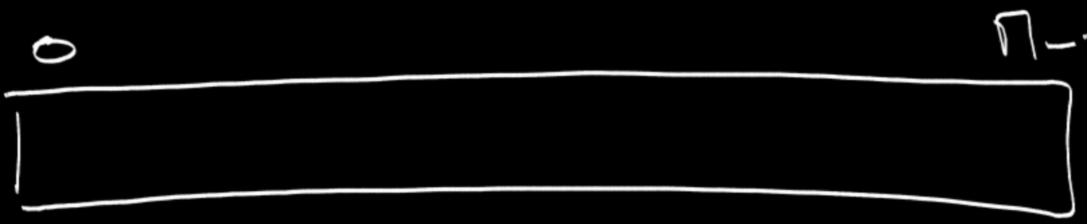
Hash tables Π compartments



$h : K \longrightarrow [0 \,..\, \Pi - 1]$

$k \longmapsto 0 \leq h(k) \leq \Pi - 1$

- in $0 \,..\, \Pi - 1$
- as equally distributed in $0 \,..\, \Pi - 1$ as possible
- fast to compute

ex 1 : keys are real numbers in $[0,1]$

$$h(k) = \lfloor M k \rfloor$$

ex 2 : keys are just integers

$$h(k) = ck \% n$$

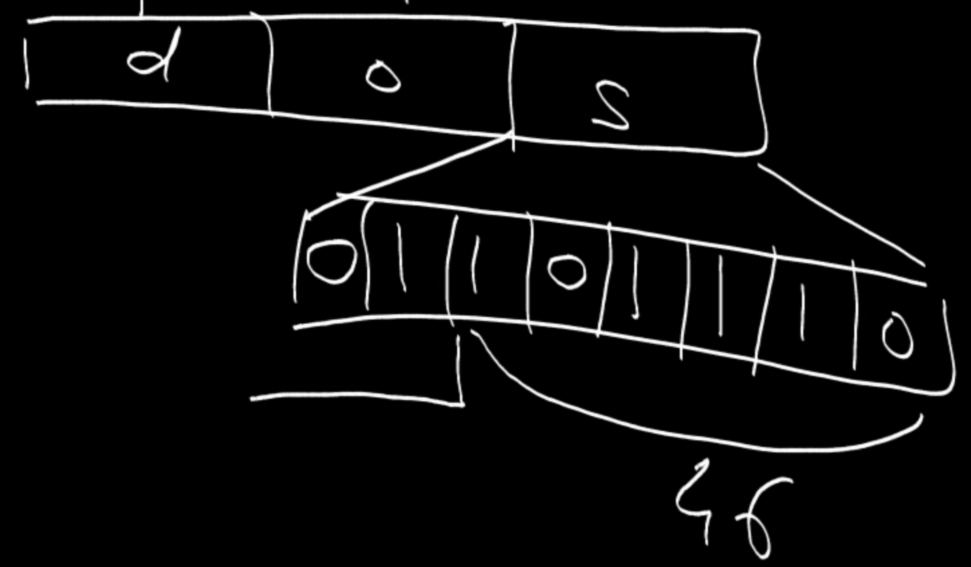
→ should be prime!

counter-example : keys are 3-letter words "dos"

ASCII : 110

$h(k) = k \% 64$

$h(\text{"dos"}) =$



Π compartments, N objects

N elements

load factor: $\frac{N}{\Pi}$

$O_1 \to k_1$

$\neq \quad \neq \quad \to h(k_1) = h(k_2)$? collision

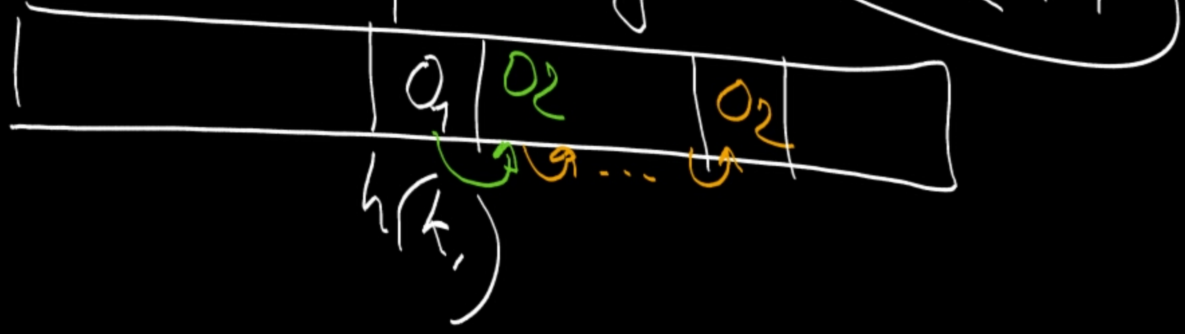
$O_2 \to k_2$

2 strategies to cope with the collision problem

separate chaining — $N \geq \Pi$ ok

elements of table are lists of objects

linear probing — $N < \Pi$



example of separate chaining:

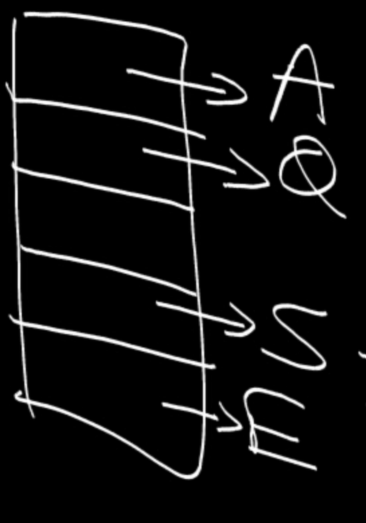
keys are letters (value: order in alphabet)

$n = 5$

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

	E	A	S	Y	Q	U	T	I	O	N
k	4	0	18	24	16	20	19	8	14	13
h	4	0	3	4	1	0	4	3	4	3

HT



load factor: $\frac{10}{5} = 2$ (avg length of list)

example of linear probing:

keys are letters, $n = 13$

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

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

0 1 2 3 4 5 6 7 8 9 10 11 12

exercise: linear probing, table is half full: $M = 2N$

Q1: what are the best/worst distributions for elements?

Q2: what is the avg length of clusters in both cases?

Q3: prove that avg number of probes for an unsuccessful search is independent from N and M in the best case

Q4: prove that, in the worst case, it is $\simeq N/4$

best | e | air | e | air | e | ..... | e | air |

$\overline{\text{length}} = \frac{1}{2}$

worst | e | e | ..... | e | air | air | ... | air |

$\overline{\text{length}} = \frac{N}{2}$

1st: $\frac{1}{2N}(1+0+1+0+\cdots+1+0) = \frac{1}{2}$

best: $\frac{1}{2N}(\underbrace{2+1+2+1+\cdots+2+1}_{3N}) = \frac{3}{2}$

worst: $\frac{\frac{1*N+1*0}{2}}{N} = \frac{N}{2}$

$\frac{1}{2N}(\underbrace{N+1+N+N-1+\cdots+2+1}_{\sum_{i=1}^{N+1} i} \underbrace{\cdots+1}_{N}) = \frac{1}{2N}\left(\frac{(N+1)(N+2)}{2} + N - 1\right)$

$= \frac{1}{2N}\left(\sum_{i=1}^{N+1} i + N - 1\right)$

$\approx \frac{N^2}{2 \cdot 2N} = N/4$

Comparing AVLs to Hash Tables

$M = 20\,000$, separate chaining

N	t(AVL)	t(HT)
5000	27.88	5.25
10000	32.88	5.5
20000	47.88	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

a an array of size N. Find the highest n values in a (is it worth sorting the array?) ④ $(N \log N)$

$$(N-M)*n = Nn - n^2$$

$$= ④(N)$$

extension to ADT List with an "orderedInsert" operation
(no duplicates)

orderedInsert((A C f k), D) → (A C D f k)

ADT extension List

Use Element (≤)

Operations

orderedInsert : List × Element → List

Axioms

A_1 orderedInsert(new, e) ≡ cons(e, new)

A_2 orderedInsert(l, first(l)) ≡ l

A_3 e ≤ first(l) ⇒ orderedInsert(l, e) ≡ cons(e, l)

A_4 orderedInsert(l, e) ≡ cons(first(l), orderedInsert(rest(l), e))

1	0.5	0.2	0.2

1.6€ ? no canPay(w, amount) → y / n

1.2€ ? yes

Q_1: add necessary params if any (index of 1st coin considered)

Q_2: write pseudo code for canPay

Q_3: complexity of canPay in worst case?

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

param : N, number of coins
init op : —

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

$C(1) = 1$

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

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

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

$D(1) = 2$

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

$= 2\left(2\,D(n-2)\right)$

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

$= 2^3 D(n-3)$

$\vdots$

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

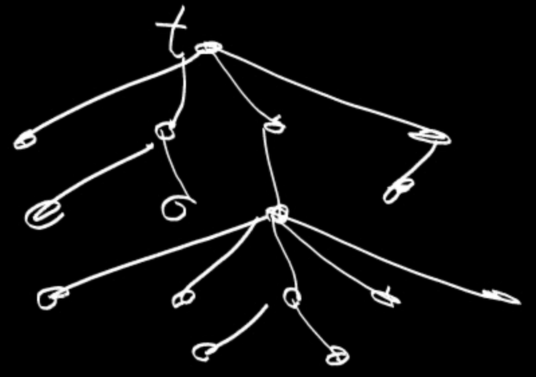
$= 2^{n-1} D(1)$

$= 2^n$

$\boxed{C(n) = 2^n - 1} = \Theta(2^n)$

define an extension to Tree (Forest) with a maxChildren operation

maxChildren(t) = 5



ADT extension Tree, Forest

Operations

maxChildren : Tree → Natural

maxChildrenF : Forest → Natural

Axioms

A_1 maxChildren(new(e, f)) ≡ max(nTrees(f), maxChildrenF(f))

A_2 maxChildrenF(newf) ≡ 0

A_3 maxChildrenF(addTree(f, t, n)) ≡ max(maxChildren(t), maxChildrenF(f))

define an extension to ADT List
with a sublist(l, from, to) operation
and make it total

hyp: indexes out of list bounds are ignored

ADT extension List

Operation

sublist : List × Natural × Natural → List

Axioms

A_1 sublist(new, from, to) ≡ new

A_2 to < from ⟹ sublist(l, from, to) ≡ new

A_3 sublist(l, 1, to) ≡ cons(first(l), sublist(rest(l), 1, to − 1))

A_4 sublist(l, from, to) ≡ sublist(rest(l), from − 1, to − 1)

l (A B C D)

sublist(l, ?, 3) ≡ (B C)

A_4 sublist((B C D), 1, 2)

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

A_3 cons(B, cons(C, sublist((D), 0, 0)))

A_4 cons(B, cons(C, sublist(new, −1, −1)))

A_1 cons(B, cons(C, new))

≡ (B C)

N sticks

Q: what is max score when N sticks?

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

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

6

8 prop(1) = prop(2) = ... prop(6) = T

9 → prove prop(n) using induction:

10 $\forall j \le n,\ S(j) = \frac{j(j-1)}{2}$

$$S(N) = i(N-i) + S(N-i) + S(i)$$

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

$$= iN - i^2 + \frac{N^2}{2} - \frac{Ni}{2} - \frac{N}{2} - iN + \frac{i^2}{2} + \frac{i}{2} + \frac{i^2}{2} - \frac{i}{2}$$

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