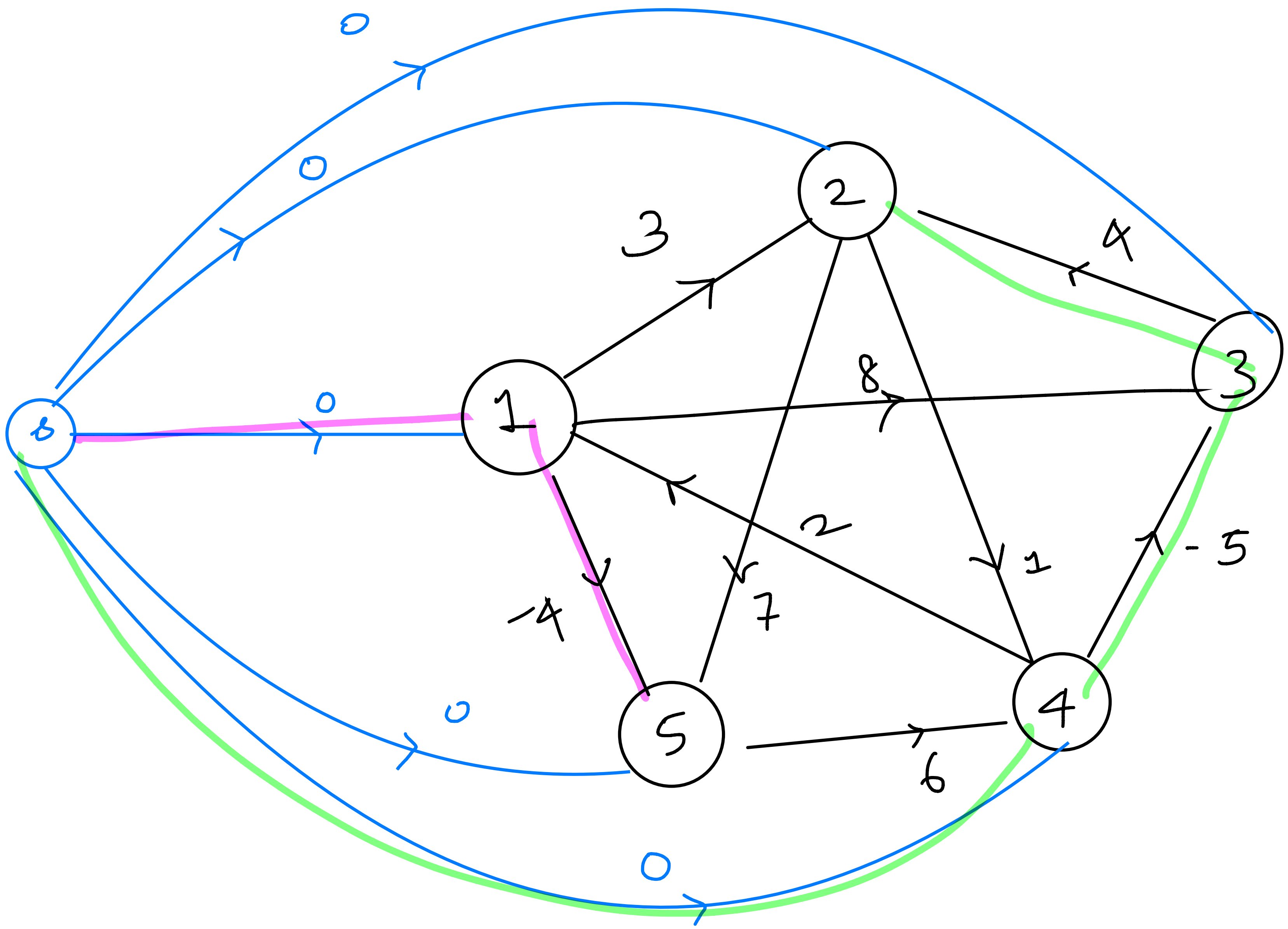
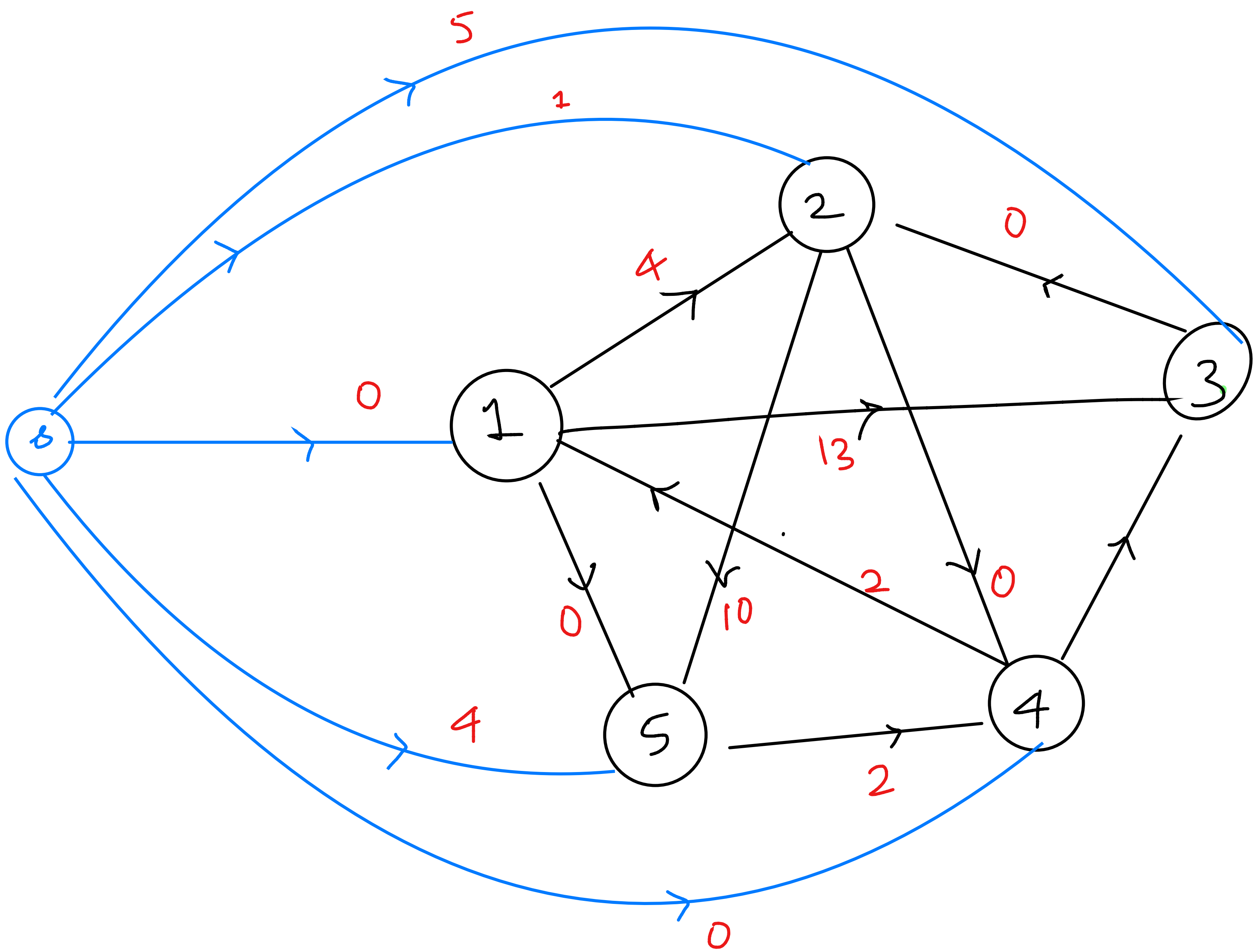


$h(v) \le 0 \Rightarrow$ better to reach from s than anywhere else in the graph.

Vertex	$h(v)$
s	0
1	0
2	-1
3	-5
4	0
5	-4



$h(v) = 0 \Rightarrow$ better to reach from s than anywhere else in the graph.

Vertex	$h(v)$
s	0
1	0
2	-1
3	-5
4	0
5	-4

Note: $\hat{w}(u,v) = 0$ cases are when u is the best way to reach v.

Johnson's Algorithm

- Construct G' by adding s to G
- Bellman Ford (G', w, s) to calculate $\delta(s, v)$ $O(VE)$
- For each $v \in V$, $h(v) \leftarrow \delta(s, v)$
- For each $(u, v) \in E$, $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$
- for each $v \in V$ (as source)

  $$\hat{\delta}(v, \cdot) = \text{Dijkstra}(G, \hat{w}, \overset{\text{source}}{\overset{\downarrow}{v}})$$

- $$\delta(i, j) = \hat{\delta}(i, j) + \underbrace{h(j) - h(i)}$$

this works for any i, j
not necessarily neighbours.

Disjoint Sets

- $S = \{ S_1, \ldots, S_k \}$, $S_i \cap S_j = \phi$, $\forall\, i \neq j$
- Each set S_x is identified by a member $x \in S_x$

New set creation

MAKE-SET(x): Create a set with x as element (one element)

UNION (x,y) : $S_z = S_x \cup S_y$

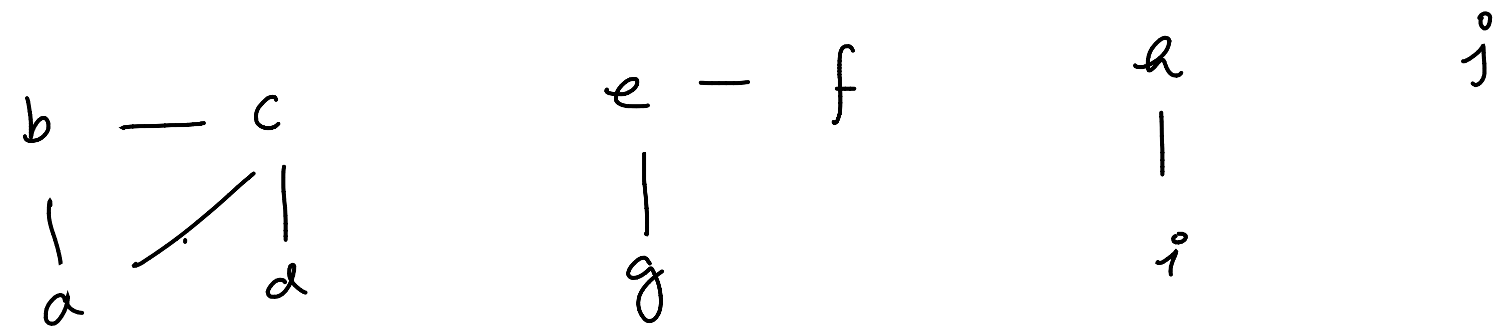
* Destroy S_x and S_y

* Nominate $z \in S_z$

belongs to either S_x or S_y

FIND-SET(x) : return pointer to the set S_x

Connected components of an undirected Graph



Set

edges	
	{a} {b} {c} {d} . . . {j}
(b, d)	{a} {b, d} {c} {e} . . . {j}
(e, g)	{a} {b, d} {c} {e, g}
(a, c)	
(h, i)	
(a, b)	
(e, f)	{a, b, c, d} {e, f, g} {h, i} {j}
(b, c)	

CONNECTED-COMPONENT (G)

for each vertex $v \in V$

do MAKE-SET (v)

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

do if FIND-SET(u) $\neq$ FIND-SET(v)

then UNION(u, v)

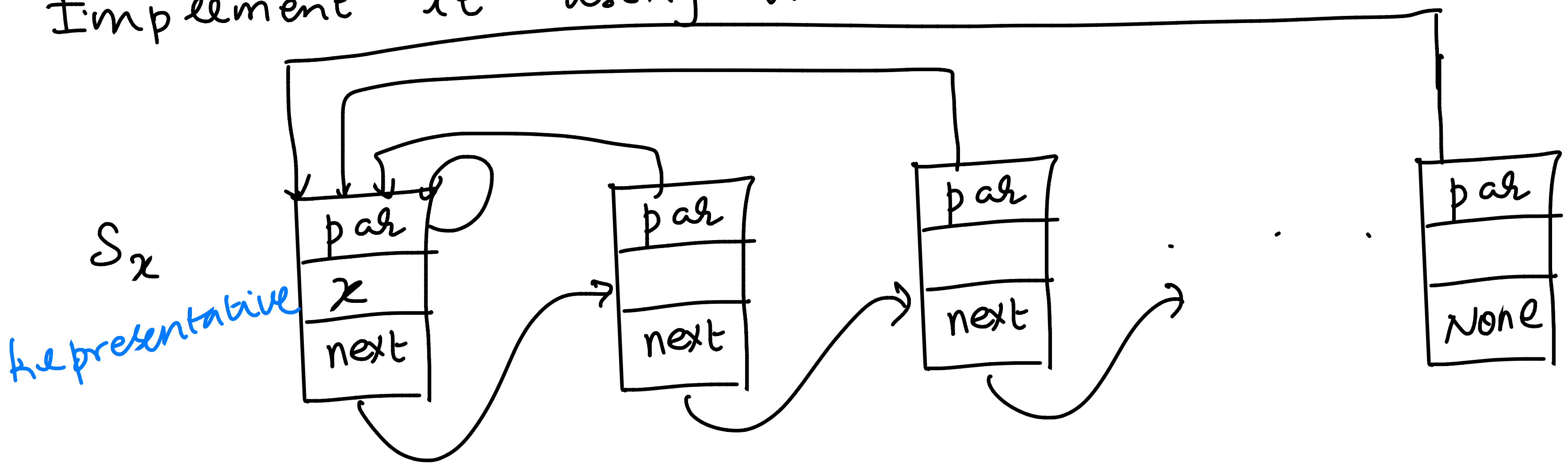
SAME-COMPONENT (u, v)

if FIND-SET (u) = FIND-SET (v)

then return TRUE

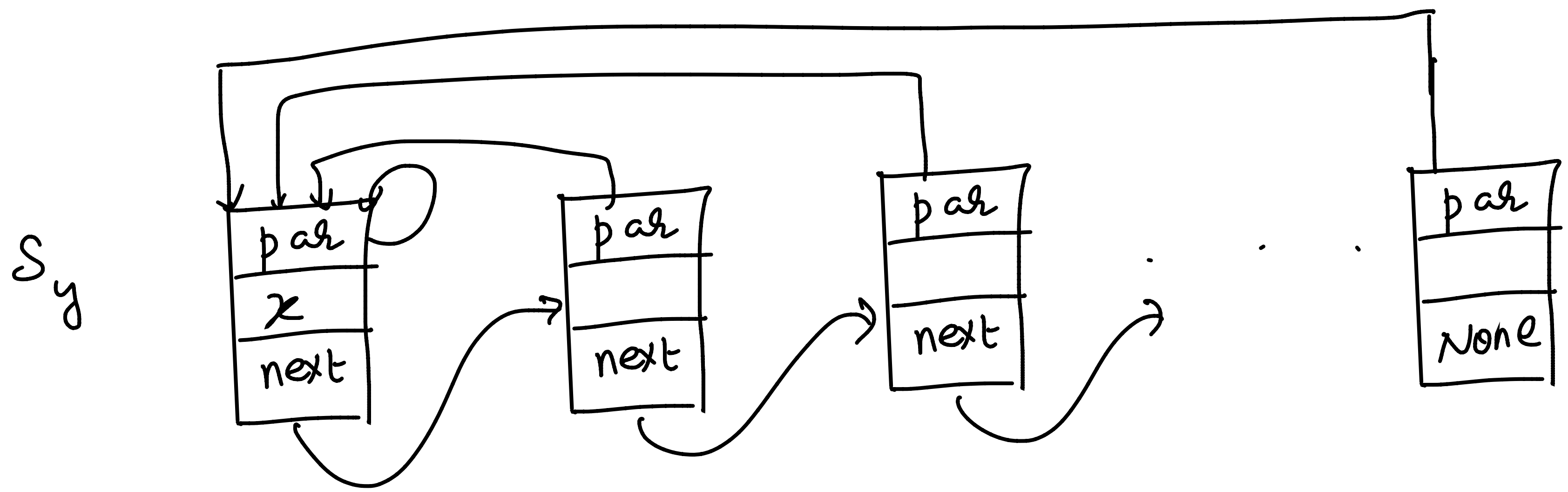
else return FALSE

Implement it using a list



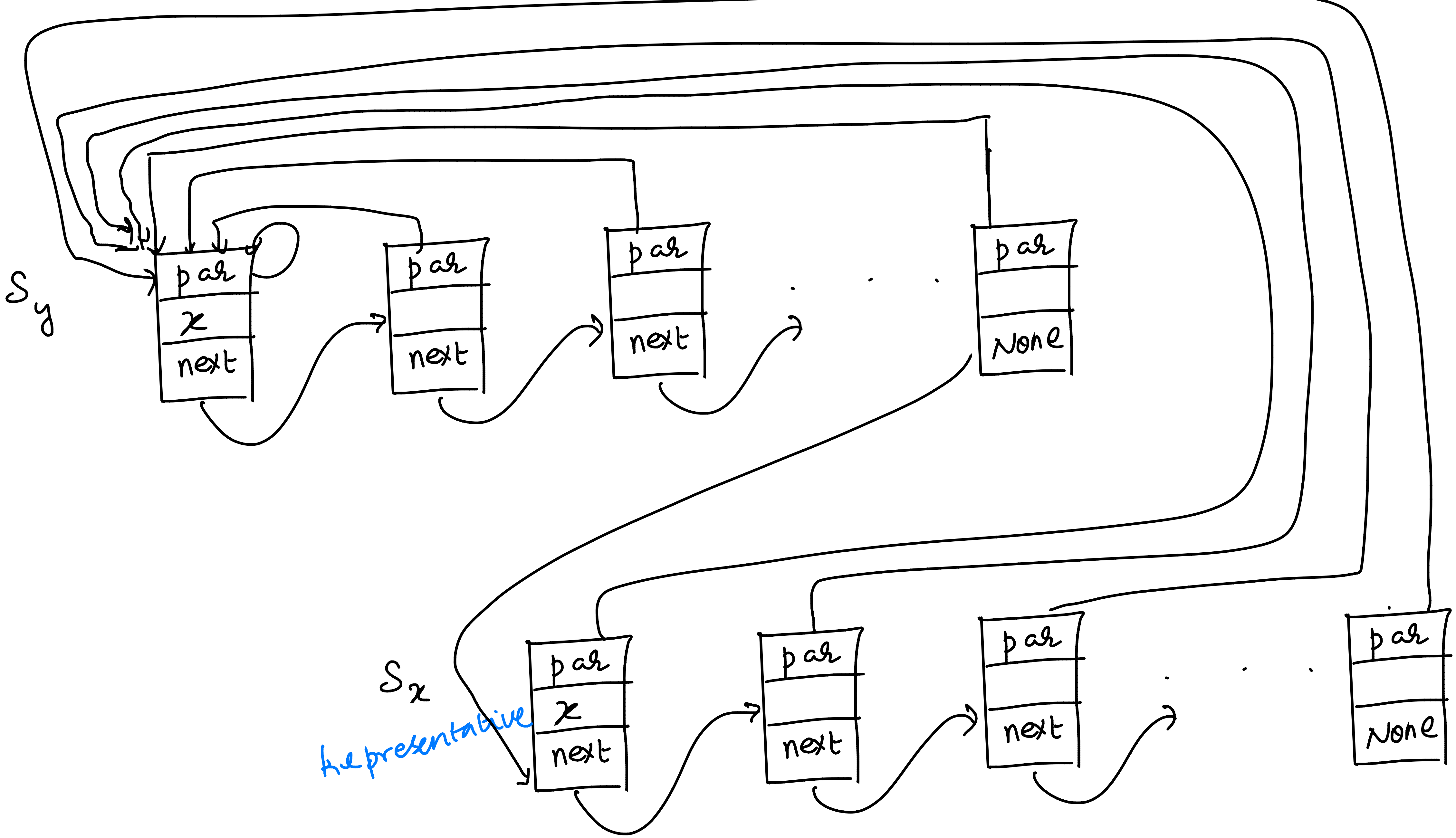
MAKE-SET(x) O(1)

FIND-SET(x)



UNION(x, y)

- Append x to end of y, and make all elements in x to point to y
- Needs as many operations as length of x



In practice, always append smaller to larger

MAKE-SET (x_1)	1	
$\vdots$	$\vdots$	$\}\ n$
MAKE-SET (x_n)	1	
UNION (x_1, x_2)	1	
UNION (x_2, x_3)	2	
$\vdots$	$\vdots$	
UNION (x_{n-1}, x_n)	$n-1$	

$(2n-1)$ operations costs $\Theta(n^2)$