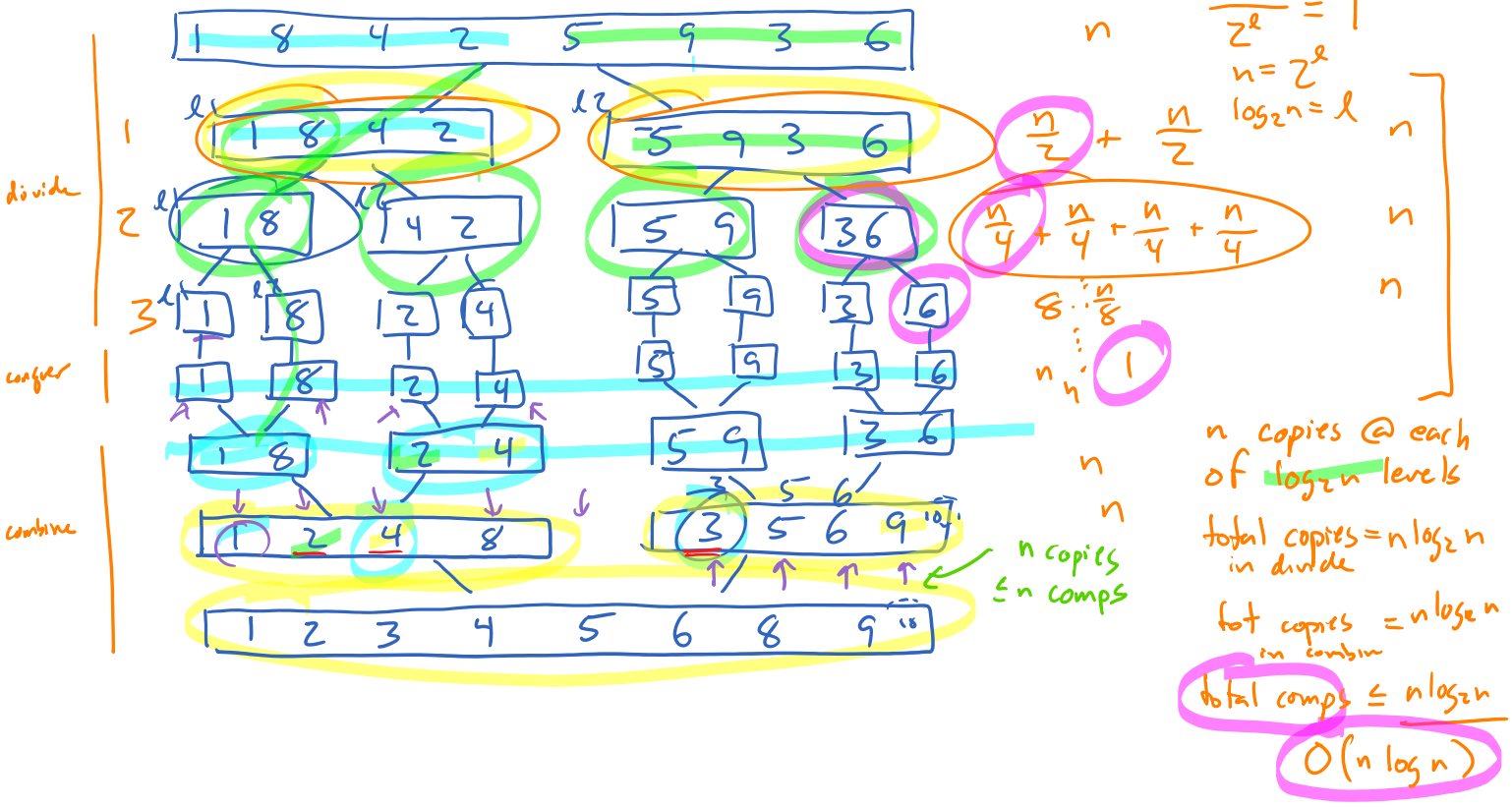
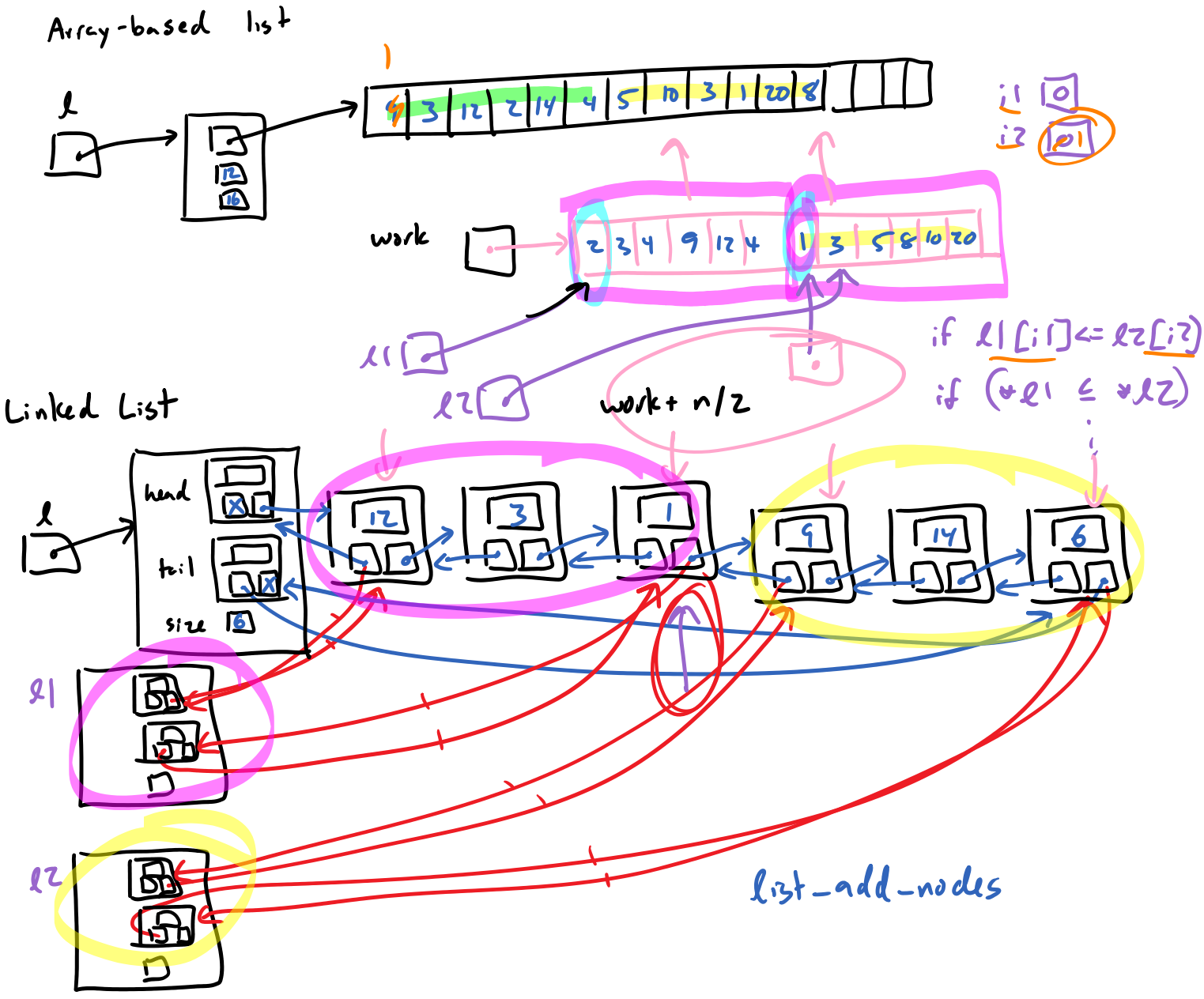
Mergesort

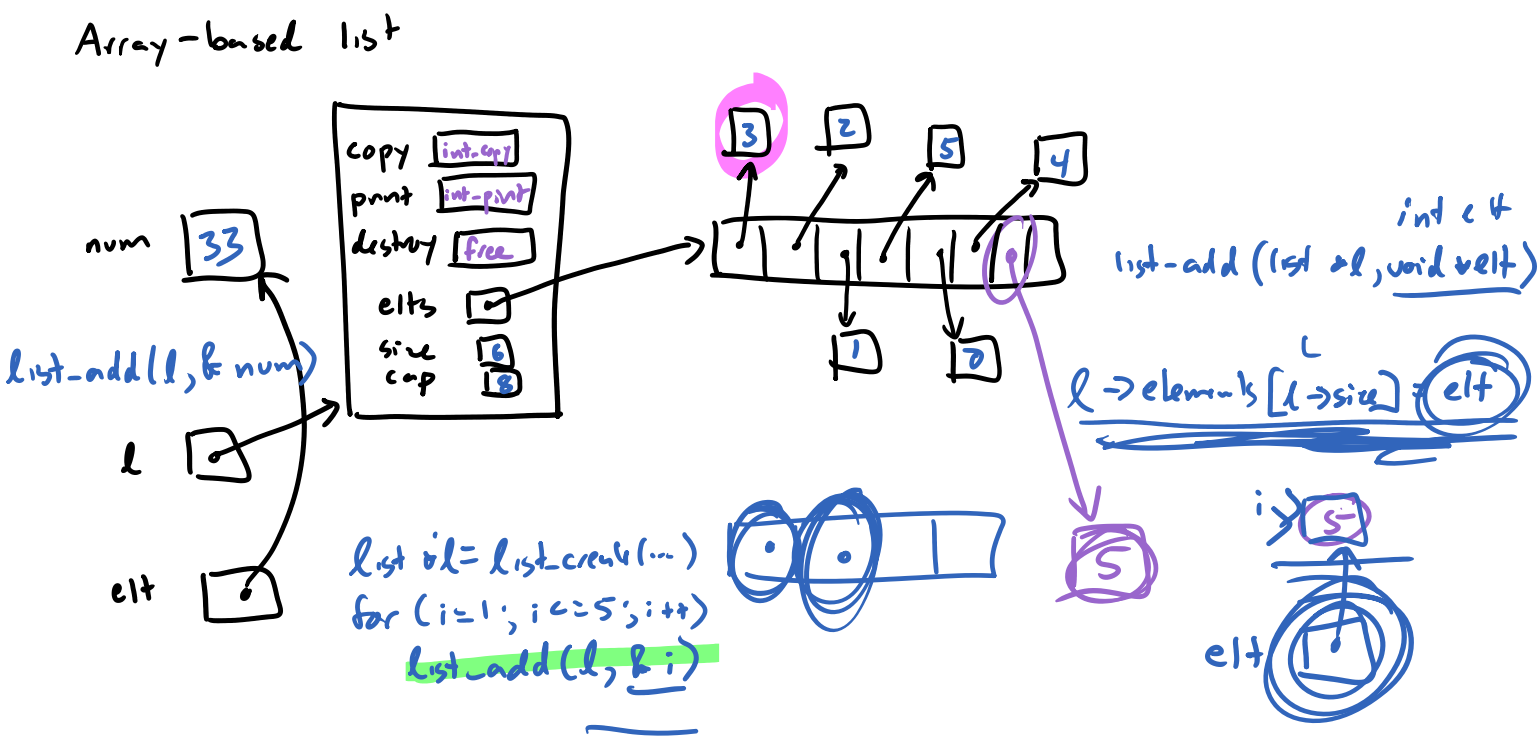
```
MERGE-SORT ( A )
    if len(A) < 2
        return a copy of A
    else
        L ← 1st half of A ✓
        R ← rest of A ✓          ] divide list into two parts
        L ← MERGESORT(L) ✓
        R ← MERGESORT(R) ✓       ] conquer the subproblems
        return MERGE(L,R)        ] combine results

    — precondition: L, R are sorted
MERGE(L,R)
    A ← empty list
    i ← 0
    j ← 0
    while i < len(L) and j < len(R)      as long as elements left in both lists
        if L[i] ≤ R[j]                   compare what's 1st among remaining items
            append L[i] to A
            i ← i+1
        else
            append R[j] to A
            j ← j+1
    append L[i ... len(L)-1] to A    ) concatenate remaining items on
    append R[j ... len(R)-1] to A    ) non-empty list to result
    return A
```

Mergesort Example

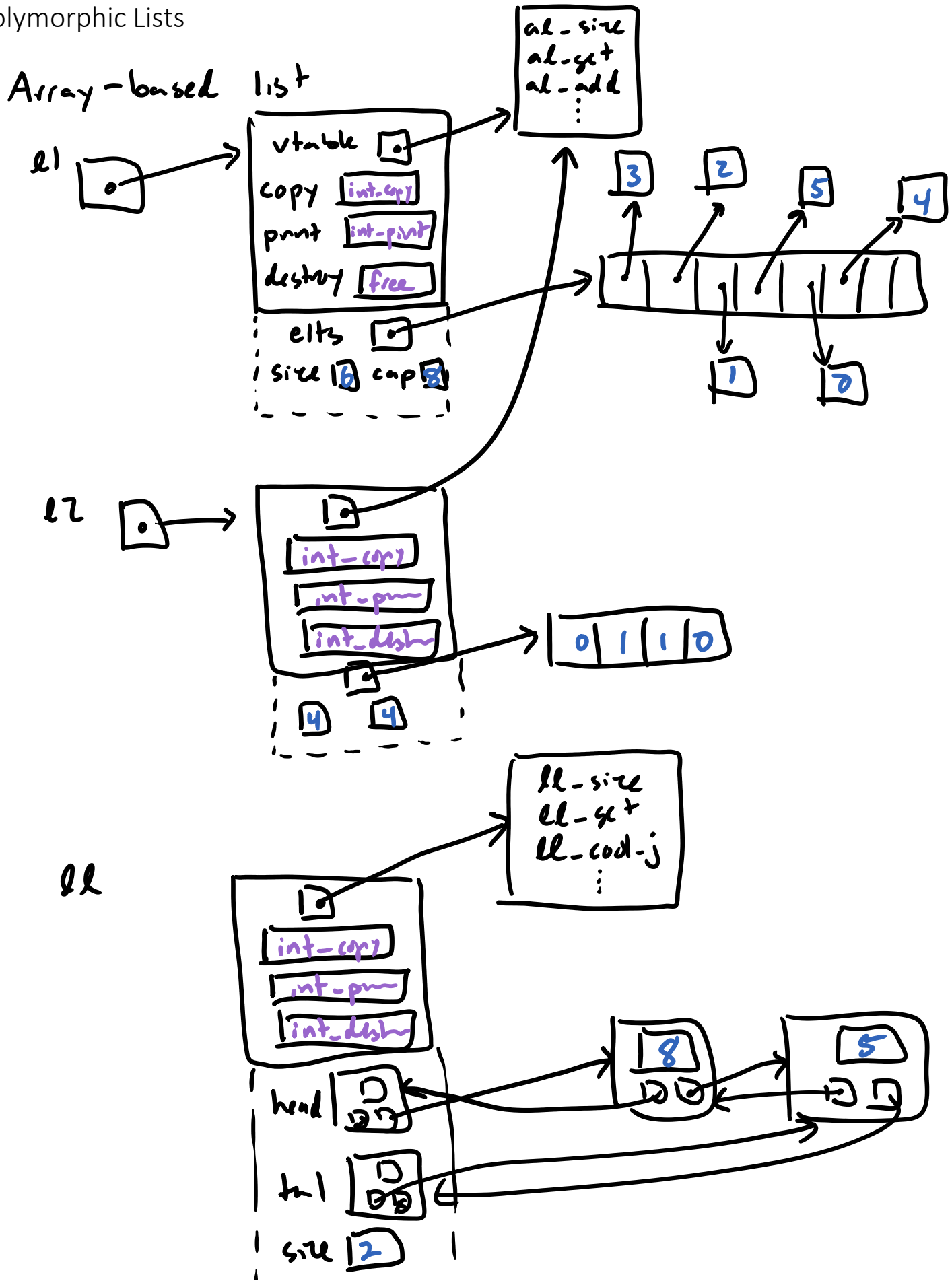


Splitting


Array-based list
l
12
16
work
work + n/2
l1
l2
i1
i2
if l1[i1] <= l2[i2]
if (*l1 <= *l2)
Linked List
head
tail
size
16
list_add_nodes


Array-based list
copy
print
destroy
elts
size
cap
int_copy
int_print
free
num
33
list_add(l, &num)
l
elt
3
2
5
4
1
0
int elt
list_add(list *l, void *elt)
l->elements[l->size] = elt
list *l = list_create(...)
for (i=1; i<=5; i++)
list_add(l, &i)
5
i
5
elt

Polymorphic Lists




size 12