Melhores momentos

AULA 16

Ordenação

$\mathtt{v}[0\,..\,\mathtt{n}-1]$ é crescente se $\mathtt{v}[0] \le \cdots \le \mathtt{v}[\mathtt{n}-1]$.

Problema: Rearranjar um vetor $\mathtt{v}[0\,..\,\mathtt{n}-1]$ de modo que ele fique crescente.

Entra:

1										$\mathtt{n}-1$
33	55	33	44	33	22	11	99	22	55	77

Ordenação

$\mathtt{v}[0\,..\,\mathtt{n}-1]$ é crescente se $\mathtt{v}[0] \le \cdots \le \mathtt{v}[\mathtt{n}-1]$.

Problema: Rearranjar um vetor $\mathtt{v}[0\,..\,\mathtt{n}-1]$ de modo que ele fique crescente.

Entra:

1										$\mathtt{n}-1$
33	55	33	44	33	22	11	99	22	55	77

Sai:

0										$\mathtt{n}-1$
11	22	22	33	33	33	44	55	55	77	99

insercao

Função rearranja $v[0\,..\,n-1]$ em ordem crescente.

```
void insercao (int n, int v[])
{
    int i, j, x;
1   for (i = 1; /*A*/ i < n; i++){
2       x = v[i];
3       for (j=i-1; j >= 0 && v[j] > x;j--)
4           v[j+1] = v[j];
5       v[j+1] = x;
    }
}
```

insercaoBinaria

Função rearranja $v[0..n-1]$ em ordem crescente.

```
void insercaoBinaria (int n, int v[])
{
    int i, j, k, x;
1   for (i = 1; /*A*/ i < n; i++){
2       x = v[i];
3       j = buscaBinaria(x,i,v);
4       for (k = i; k > j+1; k--)
5           v[k] = v[k-1];
6       v[j+1] = x;
    }
}
```

Função selecao

Algoritmo rearranja $v[0\,..\,n-1]$ em ordem crescente

```
void selecao (int n, int v[])
{
    int i, j, max, x;
1   for (i = n-1; /*A*/ i > 0; i--) {
2       max = i;
3       for (j = i-1; j >= 0; j--)
4           if (v[j] > v[max]) max = j;
5       x=v[i]; v[i]=v[max]; v[max]=x;
    }
}
```

AULA 17

Ambiente experimental

A plataforma utilizada nos experimentos foi um computador rodando Ubuntu GNU/Linux 3.5.0-17

As especificações do computador que geraram as saídas a seguir são

```
model name: Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz
cpu MHz   : 1596.000
cache size: 4096 KB

MemTotal  : 3354708 kB
```

Ambiente experimental

Os códigos foram compilados com o gcc 4.7.2 e com opções de compilação

-Wall -ansi -O2 -pedantic -Wno-unused-result

As implementações comparadas neste experimento são `bubble`, `selecao`, `insercao` e `insercaoBinaria`,

Ambiente experimental

A estimativa do tempo é calculada utilizando-se:

```
#include <time.h>
[...]
clock_t start, end;
double time;

start = clock();

[...implementação...]

end  = clock();
time = ((double)(end - start))/CLOCKS_PER_SEC;
```

Resultados experimentais: aleatórios

n	bubble	selecao	insercao	insercaoB
1024	0.00	0.00	0.00	0.00
2048	0.01	0.00	0.00	0.00
4096	0.03	0.01	0.00	0.00
8192	0.12	0.04	0.01	0.01
16384	0.51	0.17	0.05	0.03
32768	2.03	0.68	0.23	0.17
65536	8.12	2.70	0.90	0.69
131072	32.51	10.80	3.62	2.80
262144	130.05	43.14	14.49	11.26
524288	521.26	172.87	58.26	45.64

tempos em segundos

Resultados experimentais: crescente

n	bubble	selecao	insercao	insercaoB
1024	0.00	0.00	0.00	0.00
2048	0.00	0.00	0.00	0.00
4096	0.01	0.01	0.00	0.00
8192	0.03	0.04	0.00	0.00
16384	0.12	0.17	0.00	0.00
32768	0.48	0.67	0.00	0.00
65536	1.91	2.70	0.00	0.00
131072	7.67	10.77	0.00	0.00
262144	30.68	43.06	0.00	0.02
524288	123.11	172.57	0.00	0.02
1048576	500.89	696.91	0.00	0.06

tempos em segundos

Resultados experimentais: decrescente

n	bubble	selecao	insercao	insercaoB
1024	0.00	0.00	0.00	0.00
2048	0.01	0.00	0.00	0.00
4096	0.01	0.01	0.00	0.01
8192	0.04	0.04	0.03	0.01
16384	0.26	0.18	0.11	0.08
32768	1.12	0.72	0.45	0.34
65536	4.56	2.87	1.81	1.40
131072	18.23	11.47	7.24	5.64
262144	70.51	45.95	28.99	22.50
524288	203.44	183.87	116.93	92.19
1048576	754.52	742.56	493.33	405.10

tempos em segundos

Intercalação

Fonte: http://csunplugged.org/sorting-algorithms

PF 9

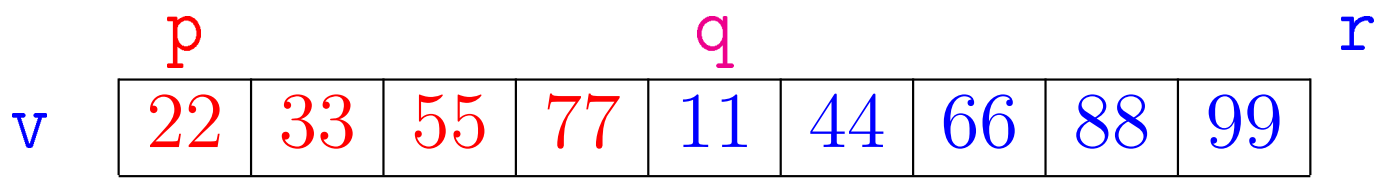
http://www.ime.usp.br/~pf/algoritmos/aulas/mrgsrt.html

Intercalação

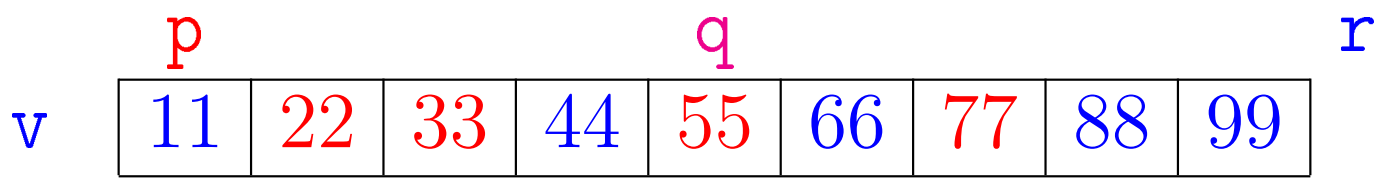
Problema: Dados $v[p\,..\,q-1]$ e $v[q\,..\,r-1]$ crescentes, rearranjar $v[p\,..\,r-1]$ de modo que ele fique em ordem crescente.

Para que valores de q o problema faz sentido?

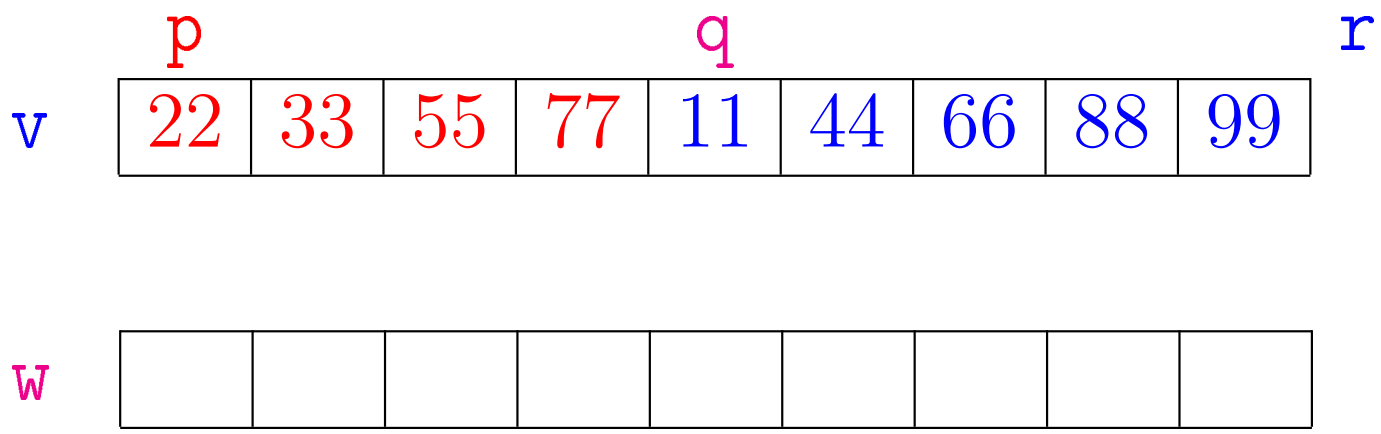
Entra:



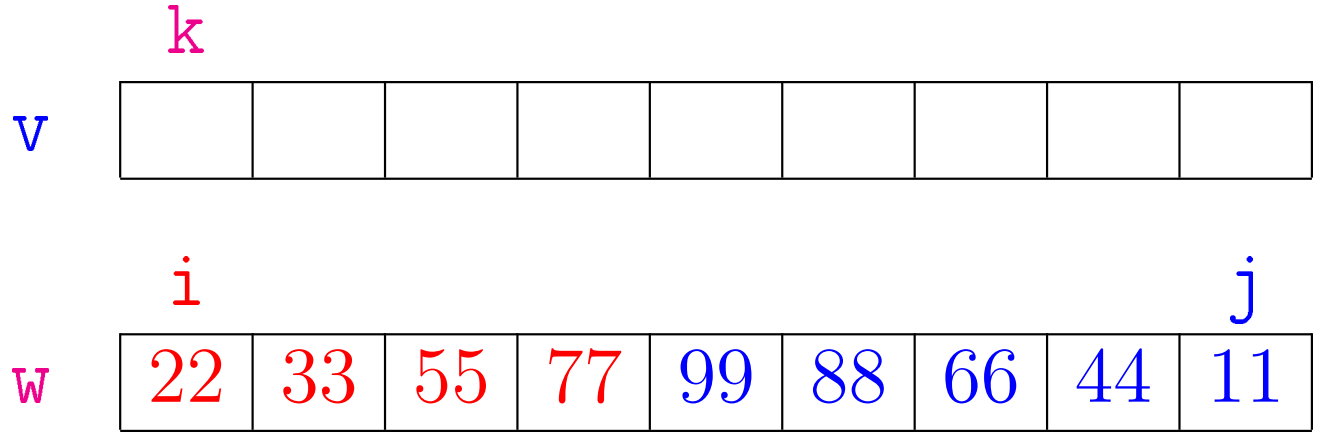
Sai:



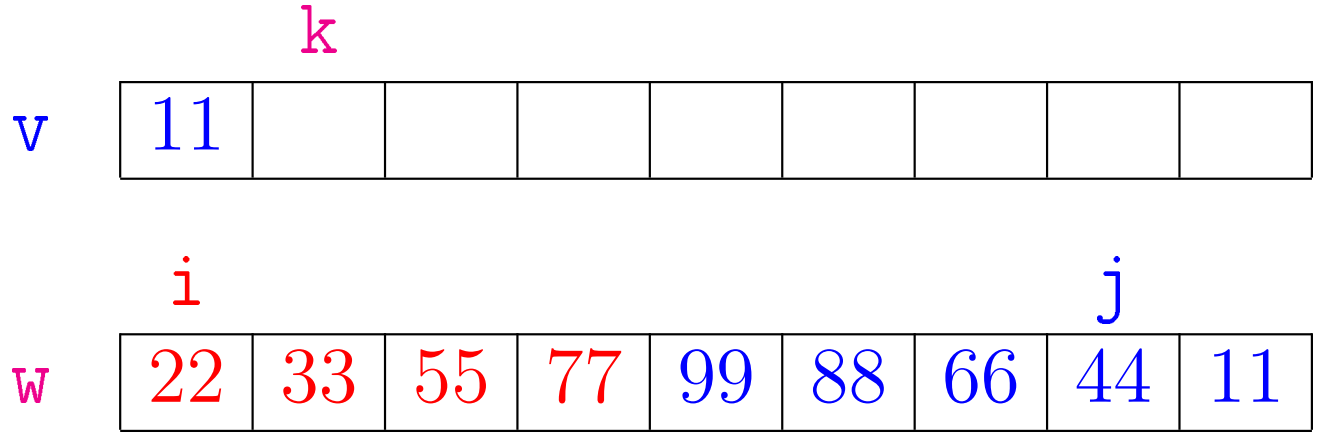
Intercalação



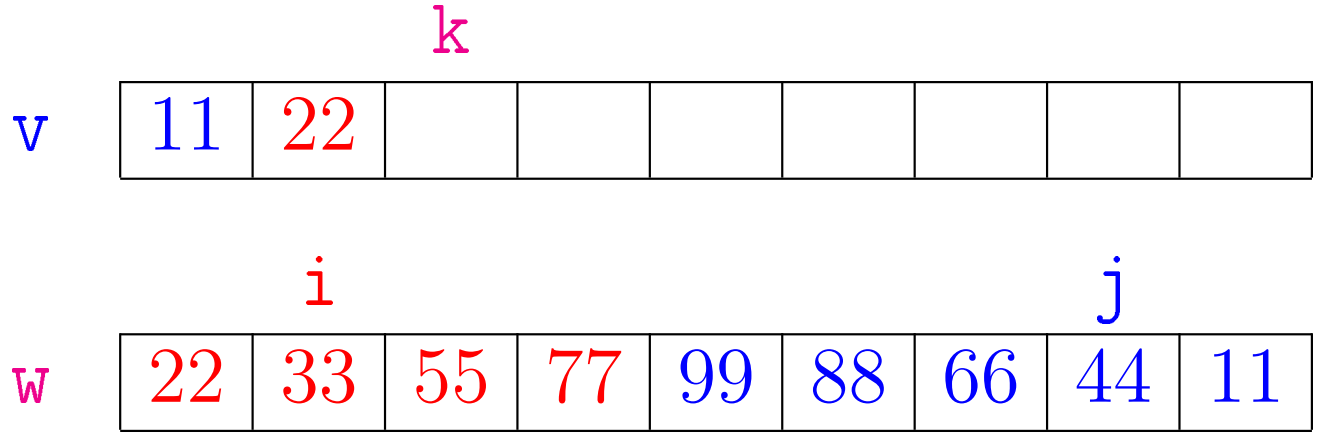
Intercalação



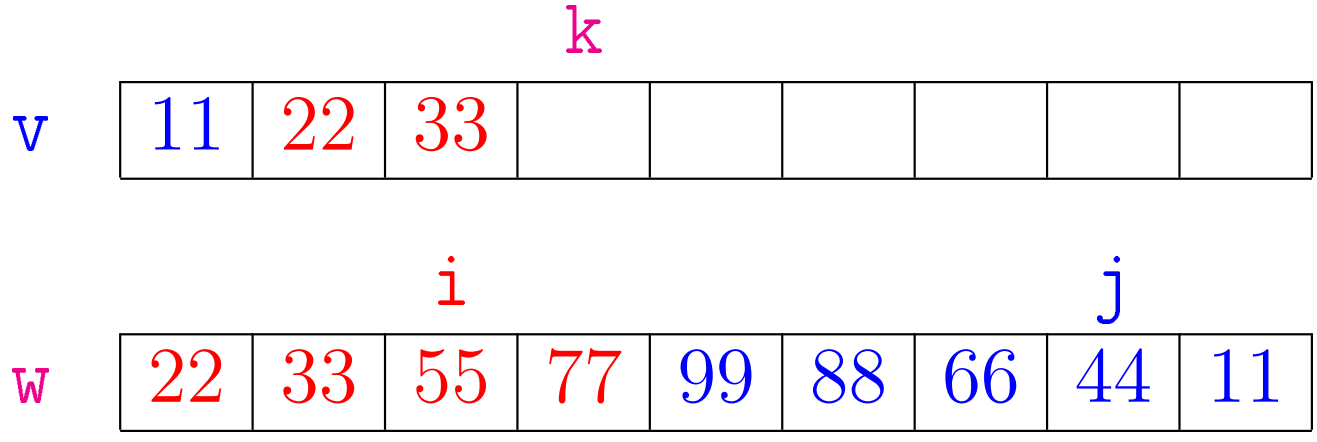
Intercalação



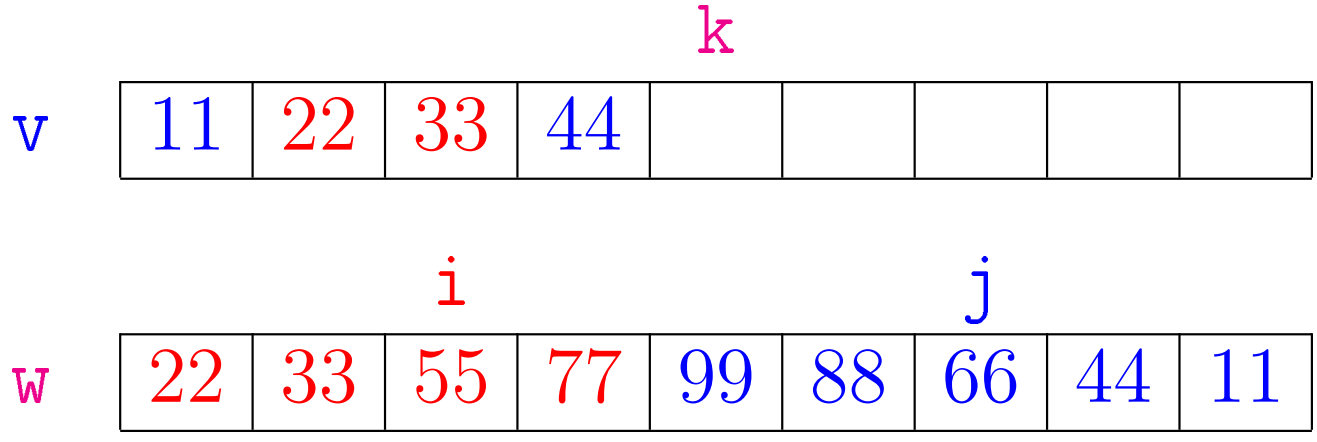
Intercalação



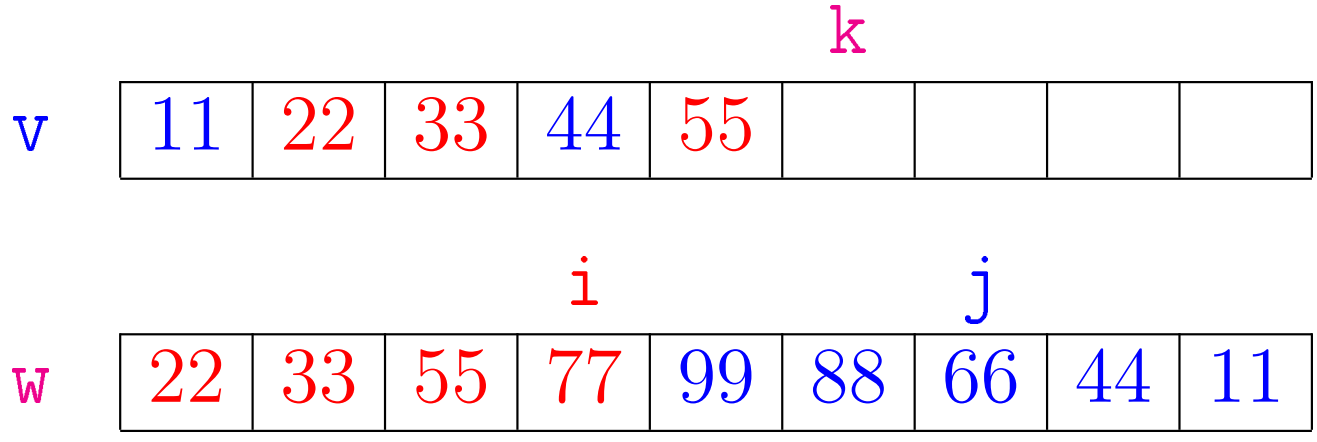
Intercalação



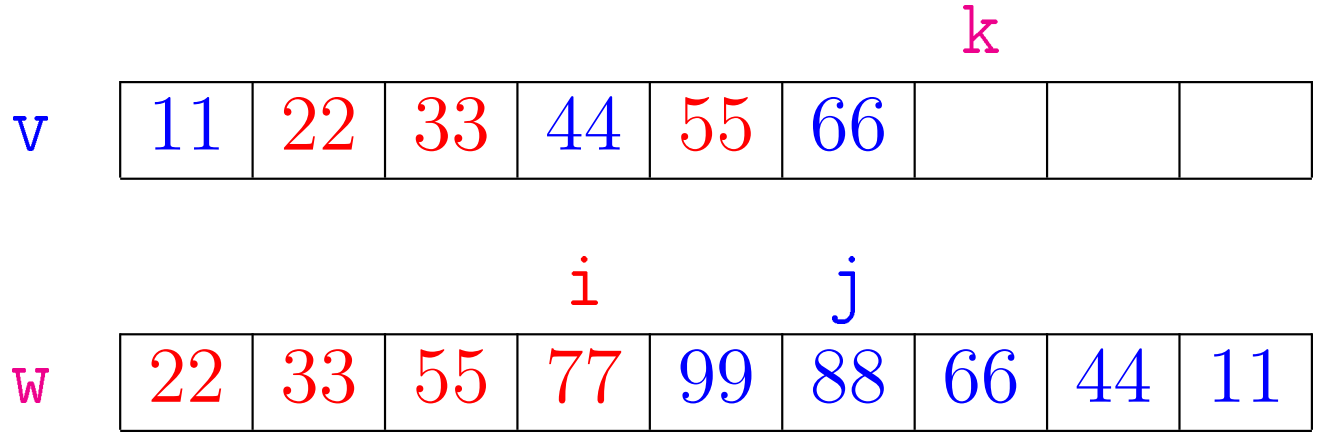
Intercalação



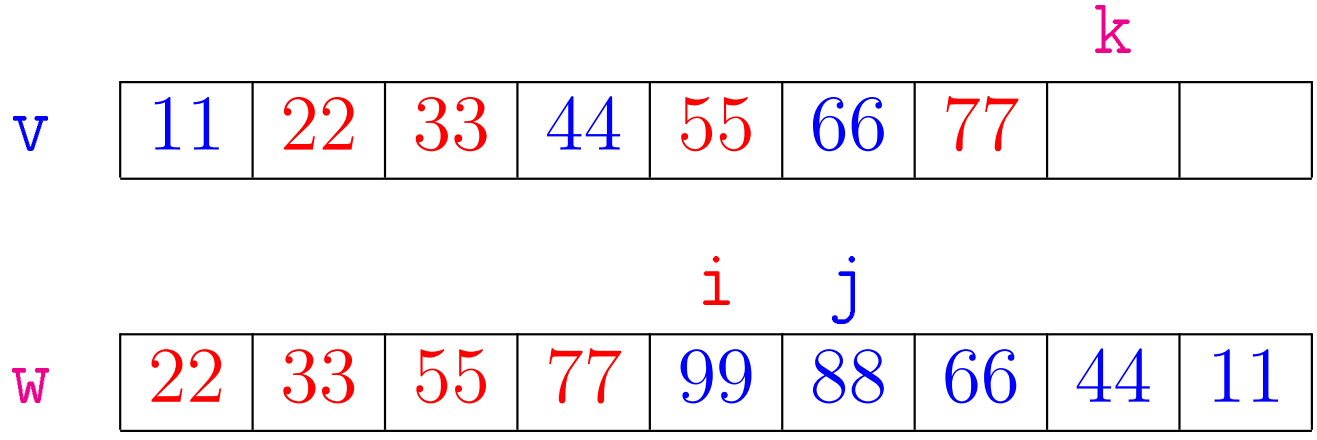
Intercalação



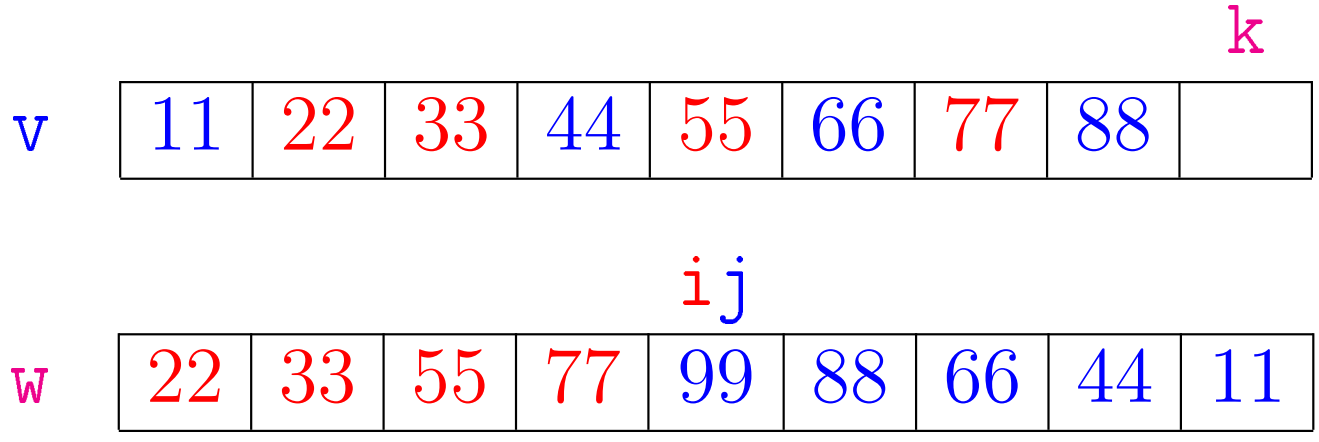
Intercalação



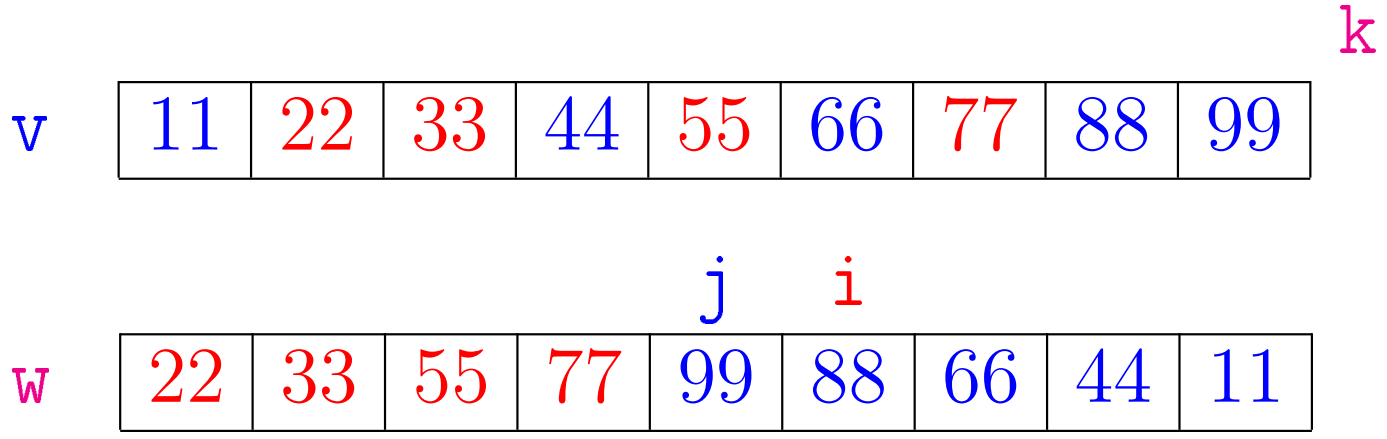
Intercalação



Intercalação



Intercalação



Intercalação

```
void intercala(int p,int q,int r,int v[]){
   int i, j, k, *w;
   w = mallocSafe((r-p)*sizeof(int));
1   for (i = 0, k = p; k < q; i++, k++)
2       w[i] = v[k];
3   for (j = r-p-1; k < r; j--, k++)
4       w[j] = v[k];
5   i = 0; j = r-p-1;
6   for (k = p; k < r; k++)
7       if (w[i] <= w[j])
8           v[k] = w[i++];
9       else v[k] = w[j--];
    free (w);
}
```

Consumo de tempo

Se a execução de cada linha de código consome 1 unidade de tempo o consumo total é:

linha	execuções da linha
1	?
2	?
3	?
4	?
5	?
6	?
7	?
8–9	?
total	?

Consumo de tempo

Se a execução de cada linha de código consome 1 unidade de tempo o consumo total é $(\mathtt{n} := \mathtt{r} - \mathtt{p})$:

linha	execuções da linha		
1	$=$	$\mathtt{q} - \mathtt{p} + 1$	$= \mathtt{n} - \mathtt{r} + \mathtt{q} + 1$
2	$=$	$\mathtt{q} - \mathtt{p}$	$= \mathtt{n} - \mathtt{r} + \mathtt{q}$
3	$=$	$\mathtt{r} - \mathtt{q} + 1$	$= \mathtt{n} - \mathtt{q} + \mathtt{p} + 1$
4	$=$	$\mathtt{r} - \mathtt{q}$	$= \mathtt{n} - \mathtt{q} + \mathtt{p}$
5	$=$	1	
6	$=$	$\mathtt{r} - \mathtt{p} + 1$	$= \mathtt{n} + 1$
7	$=$	$\mathtt{r} - \mathtt{p}$	$= \mathtt{n}$
8–9	$=$	$\mathtt{r} - \mathtt{p}$	$= \mathtt{n}$
total	$=$	$7\mathtt{n} - 2(\mathtt{r} - \mathtt{p}) + 4$	$= 5\mathtt{n} + 4$

Conclusão

A função `intercala` consome $5\mathtt{n} + 4$ unidades de tempo.

O algoritmo `intercala` consome $\mathrm{O}(\mathtt{n})$ unidades de tempo.

Também escreve-se

O algoritmo `intercala` consome tempo $\mathrm{O}(\mathtt{n})$.

Ordenação: algoritmo Mergesort

Fonte: https://www.youtube.com/watch?v=XaqR3G_NVoo

PF 9

http://www.ime.usp.br/~pf/algoritmos/aulas/mrgsrt.html

Ordenação

$\mathtt{v}[0\,..\,\mathtt{n}-1]$ é crescente se $\mathtt{v}[0] \le \cdots \le \mathtt{v}[\mathtt{n}-1]$.

Problema: Rearranjar um vetor $\mathtt{v}[0\,..\,\mathtt{n}-1]$ de modo que ele fique crescente.

Entra:

0										$\mathtt{n}-1$
33	55	33	44	33	22	11	99	22	55	77

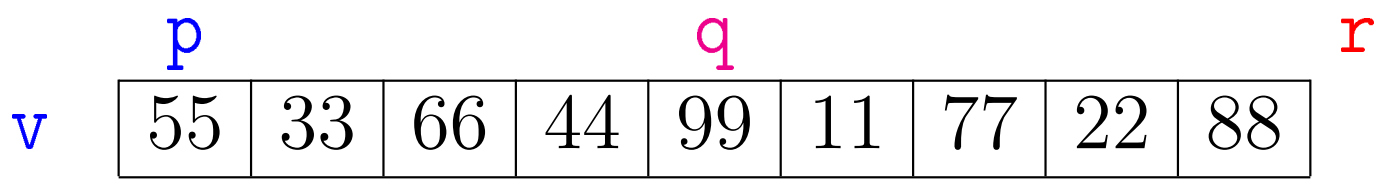
Sai:

0										$\mathtt{n}-1$
11	22	22	33	33	33	44	55	55	77	99

Função mergeSort

Rearranja v[p..r−1] em ordem crescente.

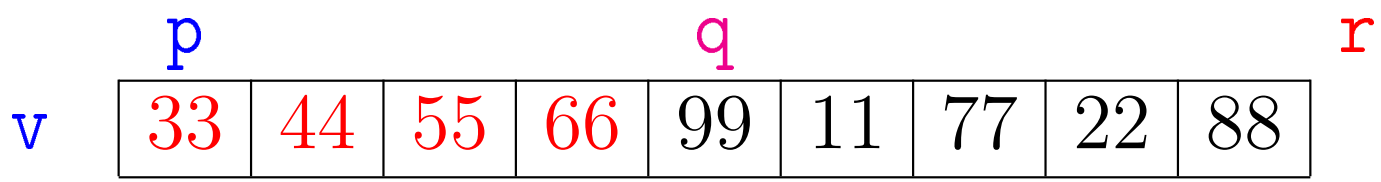
```
void mergeSort (int p, int r, int v[]) {
1   if (p < r-1) {
2       int q = (p + r)/2;
3       mergeSort(p, q, v);
4       mergeSort(q, r, v);
5       intercala(p, q, r, v);
    }
}
```



função mergeSort

Rearranja v[p..r−1] em ordem crescente.

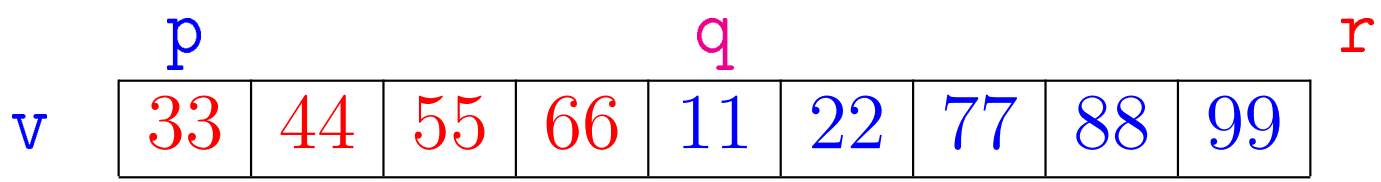
```
void mergeSort (int p, int r, int v[]) {
1   if (p < r-1) {
2       int q = (p + r)/2;
3       mergeSort(p, q, v);
4       mergeSort(q, r, v);
5       intercala(p, q, r, v);
    }
}
```



função mergeSort

Rearranja v[p..r−1] em ordem crescente.

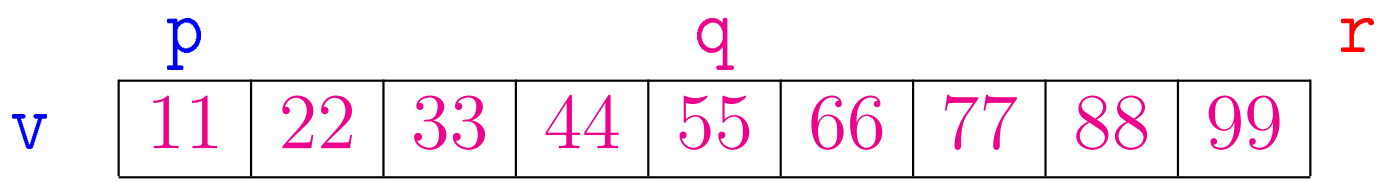
```
void mergeSort (int p, int r, int v[]) {
1   if (p < r-1) {
2       int q = (p + r)/2;
3       mergeSort(p, q, v);
4       mergeSort(q, r, v);
5       intercala(p, q, r, v);
    }
}
```



função mergeSort

Rearranja $v[p\,..\,r-1]$ em ordem crescente.

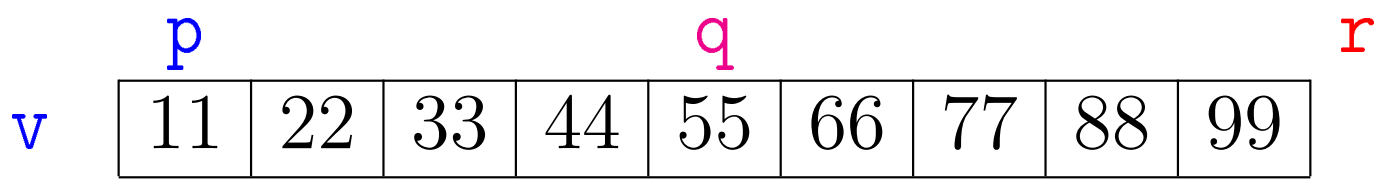
```
void mergeSort (int p, int r, int v[]) {
1   if (p < r-1) {
2       int q = (p + r)/2;
3       mergeSort(p, q, v);
4       mergeSort(q, r, v);
5       intercala(p, q, r, v);
    }
}
```



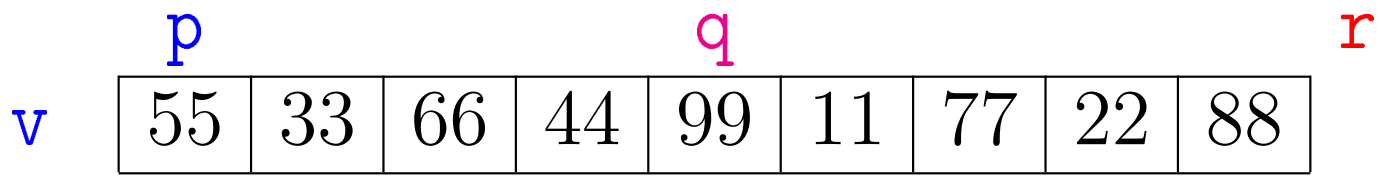
função mergeSort

Rearranja v[p..r−1] em ordem crescente.

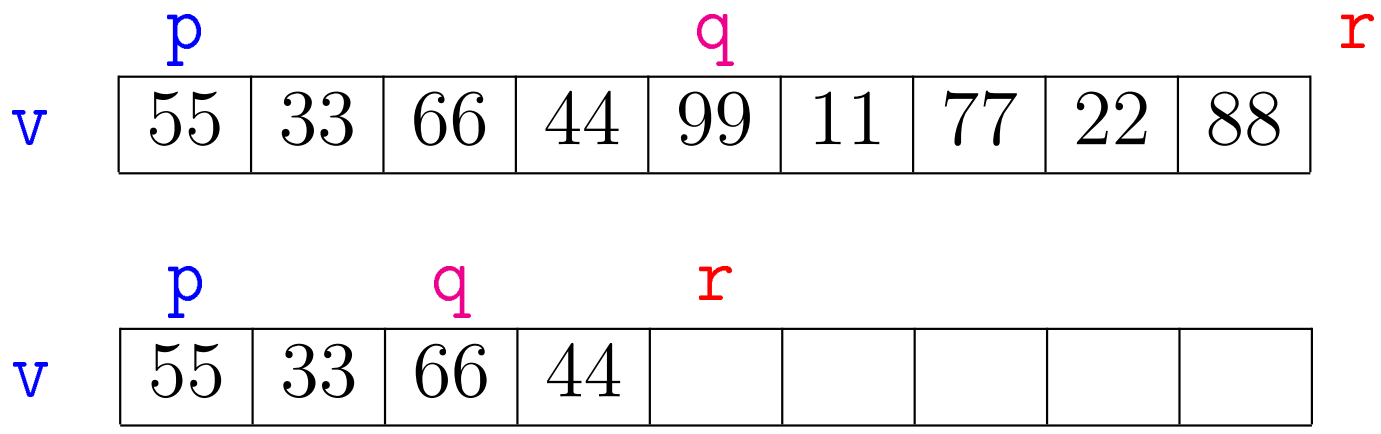
```
void mergeSort (int p, int r, int v[]) {
1   if (p < r-1) {
2       int q = (p + r)/2;
3       mergeSort(p, q, v);
4       mergeSort(q, r, v);
5       intercala(p, q, r, v);
    }
}
```



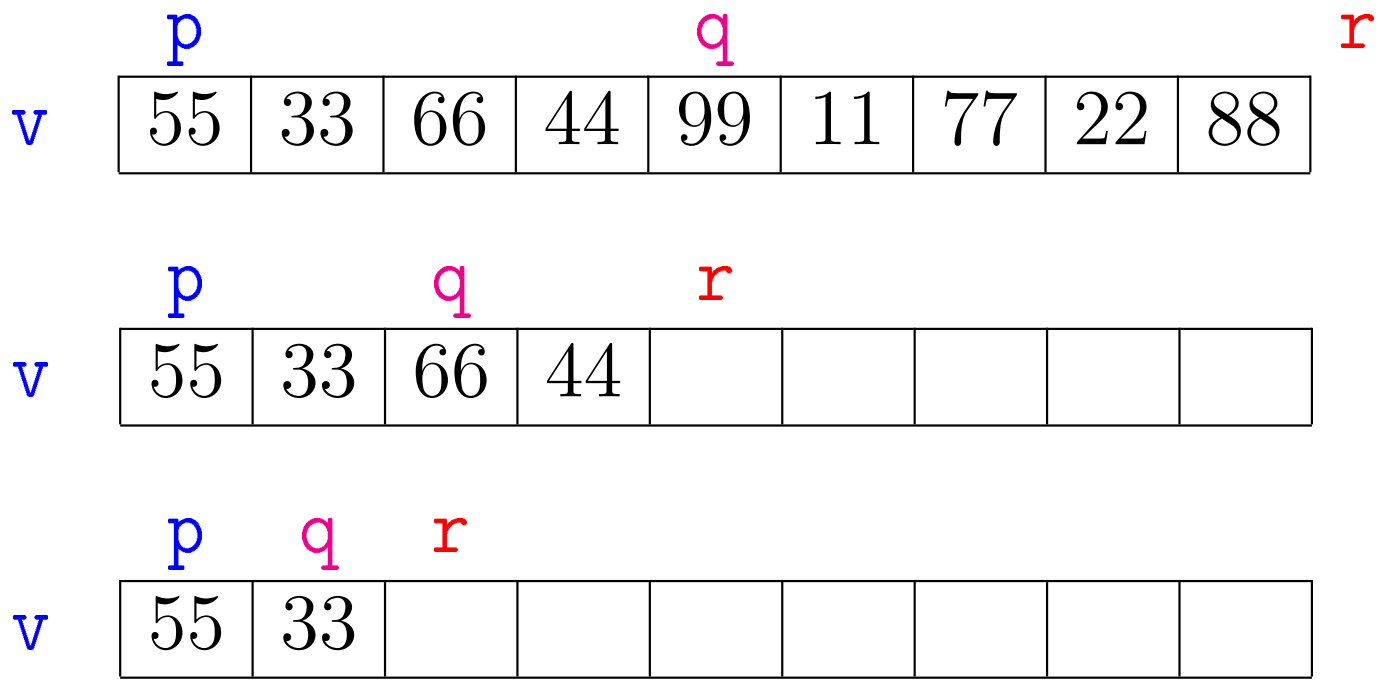
Mergesort



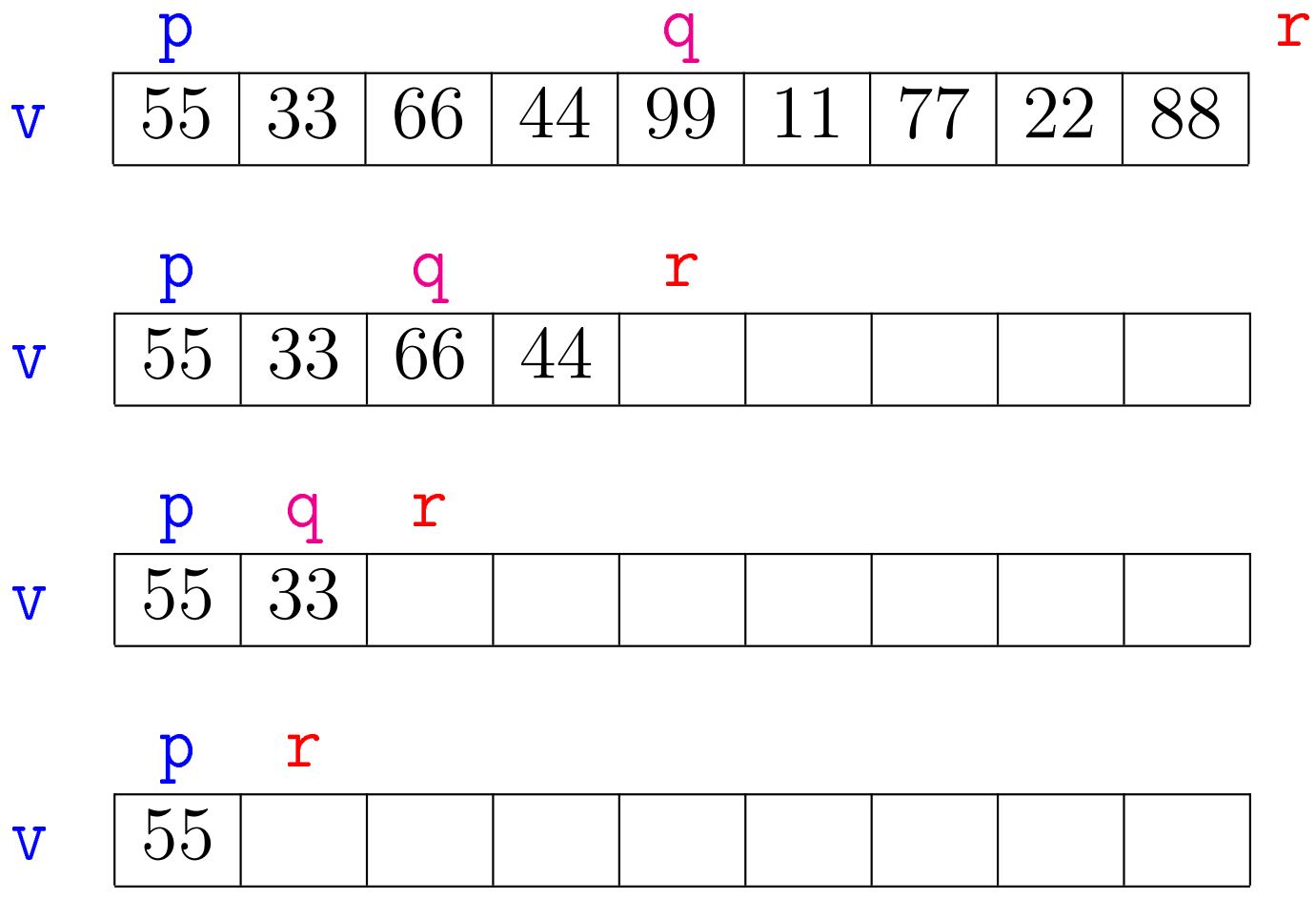
Mergesort



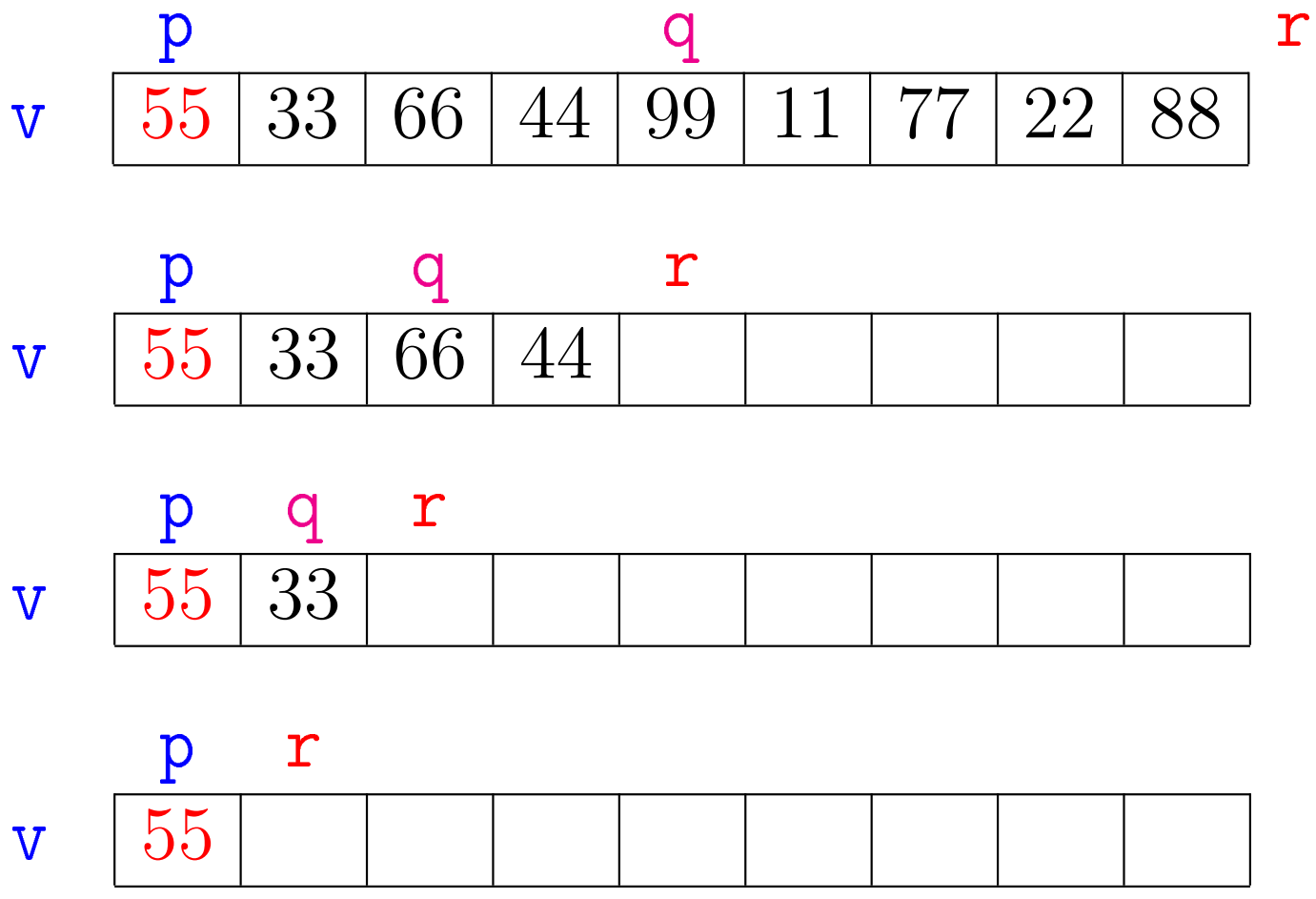
Mergesort



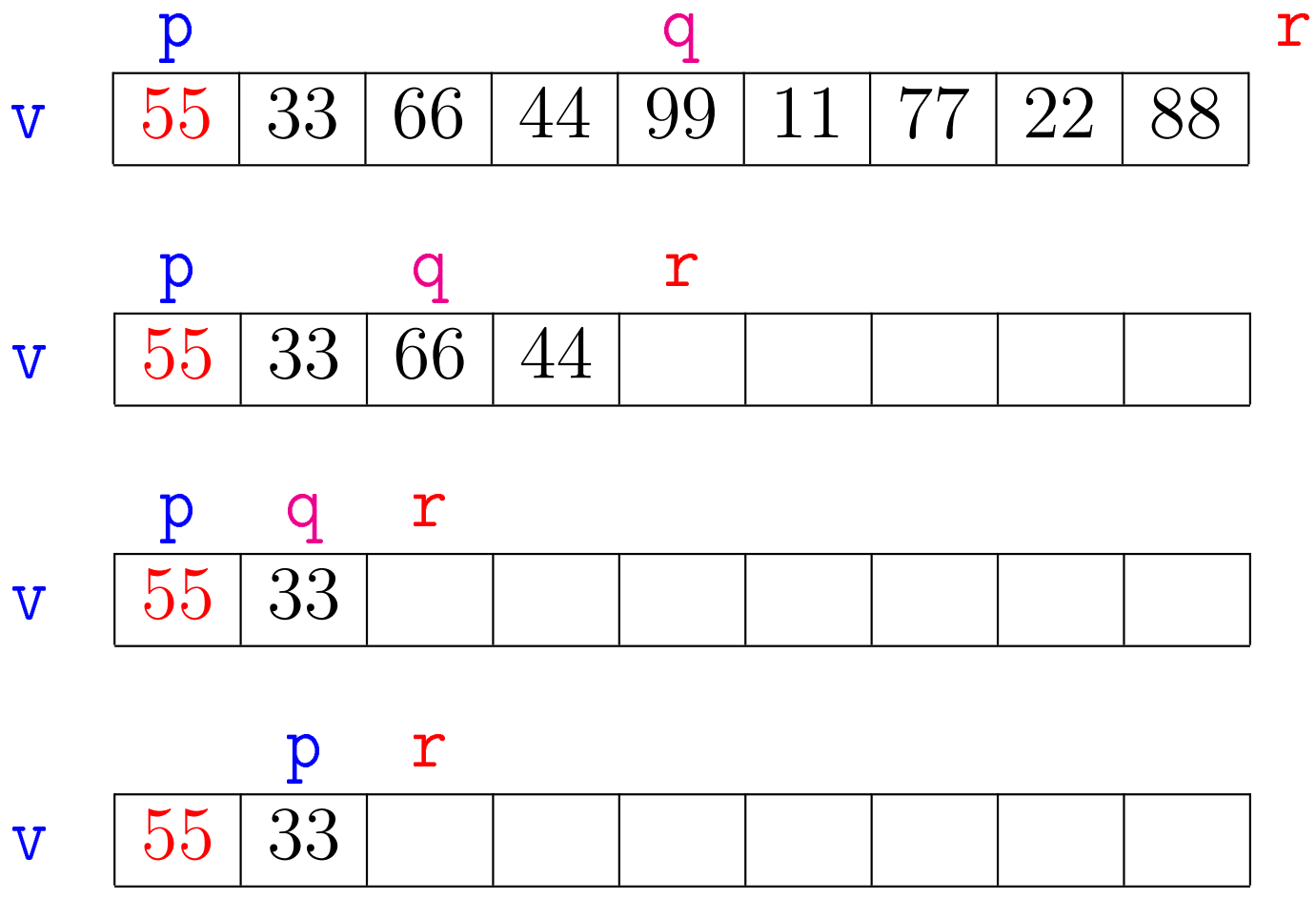
Mergesort



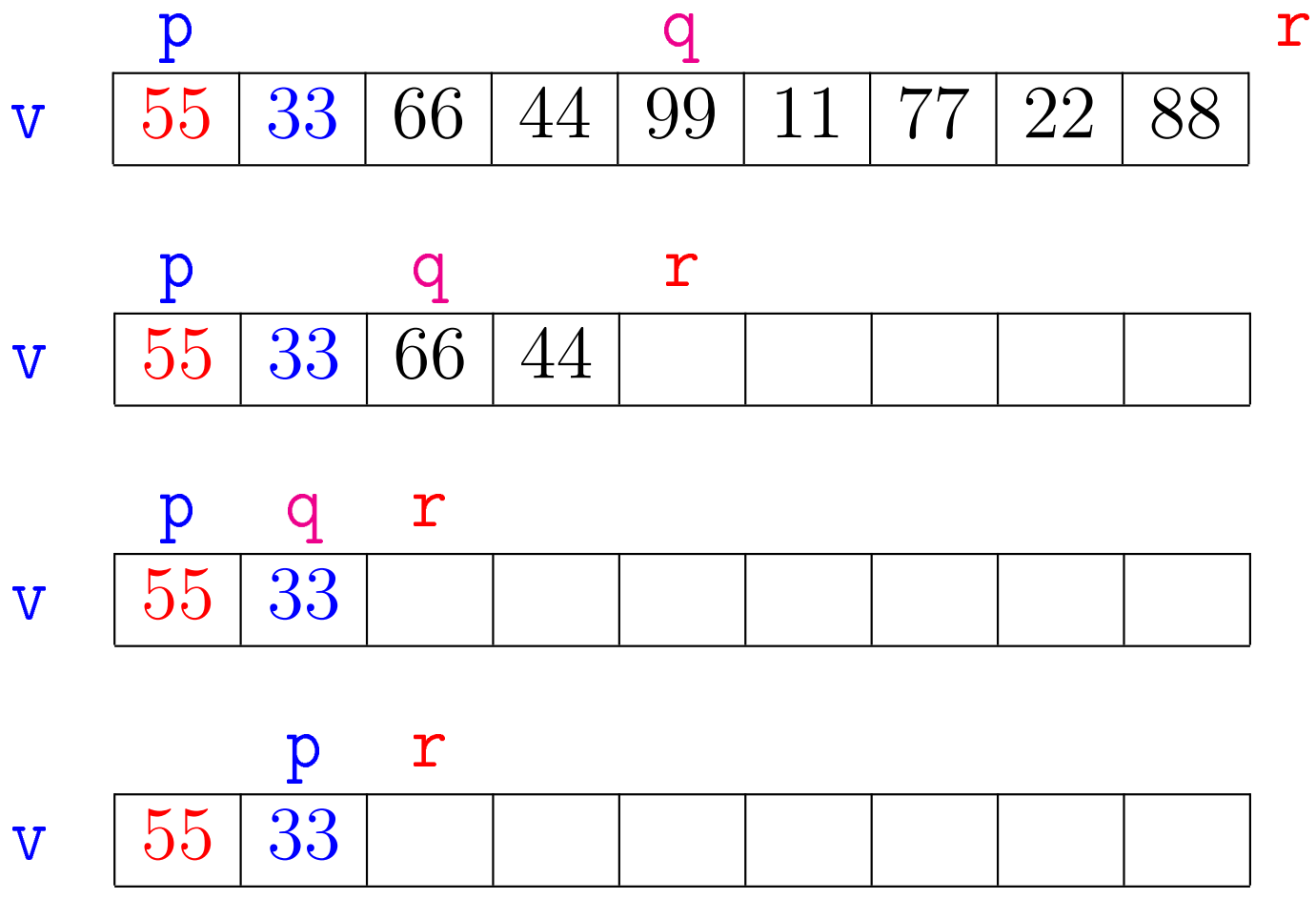
Mergesort



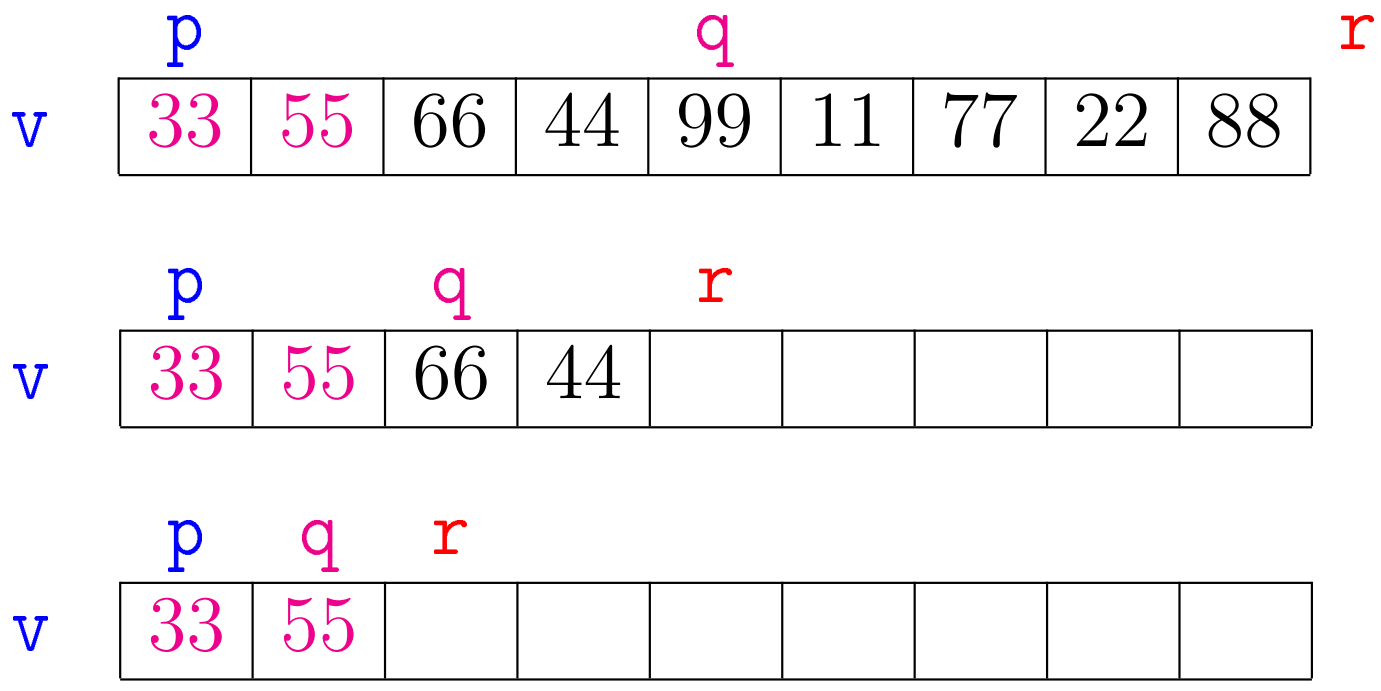
Mergesort



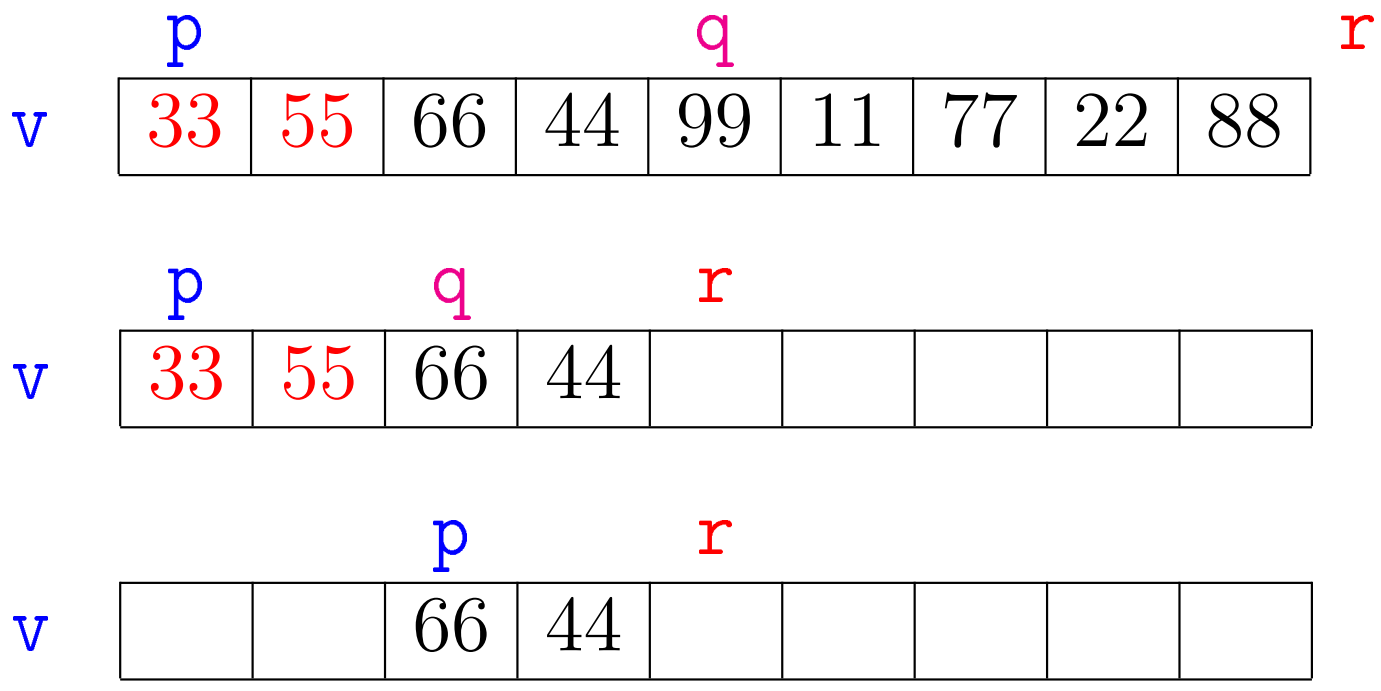
Mergesort



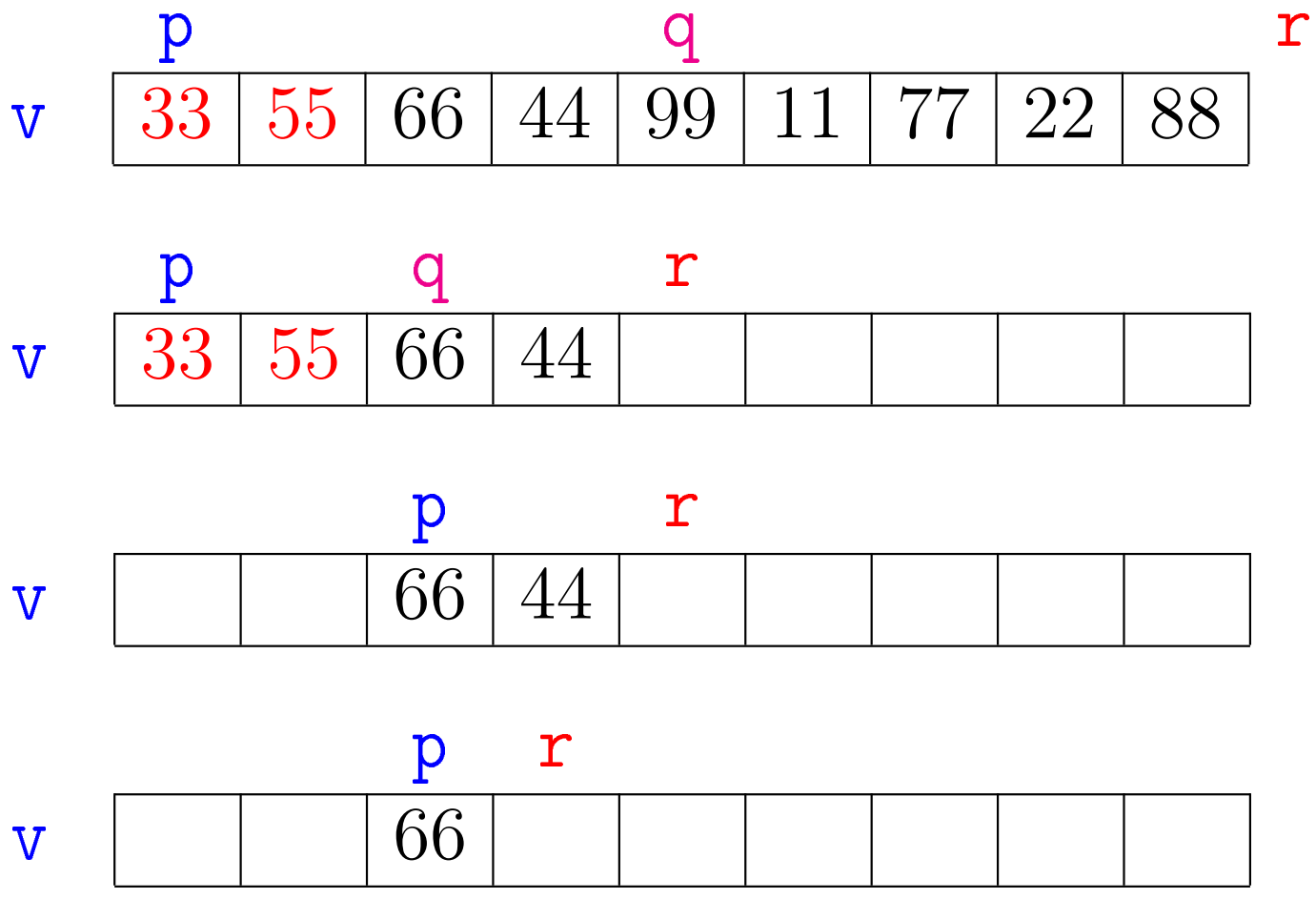
Mergesort



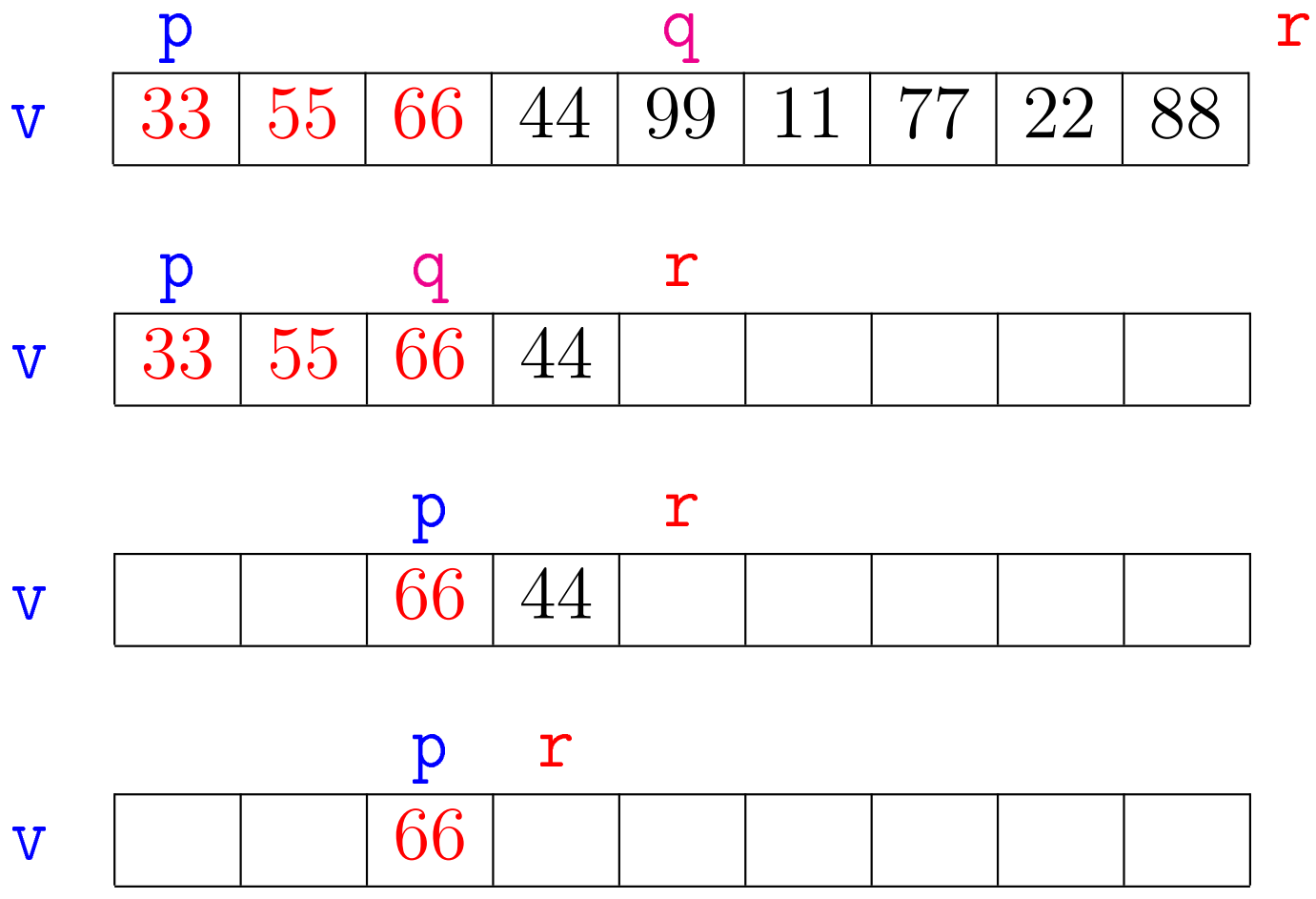
Mergesort



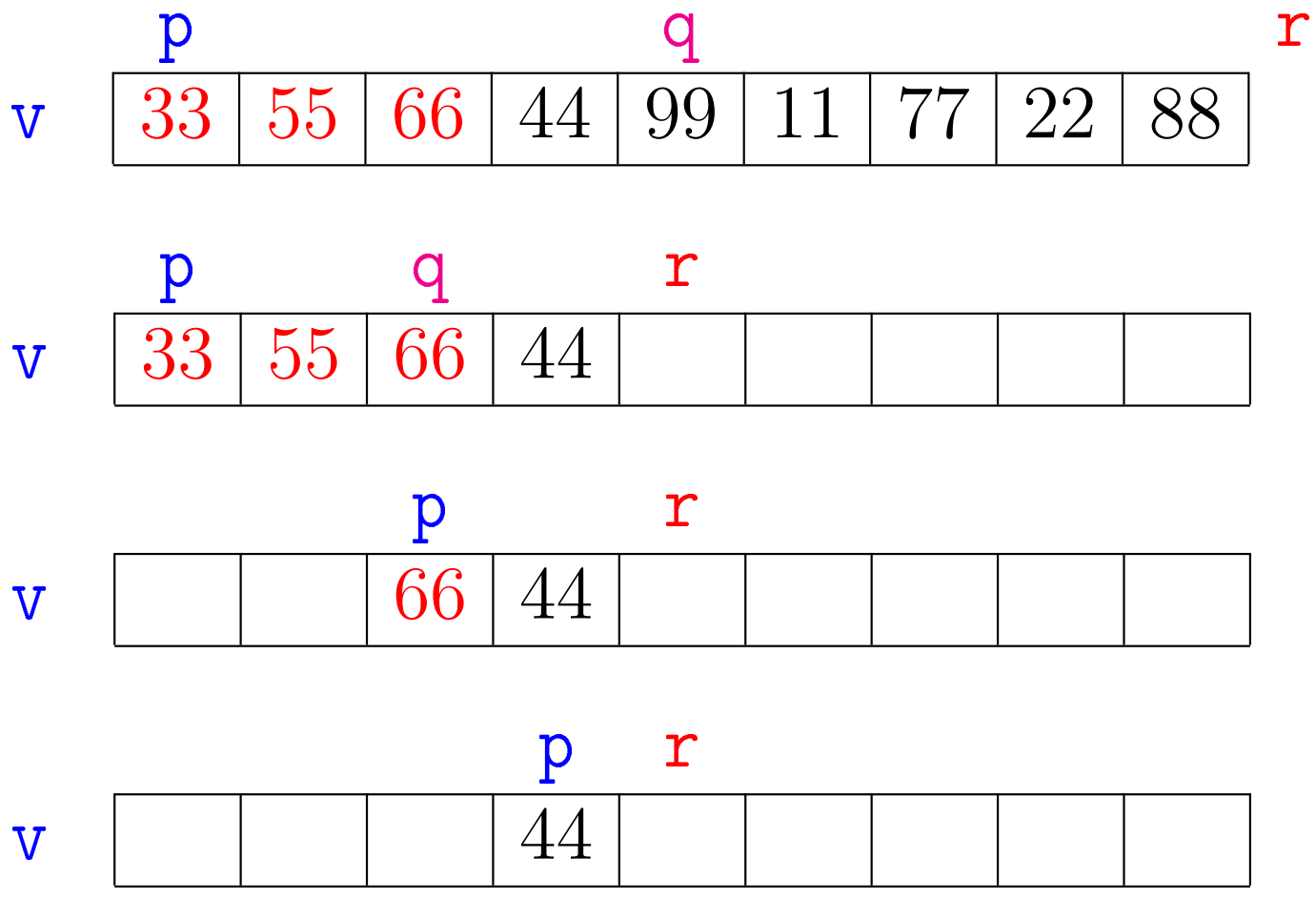
Mergesort



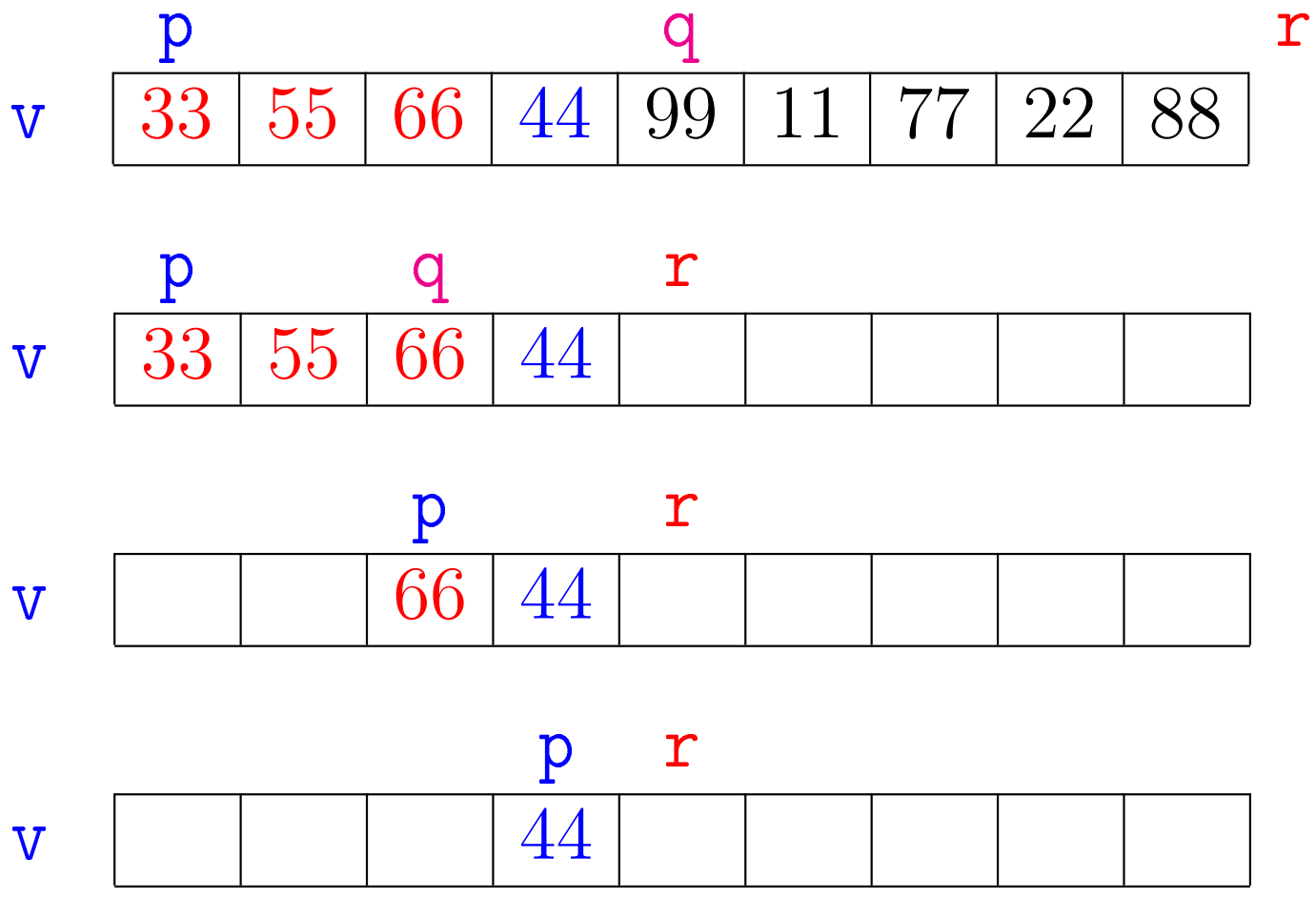
Mergesort



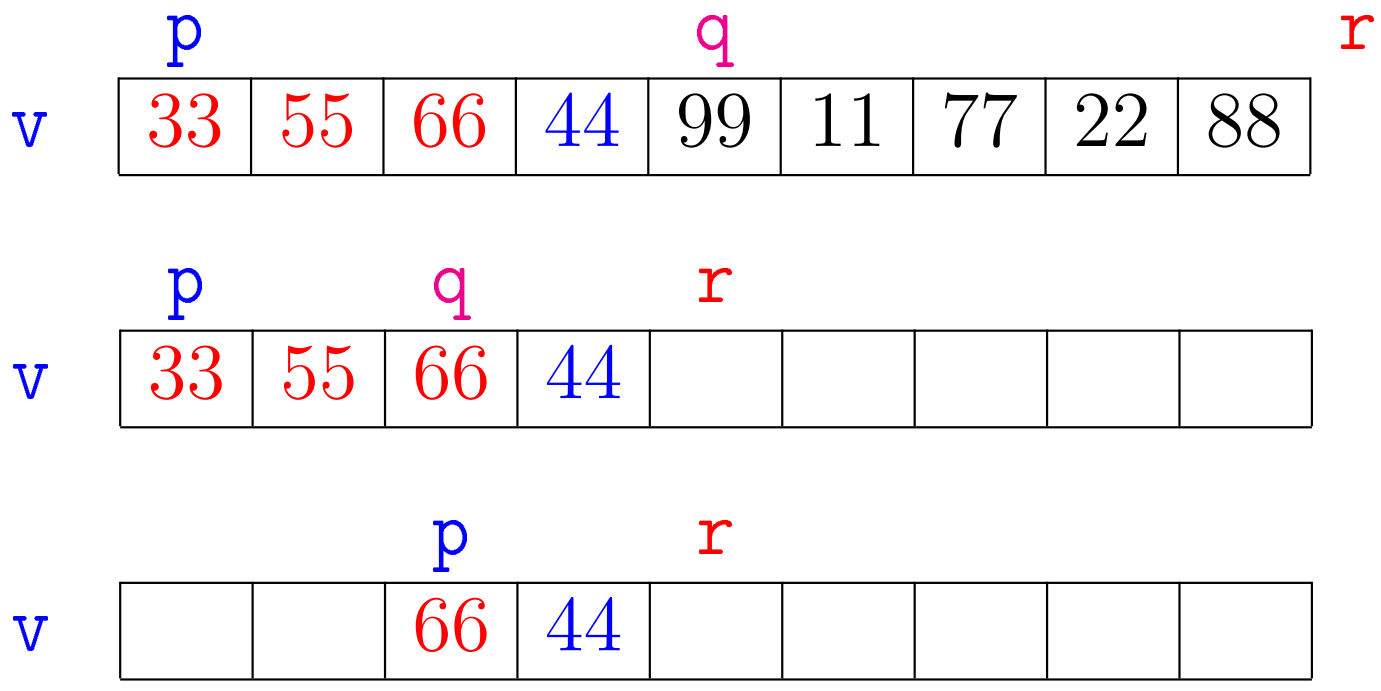
Mergesort



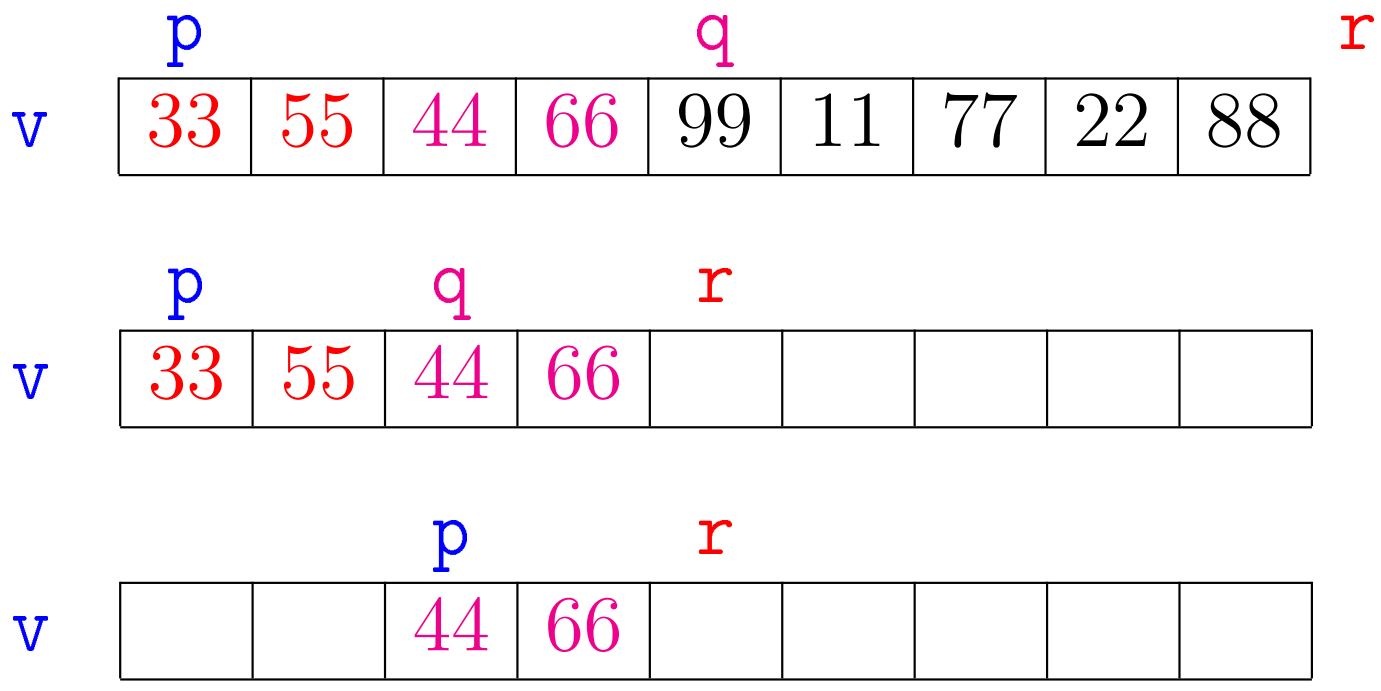
Mergesort



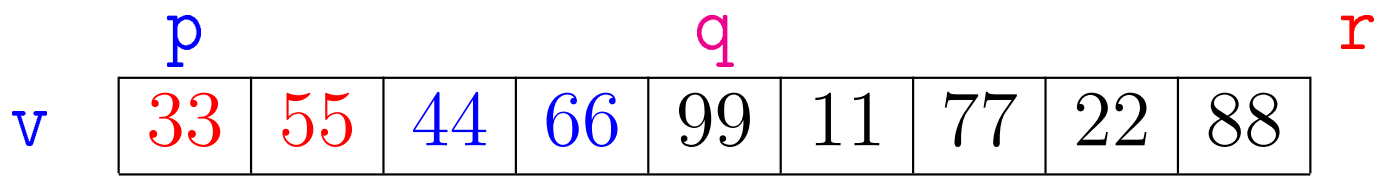
Mergesort

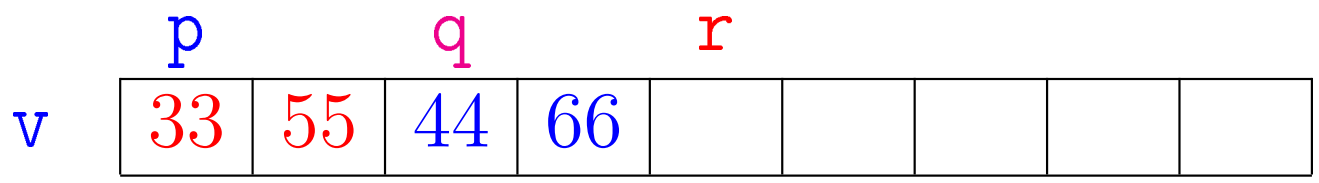


Mergesort

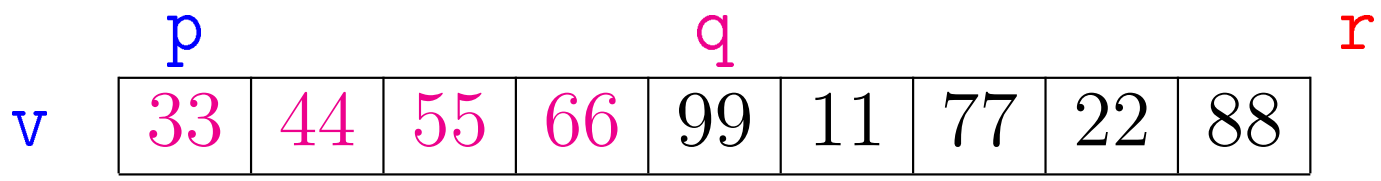


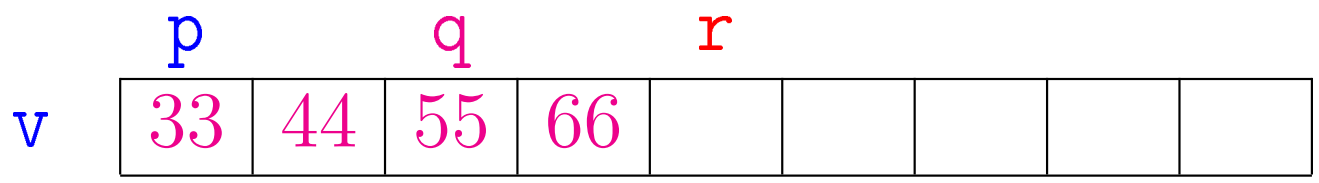
Mergesort



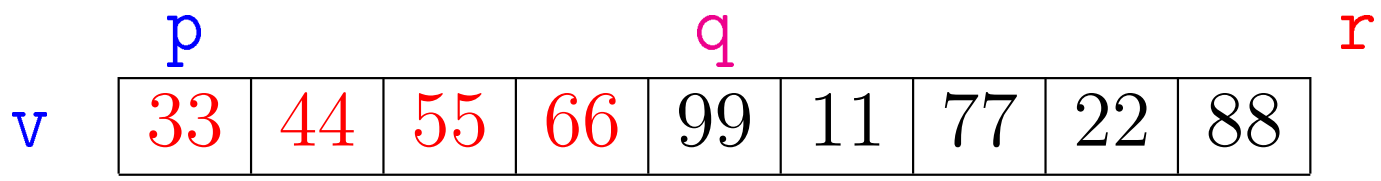


Mergesort

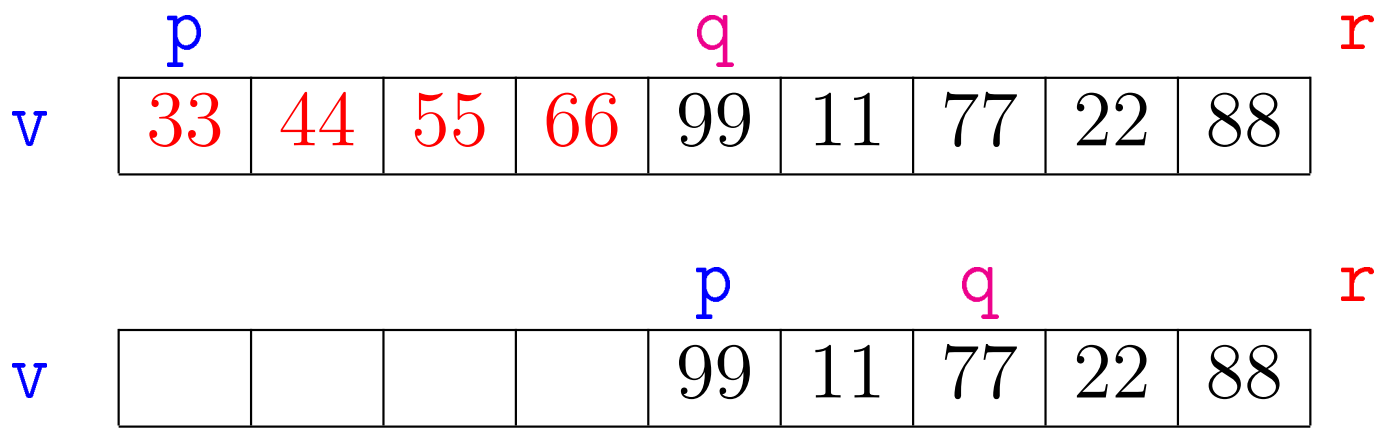




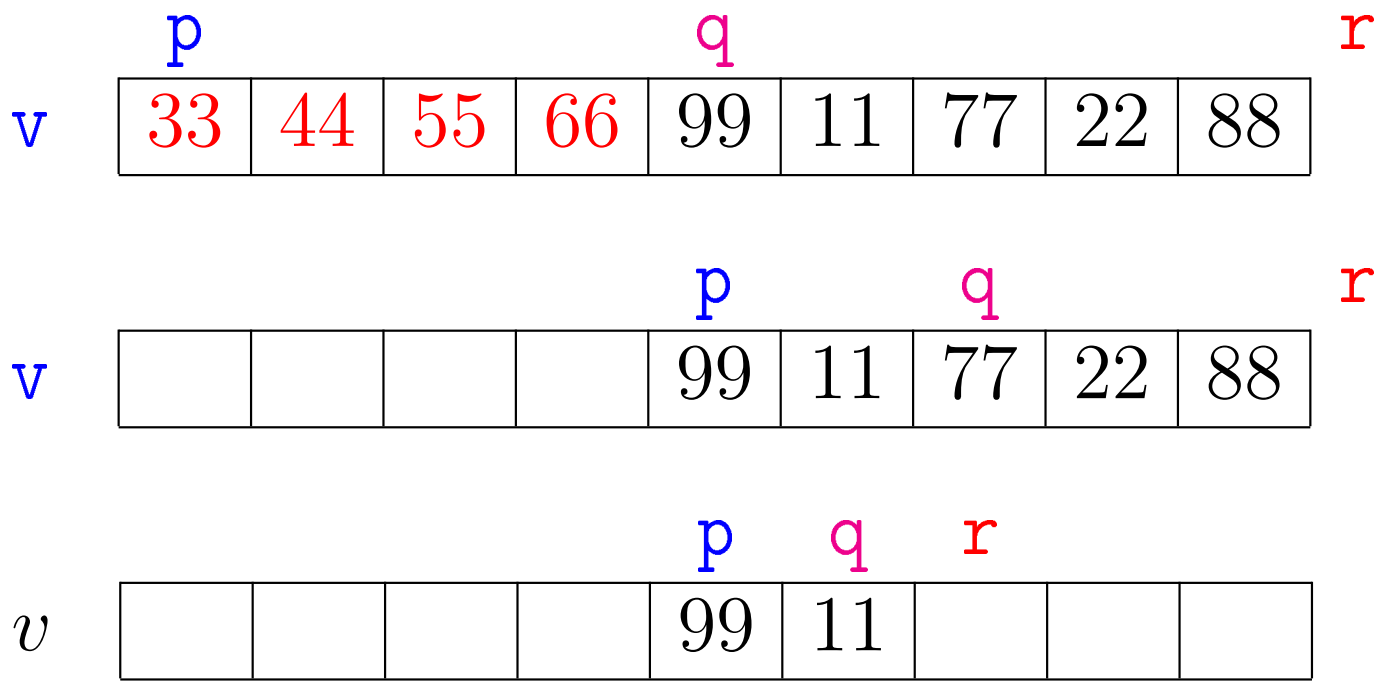
Mergesort



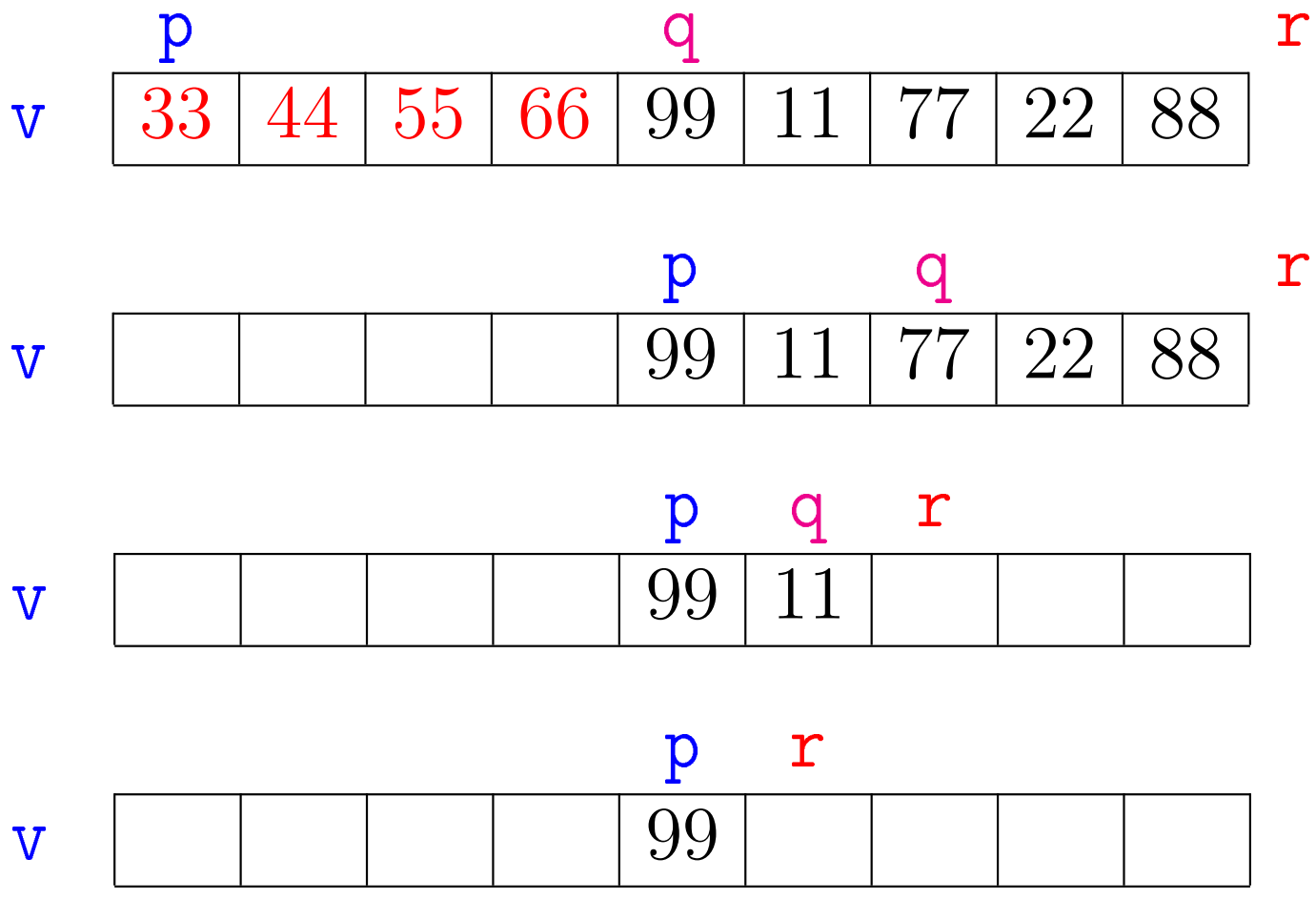
Mergesort



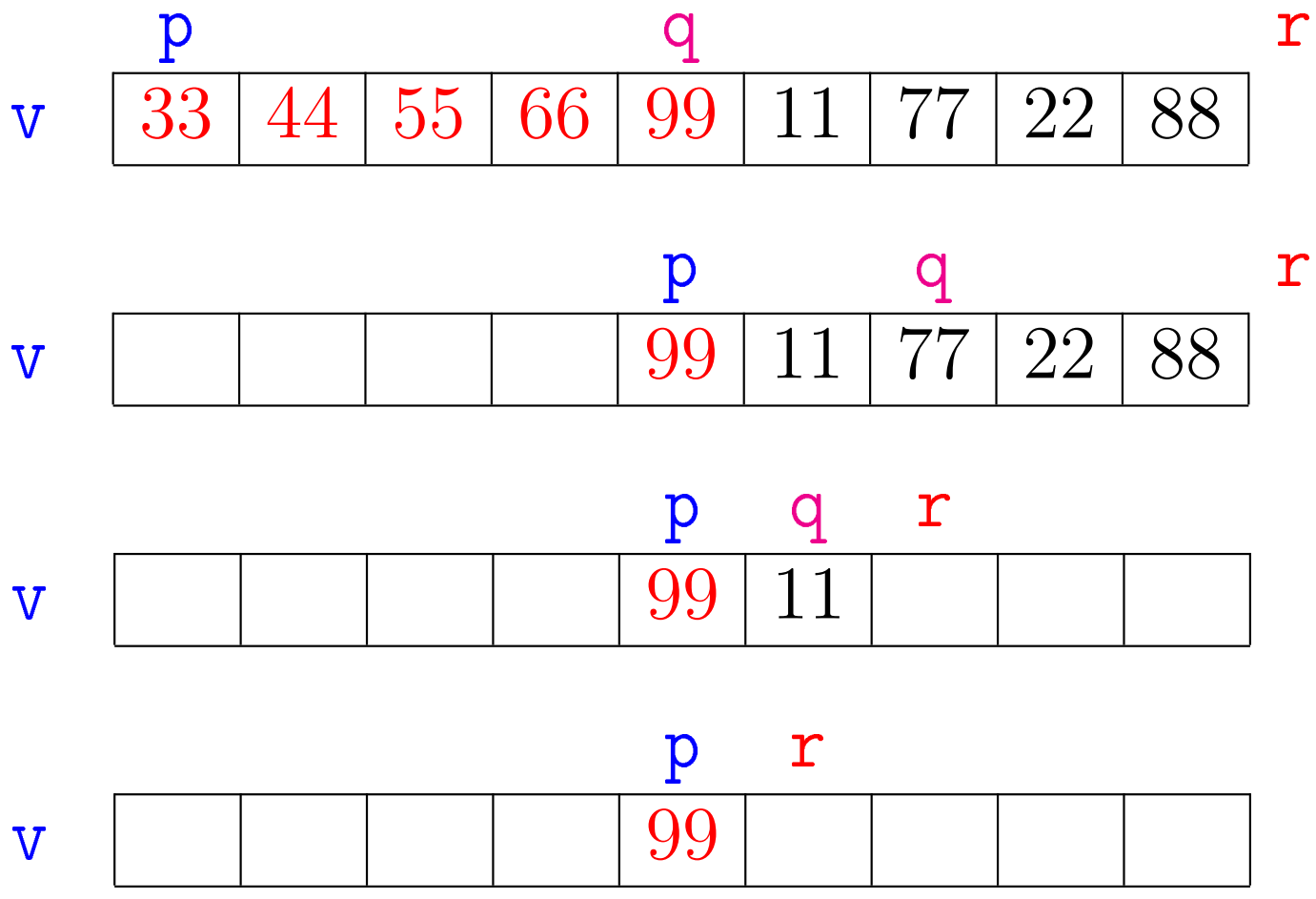
Mergesort



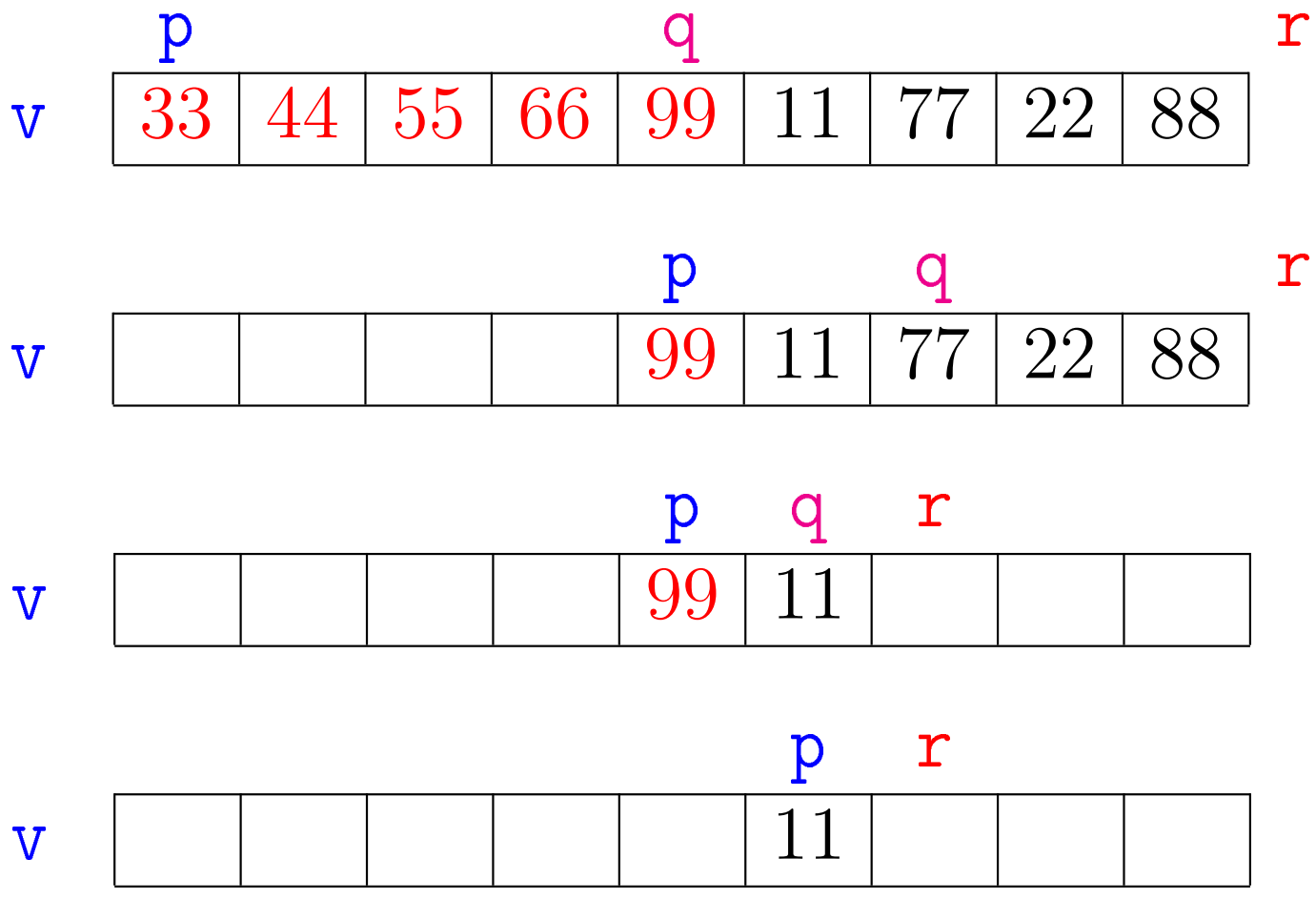
Mergesort



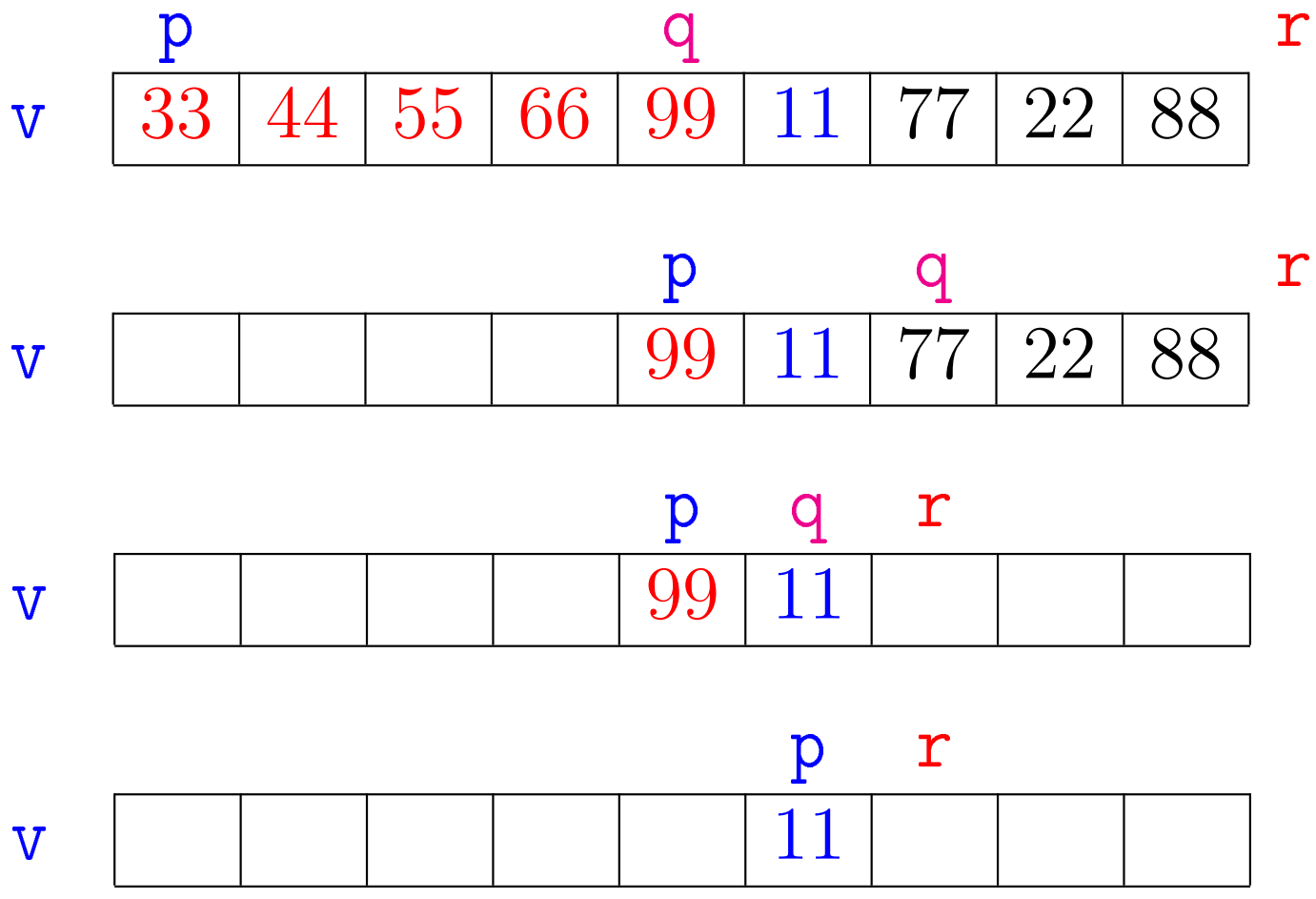
Mergesort



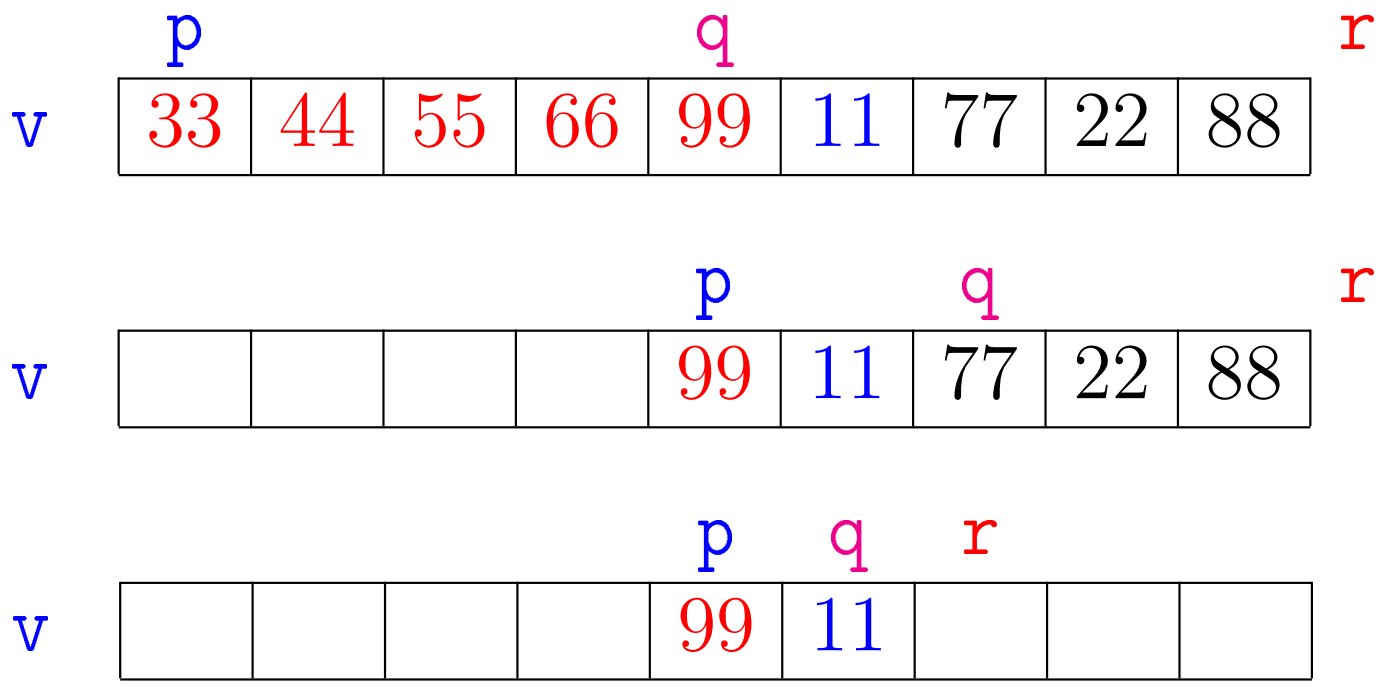
Mergesort



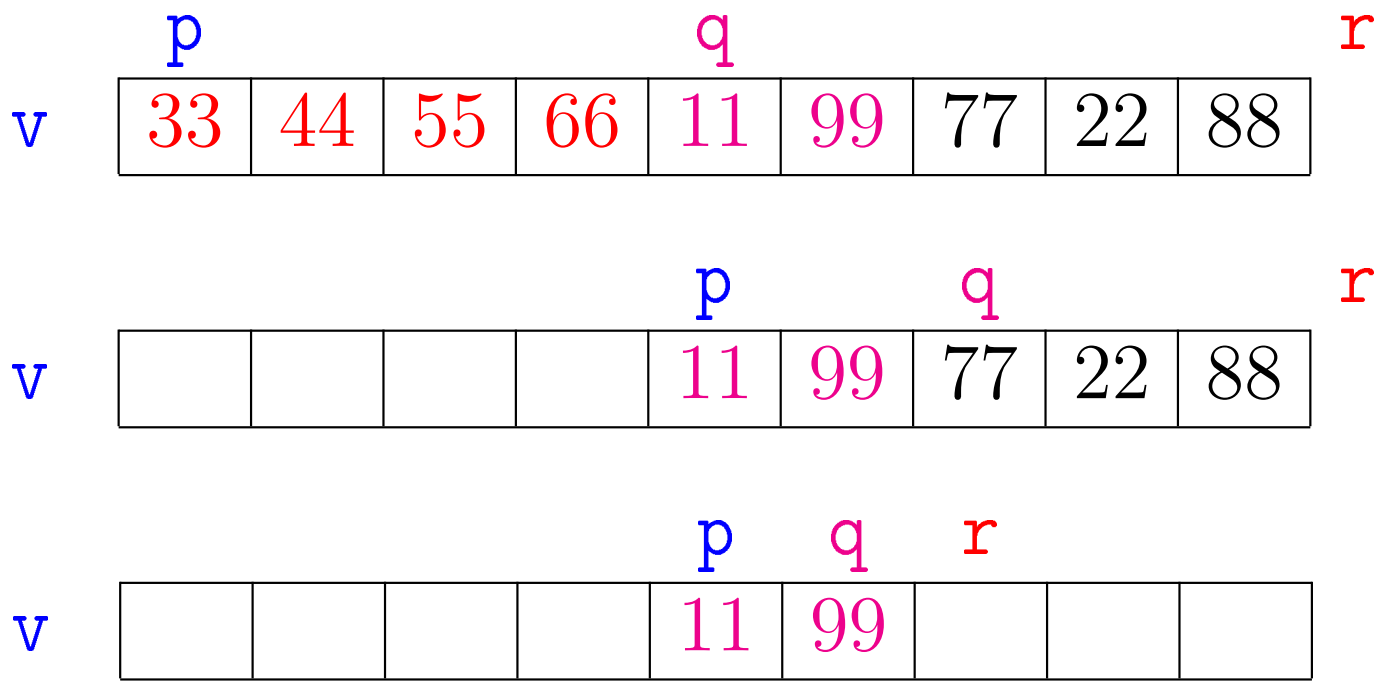
Mergesort



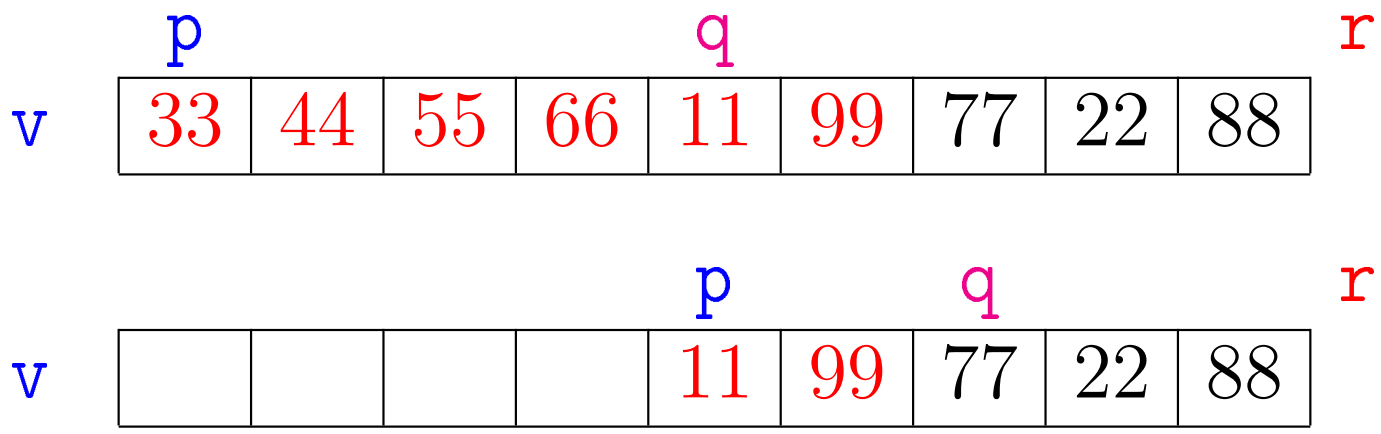
Mergesort



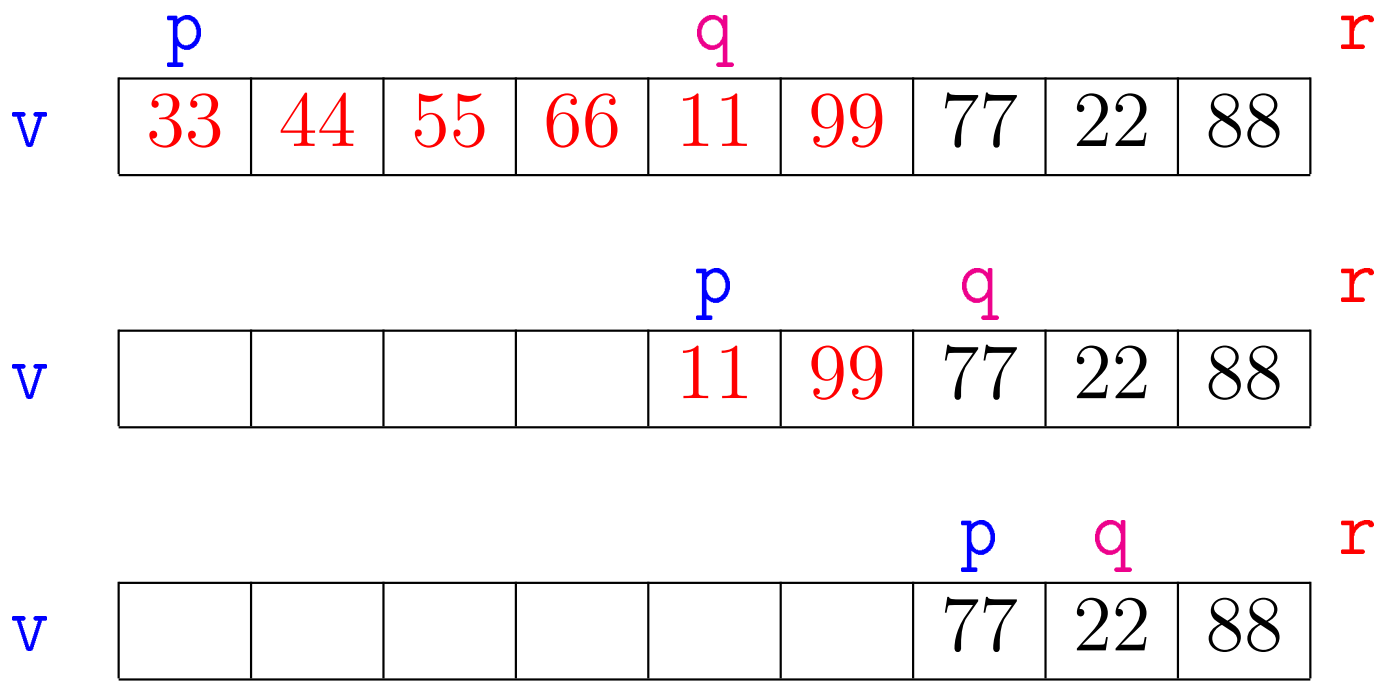
Mergesort



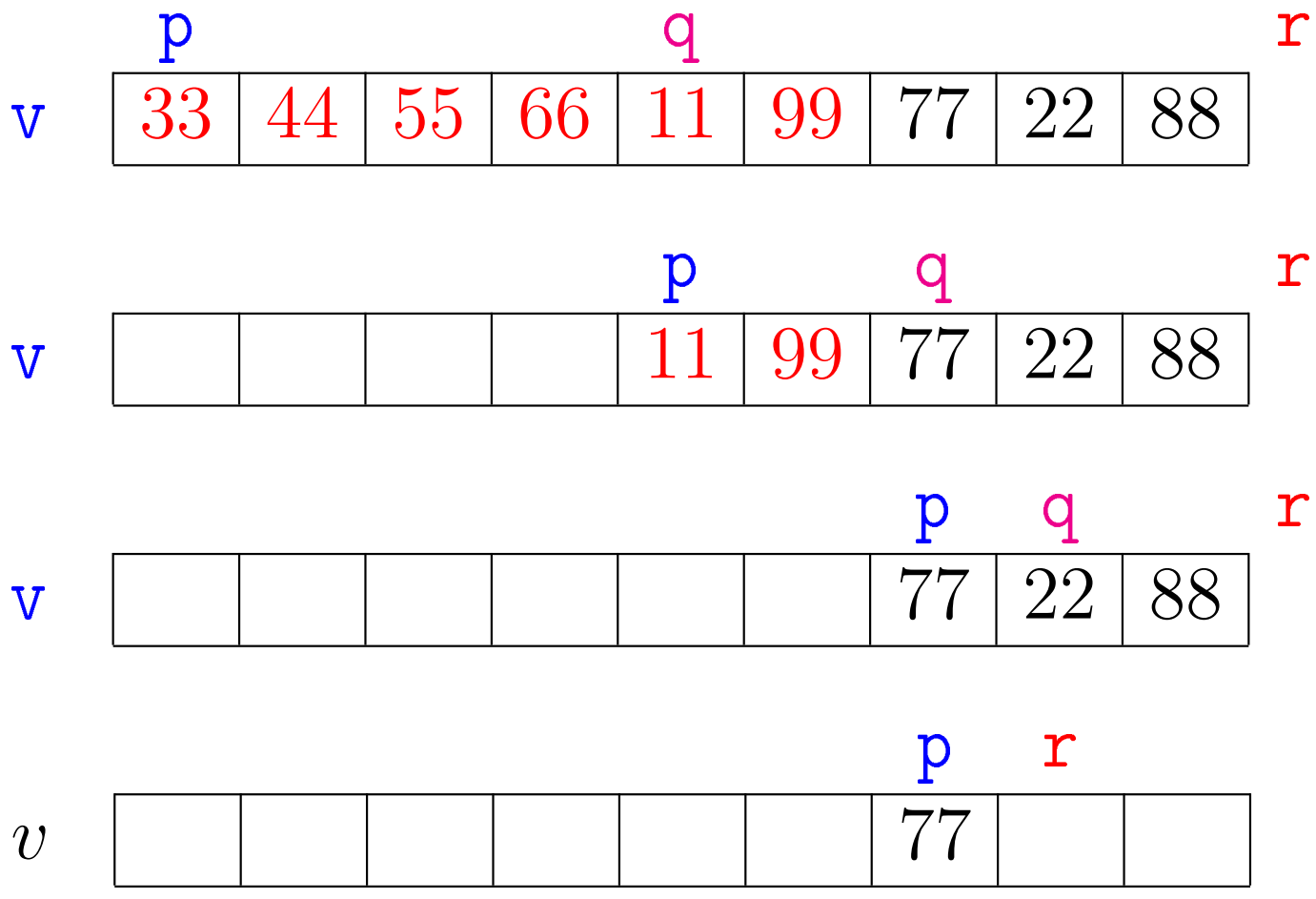
Mergesort



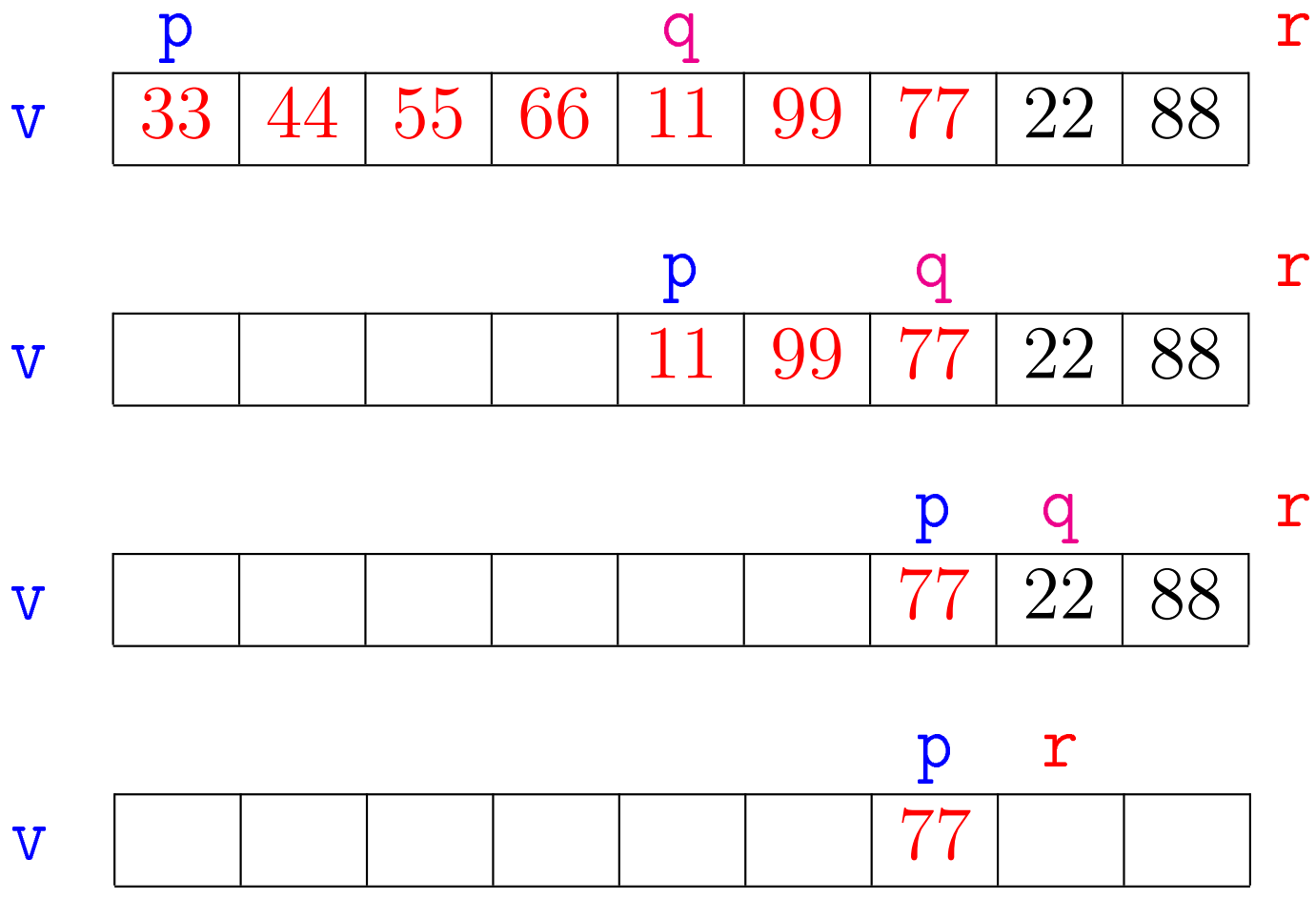
Mergesort



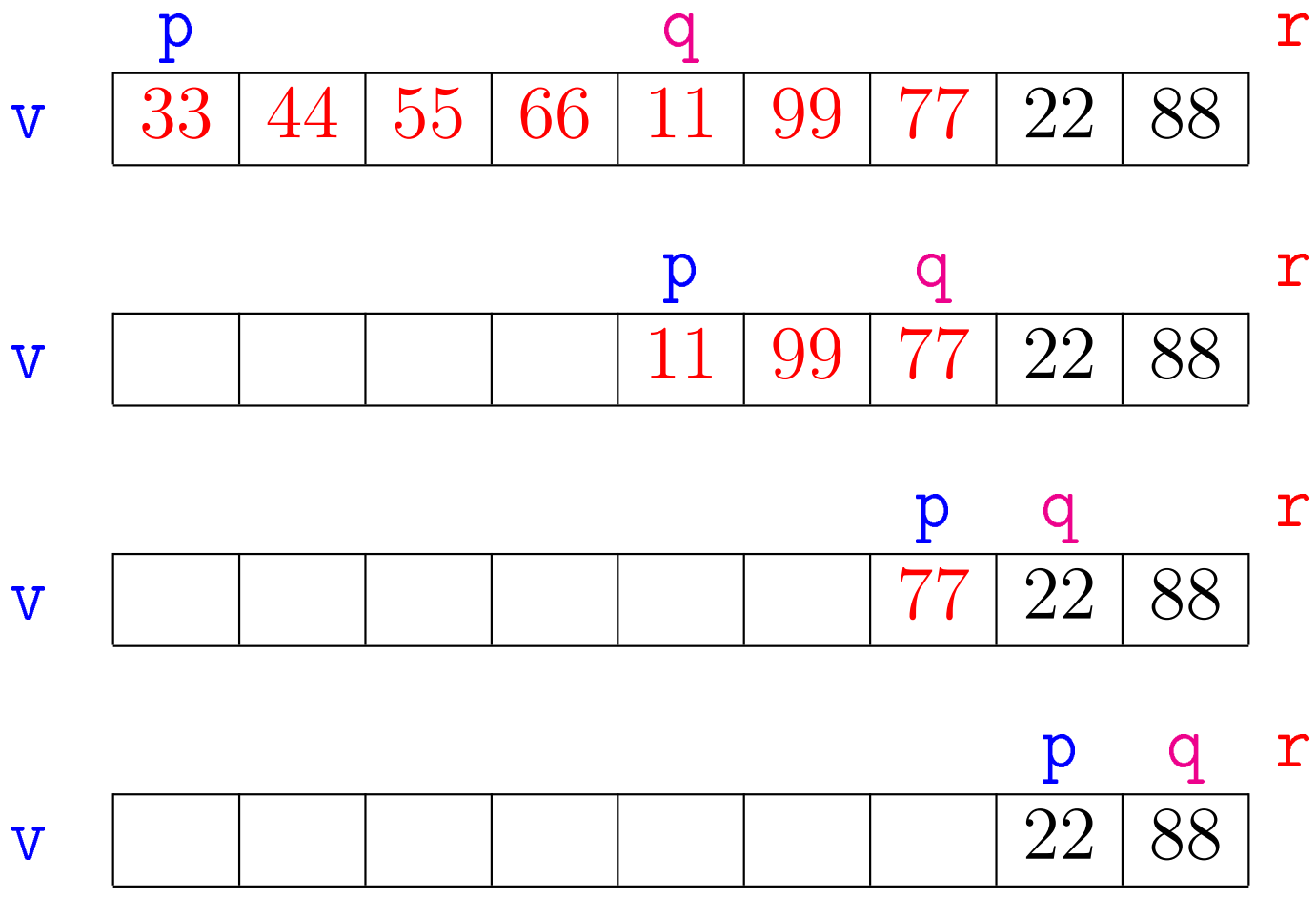
Mergesort


p
q
r
v
33
44
55
66
11
99
77
22
88
p
q
r
v
11
99
77
22
88
p
q
r
v
77
22
88
p
r
v
77

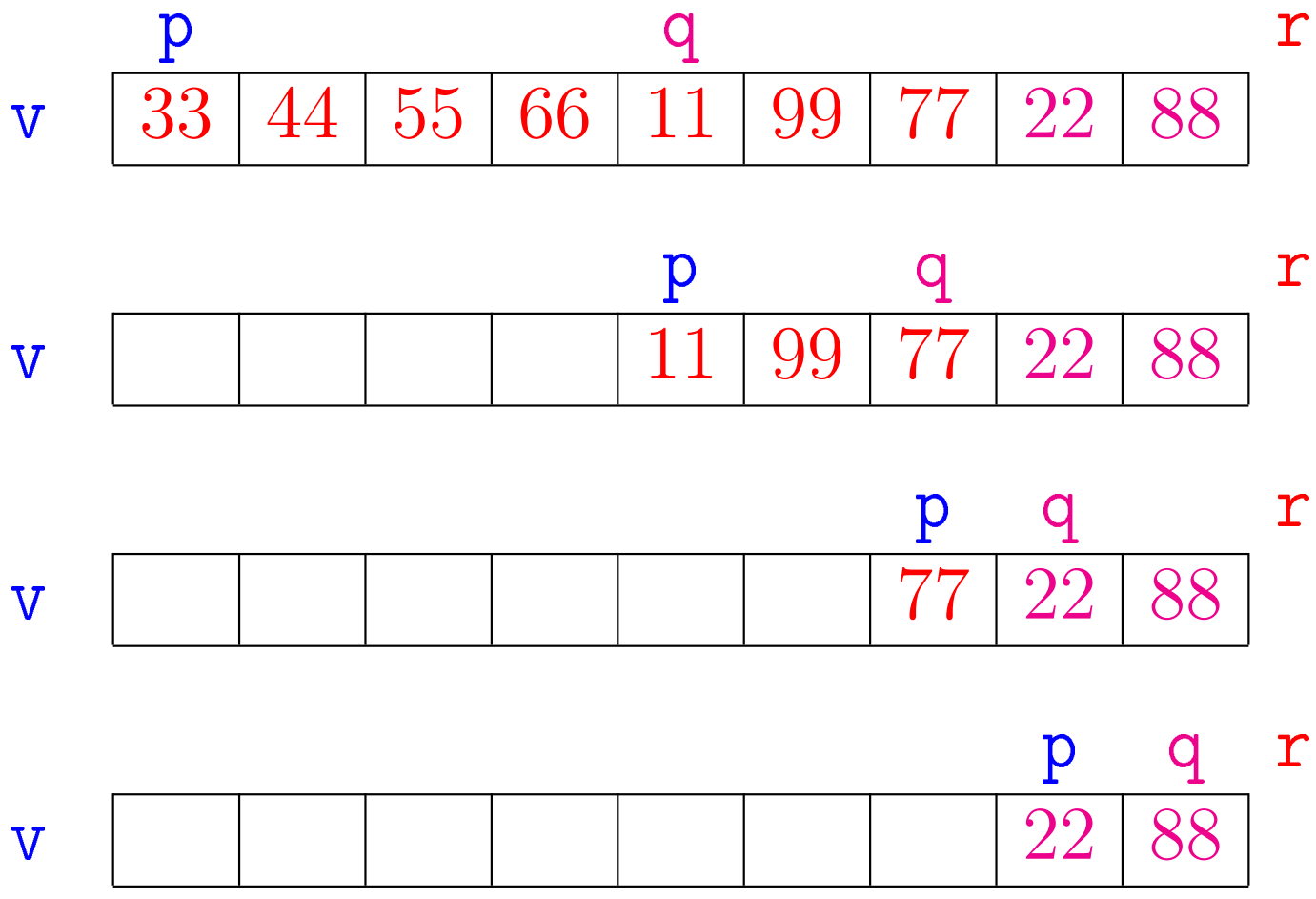
Mergesort


p
q
r
v
33
44
55
66
11
99
77
22
88
p
q
r
v
11
99
77
22
88
p
q
r
v
77
22
88
p
r
v
77

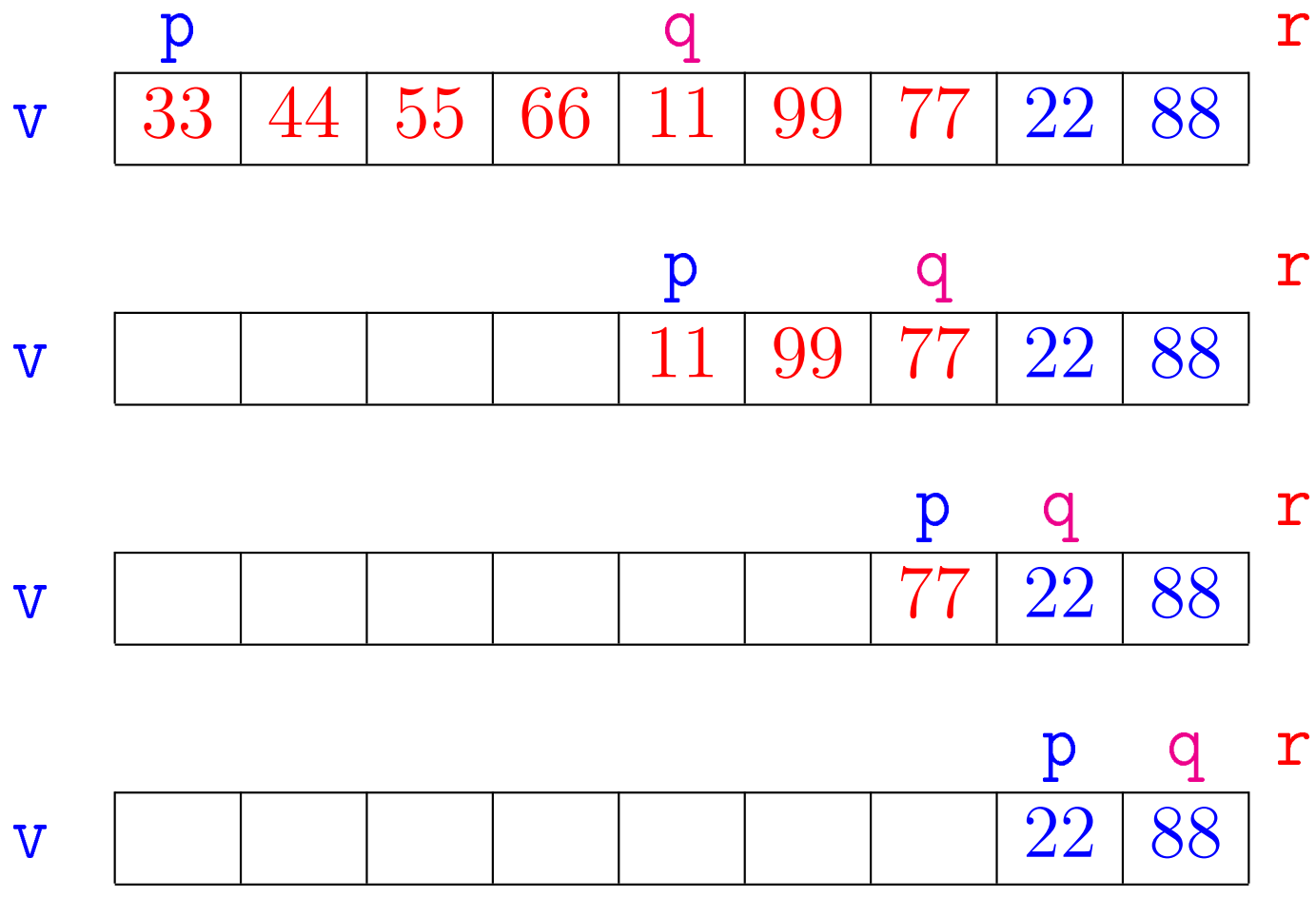
Mergesort


p
q
r
v
33
44
55
66
11
99
77
22
88
p
q
r
v
11
99
77
22
88
p
q
r
v
77
22
88
p
q
r
v
22
88

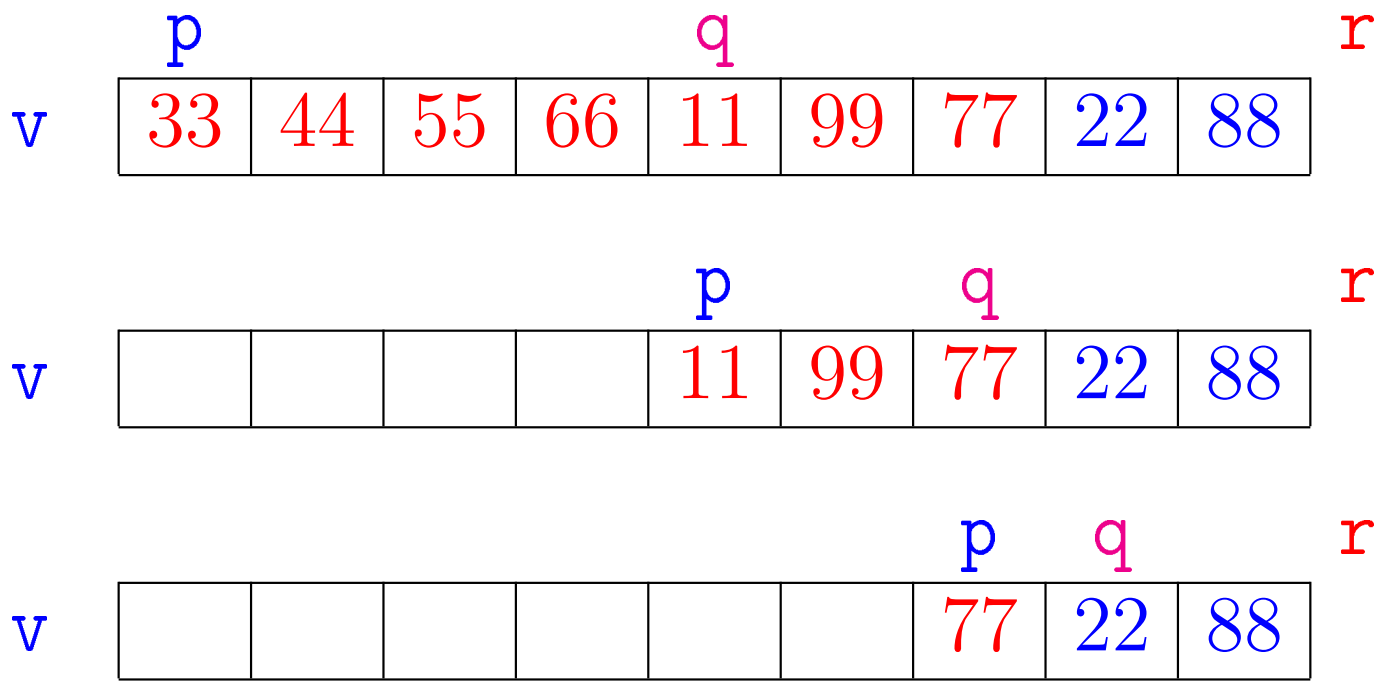
Mergesort


p
q
r
v
33
44
55
66
11
99
77
22
88
p
q
r
v
11
99
77
22
88
p
q
r
v
77
22
88
p
q
r
v
22
88

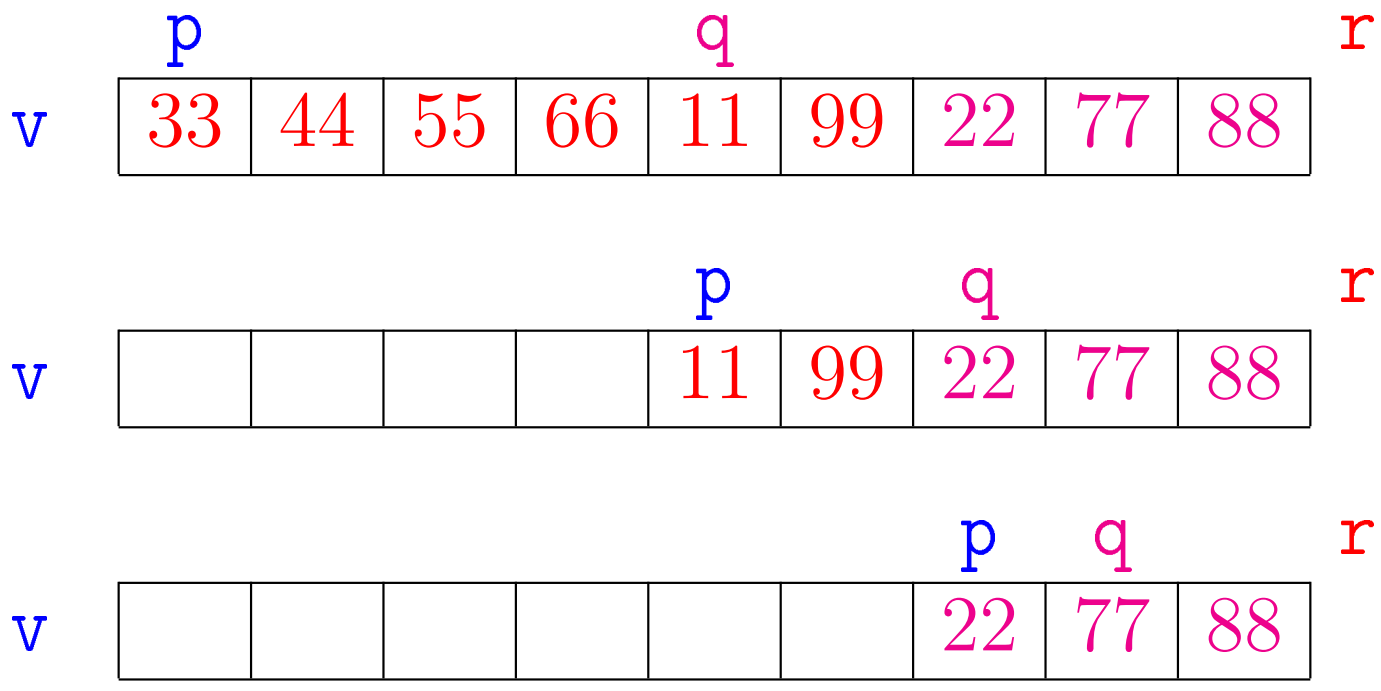
Mergesort


p
q
r
v
33
44
55
66
11
99
77
22
88
p
q
r
v
11
99
77
22
88
p
q
r
v
77
22
88
p
q
r
v
22
88

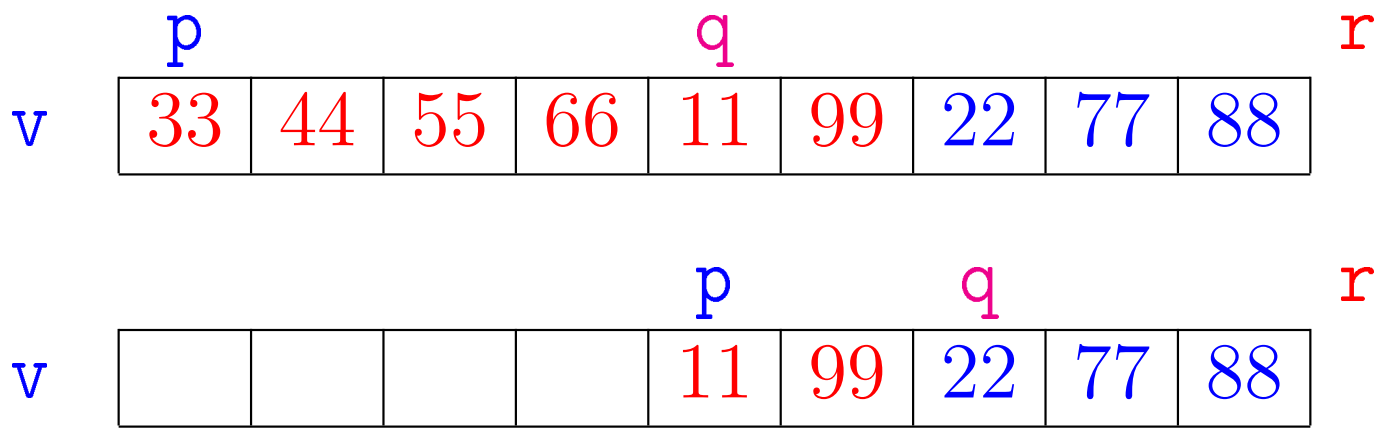
Mergesort



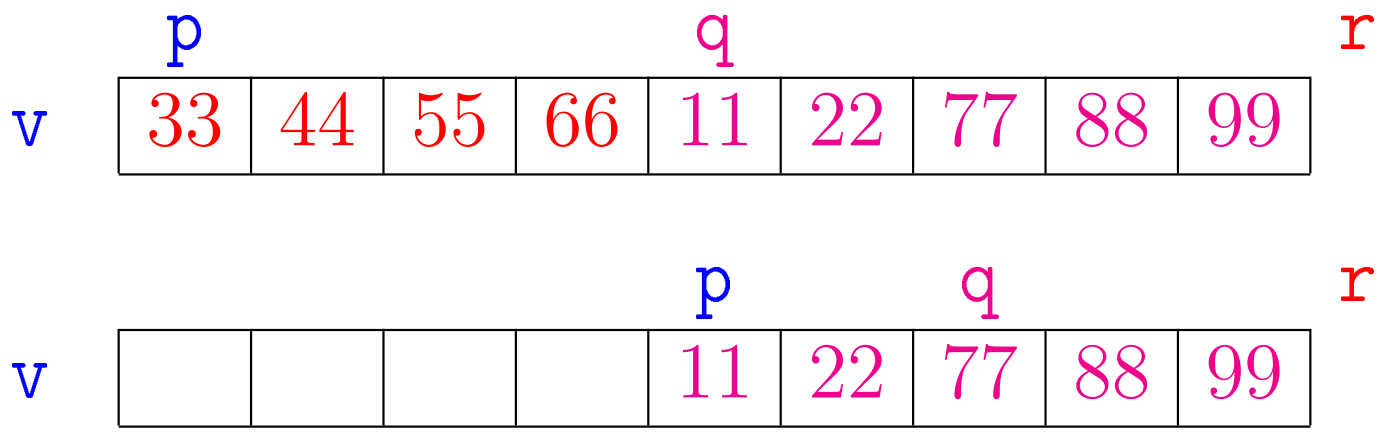
Mergesort



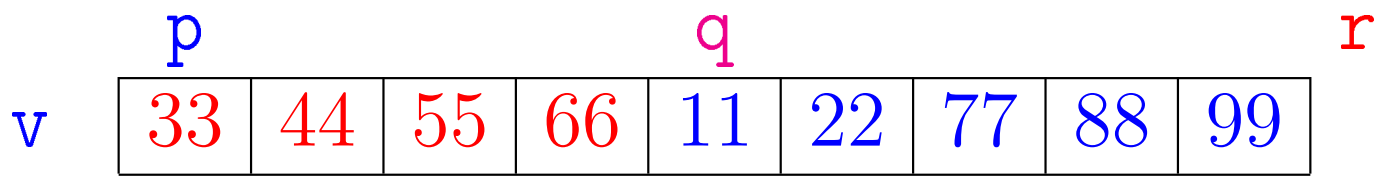
Mergesort



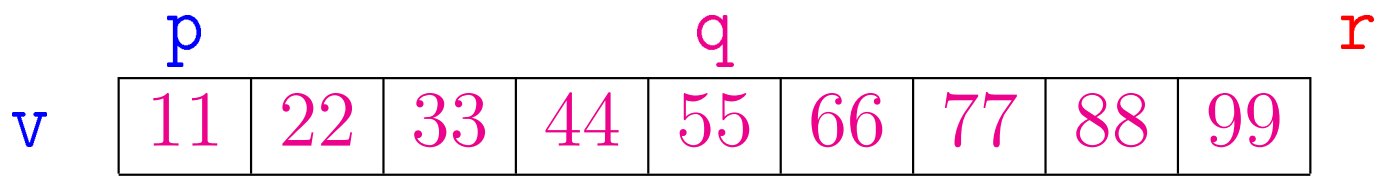
Mergesort



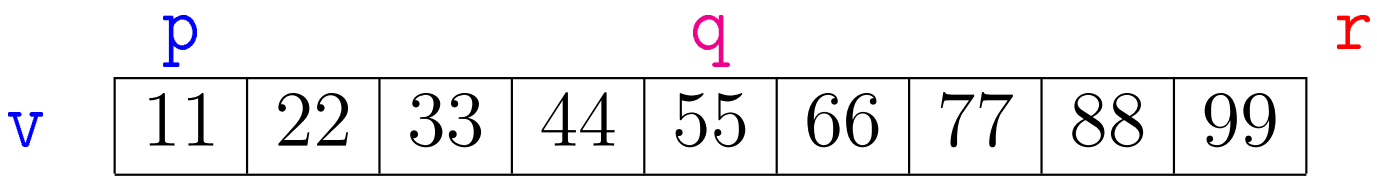
Mergesort



Mergesort



Mergesort

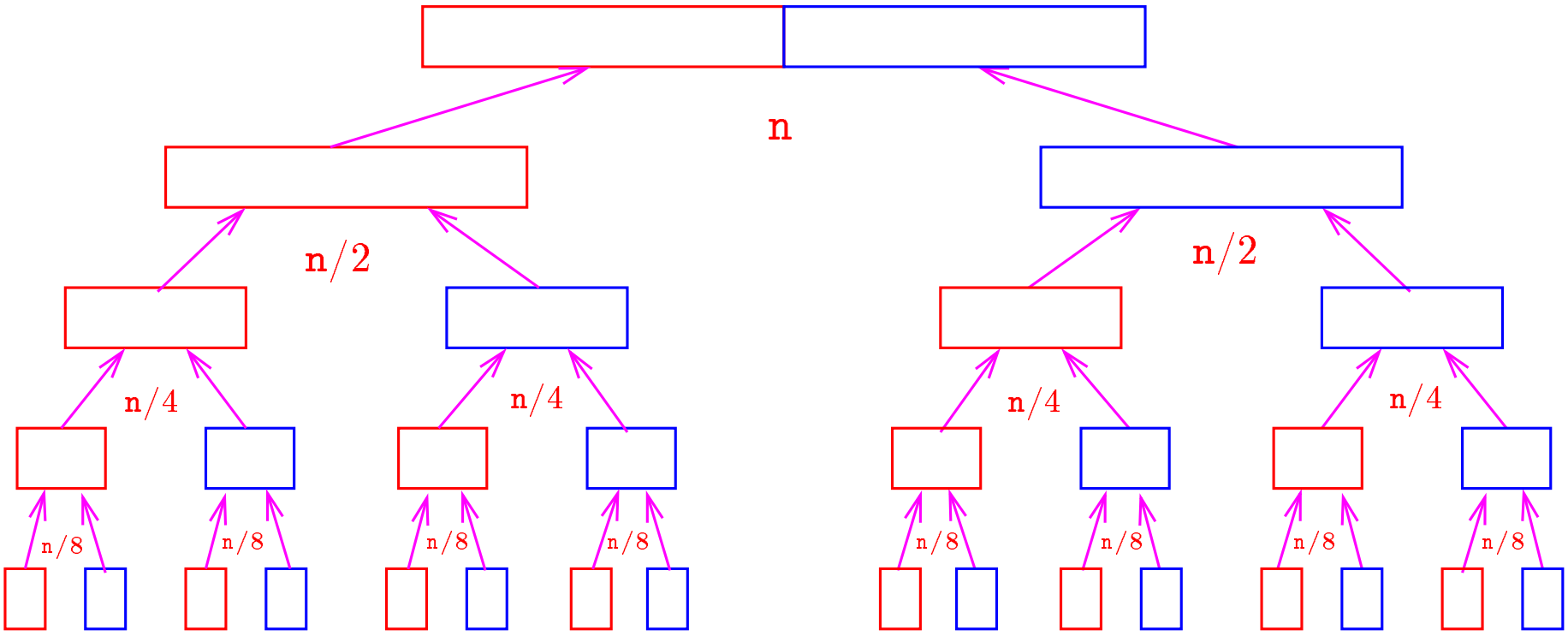


Correção

```
void mergeSort (int p, int r, int v[]) {
1   if (p < r-1) {
2       int q = (p + r)/2;
3       mergeSort(p, q, v);
4       mergeSort(q, r, v);
5       intercala(p, q, r, v);
    }
}
```

A função está correta?
A correção da função, que se apoia na correção do `intercala`, pode ser demonstrada por indução em $n := r - p$.

Consumo de tempo: versão MAC0121



Consumo de tempo: versão MAC0121

O consumo de tempo em cada nível da recursão é proporcional a n.

Há cerca de $\lg n$ níveis de recursão.

nível	consumo de tempo (proporcional a)	
1	$\approx n$	$= n$
2	$\approx n/2 + n/2$	$= n$
3	$\approx n/4 + n/4 + n/4 + n/4$	$= n$
$\cdots$	$\cdots$	
$\lg n$	$\approx 1 + 1 + 1 + 1 \cdots + 1 + 1$	$= n$
Total	$\approx n \lg n = O(n \lg n)$	

Consumo de tempo: outra versão

```
void mergeSort (int p, int r, int v[]) {
1   if (p < r-1) {
2       int q = (p + r)/2;
3       mergeSort(p, q, v);
4       mergeSort(q, r, v);
5       intercala(p, q, r, v);
    }
}
```

Consumo de tempo?

$T(n) :=$ consumo de tempo quando $n = r - p$

Consumo de tempo: outra versão

```
void mergeSort (int p, int r, int v[]) {
1   if (p < r-1) {
2       int q = (p + r)/2;
3       mergeSort(p, q, v);
4       mergeSort(q, r, v);
5       intercala(p, q, r, v);
    }
}
```

linha	consumo na linha (proporcional a)
1	?
2	?
3	?
4	?
5	?

$T(n) = ?$

Consumo de tempo: outra versão

```
void mergeSort (int p, int r, int v[]) {
1   if (p < r-1) {
2       int q = (p + r)/2;
3       mergeSort(p, q, v);
4       mergeSort(q, r, v);
5       intercala(p, q, r, v);
    }
}
```

linha	consumo na linha (proporcional a)
1	$= 1$
2	$= 1$
3	$= T(\lfloor n/2 \rfloor)$
4	$= T(\lceil n/2 \rceil)$
5	$= n$

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n + 2$$

Consumo de tempo: outra versão

$\mathtt{T}(\mathtt{n})$:= consumo de tempo quando $\mathtt{n} = \mathtt{r} - \mathtt{p}$

$$\mathtt{T}(1) = 1$$
$$\mathtt{T}(\mathtt{n}) = \mathtt{T}(\lceil \mathtt{n}/2 \rceil) + \mathtt{T}(\lfloor \mathtt{n}/2 \rfloor) + \mathtt{n} \quad \text{para } \mathtt{n} = 2, 3, 4, \ldots$$

Solução: $\mathtt{T}(\mathtt{n})$ é $\mathrm{O}(\mathtt{n} \log \mathtt{n})$.

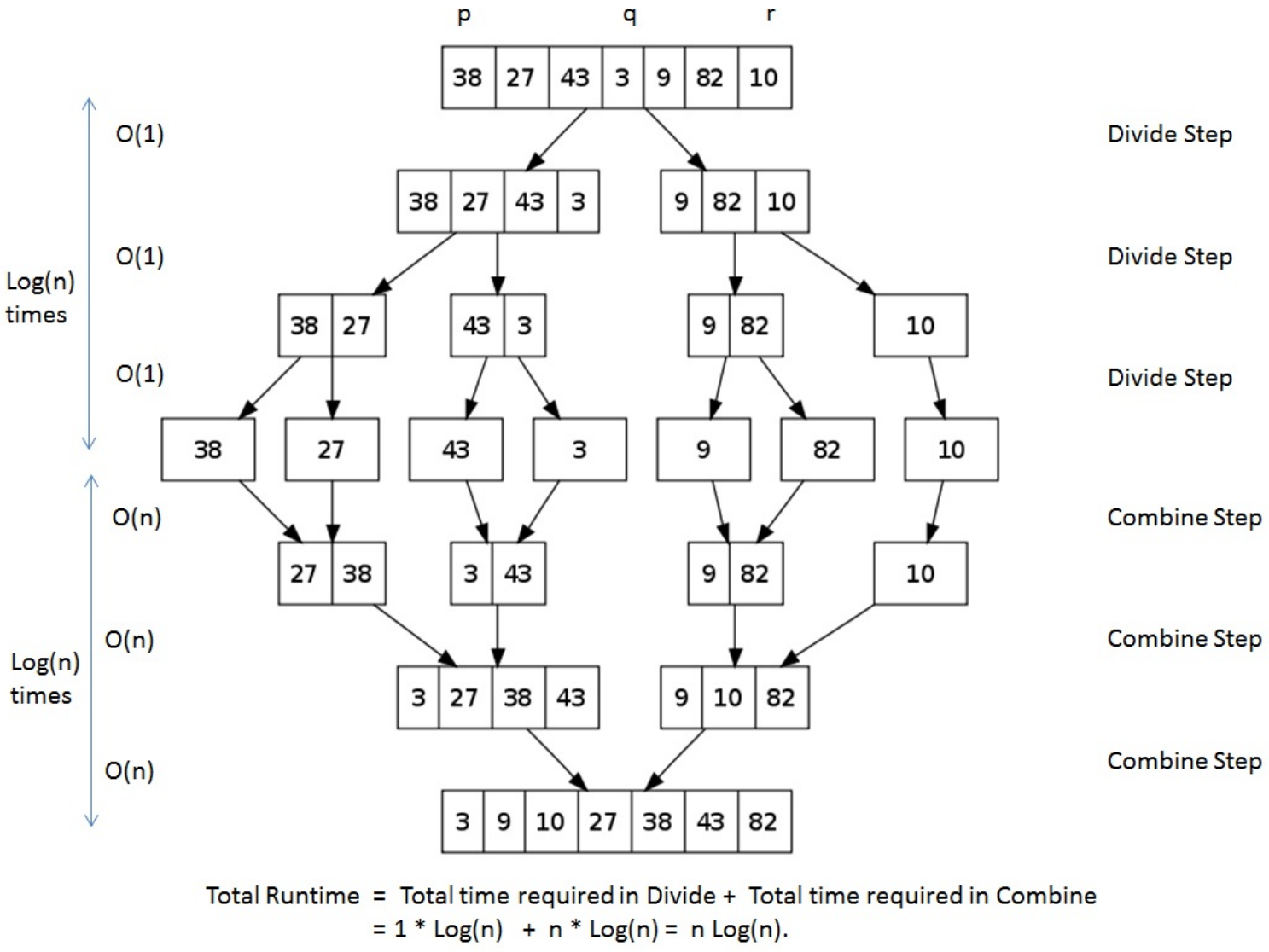
Demonstração: ...

Conclusão

O consumo de tempo da função `mergeSort` é proporcional a $n \lg n$.

O consumo de tempo da função `mergeSort` é $O(n \lg n)$.

Consumo de tempo



Total Runtime = Total time required in Divide + Total time required in Combine
= 1 * Log(n) + n * Log(n) = n Log(n).

Fonte: http://images.1233.tw/in-place-quicksort-algorithm/

Divisão e conquista

Algoritmos por divisão-e-conquista têm três passos em cada nível da recursão:

Dividir: o problema é dividido em subproblemas de tamanho menor;

Conquistar: os subproblemas são resolvidos recursivamente e subproblemas "pequenos" são resolvidos diretamente;

Combinar: as soluções dos subproblemas são combinadas para obter uma solução do problema original.

Exemplo: ordenação por intercalação (`mergeSort`).

mergeSort: versão iterativa

```
void mergeSort (int n, int v[]){
   int p, r;
   int b = 1;
   while (b < n) {
       p = 0;
       while (p + b < n) {
           r = p + 2*b;
           if (r > n) r = n;
           intercala(p, p+b, r, v);
           p = p + 2*b;
       }
       b = 2*b;
   }
}
```