0. Recap: data structure.

a. Trees:

- Binary trees: { linked list / array
- Binary Search Tree (BST)

= sorted array + fast Ins/DEL
O(log n)

BST property: $\forall x$, key(x) l.s.t $\leq$ $\leq$ r.s.t.

Supp' ops: search, min/max, rank, ...

→ use rotation to keep it balanced

→ use case: an evolving set of obj's + totally ordered rep.

- Heap (堆): a special B.T.

Heap property (MIN): $\forall x$, key(x) $\leq$ key(x's children)

→ use case: fast min/max on evolving data set.

b. Hash table.

hash function + n buckets (Array)

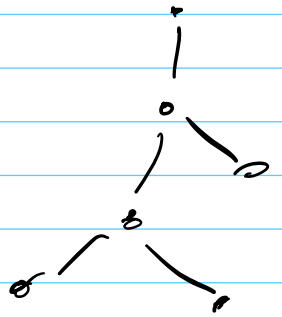
→ Collision: { chaining.
{ open addressing: linear probing

↓

choose "good" hash function: random hash function

→ use case: constant time Ins/Del/Lookup

1. Call back: succinct pfs.



? A rail trip to visit every city exactly once.

a. graphs

- DEF: An (undirected) graph. $G = (V, E)$.
  - V: set of nodes
  - $E \subseteq V \times V$: $\{\{u, v\} : u, v \in V\}$.
- PATH: $u \sim v$, $u - w_1 - w_2 - \cdots - w_k - v$
- connected graph: $\forall u, v \in V$, $\exists$ PATH $u \sim v$.

b.

H.P. (Hamiltonian PATH)

Given : $G = (V, E)$ $|V| = n$

Goal : decide if $\exists$ PATH $v_{i_1} - v_{i_2} \cdots - v_{i_n}$?
visiting every node exactly once

$(v_{i_j} \neq v_{i_k})$

- Such problems are called Decision problems.
  → Answer : YES / NO

» Direct computation : exptime by exhaustive search.

→ Deciding by (interactive proofs.

Prover (P) — π (pf) → Verifier (V) — Y/N

In HP : $\pi = v_{i_1} - \ldots - v_{i_n}$

① $\{v_{i_j}, v_{i_{j+1}}\} \in E$.

② v_{i_j} distinct.

Accept, output YES.

* Completeness : if G is YES instance,
$\exists \pi$, V(π) output YES.

* Soundness : if G is NO instance,
$\forall \pi$, V(π) output NO!

Verifier time in above : $\Theta(n)$

⇓

A breakthrough in TCS : PCP Theorem

(Probabilistic Checkable Proofs)

$P \xrightarrow{\pi = v_1 - \cdots - v_n} V$

⇓ PCP.

$PCP(P) \xrightarrow{\tilde{\pi} = \text{poly}(n)\ (n^2, n^3)} PCP(V)$

"super efficient"

PCP(V) : checks 3 random bits in $\tilde{\pi}$

c. succinct pfs (CS pfs : Computationally Sound)

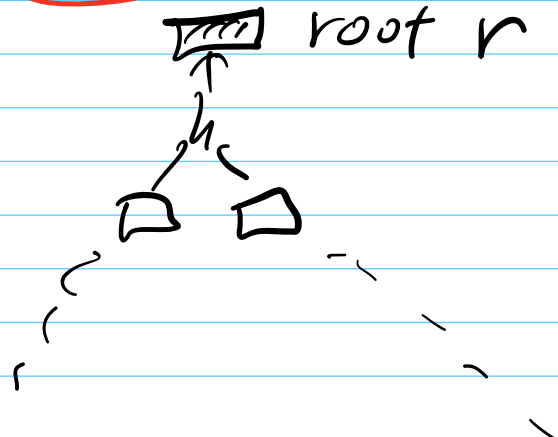
Goal : compress $\tilde{\pi}$

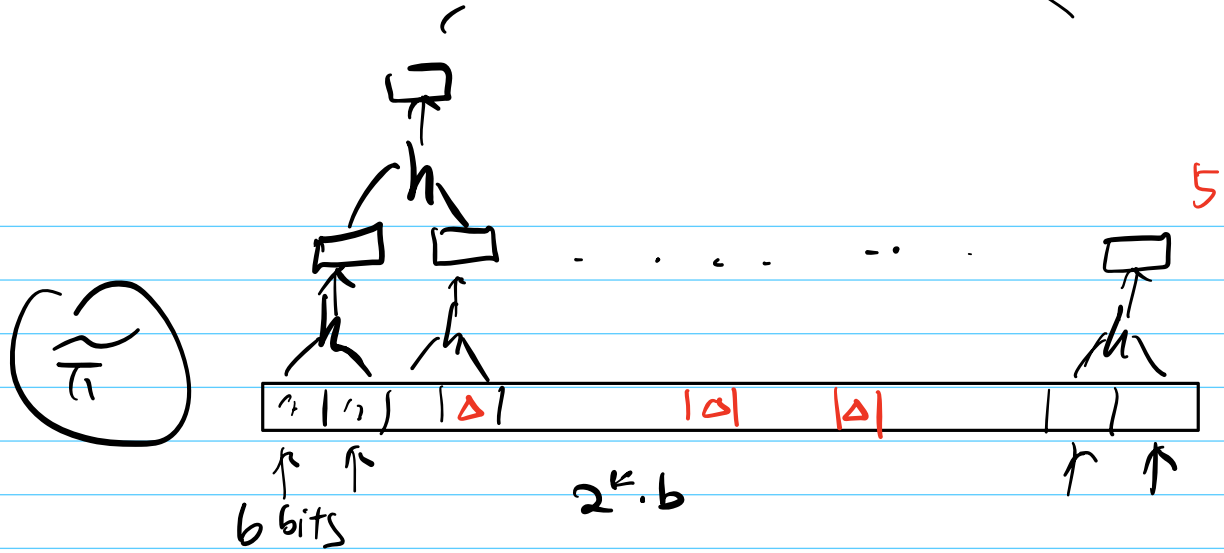
Assuming : $|\tilde{\pi}| = b \cdot 2^k$, 2^k blocks.

b bits each b bits

$h: \{0,1\}^{2b} \to \{0,1\}^b$

(random)





- Merkle tree : $MT(\tilde{\pi})$ from leaves apply h iteratively until root r

  height : k

- cs pf.

P —— b bits, r : root of $MT(\tilde{\pi})$ ——→ V

V ←—— random blocks V wants to check ——

P ——→ block values + ☆ ——→ ✓

☆ completeness ✓

↑ Soundness.

→ making it non-interactive.

2. Zaritation to ToC

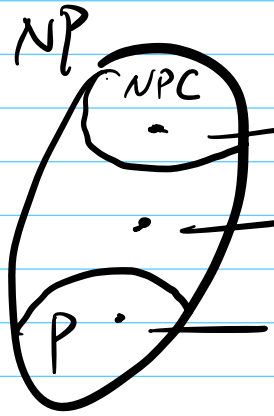
a.

Q: H.P. can be solved by one-msg. pf system.

$P \xrightarrow{\pi} V$ → Y / N

what else can be solved this way?

NP : set of problems where solns. can be verified in poly-time.



- H.P.
- factoring: Given N., is there a factor $< k$?
- sorted: Given A[n], is it sorted?

→ P: set of problems solvable in poly-time

$P \subseteq NP$.

→ H.P: seems most hard.
if H.P. $\in P$, $NP = P$
($\forall A \in NP, A \leq$ H.P.)
NP complete.

★ $P \stackrel{?}{=} NP$.
(Million dollar problem)

b. Complexity theory.

complexity classes:

- Space: PSPACE: set of problems solvable in poly space.
  (poly)

EXP — EXP: problems solvable in exp. time.
(time)

PSPACE

NP

P

IV