PAL

- STANDARTNÍ GRAFOVÉ ÚLOHY S POLYNOMIÁLNÍ SLOŽITOSTÍ ŘEŠENÍ
- KOMBINATORICKÉ A ČÍSELNÉ TEORETICKÉ ALGORITMY
- IZOMORFISMUS
- PRVOČÍSELNOST
- VYHLEDÁVACÍ STROMY A JEJICH VYUŽITÍ
- VYHLEDÁVÁNÍ V TEXTU ZALOŽENÉ NA KONEČNÝCH AUTOMATECH

- ASYMPTOTICKÝ RŮST FUNKCE

- $f(n) \in O(g(n))$

  $\exists c > 0 \ \exists n_0 \ \forall n > n_0 : |f(n)| \leq |c \cdot g(n)|$

- $f(n) \in \Omega(g(n))$

  $\exists c > 0 \ \exists n_0 \ \forall n > n_0 : |f(n)| \geq |c \cdot g(n)|$

- $f(n) \in \Theta(g(n))$

  $\exists c_1 > 0 \ \exists c_2 > 0 \ \exists n_0 \ \forall n > n_0 : |c_1 \cdot g(n)| \leq |f(n)| \leq |c_2 \cdot g(n)|$

- GRAPH
  - $G = (V, E)$
    - V = VERTICES
    - E = EDGES
  - DIRECTED, UNDIRECTED
  - WEIGHTED $W : E \to \mathbb{R}$
  - NODE DEGREE
    - NODES X AND Y LINKED BY EDGE E
      - NODES X AND Y ARE INCIDENT TO E
    - FUNCTION RETURNING NUMBER OF EDGES INCIDENT TO GIVEN NODE

- FOR DIRECTED GRAPH THERE IS IN AND OUT DEGREE

- HANDSHAKING LEMA

$$\sum_{n \in V} DEG(n) = 2|E|$$

- SUM OF DEGREES OF NODES IS EQUAL TO TWO TIMES THE NUMBER OF EDGES

- EACH EDGE IS CONNECTED TO TWO VERTICES

- COMPLETE GRAPH

$$E = \binom{V}{2}$$

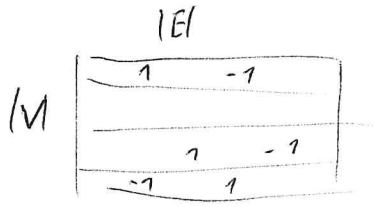
$$- (\forall n \in V): DEG(n) = |V| - 1$$

- PATH
  - SEQUENCE OF VERTICES AND EDGES
  - EACH VERTICES HAVE TO DIFFER FROM EACH OTHER
- CIRCUIT
  - ORIENTED
  - CLOSED PATH
- CYCLE
  - NON-ORIENTED CIRCUIT

- CONNECTED GRAPH
  - FOR EACH VERTICES X AND Y IN G
    - THERE IS PATH FROM X AND Y

- TREE
  - CONNECTED GRAPH WITHOUT CYCLES
  - IF ARBITRARY NEW EDGE IS ADDED, THAN CYCLE OCCURS
  - IF WE REMOVE EDGE, IT BECOMES DISCONNECTED
  - THERE ARE $|V|-1$ EDGES
  - EVERY TWO VERTICES ARE CONNECTED BY SINGLE PATH ONLY
  - LEAF
    - DEGREE 1
  - ROOT
    - NO INCOMING EDGES

- ADJACENCY MATRIX
- LAPLACIAN MATRIX
  - ADD VERTEX DEGREE ON DIAGONAL
- DISTANCE MATRIX
  - ADD DISTANCES (WEIGHTS)
- INCIDENCE MATRIX



- ADJACENCY LIST

- DAG
  - DIRECTED ACYCLIC GRAPH
  - NO CYCLES
- MULTIGRAPH
  - MULTIPLE EDGES BETWEEN TWO VERTICES
  - EDGE FROM NODE TO ITSELF

- DFS
  - STACK

- BFS
  - QUEUE

- PRIORITY QUEUE
  - INSERT WITH PRIORITY

- SUBGRAPH
  - TWO INCLUSIONS MUST BE SATISFIED

$$V(H) \subseteq V(G)$$

$$E(H) \subseteq E(G) \cap \binom{V(H)}{2}$$

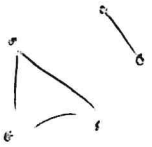
  - CREATED BY
    - REMOVE SOME VERTICES
    - REMOVE EDGES ADJACENT WITH REMOVED VERTICES

- TOPOLOGICAL ORDERING
  - DAG
  - WE ASSIGN NUMBER TO VERTICES
  - FOR EACH PAIR OF VERTICES HOLDS
    - IF THERE IS DIRECTED PATH FROM X TO Y
      THAN NUMBER $X \leq_{NF} Y$
  - CAN BE ASSIGNED BY DFS

- CONNECTED COMPONENT

  - SET OF VERTICES

    $C(v) = \{ w \in V \mid$ THERE EXISTS PATH IN G FROM w TO $v \}$



- SPANNING TREE

  - SUBGRAF H OF GRAPH G
  - $V(G) = V(H)$
    - THERE ARE ALL VERTICES OF G
  - AND H IS TREE

  - MINIMUM SPANNING TREE
    - SUM OF ITS WEIGHTS OF EDGES IS MINIMAL

  - CUT OF GRAPH
    - SUBSET OF EDGES WHICH DIVIDES GRAPH INTO TWO CONNECTED COMPONENTS
    - EVERY MINIMUM SPANING TREE CONTAINS EDGE OF CUT WITH THE SMALLEST WEIGHT

  - JARNIK - PRIMS ALGORITHM
    - 1. SELECT ARBITRARY VERTEX AND ADD IT TO S
      - 2. SELECT EDGE ADJACENT WITH ANY VERTEX IN S WITH SMALLEST WEIGH AND ADJACENT WITH V', $V' \notin S$
      - 3. ADD V' TO S
      - REPEAT FROM 2. UNTILL ALL VERTICES ADDED TO S

- IT TAKES $|V(G)|$ STEPS

- VERTICES IN S AND THE REST CREATES CUT

  - WE SELECT EDGE OF THIS CUT TO SPANNING TREE WITH SMALLEST WEIGHT

    - SUCH EDGE MUST BE IN MINIMAL SPANNING TREE

- IMPLEMENTATION

  - STRAIGHTFORWARD

    - MAINTAIN WHICH VERTICES ARE IN M.S.T. AND WHICH NOT

    - TIME COMPLEXITY $O(|V(G)| \cdot |E(G)|)$

  - IMPROVED

    - STORE ONLY ~~COSTLIEST~~ EDGES OF ACTUAL CUT WITH LOWEST WEIGHT FOR EACH NODE IN ACTUAL SPANNING TREE $O(n^2)$

    - USE HEAP $O(\log(n) \cdot m)$

- BORŮVKAS ALGORITHM

  - WHILE GRAPH HAS TWO OR MORE CONNECTED COMPONENTS

  - SELECT THE EDGE WITH LIGHTEST WEIGHT CONNECTING TWO CONNECTED COMPONENTS

  - STOPS AFTER MAX $\lceil \log_2 |V(G)| \rceil$ STEPS

  - IMPLEMENTATION

    - WE DECOMPOSE FOREST TO CONNECTED COMPONENTS BY DFS
      - EACH VERTEX IS ASSIGNED TO NUMBER OF ITS COMPONENT

    - FOR EACH EDGE WE FIND OUT TO WHICH COMPONENT IT BELONGS TO AND WE STORE THE LIGHTEST EDGE ONLY

    - EACH ITERATION TAKES $O(|E(G)|)$

    - ENTIRE ALGORYTHM RUNS IN $O(|E(G)| \cdot \log |V(G)|)$

- KRUSKAL'S GREEDY ALGORITHM
  - SORT EDGES BY WEIGHT
  - ADD LIGHTEST EDGE TO MINIMAL SPANNING TREE IF IT IS STILL ACYCLIC
  - IT STOPS AFTER $|E(G)|$ ITERATIONS
  - IMPLEMENTATION
    - SORTING TIME $O(|E(G)| \cdot \log|E(G)|) = O(|E(G)| \cdot \log|V(G)|)$
    - WE CAN STOP LOOP AFTER WE ADDED $|V(G)|-1$ EDGES
    - WE HAVE TO MAINTAIN CONNECTED COMPONENTS
      - UNION - FIND
        - FOR EACH ~~REPRESENTED~~ CONNECTED COMPONENT ONE REPRESENTATIVE IS SELECTED
          - REPRESENTATIVE OF COMPONENT $C(v)$ IS LABELED AS $r(v)$
          - IF w AND v ARE IN THE SAME COMPONENT, THAN $r(w) = r(v)$
        - FIND
          - RETURNS REPRESENTATIVE $r(v)$ OF CONNECTED COMPONENT $G(v)$
        - UNION
          - MERGES TWO CONNECTED COMPONENTS $C(w)$ AND $C(v)$
        - SIMPLE SOLUTION
          - FIND
            - JUST RETURN VALUE $r(v)$ OF NODE
            - $O(1)$
          - UNION
            - $r(w) = \text{FIND}(w)$, $r(v) = \text{FIND}(v)$
            - IF $r(w) \neq r(v)$
              - GO THROUGH V AND REWRITE $r(w)$ INTO $r(v)$ OR

- IMPROVED SOLUTION
  - EACH COMPONENT STORED AS TREE
  - ROOT ACTS AS REPRESENTATIVE
  - FIND
    - CLIMBS THE TREE
  - UNION
    - $r(u) = \text{FIND}(u)$, $r(v) = \text{FIND}(v)$
    - IF $r(u) \neq r(v)$
      - ROOT OF SMALLER COMPONENT IS MERGED INTO ROOT OF BIGGER COMPONENT
  - TIME COMPLEXITY OF BOTH UNION AND FIND IS $O(\log |V|)$
- KOMPLEXITY OF KRUSKAL'S ALGORITHM WITH UNION FIND
  - $O(|E(G)| \cdot \log |V(G)|)$

- STRONGLY CONNECTED COMPONENT
  - IN DIRECTED GRAPH
  - MAXIMAL SET OF STRONGLY CONNECTED VERTICES
    - THERE EXIST ORIENTED PATH BETWEEN EVERY COUPLE OF VERTICES

- KOSARAJU - SHARIR ALGORITHM
  - STACK
  - DFS OF GRAPH G.
  - IF WE CLOSE VERTEX N, ADD N TO STACK
  - REVERSE THE EDGES
  - WHILE SOMETHING OF STACK
    - POP STACK
    - DFS FROM POPED VERTEX (IF NOT CLOSED ALREADY)
    - ALL VERTICES VISITED BY DFS, ADD TO STRONGLY CONNECTED COMPONENT AND CLOSE THEM
  - PERFORMS TWO TRAVERSALS OF GRAPH
  - IF GRAPH REPRESENTED AS ADJACENCY LIST, THEN ALGORITHM RUNS IN $\Theta(|V|+|E|)$
  - IF ADJACENCY MATRIX, THAN $O(|V|^2)$

- TARJAN ALGORITHM
  - DFS
  - EACH NODE HAS TWO NUMBER
    - ~~EACH~~ ITS NUMBER
    - NUMBER OF LOWEST EDGE INTO WHICH WE CAN GET
  - IF WE BACKTRACK TO ~~VERO~~ NODE N WE CLOSE ALL NODES REFERENCING TO N AND ADD IT TO ITS STRONGLY CONNECTED COMPONENT
  - COMPLEXITY
    - ADJACENCY LIST $\Theta(|V|+|E|)$
    - ADJACENCY MATRIX $O(|V|^2)$
    - IT RUNS FASTER THAN KOSARAJU - SHARIR ALGORITHM

- EULER TRAIL PROBLEM
  - DOES GRAPH G CONTAINS TRAIL THAT CONNECTS EVERY EDGE EXACTLY ONCE
  - "DRAW IN SINGLE LINE"
  - CONTAINS IF
    - IT IS CONNECTED
    - AND HAS 0 OR 2 VERTICES OF ODD DEGREE
  - ALGORITHM
    - DFS
    - ADD EDGE TO TRAIL IF BACKTRACK
  - ONE COMPLETE TRAVELSAL OF GRAPH
  - ADJACENCY LIST $\Theta(|V|+|E|)$
  - ADJACENCY MATRIX $O(|V|^2)$

- HAMILTON PATH
  - DOES GRAPH CONTAINS PATH THAT VISITS EVERY NODE EXACTLY ONCE
  - NPC HARD

- HEAPS
  - TREE BASED DATA STRUCTURE
  - SATISFIES HEAP PROPERTY
    - IF B IS A CHILD NODE OF A, THEN KEY(B) $\geq$ KEY(A)
  - ONE OF THE MOST EFFICIENT IMPLEMENTATION OF PRIORITY QUEUE
  - OPERATIONS
    - INSERT (x)
    - ACCESS_MIN
    - DELETE_MIN
    - DECREASE_KEY (x, d)
    - MERGE (H_1, H_2)
    - DELETE (x)
- BINARY HEAP
  - BINARY TREE WITH TWO ADDITIONAL CONSTRAINTS
    - COMPLETE BINARY TREE, EXCEPT THE LAST LEVEL
    - IF LAST LEVEL IS NOT FILLED, THE NODES ARE FILLED FROM LEFT TO RIGHT
    - EACH NODE IS LESS THAN OR EQUAL TO TO EACH OF ITS CHILDREN

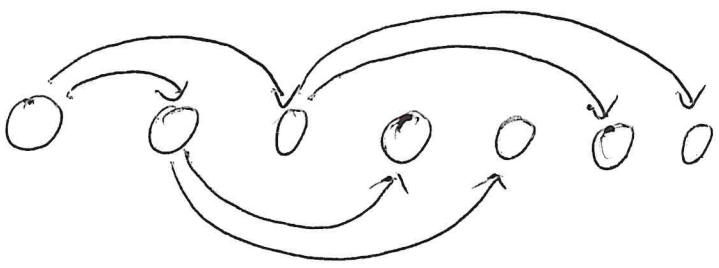
- INSERT (x)
  - ADD NODE TO THE END OF HEAP
  - PROPAGATE NODE UP IF IT IS SMALLER THAN PARENT
    - SWAP PLACES OF NODE AND PARENT

- ACCESS MIN
  - GET VALUE FROM POINTER TO ROOT

- DELETE MIN
  - SWITCH POSITION OF ROOT AND LAST NODE OF HEAP
  - REMOVE THE LAST ELEMENT
  - WHILE ROOT IS LARGER THAN ITS CHILDREN, SWITCH ROOT AND SMALLER CHILDREN

- DELETE
  - SAME AS DELETE MIN
    - BUT WE SWAP DELETED NODE (INSTEAD OF ROOT) AND LAST VALUE

- DATA REPRESENTATION
  - ARRAY



- TIME COMPLEXITIES
  - INSERT $O(\log(n))$
  - DELETE $O(\log(n))$
  - ACCESS_MIN $O(1)$
  - DELETE_MIN $O(\log(n))$
  - DECREASE_KEY $O(\log(n))$
  - BUILD_HEAP $O(n)$
  - MERGE $O(n)$

- D-ARY HEAP
  - DIFFERENT NUMBER OF CHILDREN
  - OPERATIONS ARE ANOLOGICAL
  - TIME COMPLEXITY HAS DIFFERENT BASE OF LOGARITHM
  - THEY ARE FASTER THAN BINARY HEAP FOR HEAP SIZES EXCEEDING THE SIZE OF CACHE

- BINOMIAL HEAP
  - COLLECTION OF BINOMIAL ~~HEAPS~~ TREES
    - OF DEGREES $i = 0, \dots, \lfloor \log(n) \rfloor$
    - THERE CAN BE EITHER ONE OR ZERO BINOMIAL TREES FOR EACH DEGREE
    - EACH TREE OBEYS HEAP PROPERTY
      - KEY FOR CHILD IS LOWER THAN FOR PARENT
    - IT IS DEFINED RECURSIVELY
      - ORDER 0
        - SINGLE NODE
      - DEGREE k HAS A ROOT NODE WHOSE CHILDREN ARE ROOTS OF BINOMIAL TREES OF DEGREES $k-1, k-2, \dots, 2, 1, 0$

- FOR BINOMIAL TREE OF DEGREE k IT HOLDS
  - HEAP PROPERTY
  - HEIGHT IS k
  - ROOT HAS k CHILDREN
  - THERE ARE 2^k NODES
- REPRESENTATION
  - ROOTS CAN BE IN LINKED LIST
  - POINTER TO MIN (ALWAYS ROOT)
  - MUST BE UPDATED IN $O(\log n)$
- INSERT
  - CREATE NEW HEAP OF DEGREE 0 (ONE NODE)
  - MERGE IT WITH CURRENT HEAP
- ACCESS MIN
  - RETURN VALUE OF POINTER
- MERGE
  - ADD HEAPS OF SAME DEGREES
  - THERE CAN BE CARRY TO HIGHER DEGREE
  - HEAP WITH SMALLER ROOT IS ROOT, THE SECOND IS APPENDED AS CHILD TO NEW ROOT
  - UPDATE MIN POINTER

- DELETE MIN
  - FIND TREE WITH MIN
  - REMOVE MIN IN ROOT
    - THIS CREATES NEW TREES OUT OF CHILDREN OF MIN
  - MERGE NEW TREES WITH THE REST OF BINOMIC HEAP

- DECREASE KEY
  - SAME AS IN BINARY HEAP

- DELETE
  - DECREASE TO $-\infty$
  - DELETE MINIMAN IN HEAP BY DELETE MIN

- MERGE
  - $O(\log(n))$

- INSERT
  - $O(\log(n))$
  - AMORTIZED $O(1)$

- ACCESS_MIN
  - $O(1)$

- DELETE_MIN
  - $O(\log(n))$

- DECREASE_key
  - $O(\log(n))$

- DELETE
  - $O(\log(n))$

- AMORTIZED COMPLEXITY
  - TIME TO PERFORM A SEQUENCE OF DATA-STRUCTURE OPERATIONS
  - IT IS AVERAGED OVER ALL OPERATIONS PERFORMED
  - WE CAN SHOW, THAT SEQUENCE OF OPERATIONS CAN BE FAST, EVEN IF SOME OPERATION IS EXPENSIVE

- FIBONACCI HEAP
  - LOSELY BASED ON BINOMIAL HEAP
  - HAVE MORE RELAXED STRUCTURE
  - OPERATIONS NOT DELETING NODES RUNS IN $O(1)$ AMORTIZED
  - DELETE AND DELETE_MIN RUNS IN $O(\log(n))$ AMORTIZED
  - BUT NOT SUITABLE FOR REAL TIME SYSTEMS
    - IN BAD CASES HAS LINEAR COMPLEXITY
  - COLLECTION OF TREES WITH HEAP PRIORITY
  - IN EXTREME CASES EACH TREE HAS SINGLE NODE
    - THIS ALLOWS FOR LAZY EXECUTION OF OPERATIONS
      - MERGING CAN BE JUST CONCATENATION
      - DECREASE KEY SOMEBODY JUST CUTS THE NODE FROM ITS PARENTS

- GRAPH ISOMORPHISM

  - TWO GRAPHS $G_1 = (V_1, E_1)$ AND $G_2 = (V_2, E_2)$ ARE ISOMORPHIC IF THERE IS BIJECTION $f: V_1 \to V_2$ SUCH THAT

    $$\forall x, y \in V_1 : \{f(x), f(y)\} \in E_2 \iff \{x, y\} \in E_1$$

  - THE MAPPING f IS SAID TO BE ISOMORPHISM BETWEEN G_1 AND G_2

- INVARIANT

  - FUNCTION ϕ

    - $\forall G_1, G_2 \in F : \phi(G_1) = \phi(G_2) \Leftarrow G_1$ IS ISOMORPHIC TO G_2

  - NUMBER OF VERTICES

  - SORTED LIST OF VERTEX DEGREES

- CERTIFICATE

  - FUNCTION SUCH THAT

    $$\forall G_1, G_2 \in F : CERT(G_1) = CERT(G_2) \Leftrightarrow G_1 \text{ IS ISOMORPHIC TO } G_2$$

- TREE CERTIFICATE

  - USE 0 1 ON LEAVES AND PROPAGATE TO ROOT

  - USE LEXICOGRAPHICAL ORDER

  - LENGTH IS $2|V|$

  - NUMBERS OF 1s AND 0s THE SAME

  - RECONSTRUCT

    - WITH 0 CREATE NEW NODE

    - WITH 1 BACKTRACK

- COMBINATORIAL ALGORITHMS

- RANKING FUNCTION

- BIJECTION

$$RANK: S \rightarrow \{0, \ldots, |S|-1\}$$

- UNRANK FUNCTION IS INVERSE TO RANK FUNCTION

- GIVEN RANKING FUNCTION RANK DEFINED ON S
THE SUCCESSOR FUNCTION SATISFIES THE FOLLOWING RULE

$$SUCCESSOR(s) = t \iff RANK(t) = RANK(s) + 1$$

- USAGE

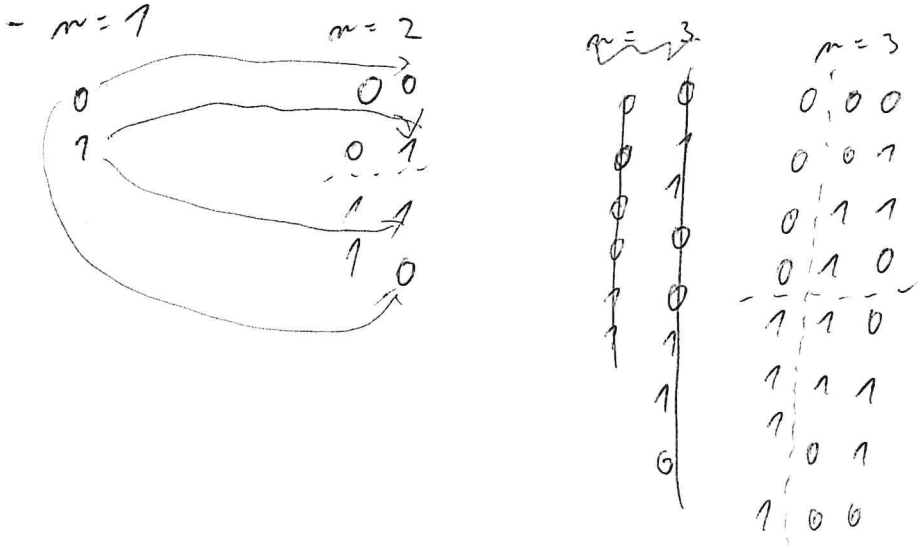
- STORING COMBINATORIAL OBJECTS INSTEAD OF
COMBINATORIAL STRUCTURE

- GENERATION OF RANDOM OBJECT FROM S ENSURING EQUAL
PROBABILITY $\frac{1}{|S|}$

- SUBSETS

T	$X(T) = [x_2, x_1, x_0]$	RANK(T)
$\emptyset$	[0 0 0]	0
{3}	[0 0 1]	1
{2}	[0 1 0]	2
{2,3}	[0 1 1]	3
{1}	[1 0 0]	4
{1,3}	[1 0 1]	5
{1,2}	[1 1 0]	6
{1,2,3}	[1 1 1]	7

- GRAY CODE
  - TWO SUCCESIVE VALUES DIFFERS ONLY IN SINGLE BIT



- PSEUDORANDOM NUMBER GENERATORS
  - DETERMINISTIC FORMULA
  - OUTPUT LOOKS RANDOM
  - ALGORITH WHICH PRODUCES A SEQUENCE

    $\{X_n\} = X_0, X_1, X_2 \ldots$
  - IMPORTANT STATISTICAL PROPERTIES
    - UNIFORMITY
      - EACH NUMBER HAS THE SAME PROBABILITY
      - NUMBER DRAWN BY PROBABILITY DENSITY FUNCTION

$$f(x) = \begin{cases} \frac{1}{b-a} & x \in [a,b] \\ 0 & \text{ELSEWHERE} \end{cases}$$

    - INDEPENDENCE
  - GOOD GENERATOR HAS LONG PERIOD $b-a$ AND USES IT WHOLE
  - AND IS FAST
    - HAS EASY FORMULA

- TWO TASKS
  - GENERATE NUMBER $[a; b]$
  - GENERATE NUMBER $[0; 1]$
- WE CAN USE TASK 1 FOR TASK 2
  - USE FORMULA $g(n) = \frac{x_n - a}{b - a - 1}$

- LINEAR CONGURENTIAL GENERATOR

$$0 \leq x_0 < M$$

$$x_{n+1} = (Ax_n + C) \bmod M \quad n \geq 0$$

- MAXIMUM PERIOD LENGTH
  - LENGTH OF PERIOD IS MAXIMUM (EQUAL TO M) IF ALL FOLLOWING HOLDS
    1. C AND M ARE COPRIMES 9 ? ✓
       9 3 ✗
    2. A-1 IS DIVISIBLE BY EACH PRIME FACTOR OF M
    3. IF 4 DIVIDES M THEN ALSO 4 DIVIDES A-1
- PROBLEM IS THAT LOW ORDER BITS HAS LACK OF RANDOMNESS
  - GENERATES ODD, EVEN, ODD, EVEN, ODD...
  - OR $x_n \bmod 3 = 1, 0, 2, 1, 0, 2, 1, 0, 2, 1, 0, 2 \ldots$
  - WE CAN IMPROVE IT BY OPERATION

$x_n \bmod W$ FOR SOME $W < \max_{n \geq 0} \{x_n\}$

- COMBINED LINEAR CONGRUENTIAL GENERATOR

  - LET THERE BE r LINEAR CONGRUENTIAL GENERATORS

    $$0 \leq y_{k,0} < m_k$$

    $$y_{k,n+1} = (A_k y_{k,n} + C_k) \bmod m_k, \quad n > 0$$

    $$1 \leq k \leq r$$

  - COMBINED LCG IS SEQUENCE $\{x_n\}$ DEFINED BY

    $$x_n = (y_{1,n} - y_{2,n} + y_{3,n} - y_{4,n} + \dots (-1)^{r-1} y_{r,n}) \bmod (m_1 - 1) \quad n \geq 0$$

  - MAXIMUM POSSIBLE PERIOD LENGTH IS

    $$(m_1 - 1)(m_2 - 1) \dots (m_r - 1) / 2^{r-1}$$

    - BUT IT IS NOT ALWAYS ATTAINED

- LEHMER GENERATOR

  $0 < x_0 < M$ $\quad x_0$ COPRIME TO M

  $x_{n+1} = A x_n \bmod M \quad n \geq 0$

  - SEQUENCE PERIOD LENGTH IS MAXIMAL AND EQUALS TO $M-1$ IF M IS PRIME
  - A IS A PRIMITIVE ROOT OF THE MULTIPLICATIVE GROUP OF INTEGERS MODULO M
    - THERE IS NO EFFECTIVE WAY KNOWN NOW TO FIND IT
    - BUT SPECIAL CASES HAVE BEEN STUDIED

- BLUM BLUM SHUB GENERATOR

$2 \leq x_0 < M$ $\quad x_0$ COPRIME TO M

$x_{n+1} = x_n^2 \mod M$

- M IS A PRODUCT OF TWO BIG PRIMES P AND Q

- PRIMES COUNTING FUNCTION

- $\pi(n)$

- COUNTS NUMBER OF PRIMES LESS THAN OR EQUAL TO n

- $\pi(10) = 4 : 2, 3, 5, 7$

- ESTIMATE

- $\frac{n}{\ln(n)} < \pi(n) < 1{,}25506 \frac{n}{\ln n}$ FOR $n > 16$

- LIMIT BEHAVIOUR

$$\lim_{n \to \infty} \frac{\pi(n)}{\frac{n}{\ln n}} = 1$$

- EULER'S TOTIENT FUNCTION $\varphi(n)$

- COUNTS INTEGERS LESS THAN OR EQUAL TO n THAT ARE RELATIVELY PRIME TO n

- $\varphi(21) = 12 : 1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20$

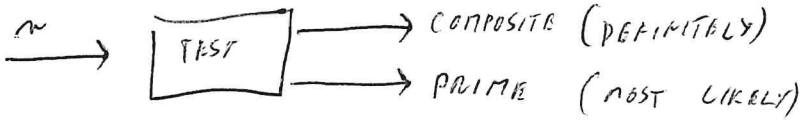
- MERSENNE PRIME M_n

- PRIME IN FORM $2^n - 1$

$n = 3 \quad M_3 = 2^3 - 1 = 7$

$n = 7 \quad M_7 = 2^7 - 1 = 127$

- RANDOMIZED PRIMALITY TESTS



- FERMAT LITTLE THEOREM

  - IF p IS PRIME AND $0 < a < p$ THEN

    $a^{p-1} \equiv 1 \pmod{p}$

- FERMAT PRIMALITY TEST

  FOR k:

    a = RANDOM INT $[2, n-2]$

    IF $a^{n-1} \not\equiv 1 \pmod{n}$ THEN COMPOSITE

  RETURN PRIME

  - BUT THERE ARE INFINITELY MANY COMPOSITE NUMBERS FAILING THIS TEST ALWAYS

- MILLER-RABIN PRIMALITY TEST

  - IF p IS PRIME AND $x^2 \equiv 1 \pmod{p}$

    - THEN $x \equiv 1 \pmod{p}$ OR $x \equiv -1 \pmod{p}$

  - LET $n > 2$ BE PRIME $n - 1 = 2^r \cdot d$ WHERE d is ODD, $1 < a < n-1$

    THEN EITHER $a^d \equiv 1 \pmod{n}$ OR $a^{2^s \cdot d} \equiv -1 \pmod{n}$ FOR SOME $0 \leq s \leq r-1$

  - ALGORITHM

    COMPUTE r, d SUCH THAT d IS ODD AND $2^r \cdot d = n-1$

    FOR $i = 1$ TO k

      a = RANDOM INTEGER IN $[2, n-2]$

      $x = a^d \bmod n$

      IF $x = 1$ OR $x = n-1$ THEN GOTO END OF LOOP

      FOR $j = 1$ TO $r-1$

        $x = x^2 \bmod n$

        IF $x = 1$ THEN RETURN COMPOSITE

        IF $x = n-1$ THEN GOTO END OF LOOP

      RETURN COMPOSITE

    RETURN PRIME

- TIME COMPLEXITY : $O(k \log^3 n)$
- IF n IS COMPOSITE, THAN TEST DECLARES n AS PRIME WITH PROBABILITY AT MOST 4^{-k}

- AKS PRIMALITY TEST
  - FIRST KNOWN DETERMINISTIC POLINOMIAL-TIME PRIMALITY TEST
  - TIME COMPLEXITY $O(\log^6 n)$
  - BIG THEORETICAL IMPORTANCE, BUT NOT USED IN PRACTICE

- FINITE AUTOMATA
  - ALPHABET
    - FINITE SET OF SYMBOLS
  - WORD
    - FINITE SEQUENCE OF SYMBOLS OF ALPHABET
  - LANGUAGE
    - SET OF WORDS
    - SPECIFIED BY
      - LISTING ALL THE WORDS
        - BUT MAY BE INFINITE SET
      - DESCRIPTION IN HUMAN LANGUAGE
      - BY FINITE AUTOMATON
        - 5-TOUPLE (A, Q, σ, S_0, Q_F)
          - A - ALPHABET
          - Q - SET OF STATES
          - σ - TRANSITION FUNCTION $\sigma: Q \times A \to Q$
          - S_0 - START STATE
          - Q_F - SET OF FINAL STATES (UNEMPTY) $Q_F \subseteq Q$

- IF LAST SYMBOL IS READ AND AUTOMATON TRANSLATES TO FINAL STATE, THE WORD IS ACCEPTED
  - OTHERWISE IT IS NOT ACCEPTED

- LANGUAGE ACCEPTED BY AUTOMATON
  - SET OF ALL WORDS ACCEPTED BY IT

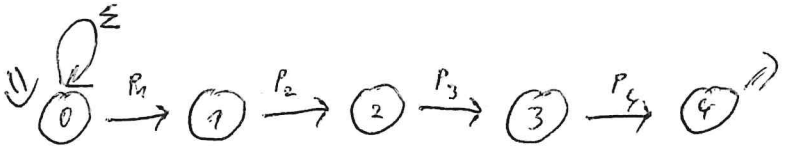
- NON-DETERMINISTIC FINITE AUTOMATA
  - DIFFERENCE IS IN TRANSITION FUNCTION
    $\delta : Q \times \Sigma \to P(Q)$ WHERE $P(Q)$ IS POWERSET OF Q
    - CAN TRANSITION TO SEVERAL STATES AT ONCE
  - IT CAN BE TRANSFORMED INTO DFA
  - EACH STATE OF DFA IS SUBSET OF STATES OF NFA (EMPTY SET TO
  - STATE OF DFA IS ~~ACCEPTING~~ FINAL STATE IF IT CONTAINS SOME FINAL STATE OF NFA

- TEXT SEARCH
  - NAIVE
    - WE HAVE PATTERN
    - WE COMPARE SYMBOLS OF PATTERN TO TEXT
    - IF MISSMATCH OCCURS, WE SHIFT THE PATTERN BY ONE POSITION

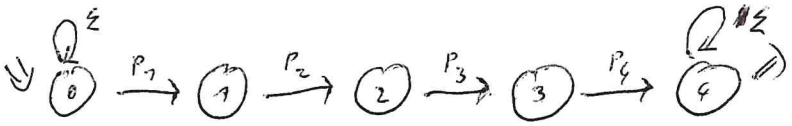
TEXT	a	b	c	d	e	f
PATTERN	a ✓	b ✓	c ✓	x ✗		

  - NFA

- ACCEPTS WORD $P_1 P_2 P_3 P_4$ EXACTLY

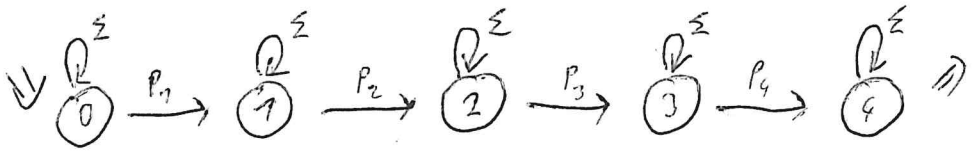


- ACCEPTS WORD WITH SUFFIX $P_1 P_2 P_3 P_4$



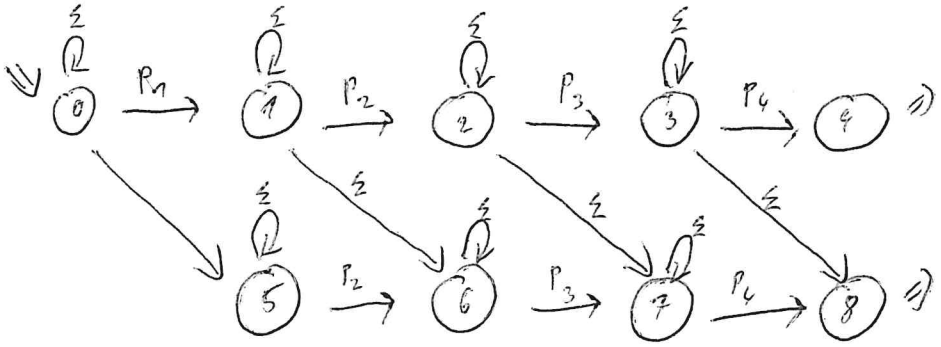
- ACCEPTS WORD WITH SUBSTRING $P_1 P_2 P_3 P_4$ ANYWHERE IN IT

-



- ACCEPTS WORD WITH SUBSEQUENCE $P_1 P_2 P_3 P_4$ ANYWHERE IN IT

-



- ACCEPTS WORD WITH SUBSEQUENCE $P_1 P_2 P_3 P_4$ ANYWHERE IN IT AND THE SEQUENCE CAN BE ALTERED BY ONE SYMBOL (HAMMING DISTANCE IS 1 AT MOST)

- ONLY LANGUAGES ABLE TO BE PROCESSED BY NFA/DFA ARE REGULAR LANGUAGES
  - FOR EXAMPLE LANGUAGE OF WELL FORMED PARENTHESIS IS NOT REGULAR (IT IS CONTEXT FREE)

- OPERATIONS ON REGULAR LANGUAGES
  - $L_1 \cup L_2$
    - SET OF ALL WORDS WHICH ARE IN L_1 OR L_2
  - $L_1 \cdot L_2$
    - CONCATENATION
    - ALL WORDS $w = w_1 w_2$, $w_1 \in L_1$, $w_2 \in L_2$
  - L_1^*
    - KLEENE STAR
    - CONCATENATION OF ANY WORD OF L_1

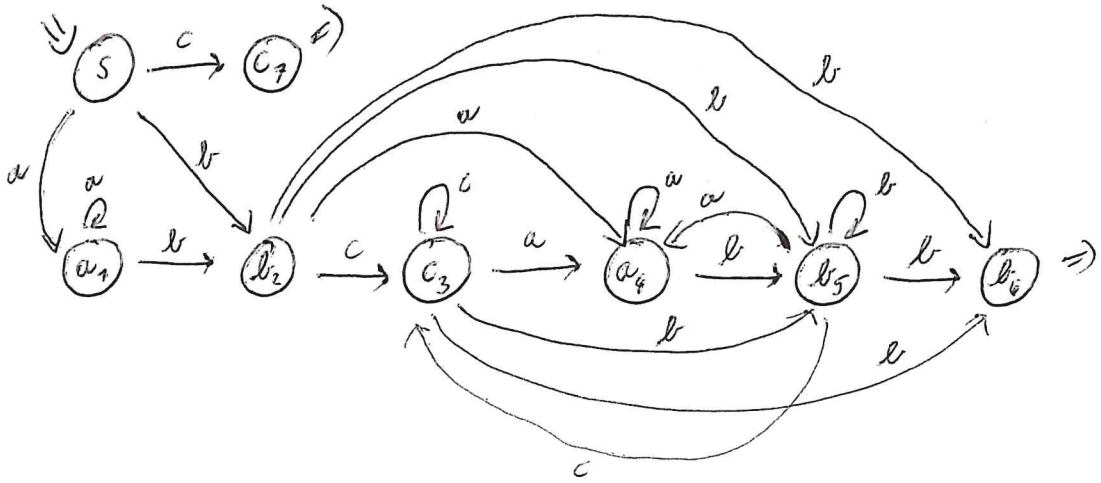
- ALL OF THESE OPERATIONS CREATES REGULAR LANGUAGES TOO

- REGULAR EXPRESSION TO NFA

$$a^*b(c+a^*b)^*b+c$$

-ADD INDICES

$$a_1^*b_2(c_3+a_4^*b_5)^*b_6+c_7$$

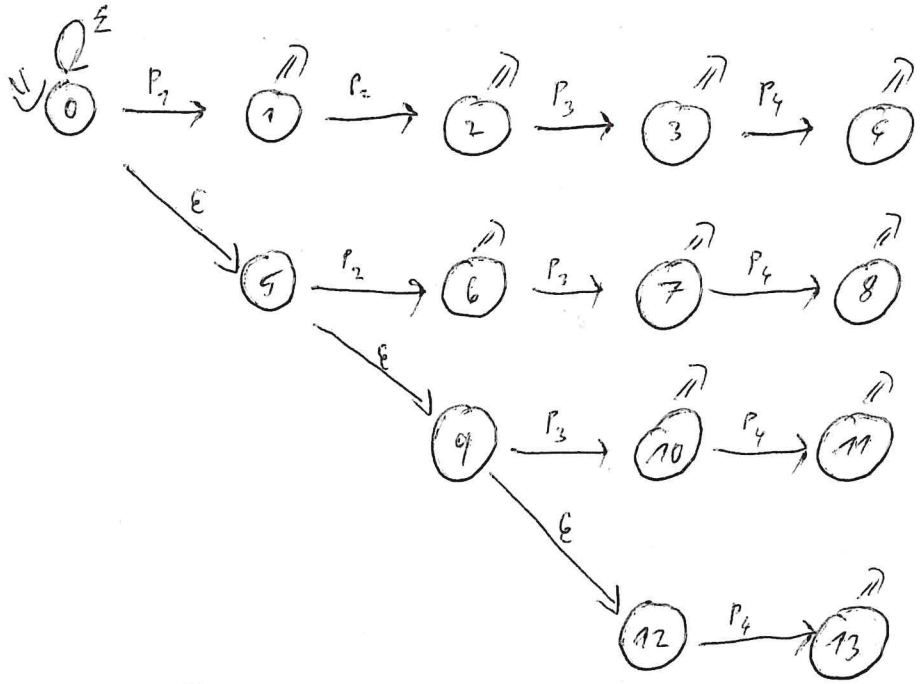


- WE CAN ADD ε TRANSITIONS

- ε-CLOSURE (P)

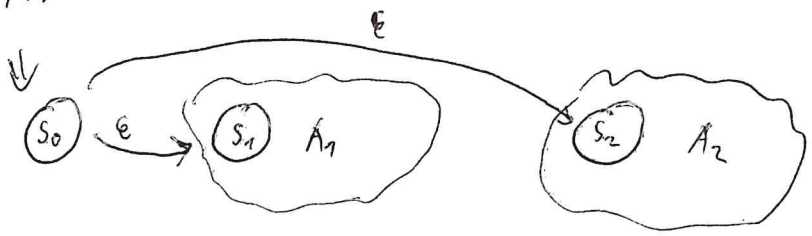
- SET OF ALL STATES WHICH CAN BE ACCESSED FROM P BY USING ONLY ε-TRANSITIONS

- BUT WE CAN ALWAYS CONSTRUCT EQUIVALENT NFA WITHOUT ε-TRANSITIONS

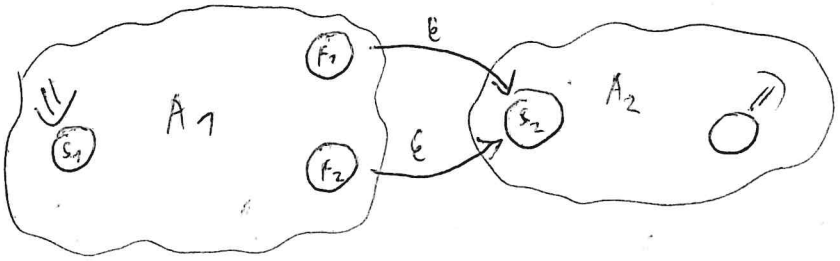


- ACCEPTS ANY UNEMPTY SUBSTRING OF PATTERN $P_1P_2P_3P_4$ OVER ALPHABET Σ

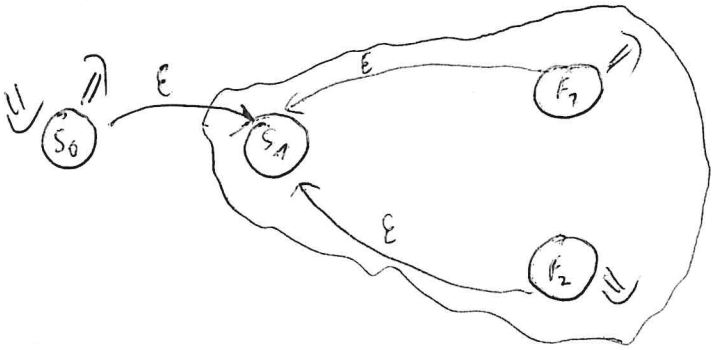
- AUTOMATON ACCEPTING UNION OF L_1 AND L_2



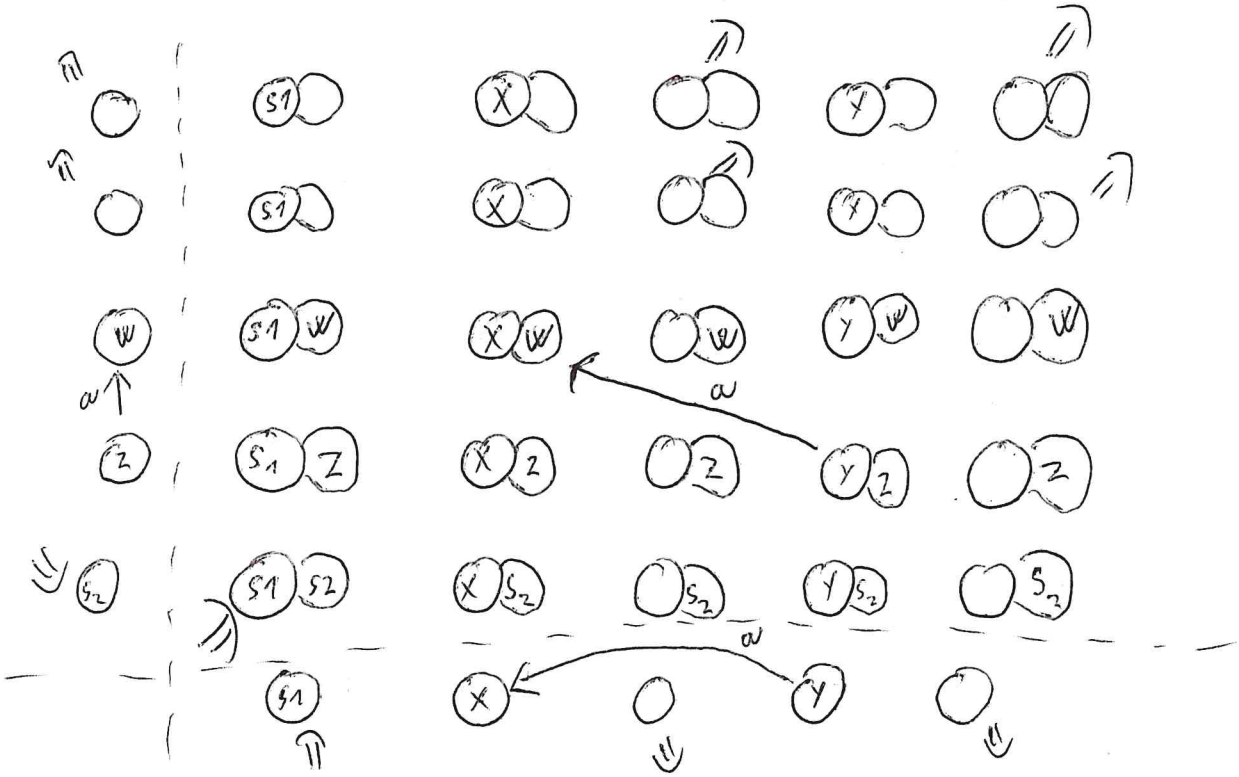
- AUTOMATON ACCEPTING CONCATENATION OF L_1 AND L_2



- AUTOMATON ACCEPTING ITERATION OF LANGUAGE L_1

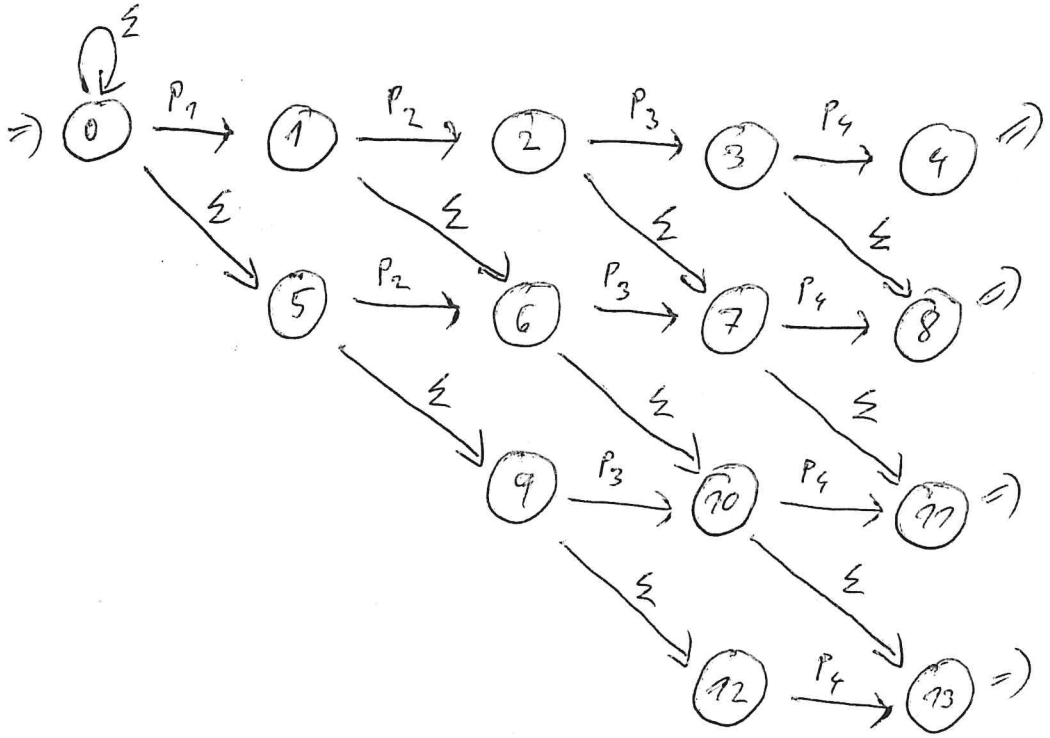


- AUTOMATON ACCEPTING INTERSECTION OF L_1 AND L_2



- HAMMING DISTANCE

  - MINIMAL NUMBER OF REWRITE OPERATIONS WHICH REWRITES ONE STRING INTO OTHER
  - WE CAN'T INSERT OR DELETE CHARACTERS
  - DEFINED ONLY FOR STRINGS OF THE SAME LENGTH
  - LOKOMOTIVA / VYKOLENILA — DISTANCE = 6



  - DETECT STRING WITH HAMMING DISTANCE MAXIMALLY 3
  - WE CAN IMPROVE IT BY DETECTING $\overline{P_i}$ INSTEAD OF Σ ON DIAGONAL LINES

- LEVENSHTEIN DISTANCE

  - NUMBER OF EDIT OPERATION TRANSFORMING STRING A TO B

    - OPERATIONS REMOVE, INSERT, REWRITE

  - STRINGS CAN BE OF DIFFERENT LENGTH

  - THE EXACT USED OPERATIONS CAN VARY, BUT THE DISTANCE IS JUST ONE

  - BRUXELLES — DELETE X
    BETELGEUSE — REWRITE R→E, U→T, L→G
    INSERT U, E

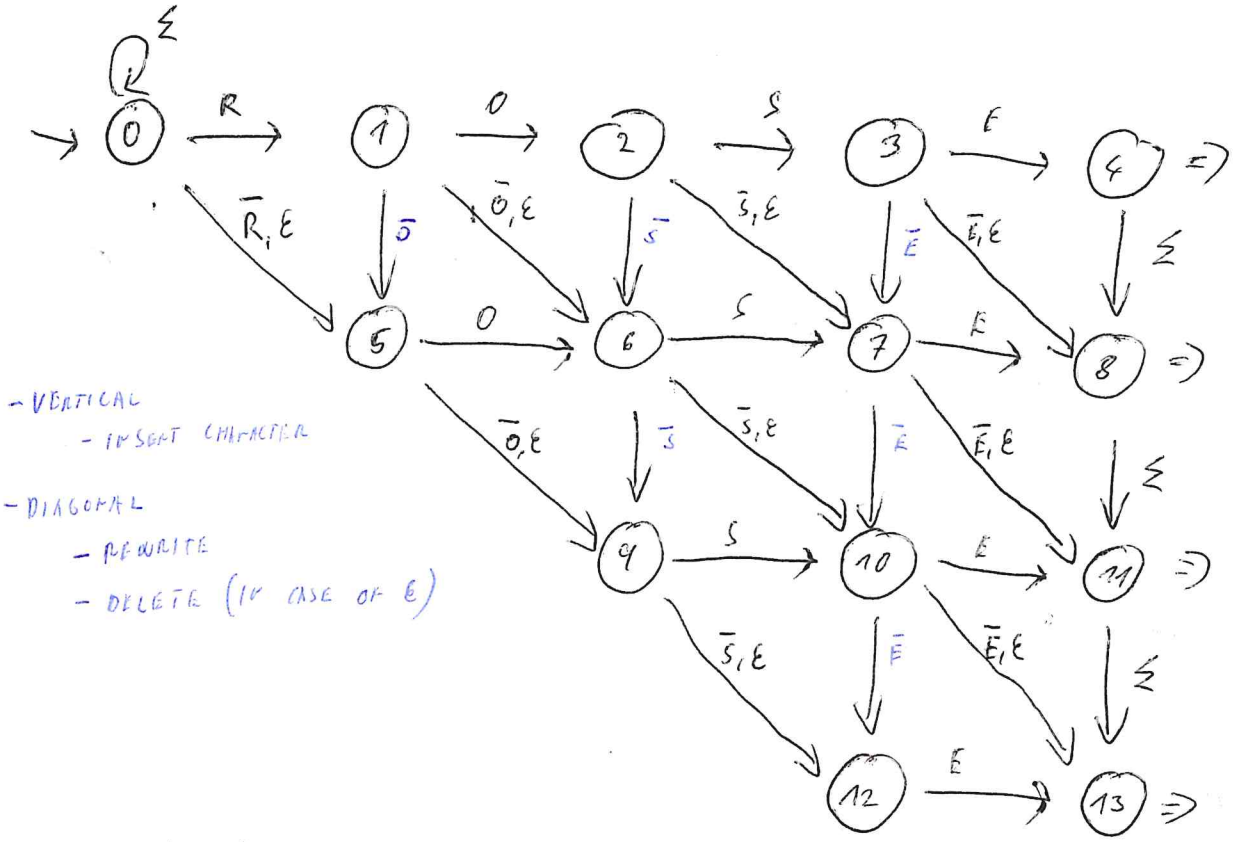
  - CAN BE CALCULATED BY DYNAMIC PROGRAMING

    IF ~~A[i]=B[j]~~ A[i] == B[j]
    D[i][j] = D[i-1][j-1]

    ELSE

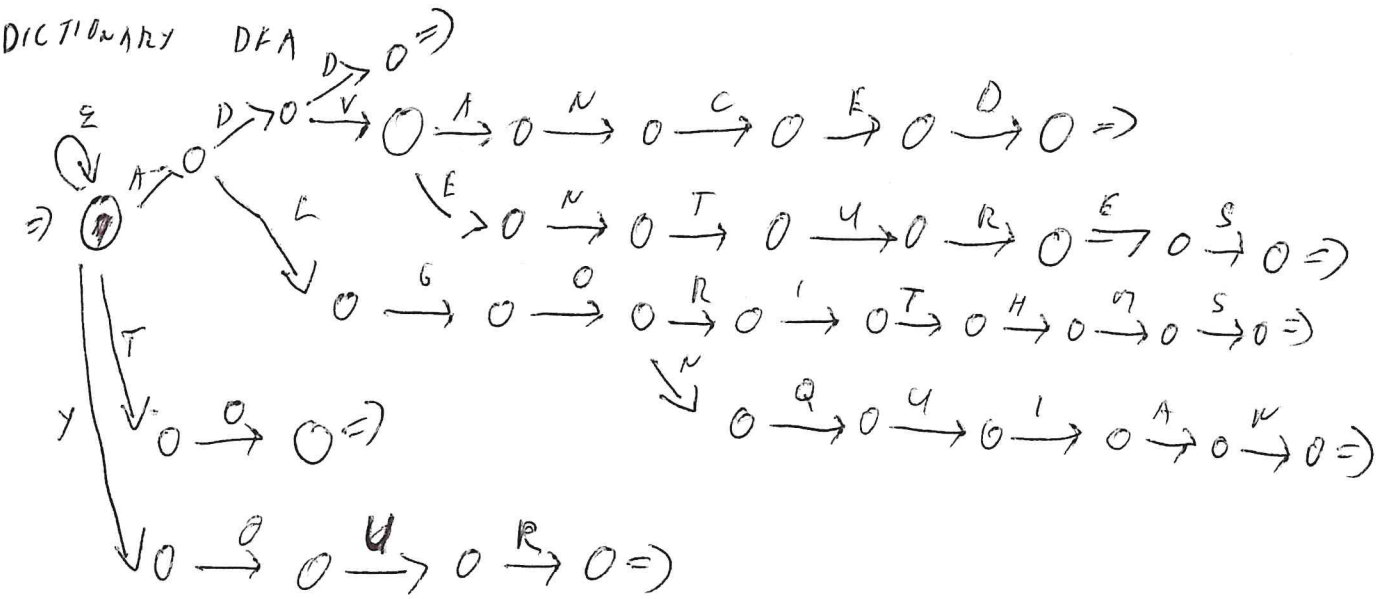
    $$D[i][j] = 1 + MIN(D[i-1][j-1], D[i-1][j], D[i][j-1])$$

		B	E	T	E	L	G	E	U	S	E
	0	1	2	3	4	5	6	7	8	9	10
B	1	0	1	2	3	4	5	6	7	8	9
R	2	1	1	2	3	4	5	6	7	8	9
U	3										
X	4										
E	5										
L	6										
L	7										
E	8										
S	9										



- FINDS STRINGS IN TEXT WITH LEVENSTAIN DISTANCE MAX 3 FROM "ROSE"

- DICTIONARY DFA



- TRANSFORMATION TO DFA DOESN'T INCREASE NUMBER OF STATES

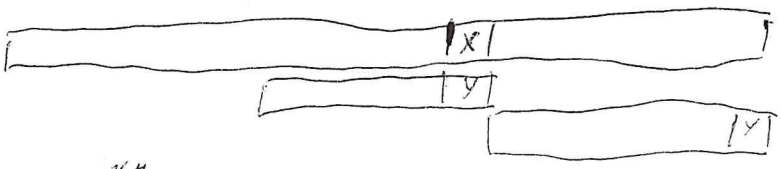
- BOYER - MOORE TEXT SEARCH

- START THE SEARCH FROM THE END OF PATTERN

AM TEXT

PATTERN

PATTERN AFTR MISMATCH

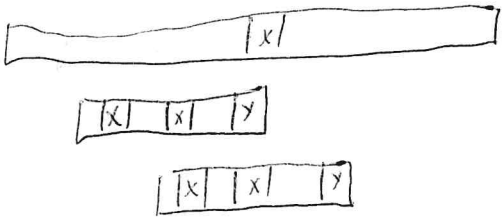


PATTERN DOESN'T CONTAIN X

- IDEAL CASE

- IF X IS IN PATTERN

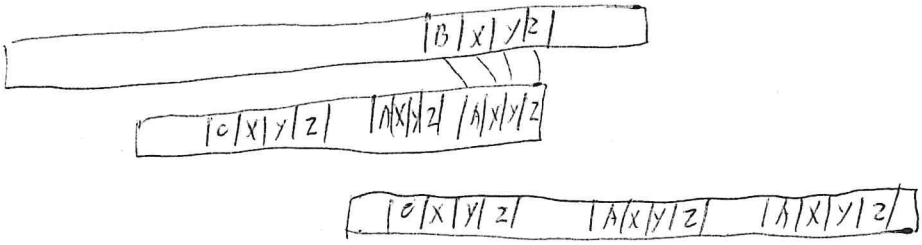
- SHIFT BY BASED BAD CHARACTER SHIFT VALUE IN TABLE



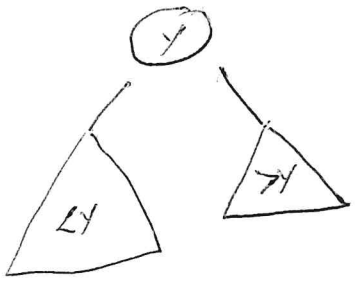
- IF MISSMATCH APPEARS IN THE MIDDLE OF PATTERN

- SITUATION COMPLICATED

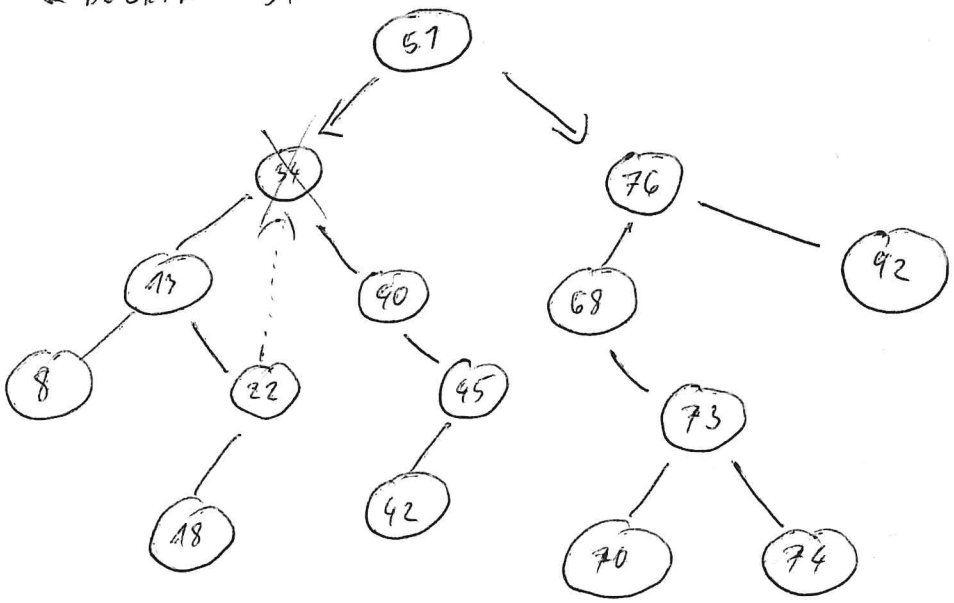
- SHIFT TO NEXT OCURANCE OF SUFFIX IN PATTERN



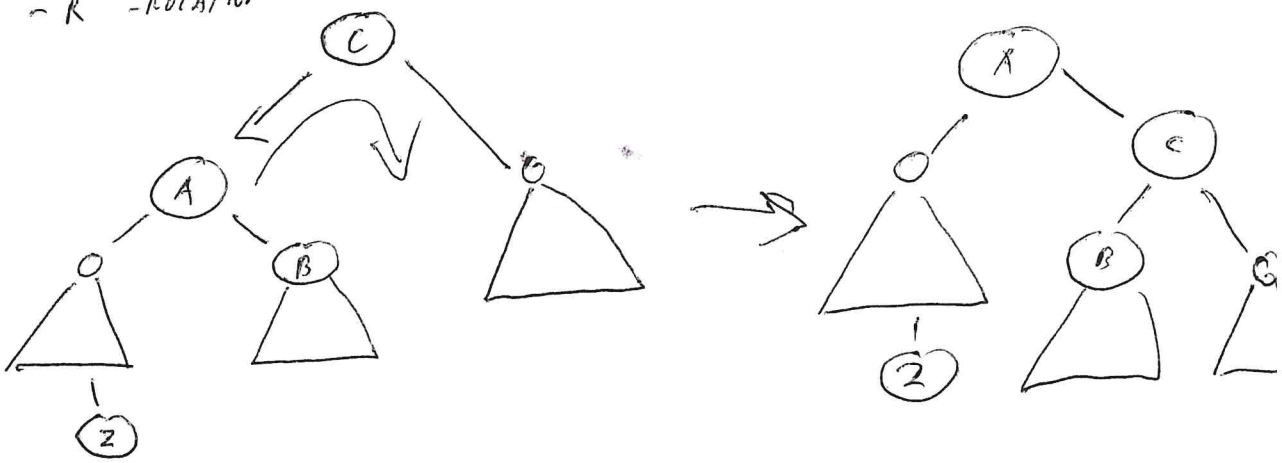
- BINARY SEARCH TREE



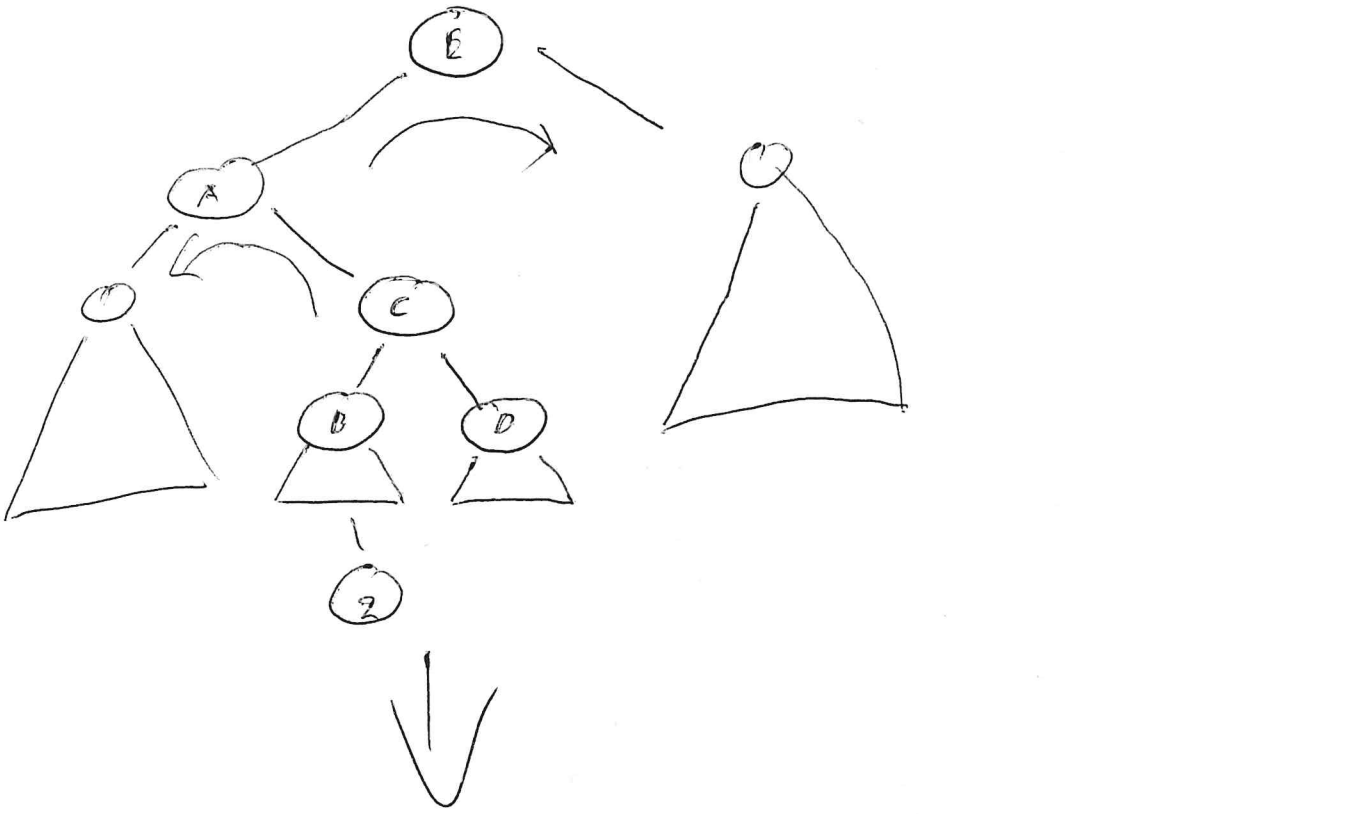
- DELETE 34

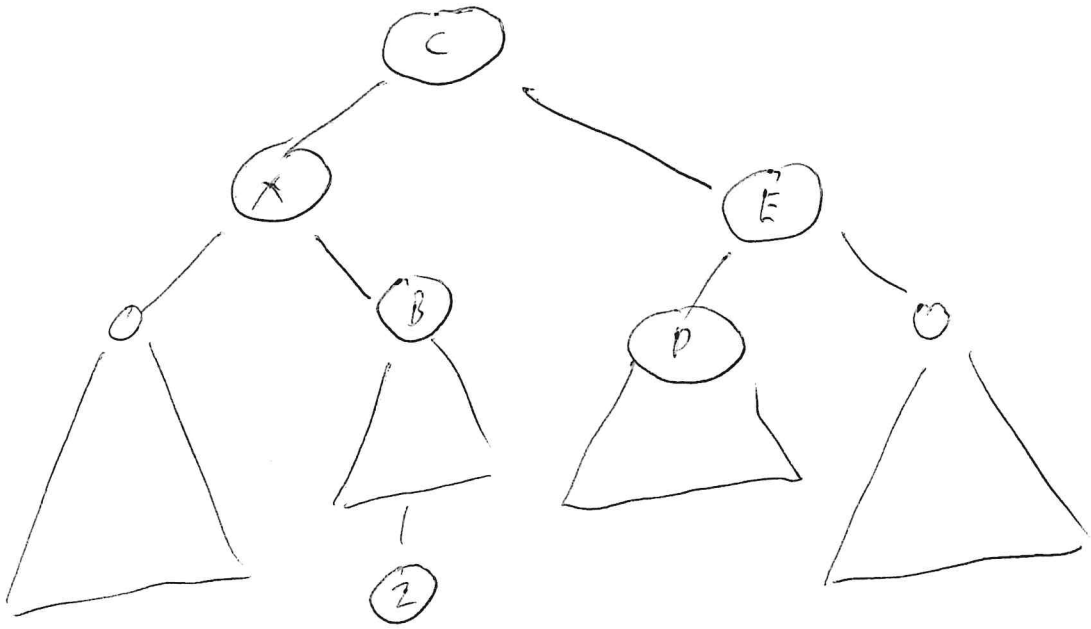


- AVL
  - KEEPS BALANCED
  - DEPTH OF LEFT AND RIGHT CHILD
  - ROTATE IF $L - R \geq 2$
    - R - ROTATION



- LR ROTATION



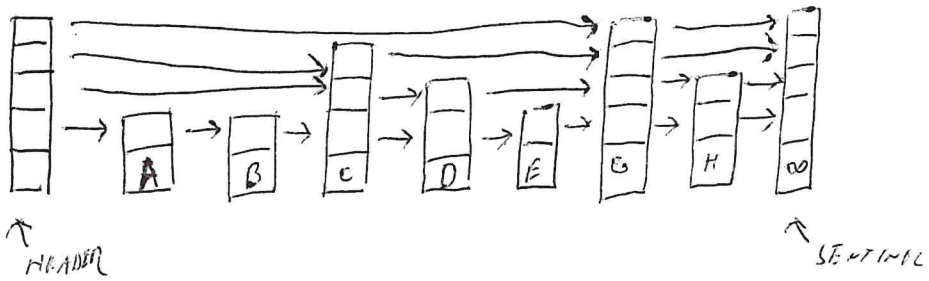


- SKIP - LIST

  - LINKED LIST WITH $\log(n)$ SEARCH

  - SORTED

  - EACH NODE CONTAINS VARIABLE NUMBER OF LINKS



  - LEVEL OF NODE IS SELECTED BY RANDOM

    - FLIP A COIN

      - INCREASE THE VALUE UNTIL YOU OBSERVE THE TAIL

    - YOU CAN ALSO SELECT DIFFERENT PROBABILITY

- SEARCH

  - START IN HIGHEST LEVEL

  - TRAVERSE UNTIL NODE > SEARCH

    - IF SO BACK TRACK AND GO ONE LEVEL DOWN

- INSERT

  - SAME AS SEARCH

  - DETERMINE LEVEL

  - ADD NEW CONNECTIONS

- CHOOSING P
  - 0,5
    - LOG(N) SEARCH TIME
    - 2 POINTERS PER NODE
  - 0,25
    - EMPIRICALY TESTED
    - ROUGHLY SAME SEARCH TIME
    - BUT ONLY 1,33 POINTERS PER NODE
    - SEARCH TIME CAN VARY
      - BUT MEMORY SAVINGS ARE WORTH IT
- AVERAGE NUMBER OF LINKS IS

$$N \cdot \frac{1}{1-P}$$

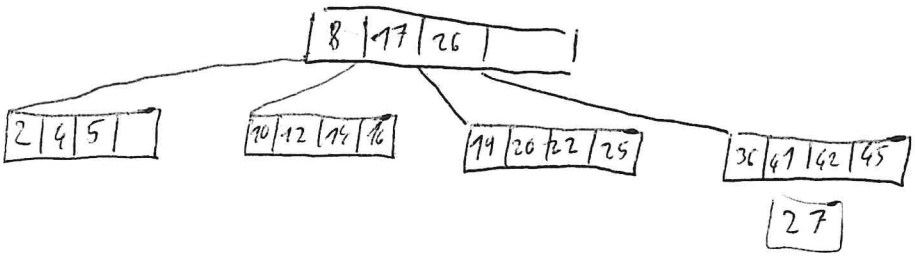
- TIME COMPLEXITY OF SEARCH, INSERT AND DELETE ARE

$O(\lg n)$

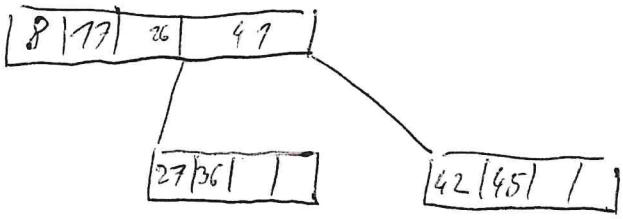
- P CAN CHANGE THE VARIABILITY OF SEARCH TIME

B- TREE

- ALL LENGTHS OF PATHS ARE EQUAL
- PERFECTLY BALANCED
- SAME SIZE OF ALL NODES
- EACH NODE (EXCEPT ROOT) CONTAINS $[k, 2k]$ KEYS
- ALSO IT HAS $[k+1, 2k+1]$ CHILDREN
- MULTI PHASE STRATEGY
- INSERT 27



- SORT
- SELECT MEDIAN (41)
- DIVIDE NODE
- MEDIAN UP



- IF NECESSARY DO RECURSIVELY UP

- DELETE
  - LEAF ENOUGH CAPACITY
    - OK
  - INNER NODE ENOUGH CAPACITY
    - SUBSTITUTE BY SMALLEST BIGGER VALUE

- DELETE TOO SMALL CAPACITY
  - REMOVE VALUE
  - MERGE WITH KEY
  - SELECT MEDIAN → NEW KEY

- SINGLE PHASE STRATEGY
  - GOES FROM ROOT TO LEAVES

- B + TREE
  - ANALOGOUS TO B-TREE
  - RECORDS (ACTUAL VALUES) ARE STORED ONLY IN LEAVES
  - INTERNAL NODES ONLY GUIDES THE SEARCHES
  - LEAVES CONNECTED TOGETHER
    - FORMS LINKED LIST
    - INFORMATIONS CAN BE RETRIEVED SEQUENTIALY TOO
    - CAN SUPPORT RANGE-SEARCH QUERIES
  - EACH ROUTE VALUES WAS KEY ONCE (WHEN VALUE WAS ADDED)
    - IF THE VALUE IS DELETED, THE ROUTE LIVES ON
  - COMPLEXITY
    - FIND, INSERT, DELETE
      - $\Theta(b \log_b n)$
        - n IS NUMBER OF ELEMENTS IN TREE
        - b IS BRANCHING FACTOR
    - RANGE SEARCH
      $\Theta(b \log_b n + \frac{k}{b})$
      - k IS RANGE OF QUERY

- SPLAY TREE

  - DATA WHICH WAS ACCESSED RECENTLY WILL PROBABLY BE ACCESSED AGAIN

  - ACCESSED NODES ARE SPLAYED (ROTATED) TO THE ROOT

  - FIND

    - FIND NODE AND SPLAY IT TO ROOT

  - INSERT

    - INSERT NODE AND SPLAY IT TO ROOT

  - DELETE

    - SPLAY TO ROOT AND DELETE IT

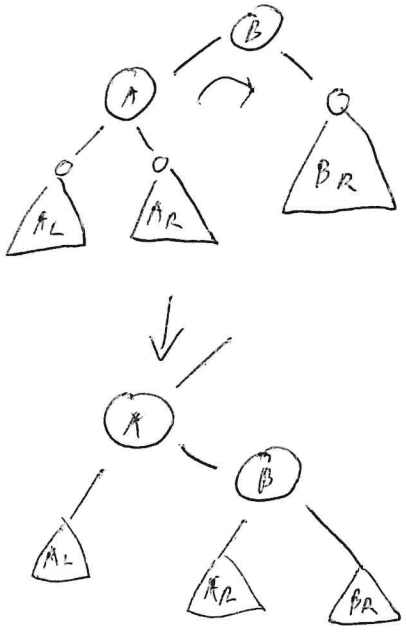
- ZIG, ZIG-ZIG AND ZIG-ZAG ROTATIONS

- OPERATION TIMES ARE $O(n)$, TREE HEIGHT CAN BE $\Theta(n)$

- AMORTIZED RUN OF ALL OPERATIONS IS $O(\log(n))$

- ZIG

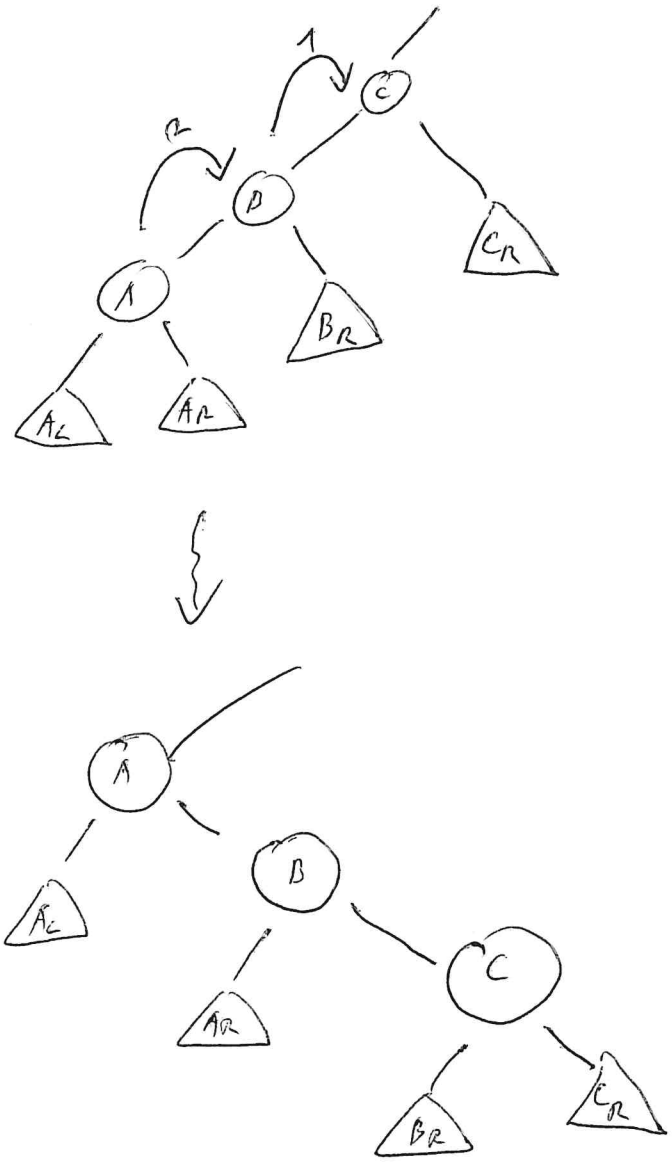
  - AS L OR R ROTATIONS



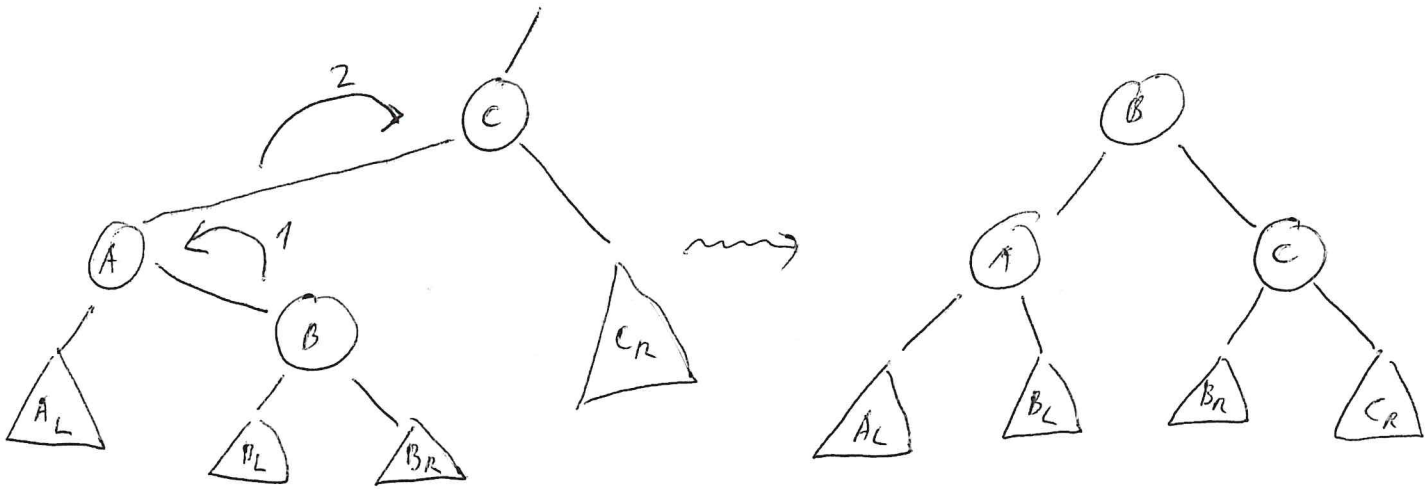
- ZIG - ZIG

- AS R R

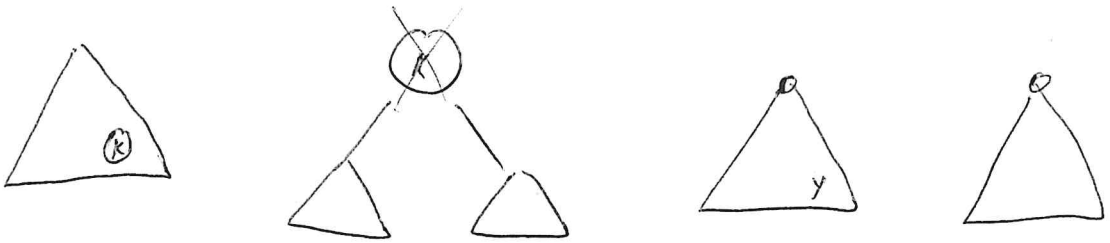
- OR L L

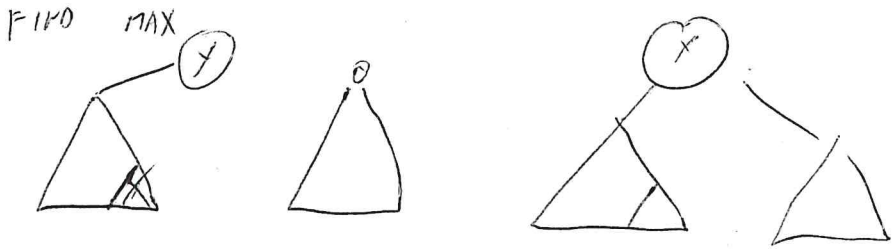


- ZIG - ZAG



- DELETE





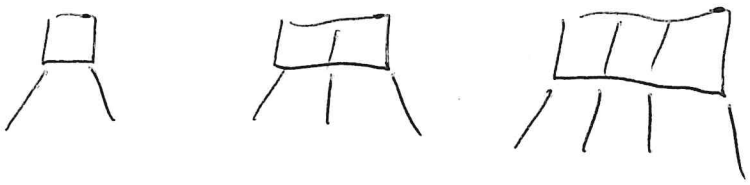
- RUN OF SPLAY TREE IS COMPARABLE TO ANY BALANCED TREE IF N OPERATIONS ARE PERFORMED

- ADVANTAGES
  - AMORTIZED RUNE TIMES ARE SIMILAR TO AVL OR R-B TREES
  - EASIER IMPLEMENTATION
  - NO ADDITIONAL INFORMATIONS (LIKE DEPTHS IN AVL)

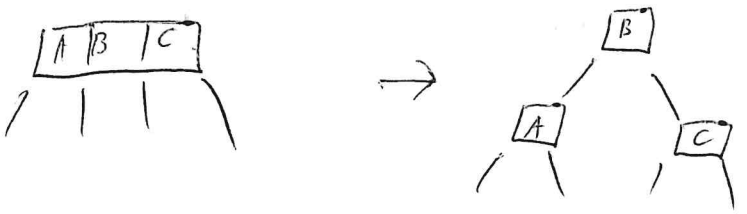
- DISADVANTAGE
  - TREE CHANGES WITH READ ONLY OPERATIONS

- 2 -3- 4 TREE

- LIKE B# -TREE WITH MIN DEGREE 2 AND MAX 4
- NODES ARE
  - 2, 3, 4 (NUMBER OF CHILDREN)



- FIND WORKS AS IN B -TREE
- INSERT ——— // ———
- DELETE ——— // ———
- SPLITING STRATEGY
  - IF WE REACH 4-NODE
    - SPLIT INTO 2 2-NODES
    - MOVE MIDDLE ELEMENT TO PARENT



- 

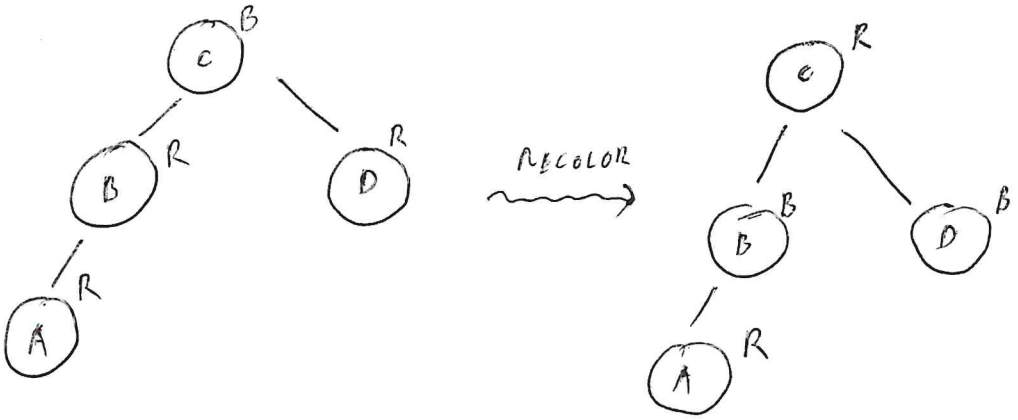
- COMPARISION OF TREES
  - IF RANDOM ORDER INPUT
    - BEST IS UNBALANCED BST
  - IF RANDOM ORDER INPUT AND SOMETIMES SORTED
    - RED-BLACK TREE
  - IF SORTED ORDER AND THAN RANDOM ACCESS
    - AVL
  - IF SORTED ORDER AND THAN CLUSTERED OR SEQUENTIAL ACCESS
    - SPLAY TREE

- RED-BLACK TREE
  - APPROXIMATELY BALANCED BST
    - HEIGHT $\leq$ 2x HEIGHT OF BALANCED TREE
  - STORES BIT TO MARK COLOR OF NODE
  - PROPERTIES
    - EACH NODE IS RED OR BLACK
    - EACH NIL (LEAF) IS BLACK
    - IF NODE IS RED, BOTH CHILDREN IS BLACK
    - EVERY SIMPLE PATH FROM NODE TO DESCENDANT LEAF CONTAINS THE SAME NUMBER OF BLACK NODES
    - ROOT IS BLACK
  - SEARCH IS THE SAME AS IN BST

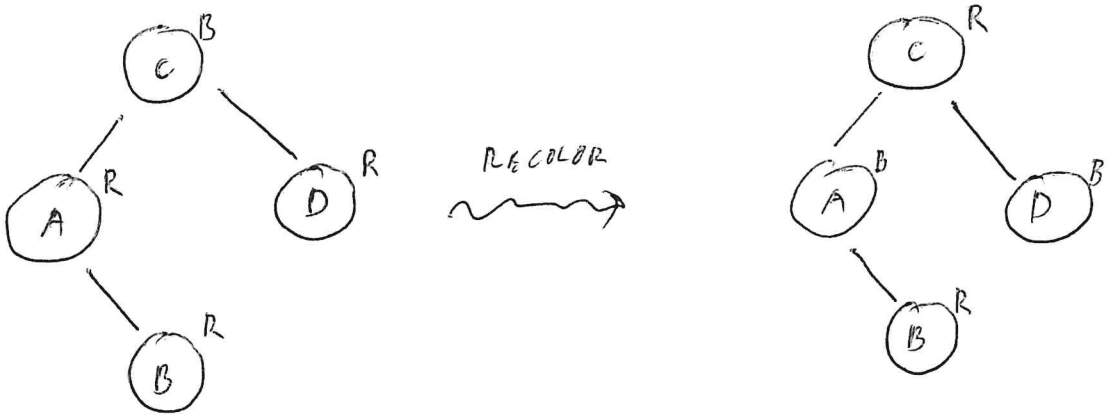
INSERT

- NEW NODE IS RED
- ADD IT AS TO STANDART BST
- IF PARENT OF NEW NODE IS BLACK
  - STOP
- IF PARENT OF NEW NODE IS RED

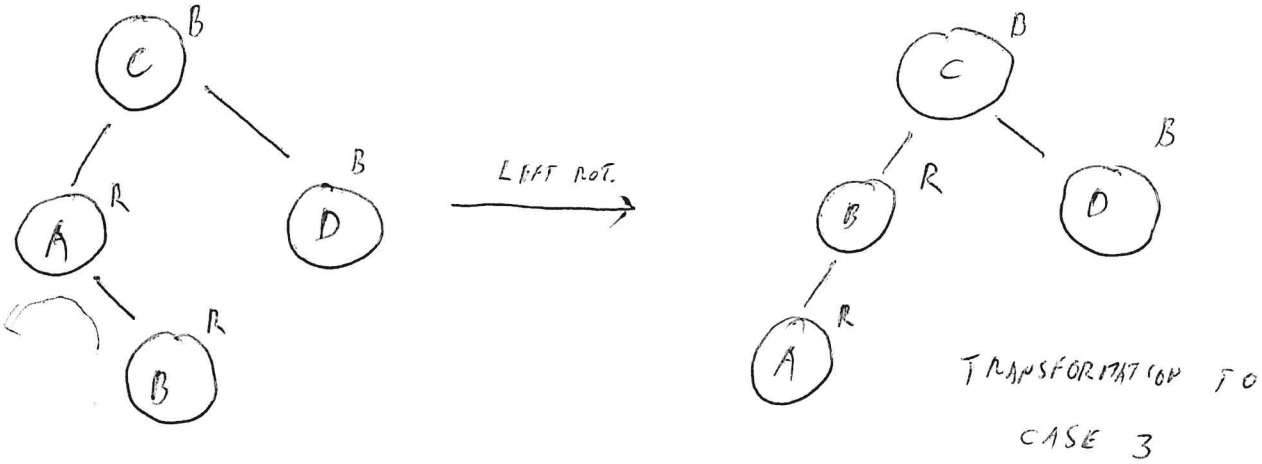
CASE 1a



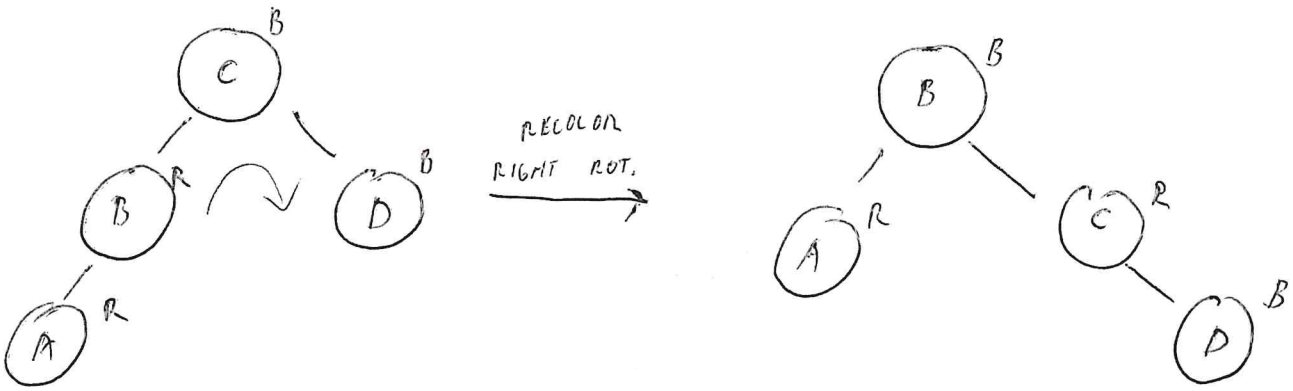
CASE 1b



CASE 2



CASE 3



- INSERTION IN $\Theta(\log(n))$ TIME
- REQUIRES MAX. TWO ROTATIONS

- DELETE

  - DELETE AS IN CASE OF BST

  - IF NODE IS RED

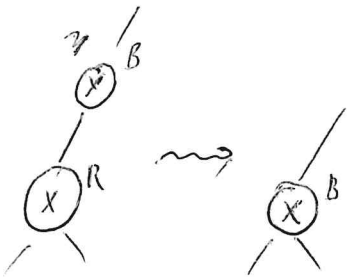
    - ALL OK

  - IF BLACK NODE WAS DELETED

    - CHILD OF DELETED NODE IS ~~BLACK~~ RED

      - RECOLOR IT TO RED

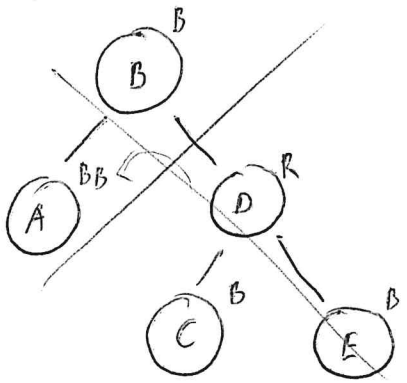
      - STOP



    - CHILD OF DELETED NODE IS BLACK

      - MARK CHILD AS DOUBLE BLACK

      - CASE ~~IN~~ A FIVE CASES



- COMPLEXITY IS $\Theta(\log(n))$

- AFTER THREE ROTATIONS ARE DONE

- KD-TREES

  - SEARCH TREE IN D DIMENSIONS

  - SPACE DIVIDED INTO RECTANGULAR CELLS

  - FIND IS ANALOGOUS TO 1D VARIANT

    - CUTTING DIMENSIONS ALTER

  - INSERT IS ANALOGOUS TO 1D VARIANT

  - FIND MIN OF x-TH DIMENSION

    - NO EASY WAY HOW TO KEEP TRACK OF IT

    - MOST COSTLY OPERATION

      - COMPLEXITY $O(n^{1-\frac{1}{d}})$ n - NODES, d - DIMENSIONS

    - MIN OF y COORDINATE CAN BE IN BOTH HALVES OF CUT BY x COORDINATE

      - BOTH MUST BE SEARCHED

- DELETE

  - ONLY LEAVES ARE PHYSICALLY DELETED

  - IF REMOVE INNER NODE CUTTING BY d DIMENSION, FIND NODE WITH SMALLEST VALUE IN d DIMENSION IN RIGHT SUBTREE AND REPLACE DELETED NODE BY IT

    - RUN DELETE ON NODE WHICH IS SUBSTITUENT

- NEAREST - NEIGHBOR SEARCH
  - RUNS RECURSIVELY IN BOTH L AND RIGH SUBTREES OF ACTUAL NODE
  - REGISTER AND UPDATE PARTIAL RESULTS
  - PERFORMS PRUNING DURING SEARCH
  - COMPLEXITY CAN BE $O(n)$ WHEN DATA IS UNFAVORABLY ARRANGED
    - HIGH NUMBER OF DIMENSIONS (7, 8... 10000)
    - ARRANGMENT OF POINTS IN LOW DIMENSION IS SPECIAL (ARTIFICALLY CONSTRUCTED)
  - EXPECTED TIME IS $O(2^D + \log n)$ WITH UNIFORMLY DISTRIBUTED DATA
  - IT IS EFFECTIVE ONLY WHEN 2^D IS SIGNIFICANTLY SMALLER THAN n

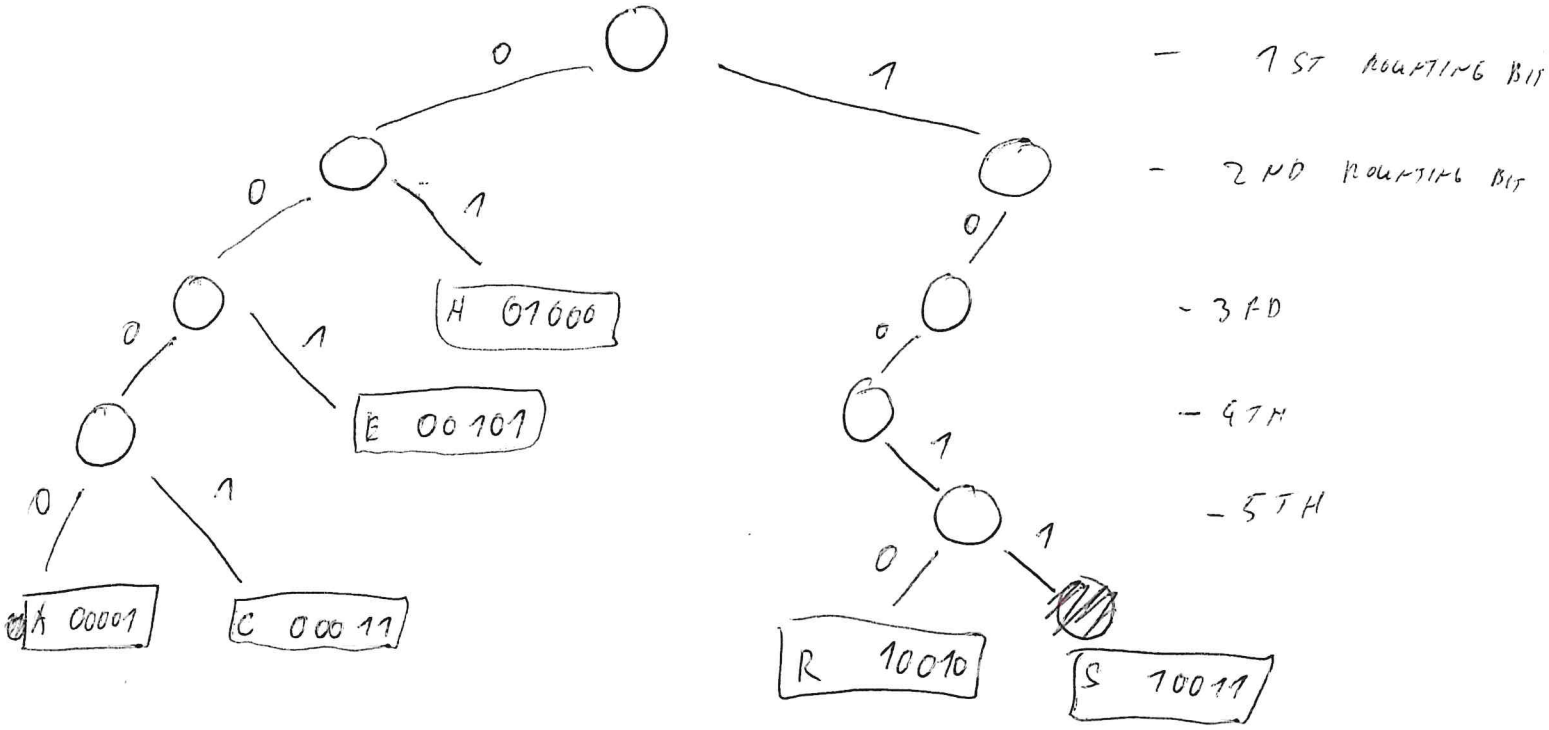
- BINARY TRIE

  - KEYS ARE SEQUENCES OF BITS

  - KEYS IN ROOTS ONLY, INNER NODES ARE ROUTERS ONLY

  - KEY MUST NOT BE PREFIX OF ANOTHER KEY

    - THIS CAN BE MADE TRUE BY REPRESENTATION HAVING THE SAME LENGTH



- 1ST ROUTING BIT
- 2ND ROUTING BIT
- 3RD
- 4TH
- 5TH

- INSERT CAN MAKE SEVERAL NEW NODES
  - INSERT I 01001