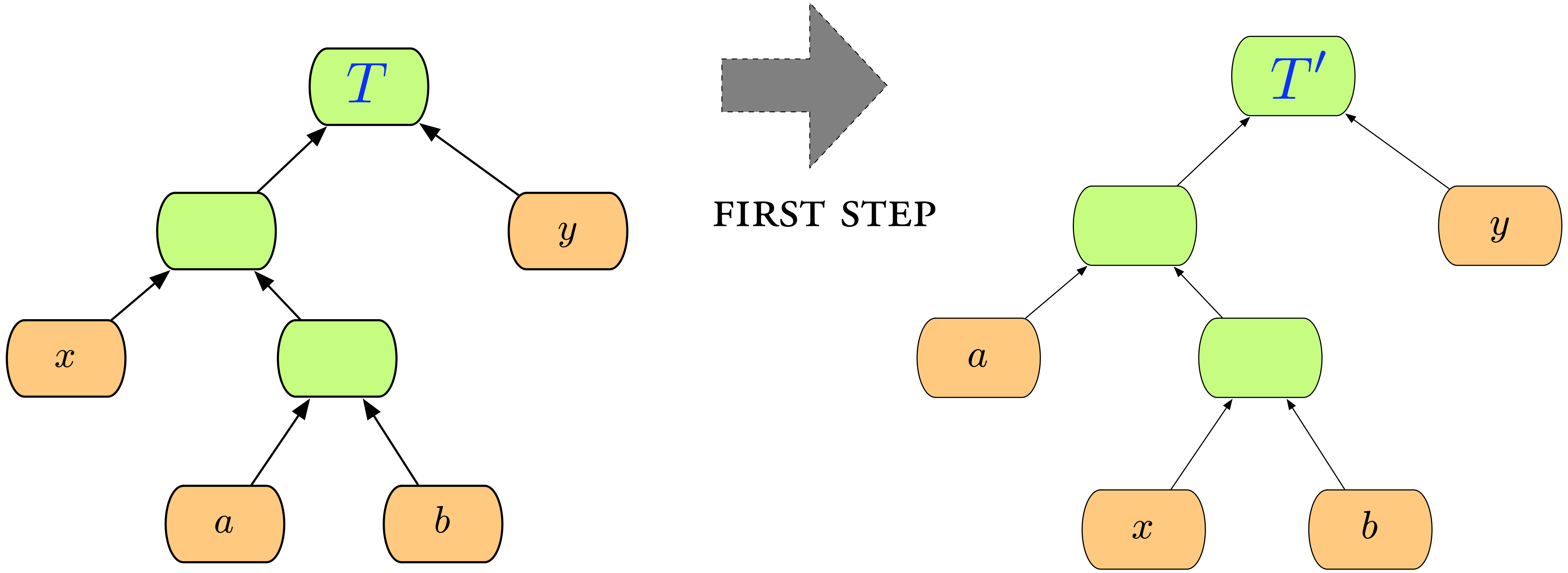
LEMMA: Let $x, y \in C$ be characters with smallest frequencies f_x, f_y. There exists an optimal prefix code T'' for C in which x, y are siblings. That is, the codes for x, y have the same length and only differ in the last bit.



FIRST STEP

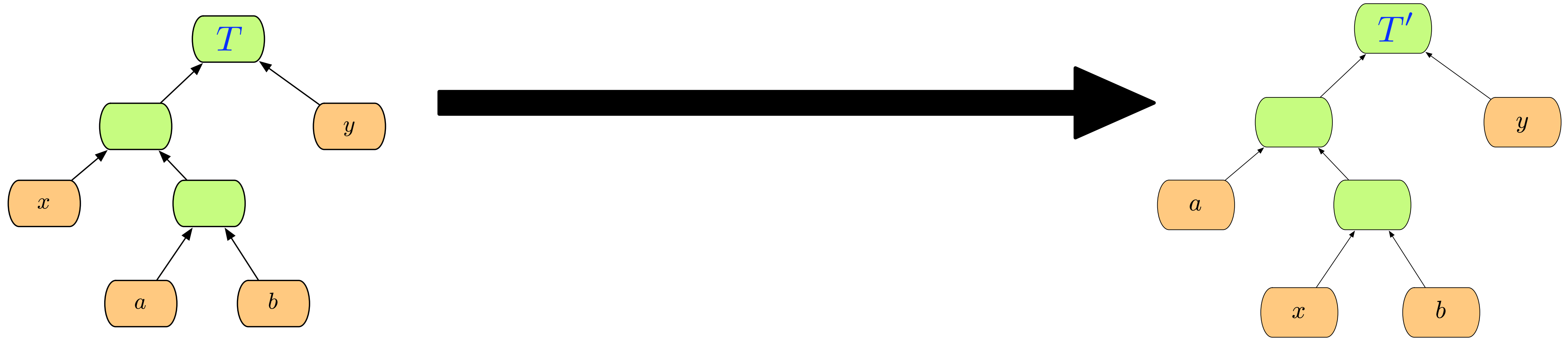
EXCHANGE ARGUMENT

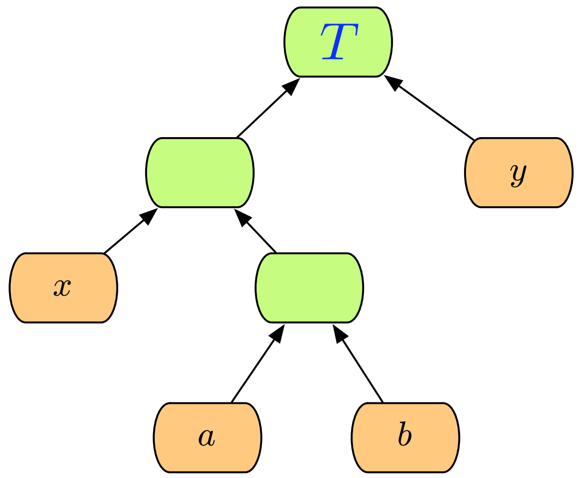
LEMMA: Let $x, y \in C$ be characters with smallest frequencies f_x, f_y. There exists an optimal prefix code T'' for C in which x, y are siblings. That is, the codes for x, y have the same length and only differ in the last bit.



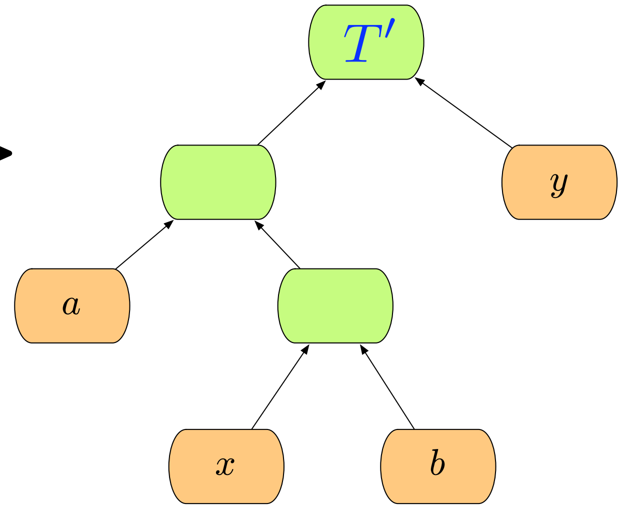
$$f_a \leq f_b \qquad f_x \leq f_a$$

$$f_x \leq f_y \qquad f_y \leq f_b$$


T
y
x
a
b
T'
y
a
x
b

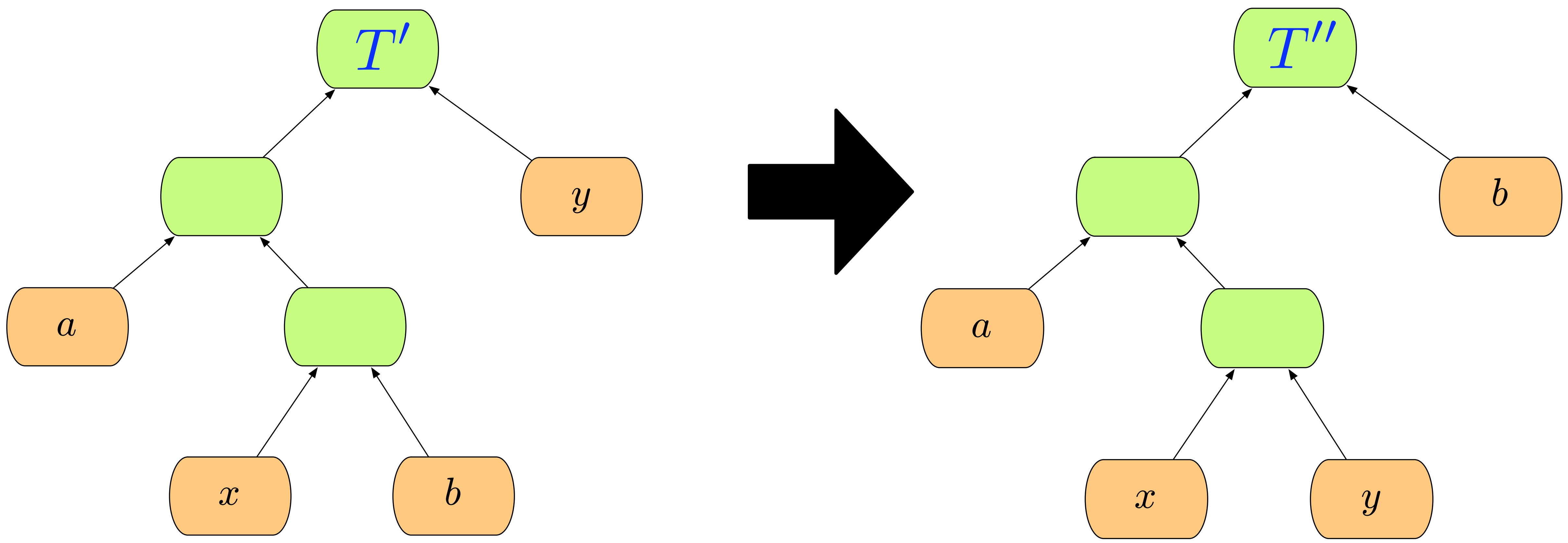


$$f_x \leq f_a$$

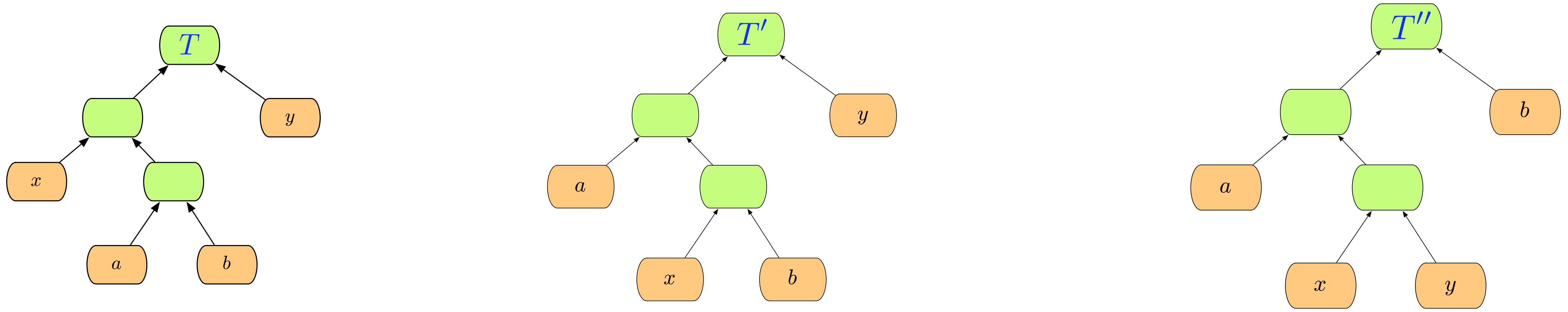


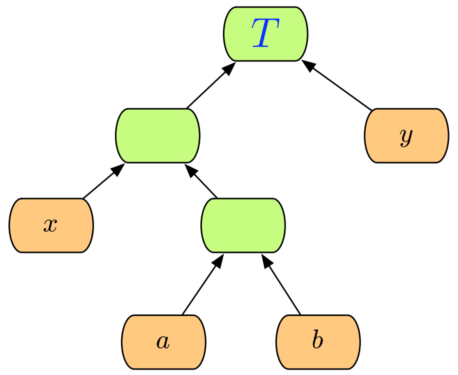
$$B(T) = \sum_c f_c \ell_c + f_x \ell_x + f_a \ell_a \qquad B(T') = \sum_c f_c \ell'_c + f_x \ell'_x + f_a \ell'_a$$

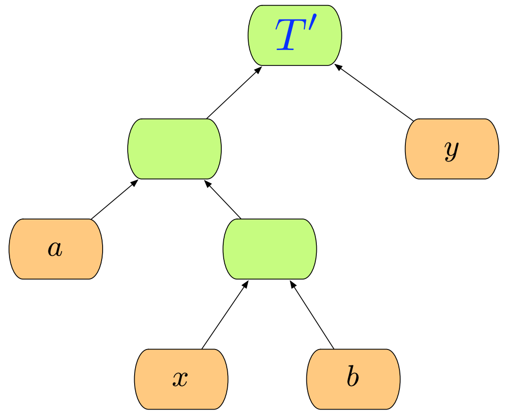
$$B(T) - B(T') \geq 0$$

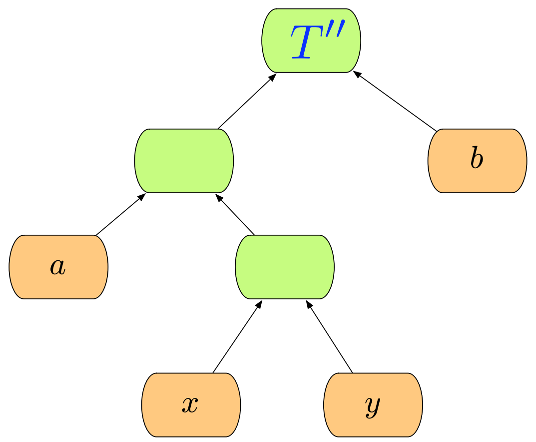


$$B(T') - B(T'') \geq 0$$


T
y
x
a
b
T′
y
a
x
b
T″
b
a
x
y





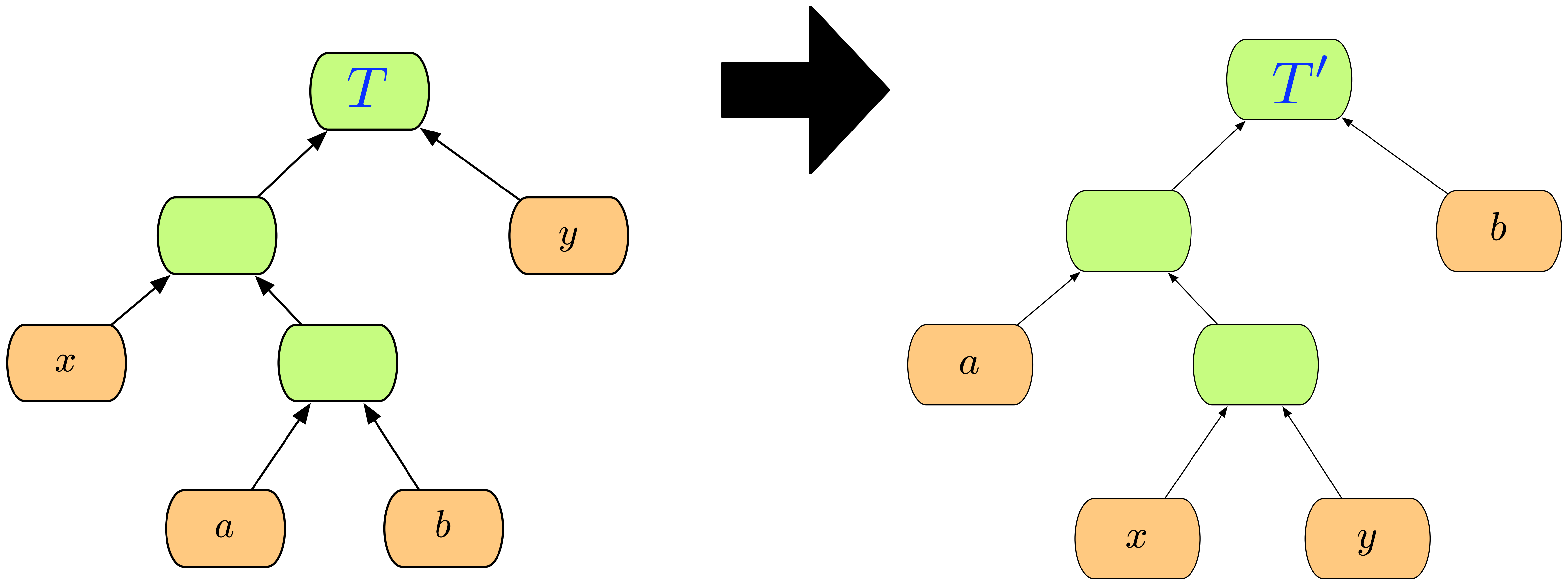


$$B(T) - B(T') \geq 0 \qquad B(T') - B(T'') \geq 0$$

T'' IS ALSO OPTIMAL

EXCHANGE ARGUMENT

LEMMA: Let $x, y \in C$ be characters with smallest frequencies f_x, f_y. There exists an optimal prefix code T'' for C in which x, y are siblings. That is, the codes for x, y have the same length and only differ in the last bit.



OPTIMAL SUB-STRUCTURE

								f_x	f_y
f_C	235	200	170	87	78	47	40	24

OPTIMAL SUB-STRUCTURE

							f_x	f_y
f_c	235	200	170	87	78	47	40	24

PROBLEM OF SIZE `n`

$f_{c'}$	235	200	170	87	78	47	64
							f_z

PROBLEM OF SIZE `n-1`

OPTIMAL SUB-STRUCTURE

f_c

						f_x	f_y
235	200	170	87	78	47	40	24

PROBLEM OF SIZE n

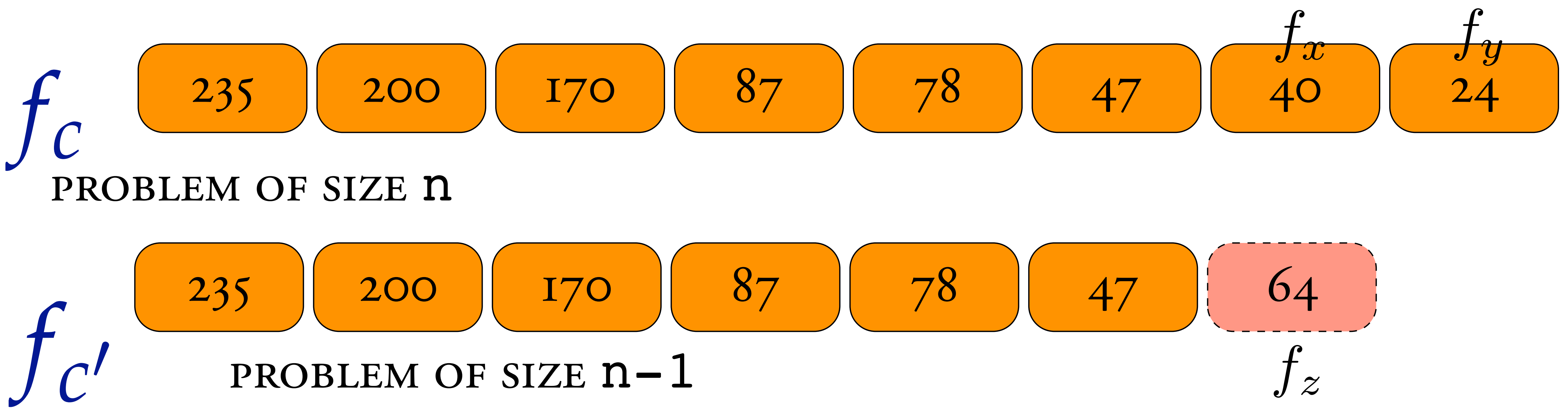
$f_{c'}$

235	200	170	87	78	47	64
						f_z

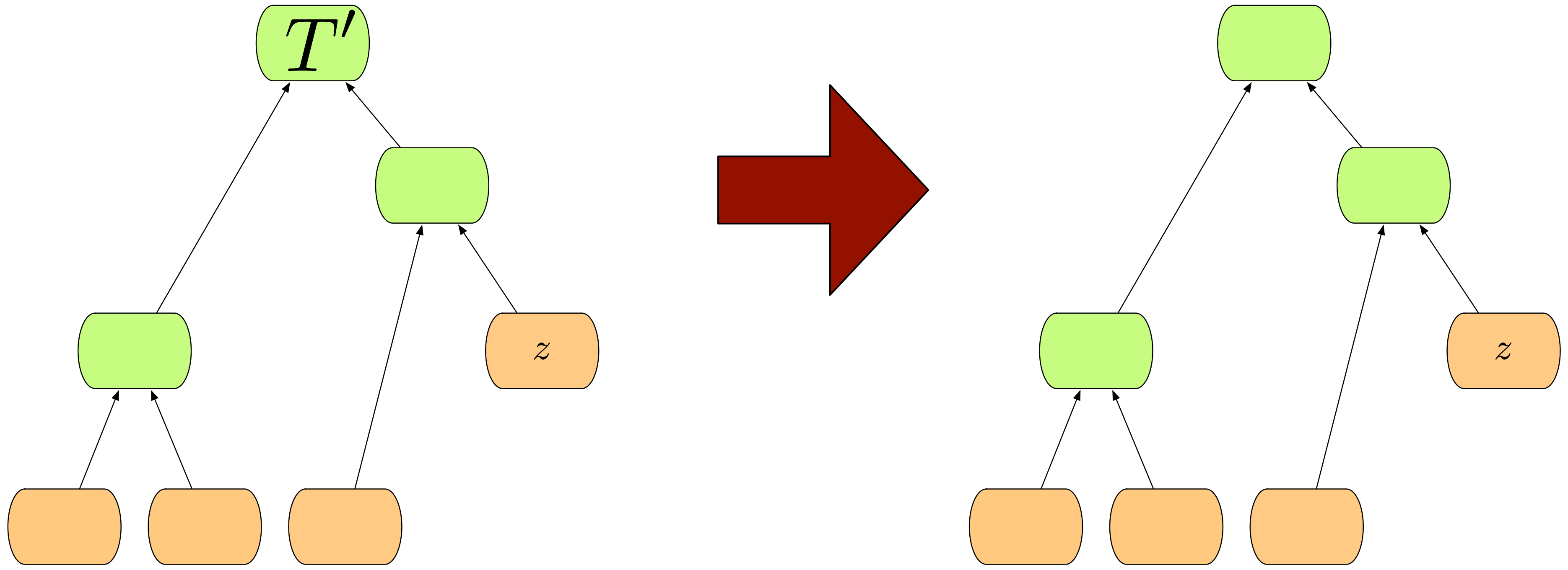
PROBLEM OF SIZE n−1

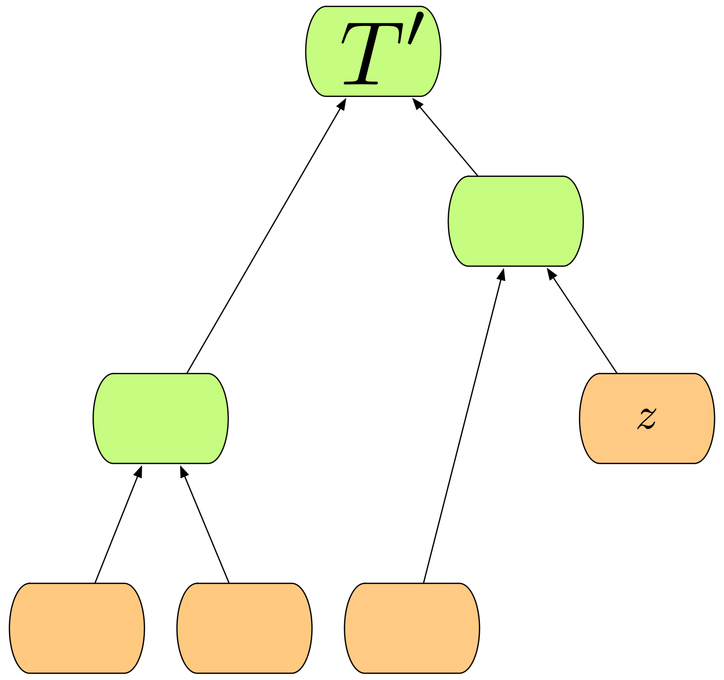
LEMMA:

OPTIMAL SUB-STRUCTURE

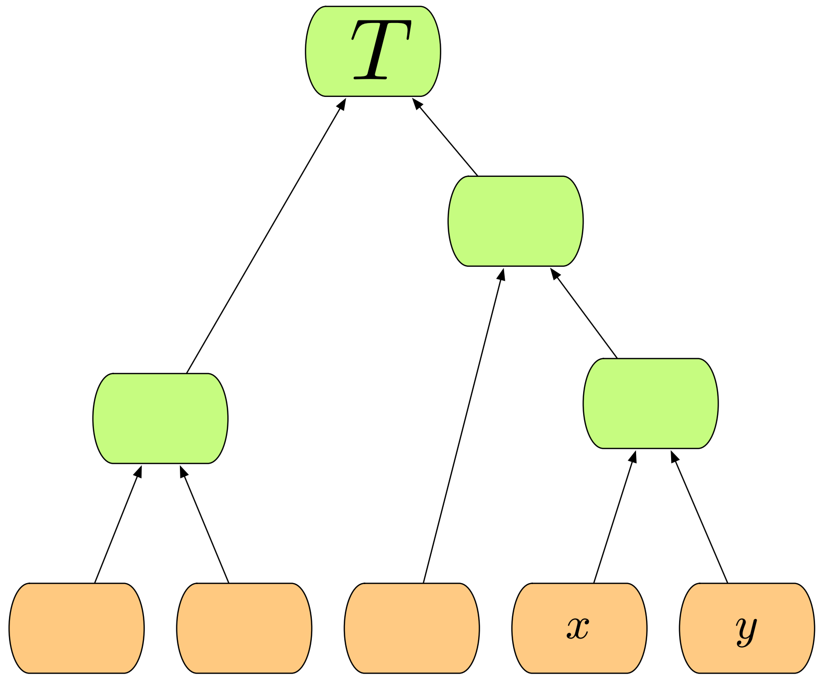


Lemma: The optimal solution for T consists of computing an optimal solution for T' and replacing the left z with a node having children x, y.

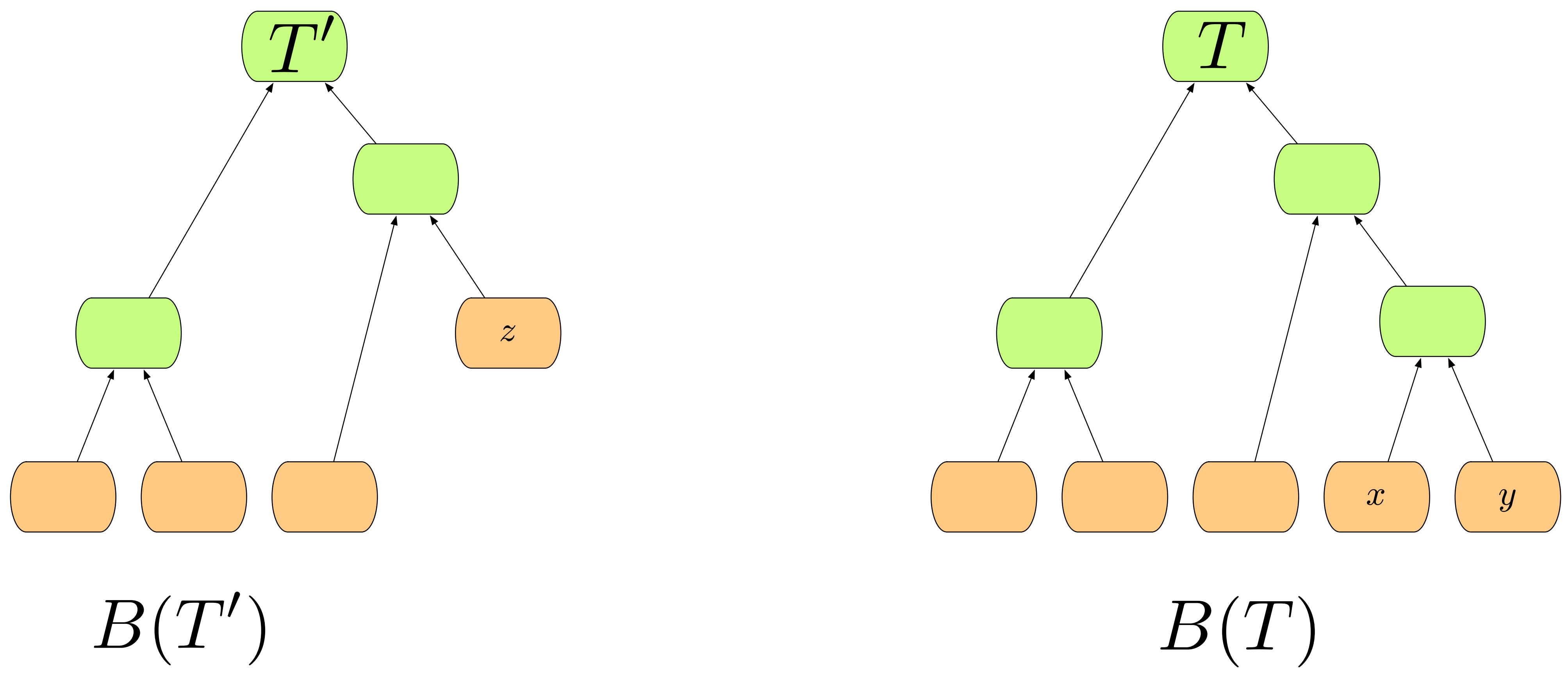

T'
z
z



$B(T')$



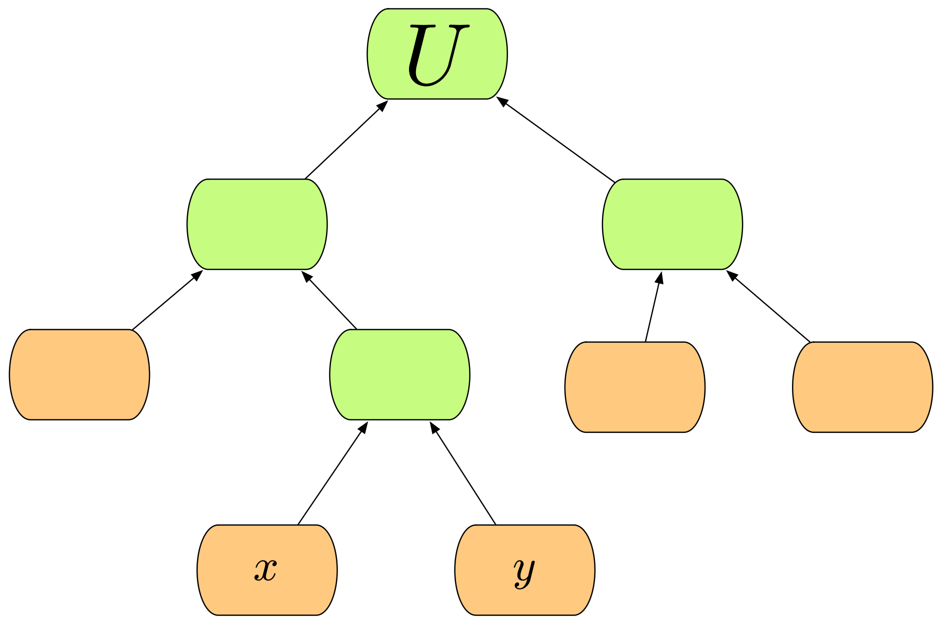
$B(T)$



$$B(T') = B(T) - f_x - f_y$$

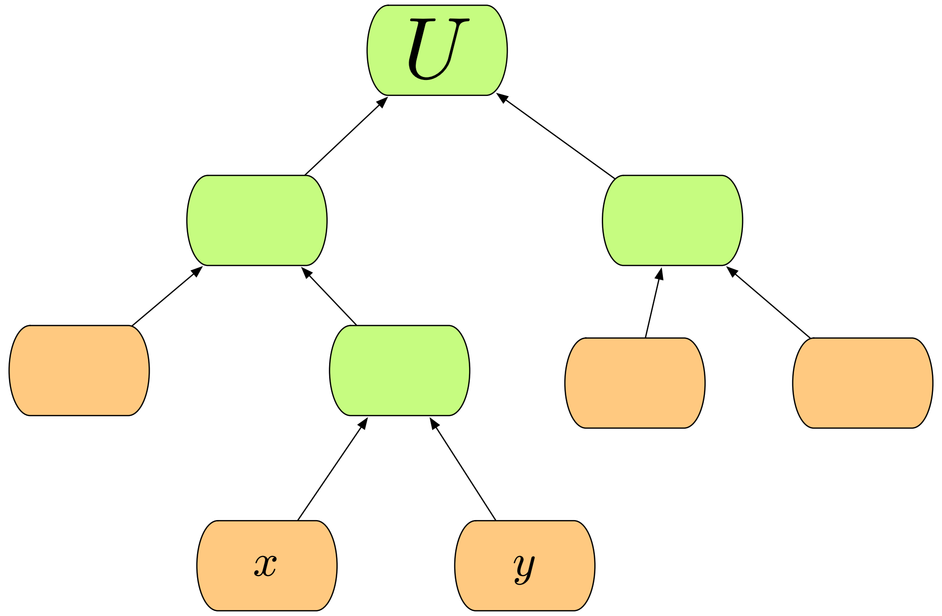
Suppose T is not optimal

Suppose T is not optimal

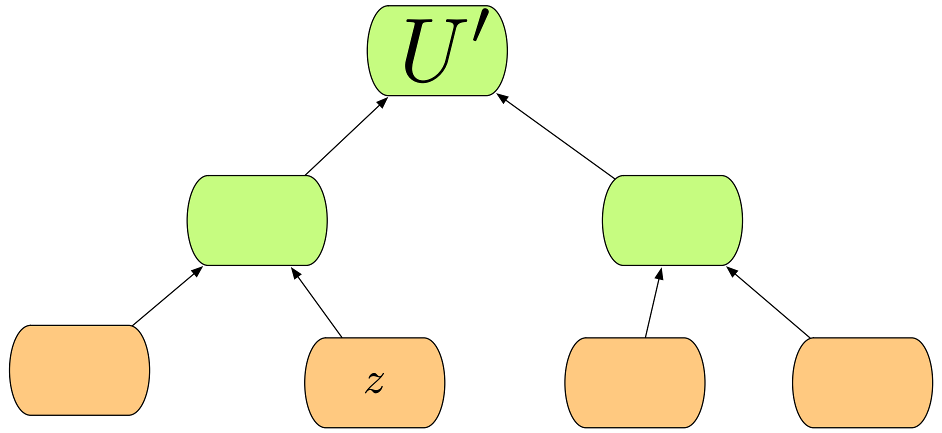


$$B(U) < B(T)$$

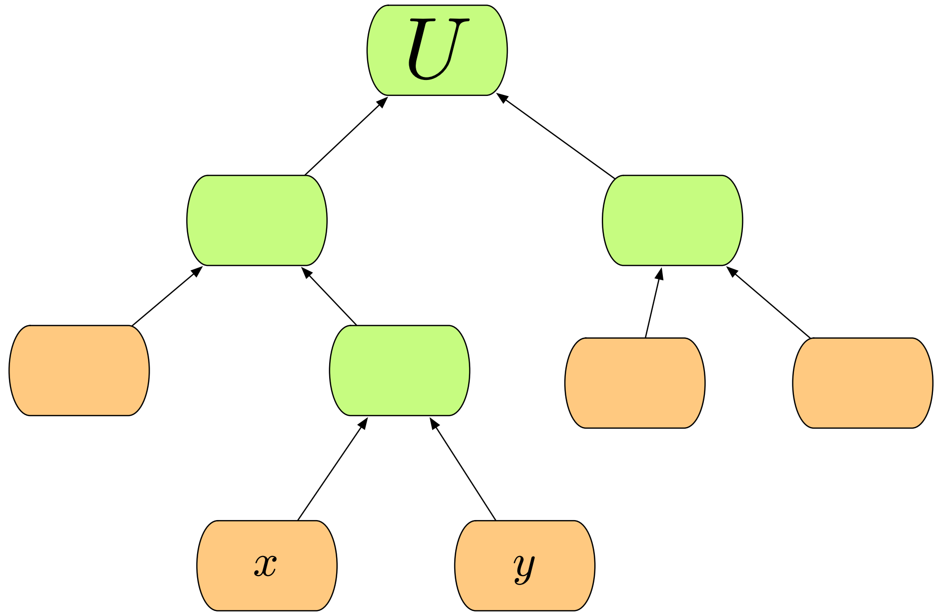
Suppose T is not optimal



$$B(U) < B(T)$$



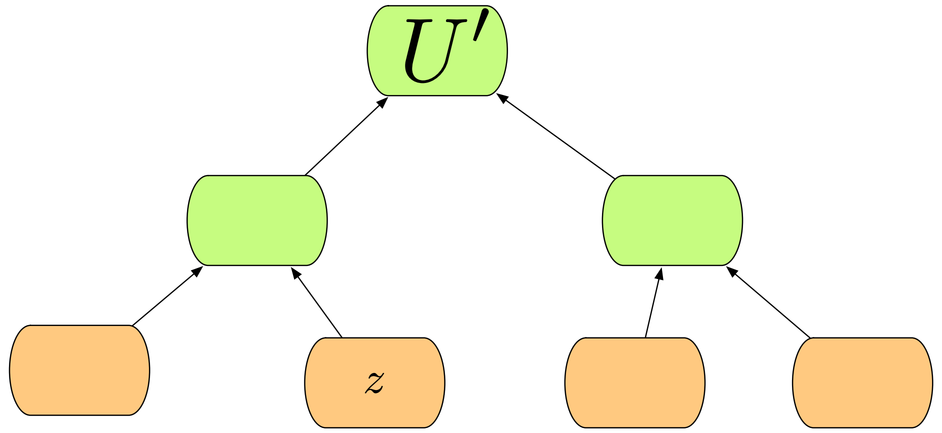
Suppose T is not optimal



$$B(U) < B(T)$$

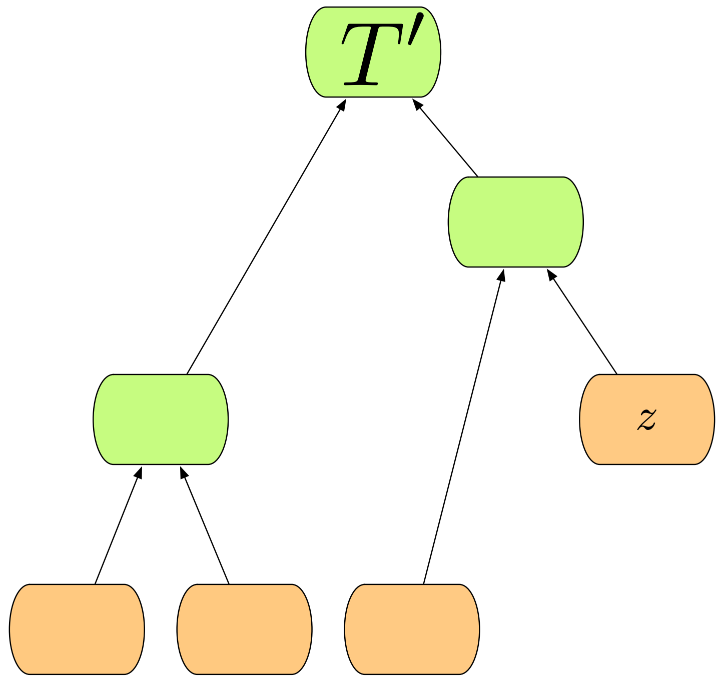
$$B(U') = B(U) - f_x - f_y$$

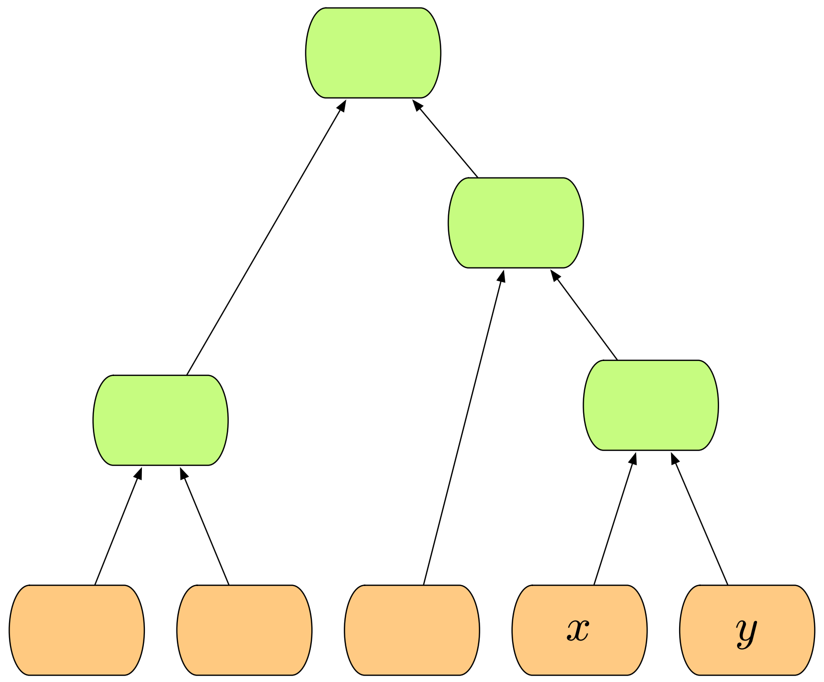
< B(T) - fx - fy



But this implies that B(T') was not optimal.

THERE FORE





SUMMARY OF ARGUMENT