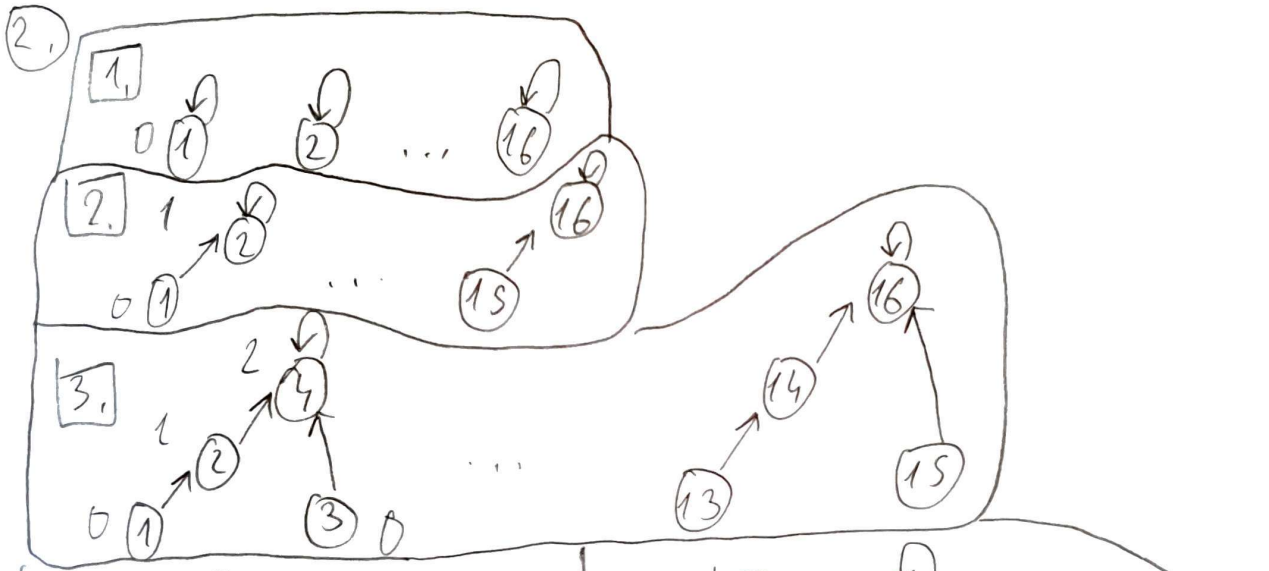
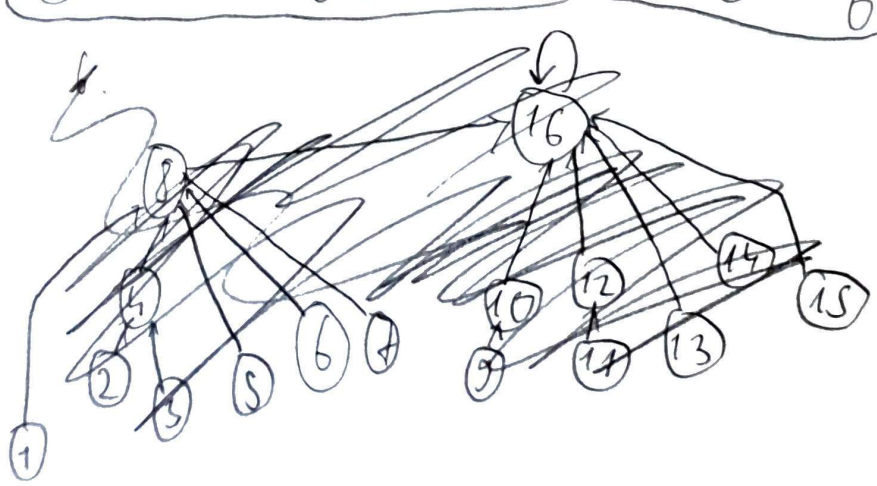
DZ 7

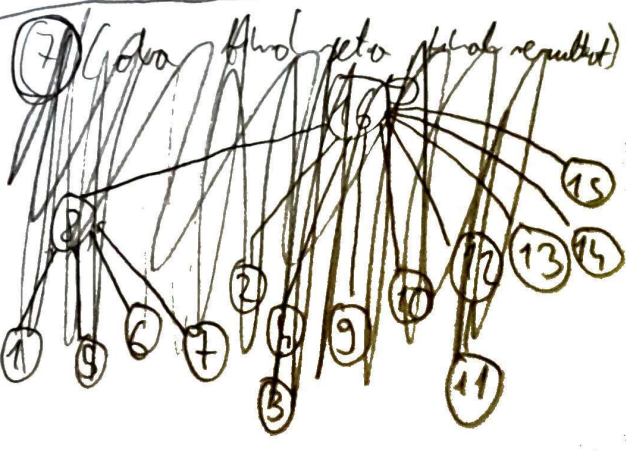
1.

```
FIND-SET(x);
A = linked list with a pointer to x
while x ≠ r[x]:
    A.insert(x)
    x = r[x]
for each node y: A
    r[y] = x
```

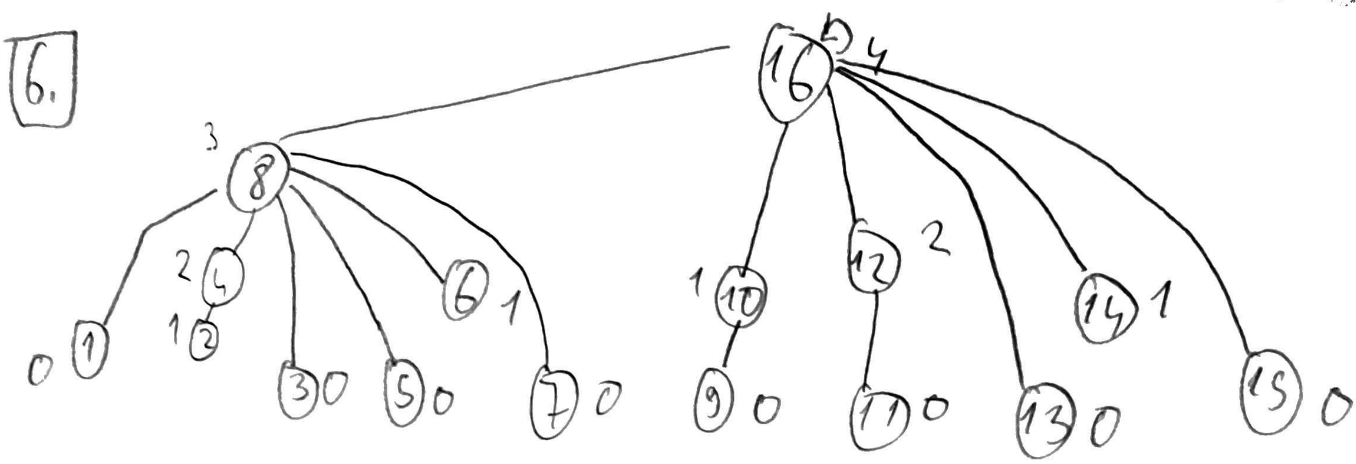
2.





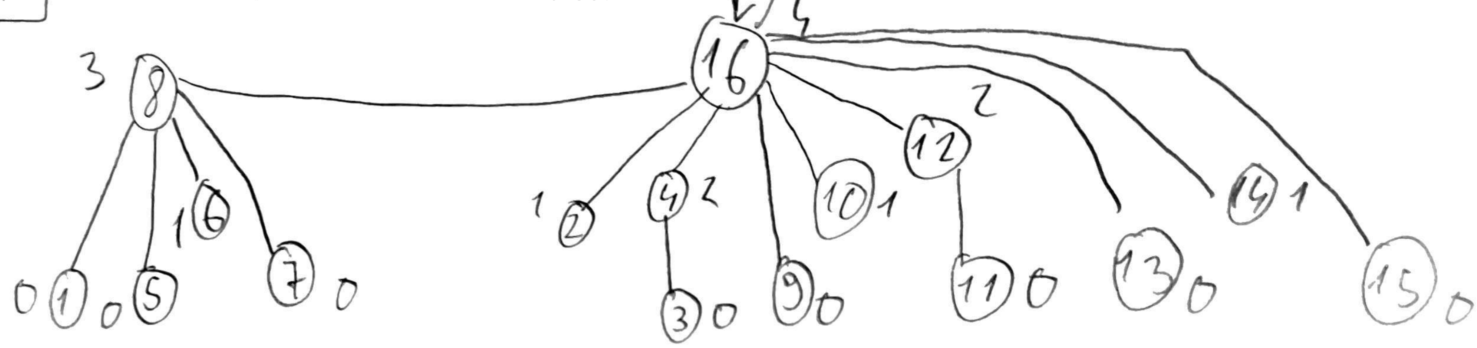


6.



7. FINAL, nakon oba FIND-PATH

find-set*



3.

niz:

1. for $i = 1 \ldots n$:
2. MAKE-SET(x_i)
3. for $i = 1 \ldots k$:
4. for $j = 1 \ldots m' - 2^{i-1}$; $j += 2^i$:
5. UNION$(x_j, x_{j+2^{i-1}})$
6. for $i = 1 \ldots m$:
7. FIND-SET(x_1)

Pretpostavimo da je $m' = 2^k$ najveća potencija broja 2 strogo manja od n. Nakon i-te iteracije unutarnje for petlje (4.) elementi $x_1, \ldots, x_{m'}$ su u stablima dubine i. Nakon što smo gotovi s procedurama (3.-5.) $x_1, \ldots, x_{m'}$ su u istom skupu, ali su reprezentirani stablom dubine $k \in \Omega(\log n)$. Nakon toga m puta radimo FIND_SET na x_1, što znači da m puta tražimo predstavnika koji je $\log n$ udaljen od čvora, pa je ovaj niz operacija $\Omega(m \log n)$.

4.

MAKE-SET(x):
1. napravi node n s atributima next, value, set
2. napravi linked listu L s head = tail = n, size = 1
3. n.next = NIL
4. n.set = L
5. n.value = x
6. return L

FIND-SET(x):
1. return x.set.head

UNION(x, y):
L1 = x.set
L2 = y.set
if L1.size ≤ L2.size
  L1.tail.next = L2.head
  r = L2.head
  while r != NIL:
    r.set = L1
    r = r.next
  L1.tail = L2.tail
  L1.size = L1.size + L2.size
  return L1
else

else sve isto samo simetricno, sada stavimo L2 na L1

5.

Za svaki član skupa napravimo pointer koji pointa na zadnji element linked liste. Zato za svaki skup možemo naći njegovog predstavnika (zadnji el. u listi) tako što odemo na head pa na pointer koji pointa na zadnji element liste. Pošto samo pratimo pointere, VSA je O(1). MAKE-SET ostaje isti, a UNION se malo promijeni. Kod UNIONA moramo postaviti manji skup prije većeg (heuristika težinske unije) i updateati pointer na predstavnika skupa za manji skup da pointa na novog predstavnika (zadnji el. većeg skupa).