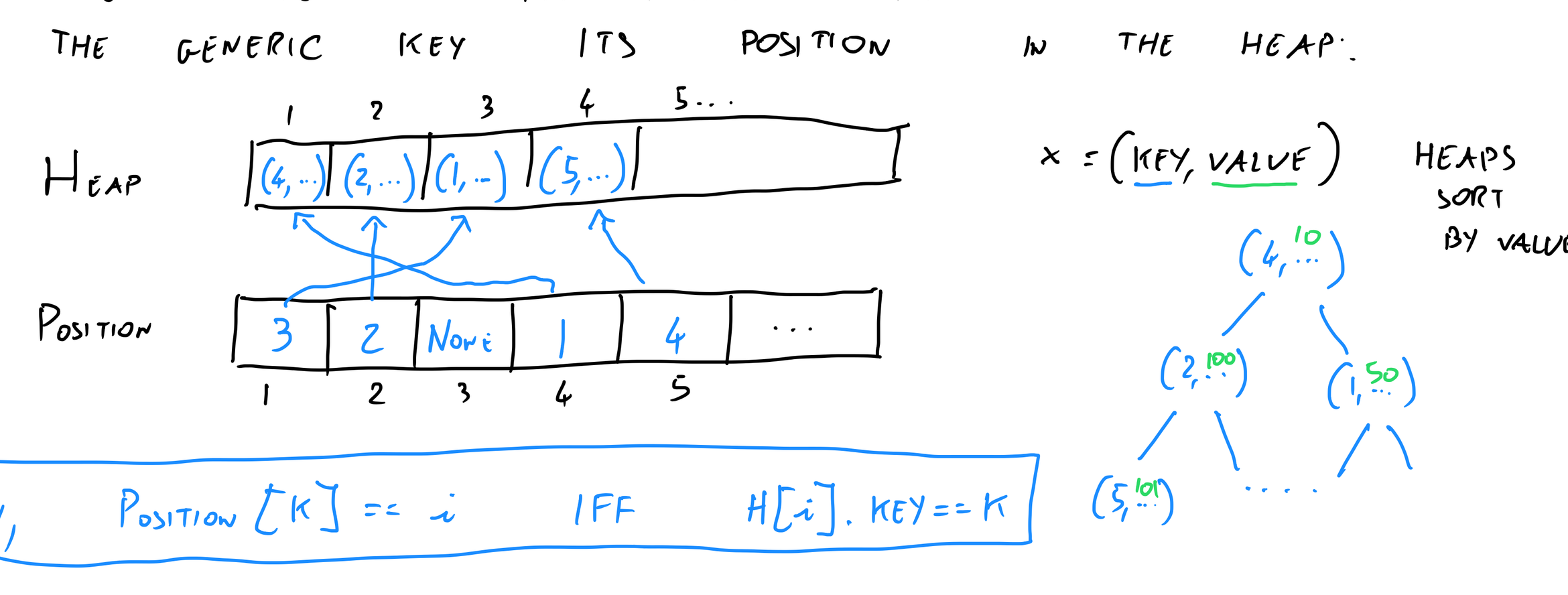
HEAPS

- ADD(H, x) ADDS ITEM x TO THE HEAP IN $O(\lg n)$ TIME;
- REMOVE(H, i) REMOVE THE ITEM IN POSITION i, " $O(\lg n)$ ";
- FINDMIN(H) RETURNS THE ITEM OF MIN-VALUE OF H, IN $O(1)$ TIME.

IN HEAPS, ITEMS HAVE "JUST" A VALUE (x.VALUE).
SOMETIMES, WE NEED ITEMS TO HAVE TWO FEATURES: A VALUE AND A KEY. (KEYS CANNOT BE SHARED BY MULTIPLE ITEMS).
(IN MOST TEXTBOOKS VALUES/KEYS ARE SWAPPED; WE FOLLOW, INSTEAD, THE "PYTHON" PERSPECTIVE).

SUPPOSE THAT OUR KEYS ARE FROM THE SET $\{1, 2, \dots, N\}$.
WE ARE GOING TO USE AN ARRAY POSITION THAT ASSIGNS TO THE GENERIC KEY ITS POSITION IN THE HEAP.



THEN, POSITION[K] == i IFF H[i].KEY == K

FOR POSITION TO KEEP TRACK OF THE POSITIONS OF THE KEYS, WE NEED TO UPDATE IT WHEN WE MODIFY THE HEAP, THAT IS, WHEN
- WE ADD AN ITEM,
- " REMOVE " " ,
- " SWAP TWO ITEMS.

EACH OF THESE 3 OPERATIONS REQUIRE AT MOST 2 UPDATES TO POSITION — THEY CAN THEN BE IMPLEMENTED IN $O(1)$ TIME.

THE RUN TIMES OF ADD, REMOVE (AND FINDMIN) REMAIN AS THEY WERE.

THE RESULTING DATA STRUCTURE IS SOMETIMES CALLED A PRIORITY QUEUE. IT GIVES US THE FOLLOWING OPERATIONS:

$x = (\text{KEY}, \text{VALUE})$

- ADD(H, x) TAKES $O(\lg n)$ TIME;
- FINDMIN(H) " $O(1)$ " ;
- REMOVE(H, i) " $O(\lg n)$ " ;
- EXTRACTMIN(H) (RETURN AND REMOVE THE ITEM OF MIN-VALUE) TAKES $O(\lg n)$ TIME:

    x = FINDMIN(H)
    REMOVE(H, 1)
    RETURN x

- REMOVEK(H, KEY) (REMOVES THE ITEM OF KEY KEY, IF IT EXISTS) TAKES $O(\lg n)$ TIME

    IF POSITION[KEY] != NONE:
        REMOVE(H, POSITION[KEY])

- UPDATEVALUE(H, KEY, NEWVALUE) (UPDATES THE VALUE ASSOCIATED TO A KEY) TAKES $O(\lg n)$ TIME

    REMOVEK(H, KEY)
    ADD(H, (KEY, NEWVALUE))

DIJKSTRA'S ALGORITHM $(G(V,E), \ell, s)$:

// $V = \{1, 2, \dots, N\}$

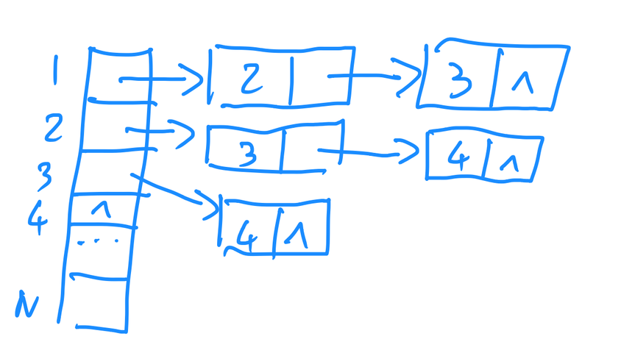
(A)
- $O(N)$ — INITIALIZE A PRIORITY QUEUE H, POSITION OF SIZE N
- $O(\lg N)$ — ADD(H, (s, 0)) // KEY == s, VALUE == 0
- $O(N)$ — LET d BE AN ARRAY S.T. $d[i] = $ NONE $\forall i \in \{1, \dots, N\}$

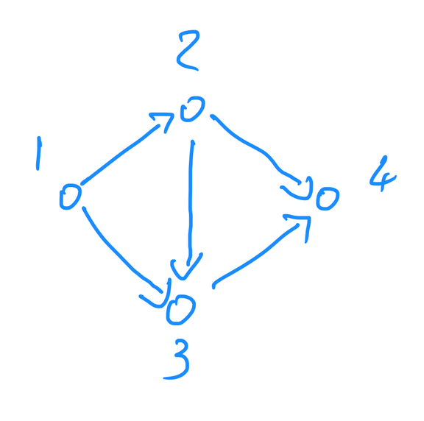
- FOR $i = 1, 2, \dots, N$

  (B)
  - NEXT = EXTRACTMIN(H) $O(\lg N)$
  - v = NEXT.KEY
  - $d[v]$ = NEXT.VALUE

  (C) $O(\deg^+(v) \cdot \lg N)$ (IF THE GRAPH IS REPRESENTED AS AN ADJ. LIST)
  - FOR EACH OUT-NEIGHBOR w OF v:
    - IF $d[w]$ == NONE:
      // WE HAVEN'T YET FOUND A SHORTEST PATH TO w
      dist = $d[v] + \ell(v, w)$
      IF POSITION[w] == NONE:
        // WE HADN'T ENCOUNTERED w YET
        ADD(H, (w, dist)) $O(\lg N)$
      ELIF H[POSITION[w]].VALUE > dist:
        UPDATEVALUE(H, w, dist) $O(\lg N)$
- RETURN d

ADJACENCY LIST.





$\deg^+(v)$ IS THE NUMBER OF OUTNEIGHBORS OF v.

$\deg^+(1) = 2$ $\deg^+(3) = 1$
$\deg^+(2) = 2$ $\deg^+(4) = 0$

ADJACENCY MATRIX

	1	2	3	4
1		1	1	0
2	0		1	1
3	0	0		1
4	0	0	0	

THE TOTAL RUNTIME IS THEN

$$\underbrace{O(N)}_{(A)} + \underbrace{O(N \lg N)}_{(B)} + \underbrace{O\Big((\lg N) \cdot \sum_{v \in V} \deg^+(v)\Big)}_{(C)} =$$

$$O(N \lg N) + O\Big((\lg N) \cdot \sum_{v \in V} \deg^+(v)\Big) \qquad \big(\le O(N^2 \lg N)\big)$$

L: IF $G(V, E)$ IS A DIRECTED GRAPH, THEN $\sum_{v \in V} \deg^+(v) = |E| \triangleq M$.

P: RECALL $E \subseteq \{(w, v) \mid w, v \in V \text{ AND } w \neq v\}$.
$\forall w \in V$, LET $E_w \subseteq E$ BE THE SET OF EDGES/ARCS LEAVING w:

$$E_w = \{(w, v) \mid (w, v) \in E\}.$$

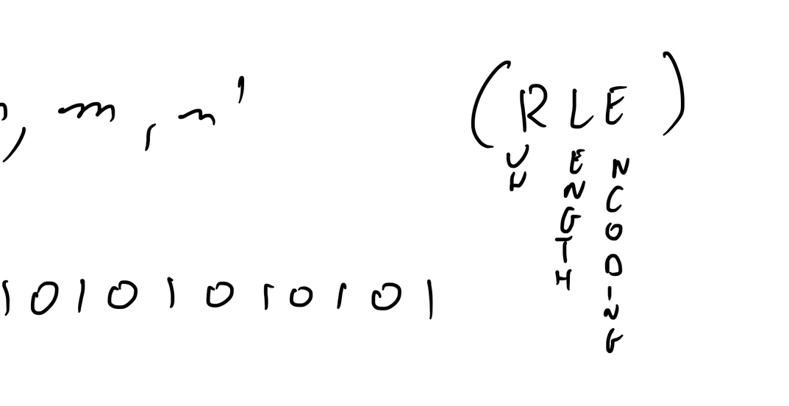
THEN, $|E_w| = \deg^+(w)$.
ALSO, FOR EACH $w, v \in V$ S.T. $w \neq v$, IT HOLDS $E_w \cap E_v = \emptyset$.
MOREOVER $E = \bigcup_{v \in V} E_v$. THUS,

$$M \triangleq |E| = \Big|\bigcup_{v \in V} E_v\Big| = \sum_{v \in V} |E_v| = \sum_{v \in V} \deg^+(v). \quad \square$$

THE RUNTIME IS THEN

$$O(N \lg N) + O\Big((\lg N) \cdot \sum_{v \in V} \deg^+(v)\Big) =$$
$$O(N \lg N) + O(M \lg N) =$$
$$O((N + M) \lg N).$$

Q: LET V BE AN ARRAY OF N ELEMENTS; V[i] IS THE SCORE THAT STUDENT i GOT IN AN EXAM.
SUPPOSE THAT THERE ARE 3 POSSIBLE SCORES ("INSUFFICIENT": 0, "PASS": 1, "HONORS": 2)
SORT V AS FAST AS POSSIBLE.



DATA COMPRESSION

$\overbrace{00000000 \cdots 0}^{n}\overbrace{1111 \cdots 111}^{m}\overbrace{00 \cdots 0}^{n'}$

$0, n, m, n'$ (RLE) RUN LENGTH ENCODING

010101010101

"01" * n (DICTIONARY-BASED COMPRESSION)

HUMAN TEXT COMPRESSION

WE ARE GIVEN A TEXT: A SEQUENCE OF CHARS 'A', 'B', ..., 'Z', ' ', ';'.

BY USING ASCII, I CAN REPRESENT A n-CHARS STRING WITH $8n$ BITS.

GIVEN THAT I ONLY HAVE 28 POSSIBLE CHARS, I CAN USE JUST 5 BITS PER CHAR

00000	'A'
00001	'B'
00010	'C'
⋮	
11001	'Z'
11010	' '
11011	';'

THE TOTAL # OF BITS SHRINKS FROM $8n$ TO $5n$ (37.5% IMPROVEMENT!)