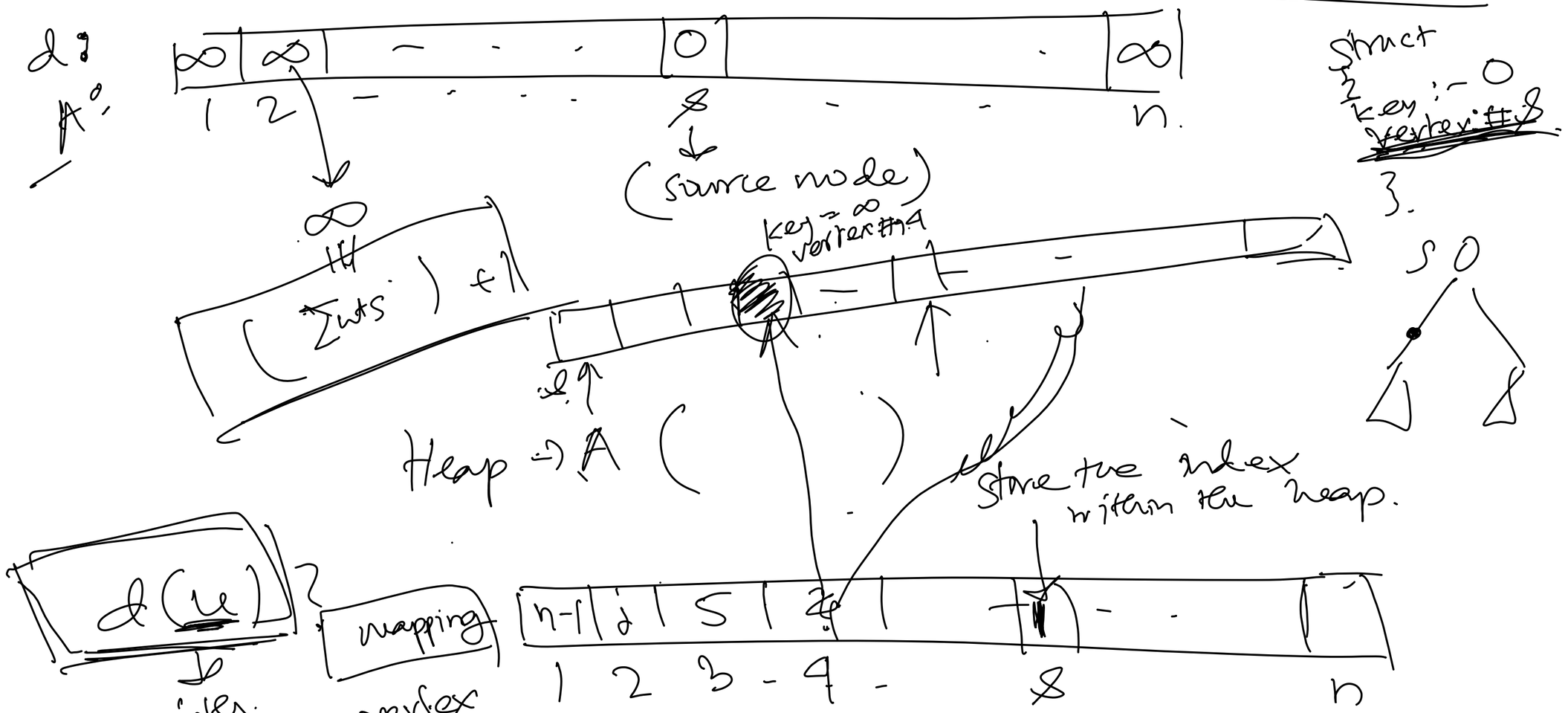
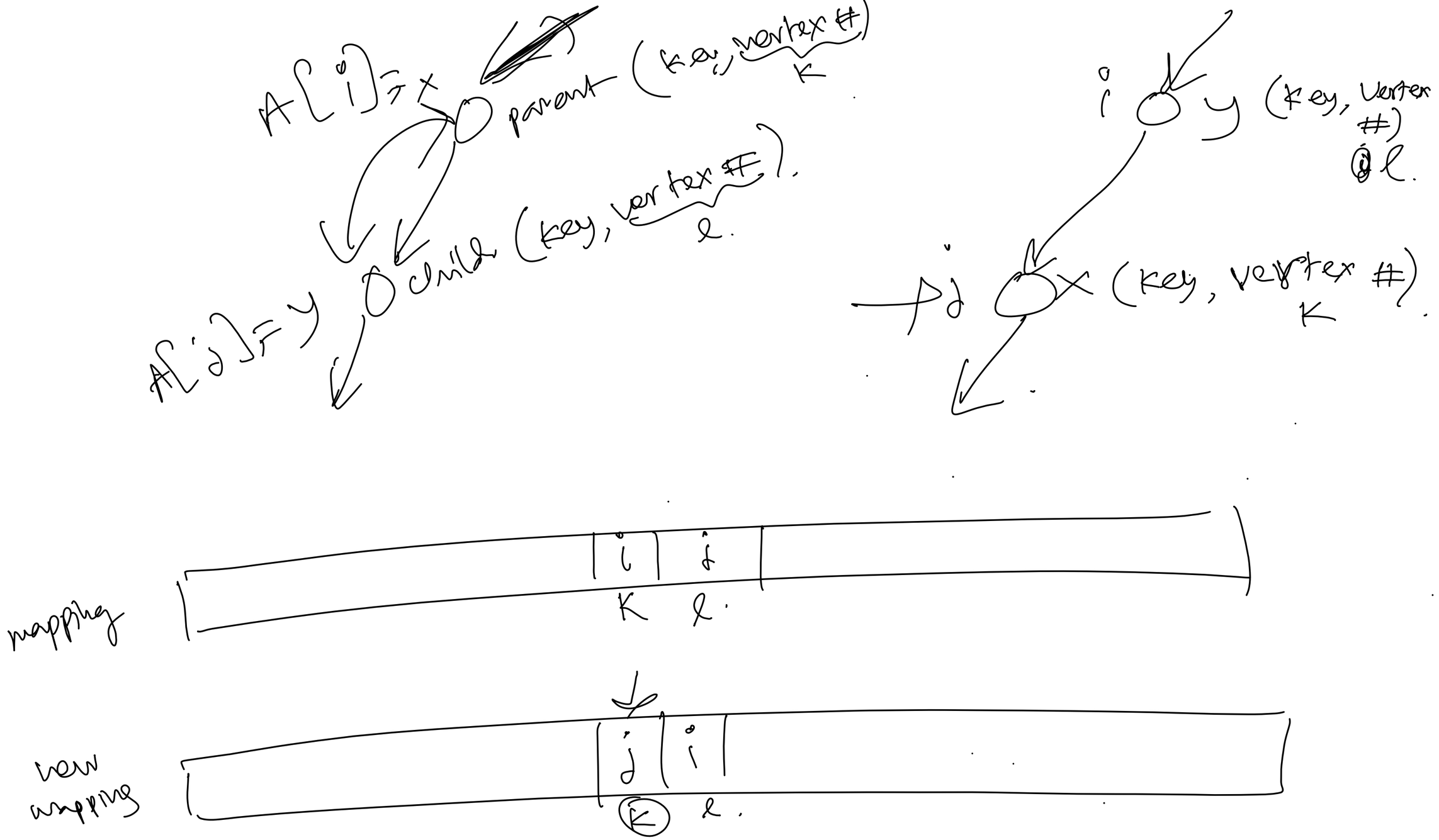
Priority Queue

Heap
mapping: vertex → heap index




A[i] = x
parent (key, vertex #)
K
A[j] = y
child (key, vertex #)
l.
i
y (key, Vertex #)
@ l.
j
x (key, vertex #)
K
mapping
i
j
K
l.
new mapping
j
i
K
l.

mapping stores the current position of vertex j within the heap A.

$A[i].key$

$A[i].vertex\# = j$

extract-min : Call the extract-min of the underlying heap.

decrease-key : Call the decrease-key of the underlying heap.

Procedure SSSP (— — —).

— —

— — .

While ($A = V$)

$\Theta(\log n$ time) → $u \in$ extract-min (A).

$n \log n$.

for each $(u,w) \in E$

if $(d(w) > d(u) + wt(u,w))$

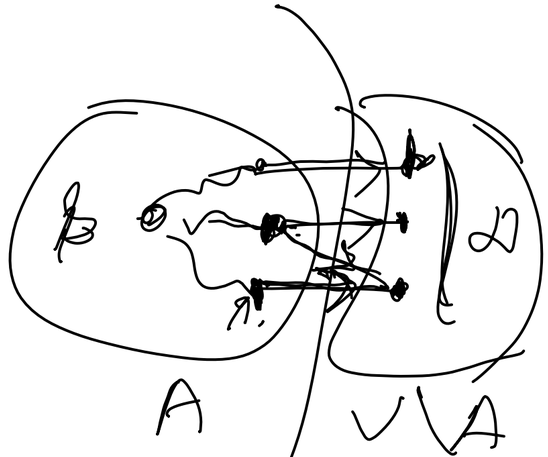
$\pi(w) \leftarrow u$.

$d(w) \leftarrow$

$\Theta(\log n)$ → decrease-key $(A, d(u) + wt(u,w))$.

$m \log n$.

$O((m+n)\log n)$



$A \times (V \setminus A)$

left endpoint

right endpoint

time complexity $O((m+n)\log n)$. $O(m+n^2)$. $m \le \binom{n}{2} = \Theta(n^2)$

$O((m+n)\log n) \ll O(n^2)$

$m \neq \Omega(n^2/\log n)$.

$O(m + n\log n) \ll O(n^2)$

↓

Fibonacci Heap.

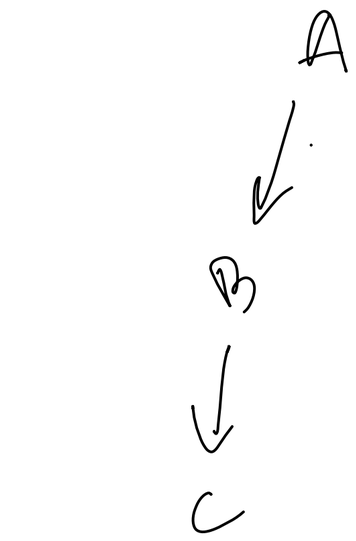
Revisit DFS

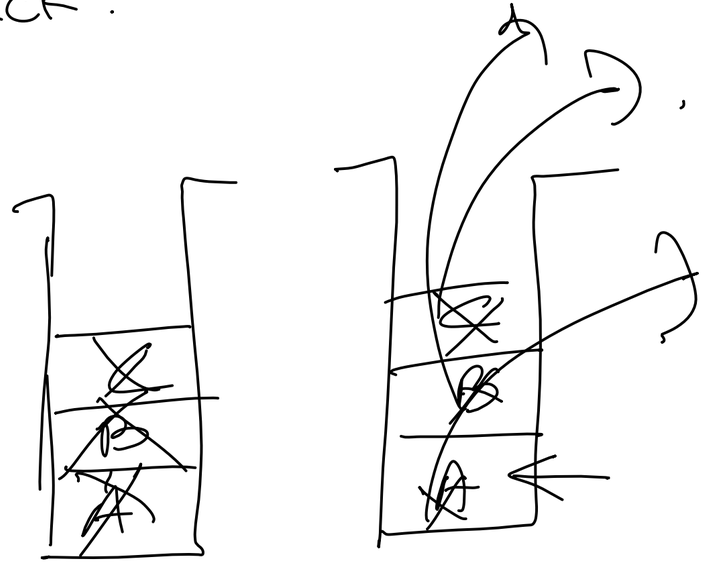
(1) maintain a clock & record the time when a vertex is visited for the first time and it is finished.

≈ pushed into the stack.

≈ popped from the stack.

(2)





children are popped before the parent is popped.

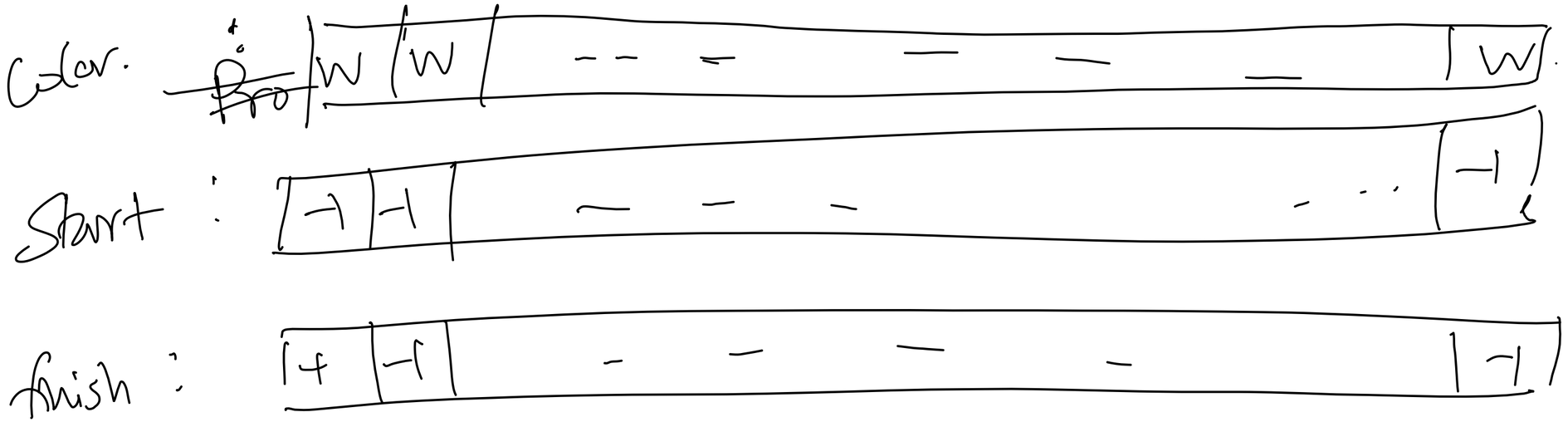
S: ~~push~~ pop : returns the top item & removes it

~~pop~~ push : put the item at the top of S.

~~peek~~ peek : returns the top item but does not remove it.

Procedure ~~peek(S)~~ peek(S)

```
u ← S.pop()
S.push(u)
return u.
```



clock: 0

Color: white, grey, black.

white → not seen

grey → if is seen & children still in stack.

black → the vertex is seen & popped.

Initially all the vertices are colored white.

Procedure DFS(G, v), color, start, finish, clock)
visited.

S ← empty stack.

S.push(v).

while(S is not empty).

u ← S.peek()

if (color[u] = white)

start[u] ← clock

clock ++.

color[u] ← grey.

for each (u, w) ∈ E

if (color[w] = white)

S.push(w).

elseif (color[u] = grey)

S.pop(u).

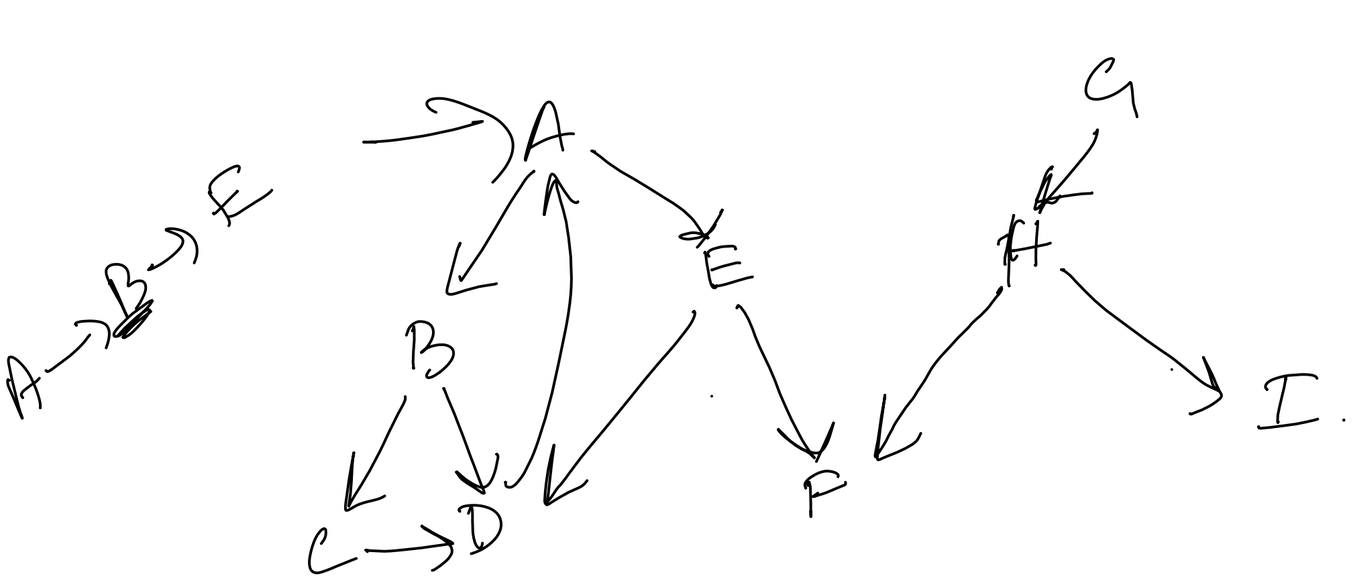
finish[u] ← clock

clock ++

color[u] ← black.

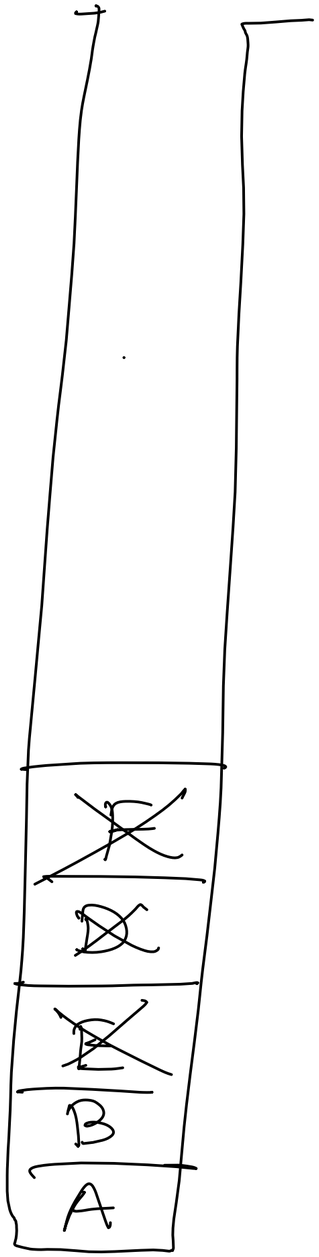
— initially all color are white.

— initially clock = 0



Clock = 1 2 3 4 5 6 7

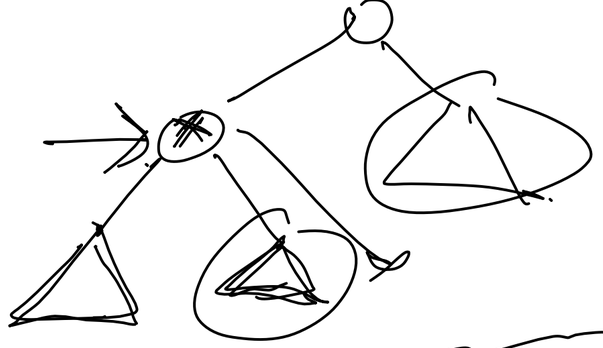
	A	B	C	D	E	F	G	H	I
Strt	0			4	1	2			
Fnsh				5	6	3			
Glbs	~~W~~ G	W	W	W ~~G~~ B.	~~W~~ ~~G~~ B.	~~W~~ ~~G~~ B	W	W	W



30.08.2024

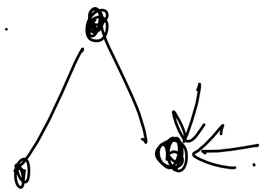
Binary min heaps :

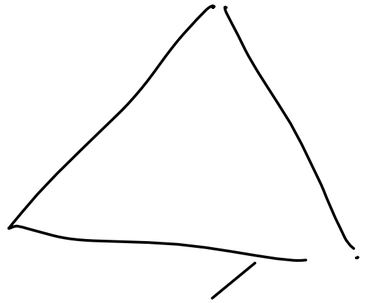
- Min-Heapify :



Build-Heap (A)

- Decrease-key :





Insert (A, K)

- put item with key = ∞
- decrease key to K.