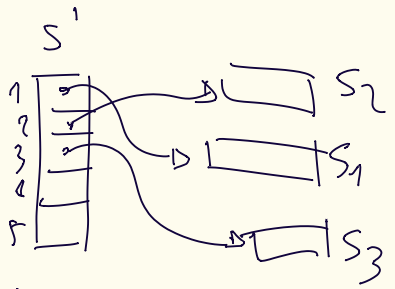
SORTING STRINGS

- strings from an alphabet Σ
- $|\Sigma| = \sigma$

- $S[1, n]$ array of pointers to strings
  - ↳ variable length
- N: total length of strings
- n: # strings
- AVG length $L = \frac{N}{n}$



pointers are sequential

strings are spread

obj

⇓

⇒ QSort (array S, n, string objects, comparison_function)

this takes ⇒ comparisons · AVG length of strings

$\Rightarrow O(L \cdot n \log n) =$

$= O\left(\frac{N}{n} \cdot n \log n\right) =$

$= O(N \log n)$

⇓

NOT OPTIMAL

LOWER BOUND

① to sort n strings ⇒ SORT 1ST CHARACTER ⇒ $\Omega(n \log n)$

② $d = \sum_S d_s \Rightarrow \Omega(d)$

⇓

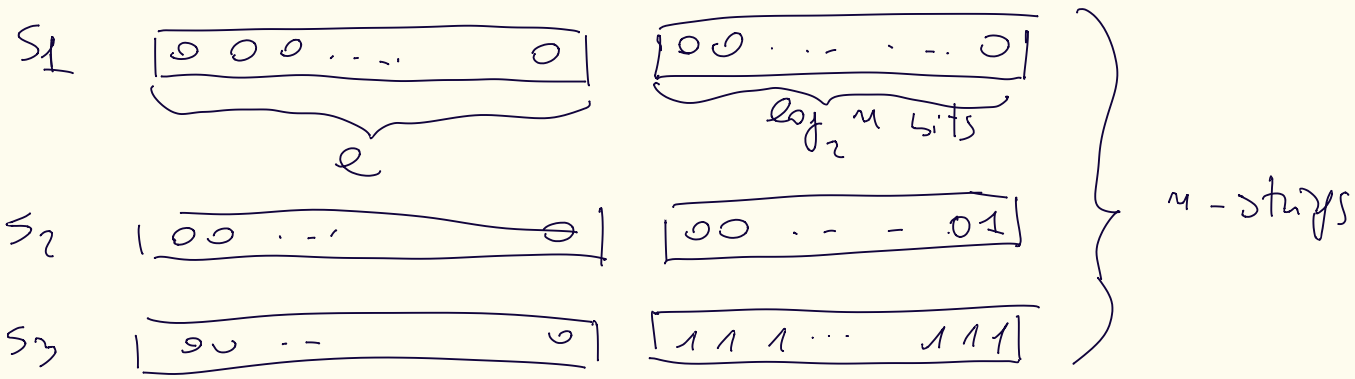
distinguish prefix of a string bytes

$\Rightarrow d = \sum_S d_s$

⇒ lower bound $\Omega(n \log n + d)$ time

- IMPACT OF lower bound of strings to QS (comparison-based algorithm)

- given set of strings:



$d_i = \ell + \log_2 n$

↑ distinguish with prefix

$\Rightarrow d = \sum_{i=1}^{n} (\ell + \log n) = \ell \cdot n \cdot \log n$

↑ overall string

lower bound

$\Omega(n \log n + d)$

qSort = $\left(\frac{L}{d} \cdot n \log n\right)$ $\frac{N}{n}$ in this case $L = \ell + \log n$

$\Rightarrow qS = \ell \cdot n \log n + n \log^2 n$

this case

qSORT $(\ell \cdot n \log n) =$

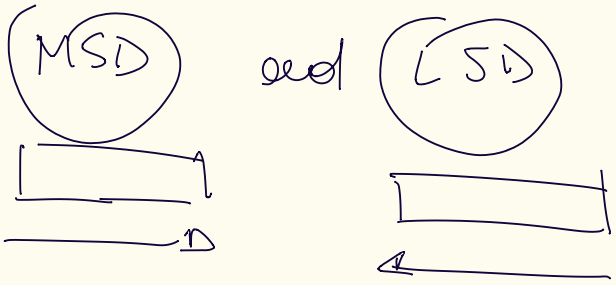
$= (\ell + \log n)(n \log n)$ ←

$O(\ell n \log n + n \log^2 n)$ UPPER BOUND

qSORT IS FAR FROM LOWER BOUND

- qSORT IS AT LEAST a $\lceil \log n \rceil$ factor from lower bound

RADIX SORT ALGOS: MSD and LSD



• strings of digits

MSD

$S = \{007, 017, 042, 911, 999\}$

$M = 5$

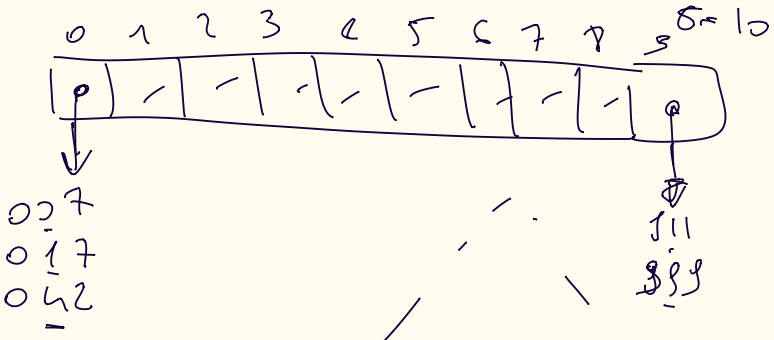
$\Sigma = |\Sigma| = 10$

total length of strings:

$\Rightarrow N = 3 \cdot 5 = 15$

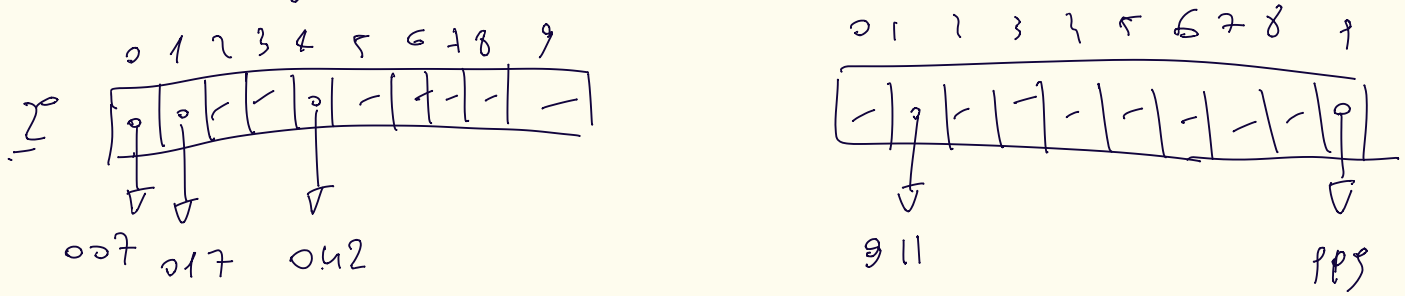
→ scan according to 1st symbol

1°



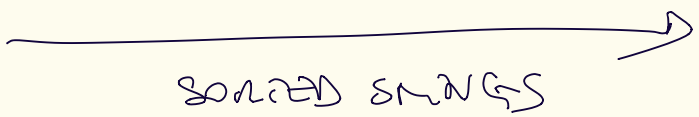
- scan according next digit

2°



- stop: every bucket 1 strings

visit



SORTED STRINGS

SIZE

" $\forall$ string s we create d_s nodes

$\#\text{arrays} = \#\text{nodes} \leq d = \sum_s d_s$

SPACE = $O(d \cdot \sigma)$ (not efficient) $\Rightarrow \forall$ node I'm allocating σ position (array of size σ)

TIME $O(d \cdot \sigma)$ (inefficient)

but usually σ is constant $\Rightarrow O(d)$ time

$\Omega(n \log n + d)$ => why this is comparison-based lower bound

=> here MSD is $O(d)$ => this is radix-based

(not a comparison based)

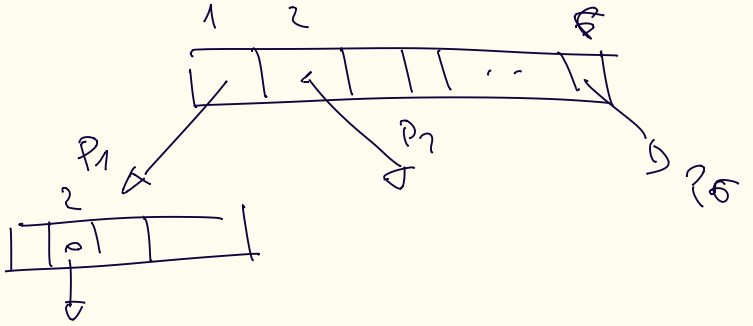
- cool for sorting $O(p)$ time => look the pattern

CAN IMPROVE MSD WITH HASH WITH CHAINING

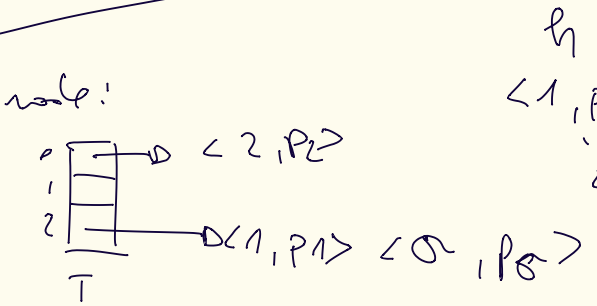
- pair: <digit, pointer> ——▷ inserted in hash table

key (digit) — satellite data (pointer)

- create an hash table of every node whose size is proportional to # of distinct pairs (# edges)



=>



h(<1,P1>) = 2 h(<2,P2>) = 0 h(<6,P6>) = 2 (collision)

$|T| = \Theta(\#keys)$

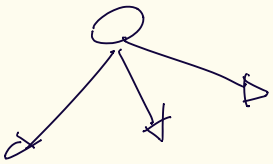
=> cost of search in T is $O(1)$ per op

$\frac{M}{m} < \frac{3}{3}$

=> space = $\Theta(\#edges) = \Theta(keys)$

could have an implementation with hash => but doesn't get sorted

MSD with hash table



- a hash table $\forall$ node

m = # edges

1) Build trie $O(1)$ insertion of every string (hit)

and I've to do $\forall$ string

$\Rightarrow \sum_s O(1) \cdot d_s \Rightarrow O(d)$ AVG

$\uparrow$ for each all strings

2) Sorting hash tables (edges) => qSORT on the pairs

$O(u \log u)$

$\downarrow$

~~# keys~~ = # edges

$O(\underbrace{\#\text{edges}}_{e_u} \cdot \log \#\text{edges})$

$\Rightarrow O(e_u \log_2 e_u)$

$\sum_u e_u \log_2 e_u \leq \sum_u e_u \log \sigma =$

$= \log_2 \sigma \left(\sum_u e_u\right) = O(\log \sigma \cdot d) =$

$\Rightarrow O(d \log \sigma)$ TIME

SPACE $= O(d)$

LOWER BOUND FOR SORTING STRINGS

=> Sorting according to 1st digit

$$\Omega(n \log_2 n)$$

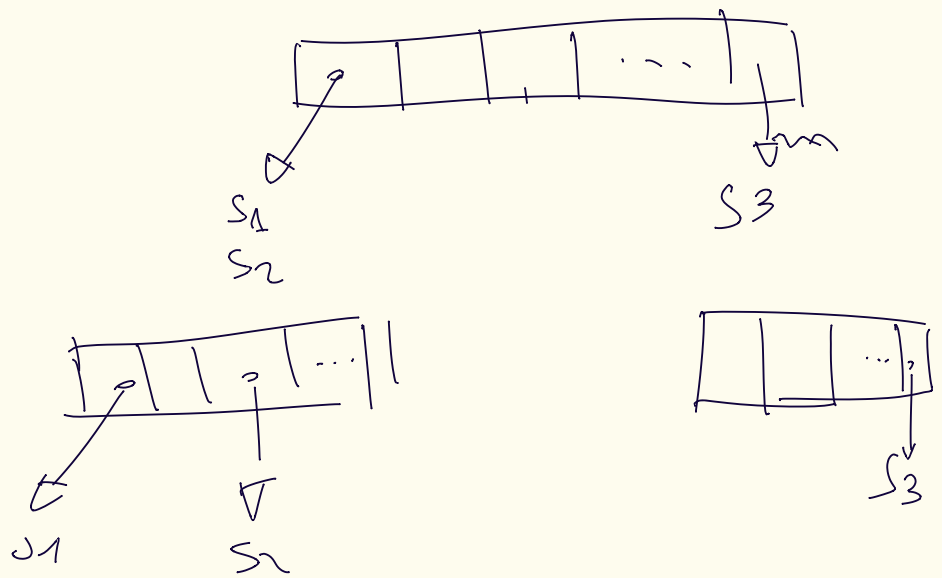
=> Consider distinguish prefixes

$$\sum_s \text{ds} : d \quad \Omega(d)$$

=> lower bound!

$$\boxed{\Omega(n \log_\sigma n + d)}$$

-> Consider MSD radix sort



$\sigma = |\Sigma|$

- given the strings
=> ~~keep~~ create and array of σ positions
=> distribute the strings according to the 1st char, the 2nd and so on UNTIL you have 1 string per bucket of the array

=> requires
$= \mathcal{O}(d \cdot \sigma)$ time and complexity

=> beats lower bound cuz the lower bound is referred to a comparison-based ALGO!

=> Improve MSD with hash table

=> $\forall$ node => store as hash table

=> edges => # keys

Create the tree $O(1) \cdot \sum_s l_s$

=> $O(d)$ on AVG TIME

=> Have to sort hash tables

$O(n \log n) =$

$= O(\#\text{edges} \log \#\text{edges}) =$

$= O(l_u \log l_u)$

let $l_u = \#\text{edges} = e_u$

$$\Rightarrow \sum_u l_u \log_\sigma l_u \leq \underbrace{\sum_u l_u}_{O(d)} \log \sigma =$$

$\Rightarrow O(d \cdot \log \sigma)$

LSD Radix sort

- STRINGS HAVE SAME length L
- SORT according to least significant bit 1st, 2nd, and so on (counting sort)

A B C
A B D
A C C

A B C
A C C
A B D

...

Counting sort costs $O(n + \text{max-integer size})$

⇒ here strings have same length L

⇒ $O(L(n + R))$ time SPACE $O(N)$

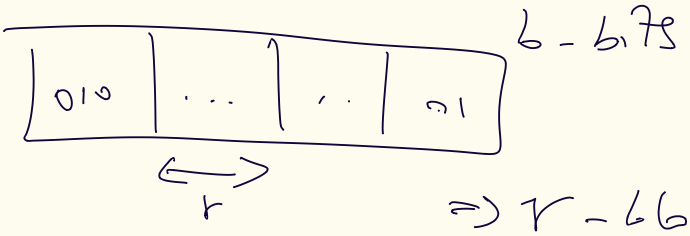
⇒ can compose more digits per time

(2) ⇒ $O(\frac{L}{2} \cdot (n + R^2))$

(3) ⇒ $O(\frac{L}{3} \cdot (n + R^3))$

digits ↓ this ↗

⇒ work on BITS instead of digits



⇒ r-blocks b-bits

$s=2$

$\Rightarrow \Theta\left(\frac{b}{r} \cdot (n + 2^r)\right)$ want find r the minimum $r \in \mathbb{R}$ the function

$2^r \geq n \Rightarrow \frac{b \cdot 2^r}{r}$

$r \geq \lg_2 n$

$\Rightarrow$ is minimized when $r = \Theta(\lg_2 n)$

$\Theta\left(\frac{b\, 2^{\lg n}}{\lg n}\right) = \frac{b\, n}{\lg_2 n} = \frac{N}{\lg_2 n}$ ← length

MULTI-KEY QUICKSORT

- Sorting strings of: VARIABLE LENGTH
- R: set of strings
- matches $\Omega(d + n \log n)$

Algorithm 7.1 MULTIKEYQS(R, i)

```
1:  if |R| ≤ 1 then
2:      return R;
3:  else
4:      choose a pivot-string p ∈ R;
5:      R_< = {s ∈ R | s[i] < p[i]};
6:      R_= = {s ∈ R | s[i] = p[i]};
7:      R_> = {s ∈ R | s[i] > p[i]};
8:      A = MULTIKEYQS(R_<, i);
9:      B = MULTIKEYQS(R_=, i + 1);
10:     C = MULTIKEYQS(R_>, i);
11:     return the concatenated sequence A, B, C;
12: end if
```

Proof: Consid invariant: all strys in A ($R_<$) (For MS)

shares $i-1$ characters prefix

string $S \in R$:

- $S \in R_< \ \vee \ S \in R_> \Rightarrow$ #strys halves $\Rightarrow$ occurs $\log_2 n$ times
- $S \in R_= \Rightarrow i{+}{+}$ (no other strings share prefix)

$i \le L$ ⇓ this invariant occurs wherever $i = d_S$

if pivot is good

$$\Rightarrow \le \sum_S d_S + \log_2 n = \sum_S d_S + \sum \log n = d + n \log n$$

(we have n objects)

SORTING STRINGS lower-bound

TAKE ACCOUNT

comparisons → $\Omega(n \log n)$ and distinguishing prefix $\sum_s d_s = d$ → $\Omega(d)$

$\Rightarrow \Omega(d + n \log n)$

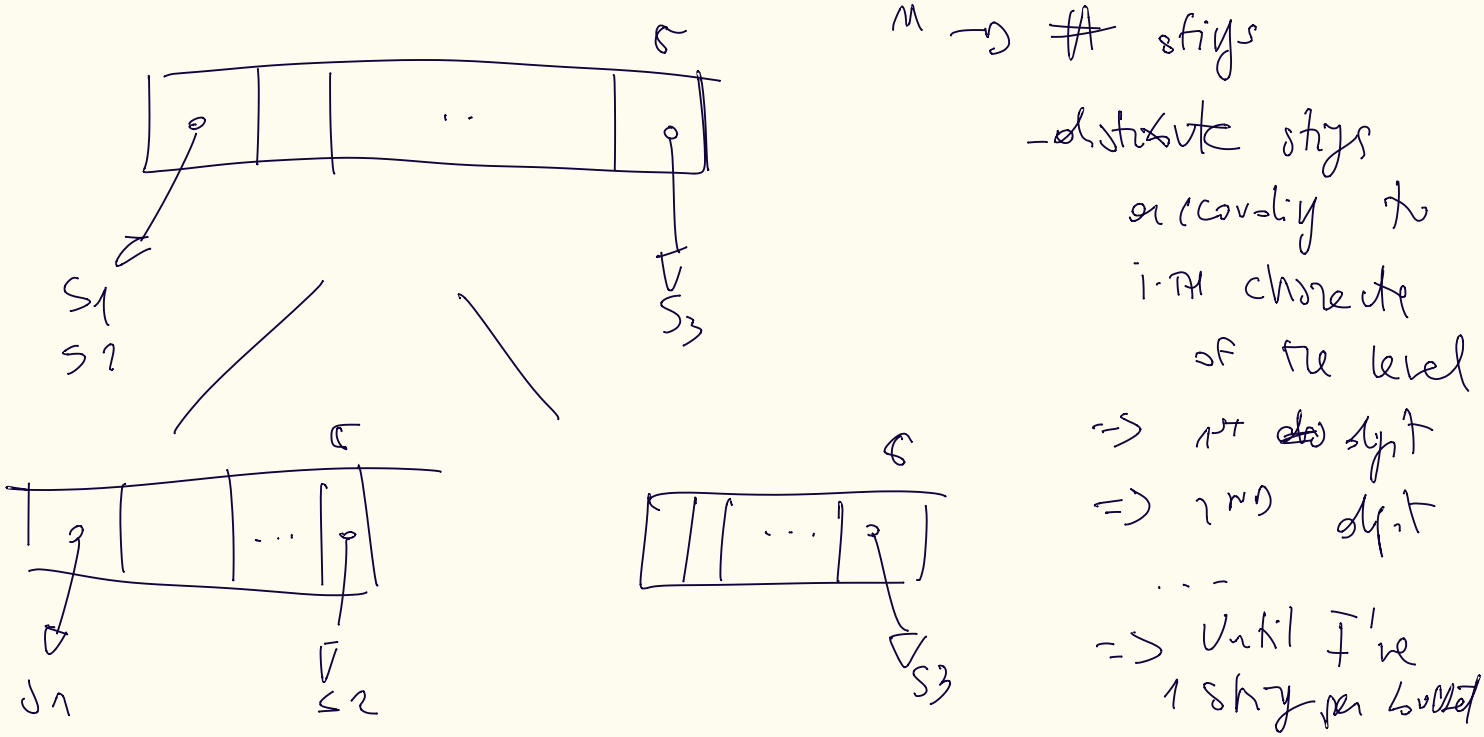
↳ lower bound for comparisons-sorting algorithms

MSD $\quad \sigma = |\Sigma|$ $\quad L$ = avg length of the strings

$L = \frac{N}{n}$ → length of strings

n → # strings

- distribute strings according to i-th character of the level



$\Rightarrow$ Consider nodes & levels (for comparisons)

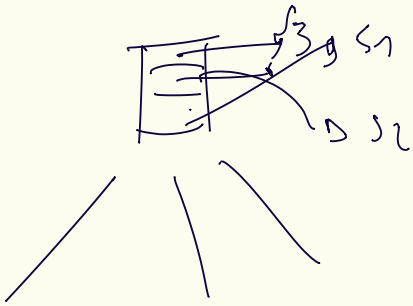
$\Rightarrow$ $|\text{array}| = \sigma$ $\quad$ $\Rightarrow$ I go to many levels as # d

$\Rightarrow O(d \cdot \sigma)$ $\quad$ $\Rightarrow \sum_s d_s = d$

Improve with hash table => the

- every node has an hash table size of $|T| = s$

=> h (< string, id string >)



#edges - keys

$$O(\underbrace{\#edges}_{e_v} \log \underbrace{\#edges}_{e_v}) = O(e_v \log e_v) =$$

$$\Rightarrow O\left(\sum_s e_v \log e_v\right) \leq O\left(\sum_s e_v \log s\right) =$$

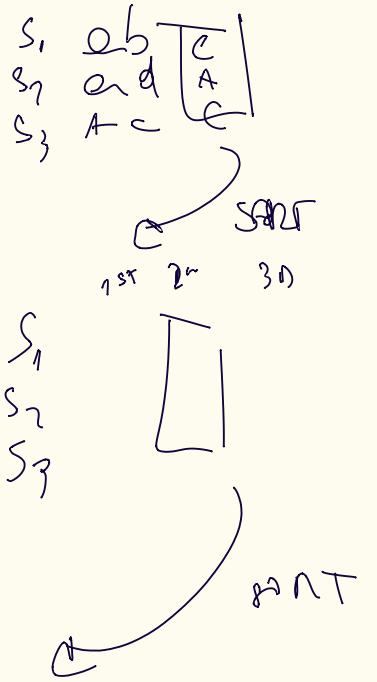
$$\underbrace{\sum_s}_{d}$$

$$\Rightarrow O(d \log s) \quad \checkmark$$

- LSD

- Counting sort → $(n + MAX_{integer\ size})$

according to least significant digit

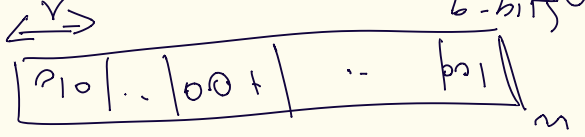


$$O(L(n + \delta))$$

comparing 2 digits at time

$$O\left(\frac{L}{2}(n + \delta^2)\right) \quad \text{3 digits} \quad O\left(\frac{L}{3}(n + \delta^3)\right)$$

Consider BITS instead of digits



Group r-BITS at time:

$$O\left(\frac{b}{r}(n + \delta^r)\right) \qquad \delta = 2$$

$$O\left(\frac{b}{r}(n \cdot 2^r)\right)$$

WANT r that minimize function

$$2^r > n \Rightarrow r > \log_2 n$$

$$\frac{b \cdot 2^r}{r}$$

$2^r \leq n$ not in code

$\Rightarrow$ Set $r = \Theta(\log_2 n)$

$\Rightarrow \quad \frac{b \cdot 2^r}{r} = \frac{b \cdot 2^{\log_2 n}}{\log_2 n} =$

$= \frac{b\,n}{\log n} = \frac{L}{\log n}$

$MS\rangle = \left(\frac{L}{\log n}\right)$

$MS\rangle \ (d \cdot \rho)$

$MS\rangle$ with hash (trie) $= (d \text{ by } \rho)$

$\Omega =$ MK quicksort $= \Omega(d + n \log n)$

Multi-key QS

- Matches lower bound $\Omega(d + n \log n)$

- Strings of w length

- R : set of strings

if $|R| < 1$ return R

. else choose a good pivot p in R (p is a string)

$R_< = \{ s \in R : s[i] < p[i] \}$

$R_= = \{ s \in R : s[i] == p[i] \}$

$R_> = \{ s \in R : s[i] > p[i] \}$

$A = MKQ(R_<, i)$

$B = MKQ(R_=, i+1)$

$C = MKQ(R_>, i)$

proof Invariant : strings in A shares i-1 character - prefix

$s \in R$ — $s \in R_< \vee R_> \Rightarrow$ occurs $\log_2 n$ times

— $s \in R_= \Rightarrow i$ increases until

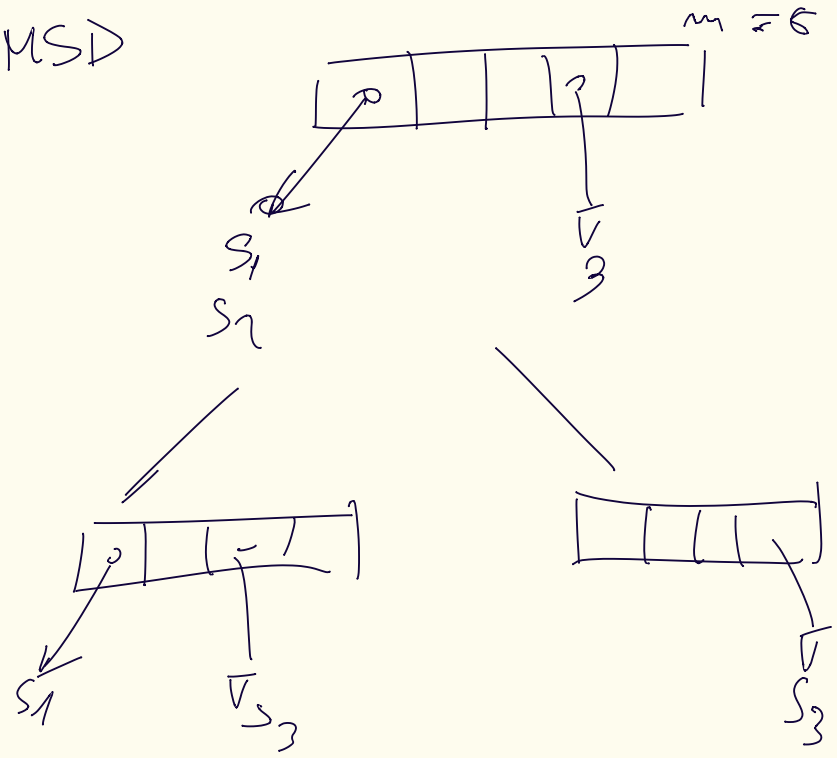
occurs $\Leftrightarrow$ $i \leq L$

when $i = ds$

$$\Rightarrow \sum_s d_s + \sum_s \log_2 M = \Omega(d \cdot n \log n)$$

$\uparrow$
strings

MSD



$\sigma = |\Sigma|$

- distribute strings according to 1st, second, ...

stop until you get $\leq$ string per bucket

read ⟶

=> sorted order

cost: I've array of σ positions at every level and I'm to account for each the number of distinguish prefixes

$$\Rightarrow O\left(\left(\sum_S d_S\right) \cdot \sigma\right) = O(d \cdot \sigma)$$

=> problem some spaces of array are null

=> hash => every node has an hash table

(hash a pair $\langle str_i, s_1 \rangle$ $\langle str_i, s_2 \rangle$ ... $\langle str_i, s_m \rangle$

ed edges:

(for where I make a comparison)

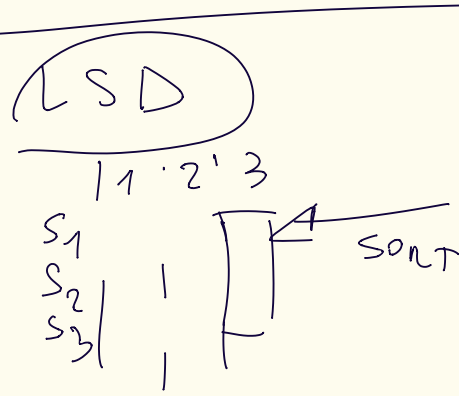
$$O\left(\#\text{edges} \log_2 \#\text{edges}\right)$$

$\#\text{edges} = \sum_{v} e_v$

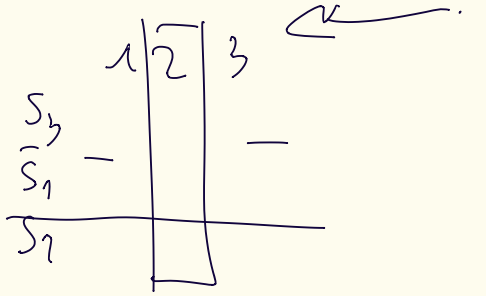
$$O(n \lg n) = O\left(\sum_s d_s \cdot \lg_2 n\right) =$$

$$= O(d \cdot \lg_2 n) : \quad d \to \sigma$$

$$= O(d \lg_2 \sigma)$$



counting sort
$(n + \max \text{representation})$



stays have fixed length L

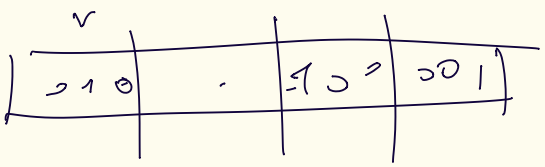
end of counting sort

$$O(L \cdot (n+\sigma))$$

can combine 2 digits per time

$$O\left(\frac{L}{2}(n+\sigma^2)\right)$$

idea: consider bits instead of digits



I've ~~k~~ digits to compare and I've b-bits

but Θ with BITS $= 2$

$$O\left(\frac{b}{r} \cdot \left(n + \Theta^r\right)\right) =$$

$$= O\left(\frac{b}{r} \cdot \left(n + 2^r\right)\right)$$

WANT r that minimize the term

$\Rightarrow$ 2) cases — $2^r > n$

$2^r \leq n$ NOT INTERESTING

$\Rightarrow$ $2^r > n$ $\Rightarrow$ $r > \log_2 n$

$$\Rightarrow \frac{b \cdot 2^r}{r} =$$

$$= \frac{b \cdot 2^{\log_2 n}}{\log_2 n} =$$

$$= \frac{b \cdot n^{=1}}{\log_2 n} = O\left(\frac{1}{\log_2 n}\right)$$

MultikeyQS(R, i) R = set of strings

if |R| ≤ 1 RETURN R

else

p = choose a "good" pivot from S, p is a string

$R_< = \{ s \in R : S[i] < p[i] \}$

$R_= = \{ s \in R : S[i] = p[i] \}$

$R_> = \{ s \in R : S[i] > p[i] \}$

A = MKQ($R_<$, i)

B = MKQ($R_=$, i+1)

C = MKQ($R_>$, i)

return A . B . C where . is concatenation

Invariant: strings in A share i-1 chars ~~of~~ suffix

=> case: strings p in $R_<$ or $R_>$

=> occurs $\lg_2 n$ time

case string in $R_=$:

=> I've i+1

=> $i \leq L$

=> occurs $\sum_s ds$ time

complexity $\leq \sum_s (ds + \lg_2 n) = \sum_s ds + \sum_s \lg_2 n =$

$= O(d + n \lg n)$

Multi-key qs

R = set of strs

MKQ(R, i)

if |R| ≤ 1

return R;

else

p = choose a good pivot (p ∈ R)

$R_< : \{ s \in R : s[i] < p[i] \}$

$R_= : \{ s \in R : s[i] == p[i] \}$

$R_> : \{ s \in R : s[i] > p[i] \}$

A = MKQ($R_<$, i)

B = MKQ(R=, i+1)

C = MKQ($R_>$, i)

L: length of str

Complexity

Invariant: in A strs sharing i-1 chars

CASES: 1) strs in $R_<$ ∨ $R_>$

=> occurs $\log_2 n$ times

(i doesn't increase)

2) str in R=

=> i++

=> i ≤ L

=> occurs $\sum_s ls$ times

=> $\sum_s (ls + \log n)$ =

$$= \sum_s d_s + \sum_s \log n \Big) = O\left(d + n \log n\right)$$

- $L = \frac{N}{n}$ → tot. length of strings
  → # of strings

$\sigma = |\Sigma|$

Lower bound

=> sorting strings according to 1st character
$\Omega(n \log_2 n)$ (is sorting)

=> Note + consider all the distinguish prefix that are in of all the strings

=> $\Omega\left(\sum_s d_s\right) = \Omega(d)$

=> $\Omega(d + n \log n)$

MKQ MATCHES THIS BOUND

Let be • R: set of strings of variable length

```
MKQ(R, i)
    if |R| ≤ 1
        Return R;
    else
        p = extract a string to be a "good" PIVOT
        R< = {s ∈ S : s[i] < p[i]}
        R= = {s ∈ S : s[i] == p[i]}
        R> = {s ∈ S : s[i] > p[i]}
    A = MKQ(R<, i); B = MKQ(R=, i++); C = MKQ(R>, i)
```

Return A . B . C // (where . is the concatenation)

Cost : Invariant: ~~strings~~ strings in A stores $i-1$ chars

(2) CASES

→ 1) ℓ strings pass $R_<$ ∨ $R_>$ ⇒ occurs $\log_2 n$ times

(i) bound increased

→ ℓ strings go in $R_=$ ⇒ i++

⇒ good pivot

mean balanced partitions

$i \le L$

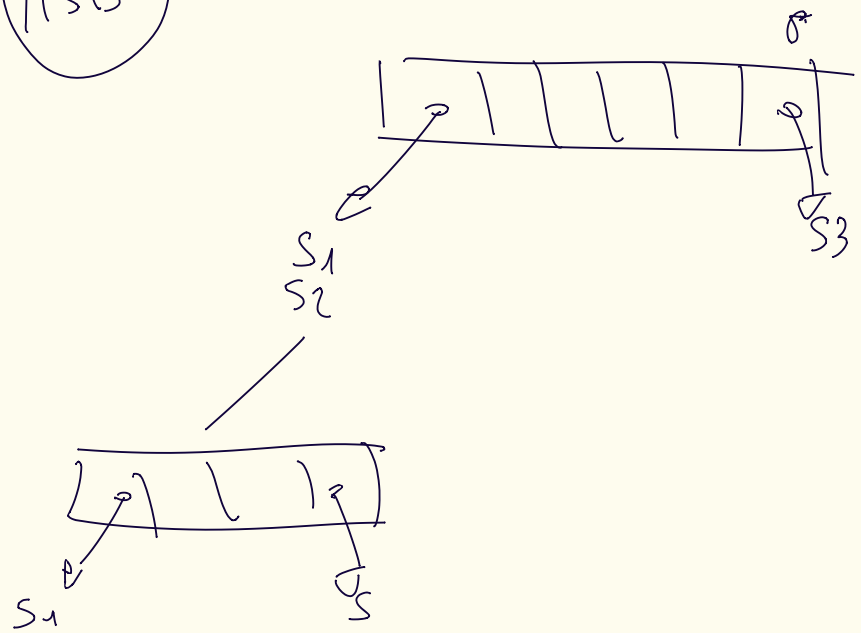
occurs $i \le \sum_s d_s$

$$\Rightarrow O\left(\sum_s d_s + \log n\right) = O(d + n \log_2 n)$$ ✓

MSD : $O(d \cdot \sigma)$ with hash $O(d + \log_2 \sigma)$

$< SS$ $O\left(\frac{L}{\log_2 \sigma}\right)$

MSD



- sort acc. to 1st digit
run until each string is in just 1 bucket

Arrays have size σ (at every level) and I'm "bucketing" the strings (~~acc~~ according to their distinguish prefixes

$$\Rightarrow O\left(\sigma \cdot \sum_s d_s\right) = O(\sigma \cdot d) = O(d \cdot \sigma)$$

---

with hash: every node has a hit $\Rightarrow$ size σ and I've ~~to~~ to take into account all the edges (they share the algorithm $\Rightarrow$ comparisons)

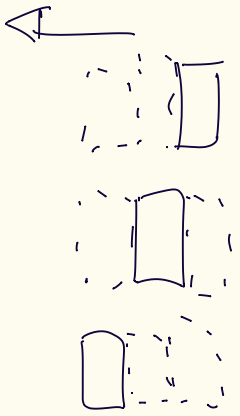
$$O\left(\underbrace{\#\text{edges}}_{e_u} \log \underbrace{\#\text{edges}}_{e_u}\right) = O\left(\sum_s e_u \log e_u\right) \leq$$

$$\leq O(d \cdot \log \sigma)$$

L8)

bucket sort $O(n + \text{max rep.})$

es L=3



with strip, complexity L digit per time

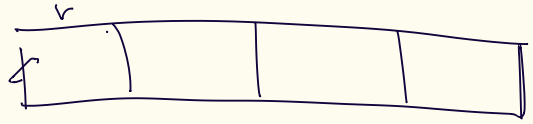
$\Rightarrow O(L \cdot (n + \sigma))$

2 digits

$O\left(\frac{L}{2} \cdot (n + \sigma^2)\right)$

$\Rightarrow$ don't use digits but bits

read compact r-bits at time



$\sigma = 2$

b = bits

$\Rightarrow O\left(\frac{b}{r} \cdot (n + \sigma^r)\right) =$ ($\sigma^r = 2^r$)

$= O\left(\frac{b}{r}(n + 2^r)\right)$

WANT R that minimize the fuction

$2^R \leq n \Rightarrow$ not the case

$2^r > n \Rightarrow r > \log_2 n$

$\frac{b + 2^r}{r} \Rightarrow \frac{b + 2^{\log_2 n}}{\log_2 n} =$

L

$= \frac{b + n}{\log_2 n} = \frac{L}{\log_2 n}$ ✓