Hash tables

M "compartments"

N elements

$$h : K \text{ (set of keys)} \longrightarrow \overset{0 \,..\, M-1}{[1 \,..\, M]}$$

- $h \to$ in $0 \,..\, M-1$
- equally distributed over $0 \,..\, M-1$
- easy to compute

ex 1: keys are integers

$$\boxed{h(k) = ck \;\%\; M}$$

ex 2: reals in $[0, 1)$

$$h(k) = \text{int}(Mk)$$

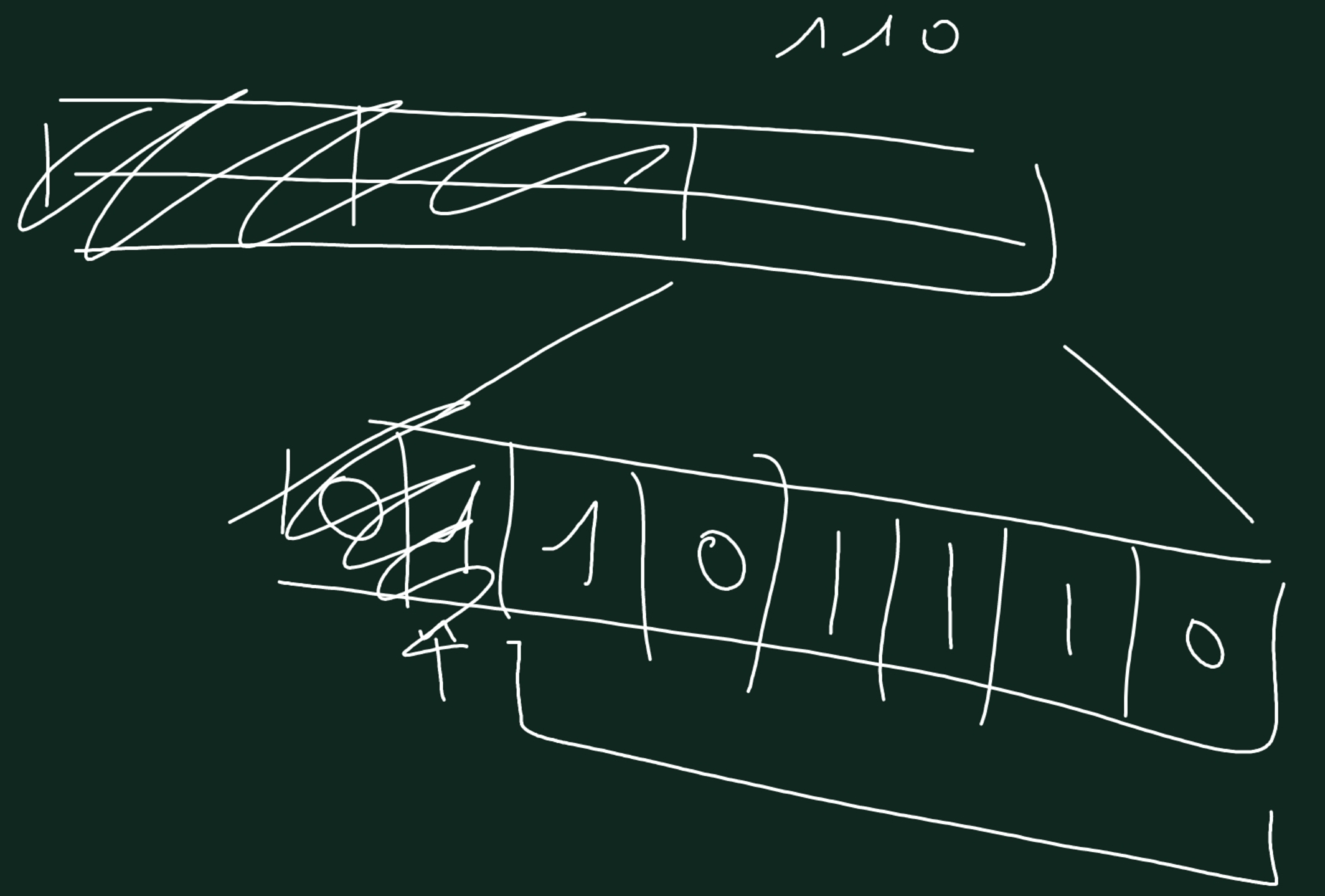
counter-example:

keys: 3 char-words (24 bits)   ~~$n = 64$~~ n should be prime

$h(k) = k \% n$

dos

96

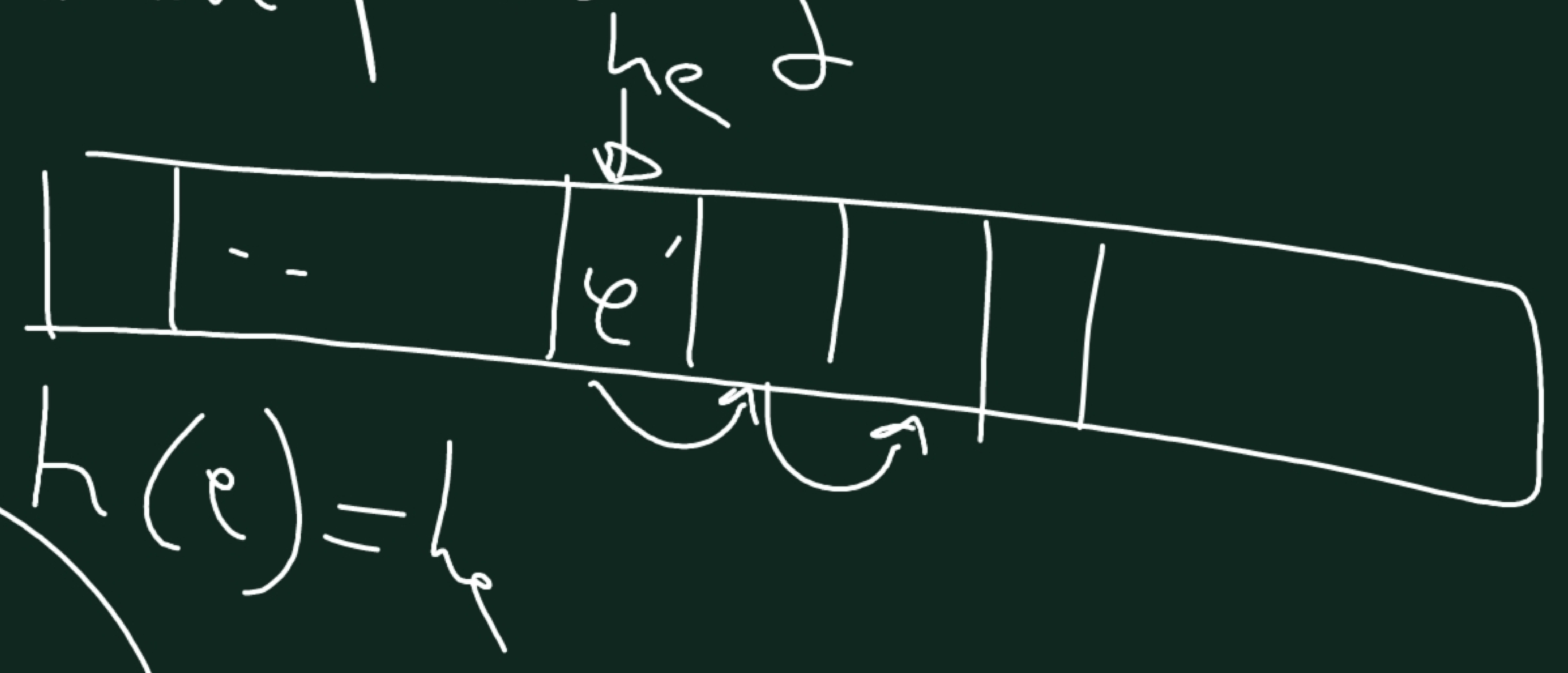


card$(K) \gg n$ sometimes

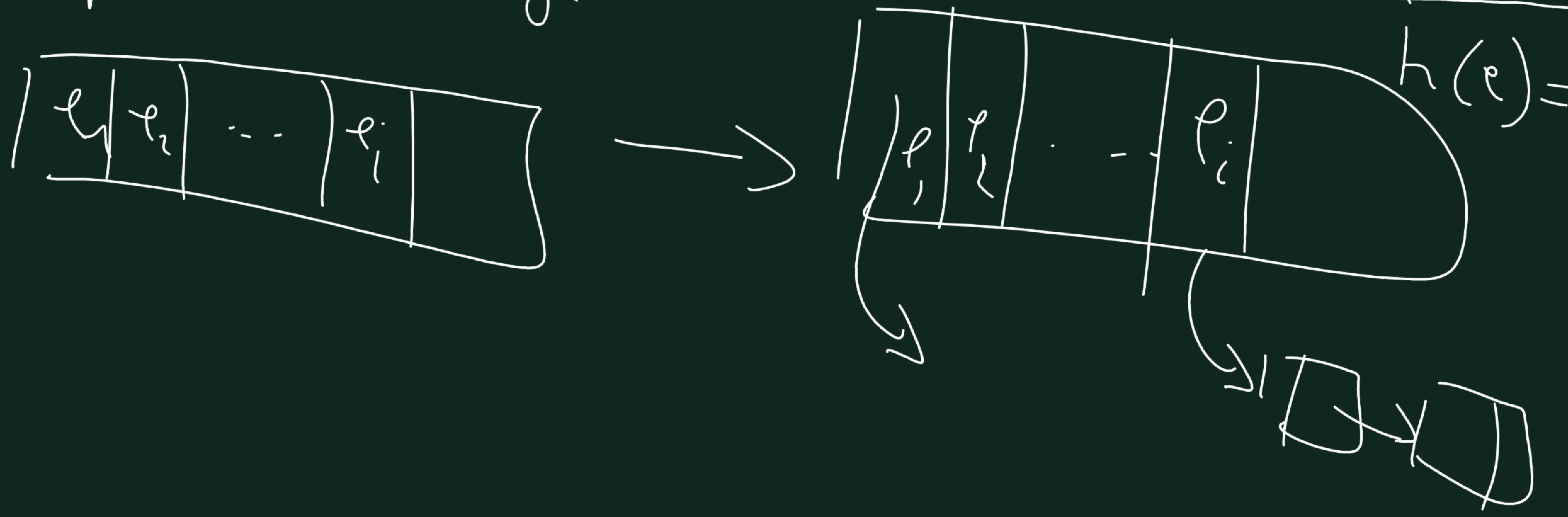
$\longrightarrow$ collision: 2 different keys have the same h-value

2 approaches to cope with this issue:

linear probing



separate chaining



def: load factor of a hash-table: $\frac{N}{M}$

- separate chaining
  - $N < M$
  - $N > M$ ok
- linear probing
  - $N < M$
  - $N \ll M$

insert "E A S Y Q U T I O N" in an initially empty table

ex: separate chaining

keys: letters $(0 \to 25)$

$M = 5$

$h(k) = 11k \,\%\, 5$

k	E	A	S	Y	Q	U	T	I	O	N
$h(k)$	4	0	3	4	1	0	4	3	4	3

HT:

- 0 → A U
- 1 → Q
- 2 →
- 3 → S I N
- 4 → E Y T O

load factor: $\frac{10}{5} = 2$ → length of lists

search is $\Theta(1)$

linear probing:

	A	S	E	R	C	H	I	N	G	X	M	P
h	7	3	9	9	8	4	11	7	10	12	0	8

M: 13

0					5					10		
G	X	M	S	H	P		A	C	E	R	I	N

0
0
0
1
0
0
0
5
3
2
2
10

→ we should always have a load factor << 1

load factor	1/2	2/3	3/4	9/10
successful search	1.5	2	3	5.5
unsuccessful	2.5	5	8.5	55.5

exercise

linear probing, $M = 2N$

1) What is the best (worst) placement for elements in the table?

2) ______ ____ average length of clusters in both cases?

3) average number of probes (unsuccessful search), in the best-case, is independent from N and M

4) average number in worst case is $\simeq N/4$

best case | e | av | e | av | - - - - - - | e | av | $\bar{l} = 1/2$

worst case | e | - - - - - - | e | av | - - - - - - | av | $\bar{l} = N/2$

avg number $$\frac{1}{2N}\left(N + N-1 + N-2 + \dots 1\right) = \frac{N(N+1)}{2 \times 2N} \sim \frac{N}{4}$$

best: $$\frac{1}{2N}\left(2+1+2+1+\dots+2+1\right) = \frac{3N}{2N} = \frac{3}{2}$$

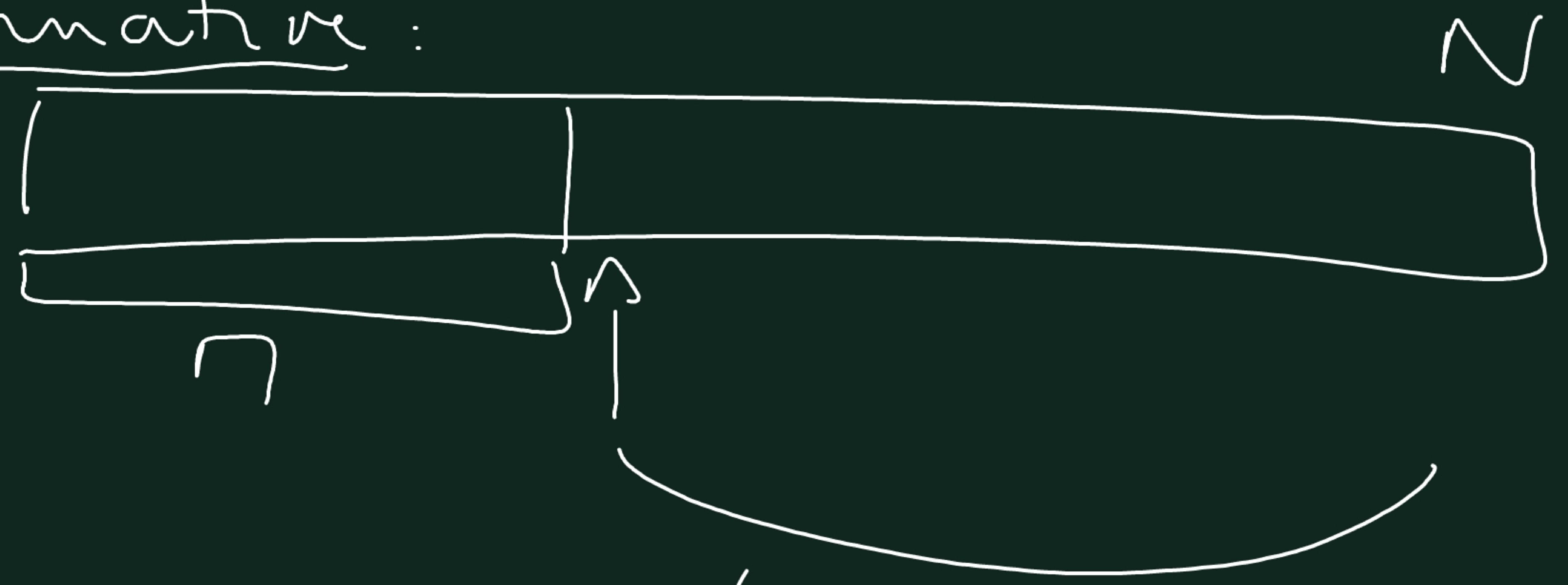
Comparison AVL / H_table for search

HT: $m = 20000$, separate chaining

N	$t(AVL)$ (ms)	$t(HT)$
5000	22.88	5.25
10000	32.88	5.5
20000	42.88	6
40000	52.88	7
80000	62.88	9
160000	72.88	13
320000	82.88	21
640000	92.88	37
1280000	102.88	69
2560000	112.88	133
5120000	122.88	261

Pb: find highest m values in an array of size N
(is it worth sorting the array? Why?) $\Theta(N \log N)$

<u>alternative</u>:



$$(N-m) \times m \sim Nm = \Theta(N)$$

extension to ADT List

with orderedInsert operation:

$L: (A \quad D \quad G \quad K)$

$orderedInsert(L, F) \rightarrow (A \quad D \quad F \quad G \quad K)$

ADT extension List

Use Element ($\leq$)

Operations

$orderedInsert: List \times Element \rightarrow List$

Axioms

A_1 $orderedInsert(new, e) \equiv cons(e, new)$

A_2 $orderedInsert(l, first(l)) \equiv l$ (if duplicates forbidden)

A_3 $e \leq first(l) \Rightarrow orderedInsert(l, e) \equiv cons(e, l)$

A_4 $orderedInsert(l, e) \equiv cons(first(l), orderedInsert(next(l), e))$

0.5	1	0.2	0.2

1.8 €? no

1.9 €? Y

from (s)

canPay? wallet and price → Y / N

1) choose more params if needed

2) write code of function

3) evaluate its complexity in the worst case

```
canPay(w, price, from) {
    if (price == 0) return true
    if (from == w.length - 1) return (price == w[from])
    if (canPay(w, price - w[from], from + 1)) return true
    return canPay(w, price, from + 1)
}
```

initial call: from = 0
(first elt)

param: N, number of coins
unit op: −

$C(1) = 1$

$C(n) = 2C(n-1) + 1$

$$C(1) = 1$$

$$C(n) = 2C(n-1) + 1$$

$$C(n) + 1 = 2\left(C(n-1) + 1\right)$$

$$D(n) = C(n) + 1$$

$$D(1) = 2$$

$$D(n) = 2D(n-1)$$

$$= 2^2 D(n-2)$$

$$\vdots$$

$$= 2^i D(n-i)$$

$$i = n-1$$

$$D(n) = 2^{n-1} \cancel{D(1)} \times 2$$

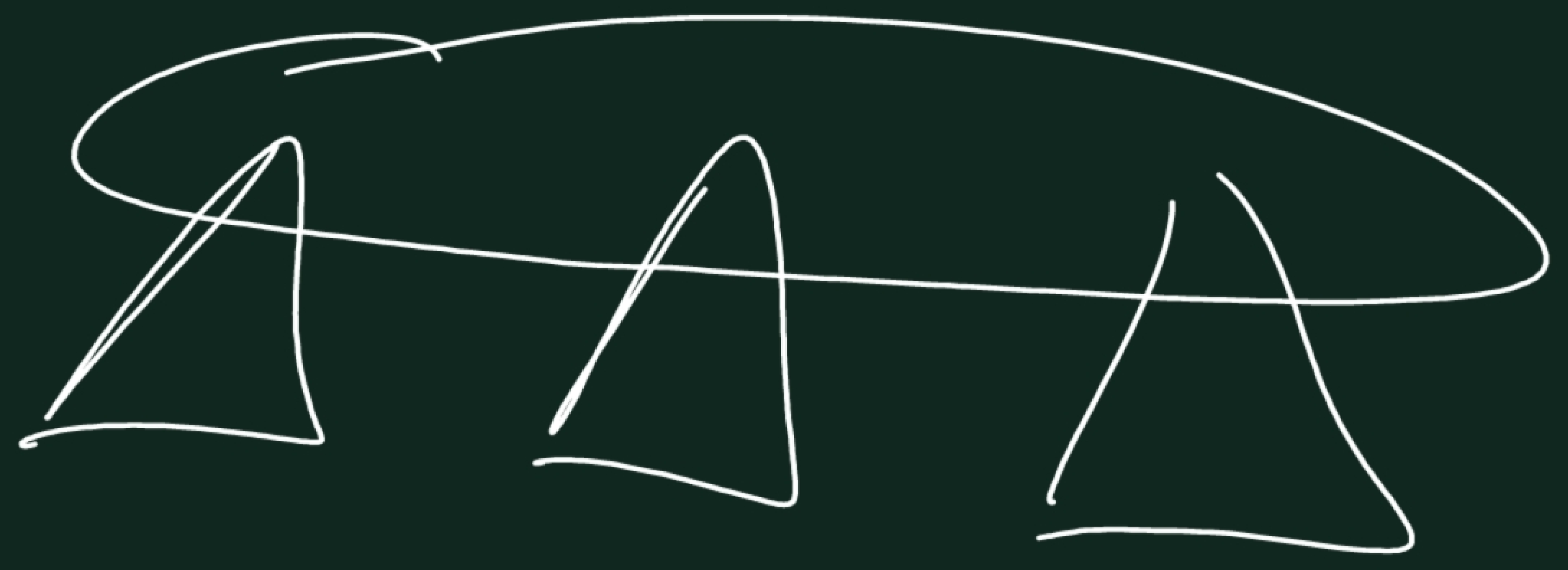
$$= 2^n$$

$$\Rightarrow C(n) = 2^n - 1$$

$$= 4 \left(2^n\right)$$

define an extension to ADT Tree (& Forest) with the maxChildren operation:

(ex. maxChildren(t) = 5)



extension ADT Tree, Forest

Operations

$\quad maxChildren : Tree \longrightarrow Integer$

$\quad maxChildrenF : Forest \longrightarrow Integer$

Axioms

A_1 $maxChildren(new(e, f)) \equiv max(nTrees(f), maxChildrenF(f))$

A_2 $maxChildrenF(newF) \equiv 0$

A_3 $maxChildrenF(addTree(f, t, n)) \equiv max(maxChildren(t), maxChildrenF(f))$

extension to ADT List with a subList(l, from, to) operation (total)

(hyp: indexes out of list bounds are ignored)

ADT extension List

Operations

subList : $List \times Integer \times Integer \longrightarrow List$

Axioms

A_1 sublist(new, from, to) $\equiv$ new

A_2 to < from => sublist(l, from, to) $\equiv$ new

A_3 sublist(l, 1, to) $\equiv$ cons(first(l), sublist(rest(l), 1, to - 1))

A_4 sublist(l, from, to) $\equiv$ sublist(rest(l), from - 1, to - 1)

l(A B C D)

sublist(l, 1, 2)

A_3 cons(A, sublist((B C D), 1, 1))

A_3 con(A, cons(B), sublist((C D), 1, 0)))

$$l = (A \quad B \quad C \quad D)$$

$$\text{sublist}(l, 2, 3)$$

$$A_4 \quad \text{sublist}((B \quad C \quad D), 1, 2)$$

$$A_3 \quad \text{cons}(B, \text{sublist}((C \quad D), 1, 1))$$

$$A_3 \quad \text{cons}(B, \text{cons}(C, \text{sublist}((D), 1, 0)))$$

$$A_2 \quad \text{cons}(B, \text{cons}(C, \text{nil})) \equiv (B \quad C)$$

| | | | | ..... | N sticks

2 subpiles

each step, score is incremented by the product of sizes of the subpiles just created

Q: best possible score? $S(n) = ?$

| | | | | 0

| | | | | 6

| | | | | 8

| | | | | 9

| | | | | 10

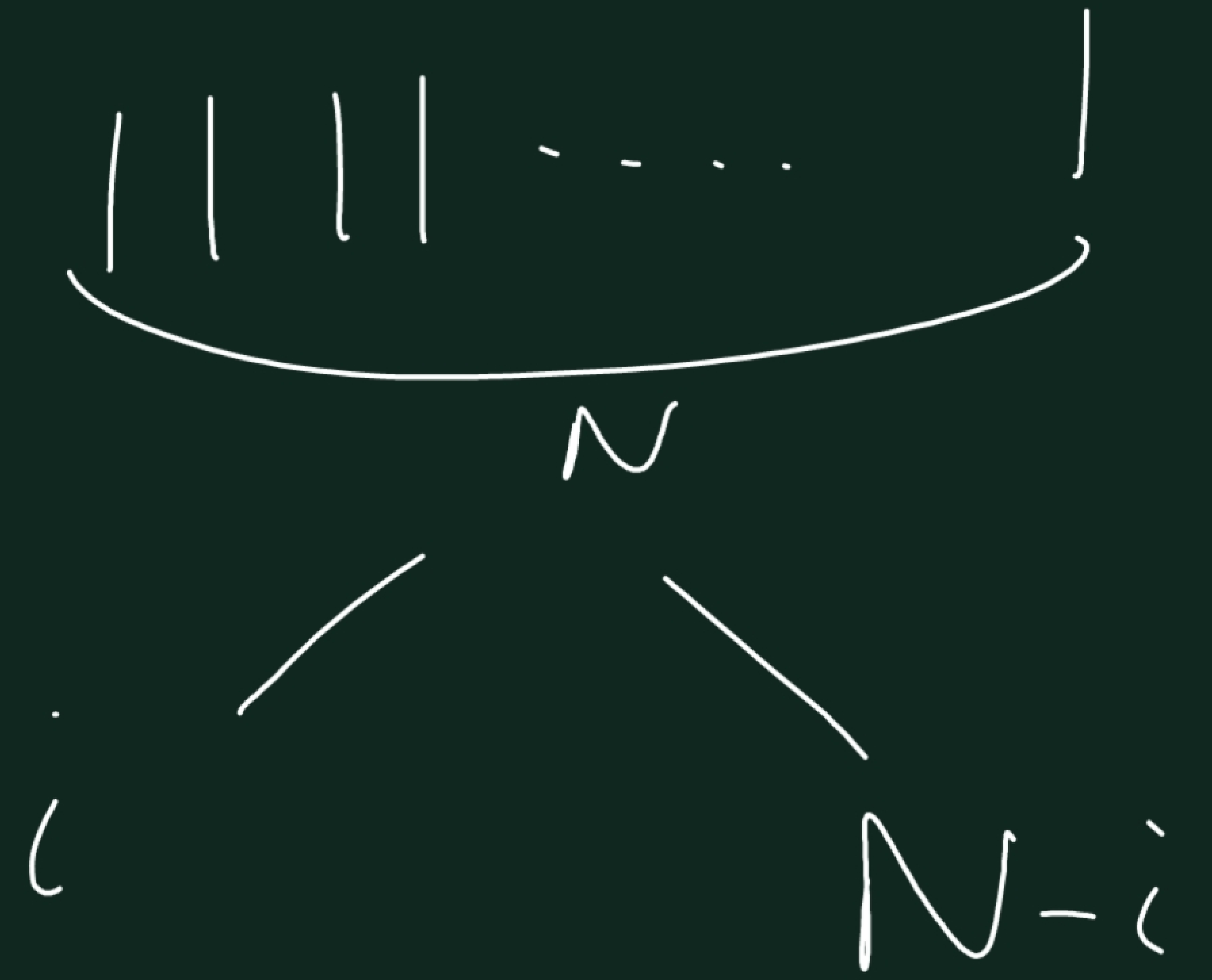
guess: $S(n) = \frac{n(n-1)}{2}$

$P(n): S(n) = \frac{n(n-1)}{2}$

$P(1), P(2) \ldots P(6)$ are T

n	$S(n)$
1	0
2	1
3	3
4	6
5	10
6	15

hyp: $\forall i < n,\ p(i) = T \iff \forall i < N,\ S(i) = \frac{i(i-1)}{2}$



Score: $S(i) + S(N-i) + i(N-i)$

$$= \frac{i(i-1)}{2} + \frac{(N-i)(N-i-1)}{2} + i(N-i)$$

$$= \frac{\cancel{i^2}}{2} - \cancel{\frac{i}{2}} + \frac{N^2 - 2iN + \cancel{i^2} - N + \cancel{i}}{2} + \cancel{i(N-i)}$$

$$= \frac{N^2 - N}{2} = \frac{N(N-1)}{2}$$

qed