Garbage Collection

Guido Wachsmuth

TU Delft
Delft University of Technology
Challenge the future

Course IN4303, 2012/13
Compiler Construction

Overview
today's lecture

garbage collection

Overview
today's lecture

garbage collection

reference counting

- deallocate records with count 0

Overview
today's lecture

garbage collection

reference counting

- deallocate records with count 0

mark & sweep

- mark reachable records
- sweep unmarked records

Overview
today's lecture

garbage collection

reference counting

- deallocate records with count 0

mark & sweep

- mark reachable records
- sweep unmarked records

copy collection

- copy reachable records
- consider generations of records

I

garbage collection

the heap

Recap: Java Virtual Machine

the heap

method area

pc: 00	constant pool
00 12 ldc	00 4303 4303
01 00 00	01 0000 0004
02 19 aload	02
03 00 00	03
04 12 ldc	04
05 01 01	05
06 2E iaload	06

stack

optop: 00	local variables
00	00 002A 002A
01	01
02	02
03	03
04	04
05	05
06	06

heap

4303 4303 "Compilers"

002A 002A [20,01,40,02,42]

Recap: Java Virtual Machine

the heap

method area

pc: 02	constant pool
00 12 ldc	00 4303 4303
01 00 00	01 0000 0004
02 19 aload	02
03 00 00	03
04 12 ldc	04
05 01 01	05
06 2E iaload	06

stack

optop: 01	local variables
00 4303 4303	00 002A 002A
01	01
02	02
03	03
04	04
05	05
06	06

heap

4303 4303 "Compilers" 002A 002A [20,01,40,02,42]

Recap: Java Virtual Machine

the heap

method area

pc: 04	constant pool
00 12 ldc	00 4303 4303
01 00 00	01 0000 0004
02 19 aload	02
03 00 00	03
04 12 ldc	04
05 01 01	05
06 2E iaload	06

stack

optop: 02	local variables
00 4303 4303	00 002A 002A
01 002A 002A	01
02	02
03	03
04	04
05	05
06	06

heap

4303 4303 "Compilers" 002A 002A [20,01,40,02,42]

Recap: Java Virtual Machine

the heap

method area

pc: 06	constant pool
00 12 ldc	00 4303 4303
01 00 00	01 0000 0004
02 19 aload	02
03 00 00	03
04 12 ldc	04
05 01 01	05
06 2E iaload	06

stack

optop: 03	local variables
00 4303 4303	00 002A 002A
01 002A 002A	01
02 0000 0004	02
03	03
04	04
05	05
06	06

heap

4303 4303 "Compilers" 002A 002A [20,01,40,02,42]

Recap: Java Virtual Machine

the heap

method area

pc: 07	constant pool
00 12 ldc	00 4303 4303
01 00 00	01 0000 0004
02 19 aload	02
03 00 00	03
04 12 ldc	04
05 01 01	05
06 2E iaload	06

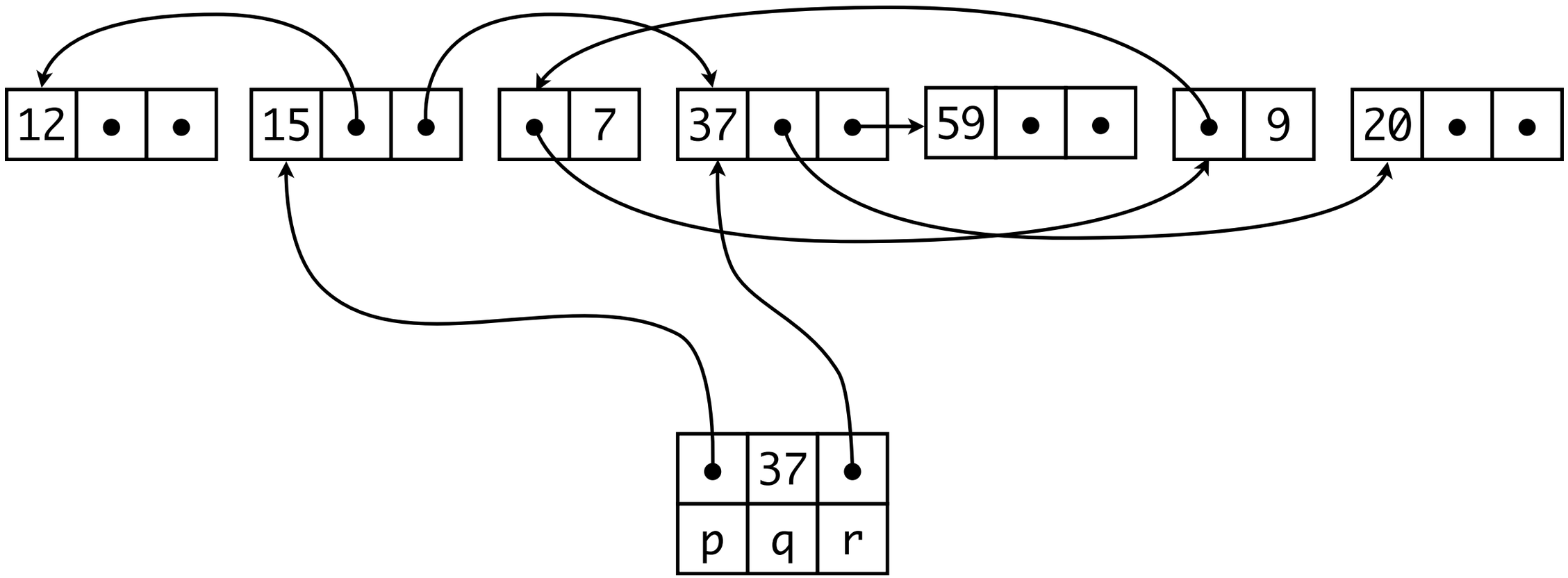
stack

optop: 02	local variables
00 4303 4303	00 002A 002A
01 0000 0042	01
02	02
03	03
04	04
05	05
06	06

heap

4303 4303 "Compilers" 002A 002A [20,01,40,02,42]

Heap
example



II

reference counting

Reference counts

idea

counts

- how many pointers point to each record?
- store with each record

counting

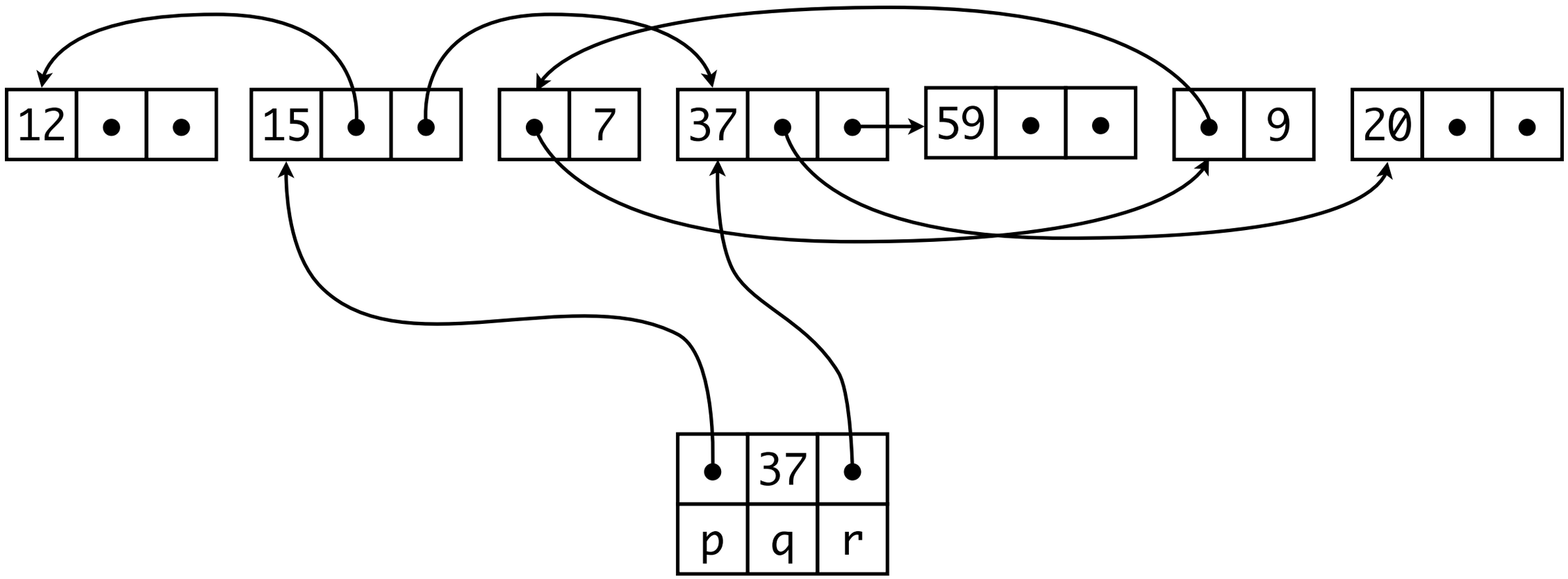
- extra instructions

deallocate

- put on freelist
- recursive deallocation on allocation

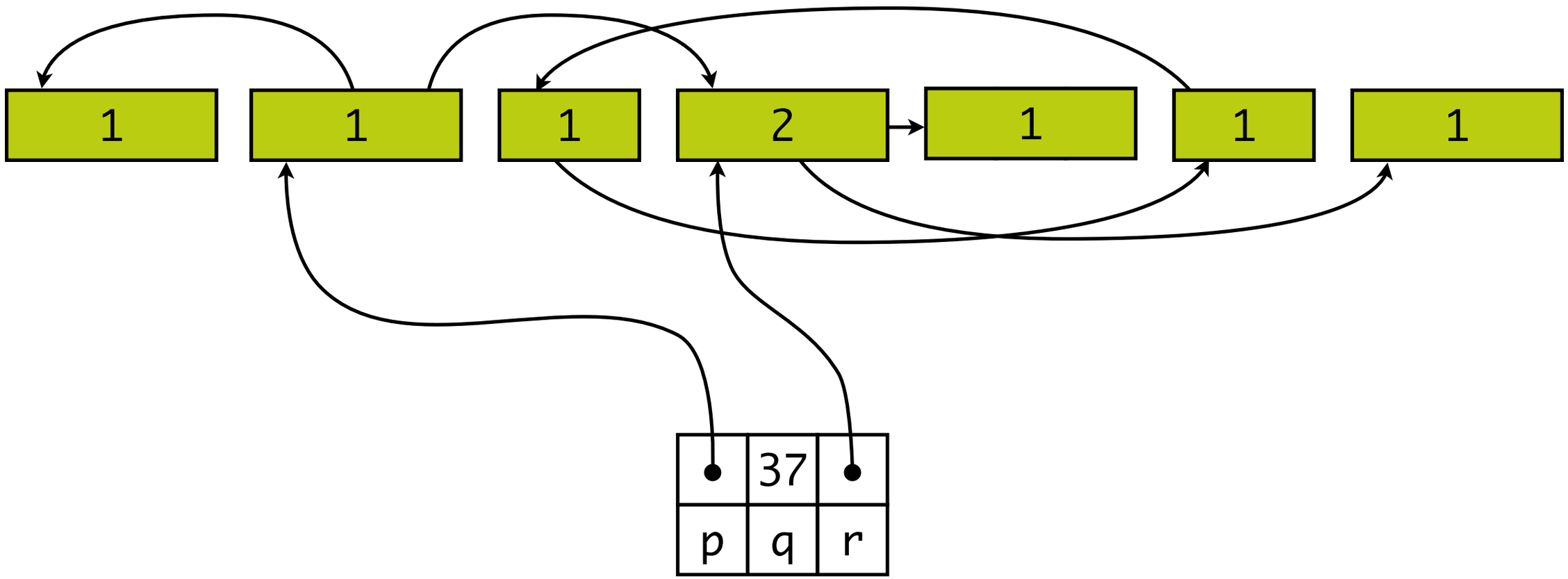
Reference counts

example



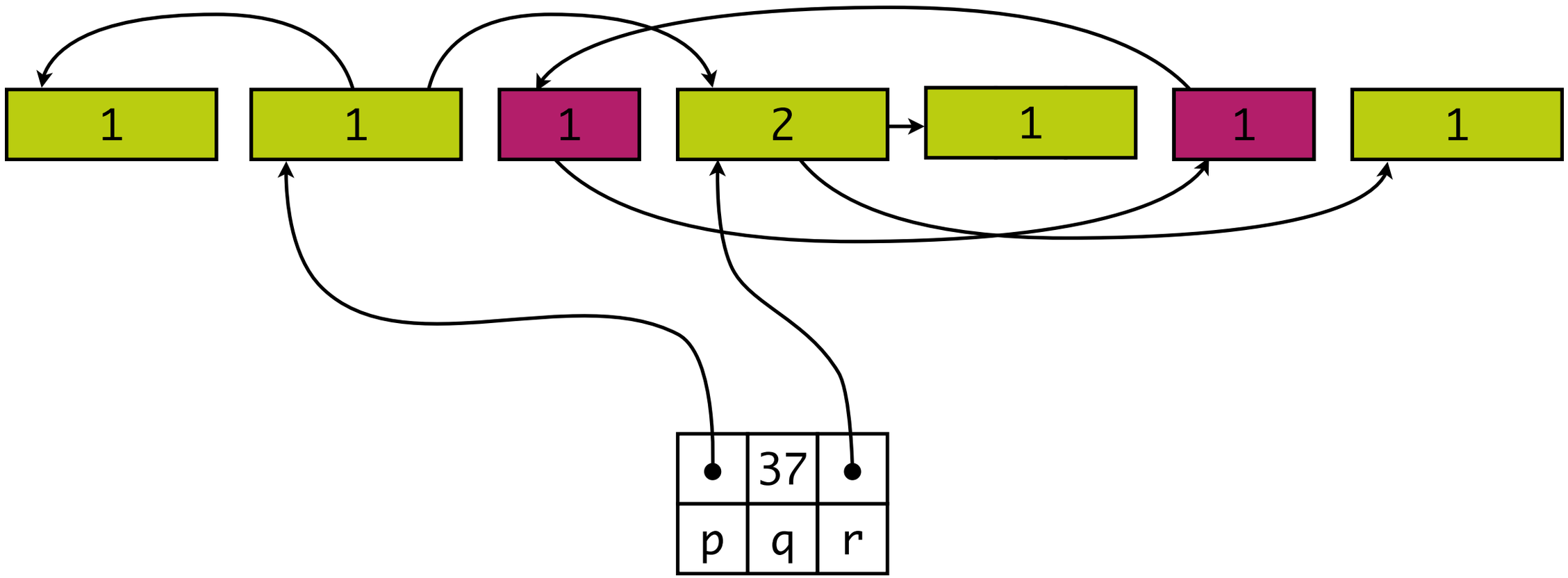
Reference counts

example



Reference counts

example



Reference counts

notes

cycles

- memory leaks
- break cycles explicitly
- occasional mark & sweep collection

expensive

- fetch, decrease, store old reference counter
- possible deallocation
- fetch, increase, store new reference counter

III

mark & sweep

Mark & sweep

idea

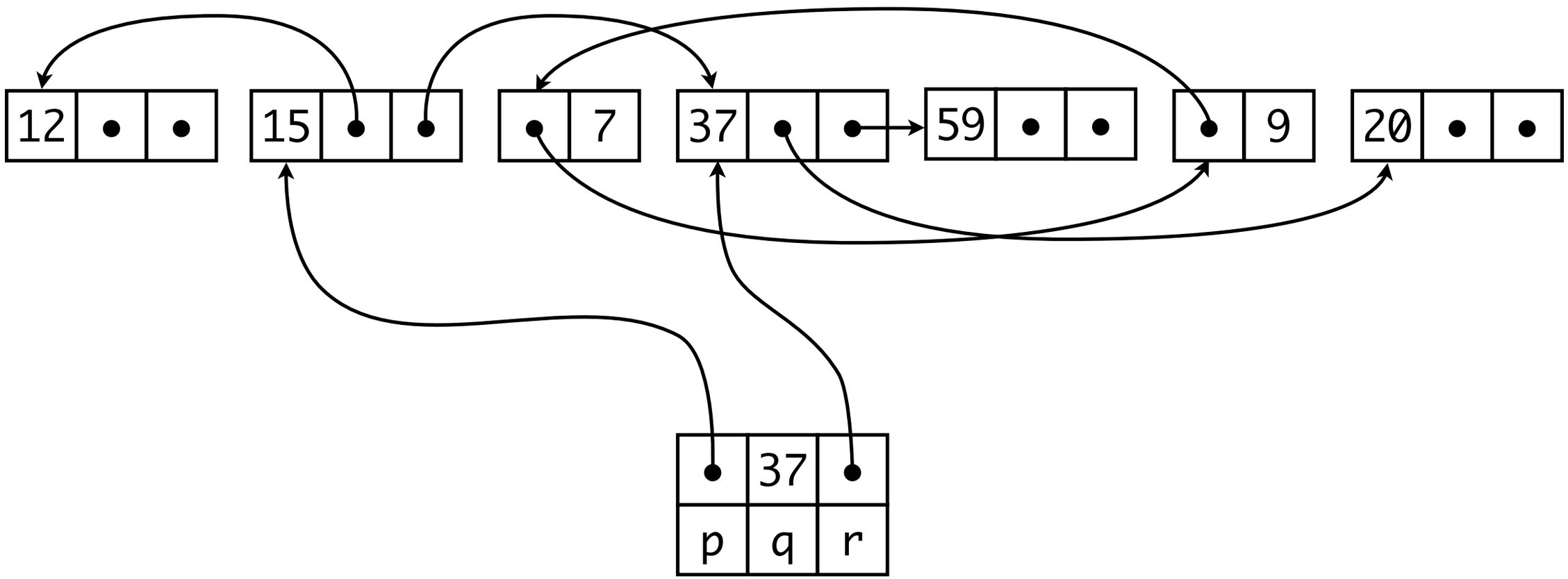
mark

- mark reachable records
- start at variables (roots)
- follow references

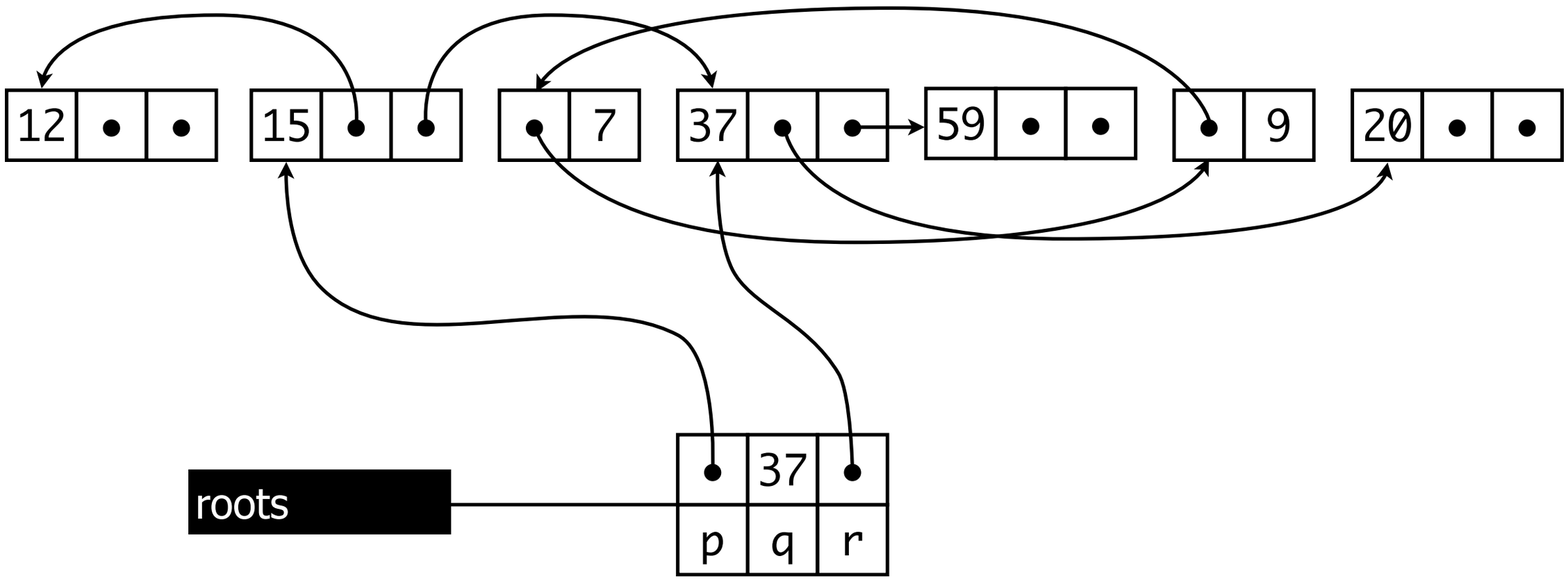
sweep

- marked records: unmark
- unmarked records: deallocate
- linked list of free records

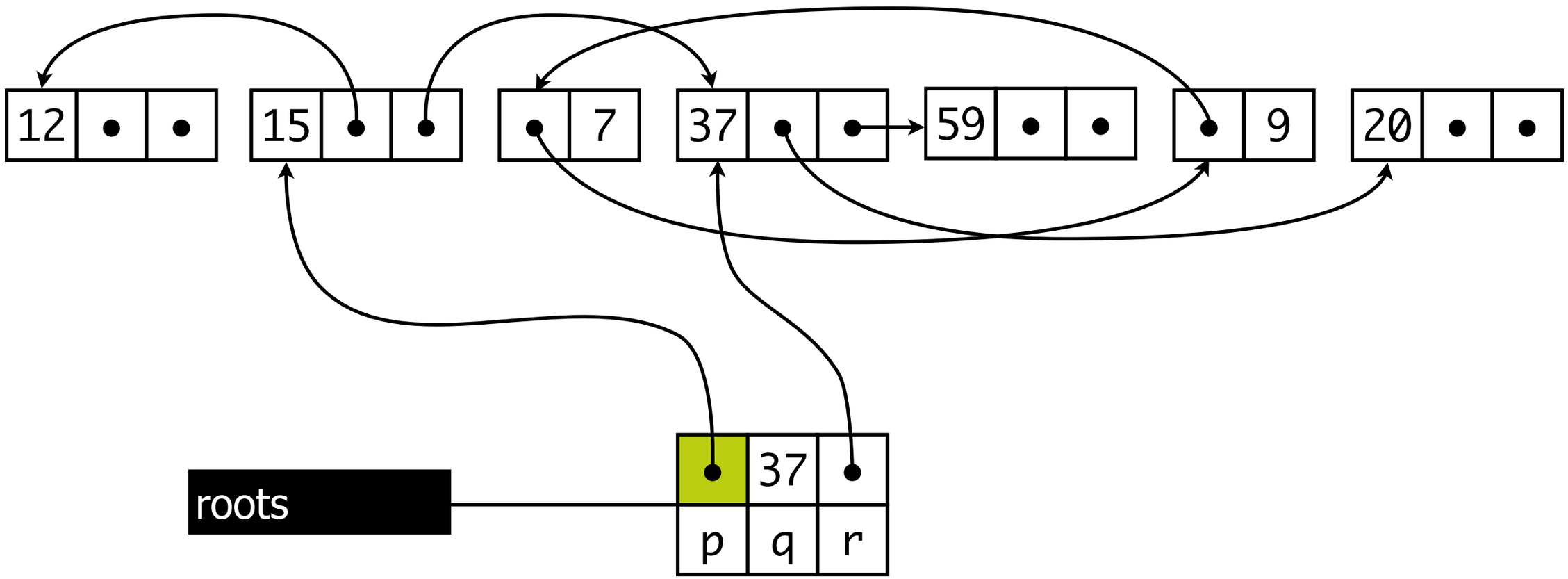
Marking
example



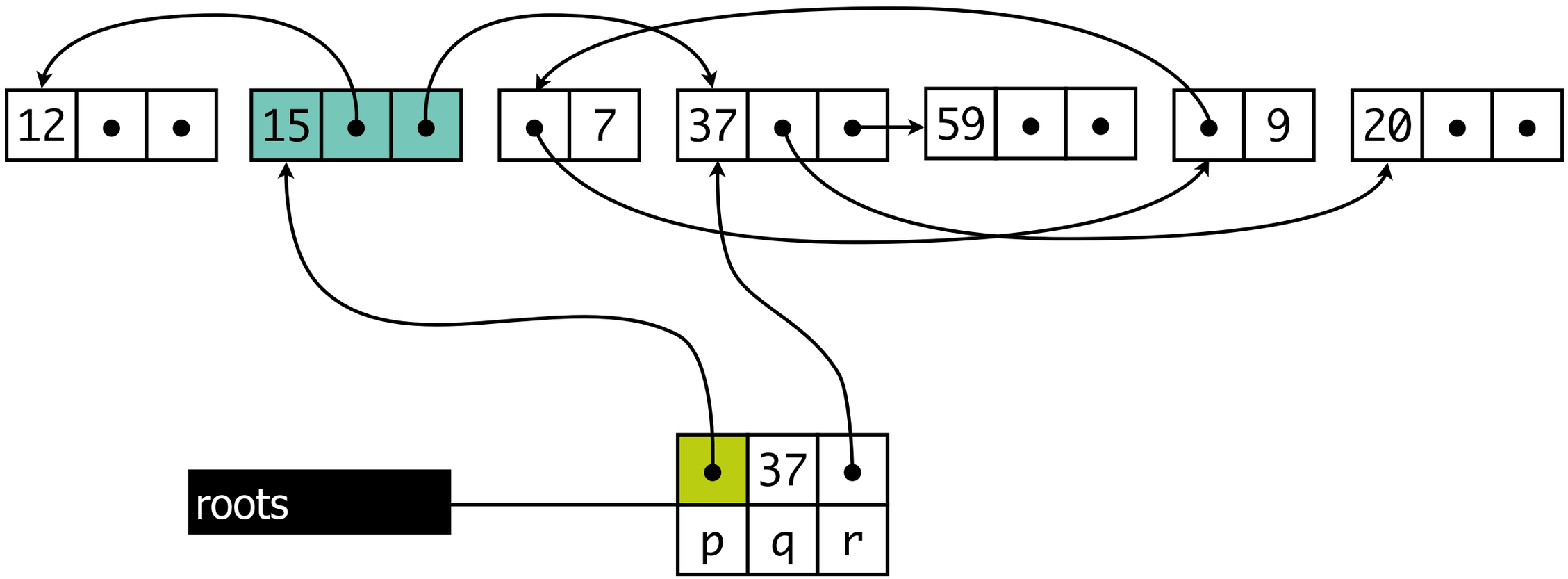
Marking
example



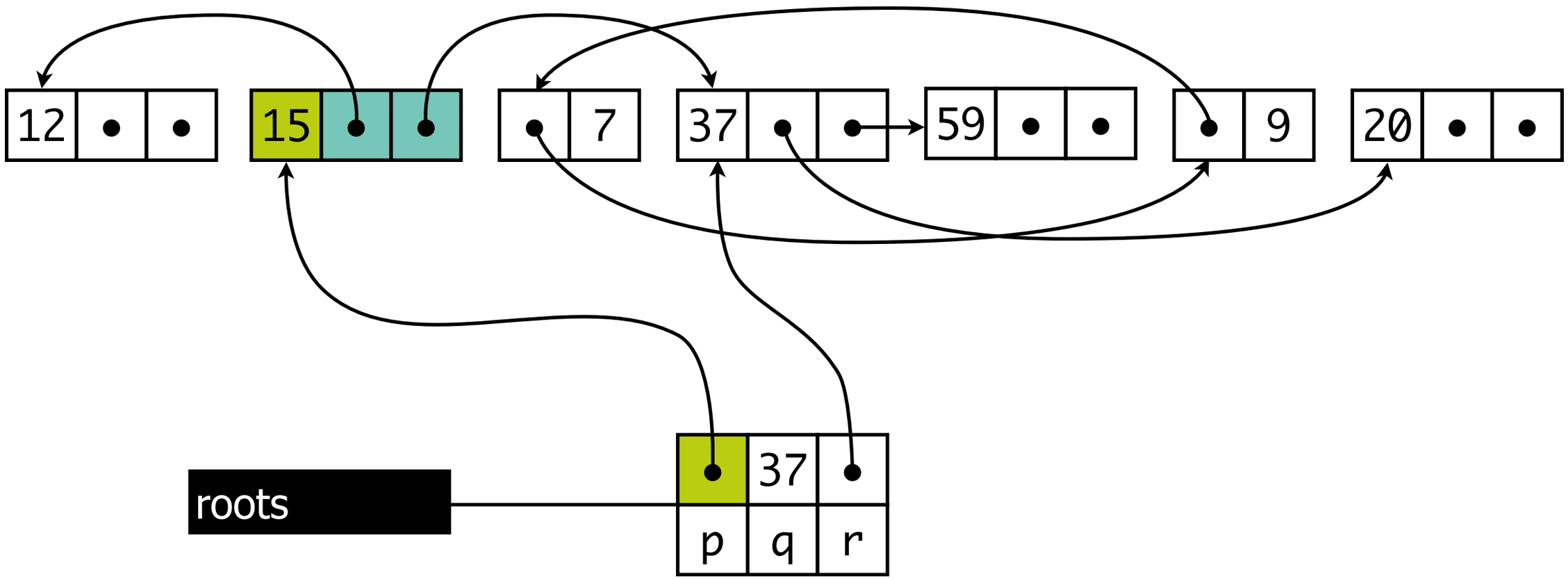
Marking
example



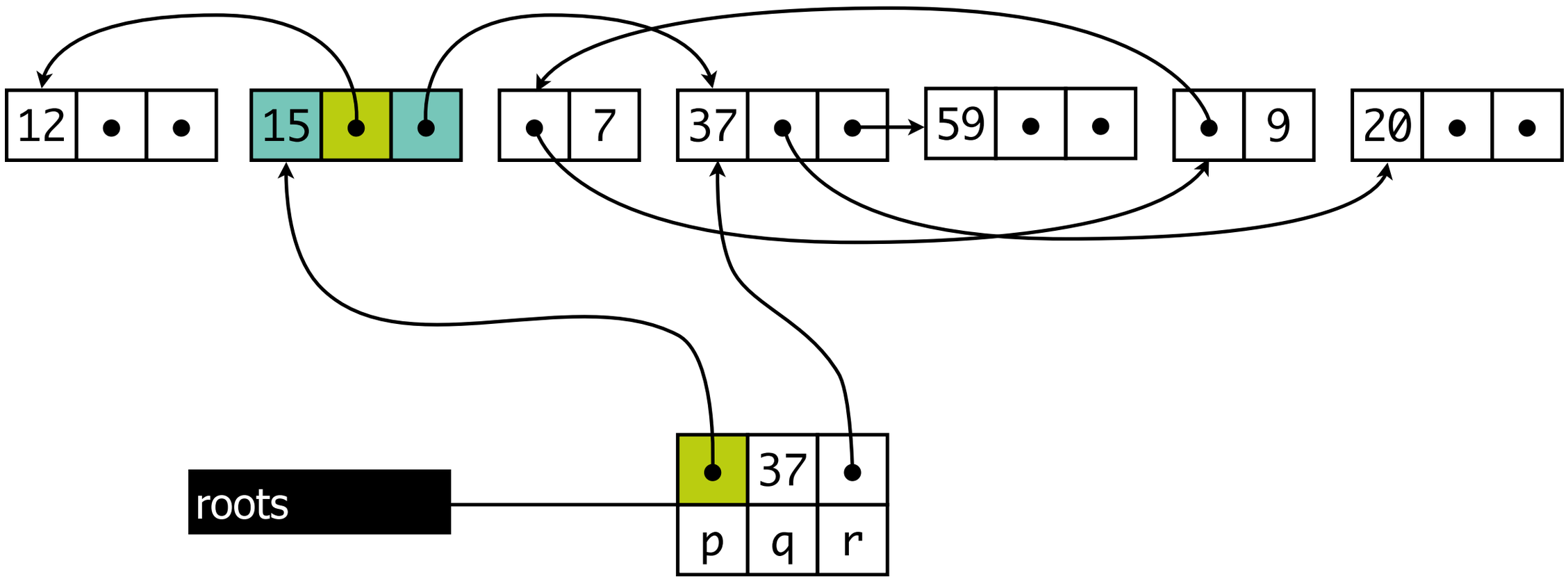
Marking
example



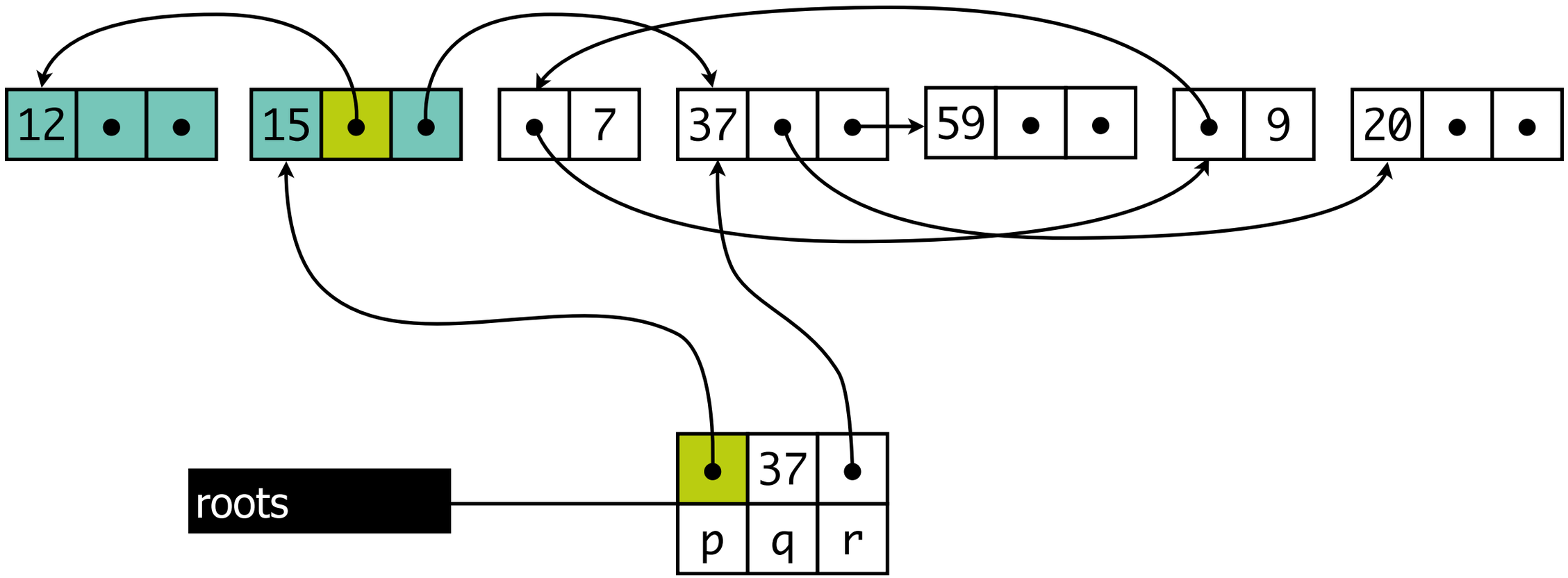
Marking
example



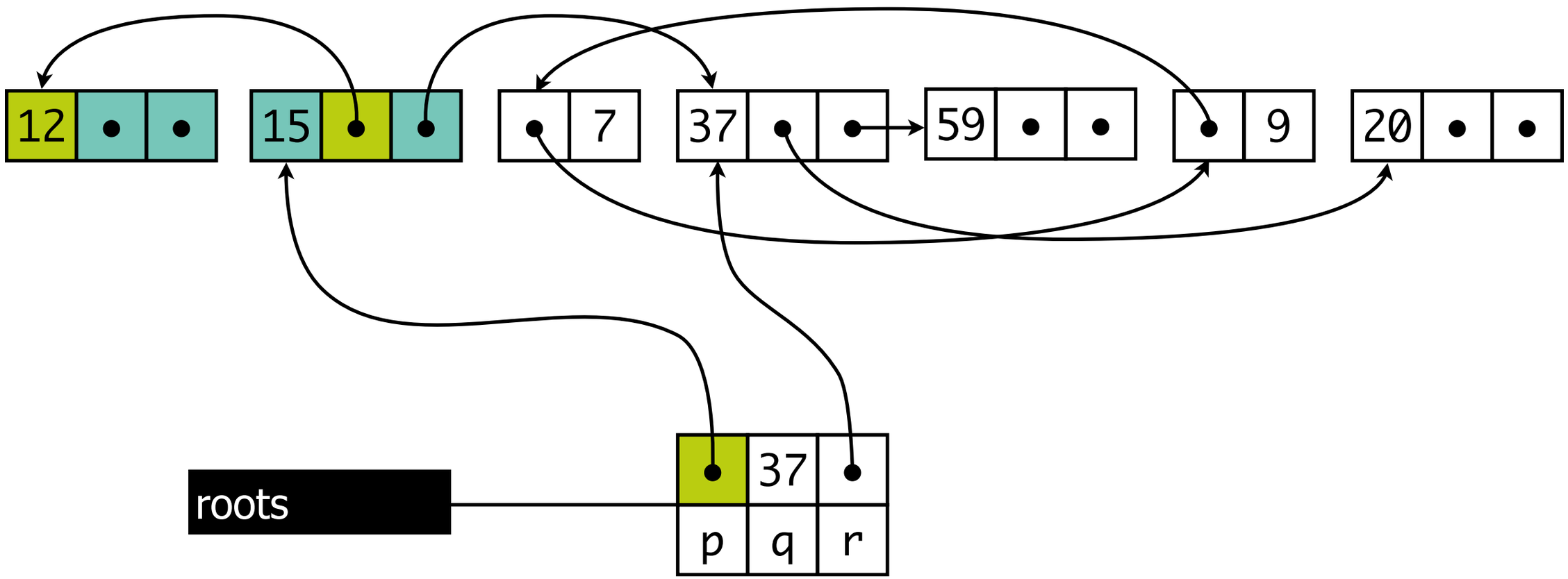
Marking
example



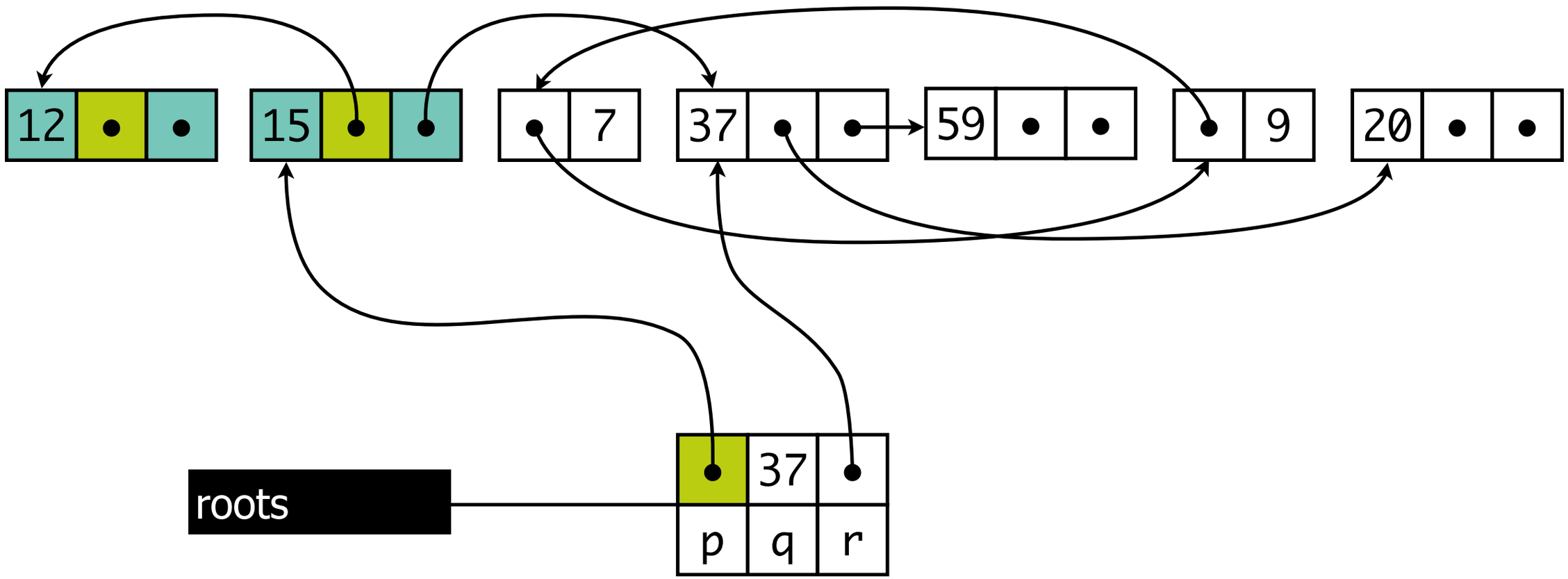
Marking
example



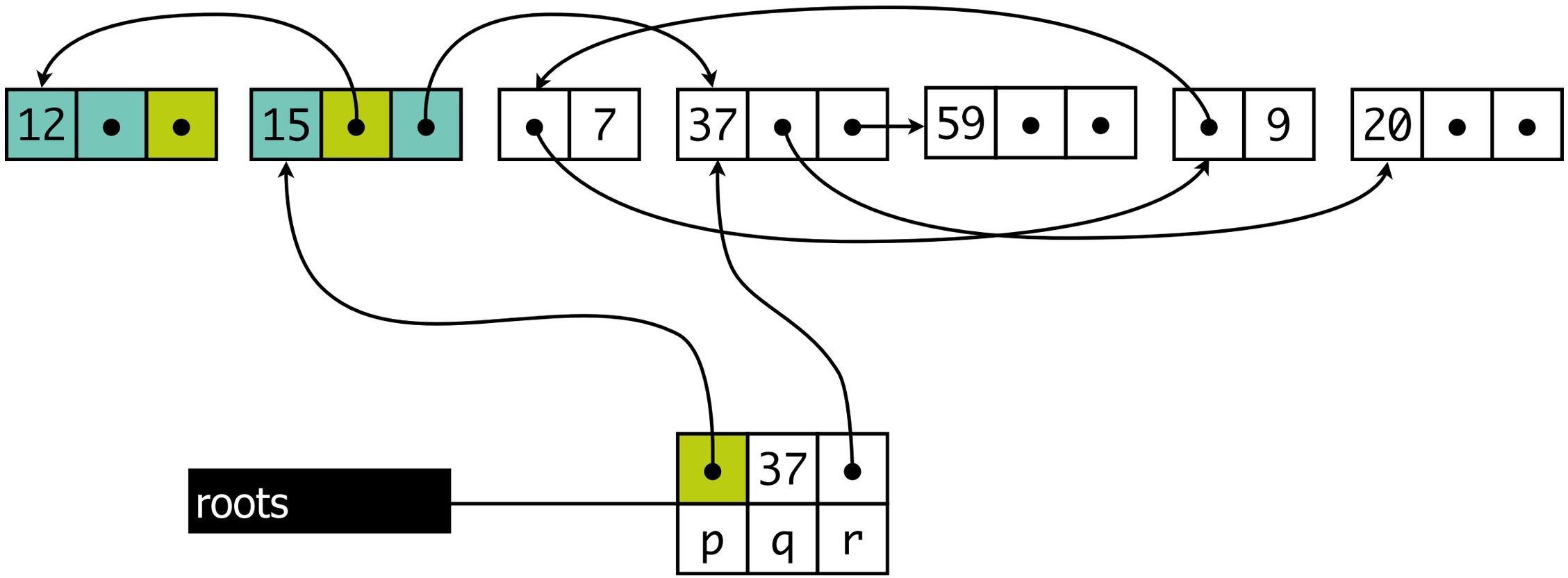
Marking
example



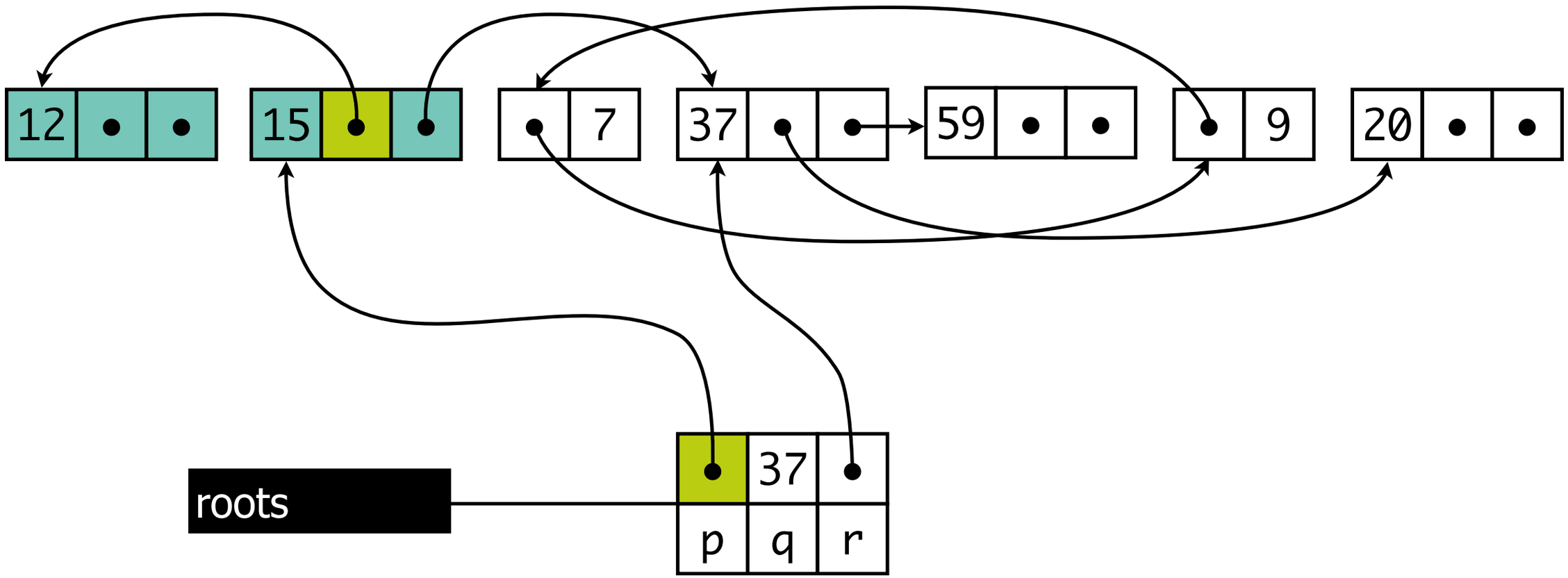
Marking
example



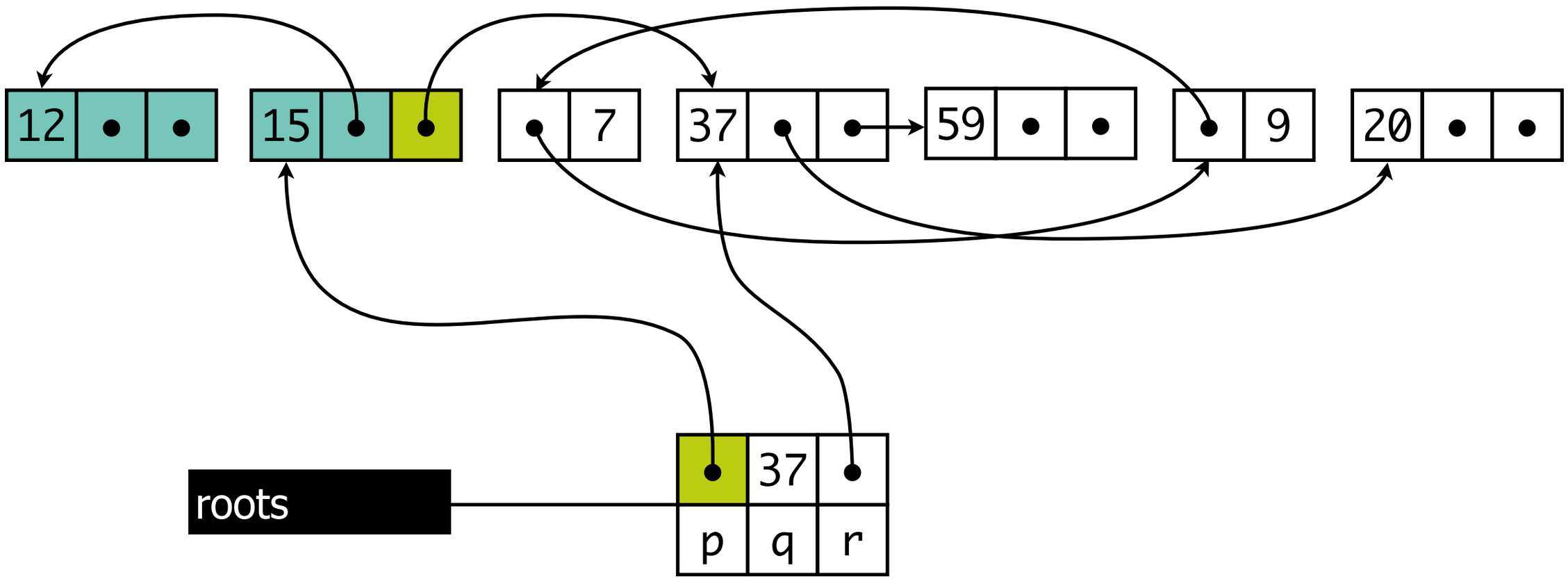
Marking
example



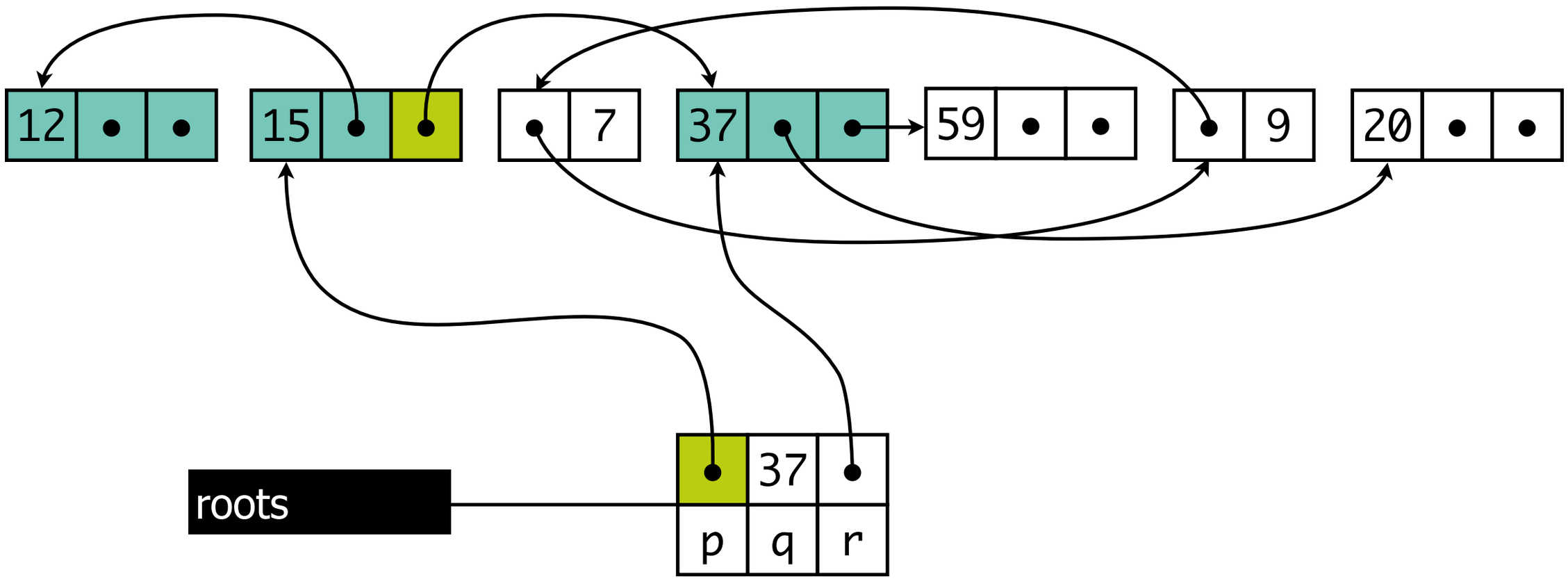
Marking
example



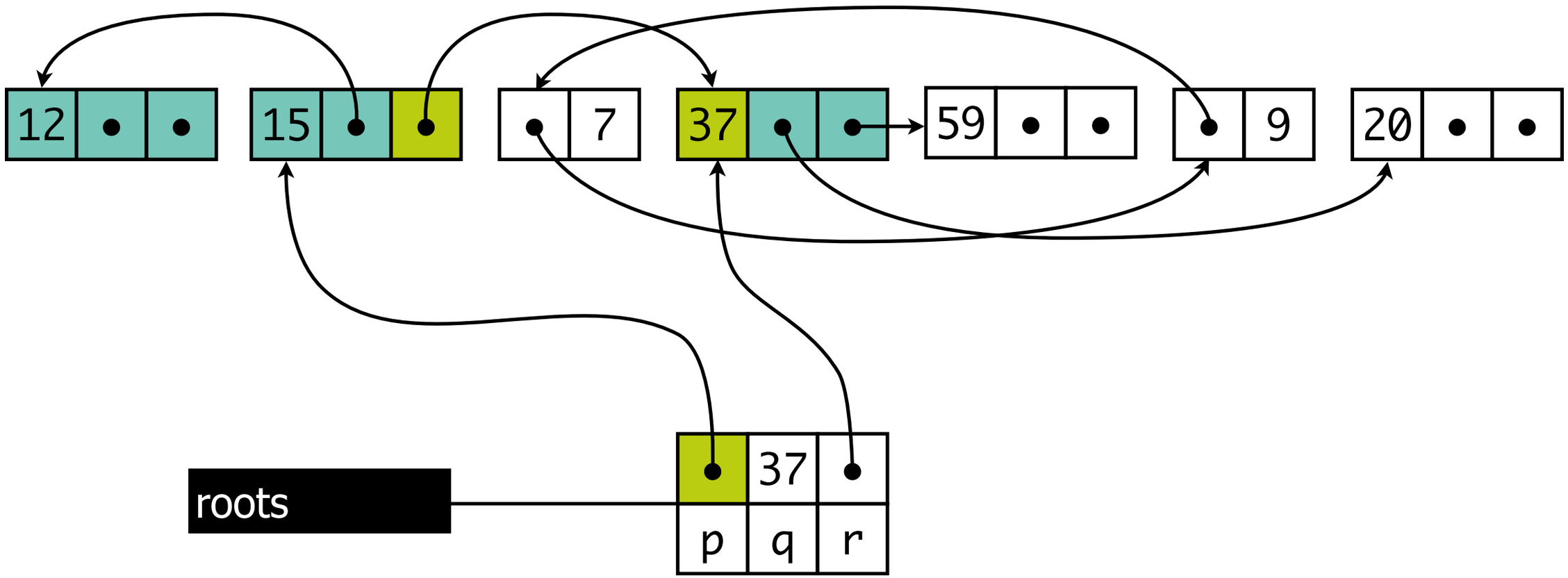
Marking
example



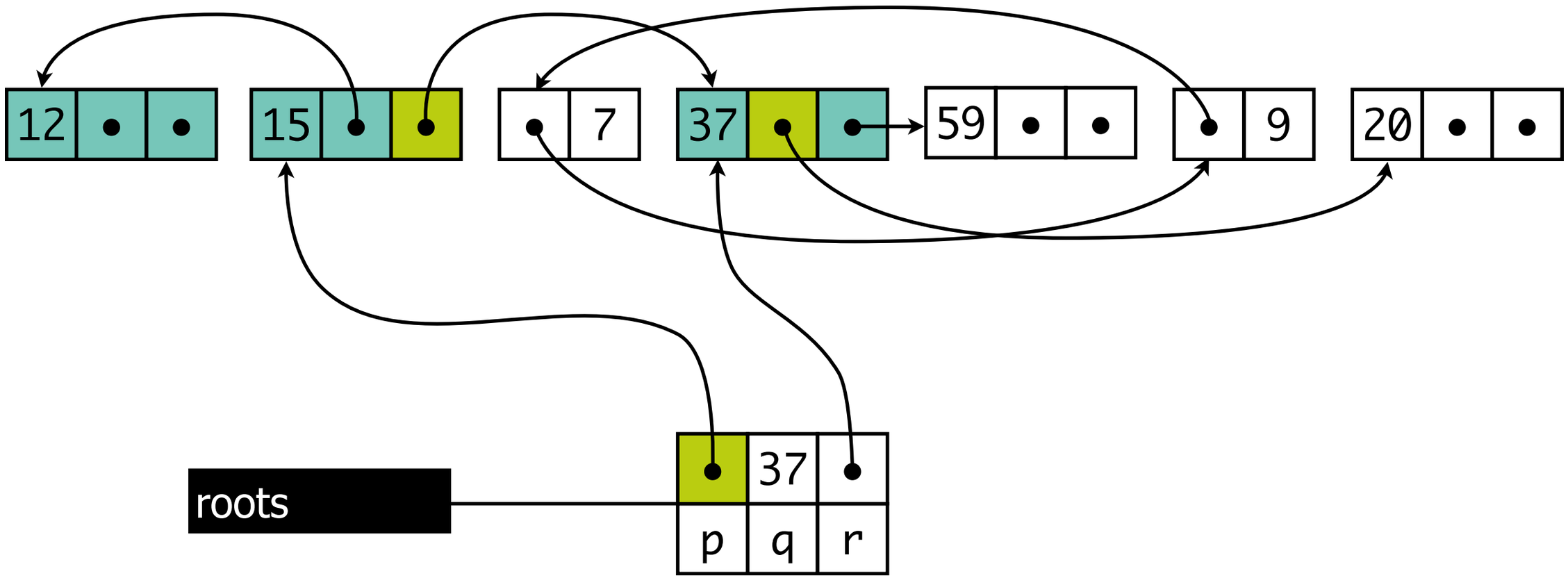
Marking
example



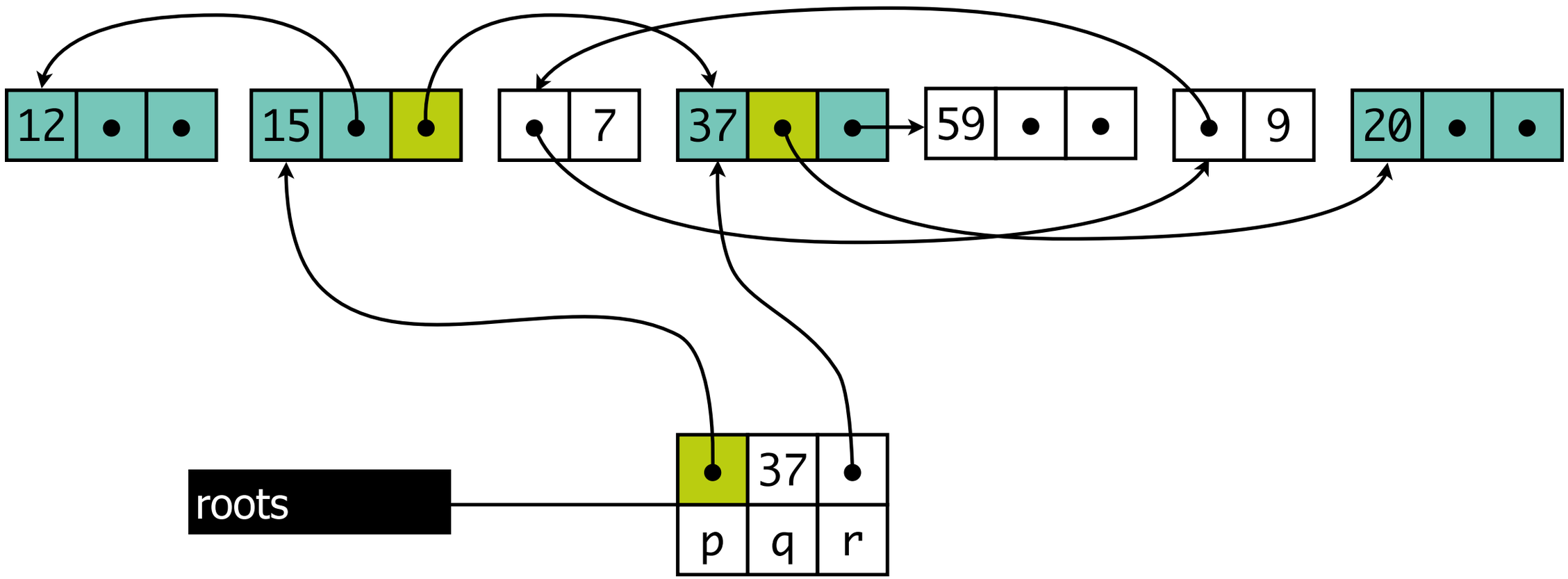
Marking
example



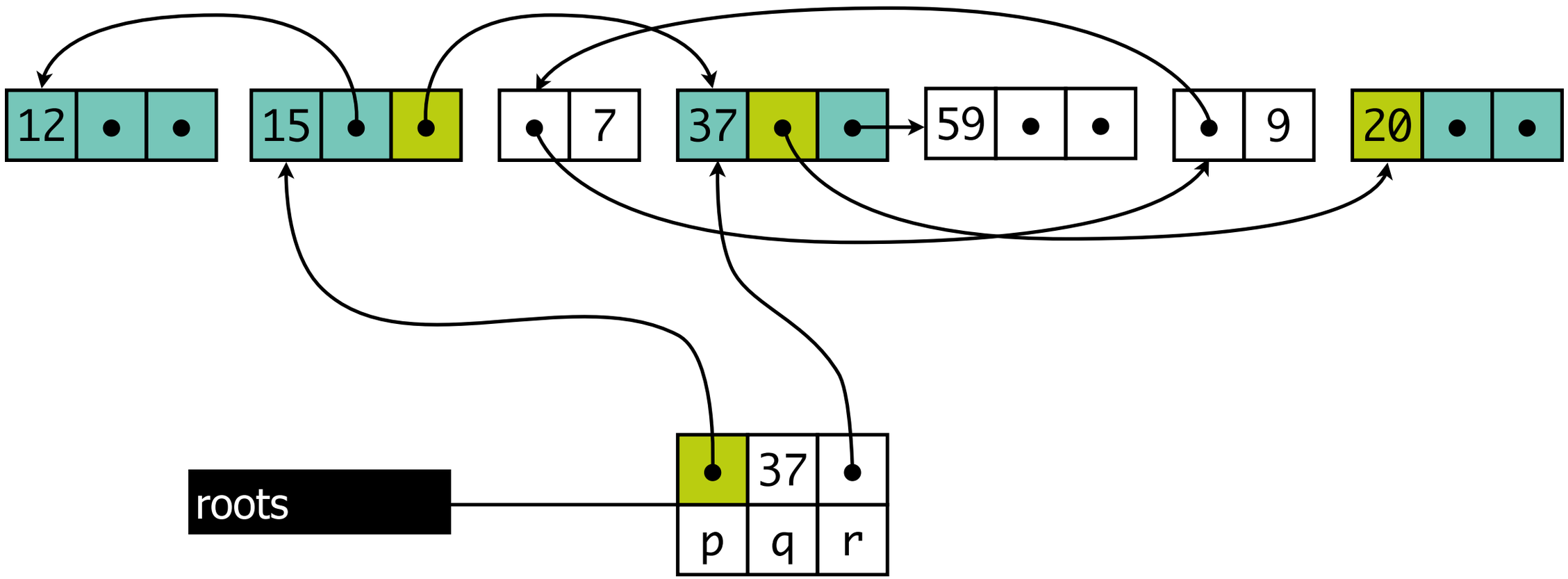
Marking
example



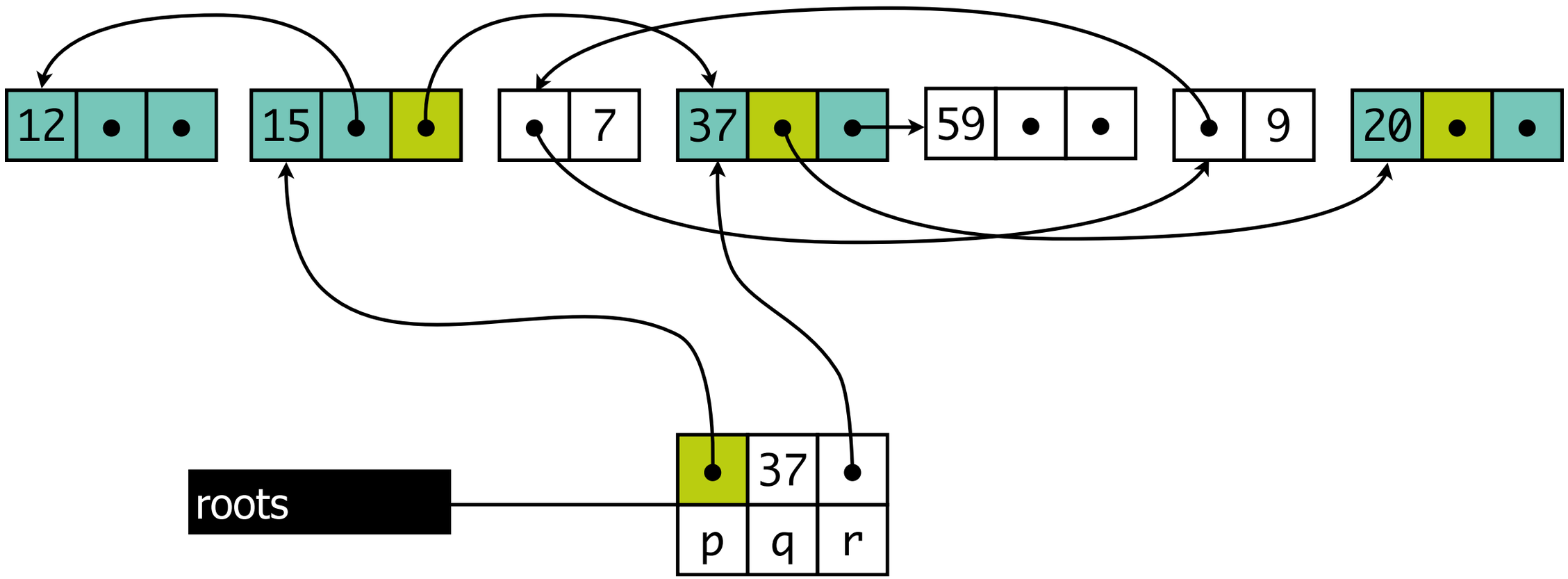
Marking
example



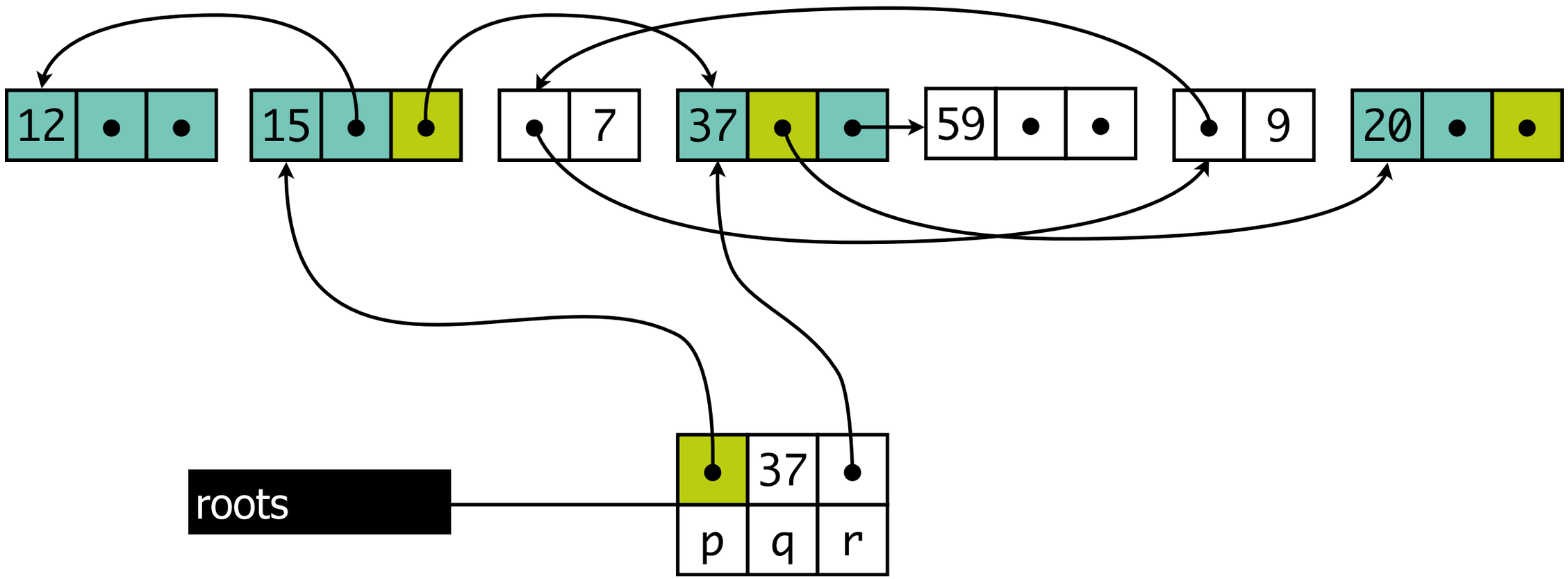
Marking
example



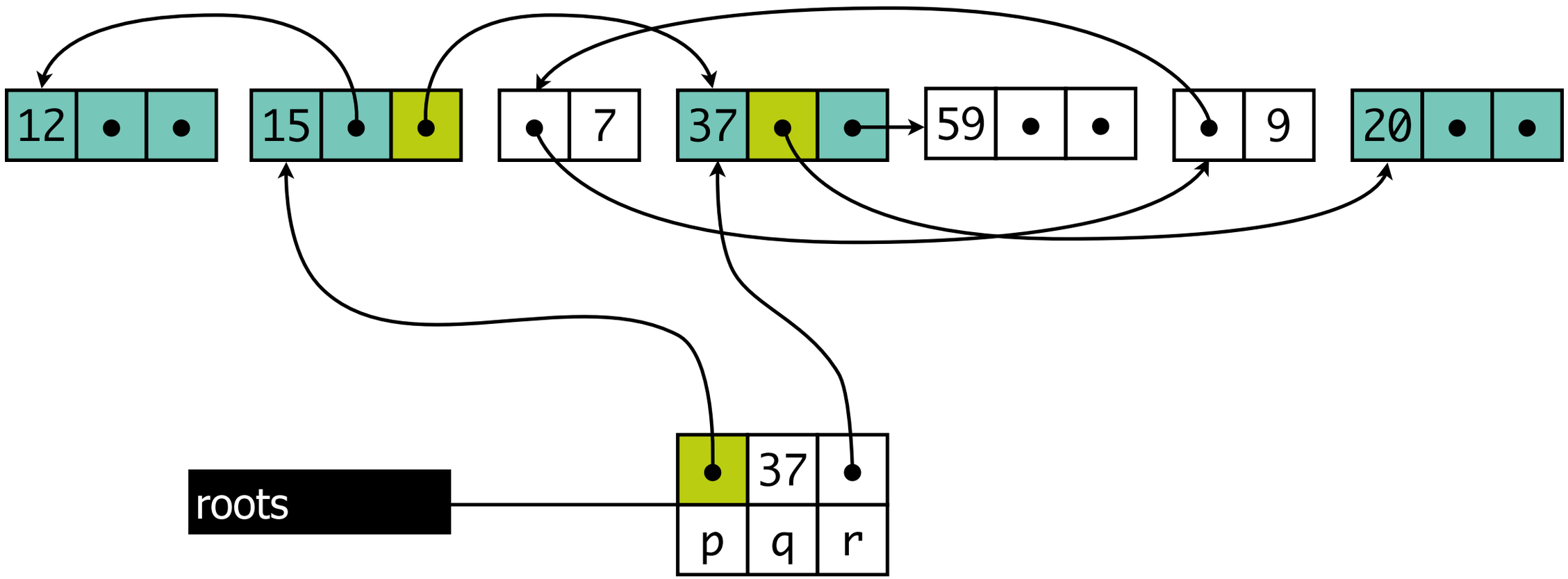
Marking
example



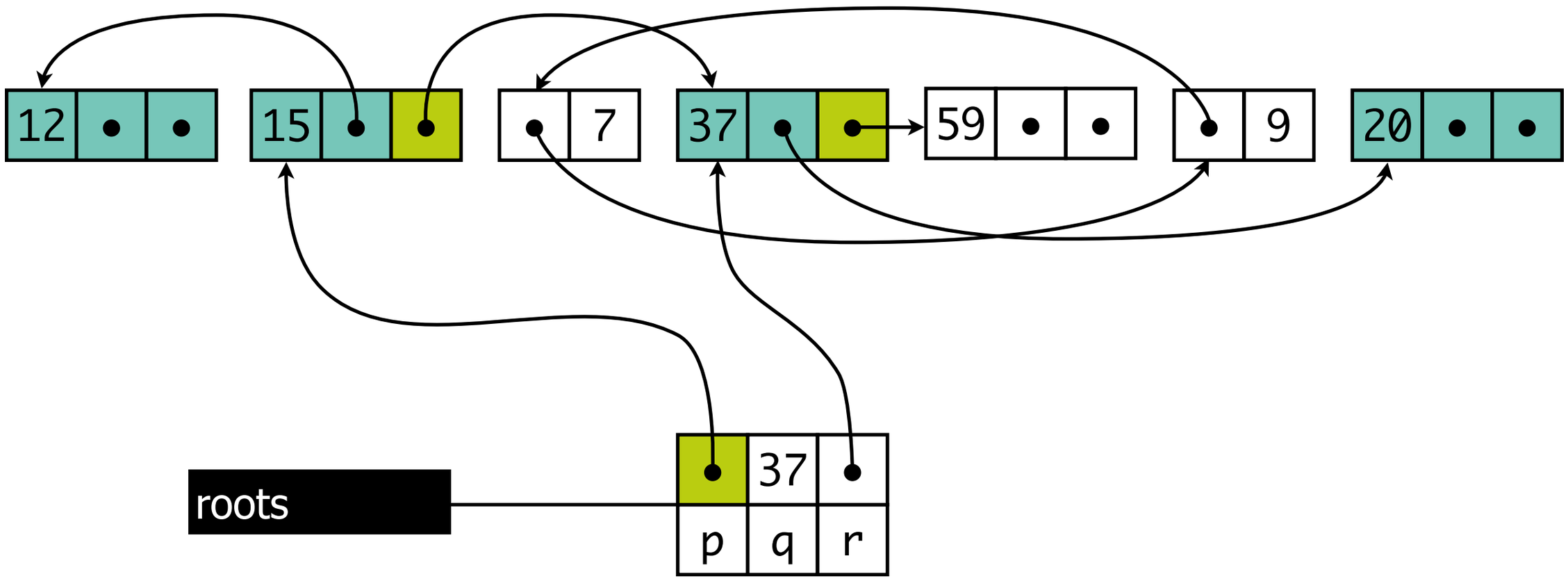
Marking
example



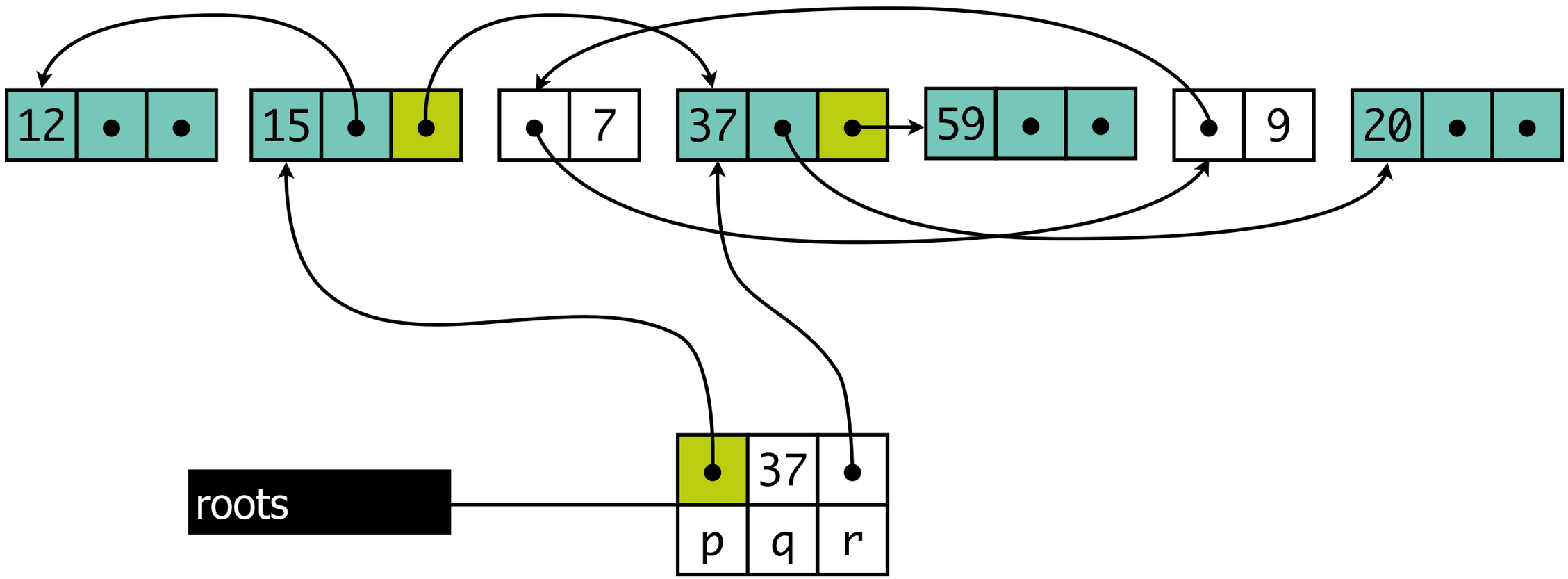
Marking
example



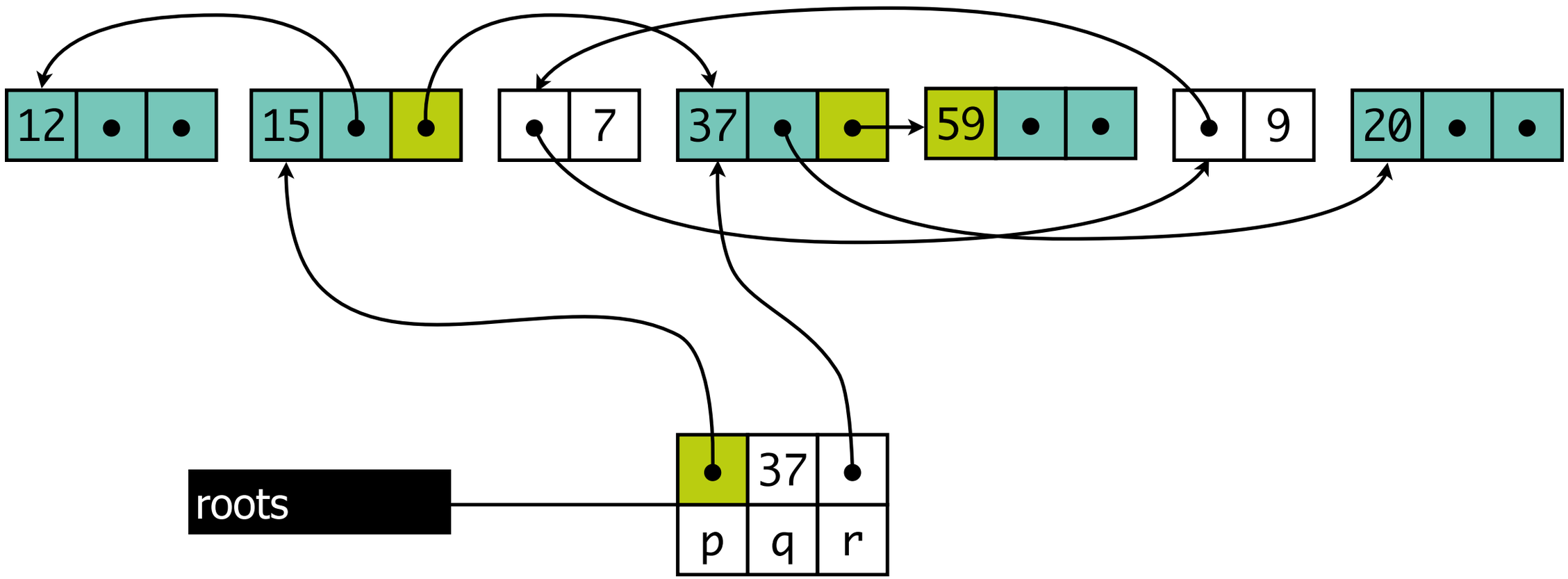
Marking
example



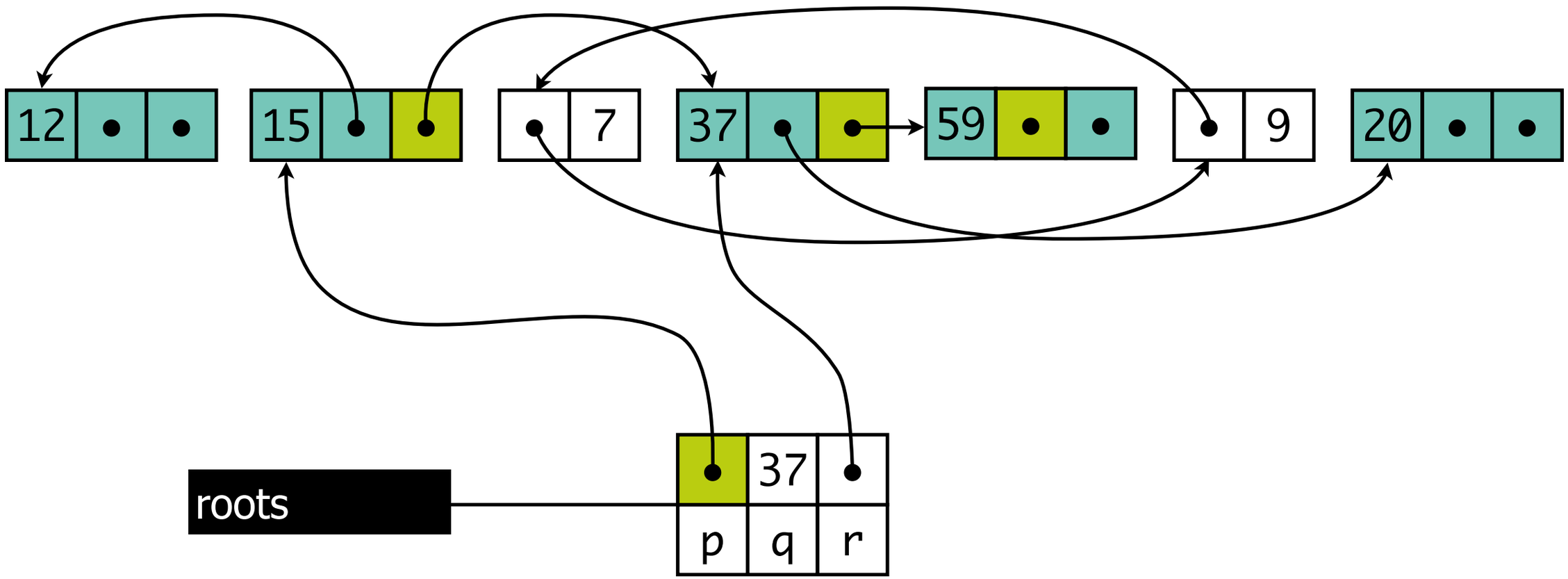
Marking
example



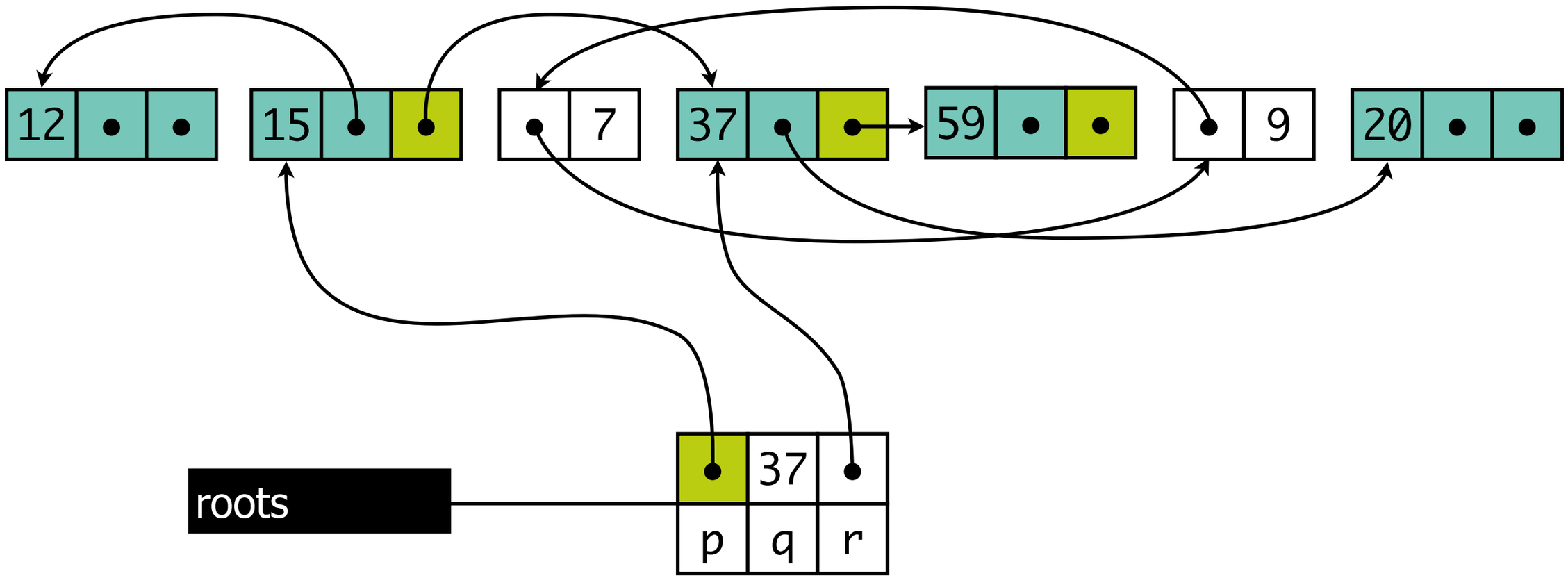
Marking
example



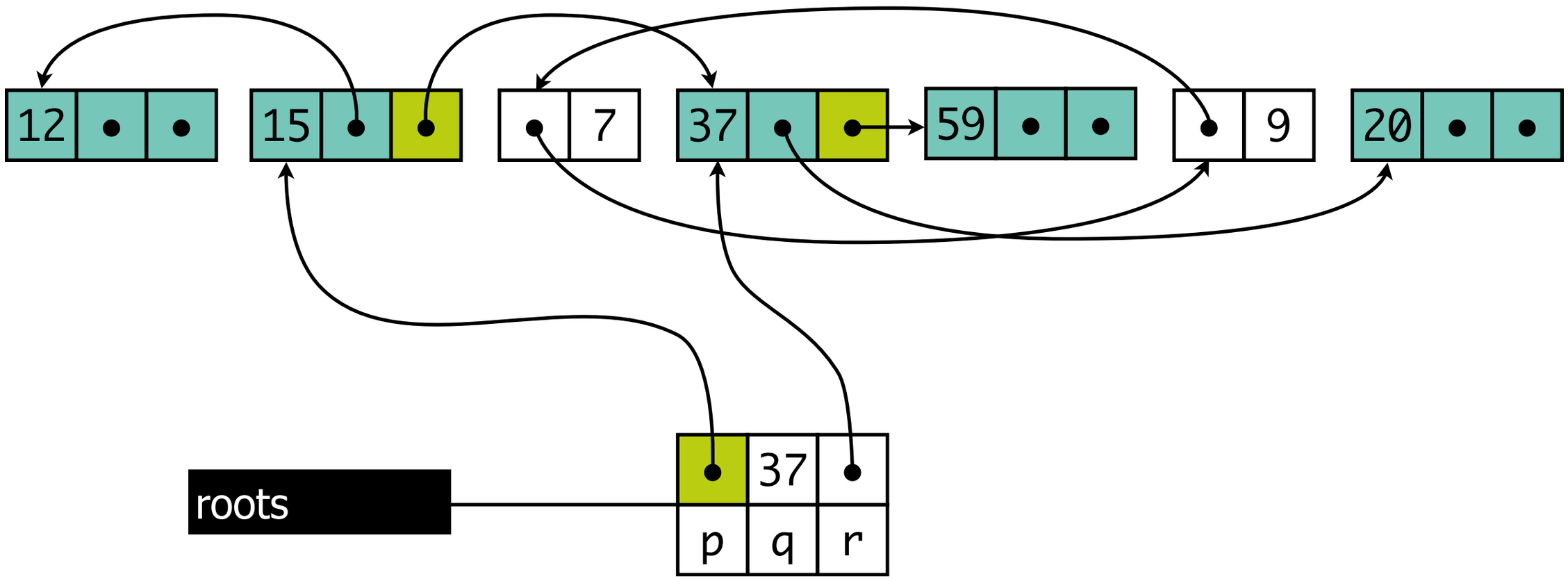
Marking
example



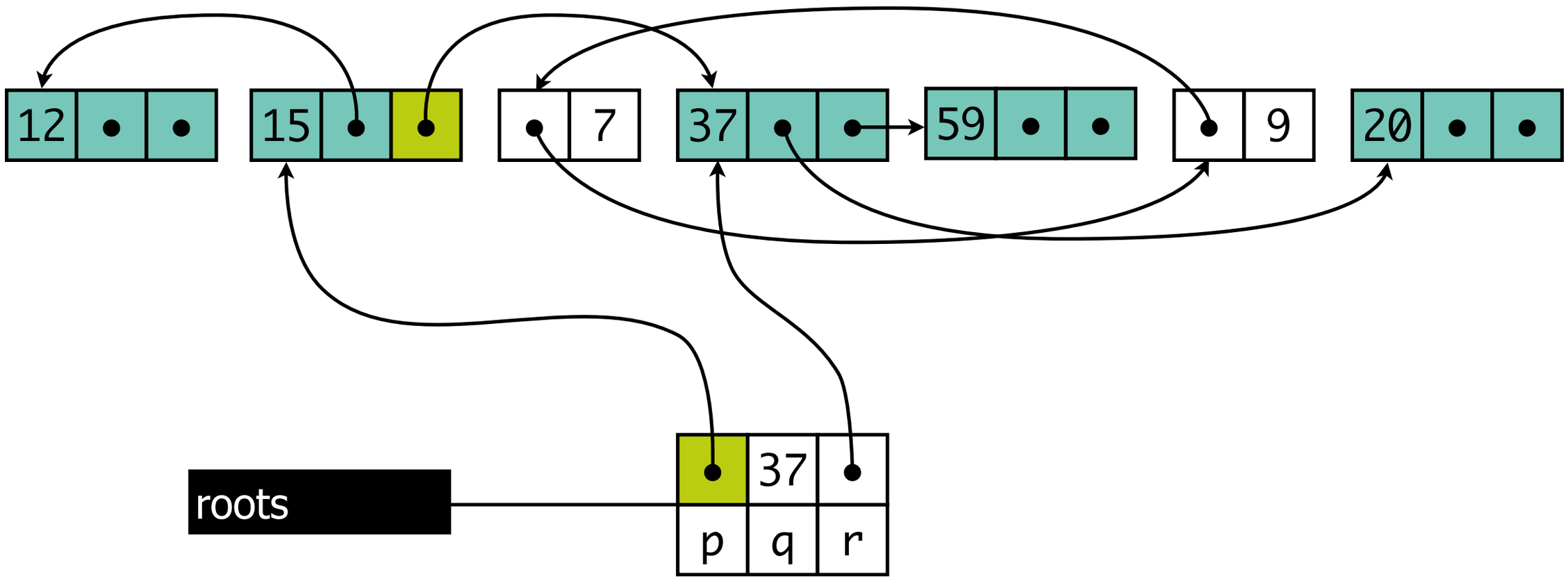
Marking
example



Marking
example

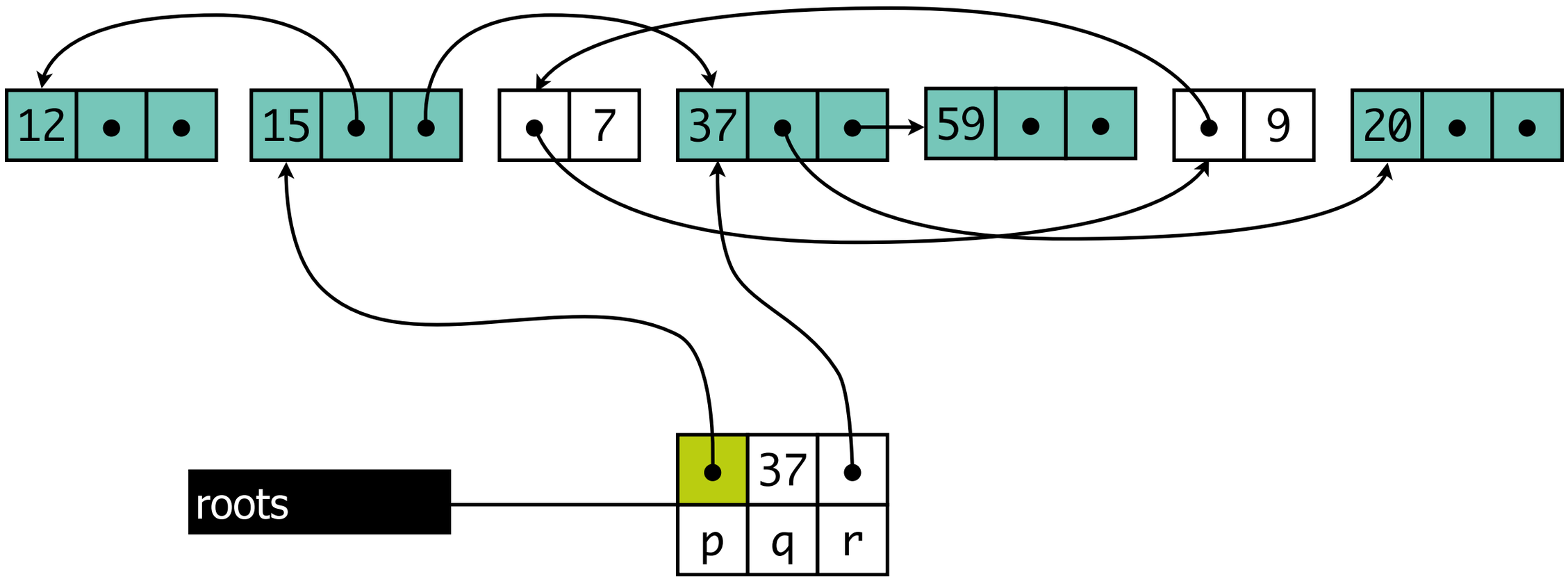


Marking
example



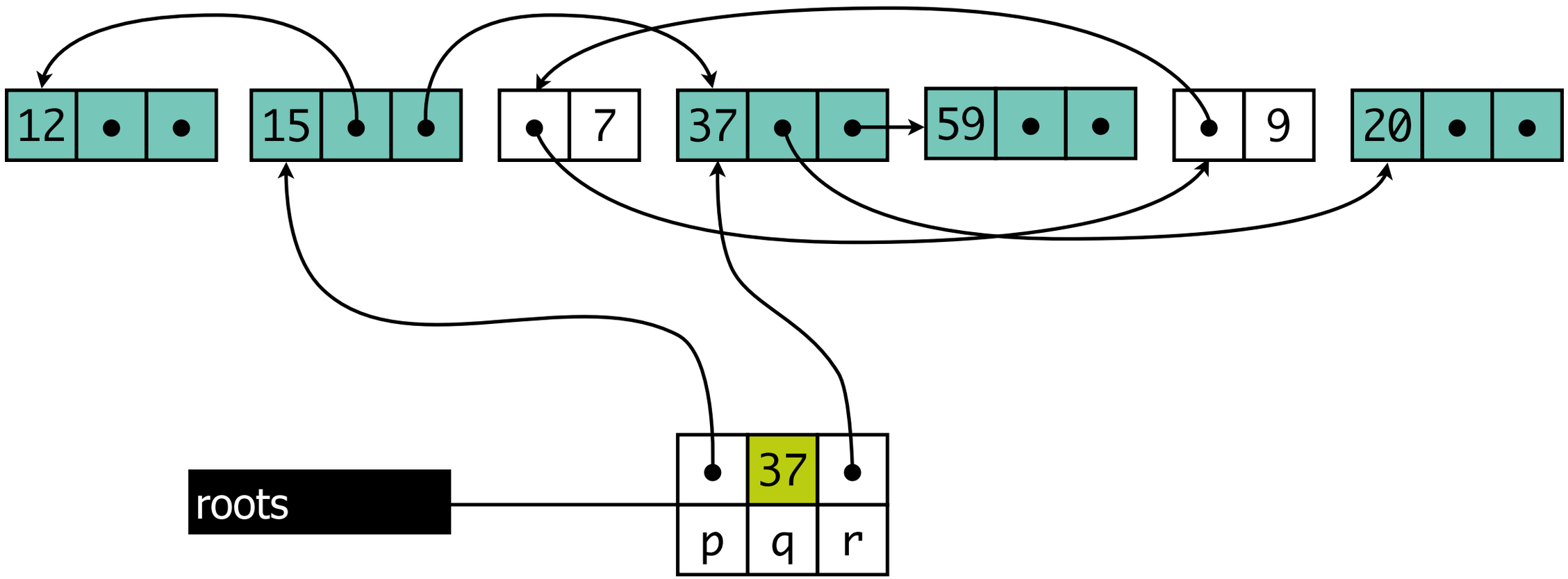
Marking

example



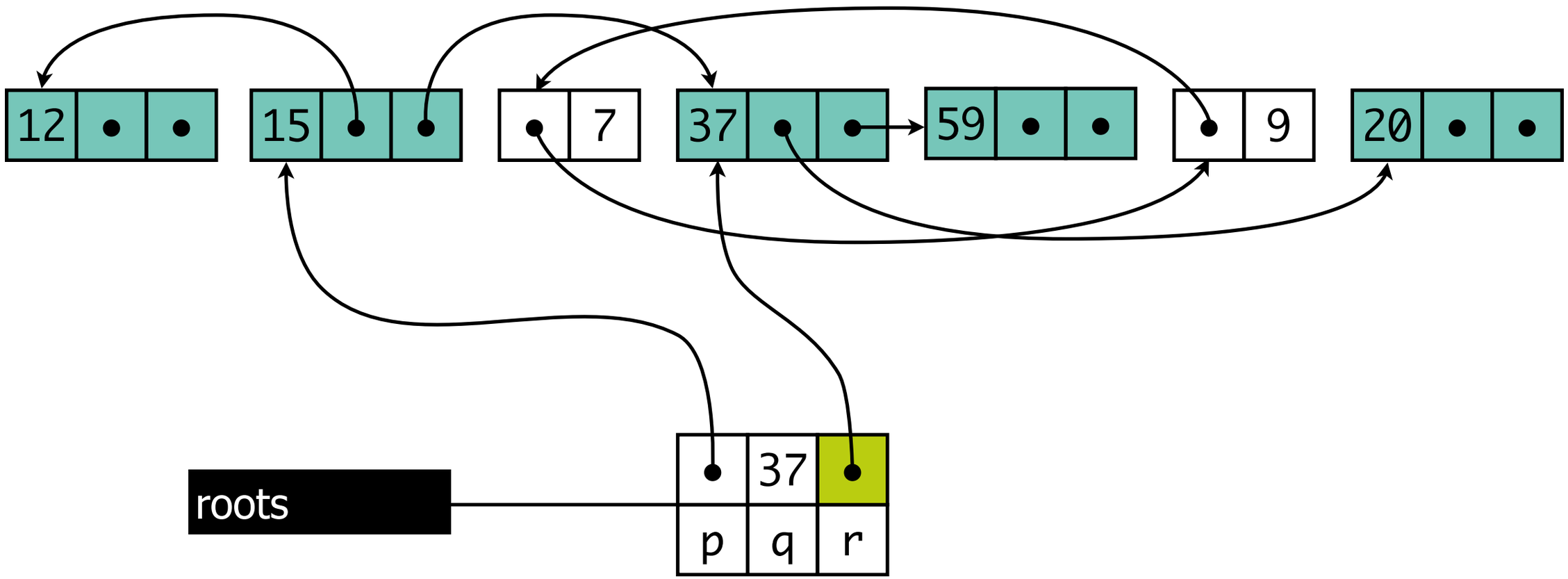
Marking

example

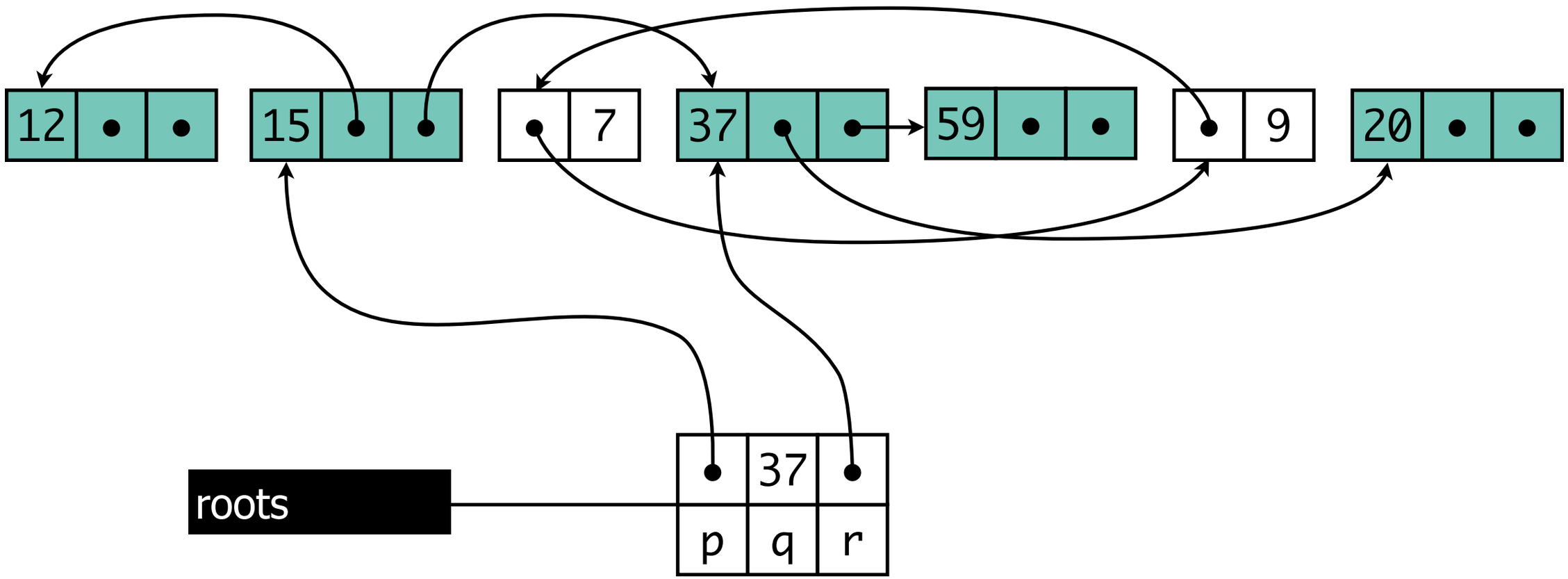


Marking

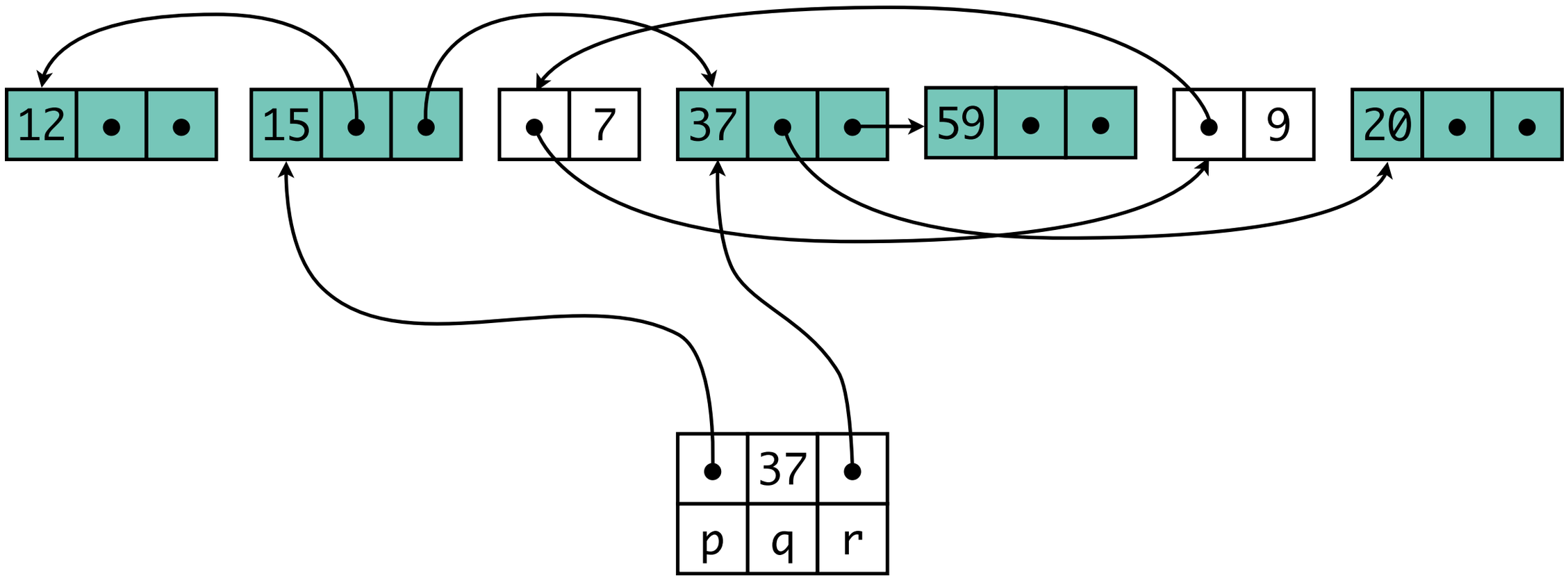
example



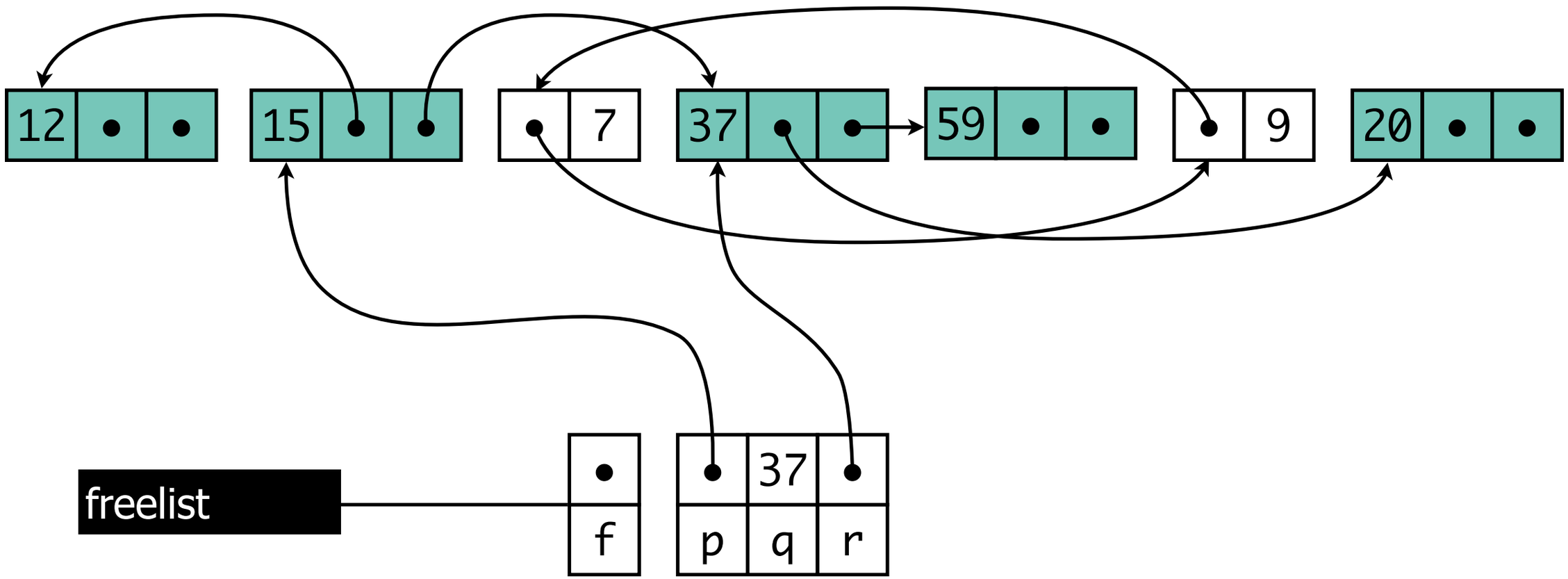
Marking
example



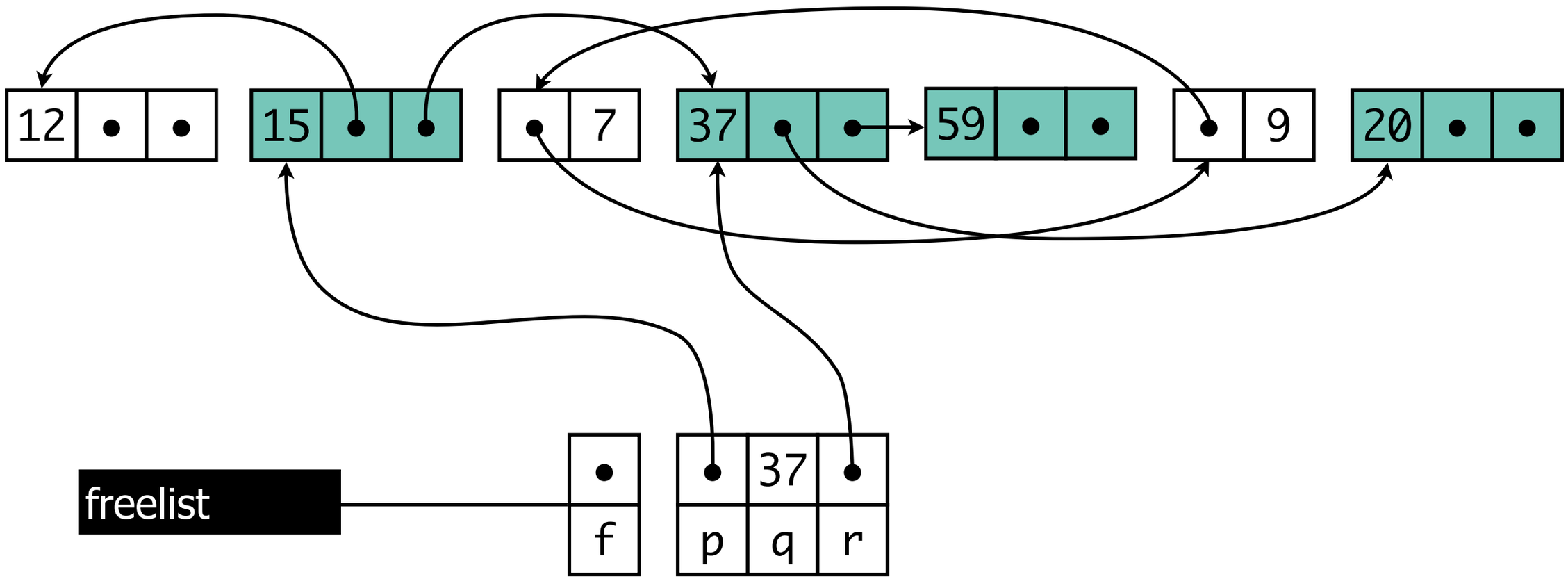
Sweeping
example



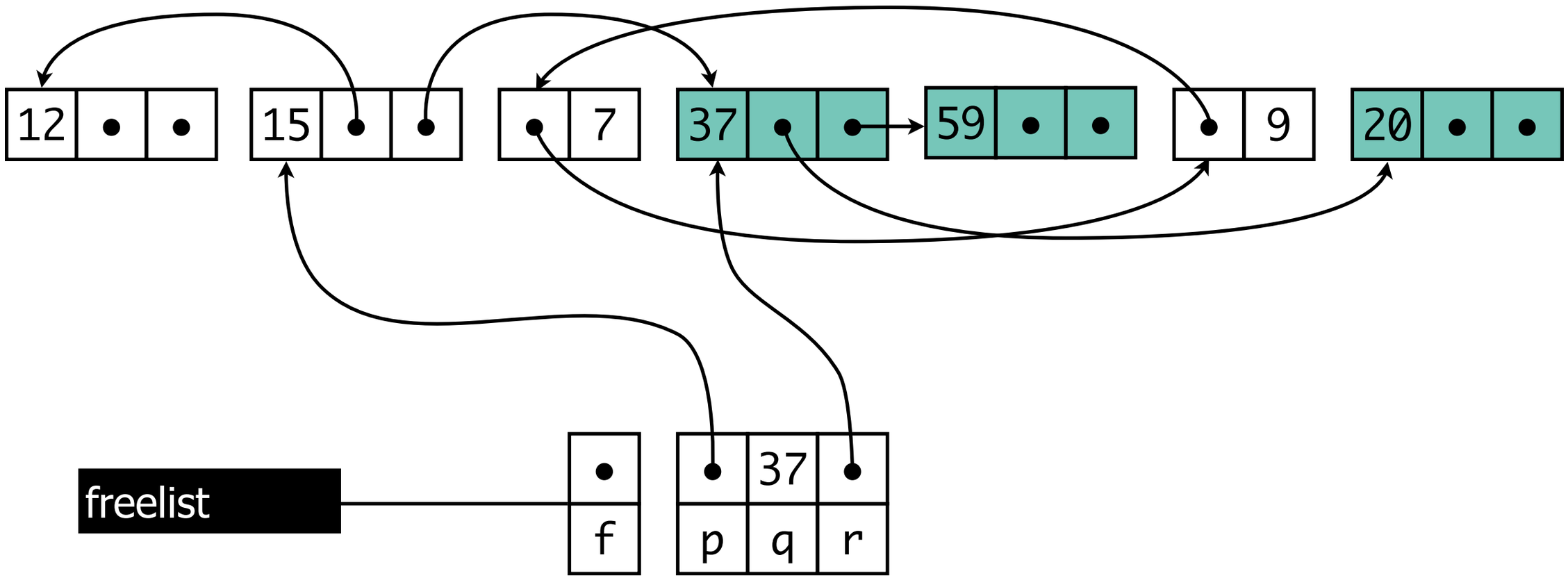
Sweeping
example



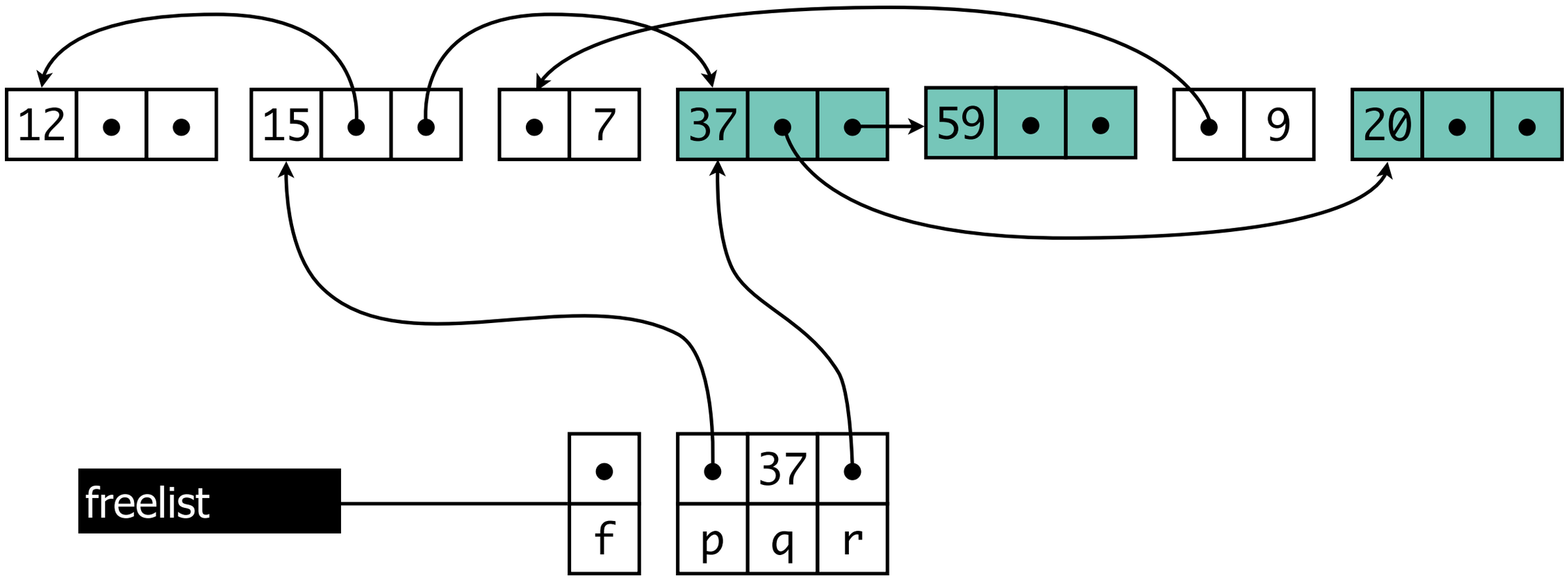
Sweeping
example



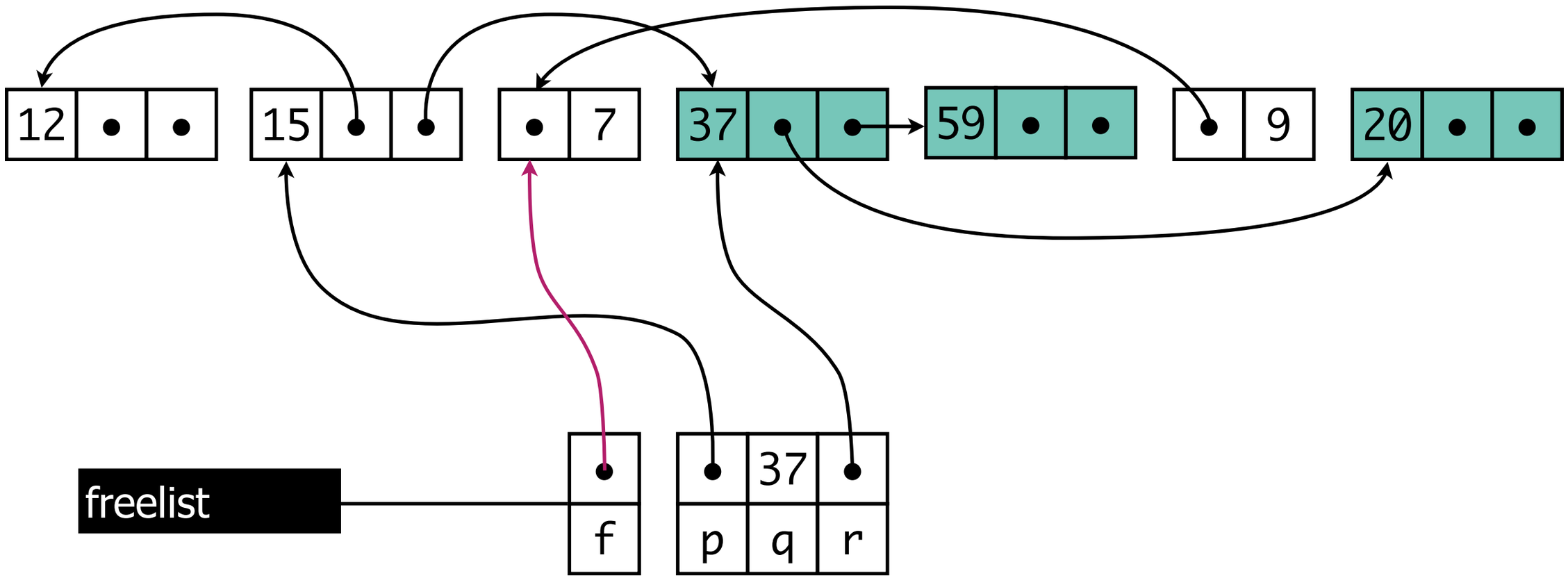
Sweeping
example



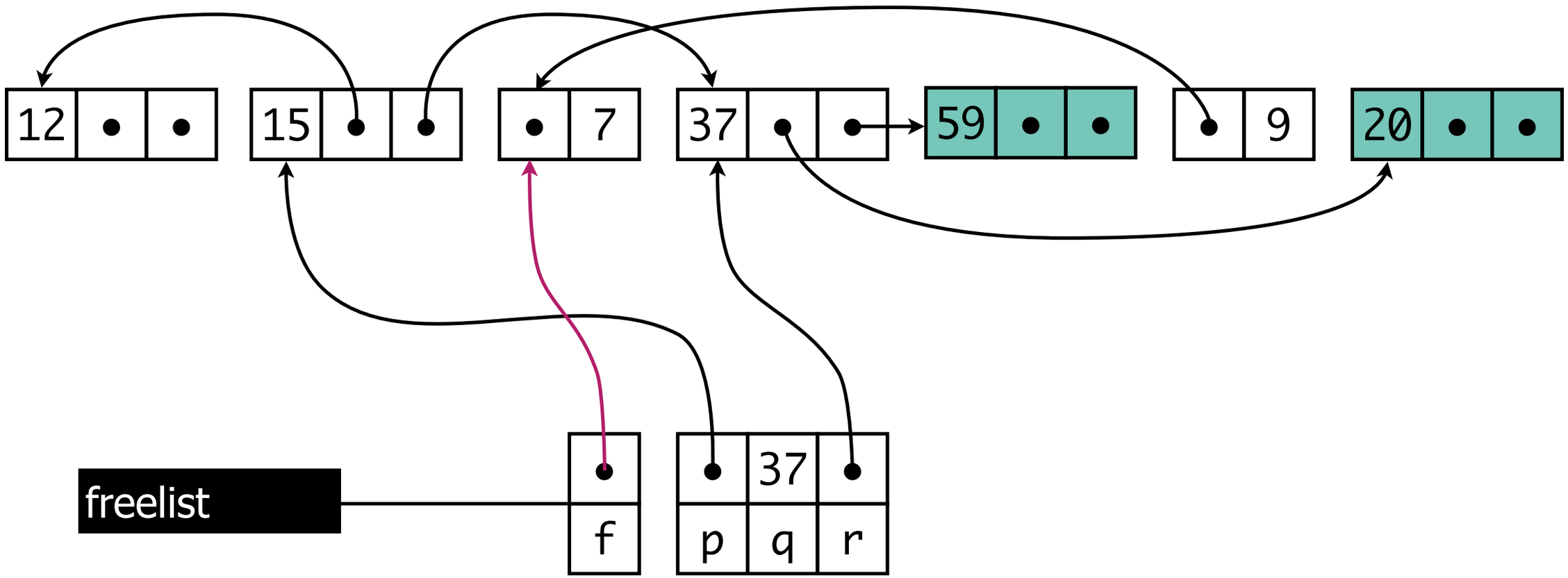
Sweeping
example



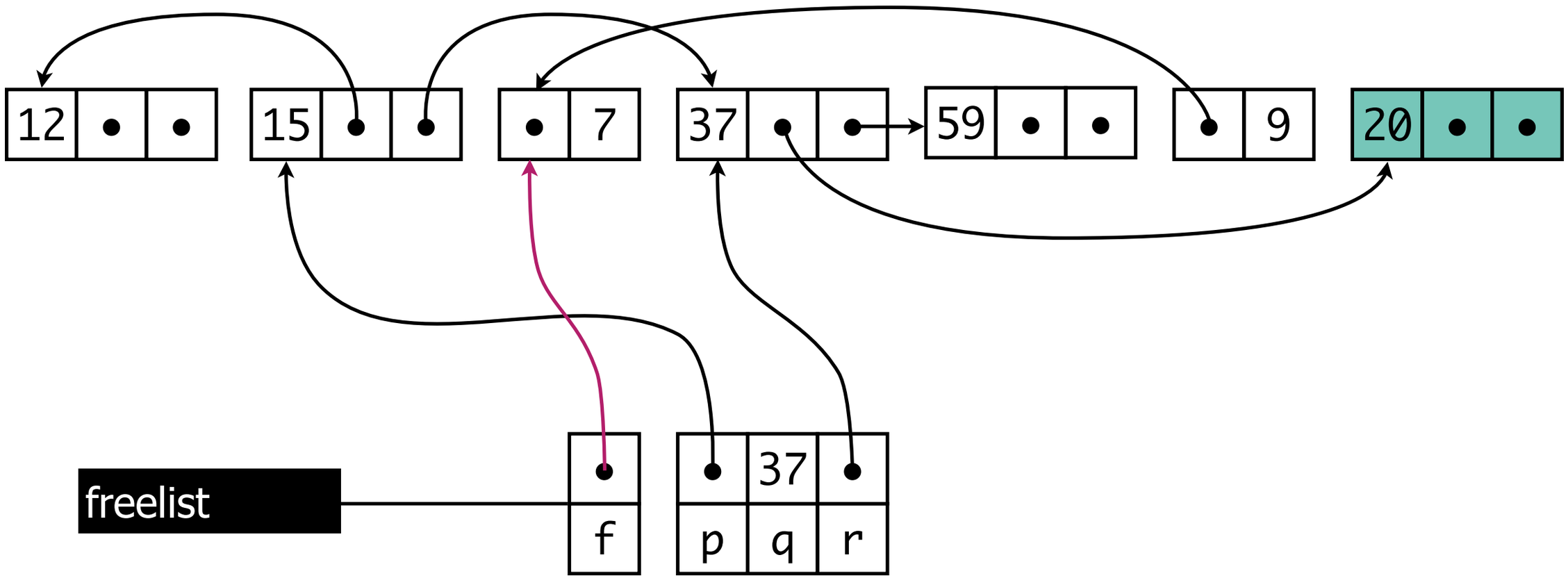
Sweeping
example



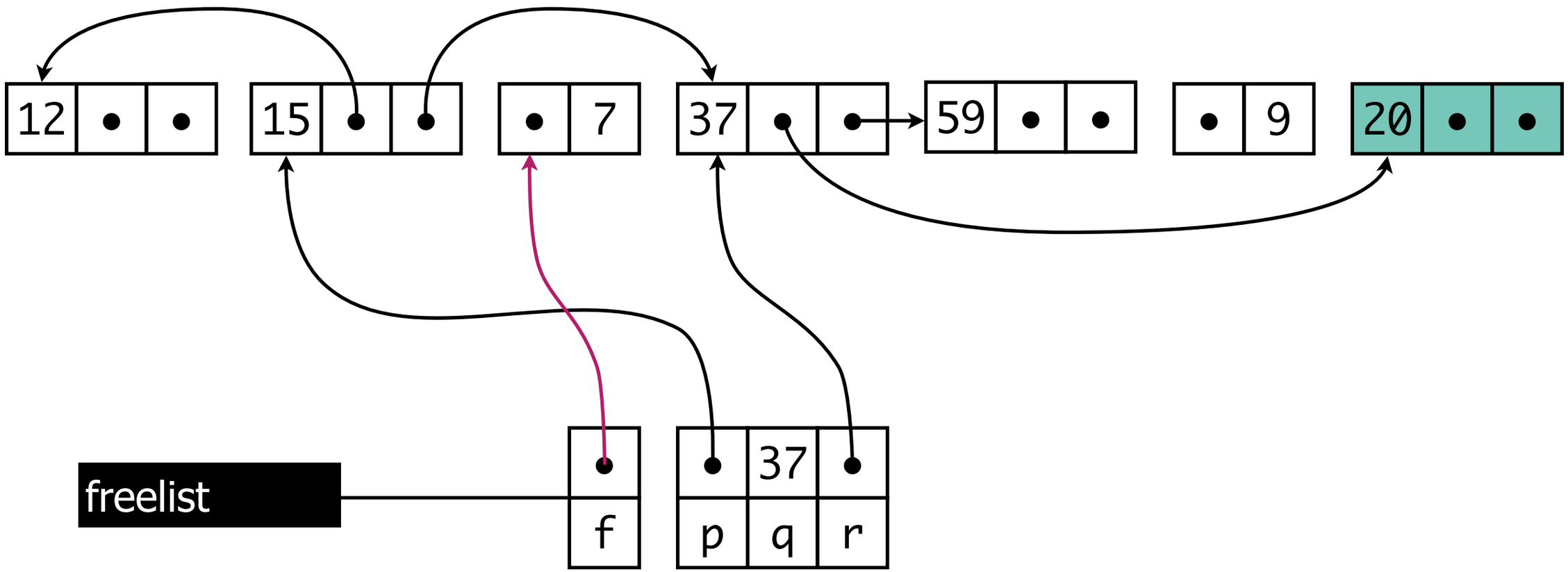
Sweeping
example



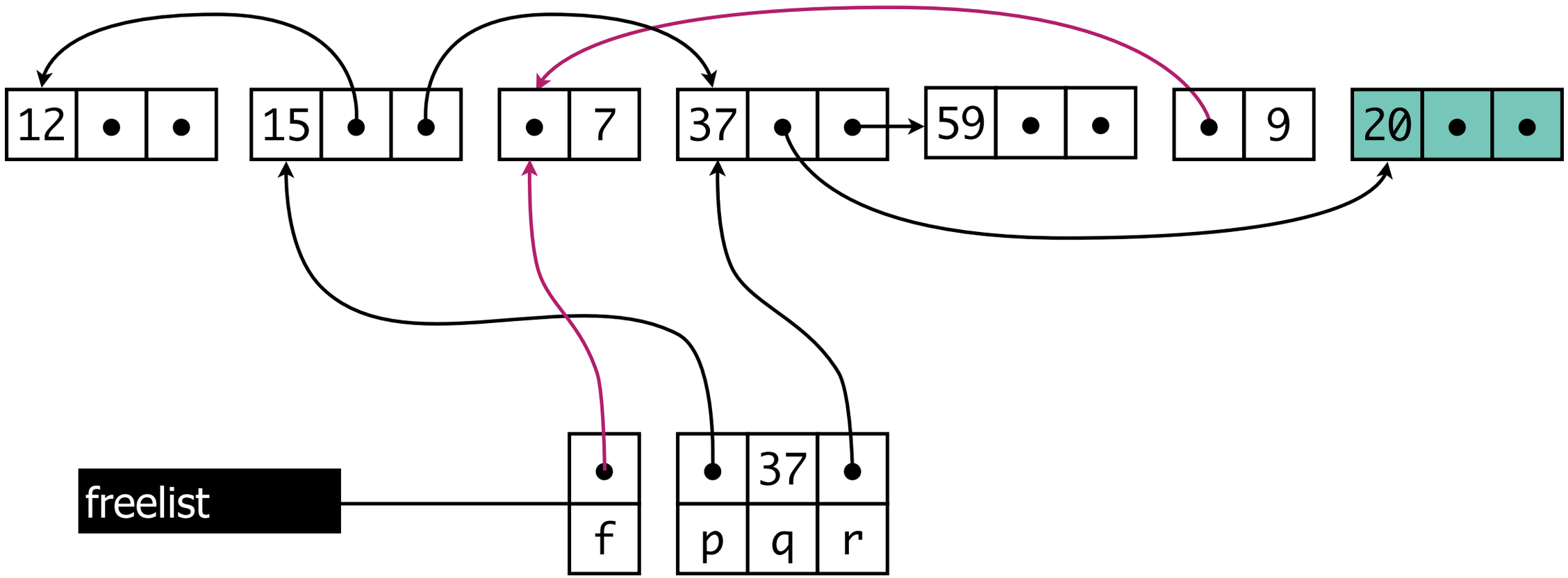
Sweeping
example



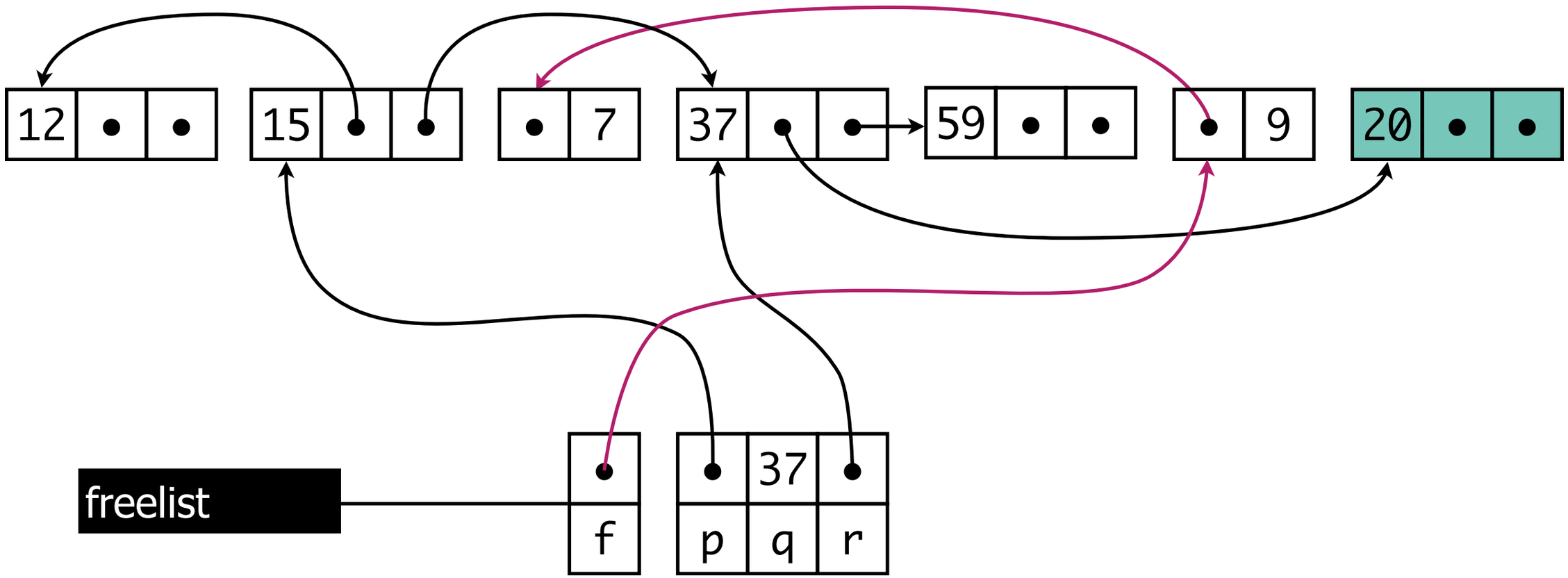
Sweeping
example



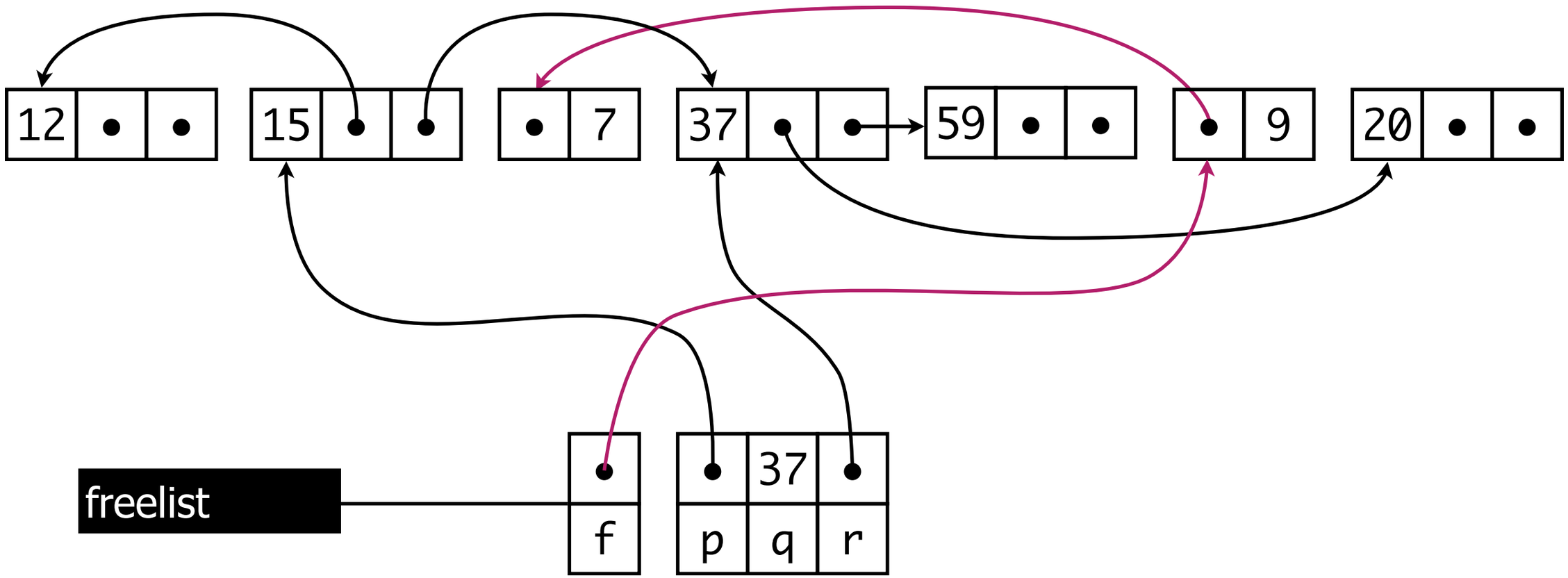
Sweeping
example



Sweeping
example



Sweeping
example



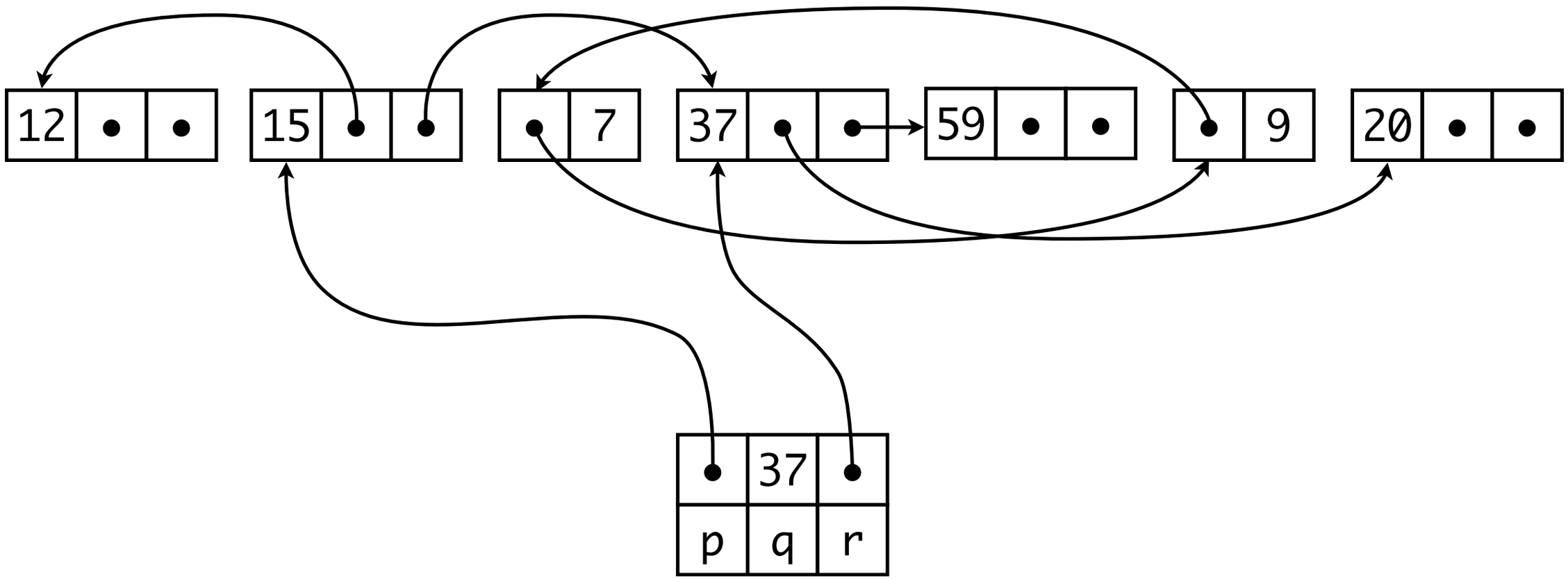
Marking
algorithms

```
function DFS(x)

  if pointer(x) & !x.marked

    x.marked = true

    foreach f in fields(x)

      DFS(f)
```

```
function DFS(x)

  if pointer(x) & !x.marked

    x.marked = true
    t = 1 ; stack[t] = x

    while t > 0

      x = stack[t] ; t = t - 1

      foreach f in fields(x)
        if pointer(f) & !f.marked

          f.marked = true
          t = t + 1 ; stack[t] = f
```

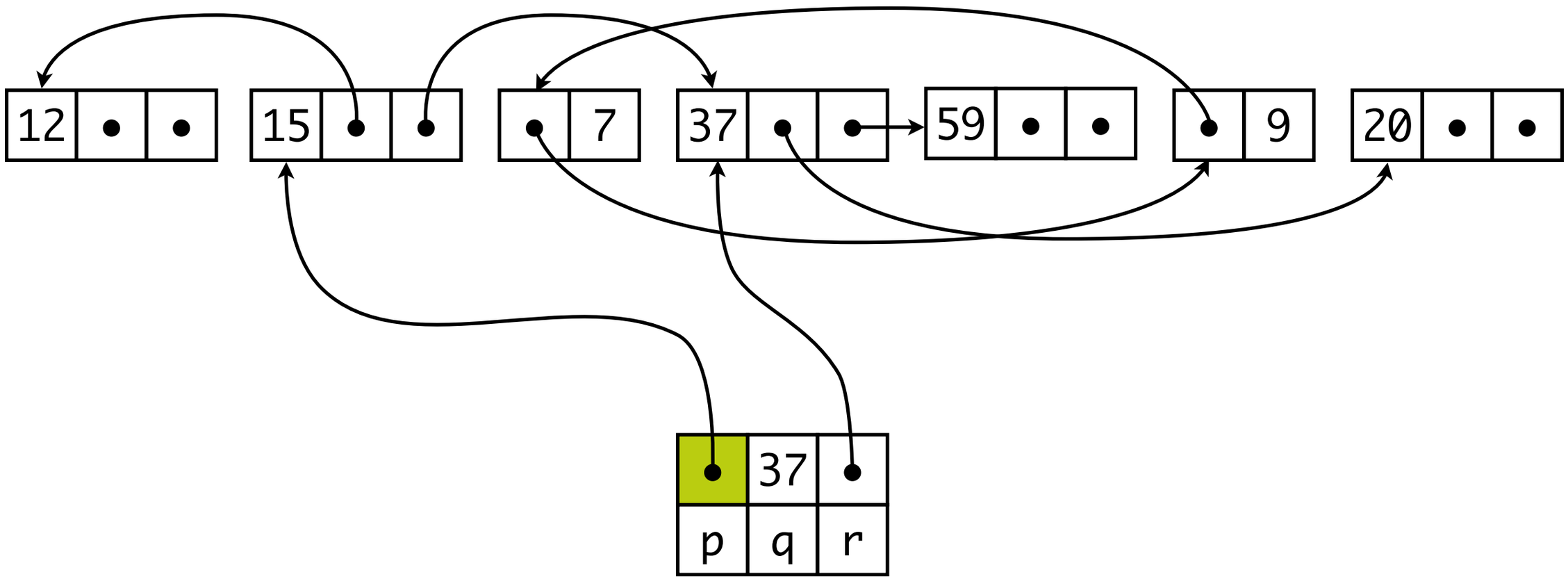
Recap: Marking

pointer reversals



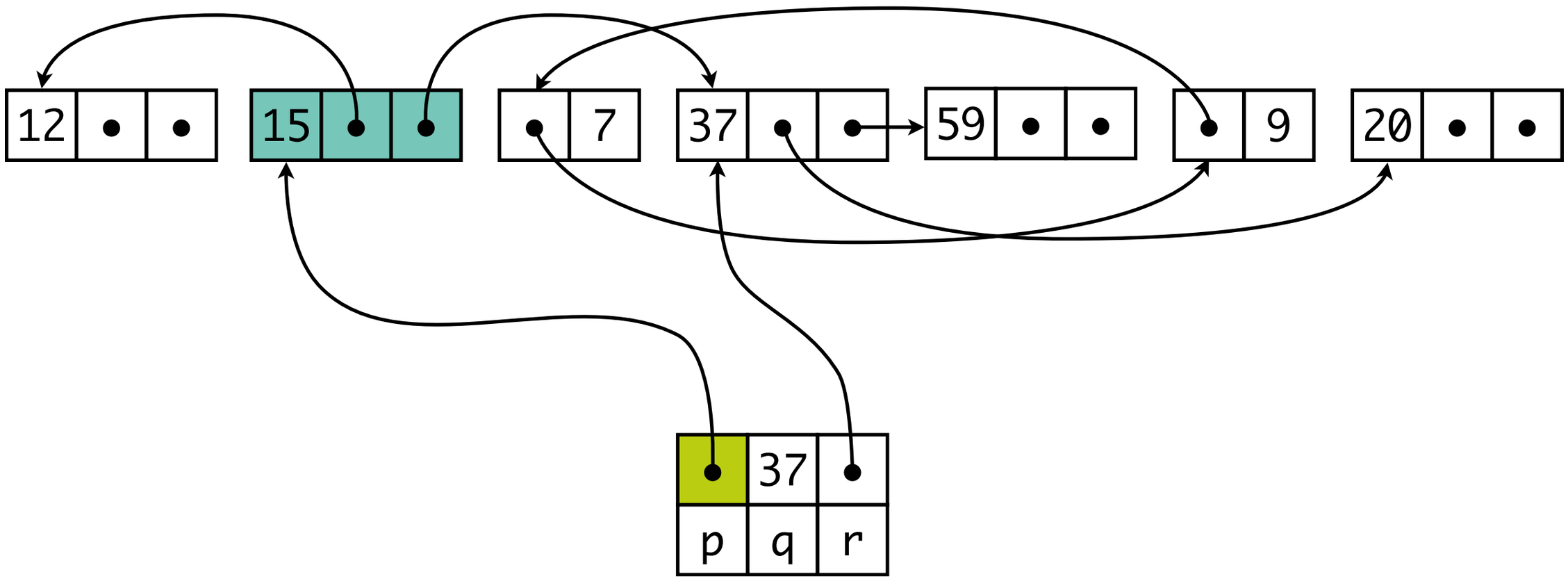
Recap: Marking

pointer reversals



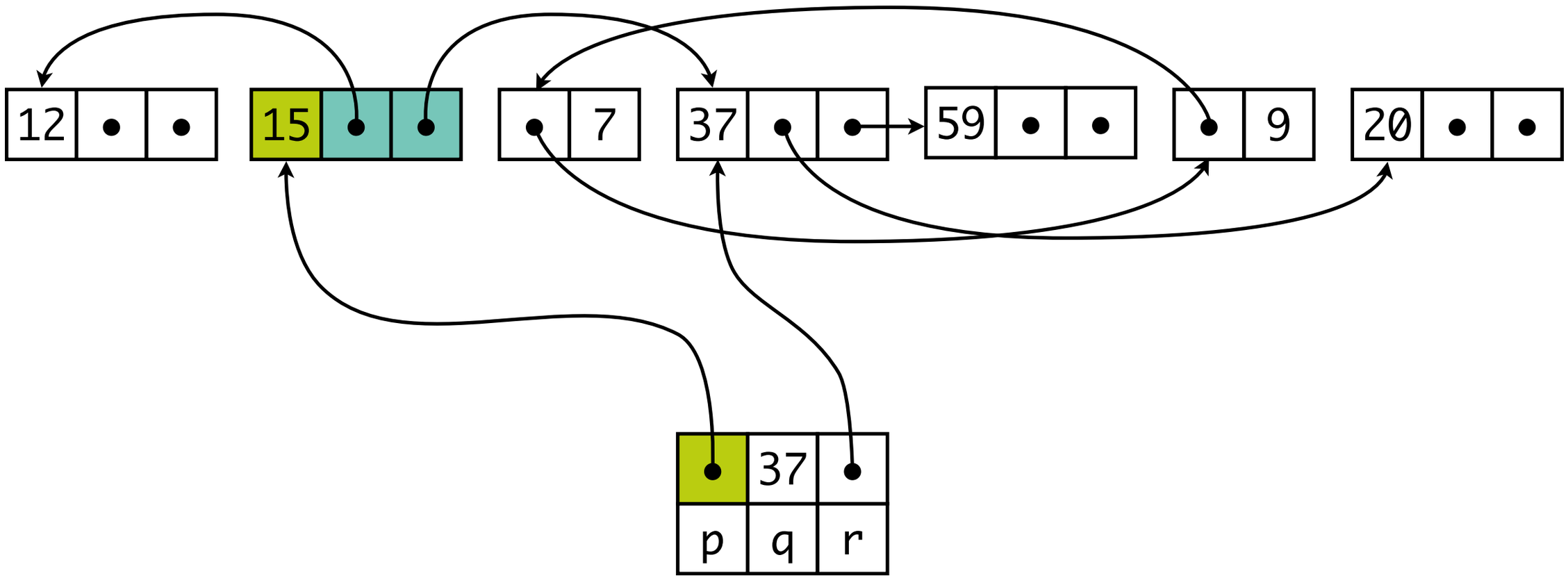
Recap: Marking

pointer reversals



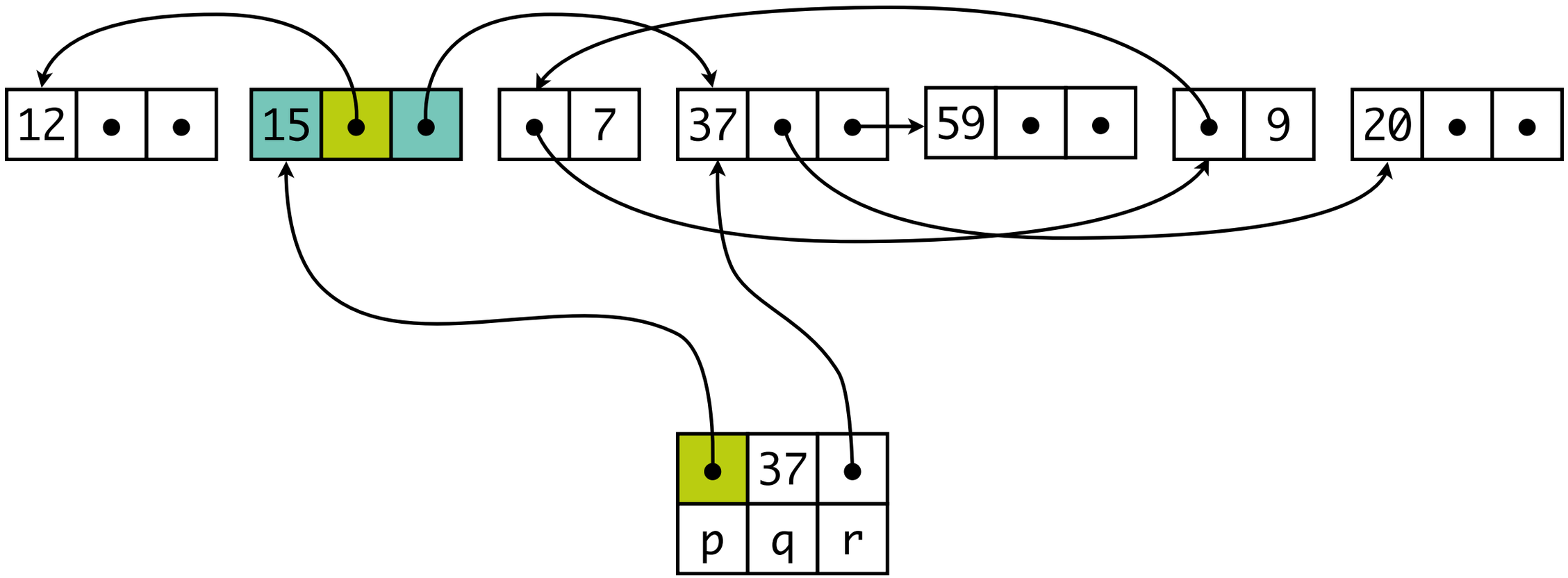
Recap: Marking

pointer reversals



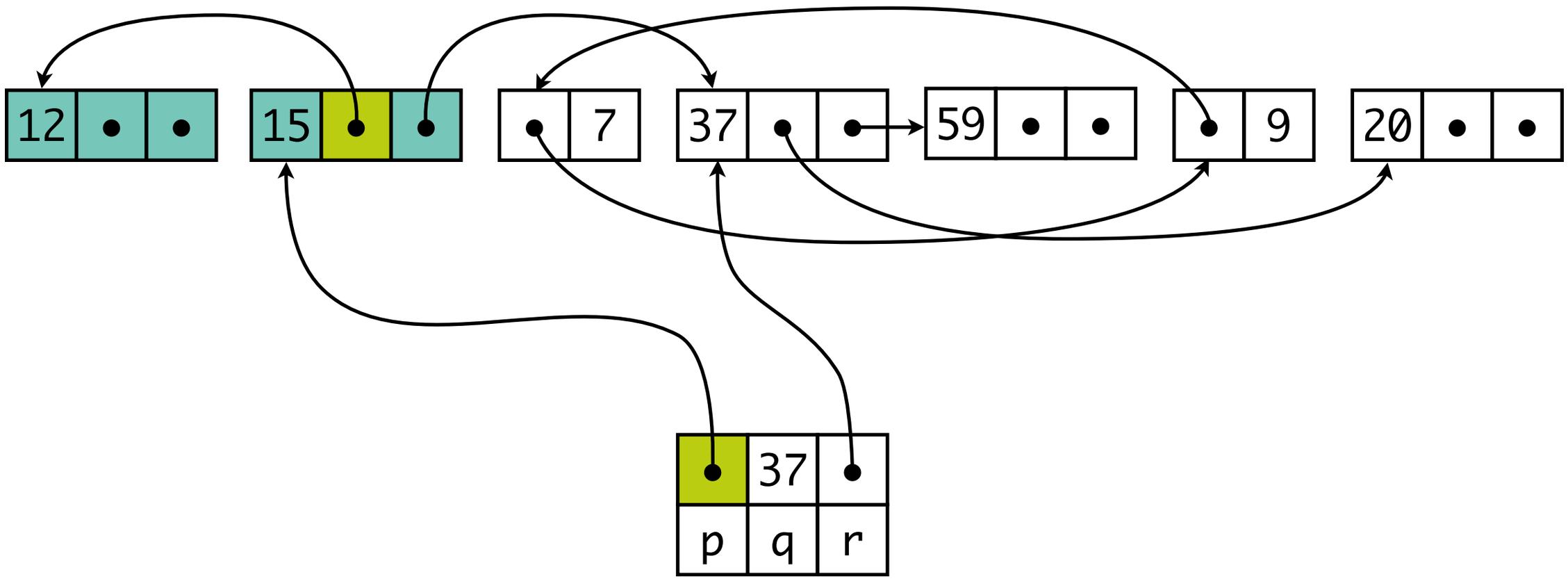
Recap: Marking

pointer reversals



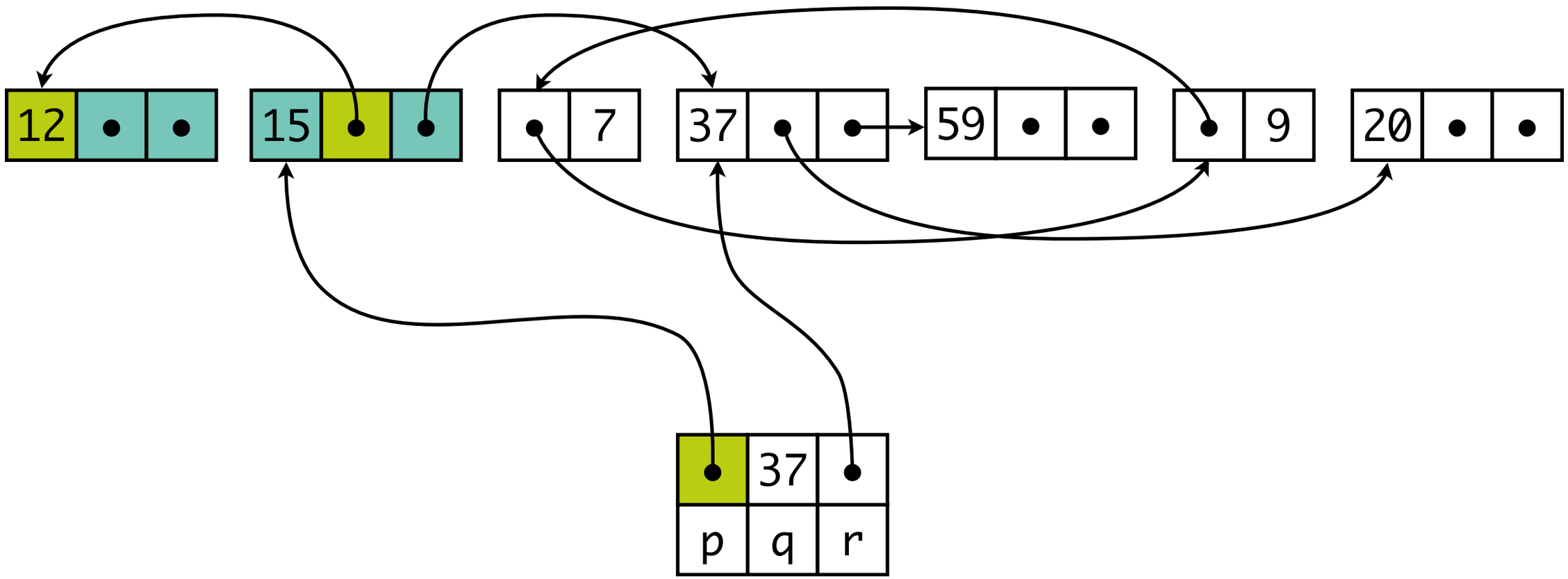
Recap: Marking

pointer reversals



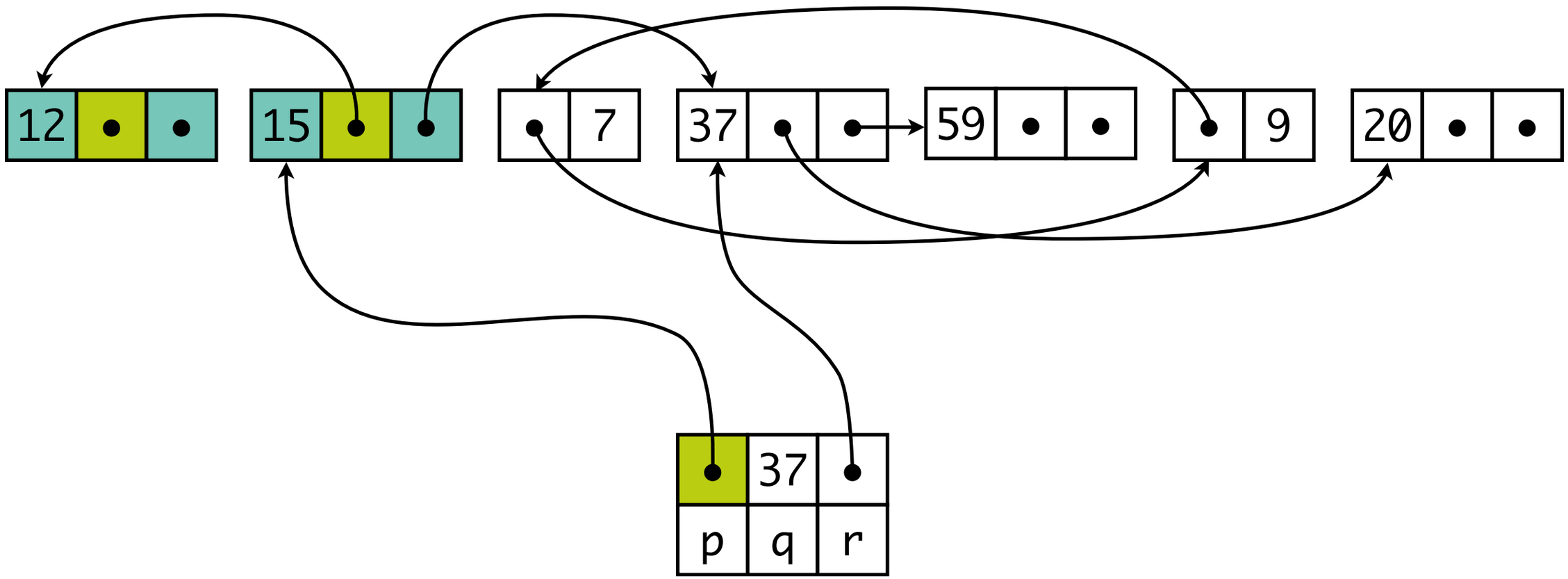
Recap: Marking

pointer reversals



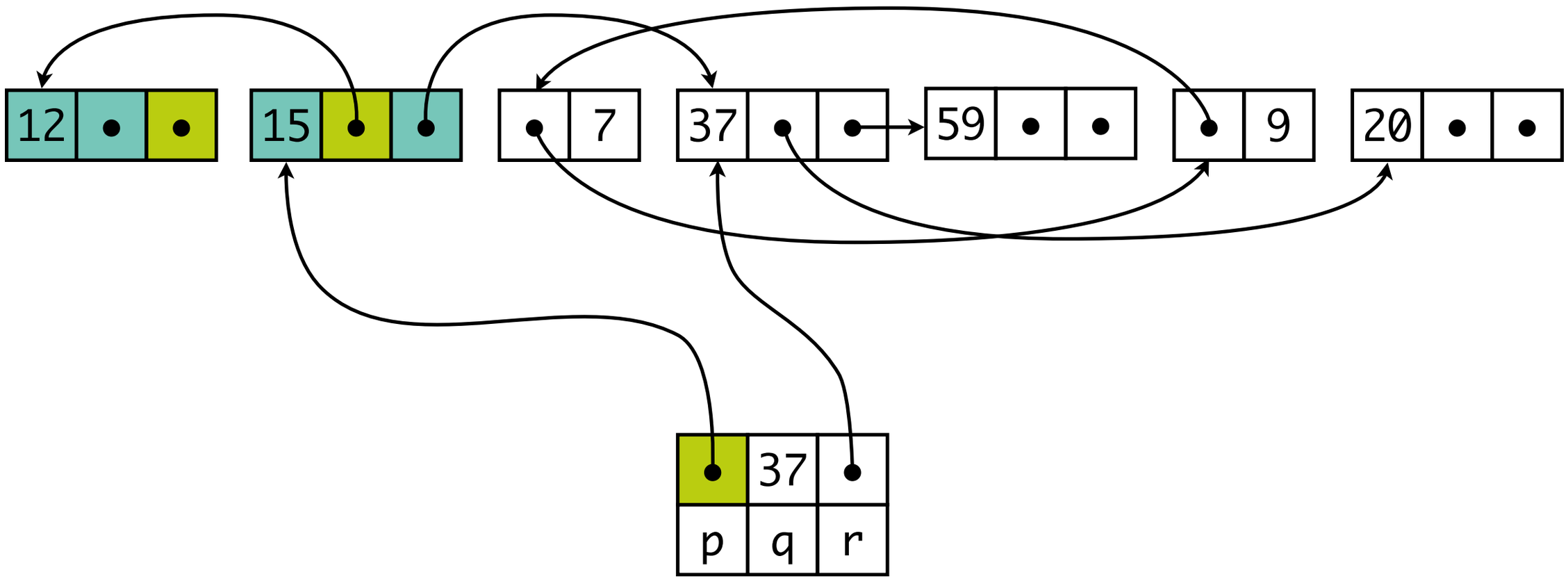
Recap: Marking

pointer reversals



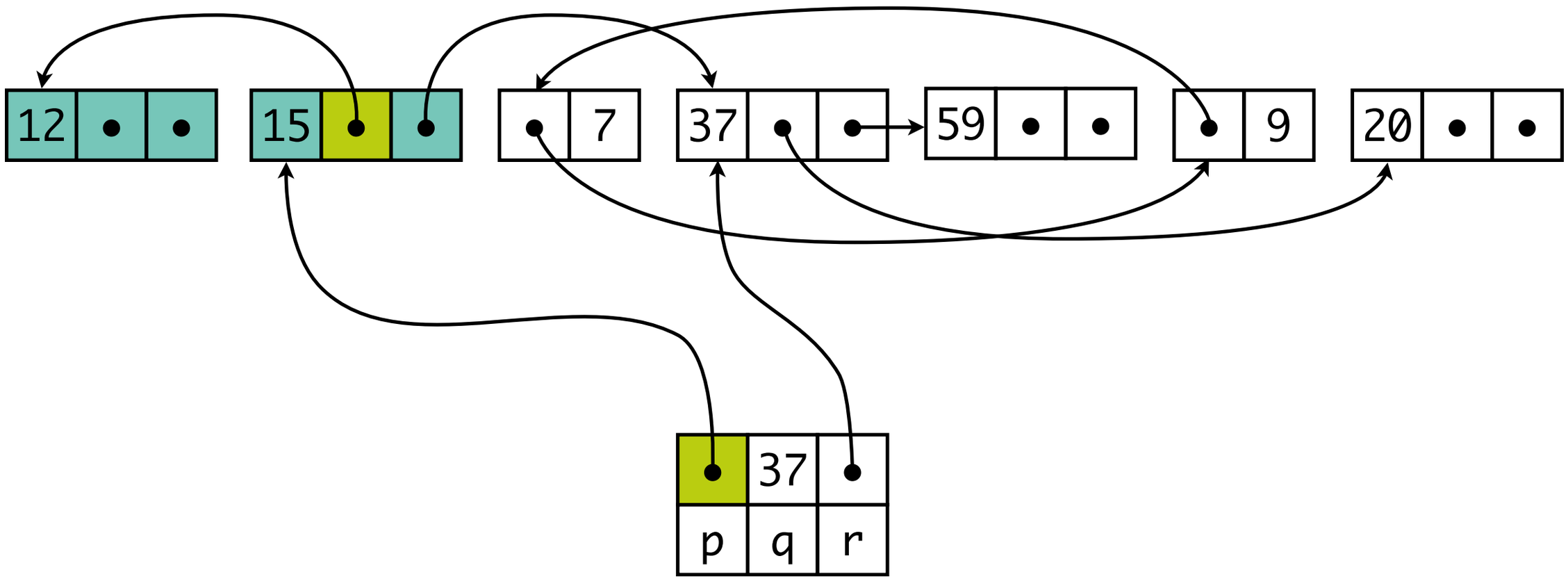
Recap: Marking

pointer reversals



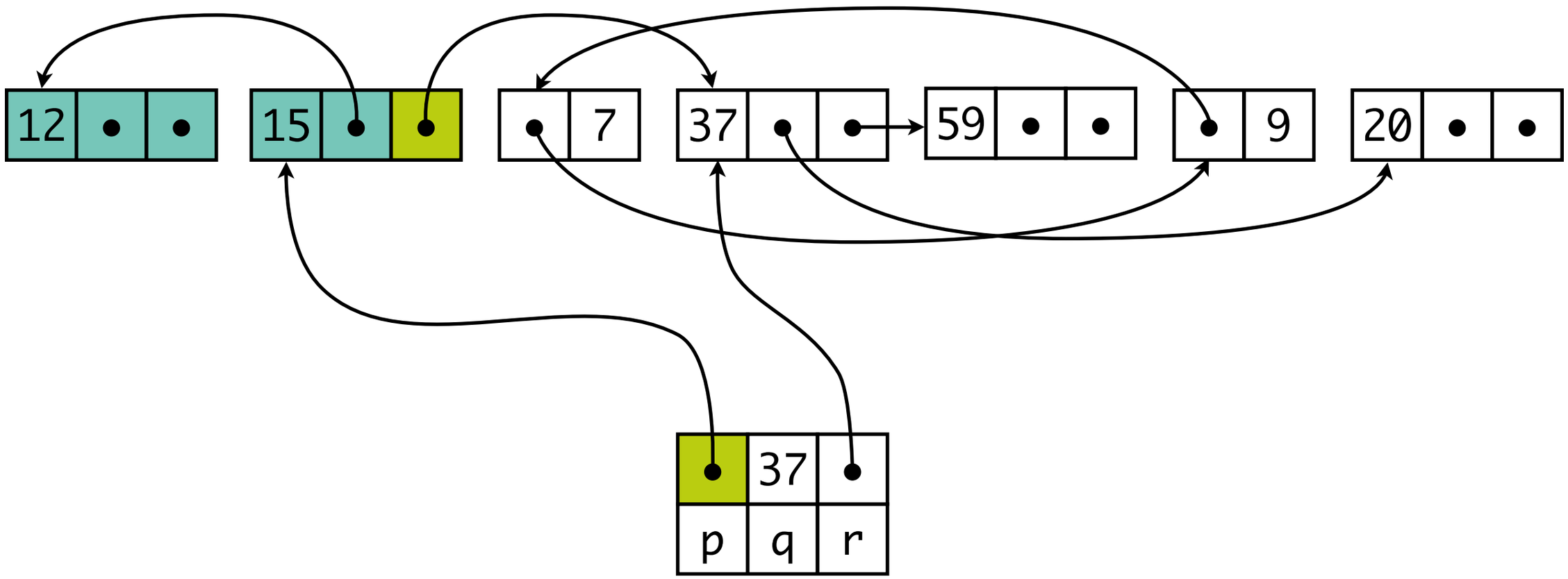
Recap: Marking

pointer reversals



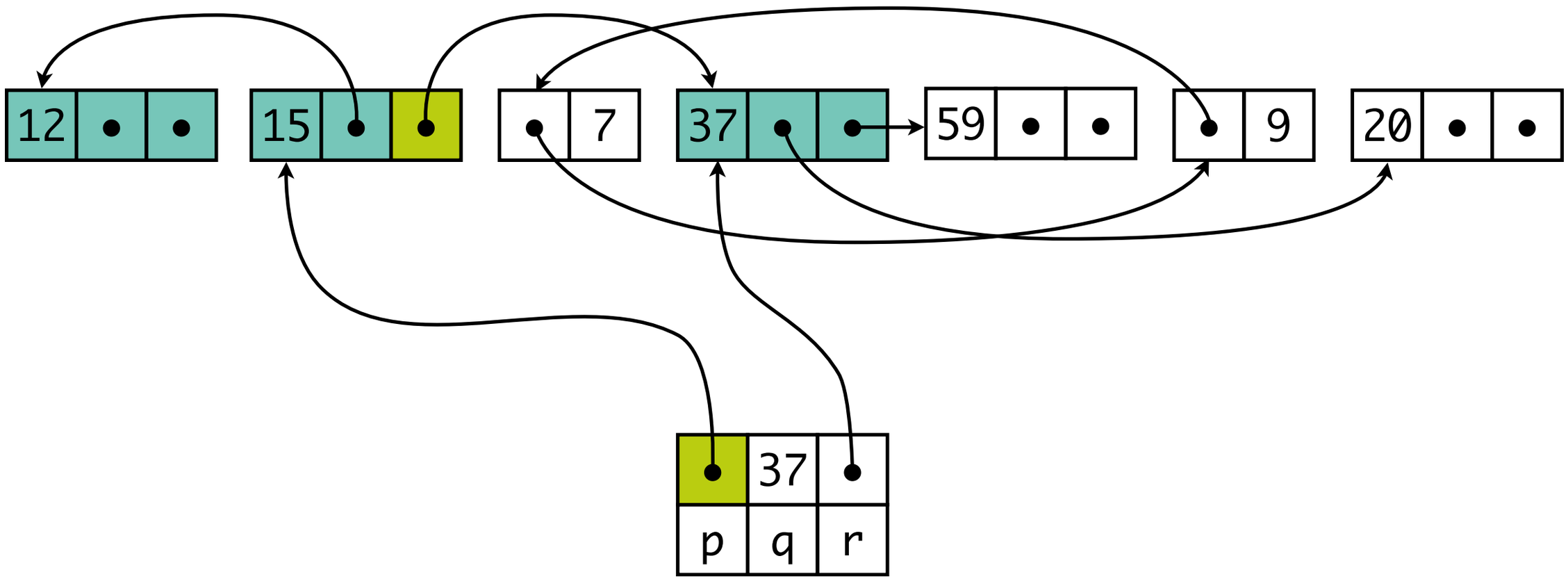
Recap: Marking

pointer reversals



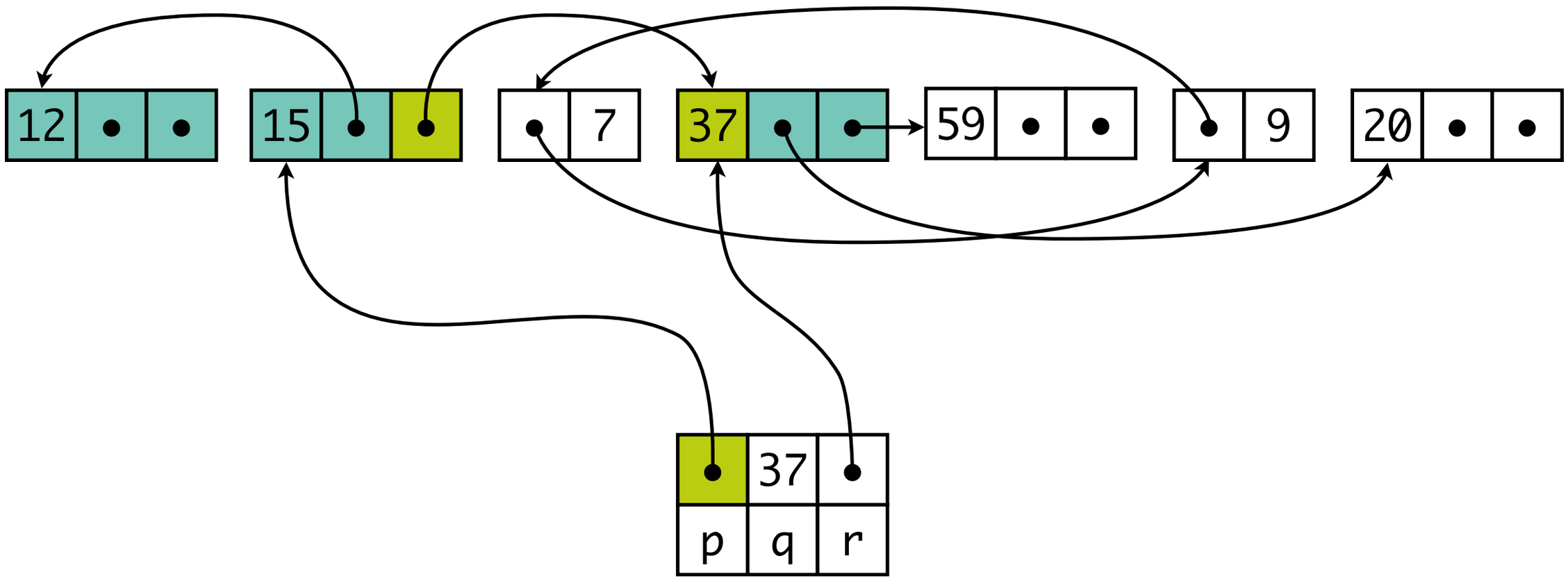
Recap: Marking

pointer reversals



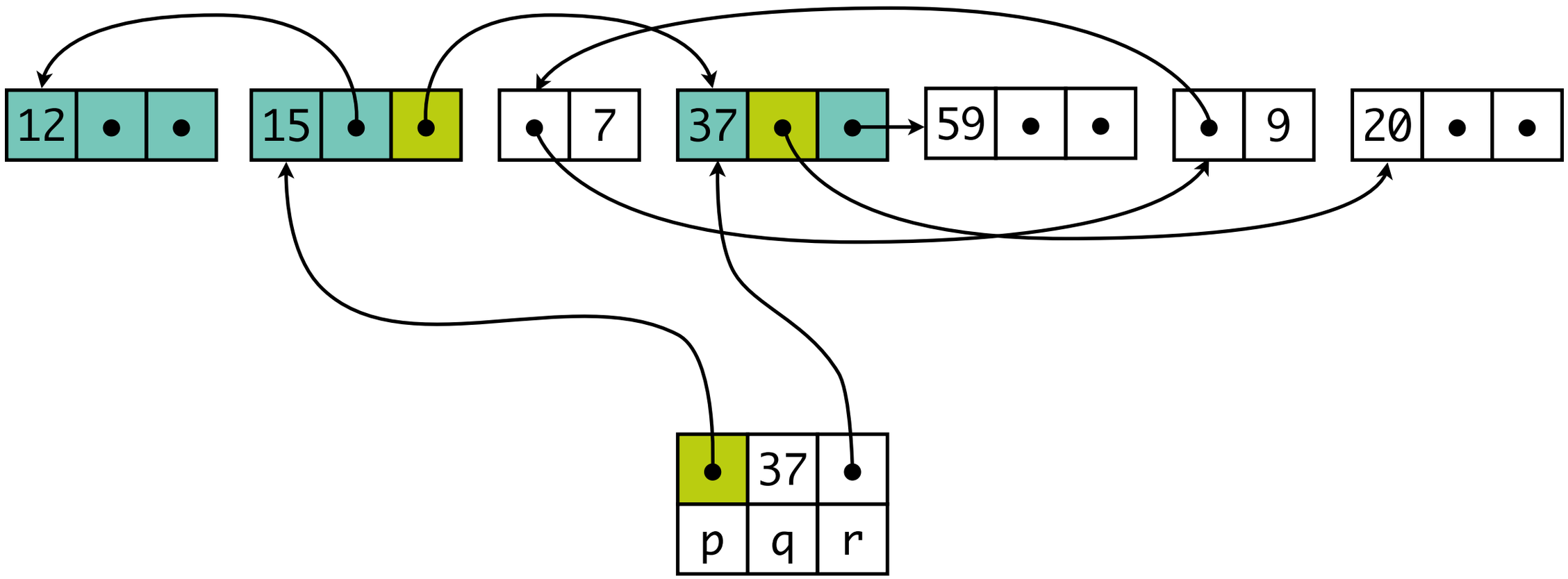
Recap: Marking

pointer reversals



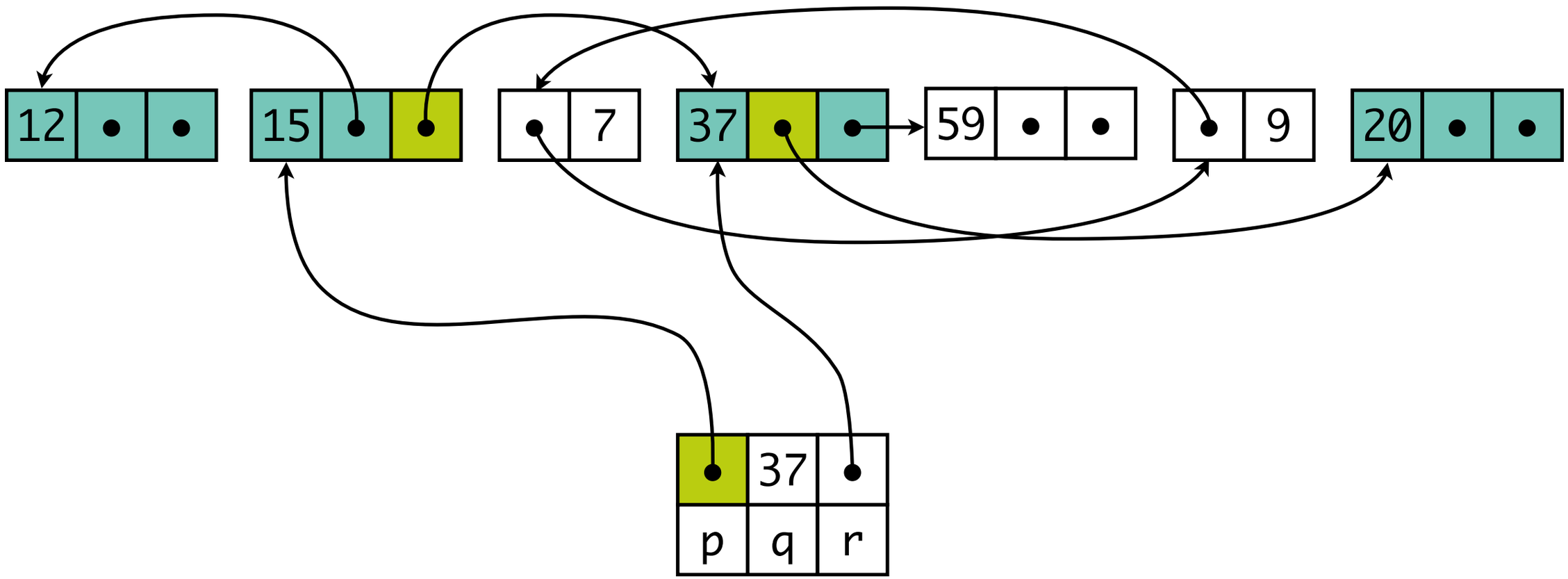
Recap: Marking

pointer reversals



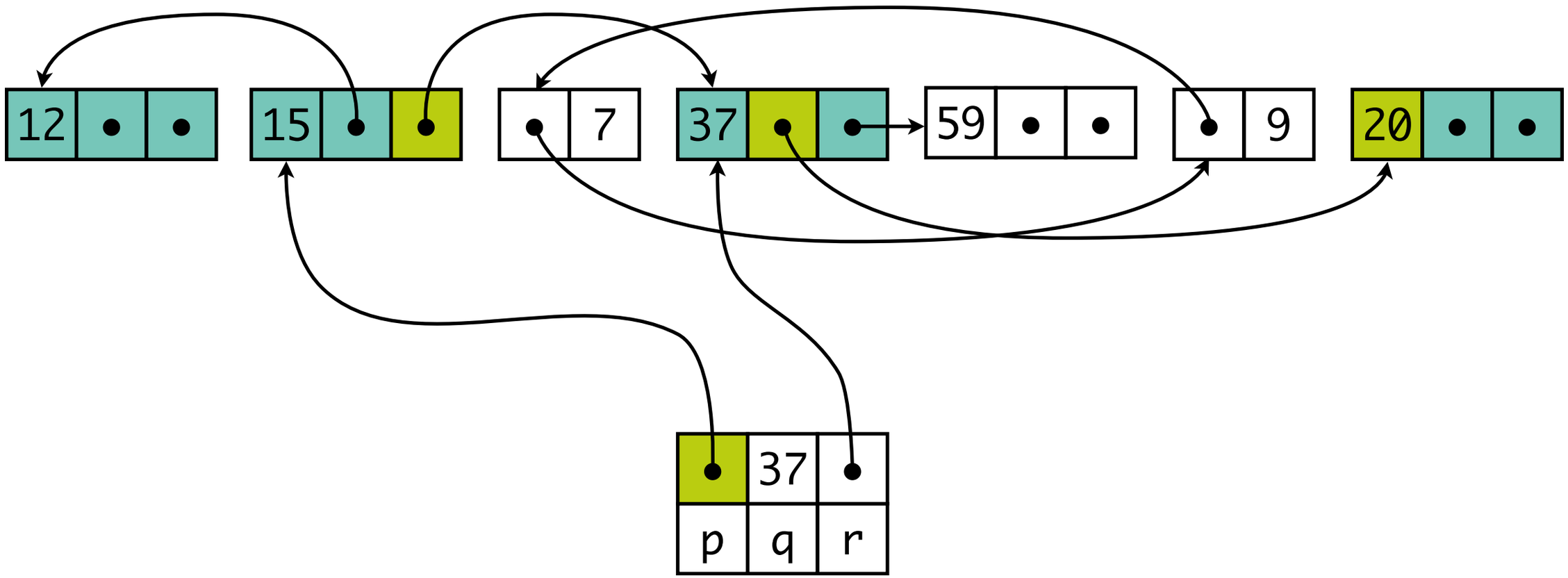
Recap: Marking

pointer reversals



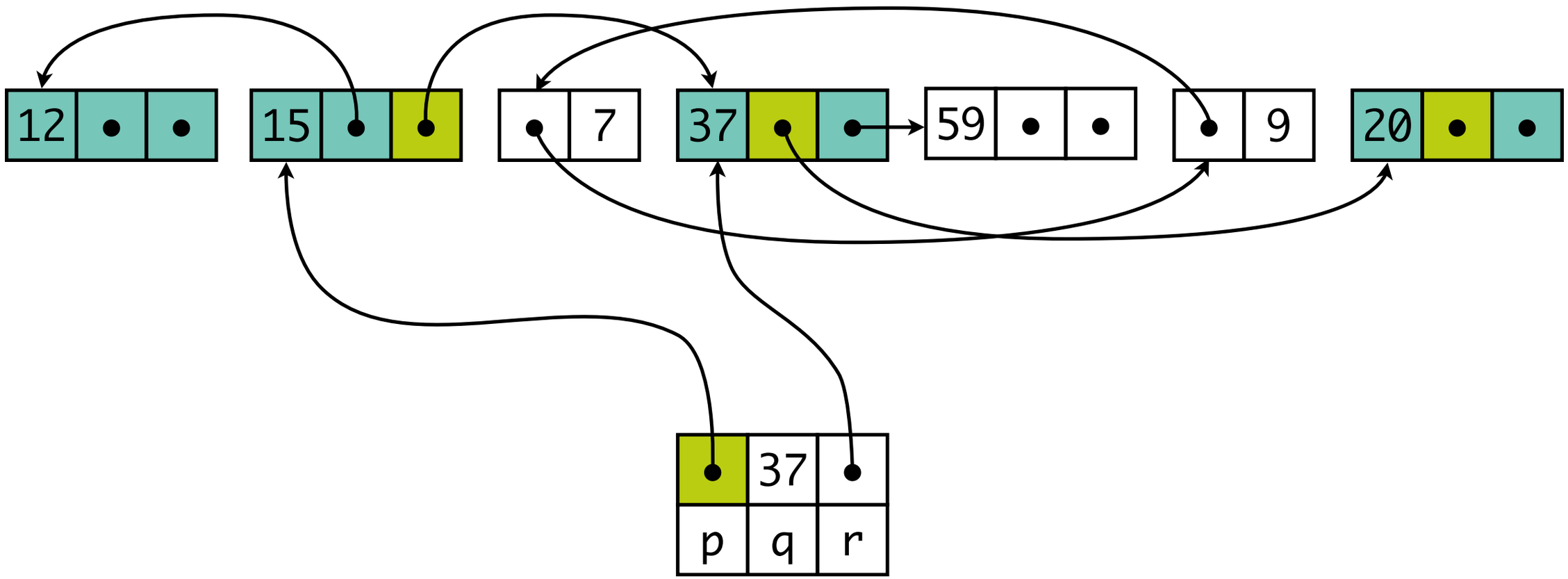
Recap: Marking

pointer reversals



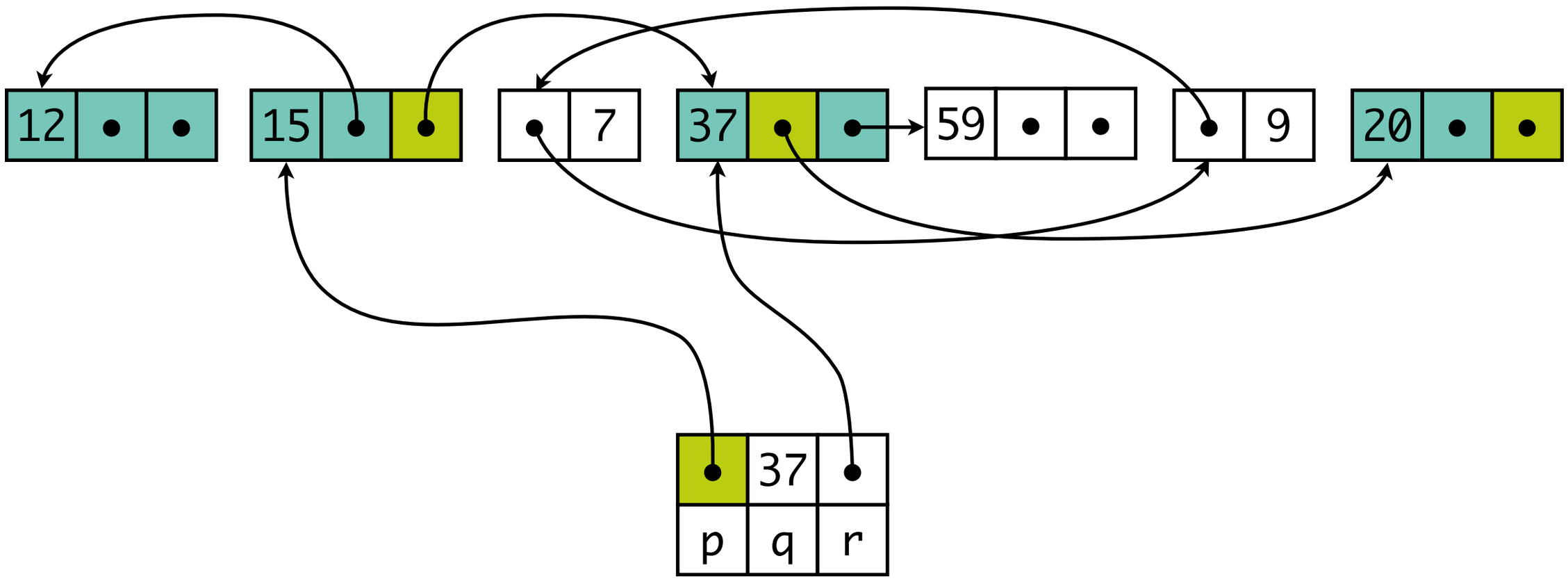
Recap: Marking

pointer reversals



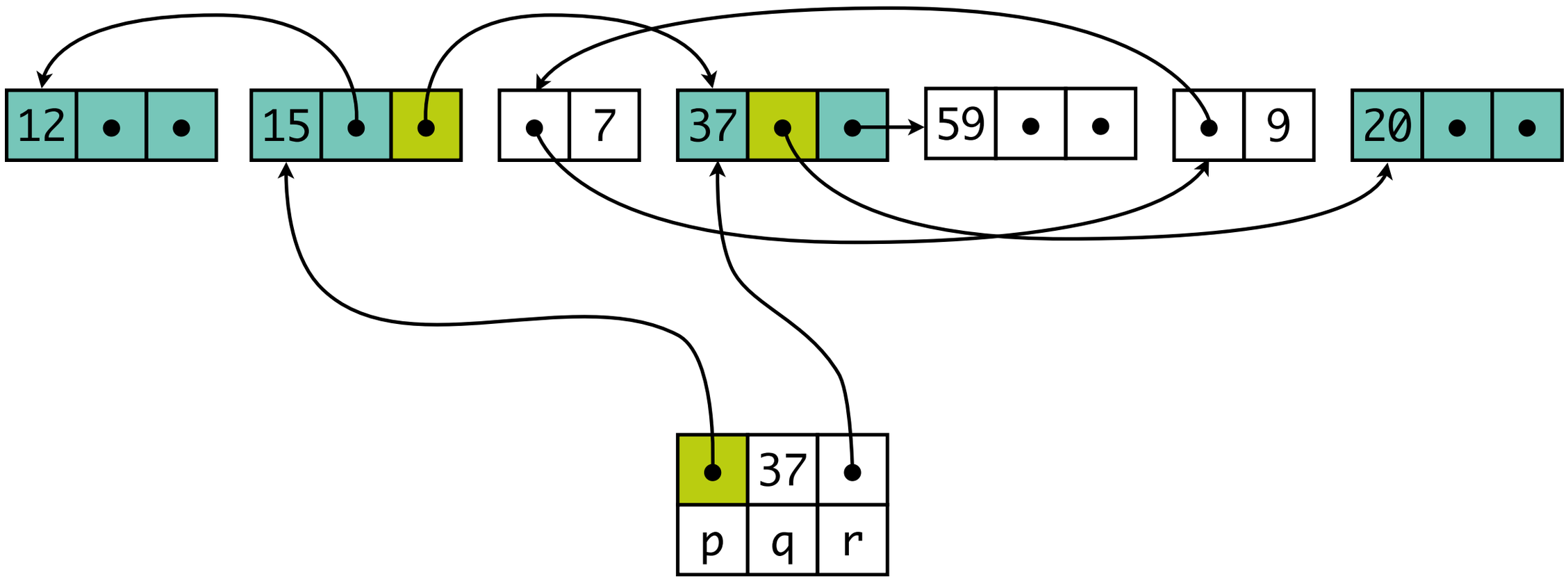
Recap: Marking

pointer reversals



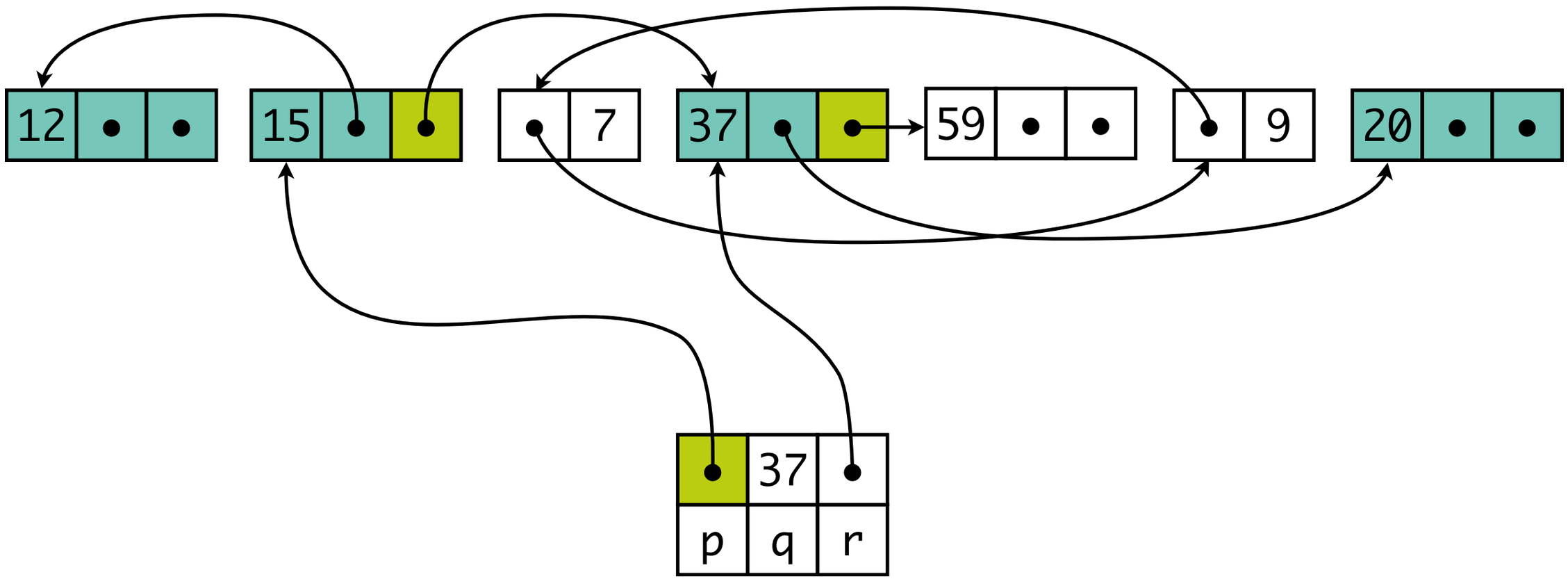
Recap: Marking

pointer reversals



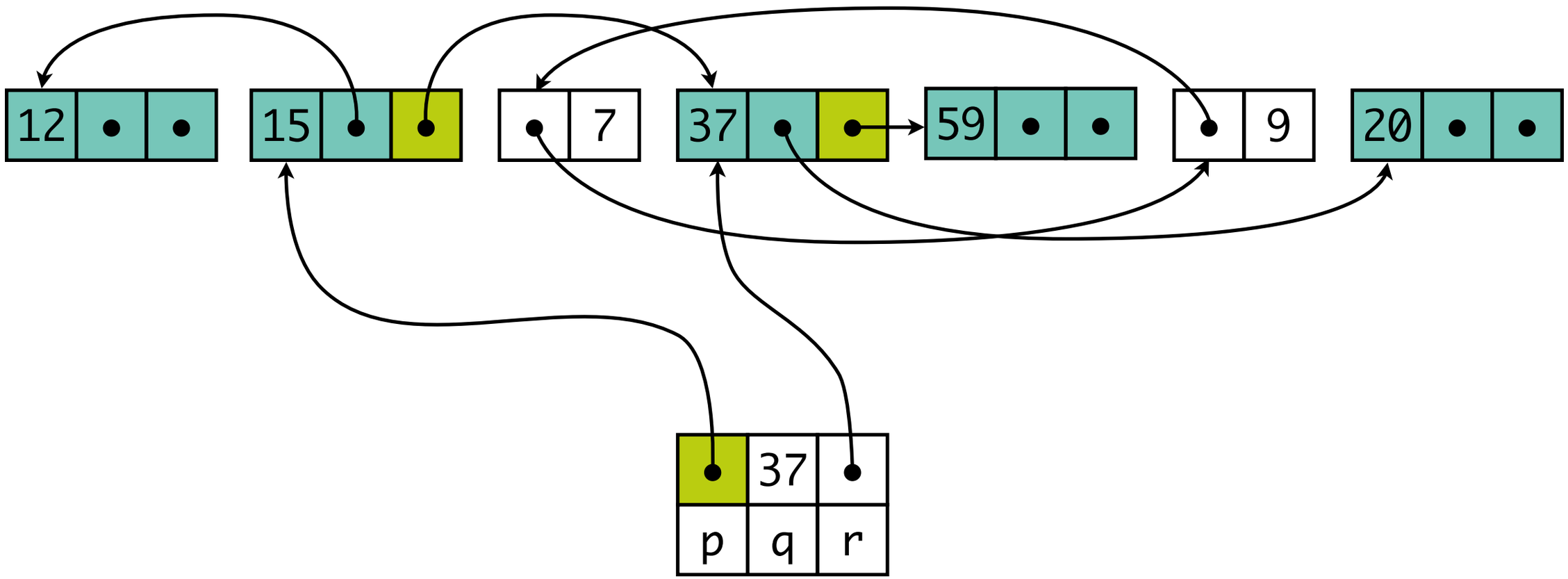
Recap: Marking

pointer reversals



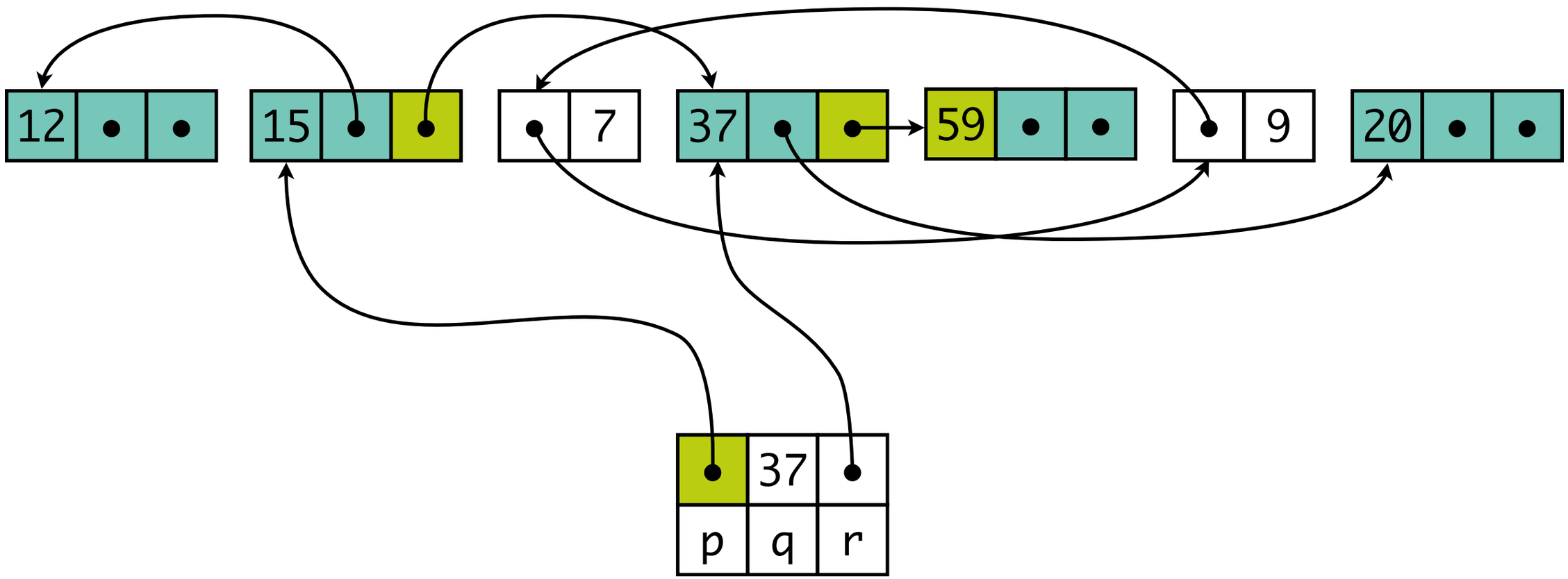
Recap: Marking

pointer reversals



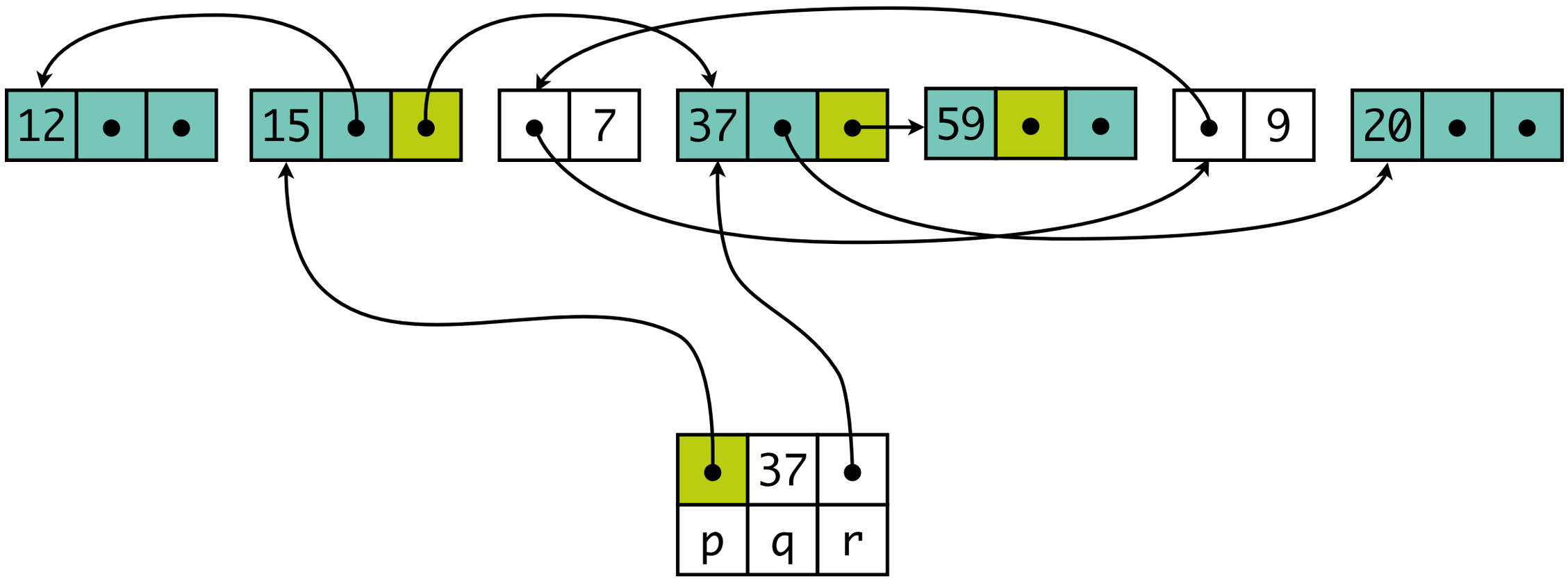
Recap: Marking

pointer reversals



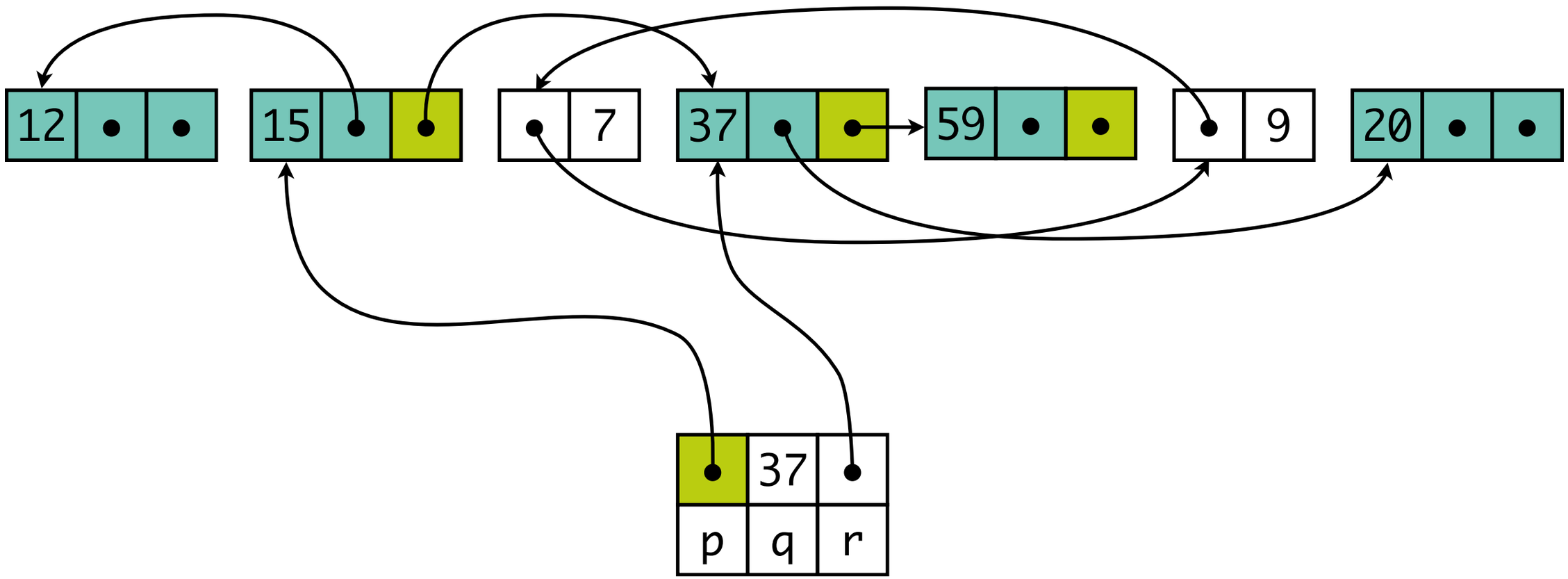
Recap: Marking

pointer reversals



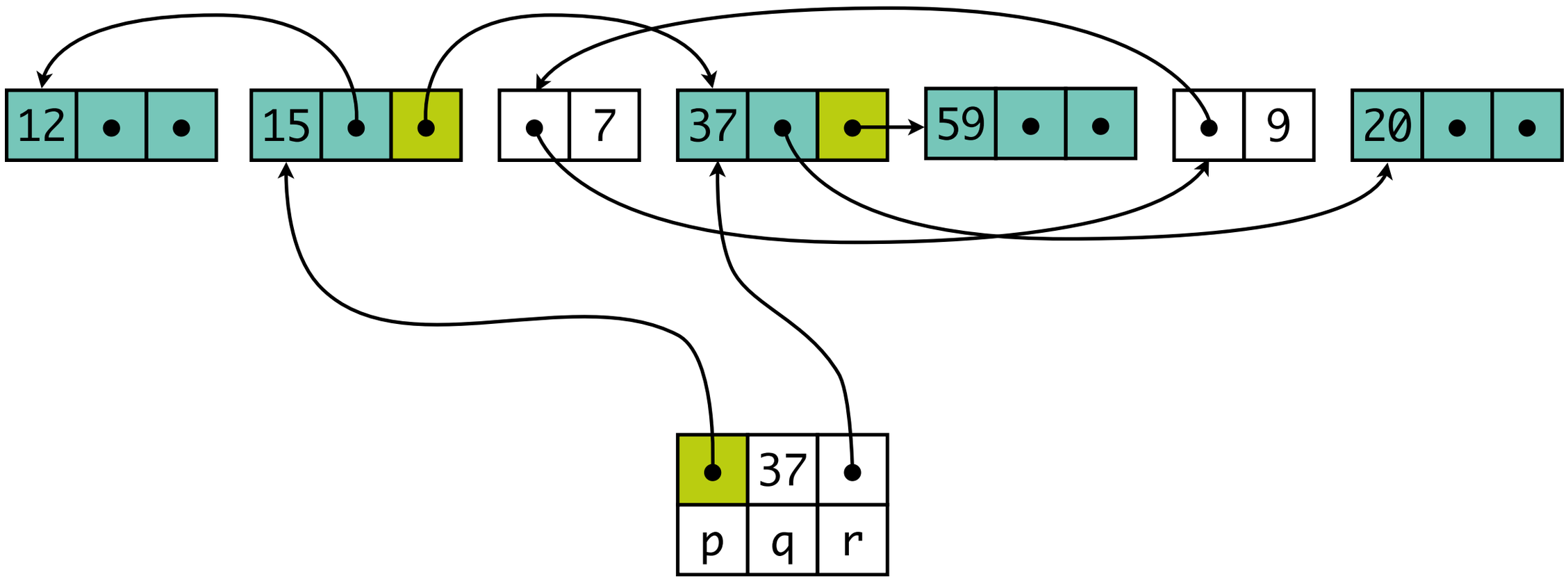
Recap: Marking

pointer reversals



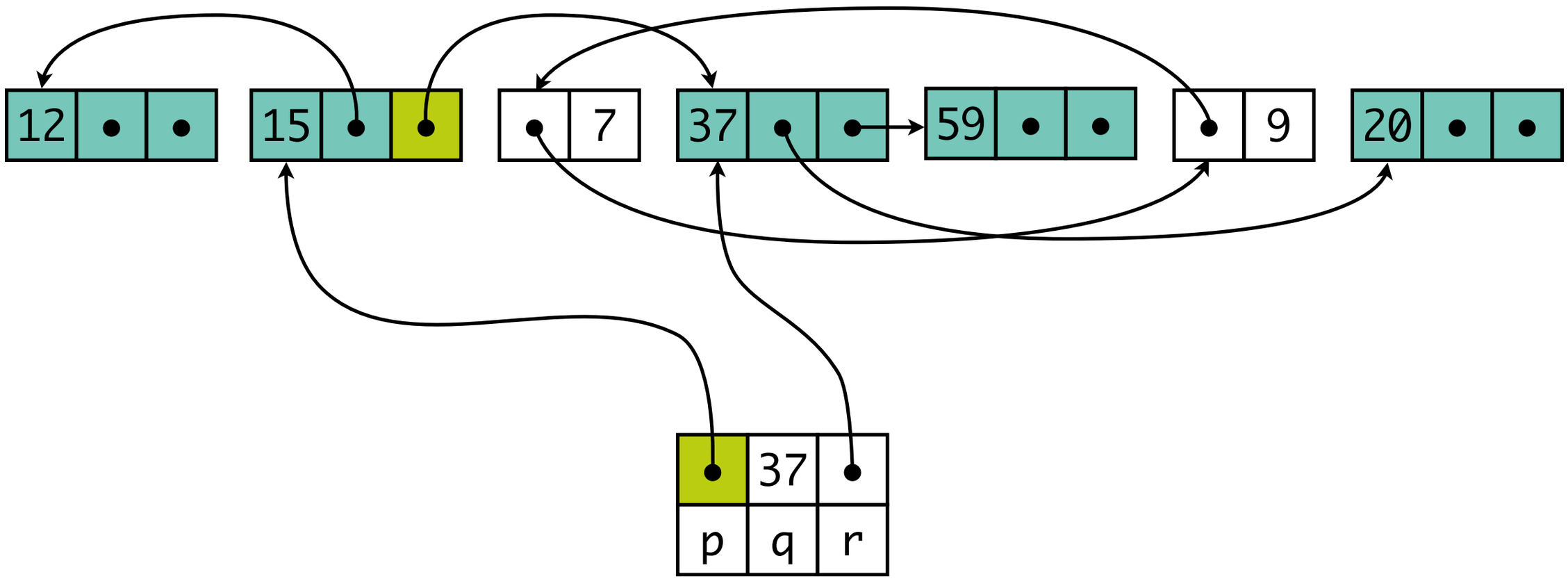
Recap: Marking

pointer reversals



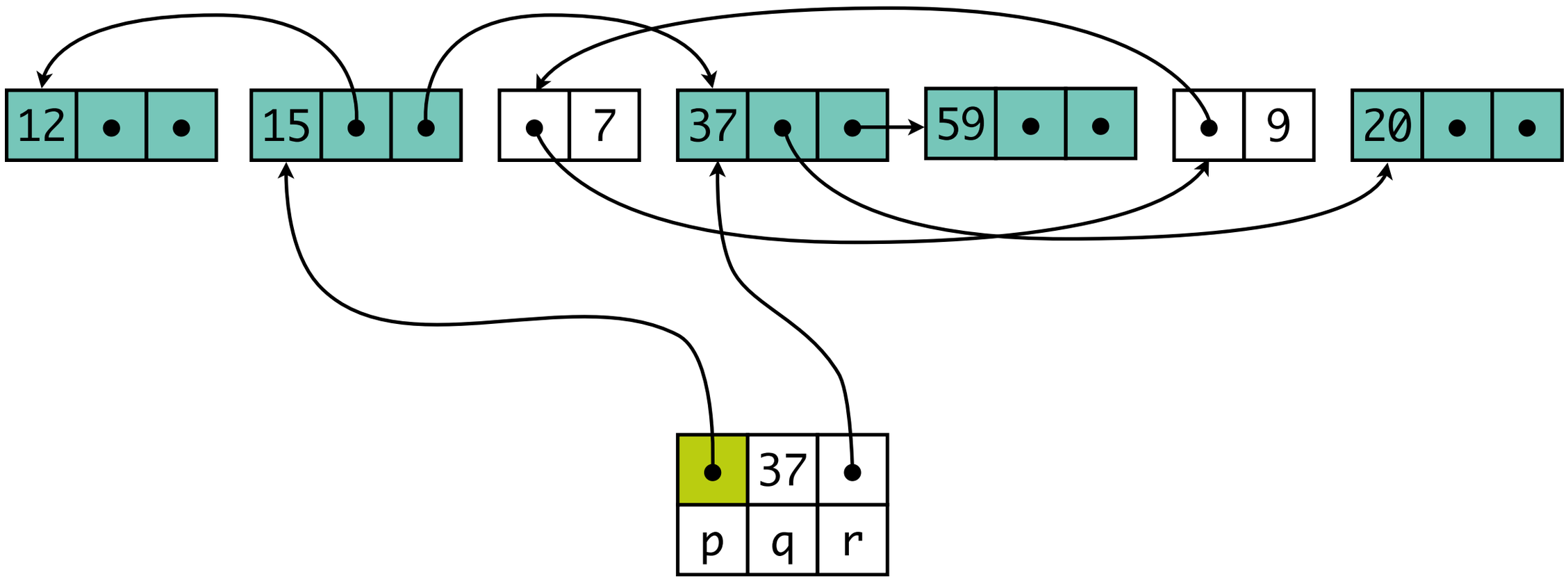
Recap: Marking

pointer reversals



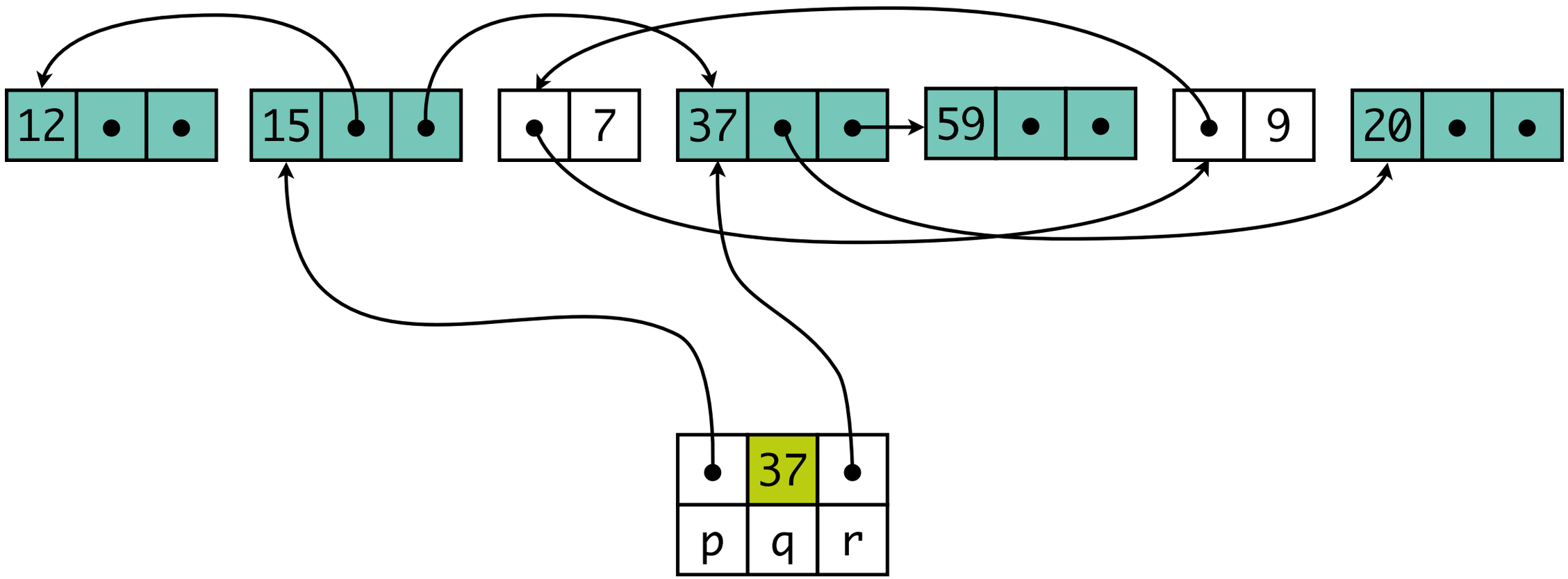
Recap: Marking

pointer reversals



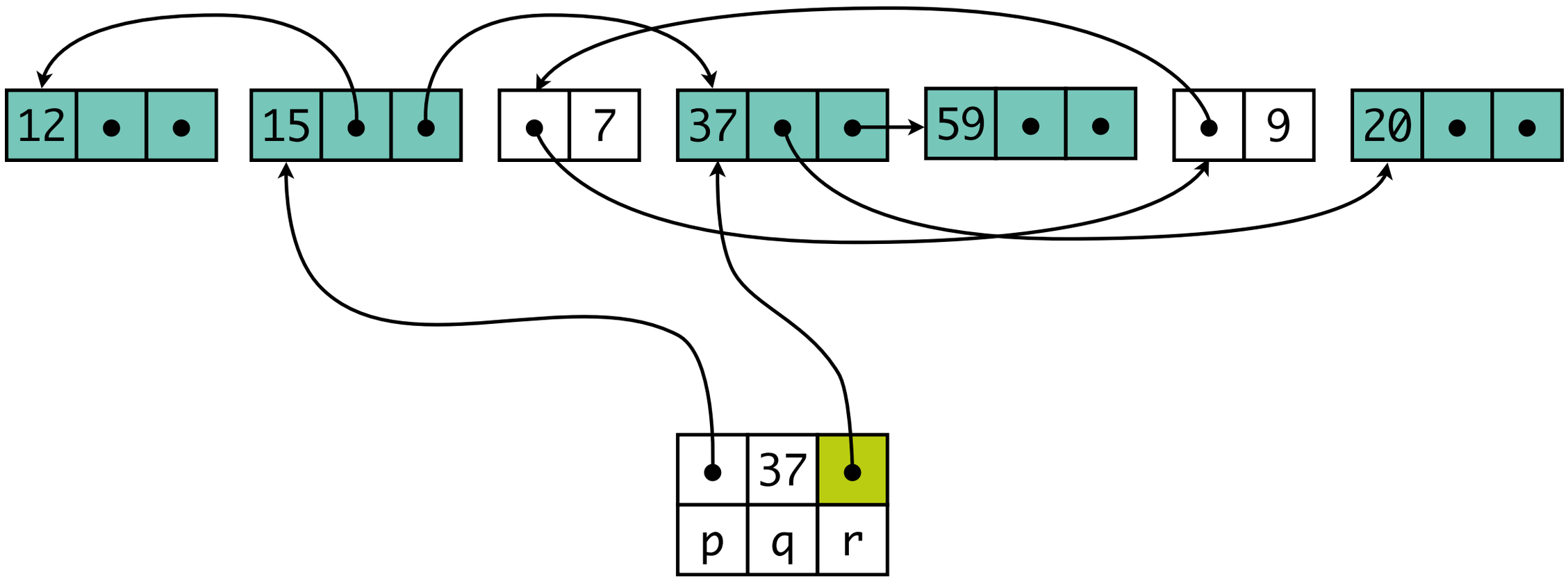
Recap: Marking

pointer reversals



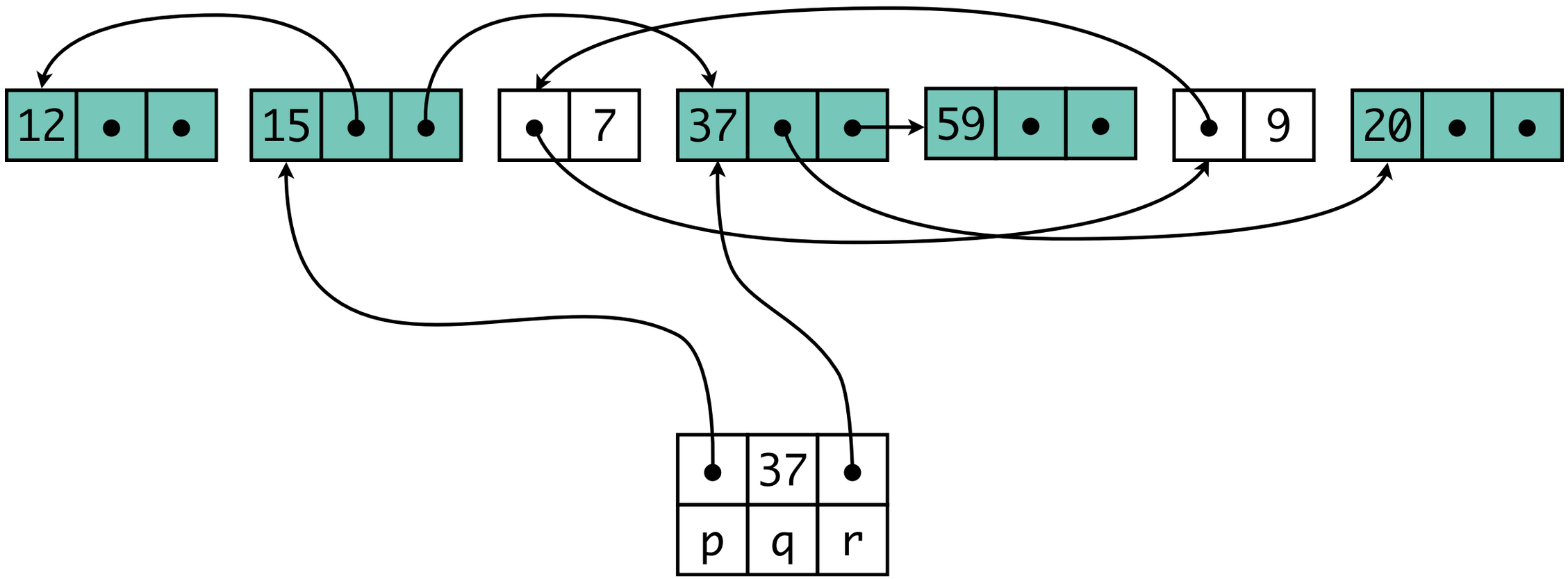
Recap: Marking

pointer reversals



Recap: Marking

pointer reversals



Marking
algorithms

```
function DFS(x)
  if pointer(x) & x.done < 0
    x.done = 0 ; t = nil

    while true
      if x.done < x.fields.size
        y = x.fields[x.done]
        if pointer(y) & y.done < 0
          x.fields[x.done] = t ; t = x ; x = y ; x.done = 0
        else
          x.done = x.done + 1

      else
        y = x; x = t
        if t = nil then return
        t = x.fields[x.done]; x.fields[x.done] = y
        x.done = x.done + 1
```

Mark & sweep

notes

sweeping

- independent of marking algorithm
- several freelists (per record size)
- split free records for allocation

fragmentation

- external: many free records of small size
- internal: too-large record with unused memory inside

coffee break

IV

copy collections

Copy collections
idea

spaces

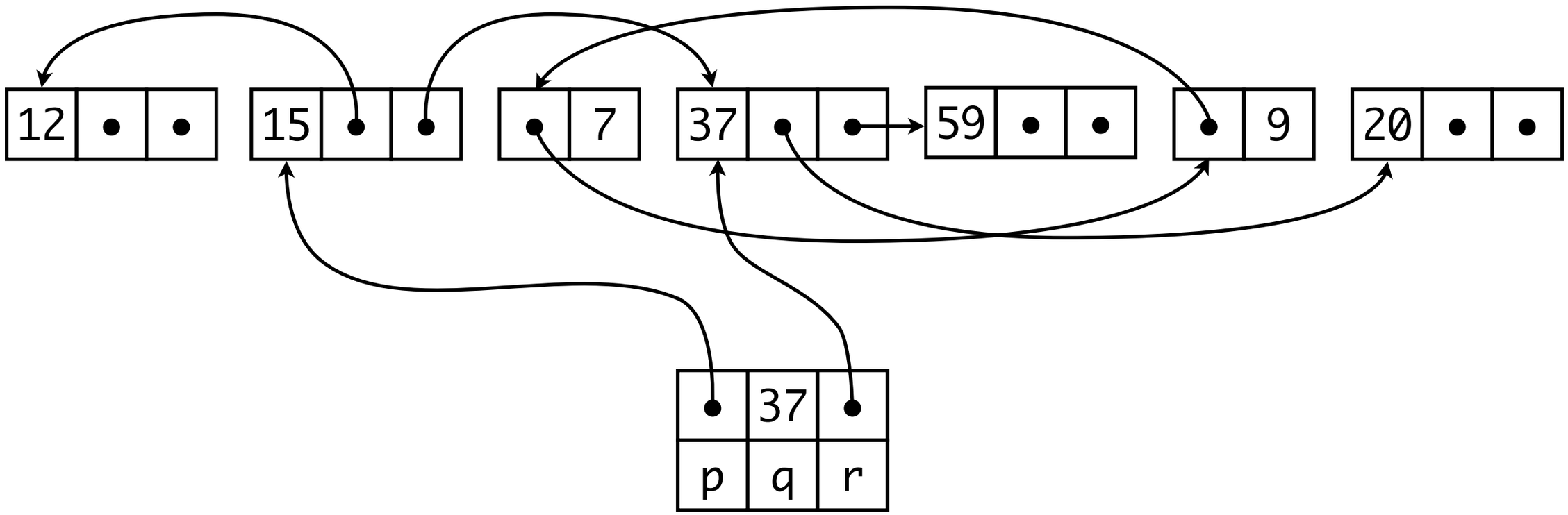
- fromspace & tospace
- switch roles after copy

copy

- traverse reachability graph
- copy from fromspace to tospace
- fromspace unreachable, free memory
- tospace compact, no fragmentation

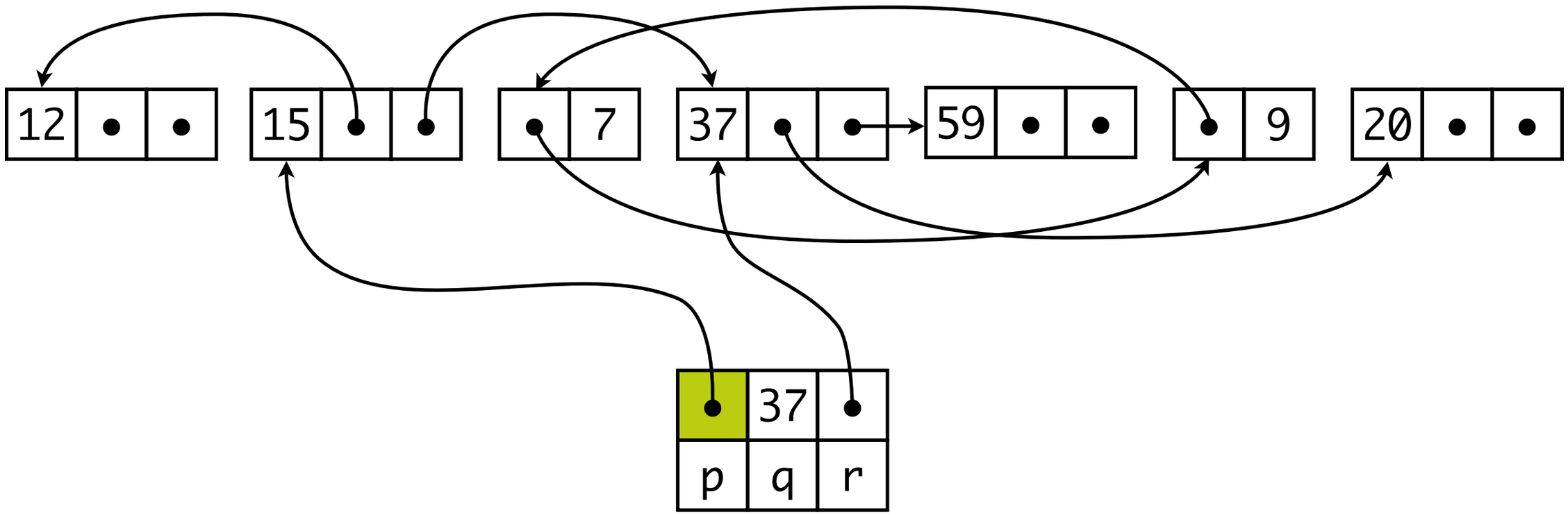
Copy collection

example



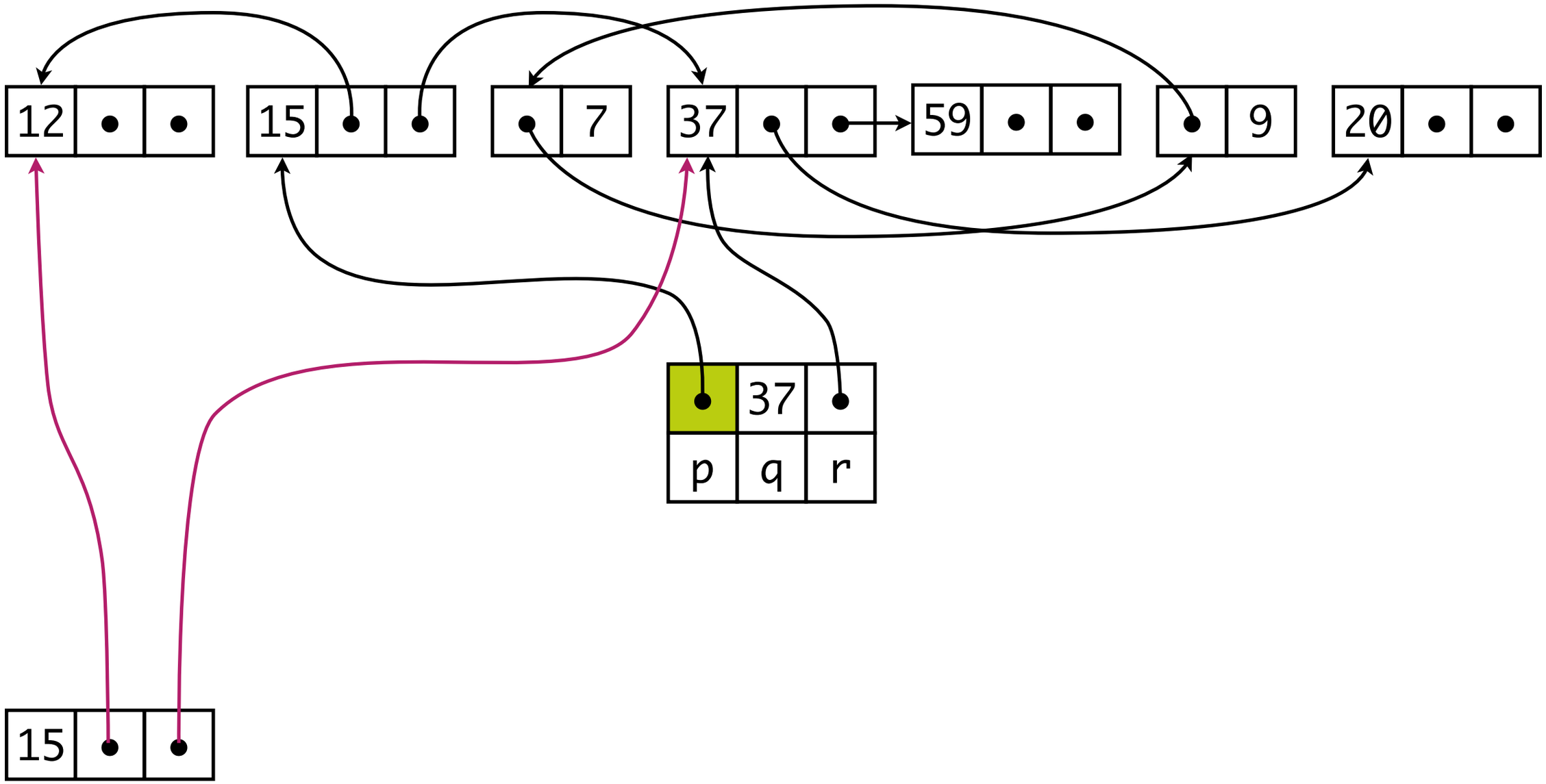
Copy collection

example



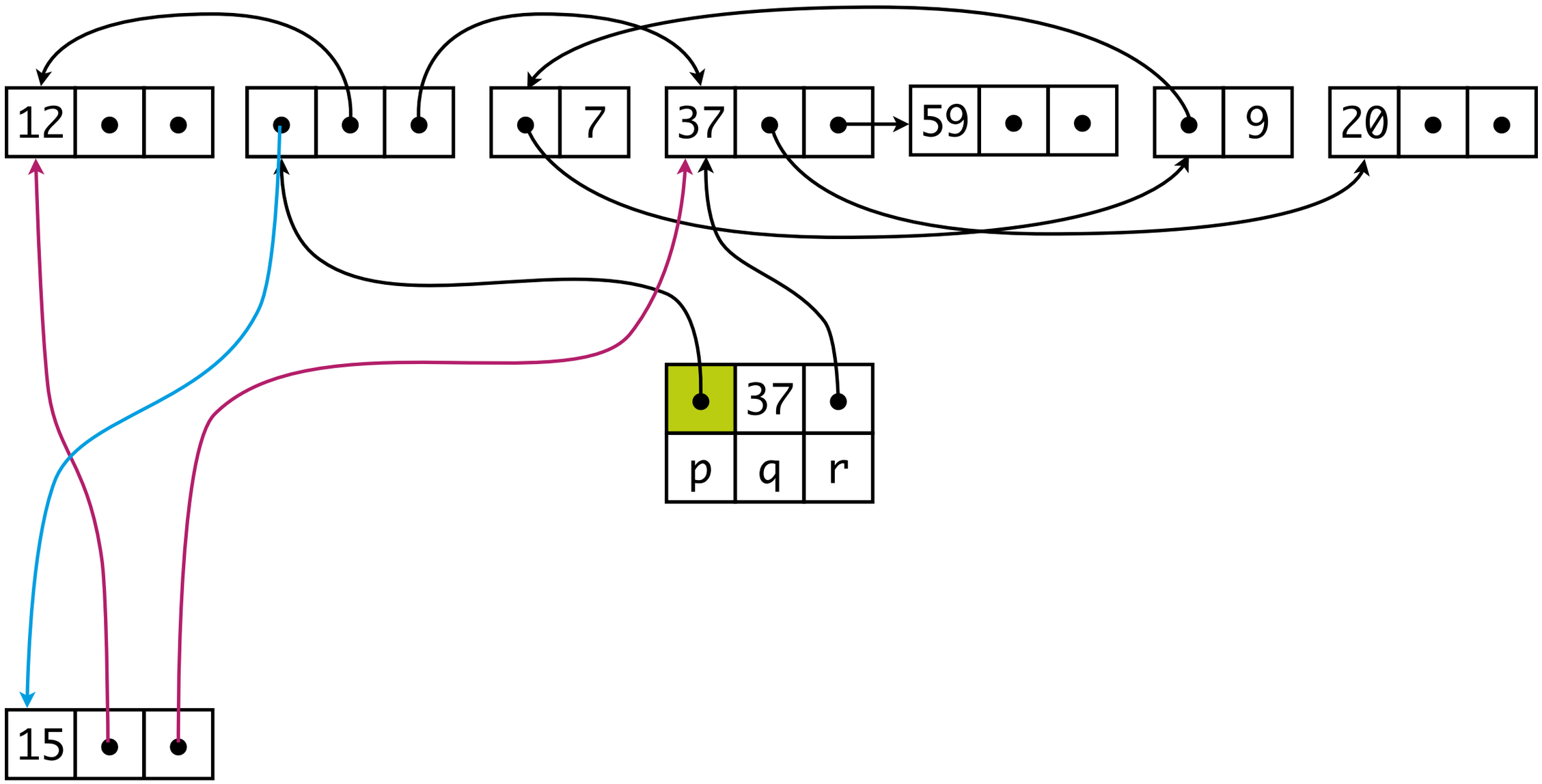
Copy collection

example



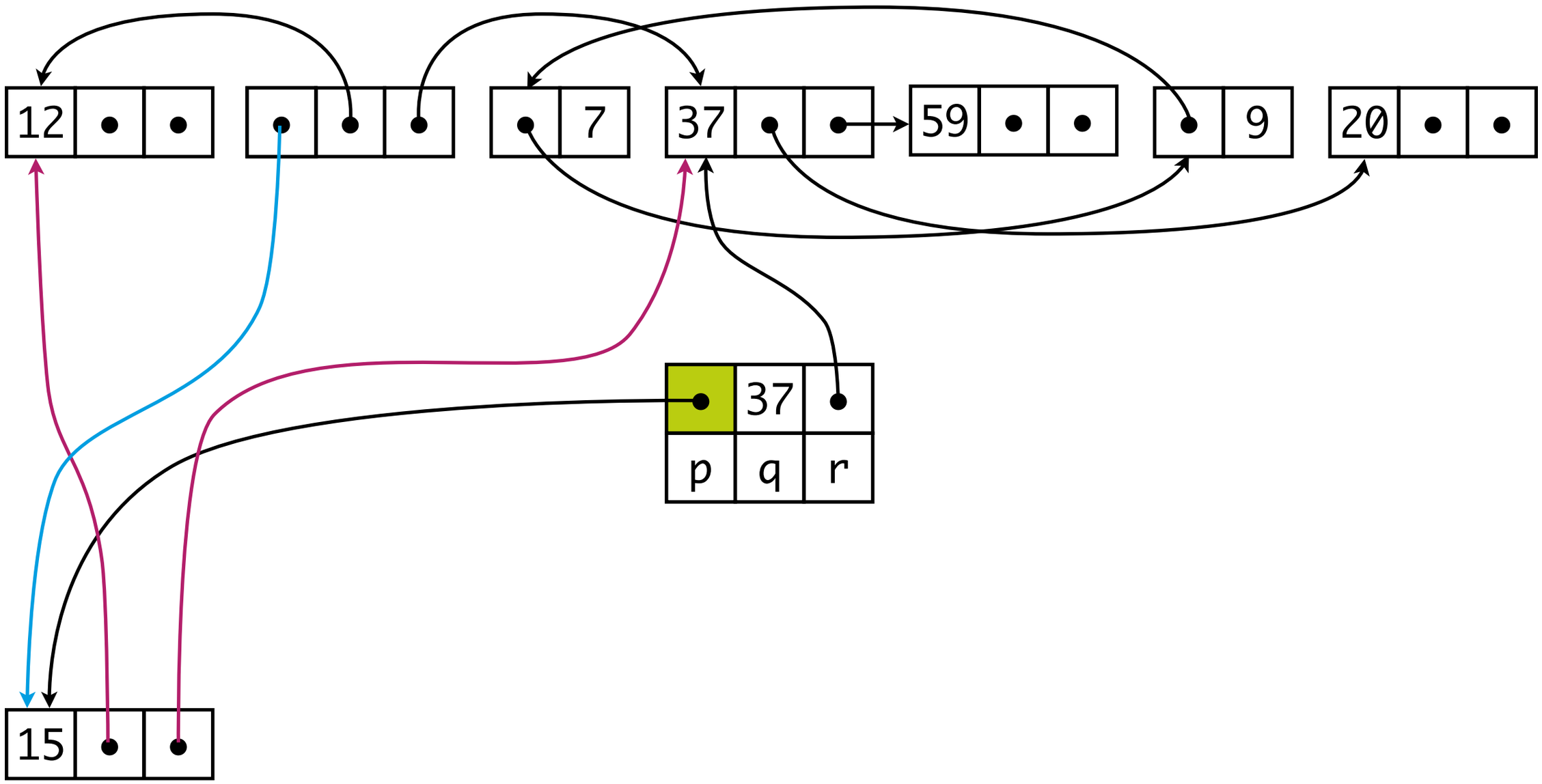
Copy collection

example



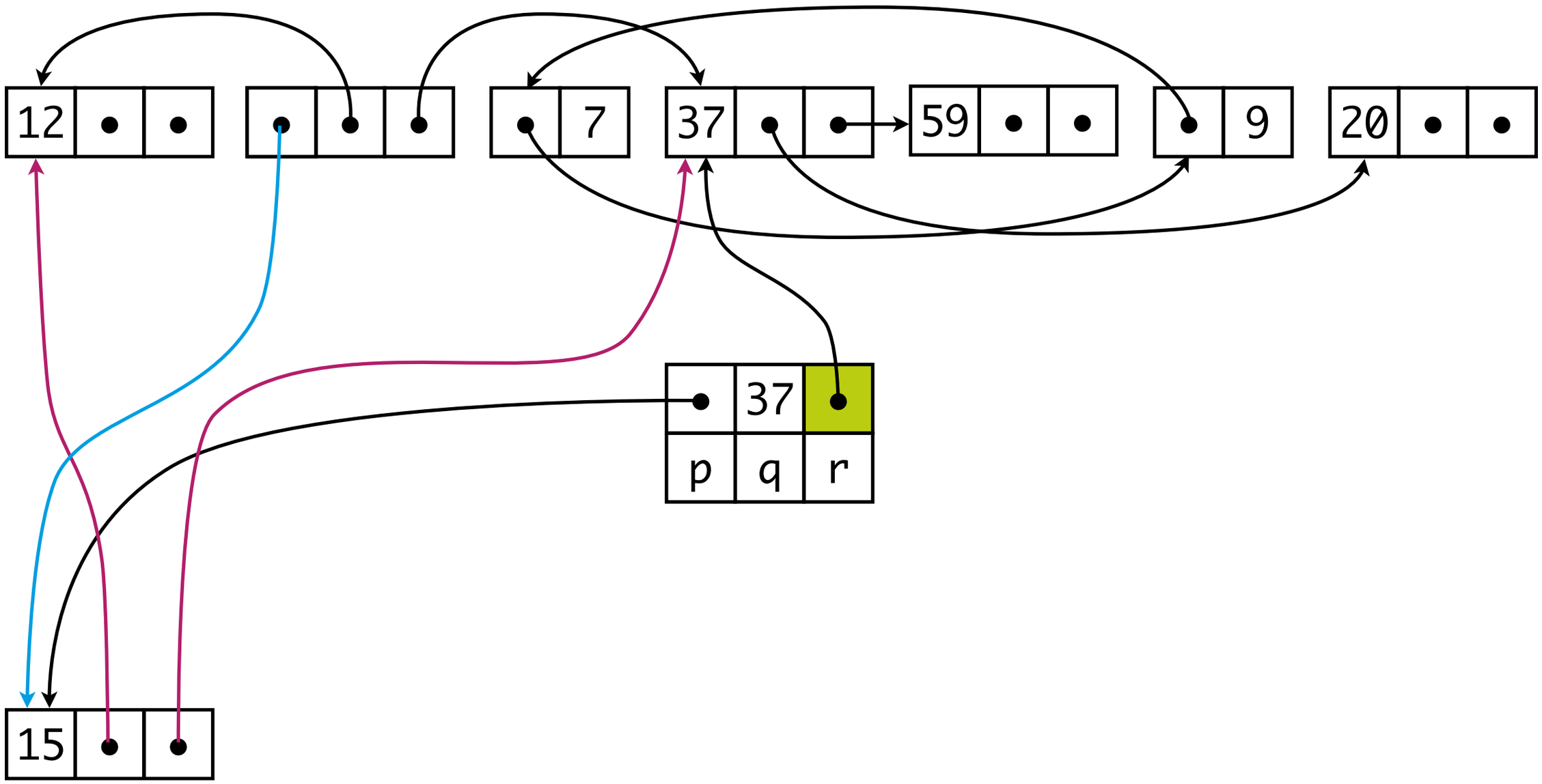
Copy collection

example



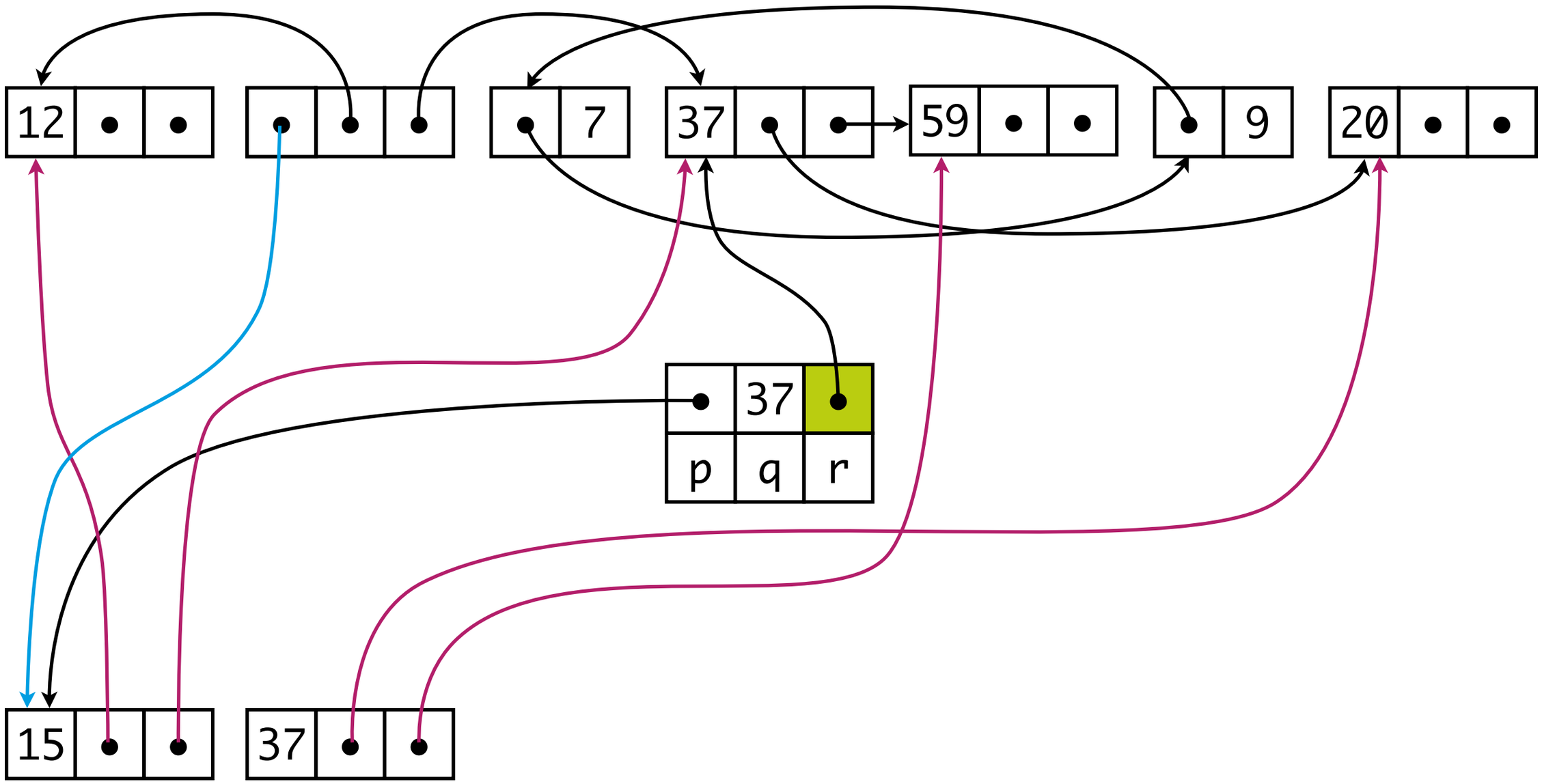
Copy collection

example



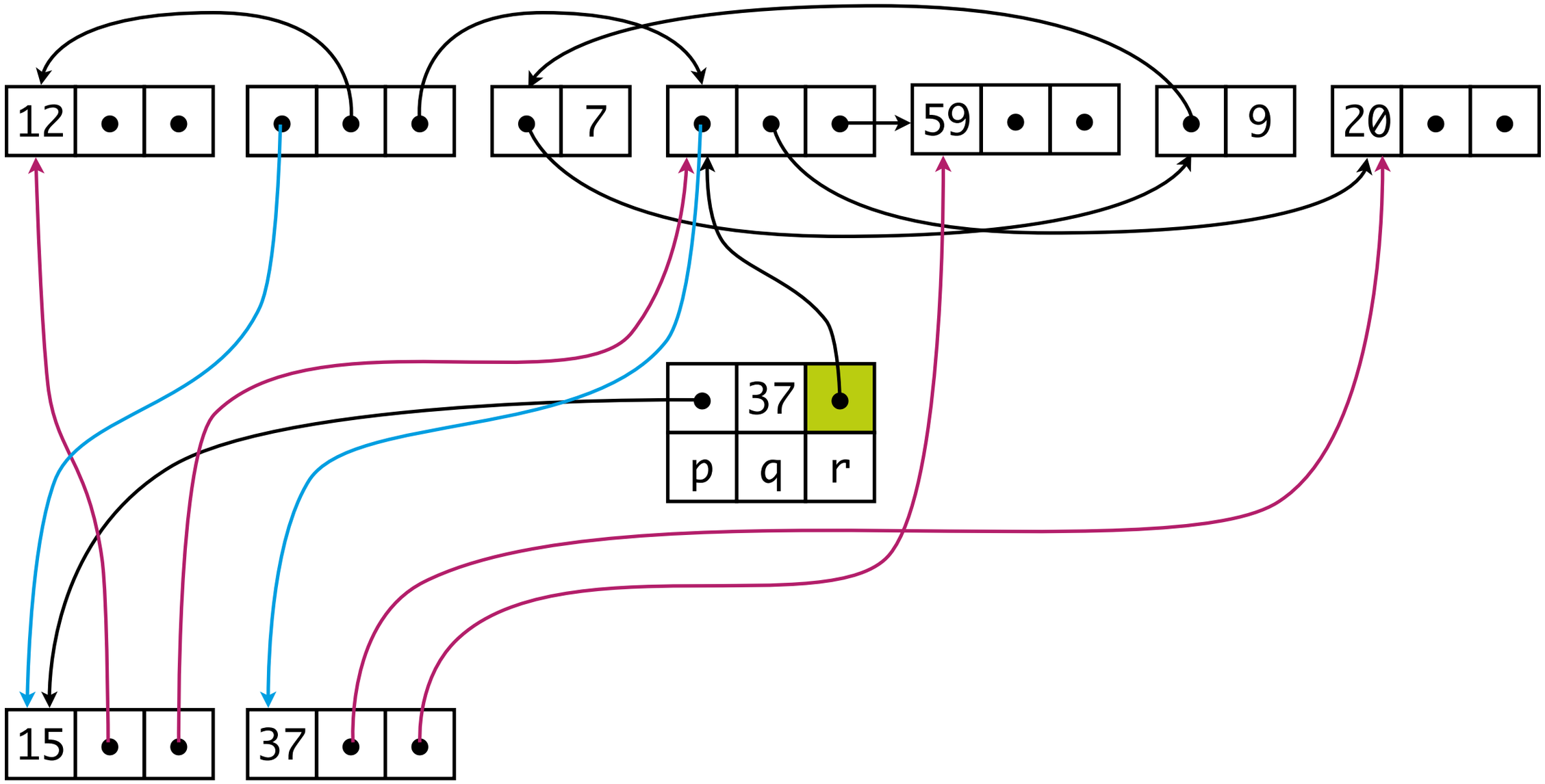
Copy collection

example



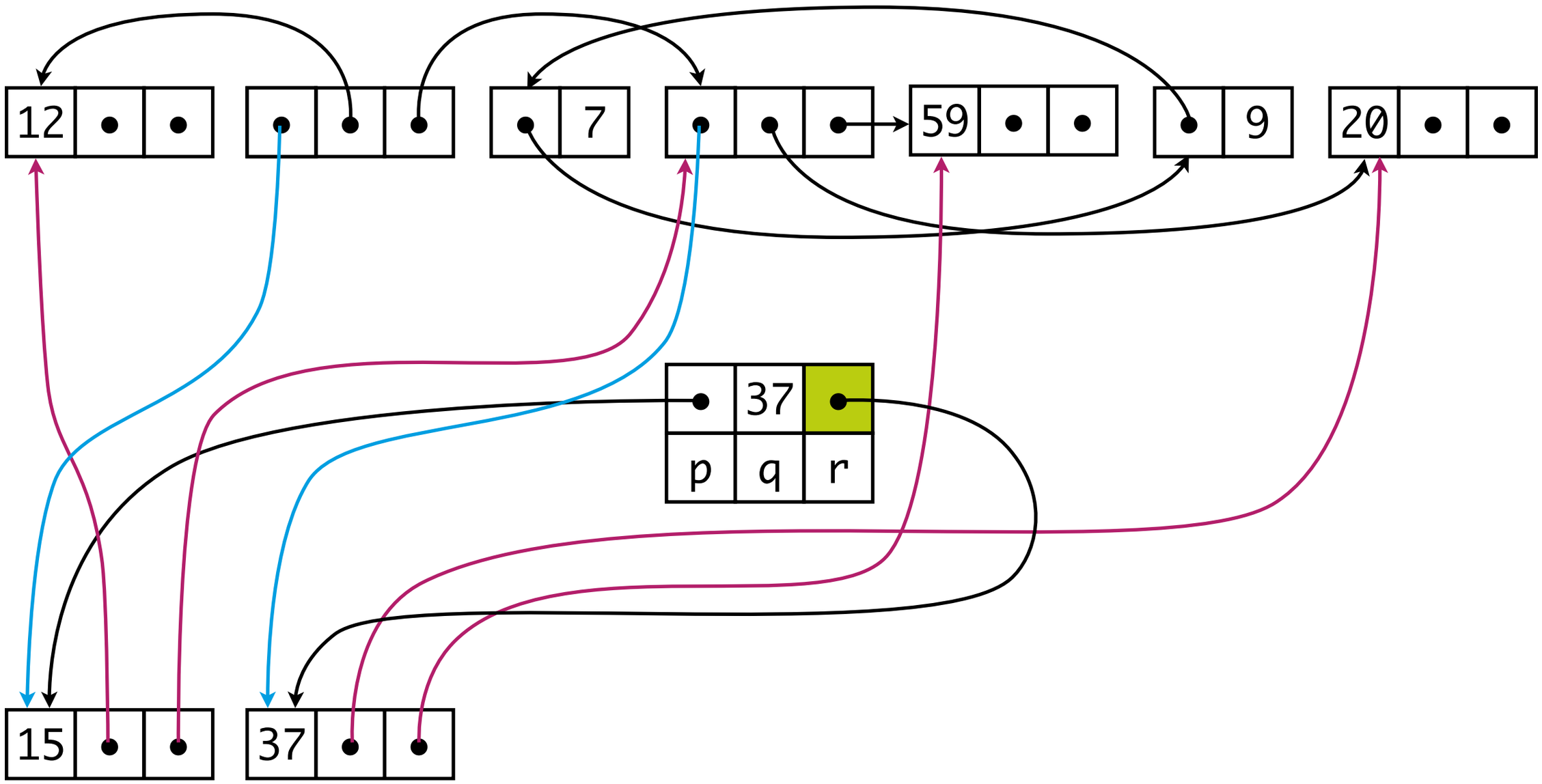
Copy collection

example



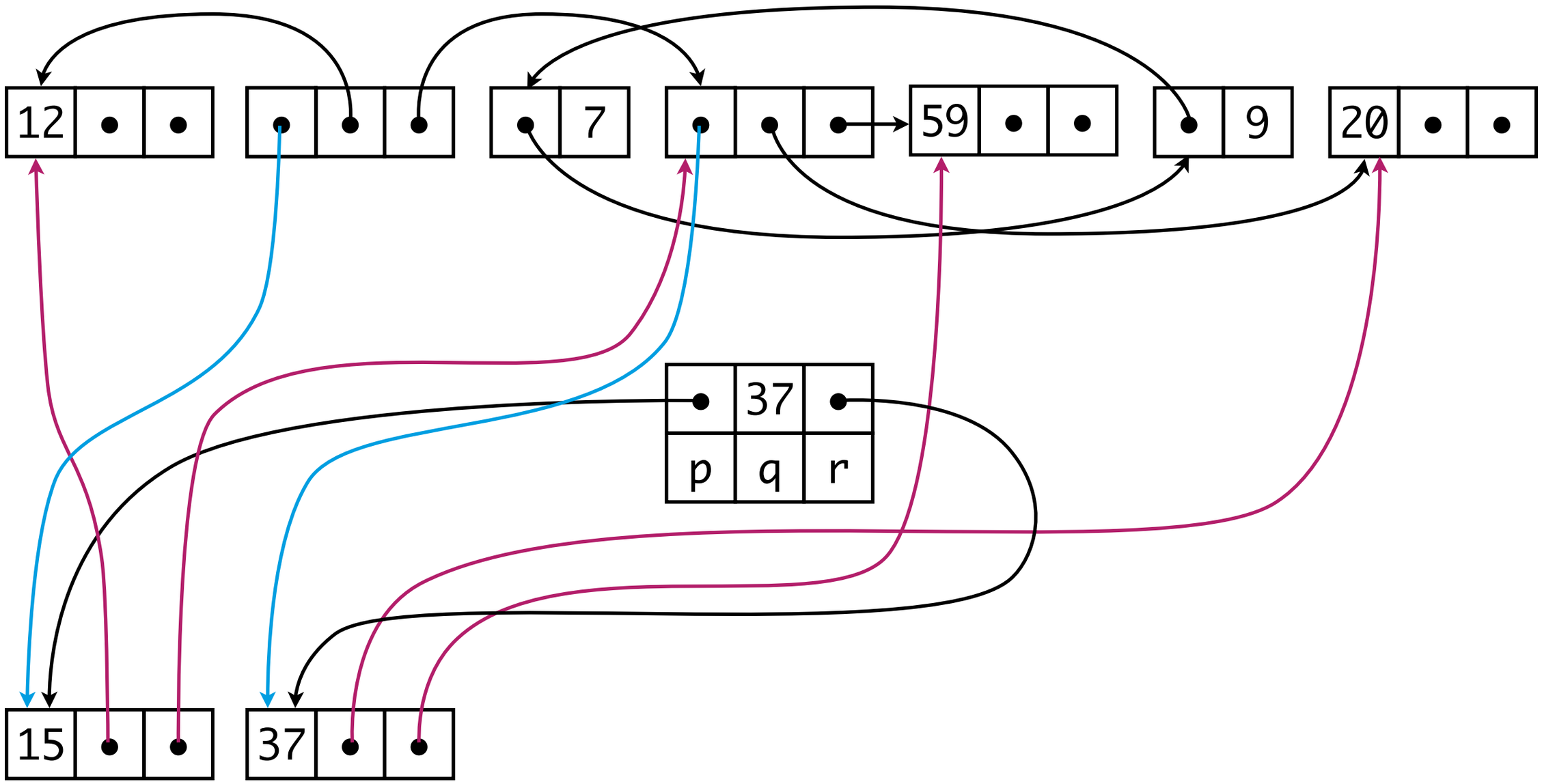
Copy collection

example

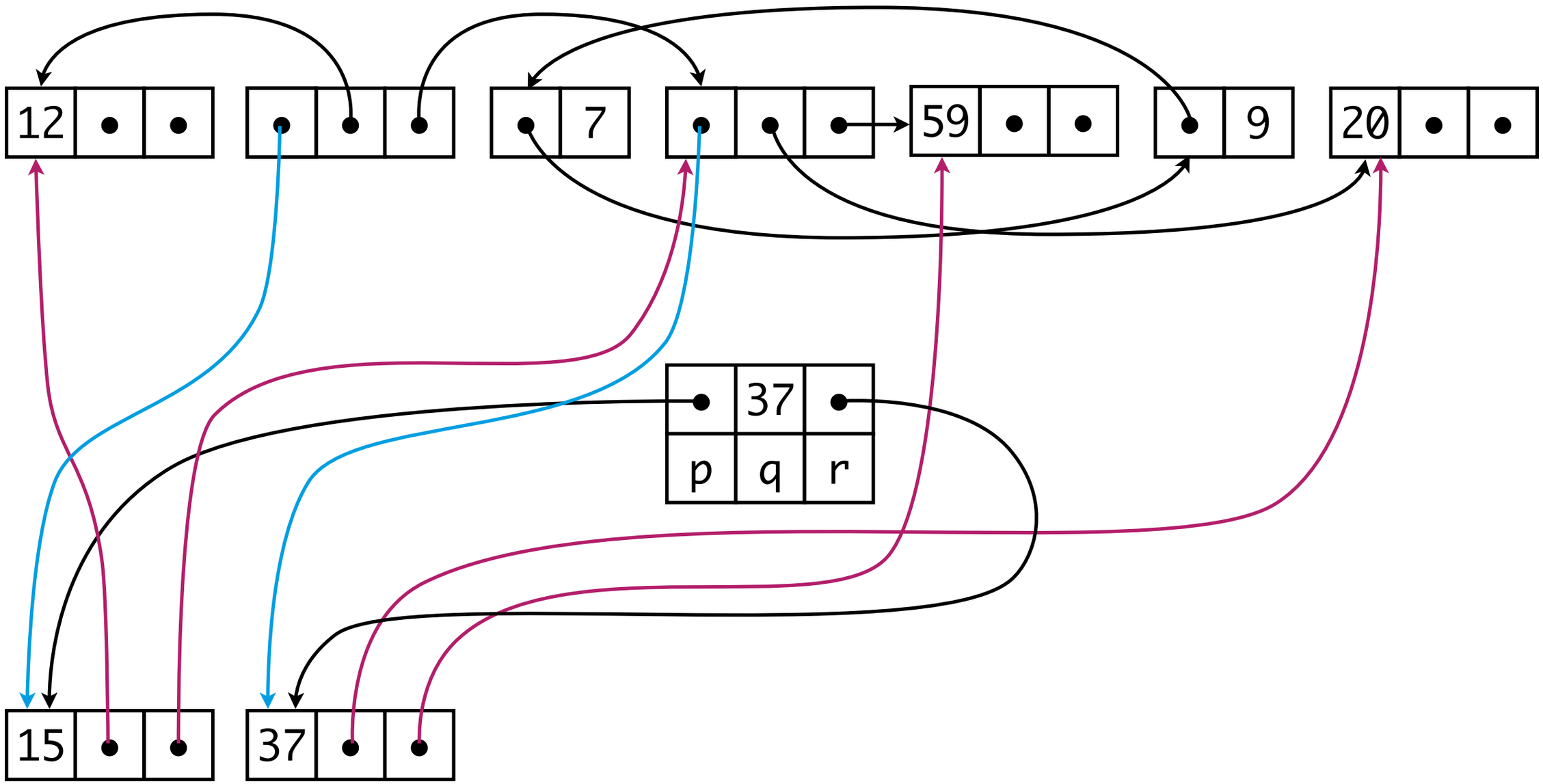


Copy collection

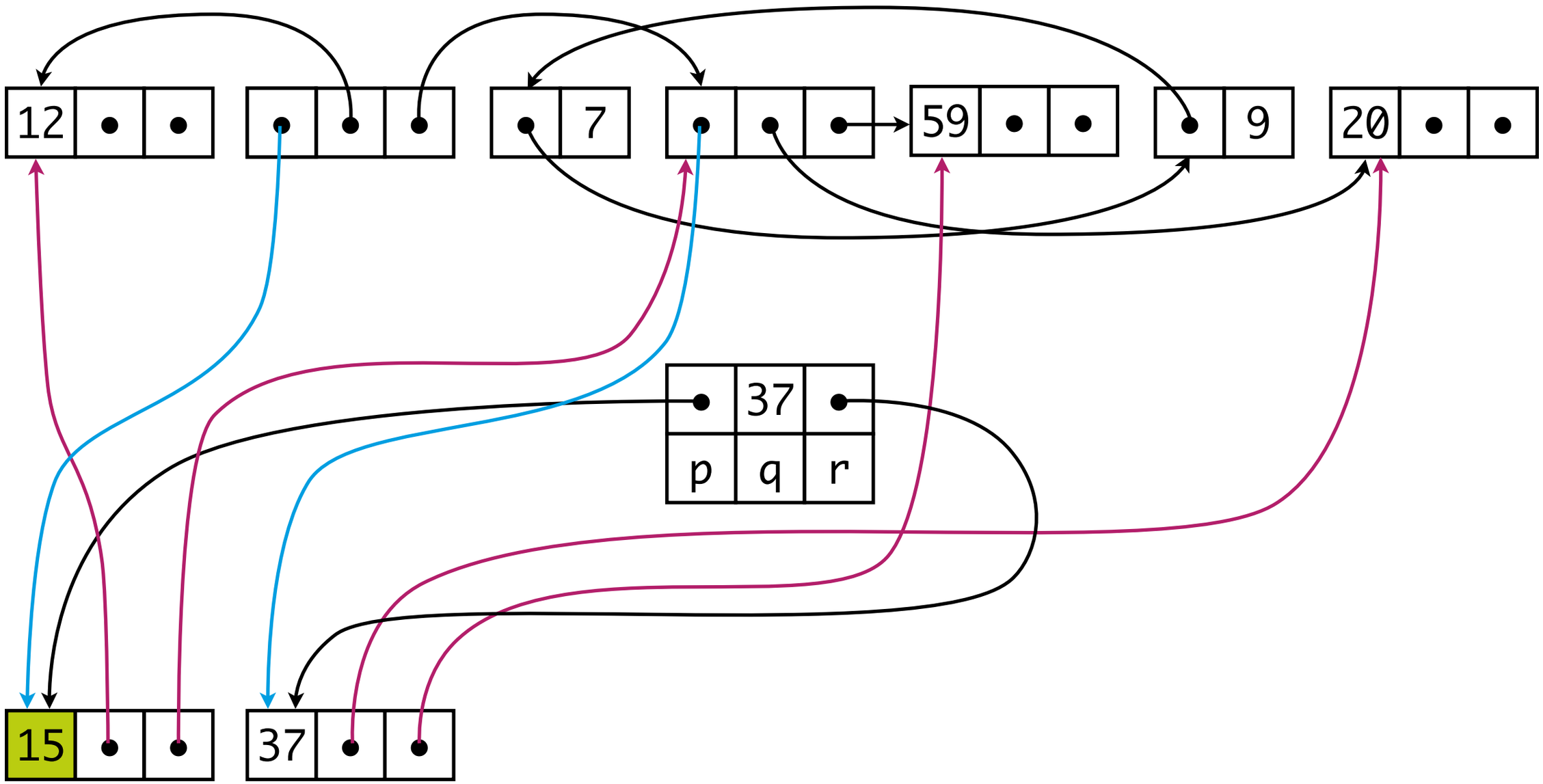
example



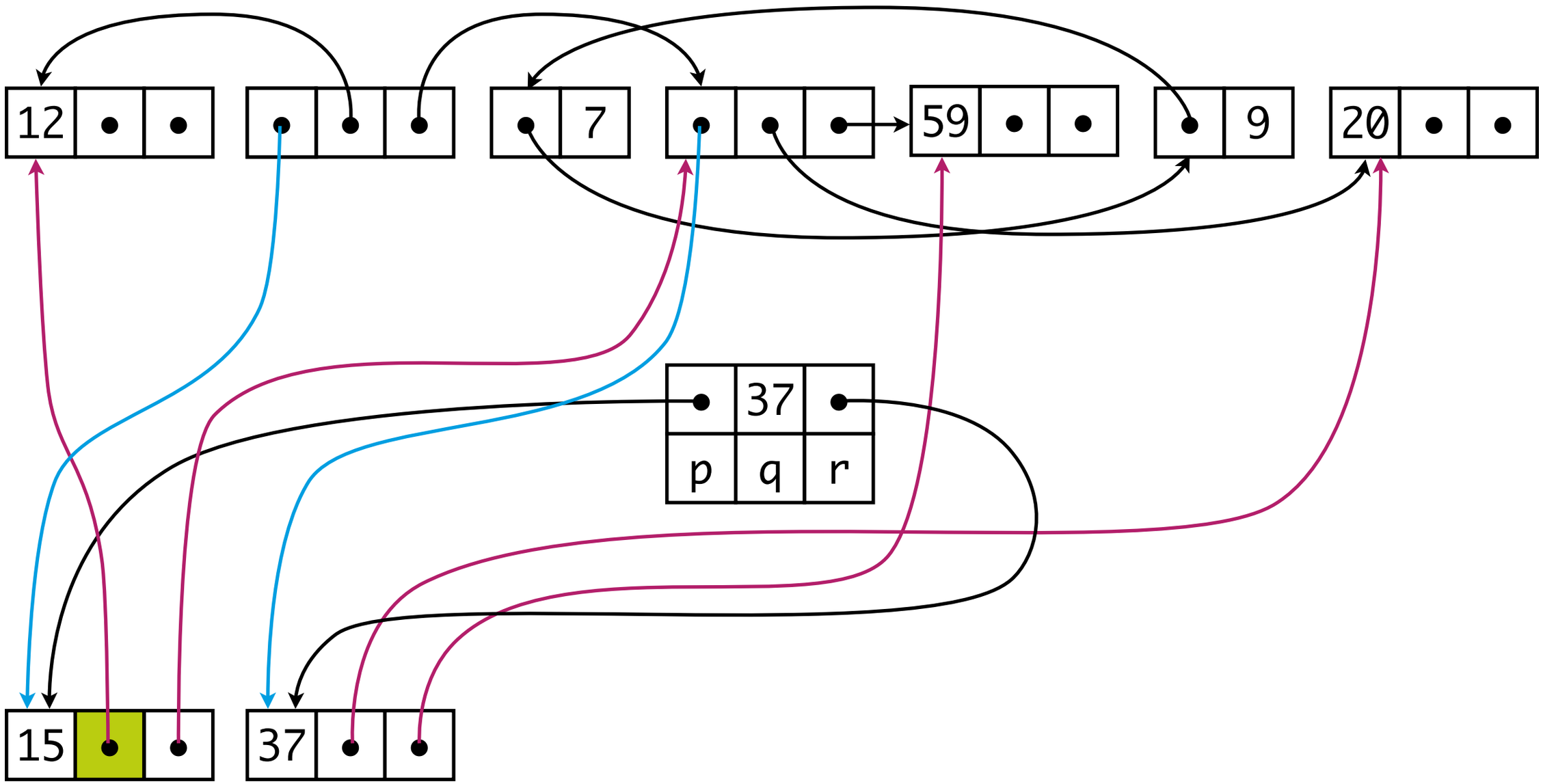
Copy collection



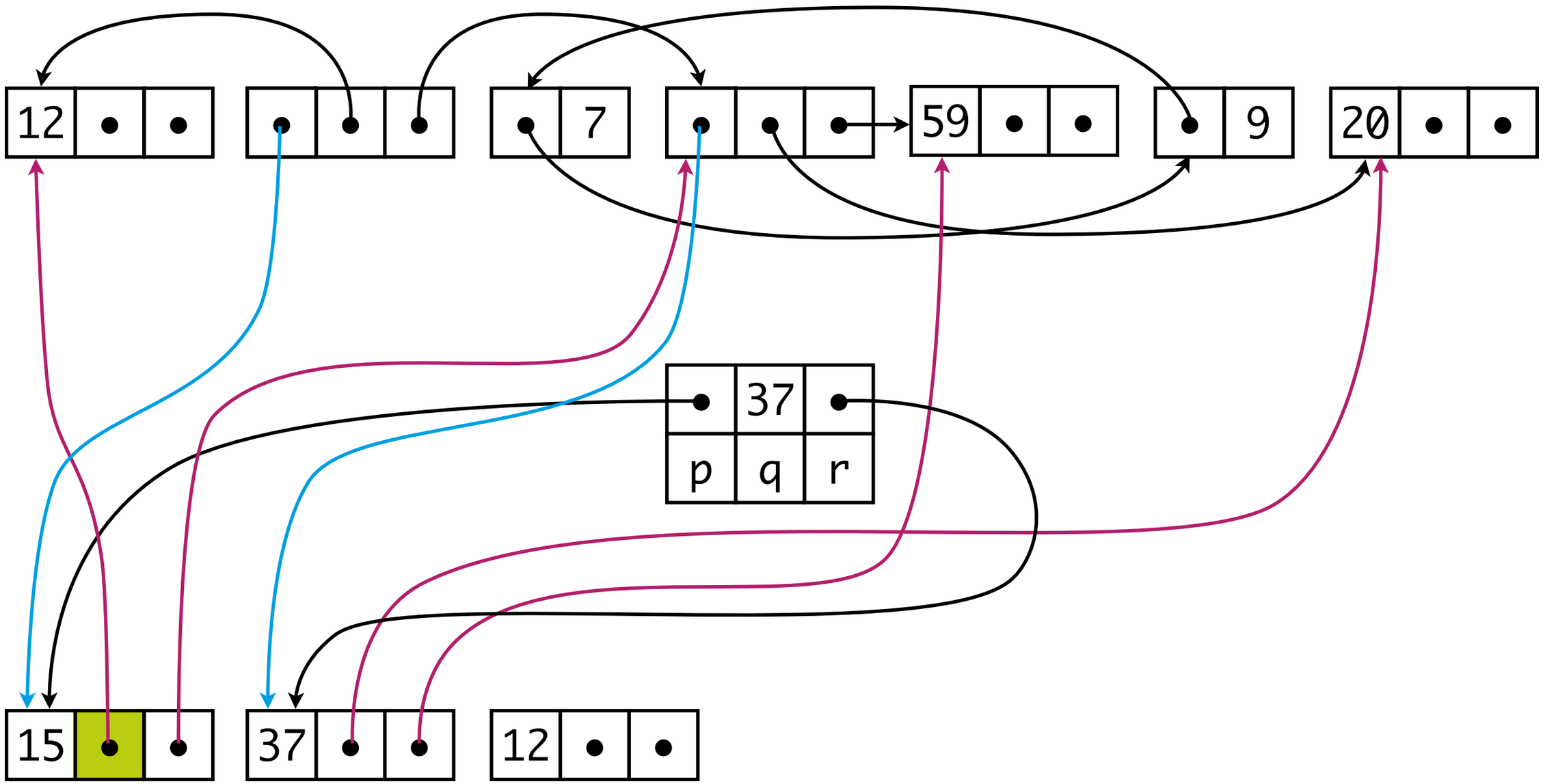
Copy collection



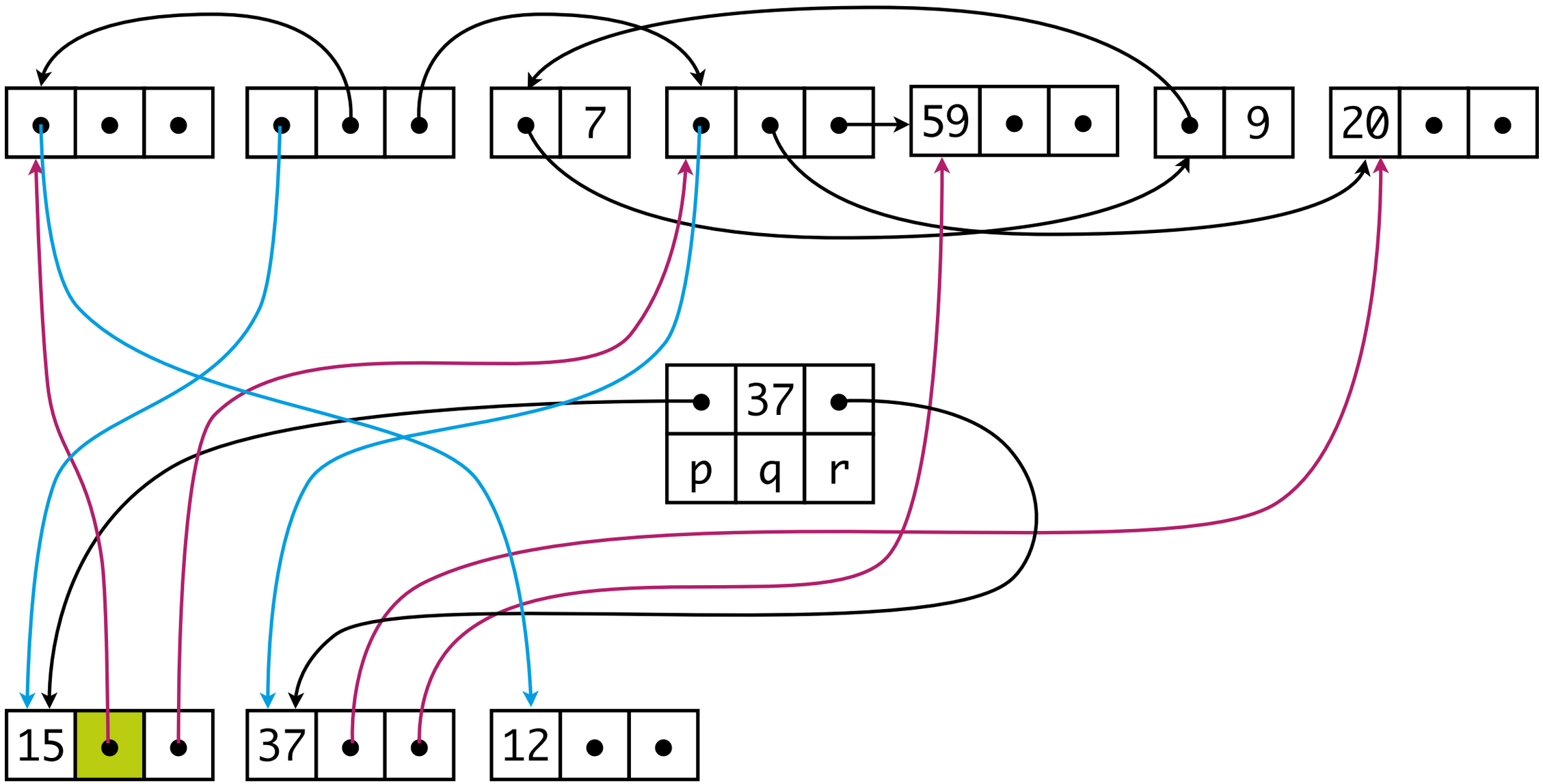
Copy collection



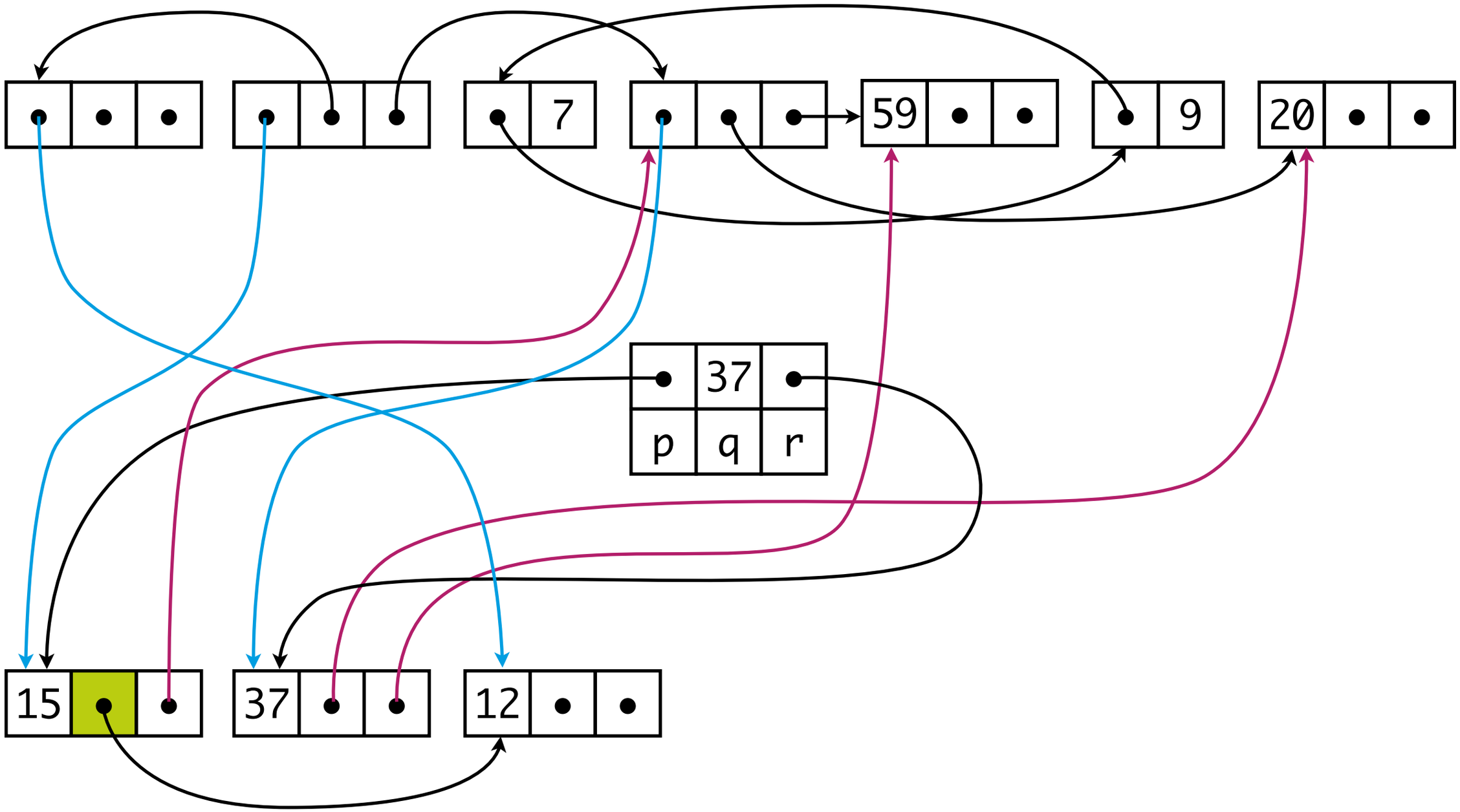
Copy collection



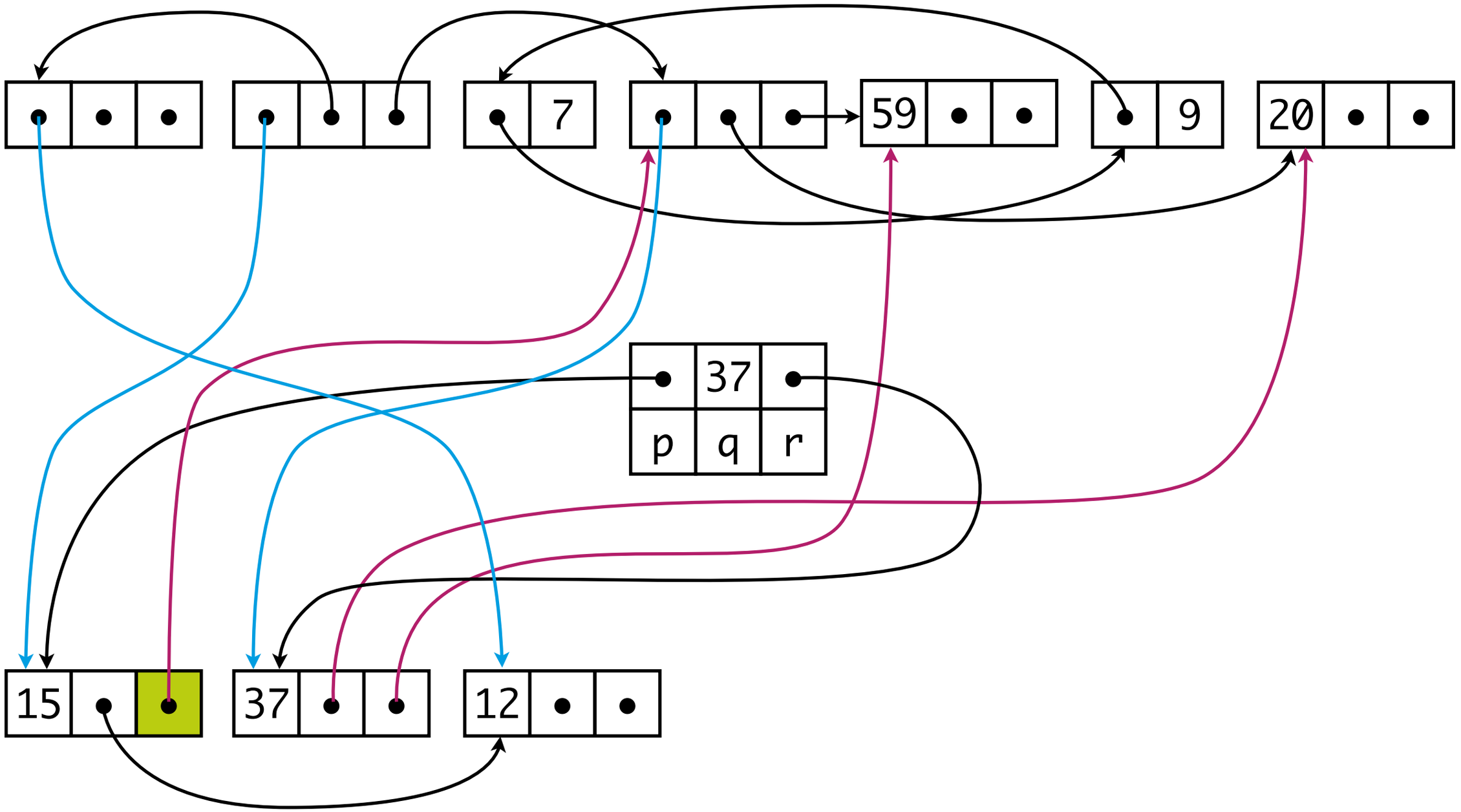
Copy collection



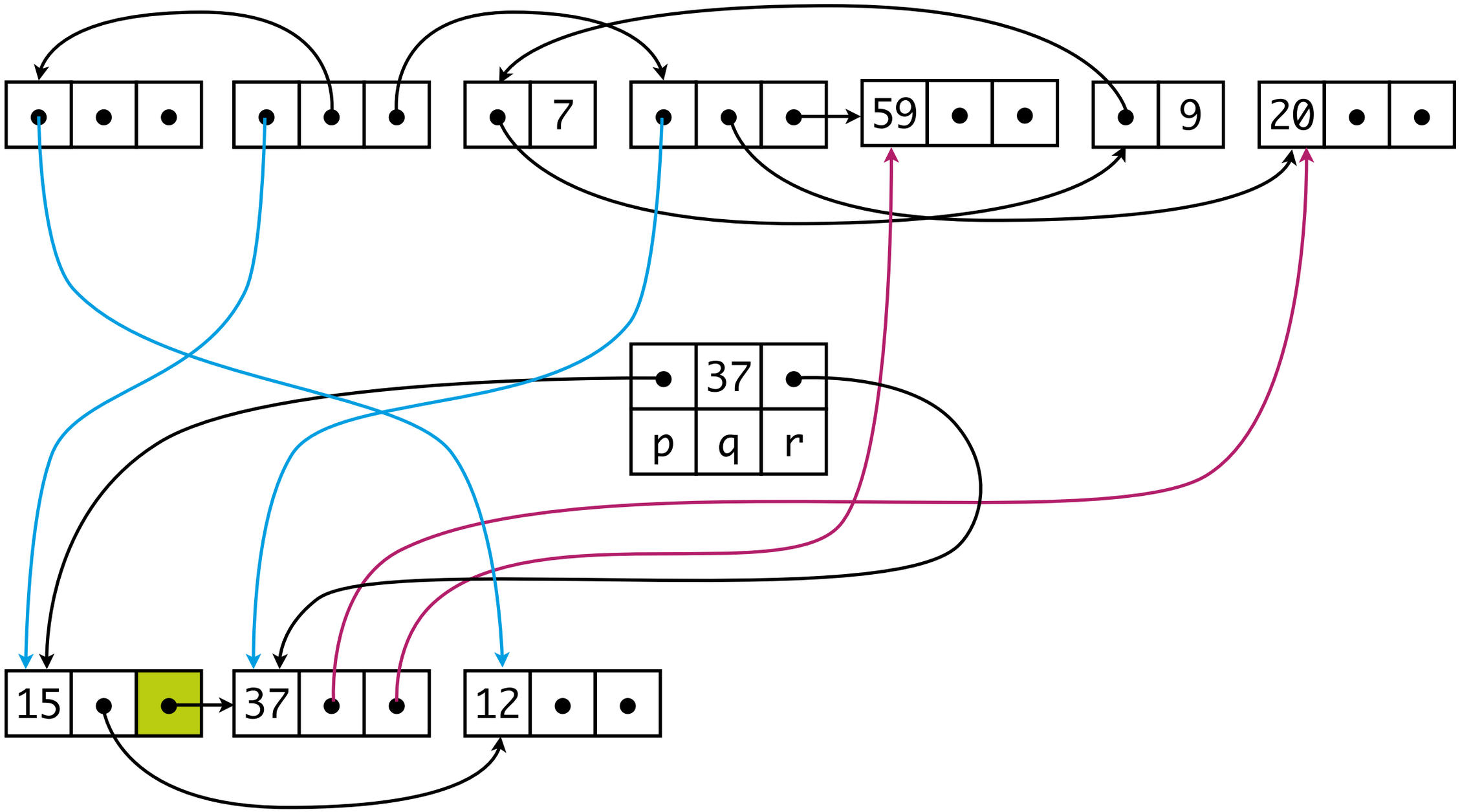
Copy collection



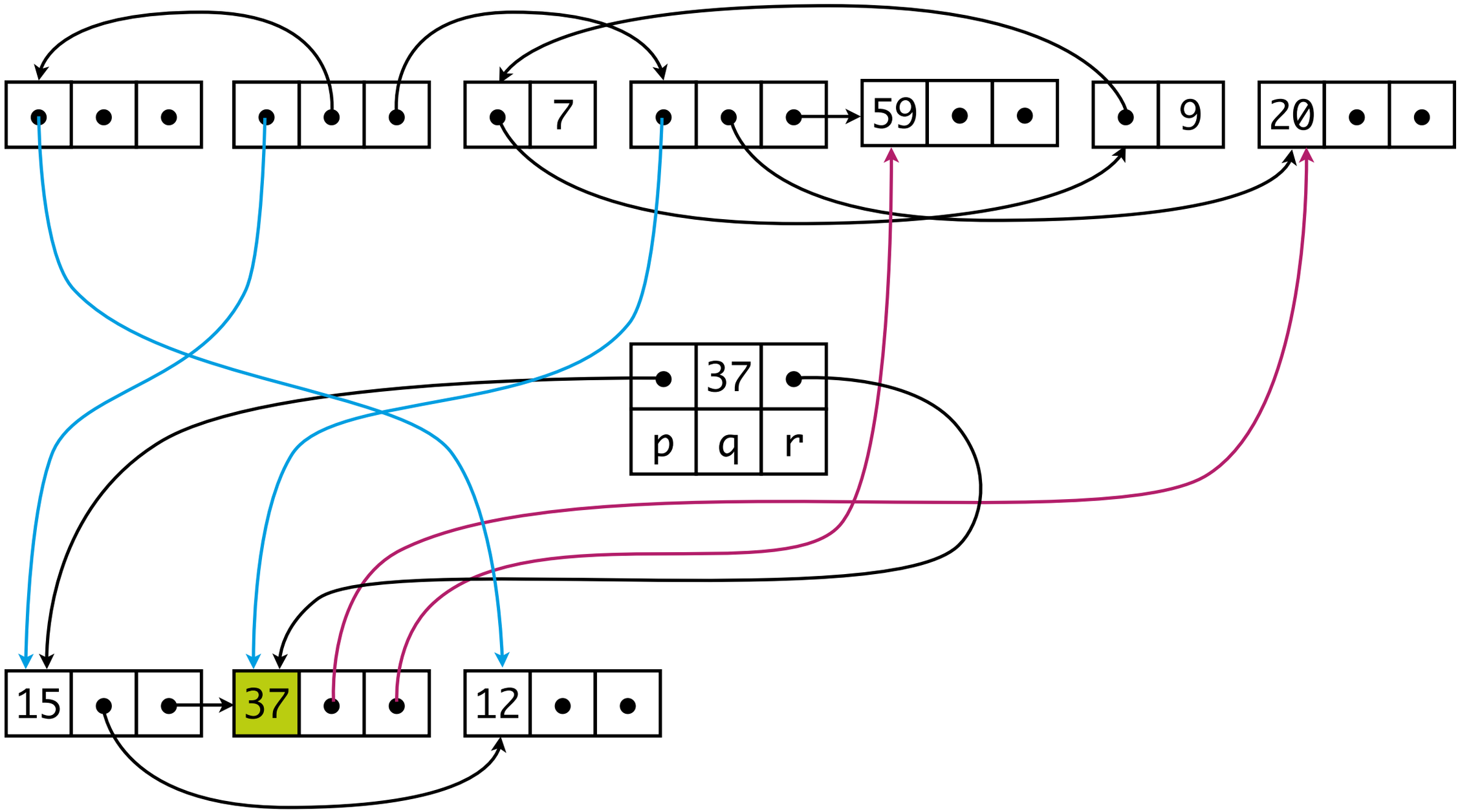
Copy collection



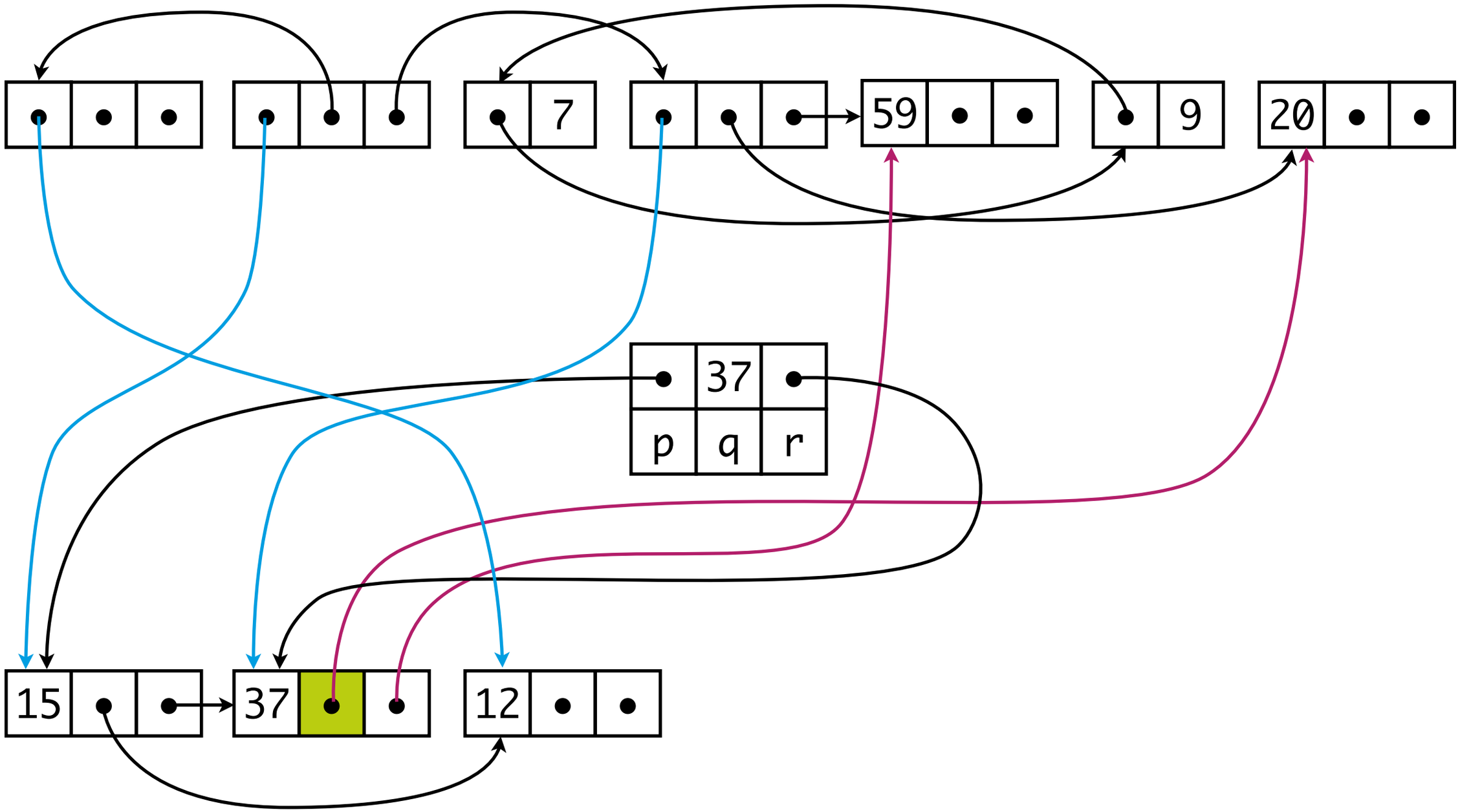
Copy collection



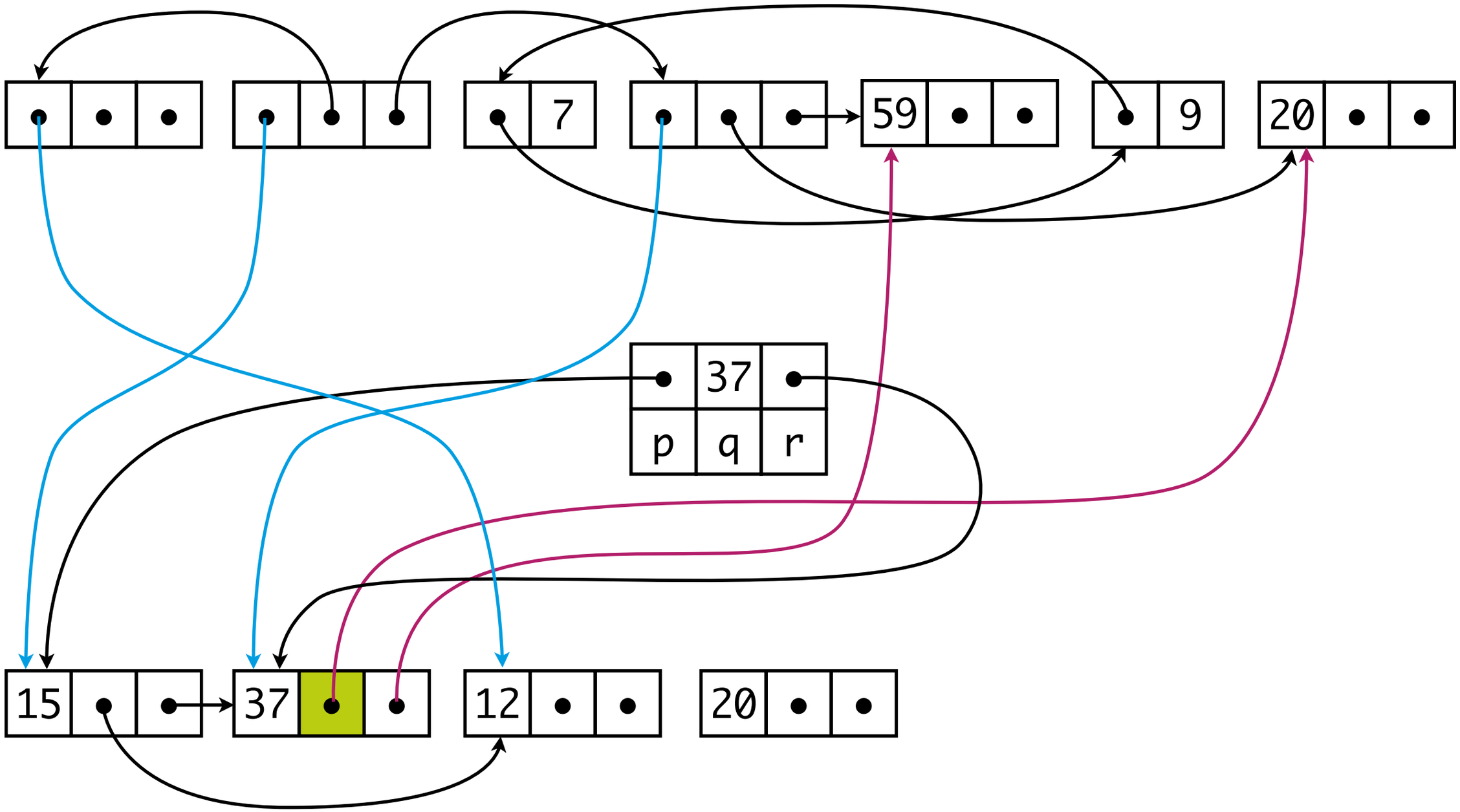
Copy collection



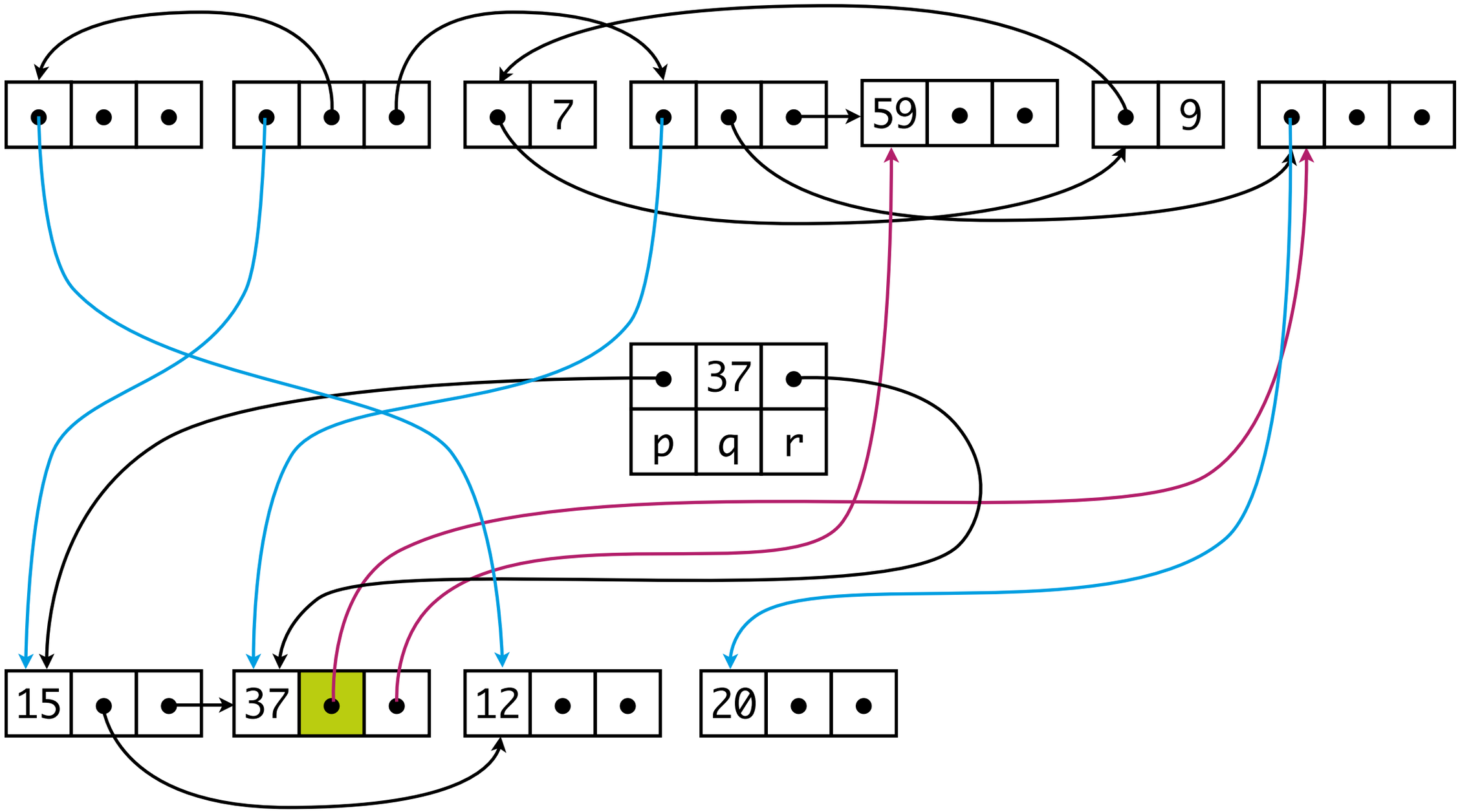
Copy collection



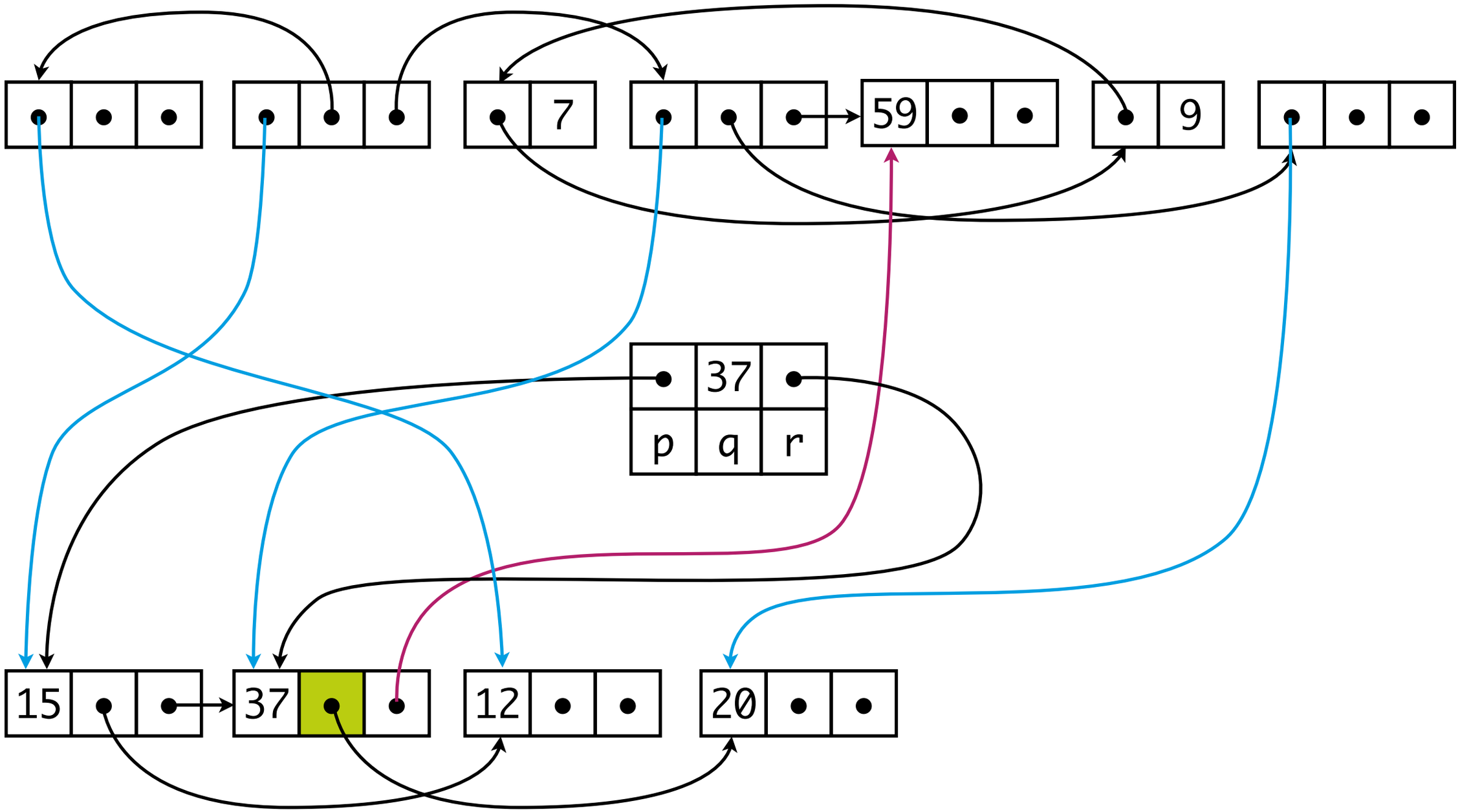
Copy collection



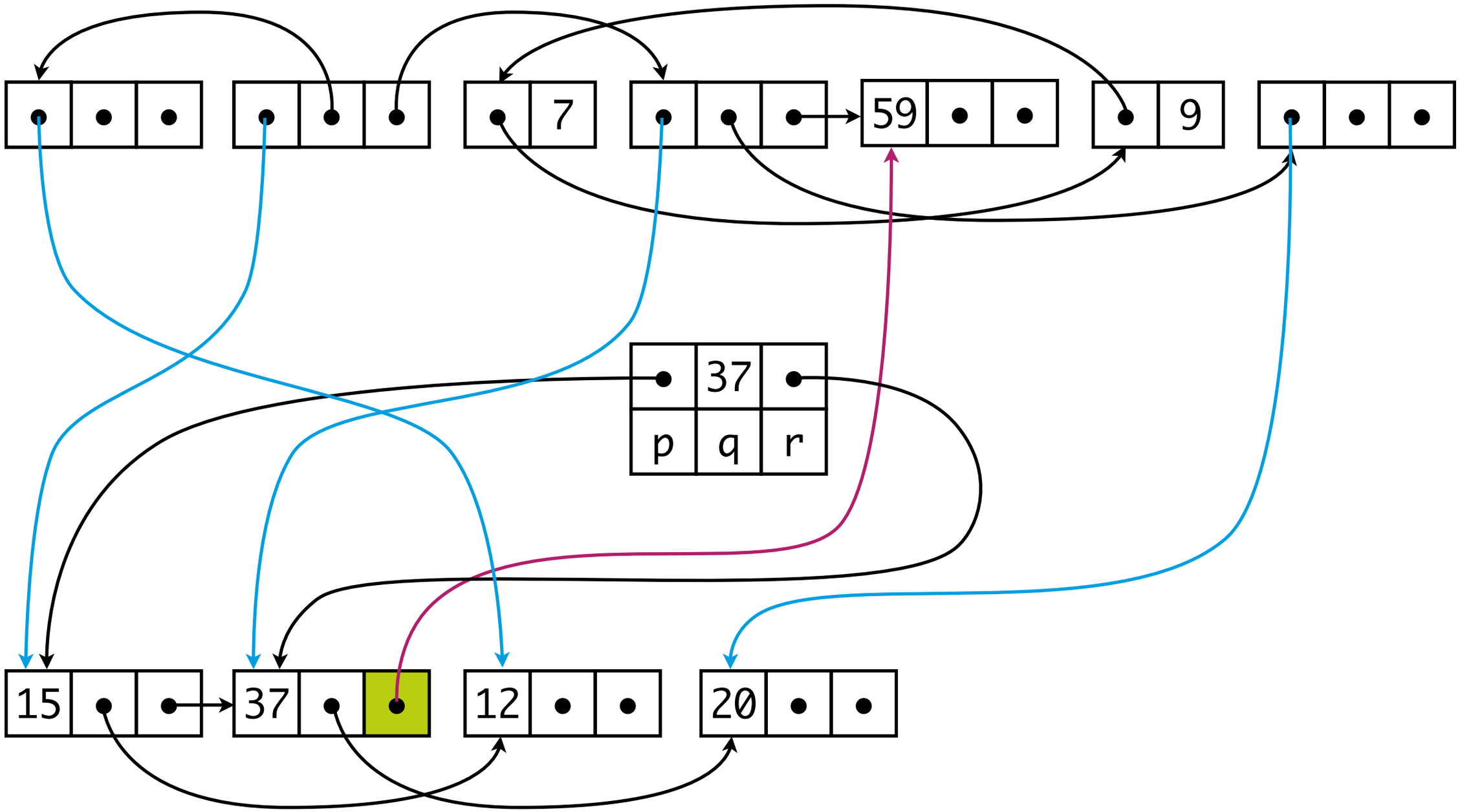
Copy collection



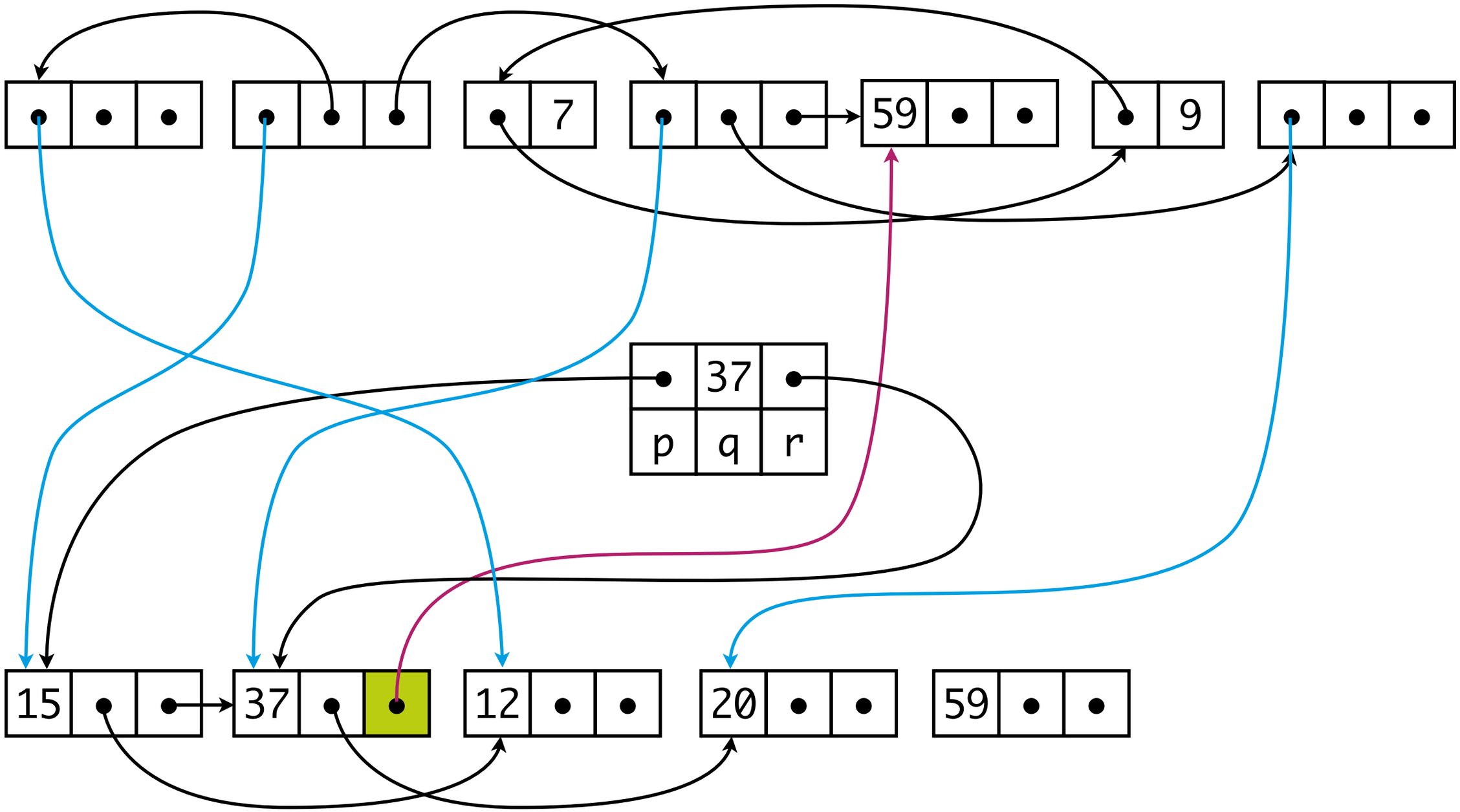
Copy collection



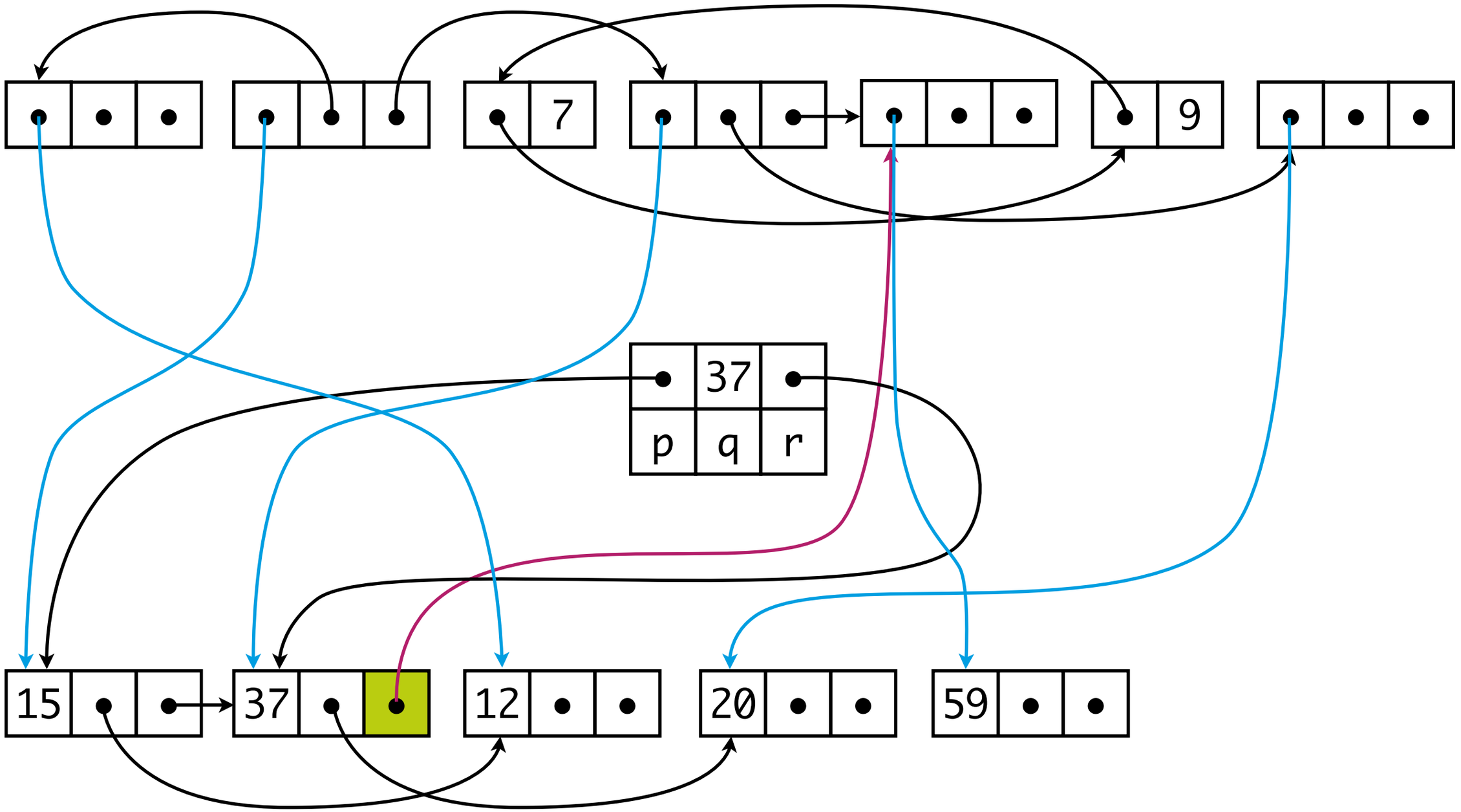
Copy collection



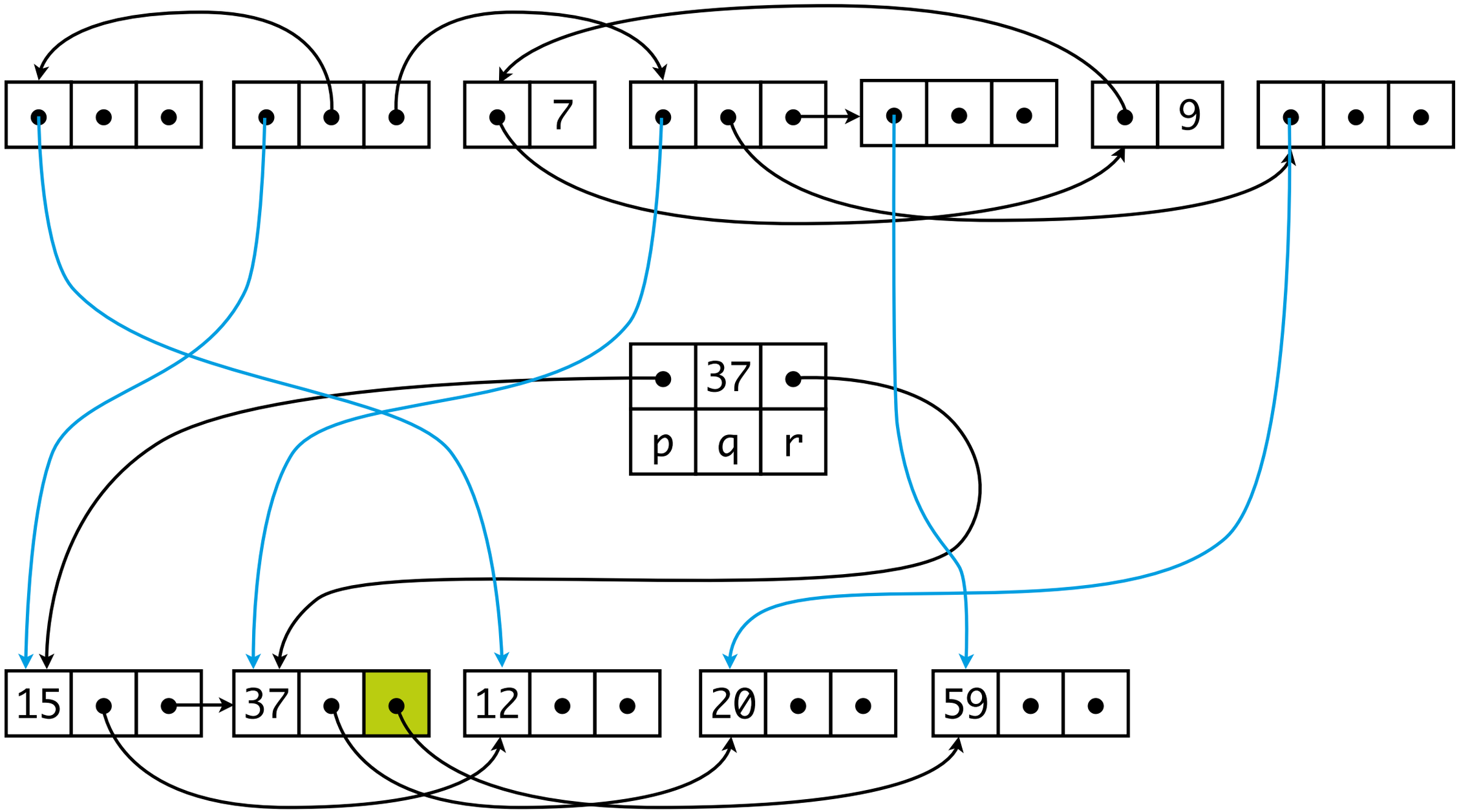
Copy collection



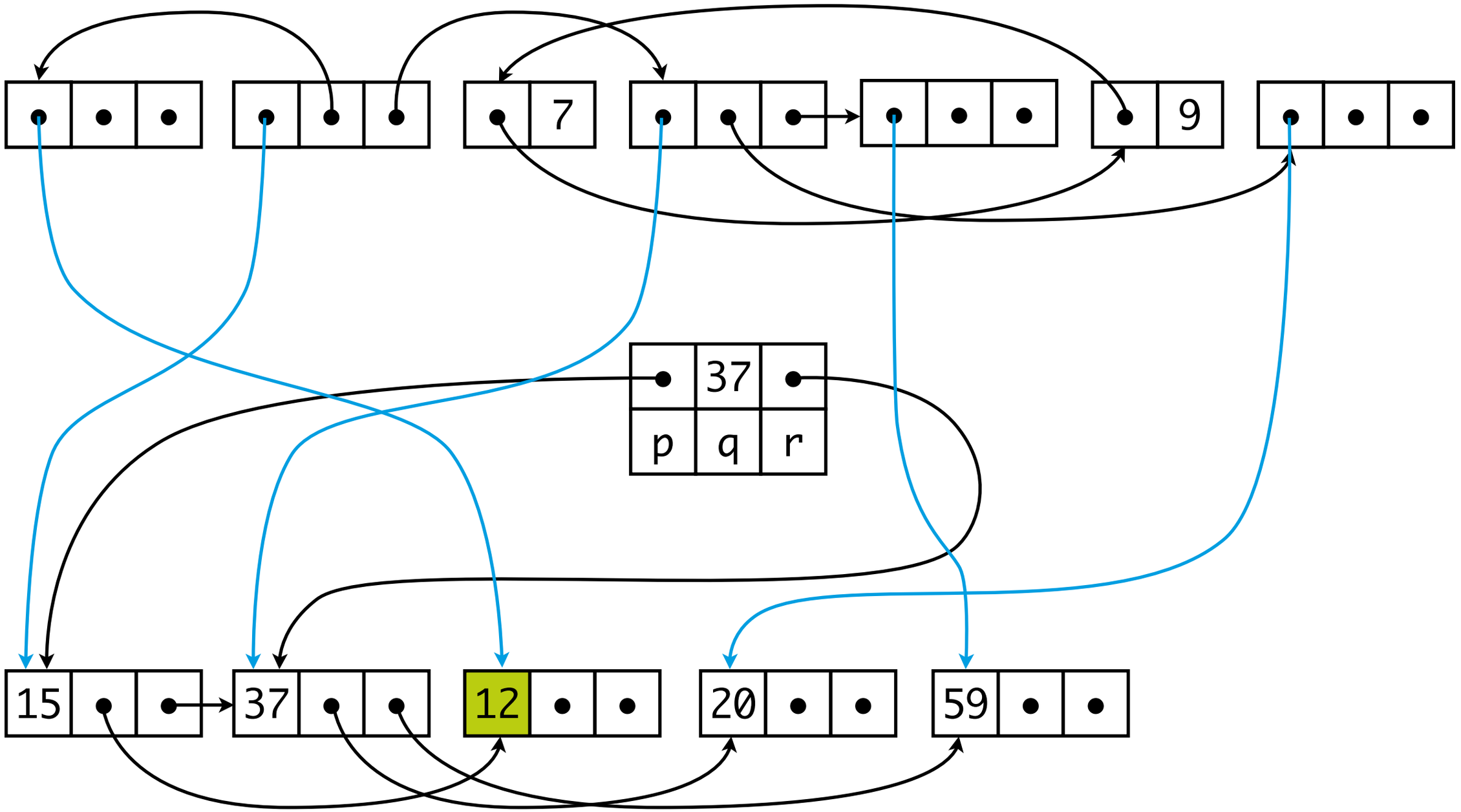
Copy collection



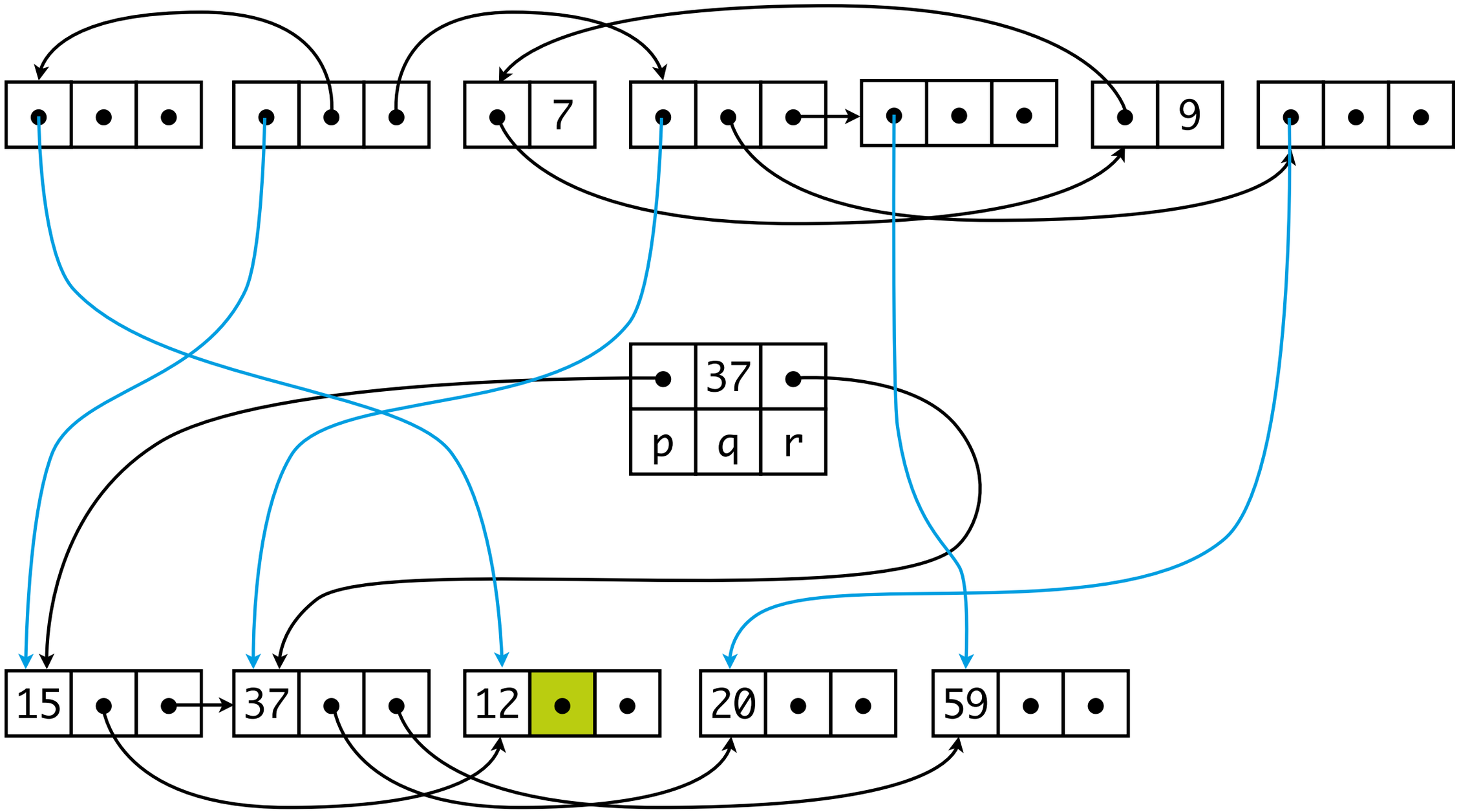
Copy collection



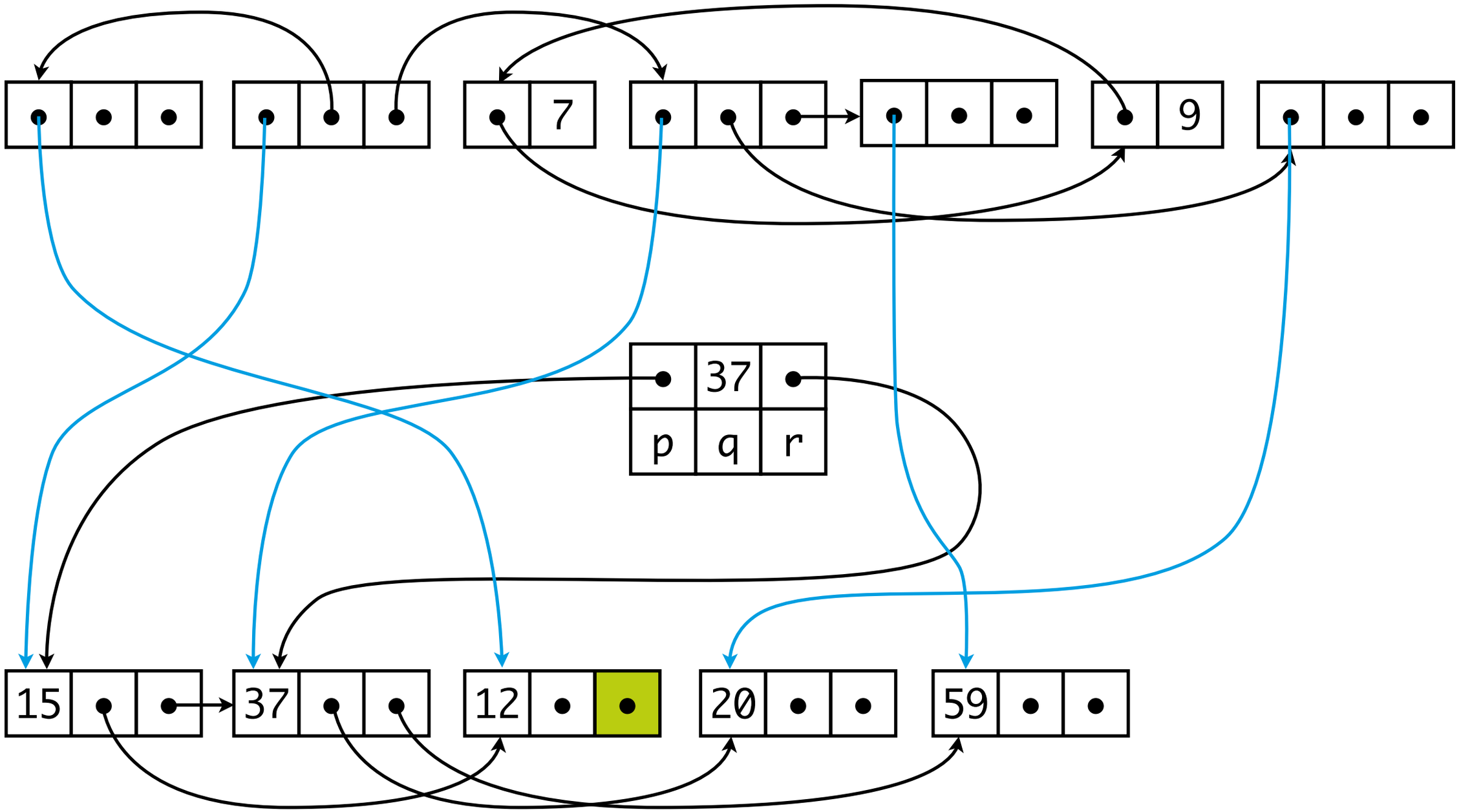
Copy collection



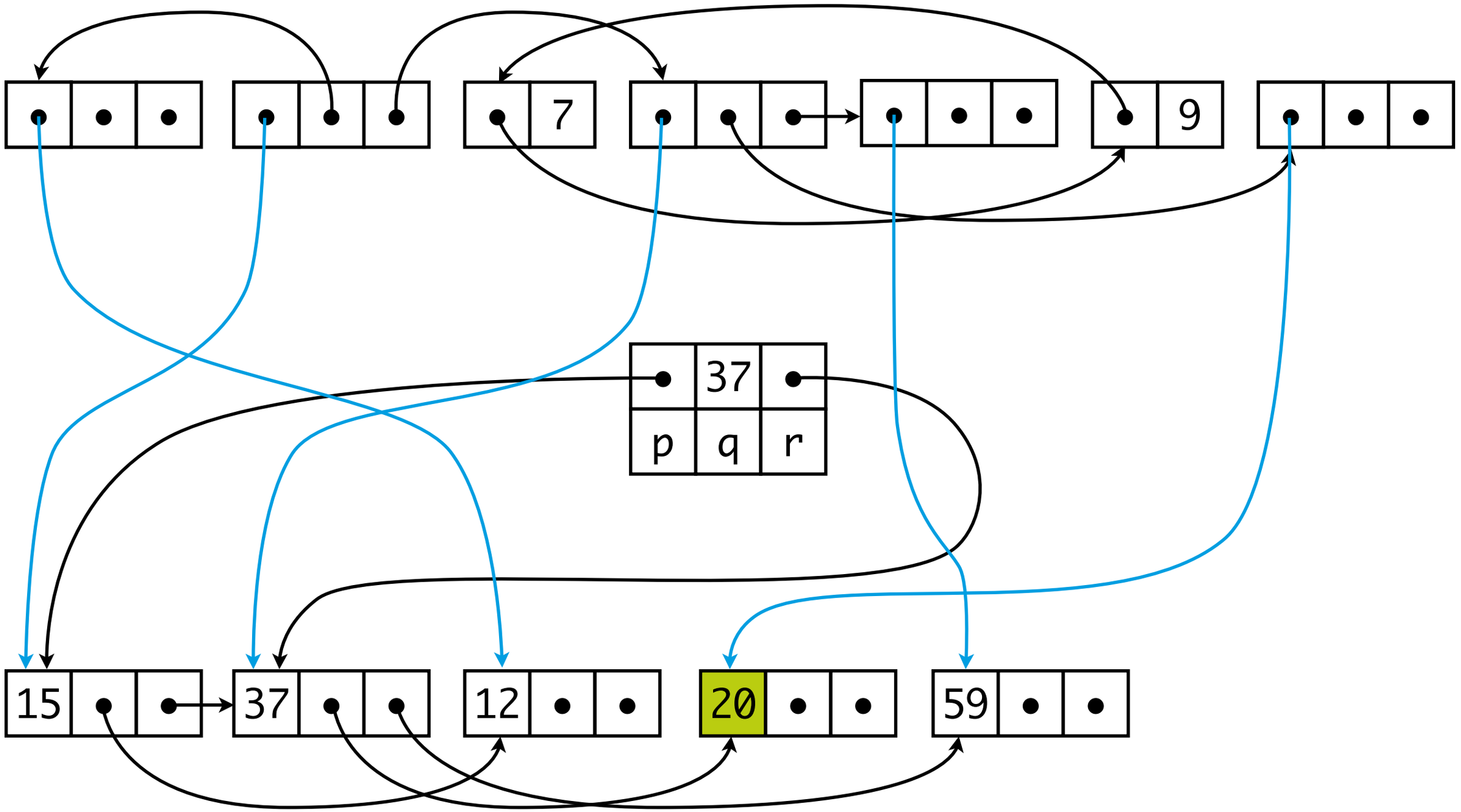
Copy collection



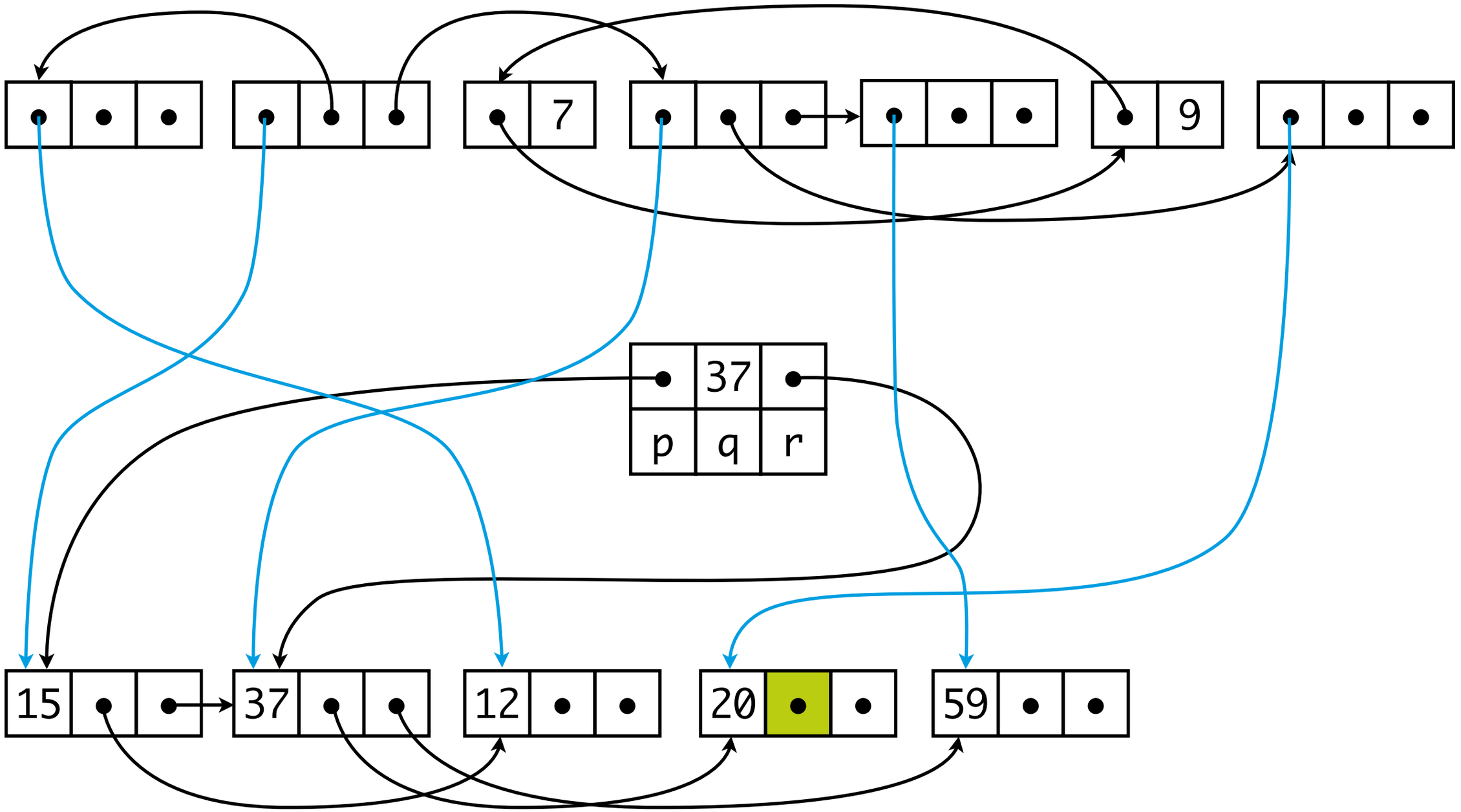
Copy collection



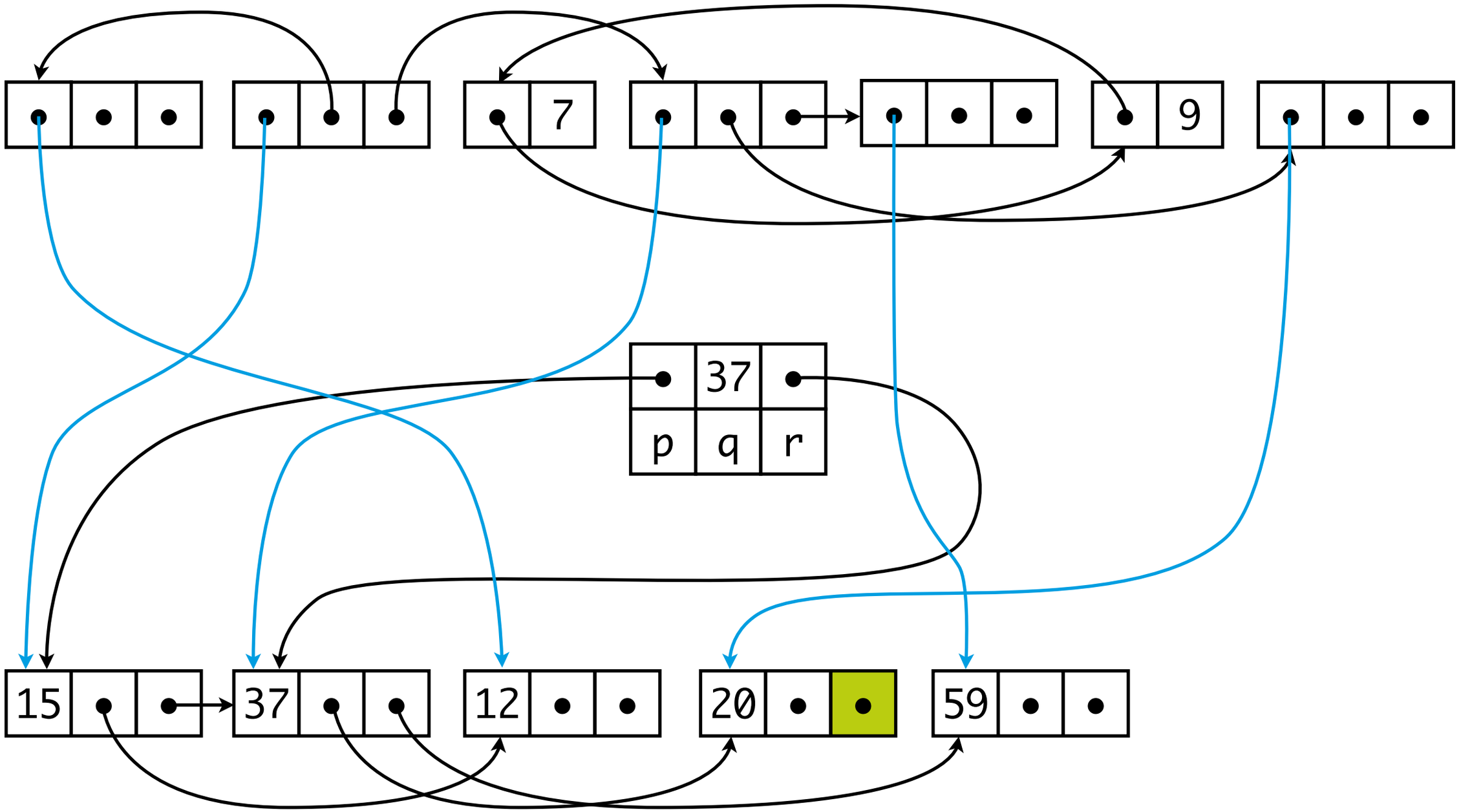
Copy collection



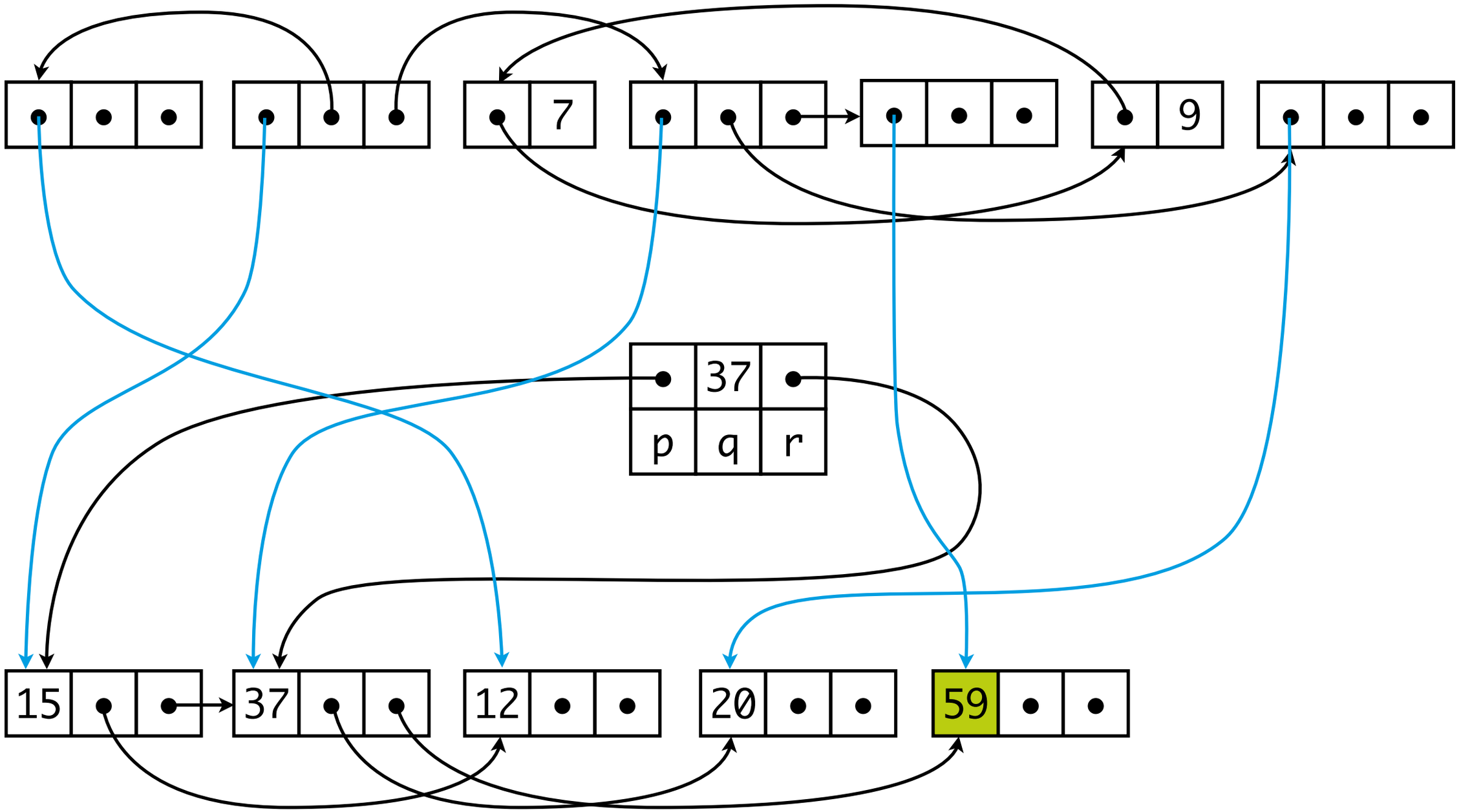
Copy collection



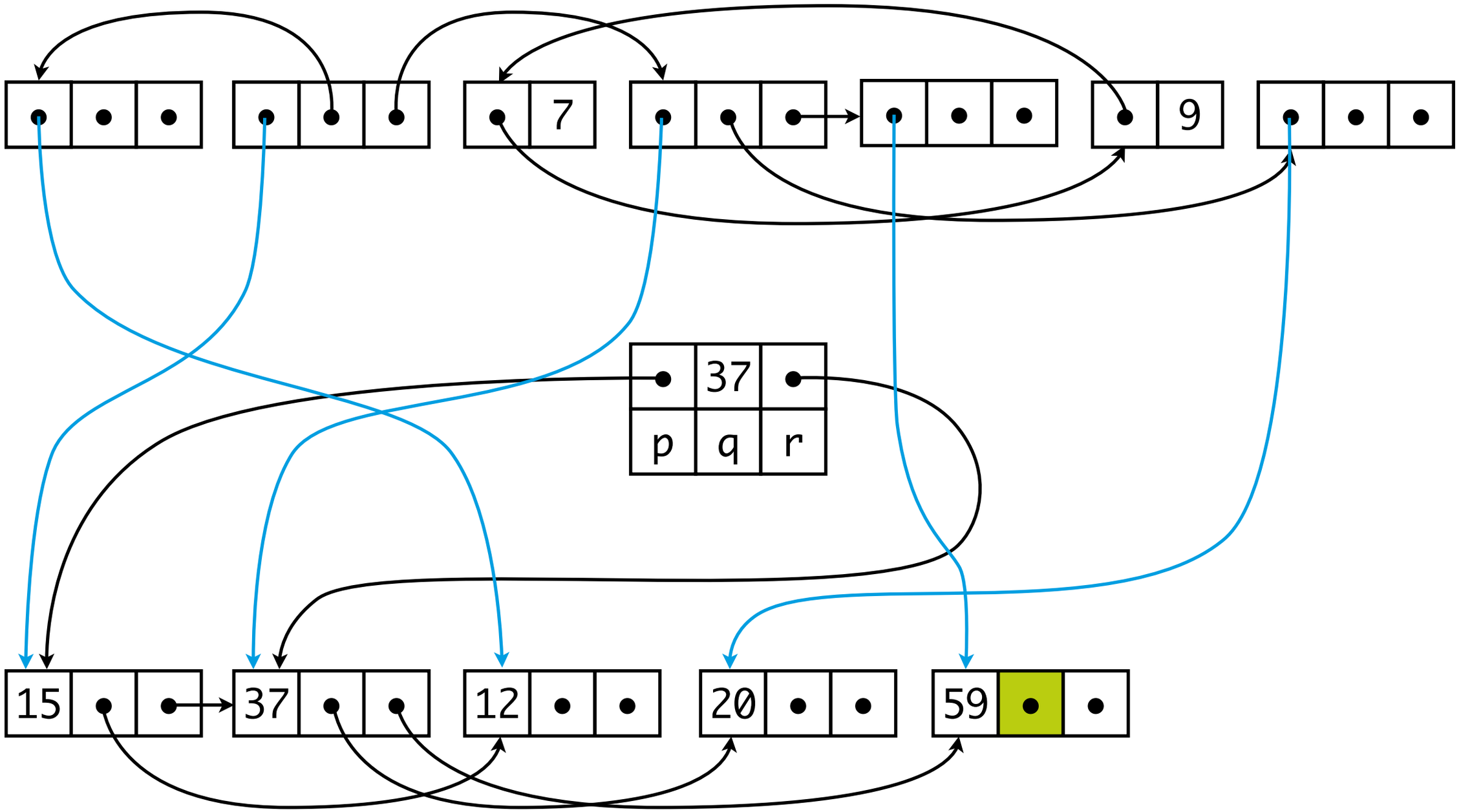
Copy collection



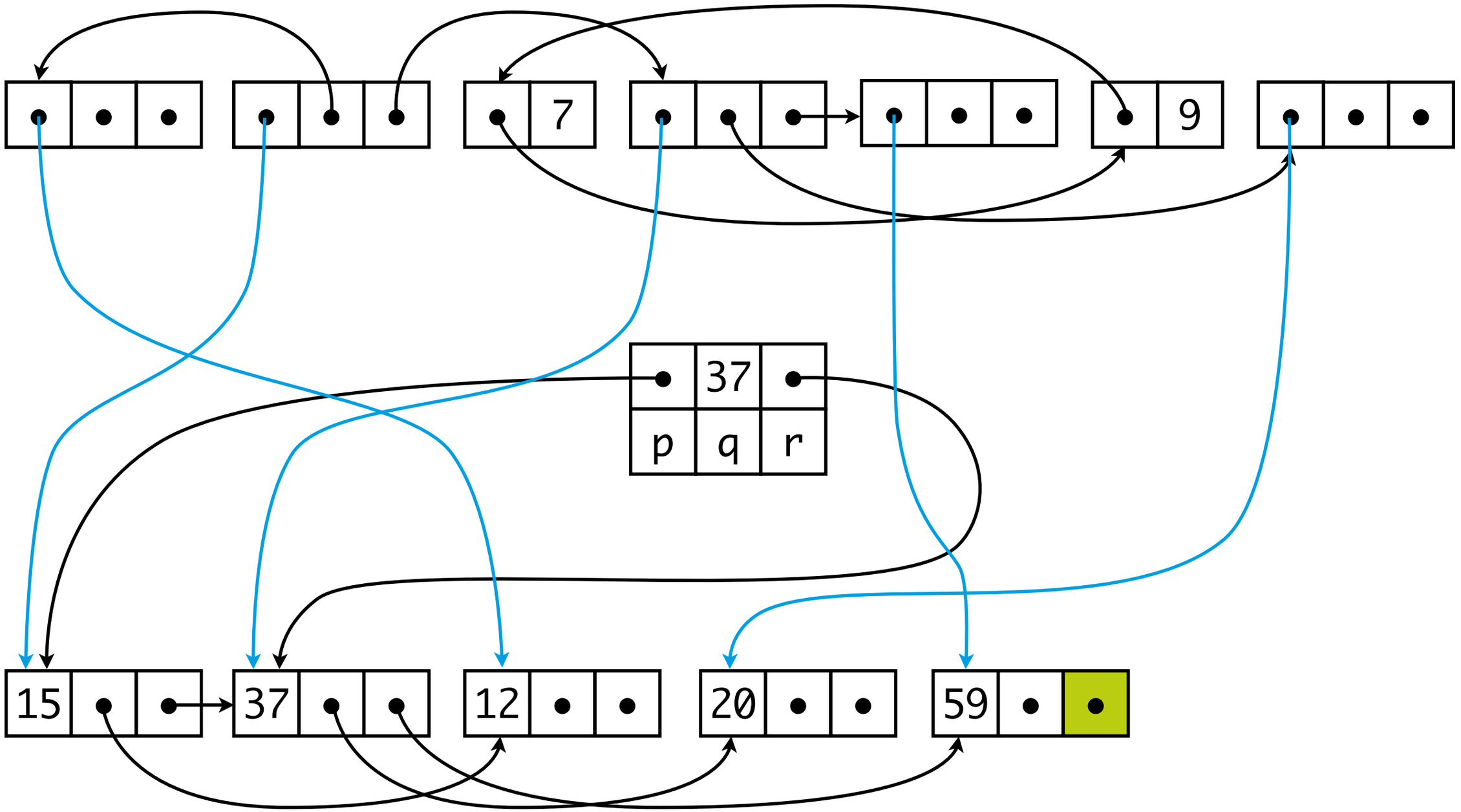
Copy collection



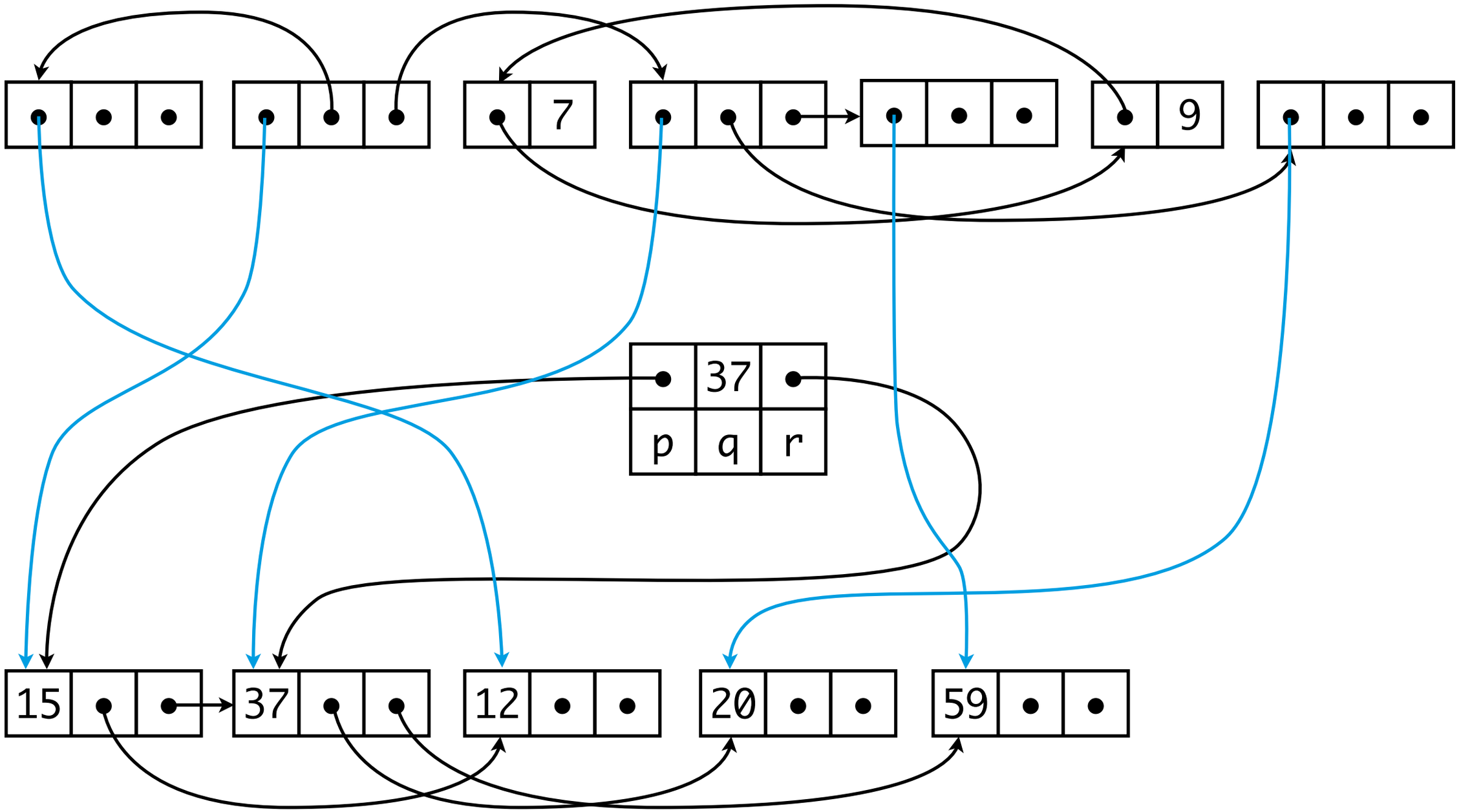
Copy collection



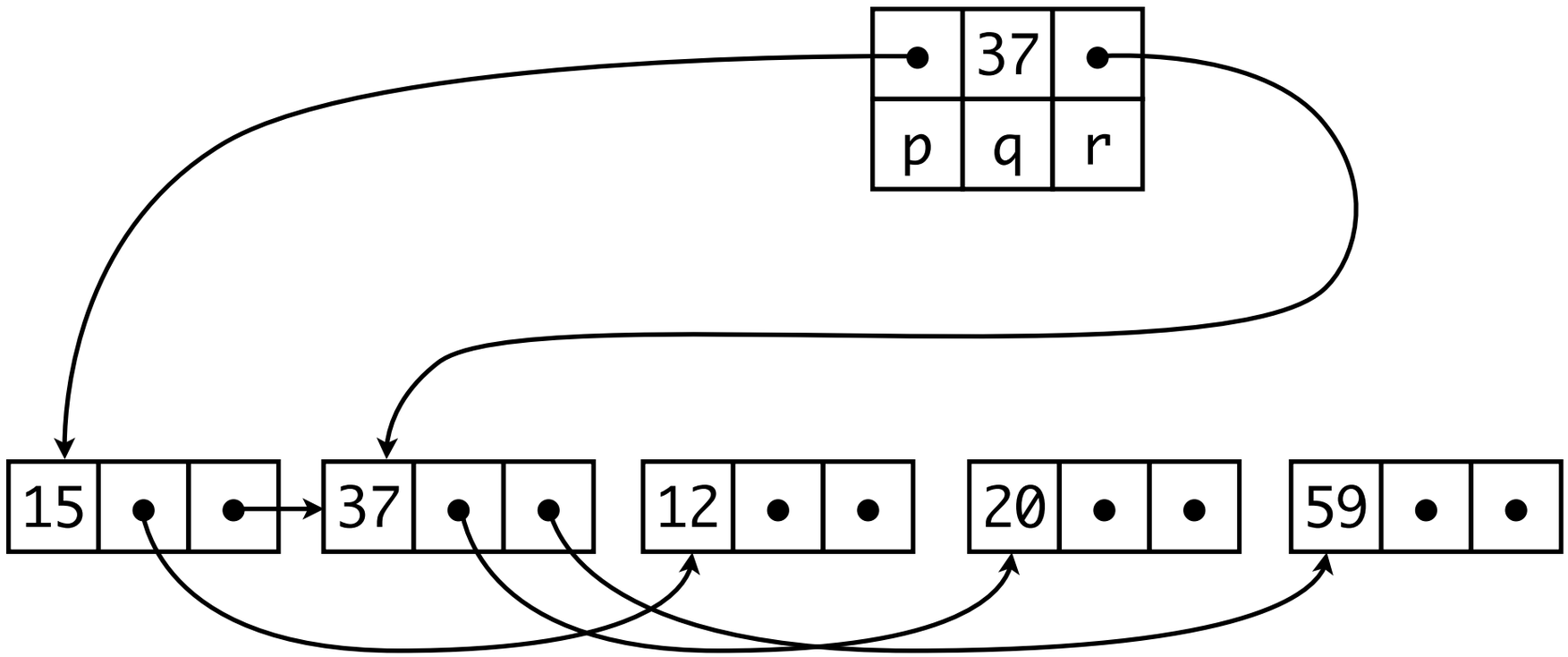
Copy collection



Copy collection



Copy collection



Copy collections
algorithms

```
function Forward(p)

  if fromspace(p)
    if tospace(fields(p)[1])
      return fields(p)[1]

    else

      foreach f in fields(p)
        next.f = f

      fields(p)[1] = next
      next = next + size(p)
      return fields(p)[1]

  else return p
```

```
function BFS()

  next = scan = start(tospace)

  foreach r in root()
    r = Forward(r)

  while scan < next

    foreach f in fields(scan)
      f = Forward(f)

    scan = scan + size(scan)
```

Copy collections
locality

adjacent records

- likely to be unrelated

pointers to records in records

- likely to be accessed
- likely to be far apart

solution

- depth-first copy: slow pointer reversals
- hybrid copy algorithm

Copy collections

generational collection

generations

- young data: likely to die soon
- old data: likely to survive for more collections
- divide heap, collect younger generations more frequently

collection

- roots: variables & pointers from older to younger generations
- preserve pointers to old generations
- promote objects to older generations

V

summary

Summary
lessons learned

How can we collect unreachable records on the heap?

- reference counts
- mark reachable records, sweep unreachable records
- copy reachable records

How can we reduce heap space needed for garbage collection?

- pointer-reversal
- breadth-first search
- hybrid algorithms

Literature

learn more

Andrew W. Appel, Jens Palsberg: Modern Compiler Implementation in Java, 2nd edition. 2002

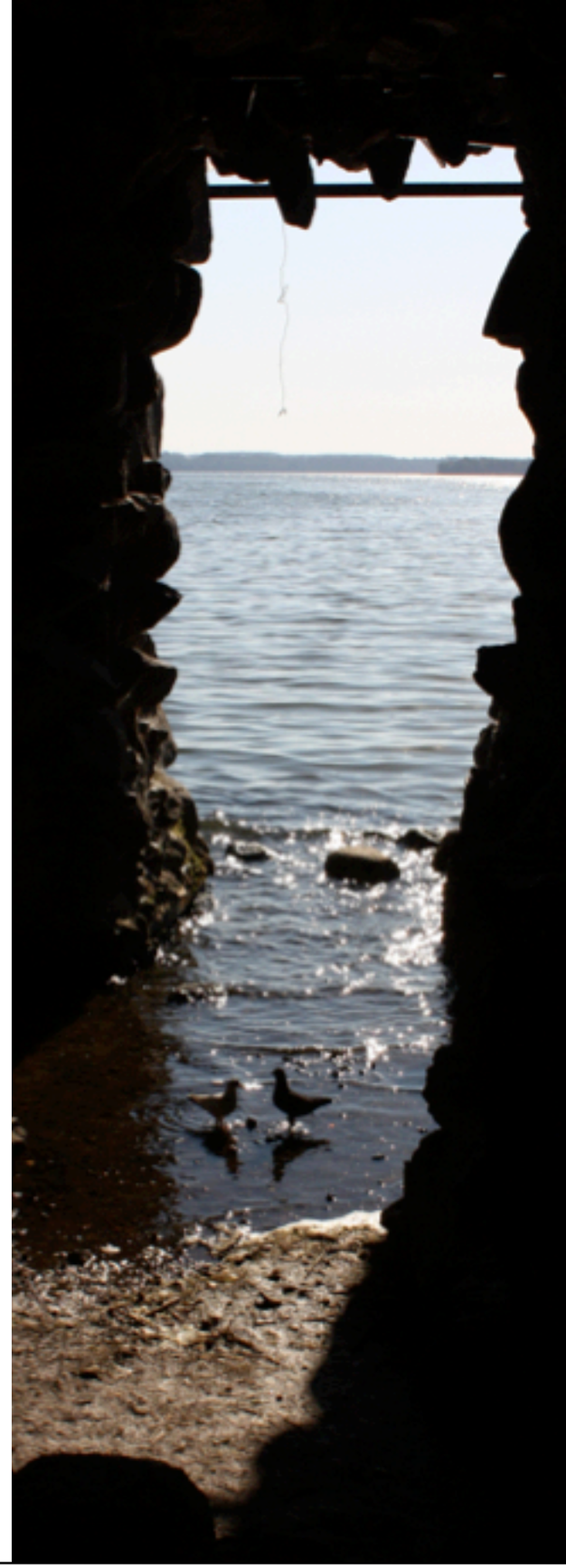
Outlook
coming next

compiler components and their generators

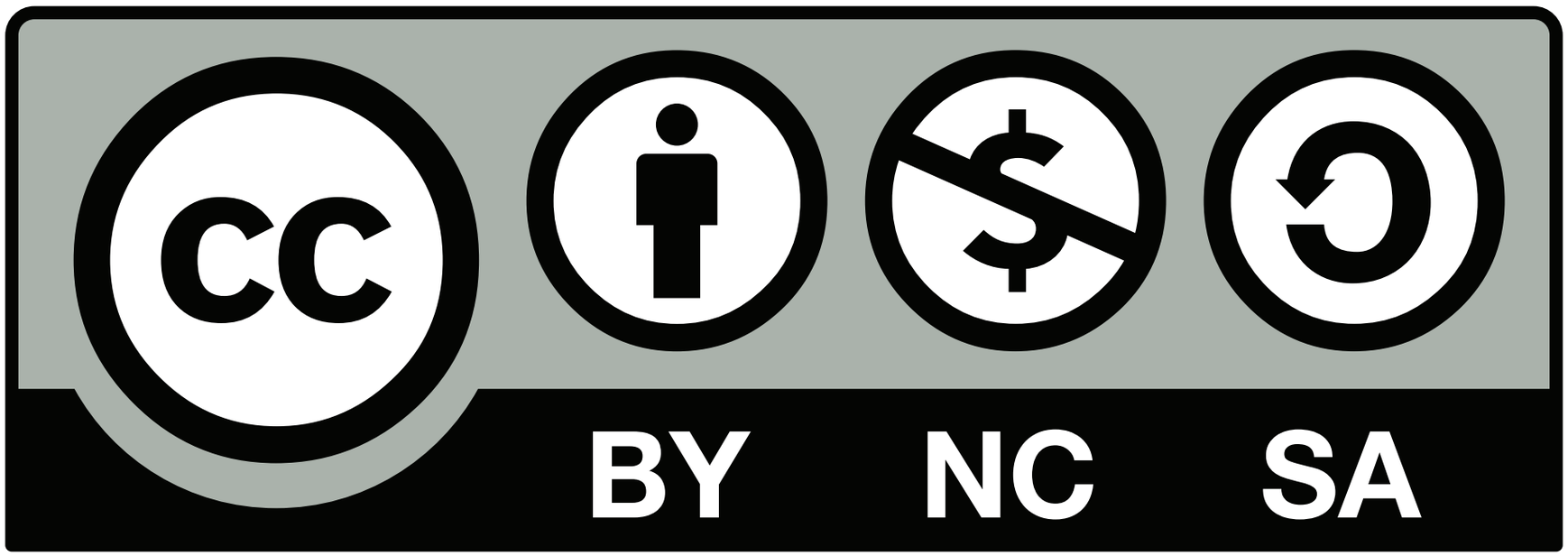
- Lecture 11: Lexical Analysis
- Lecture 12 & 13: Syntactical Analysis

Lab Dec 01

- store method references
- static analysis for method declarations
- static analysis for method calls
- overloading vs. overriding
- parameter types
- return types



copyrights


BY NC SA

Pictures

attribution & copyrights

Slide 1:
Trash by Vladimer Shioshvili, some rights reserved

Slide 4:
Gravel Pile by kenjonbro, some rights reserved

Slide 19:
Typhoo by Dominica Williamson, some rights reserved

Slide 30:
Romantic Pigeon Date by Harald Hoyer, some rights reserved