NOTE BOOK

Rokshana Ahmed

Algorithms

DATE: 31/03/2022

v = (key, value)

PRIORITY QUEUES / HEAPS

- ADD (H, v) takes O(logn) time;
- FINDMIN (H) = O(1) // ;

  // returns one item of the heap with smallest key

- REMOVE (H, i) takes O(logn) time;

  // removes item in position i

- EXTRACTMIN (H) takes O(logn) time;

  // removes, and returns, an item with smallest key (that is, the item in position i)

```
v = FINDMIN(H)
REMOVE(H, 0)
RETURN v
```

- REMOVE V(H, value) takes O(logn) time;

  // removes the item in the heap having value equal to "value" (THIS WORKS ONLY IF ALL VALUES IN THE HEAP ARE DISTINCT)

- UPDATEKEY (H, value, newkey) takes O(logn) time;

  // changes the key of the item in the heap having value "value" to "newkey"

  (THIS WORKS ONLY IF ALL THE VALUES IN THE HEAP ARE DISTINCT)

```
(key, value) = H[POSITION[value]]
REMOVE V(H, value)
ADD (H, (newkey, value))
```

With HEAPSORT we can sort an array of n elements in $O(n \log n)$ time.

DIJKSTRA'S ALGORITHM $(G(V,E), w, s)$

// $V = \{0, 1, 2, \ldots, N-1\}$

① $O(N)$

$O(N)$ INITIALIZE A HEAP OF SIZE N

$O(\log N)$ H.ADD((0, s)) // (KEY, VALUE) WITH KEY = 0 AND VALUE = s

$O(N)$ d = [NONE] * N

$O(1)$ d[s] = 0

// IN THE END, d[i] is going to contain the length
// of a shortest path from s to i

FOR EACH OUT-NEIGHBOR u of s:

ADD (H, (H(1, u), 1))

FOR i = 1 TO N-1 (we start from 2 because we already started from 1 node (s))

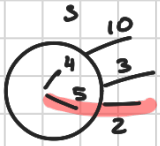
② $O(N \log N)$

$O(\log N)$ NEXT = EXTRACTMIN(H)

$O(1)$ v = NEXT.value

$O(1)$ d[v] = NEXT.key

THESE ARE RUN FOR N-1 TIMES



③ $O(\deg(v) \log N)$

FOR EACH OUT-NEIGHBOR u OF v:

IF d[u] = NONE: $O(1)$

// we haven't yet selected/visited "u"

distance

dist = d[v] + w(v, u) $O(1)$

IF H[POSITION[u]].KEY > dist: $O(1)$

UPDATEKEY(H, u, dist) $O(\log N)$

THIS IS RUN FOR degree of v (deg(v)) times

v deg(v) = -3

RETURN d

deg - n° nodes/neighbours

(1) (2)

TOTAL RUNTIME $= O(N) + O(N \log N) + \sum_{v \in V} O(\deg(v) \log N) =$

$= O(N \log N) + \left(\sum_{v \in V} \deg(v)\right) \cdot O(\log N) =$

$= O(N \log N) + 2M \cdot O(\log N)$

$= O((N+M) \log N)$

L: IF $G(V,E)$ is a graph, then $\sum_{v \in V} \deg(V) = 2|E|$

P: Recall that $E \subseteq \{\{(u,v)\} \mid u, v \in V, \text{ and } u \neq v\}$

Moreover, the set of neighbours $N(v)$ of "v" is equal to $N(v) = \{u \mid \{u,v\} \in E\}$.

And $\deg(v) = |N(v)|$

Let us also define the set of edges.

Incident on v, $I(v) = \{\{u,v\} \mid v \in \{u,v\} \text{ AND } \{w,v\} \neq E\}$

12, 15
21, 24
35
42, 45
51, 53, 54

$N(2) = \{1,4\}$
$I(2) = \{\{1,2\}, \{2,4\}\}$

Then $\deg(v) = |I(v)|$

$$\sum_{v \in V} \deg(v) = \sum_{v \in V} |I(v)| = \sum_{e \in E} 2 = 2|E| \quad \blacksquare$$

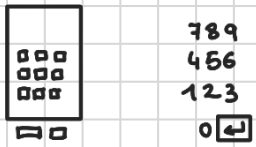
QUESTION:

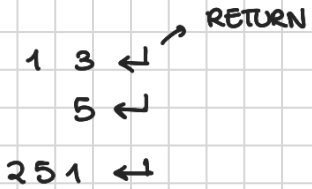
Suppose that V is an array of N elements V[i] is the score that student i got in the algorithm class (scores are "PASS": 1, "INSUFFICIENT": 0, "HONORS": 2)

Sort V increasingly as fast as you can.

Try greedy approach when they ask about interval scheduling

SALOON





Let's say there is a gunfire and you lose your numbers (keys). how can you keep inserting your entries?

If you have 3 keys, then you can write integers in binary (0,1 FOR AN INTEGERY, ↵ TO SPLIT INTEG.)

If you have only 2 keys, I could write in unary.

3 AND 5

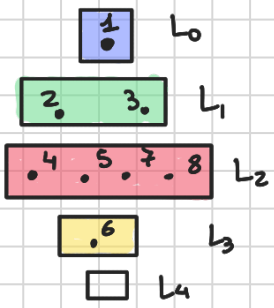
11101111110

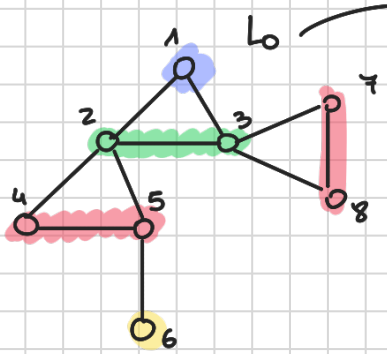
What if you only have the key "zero"?

(Personally, I think we should close down the saloon)

BFS (BREADTH FIRST SEARCH)

DFS (DEPTH = =)





Layer 0
Layer 1 (Green) reachable from L_0
Layer 2 (Red) = from L_1
Layer 3 (Yellow) = = L_2

BFS (G(V,E), s)

$S \leftarrow \{s\}$ // S is the set of visited nodes

$L_0 \leftarrow \{s\}$ // L_i will contain all the nodes at DISTANCE

$i \leftarrow 0$ // i from s

WHILE TRUE:

$L_{i+1} \leftarrow \emptyset$

WHILE $\exists v \in L_i$ AND $w \in V - S$ S.T. $\{v, w\} \in E$:

$L_{i+1} \leftarrow L_{i+1} \cup \{w\}$

$S \leftarrow S \cup \{w\}$ $O(n+m)$

IF $L_{i+1} = \emptyset$:

BREAK

ELSE:

$i \leftarrow i+1$

RETURN $S, (L_0, L_1, \ldots, L_i, \ldots)$ GREEN BECAUSE ITS OPTIONAL IF YOU WANT TO RETURN THIS OR NOT

NOTE: Works better than DIJSKSTRA but works only with unweighted graphs

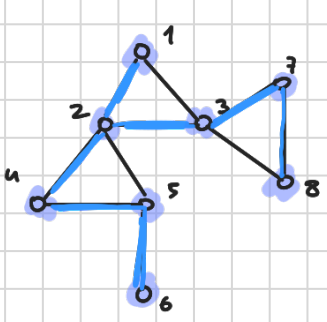
DFS ($G(V,E), v$):

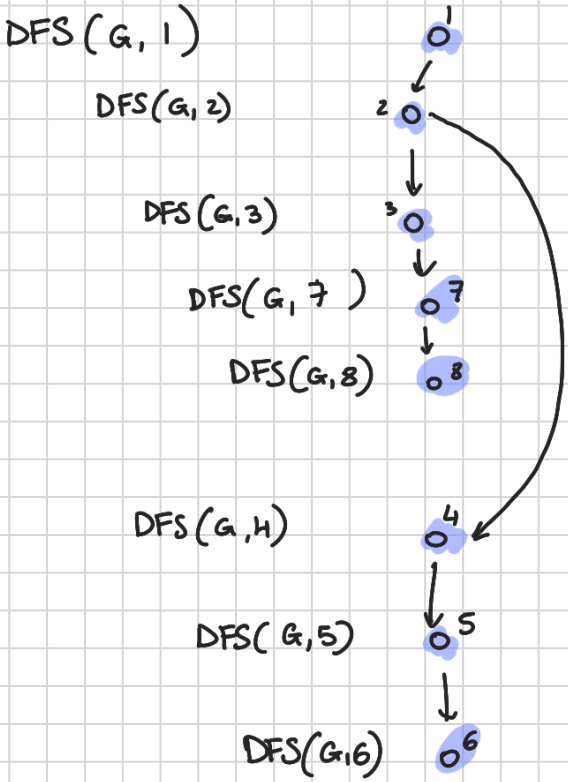
"MARK" v as EXPLORED

FOR EACH NEIGHBOUR w OF v: $O(n+m)$

IF w HAS NOT BEEN MARKED AS EXPLORED (YET):

DFS ($G(V,E), w$)





← we randomly chose 3, we could have chosen 4 too.

since it doesn't have any neighbour to visit, it goes back to 7
Since 7 doesn't have any neighbour, it goes back to 3.
3 has another node unmarked (4) so it goes there.