CI165 — Análise de Algoritmos Recursivos

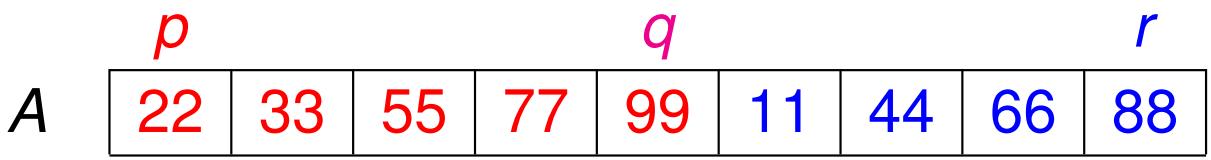
André Vignatti

Ordenação por intercalação

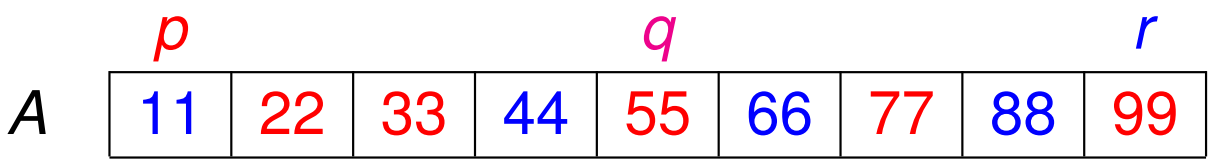
Q que significa intercalar dois (sub)vetores ordenados?

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

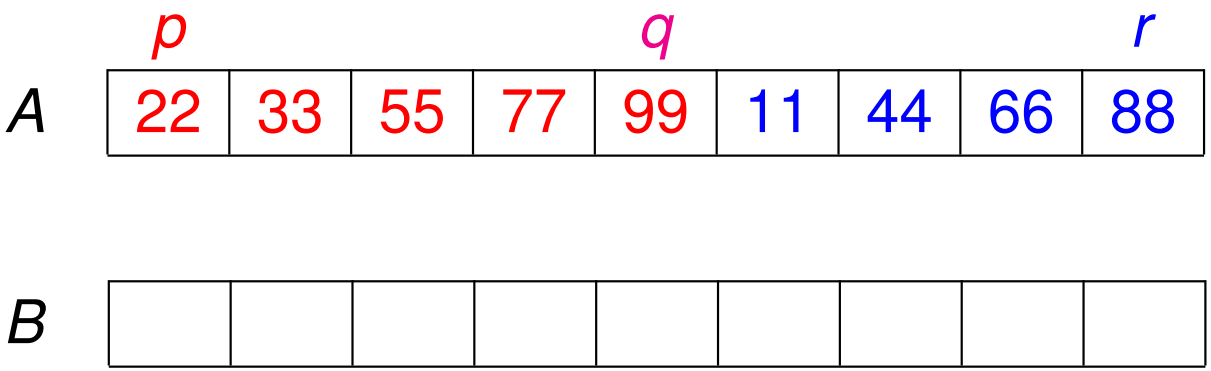
Entrada:



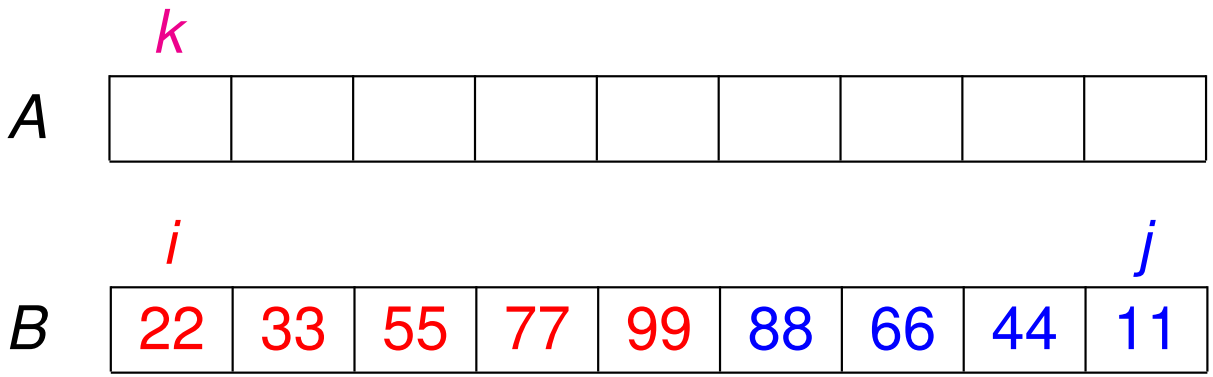
Saída:



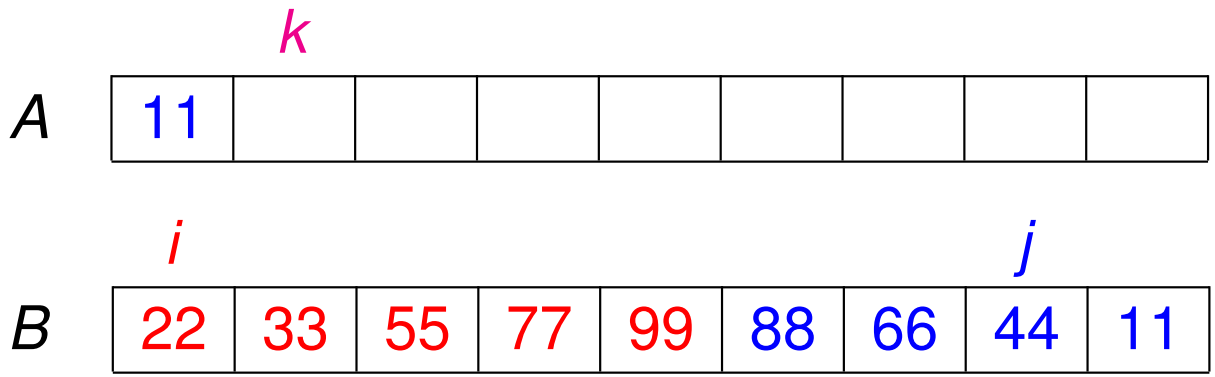
Intercalação



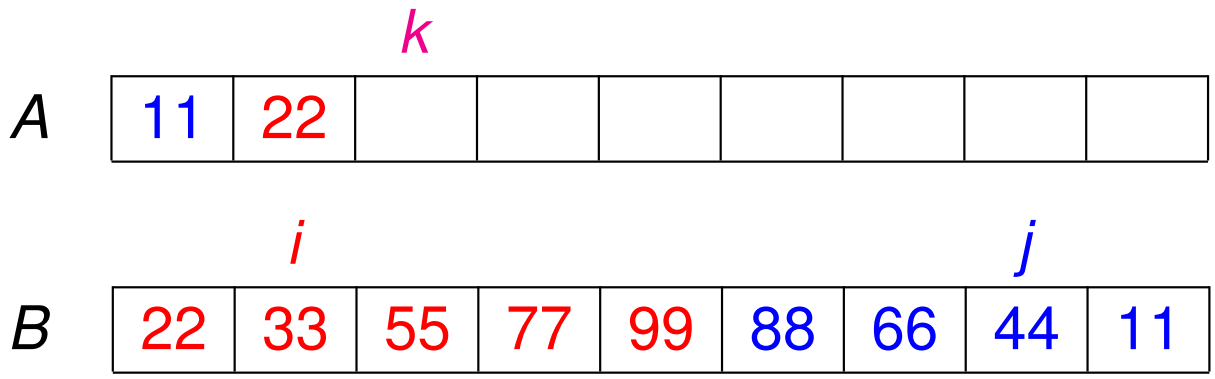
Intercalação



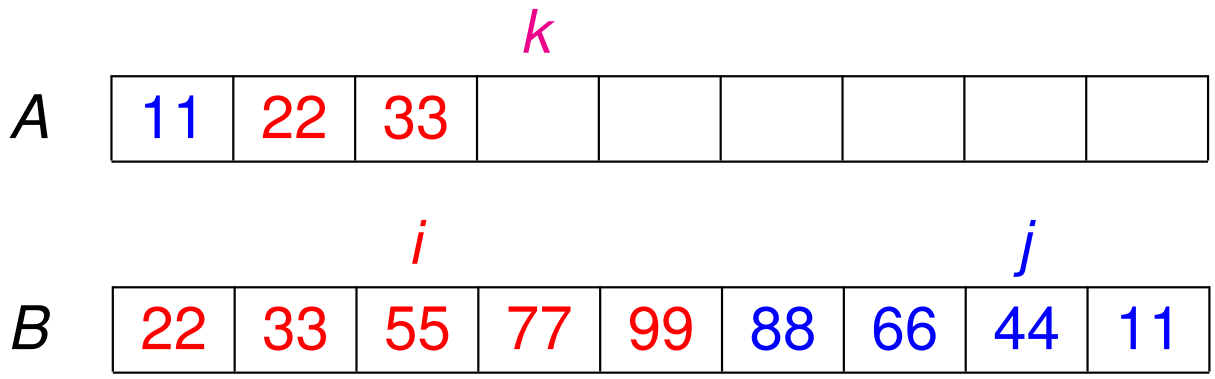
Intercalação



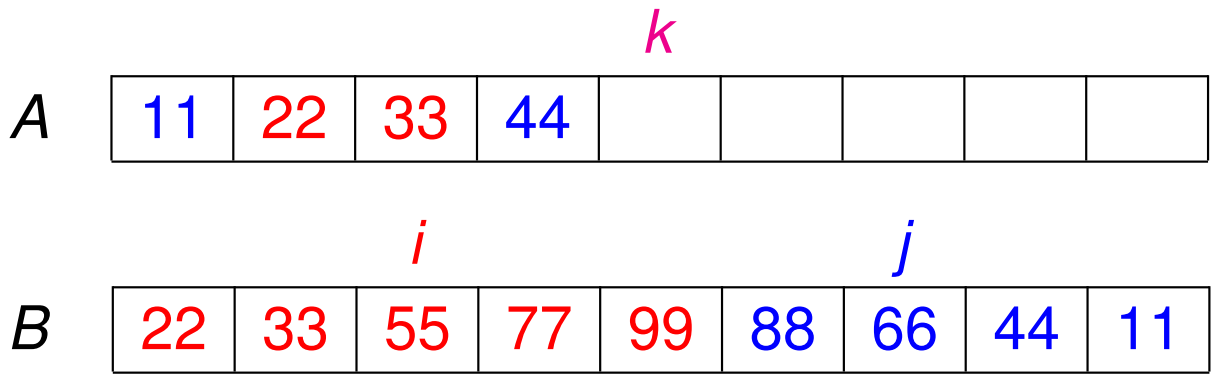
Intercalação



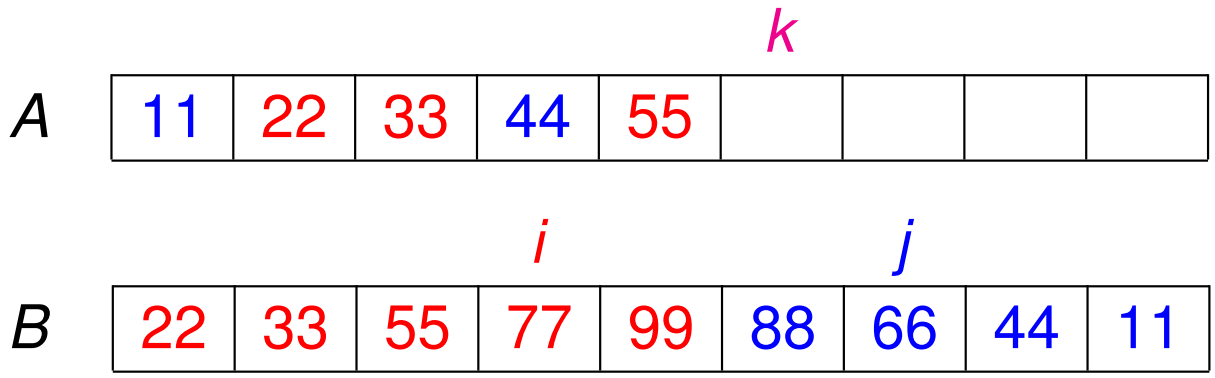
Intercalação



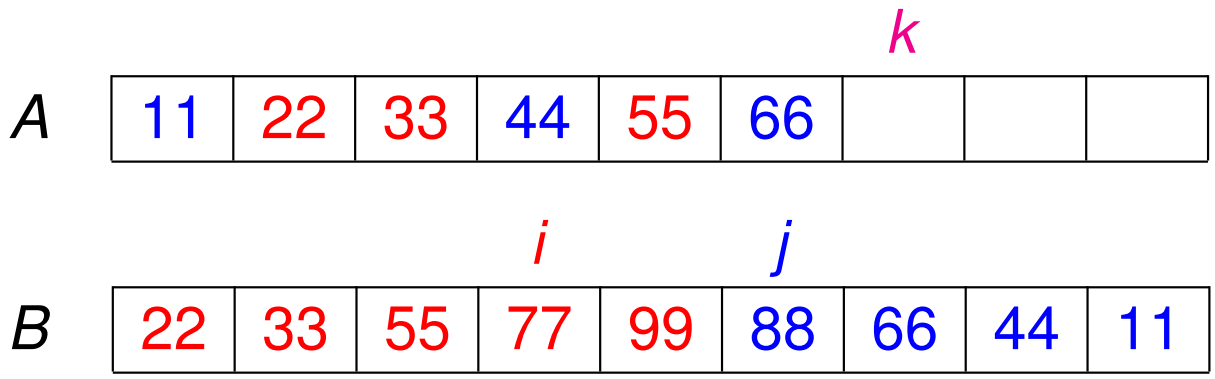
Intercalação



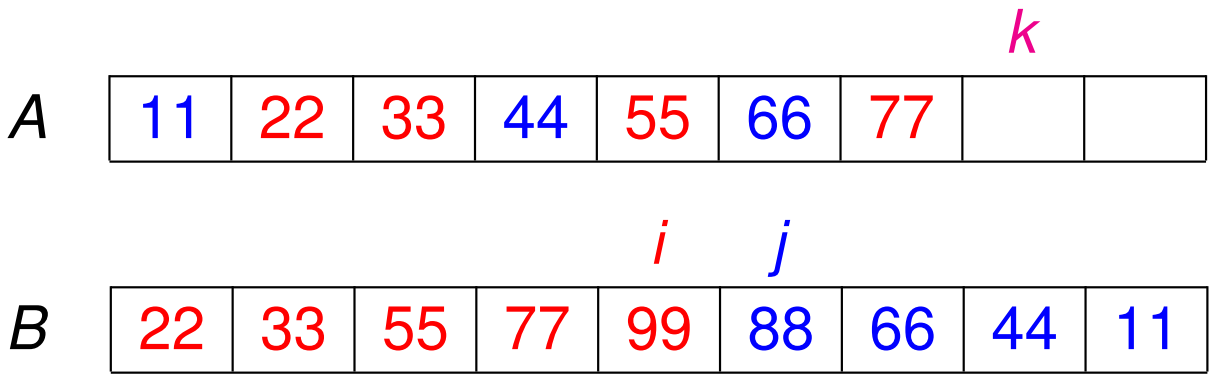
Intercalação



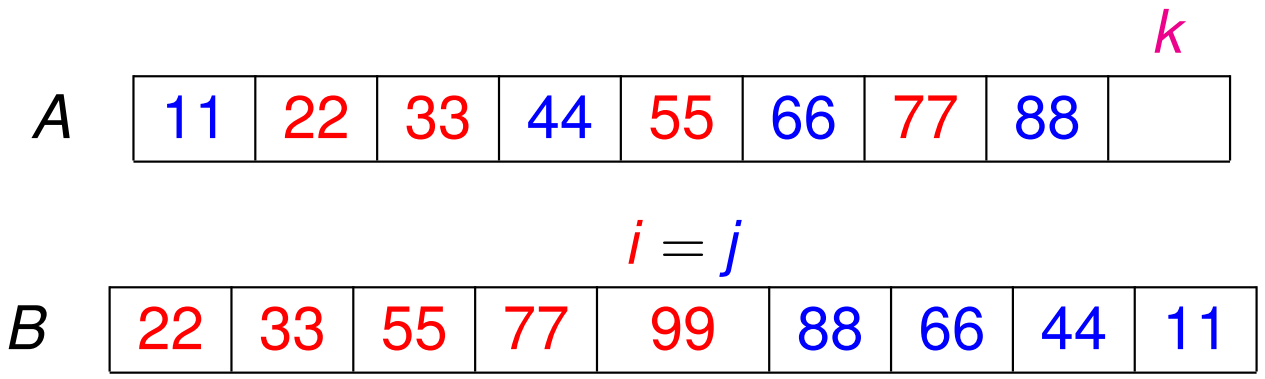
Intercalação



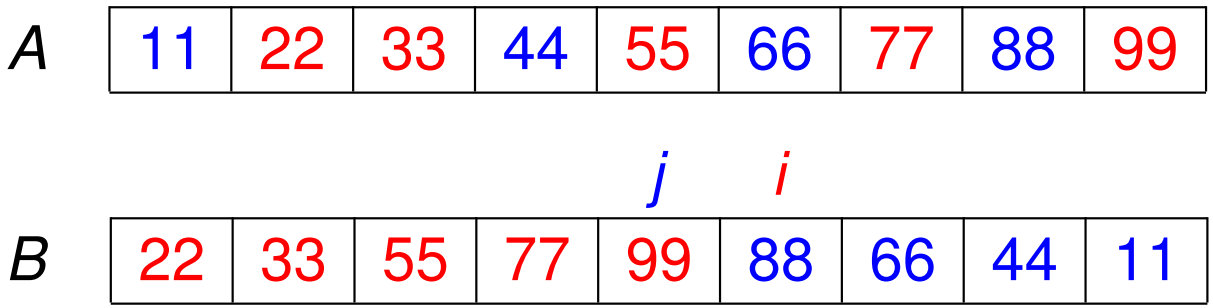
Intercalação



Intercalação



Intercalação

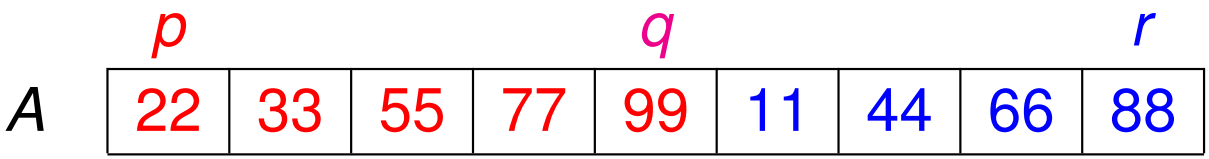


Pseudo-código

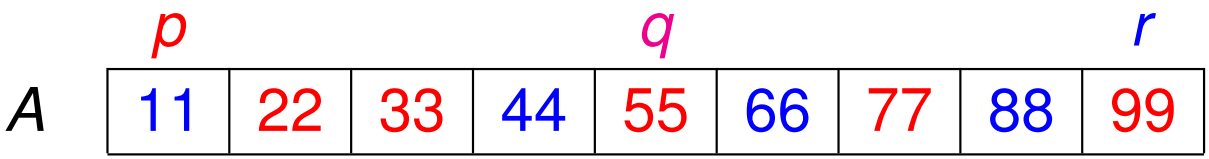
```
INTERCALA(A, p, q, r)
 1   para i ← p até q faça
 2       B[i] ← A[i]
 3   para j ← q + 1 até r faça
 4       B[r + q + 1 − j] ← A[j]
 5   i ← p
 6   j ← r
 7   para k ← p até r faça
 8       se B[i] ≤ B[j]
 9           então A[k] ← B[i]
10                 i ← i + 1
11           senão A[k] ← B[j]
12                 j ← j − 1
```

Complexidade de Intercala

Entrada:



Saída:



Tamanho da entrada: $n = r - p + 1$

Consumo de tempo: $\Theta(n)$

Corretude de Intercala

Invariante principal de Intercala:

No começo de cada iteração do laço das linhas 7–12, vale que:

1. $A[p \ldots k-1]$ está ordenado,
2. $A[p \ldots k-1]$ contém todos os elementos de $B[p \ldots i-1]$ e de $B[j+1 \ldots r]$,
3. $B[i] \geq A[k-1]$ e $B[j] \geq A[k-1]$.

Exercício. Prove que a afirmação acima é de fato um invariante de INTERCALA.

Exercício. (fácil) Mostre usando o invariante acima que INTERCALA é correto.

Algoritmos recursivos

> "To understand recursion, we must first understand recursion."
>
> *(anônimo)*

- O que é o paradigma de **divisão-e-conquista**?
- Como mostrar a corretude de um algoritmo recursivo?
- Como analisar o consumo de tempo de um algoritmo recursivo?
  - O que é uma **relação de recorrência**?
  - O que significa *resolver* uma relação de recorrência?

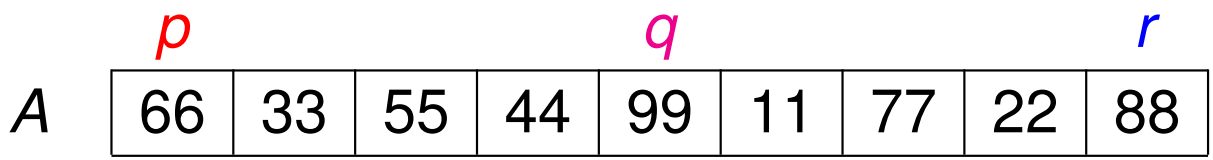
Recursão e o paradigma de divisão-e-conquista

- Um algoritmo recursivo resolve o problema chamando a si mesmo para resolver instâncias menores do mesmo problema.
- Algoritmos de divisão-e-conquista possuem três etapas em cada nível de recursão:
  1. **Divisão:** a instância do problema é dividida em instâncias de tamanho menor, gerando subproblemas.
  2. **Conquista:** cada subproblema é resolvido recursivamente.
  3. **Combinação:** as soluções dos subproblemas são combinadas para obter uma solução do problema original.

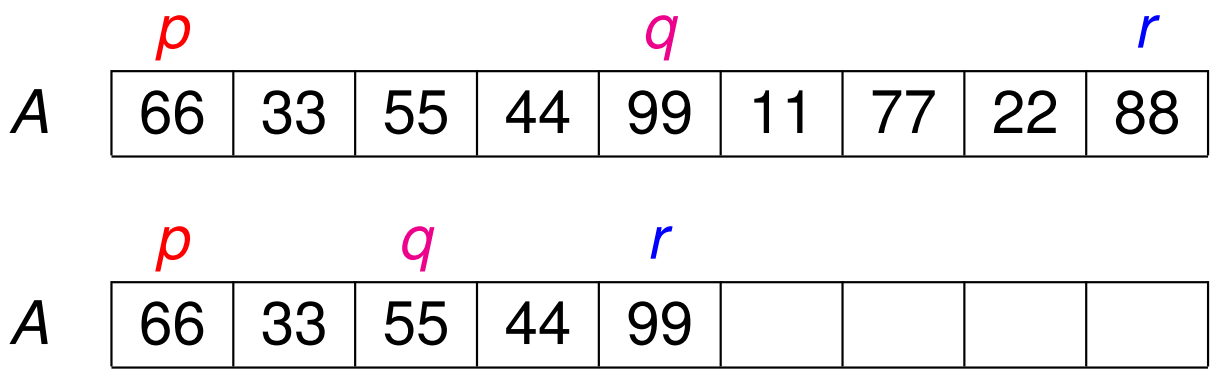
Exemplo de divisão-e-conquista: *Mergesort*

- Mergesort é um algoritmo de ordenação e um exemplo clássico do uso da técnica de divisão-e-conquista. (*to merge* = intercalar)
- Descrição do Mergesort em alto nível:
  1. **Divisão**: divida o vetor com n elementos em dois subvetores de tamanho $\lfloor n/2 \rfloor$ e $\lceil n/2 \rceil$
  2. **Conquista**: ordene os dois subvetores recursivamente usando o Mergesort
  3. **Combinação**: intercale os dois subvetores para obter um vetor ordenado usando o algoritmo Intercala

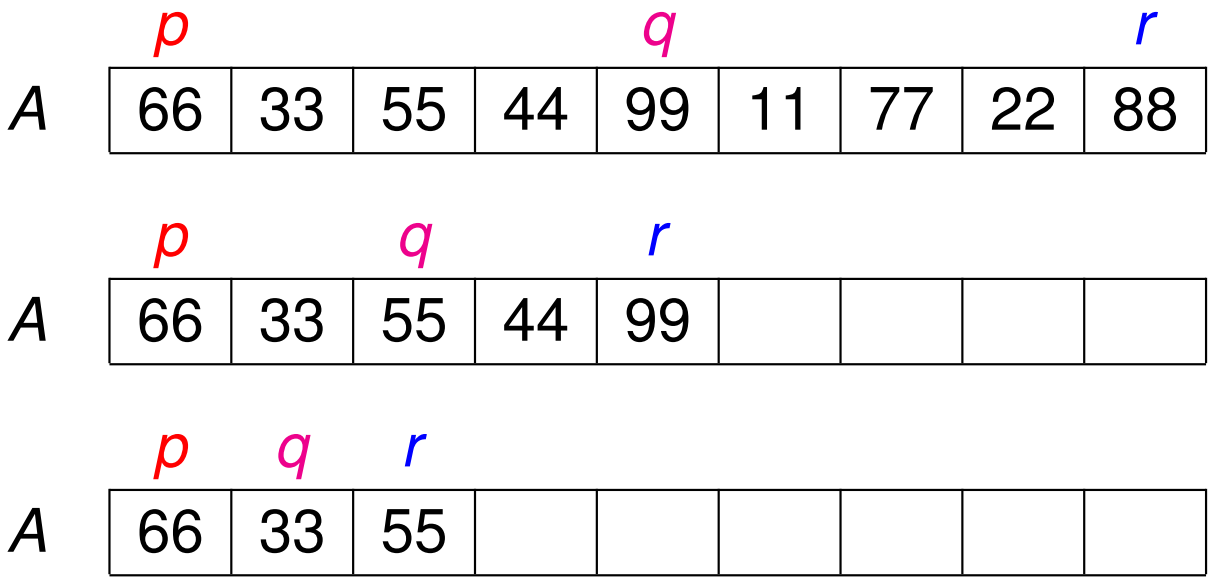
Mergesort



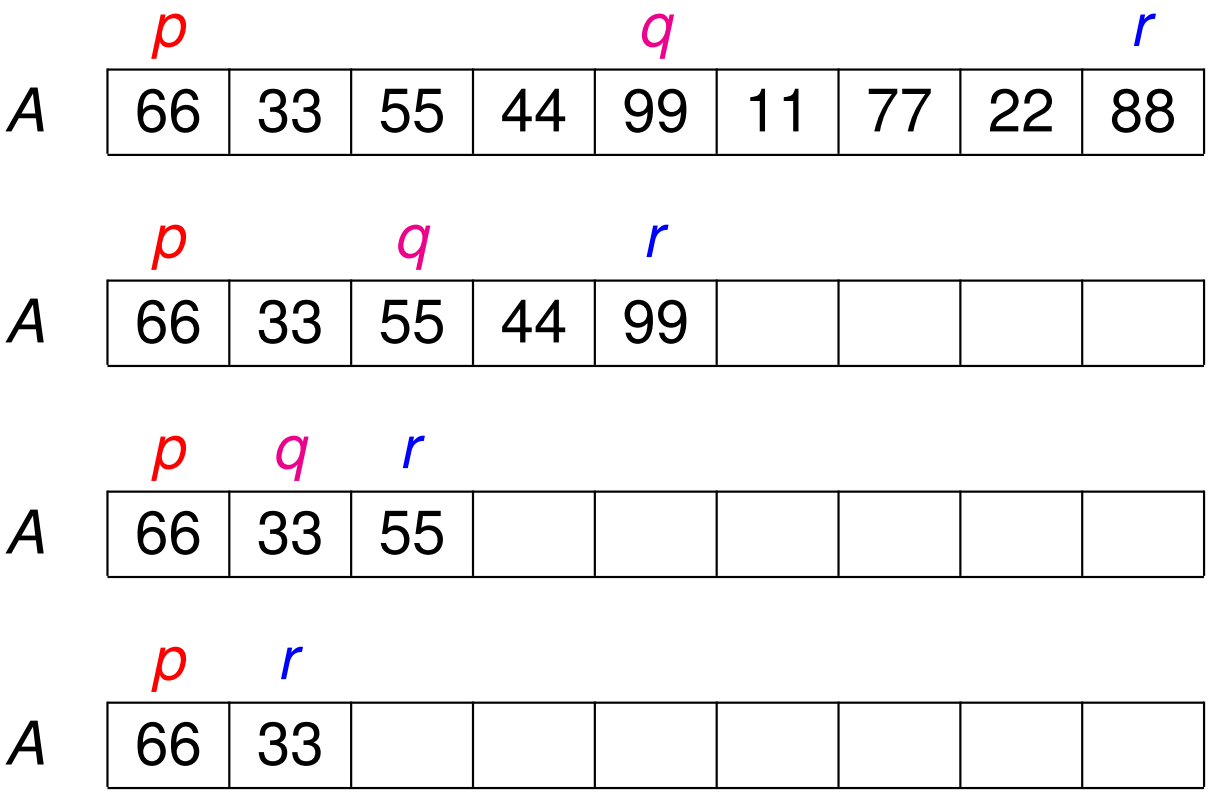
Mergesort



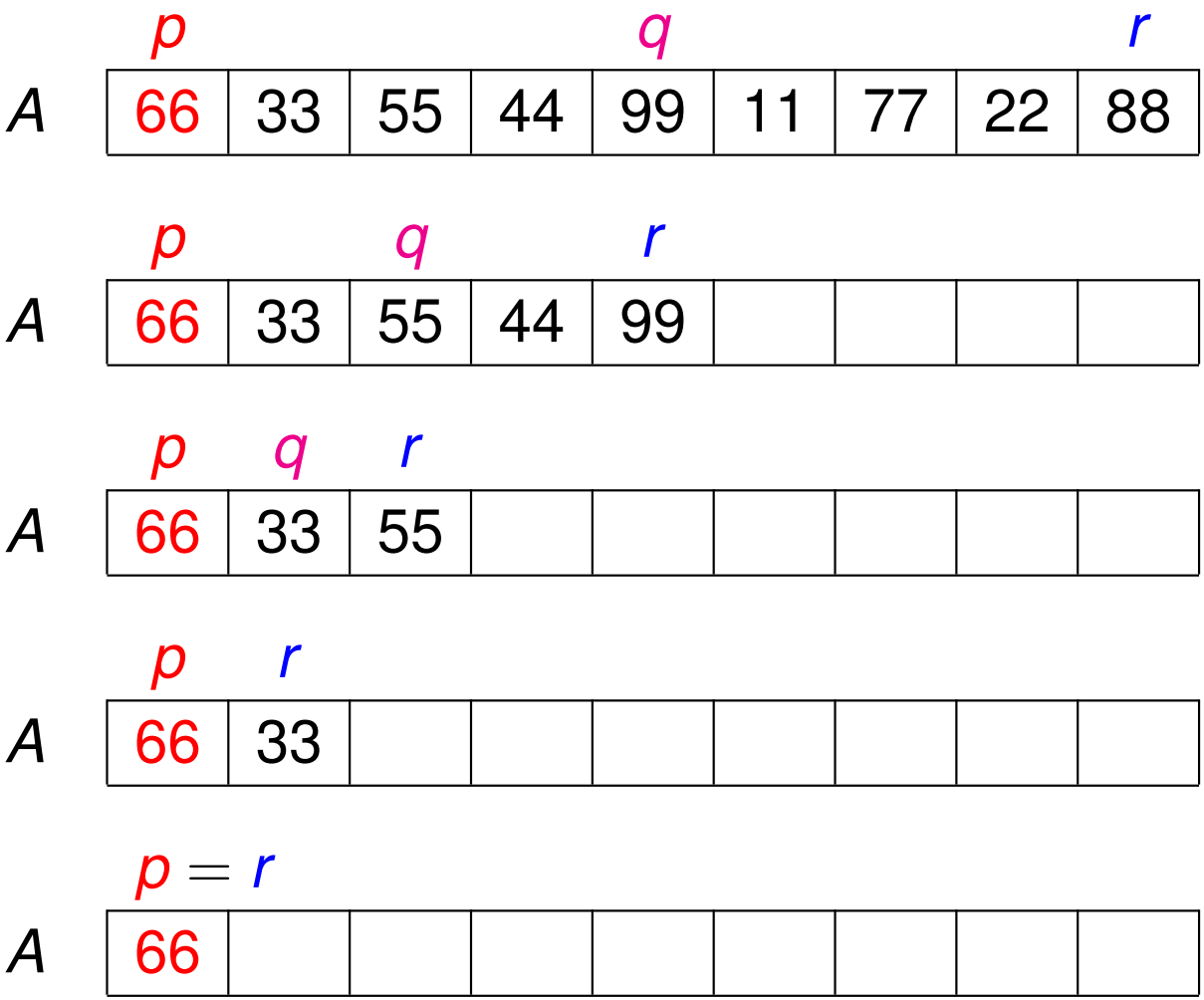
Mergesort



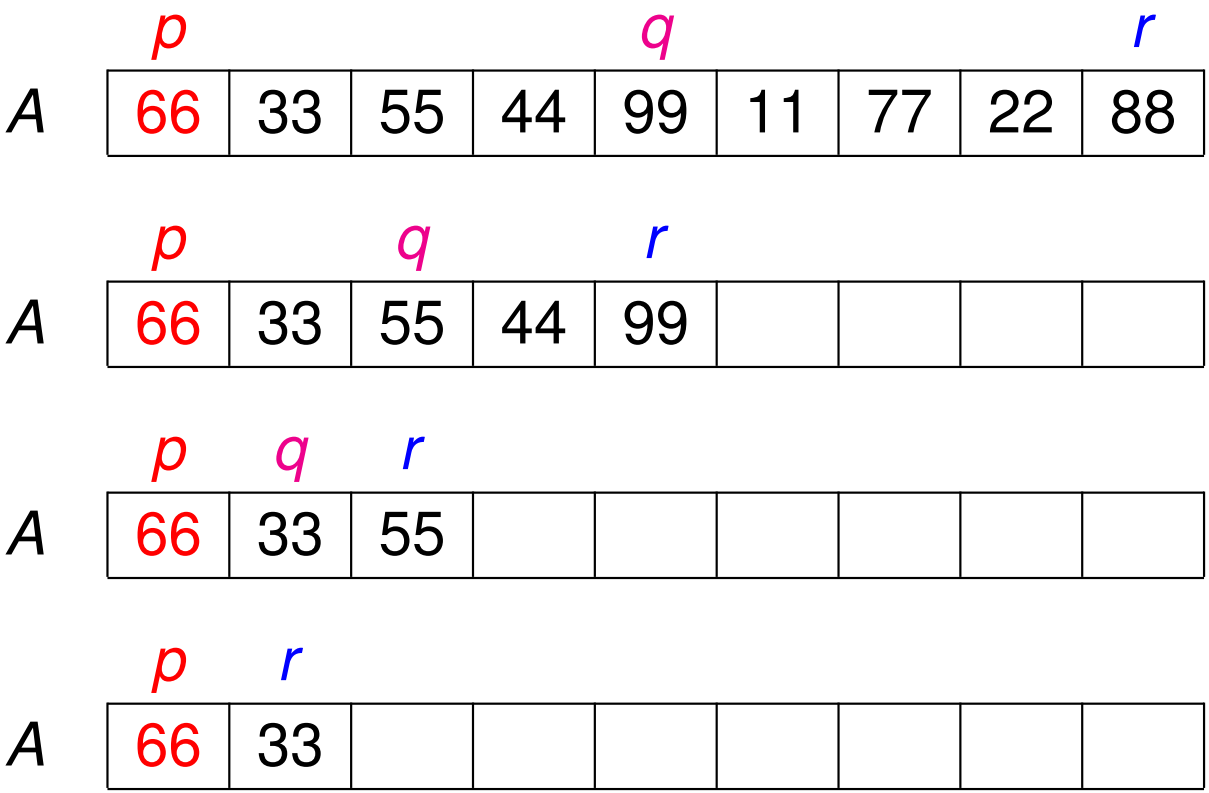
Mergesort


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

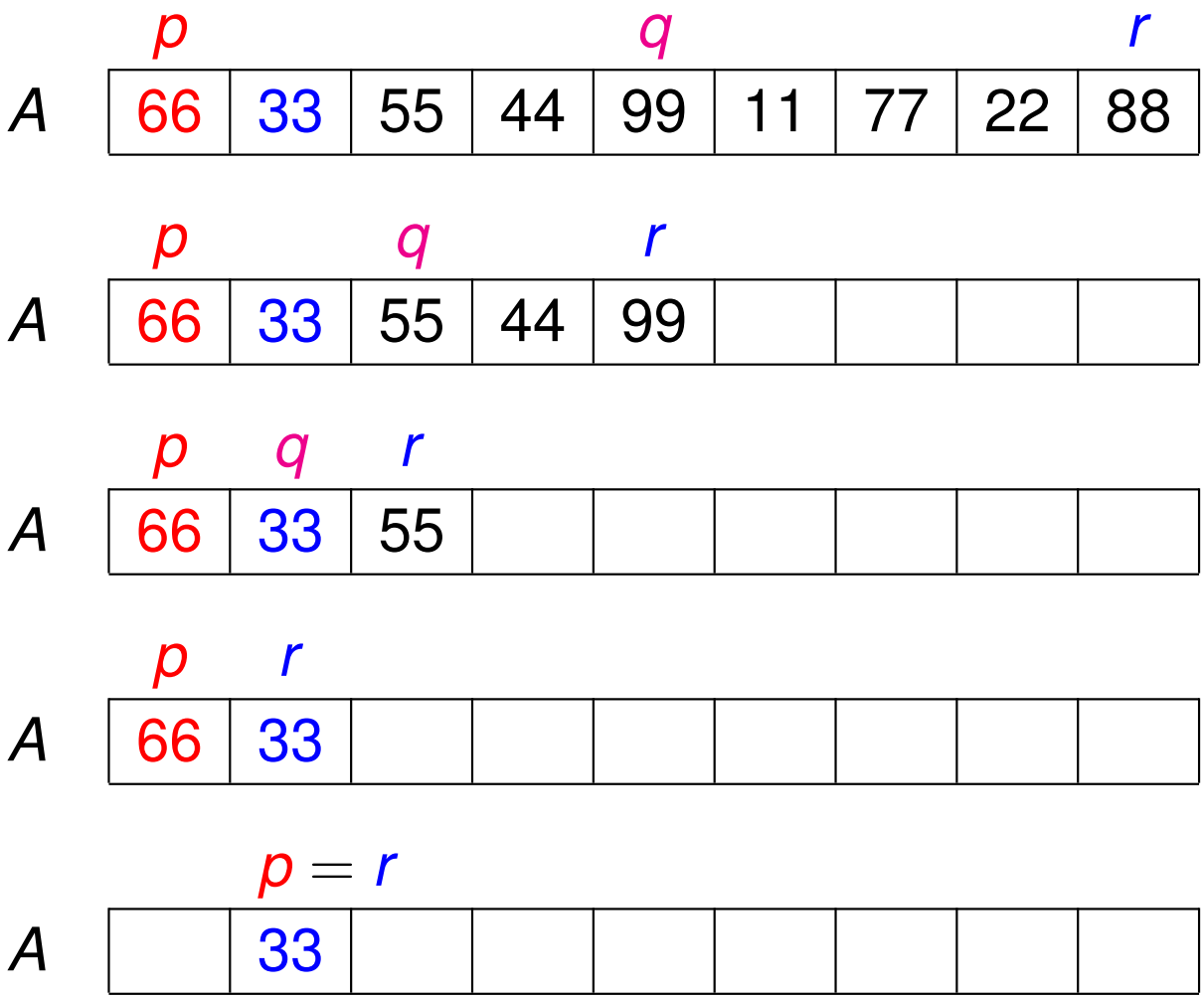
Mergesort


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

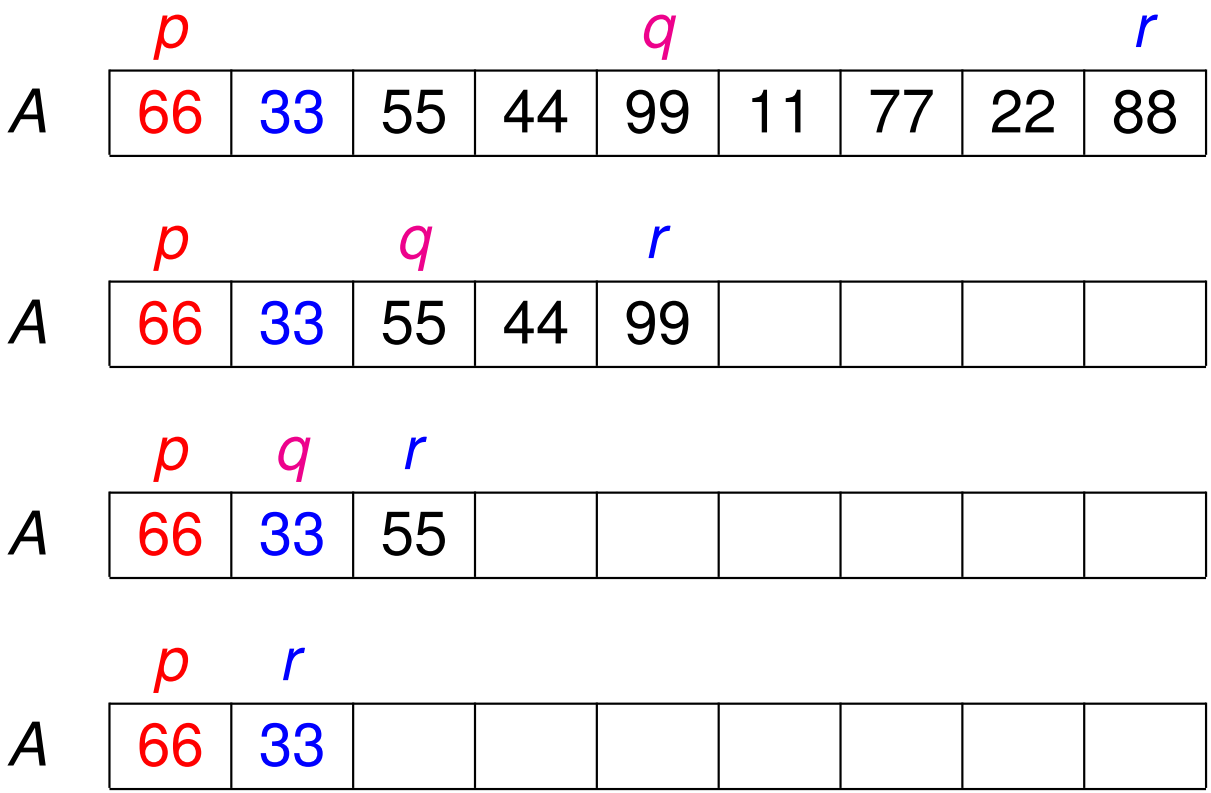
Mergesort


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

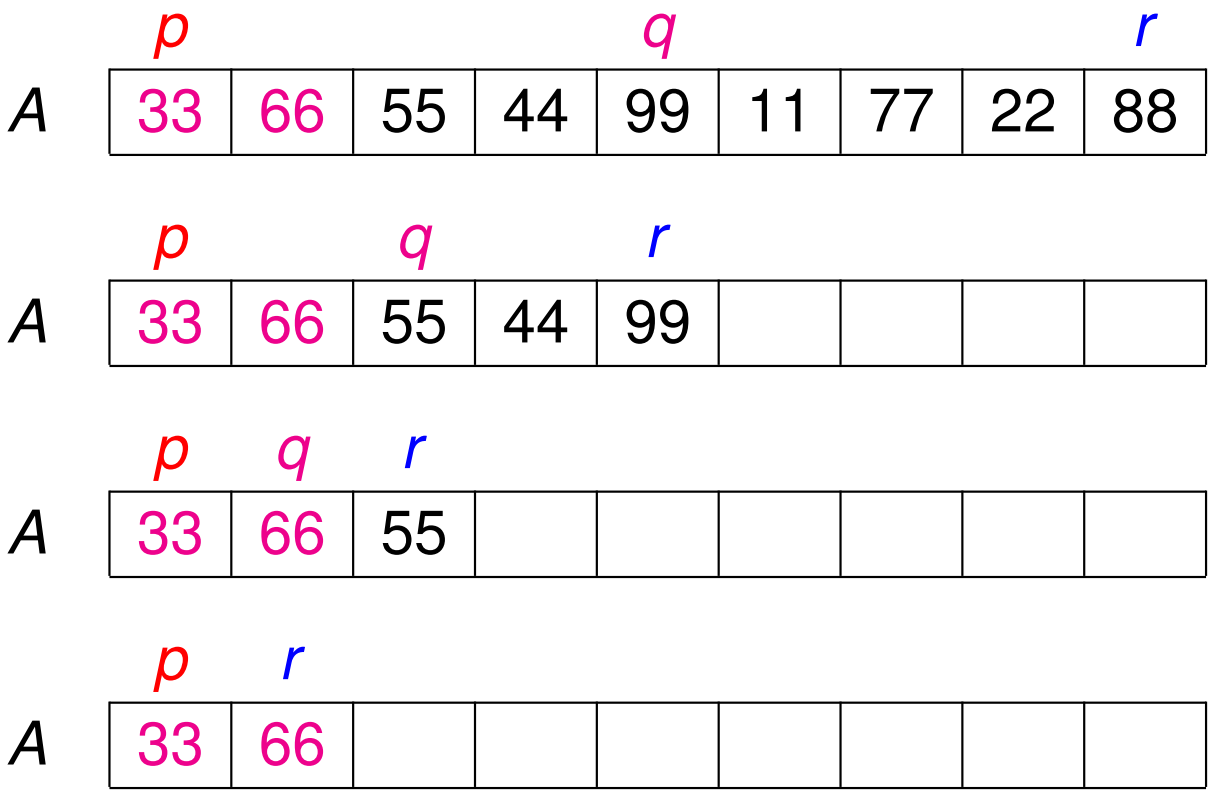
Mergesort


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

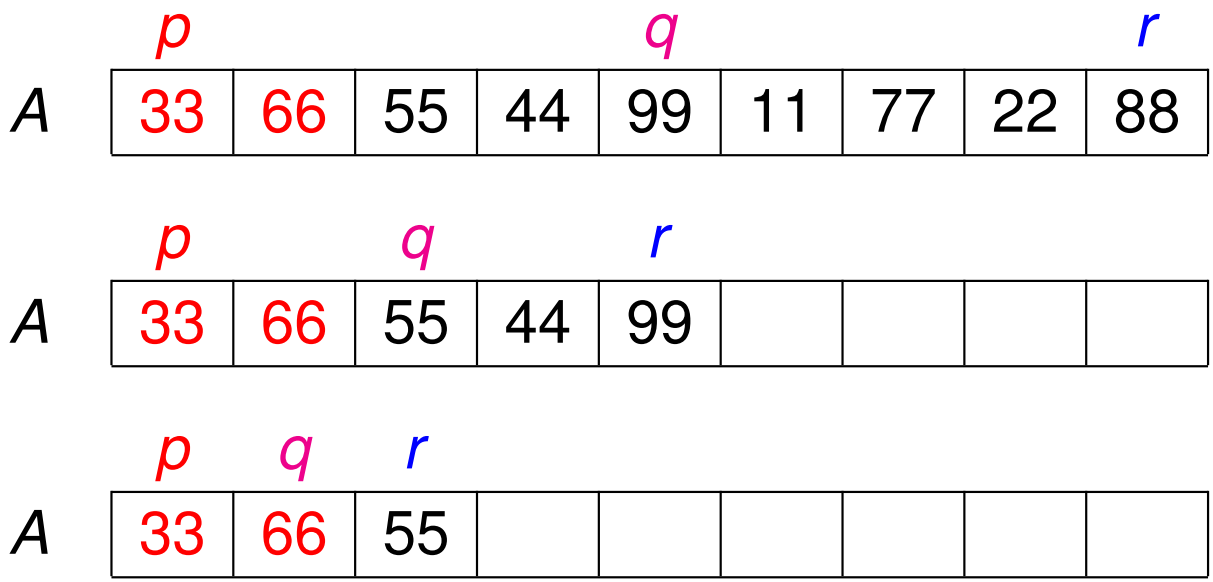
Mergesort


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

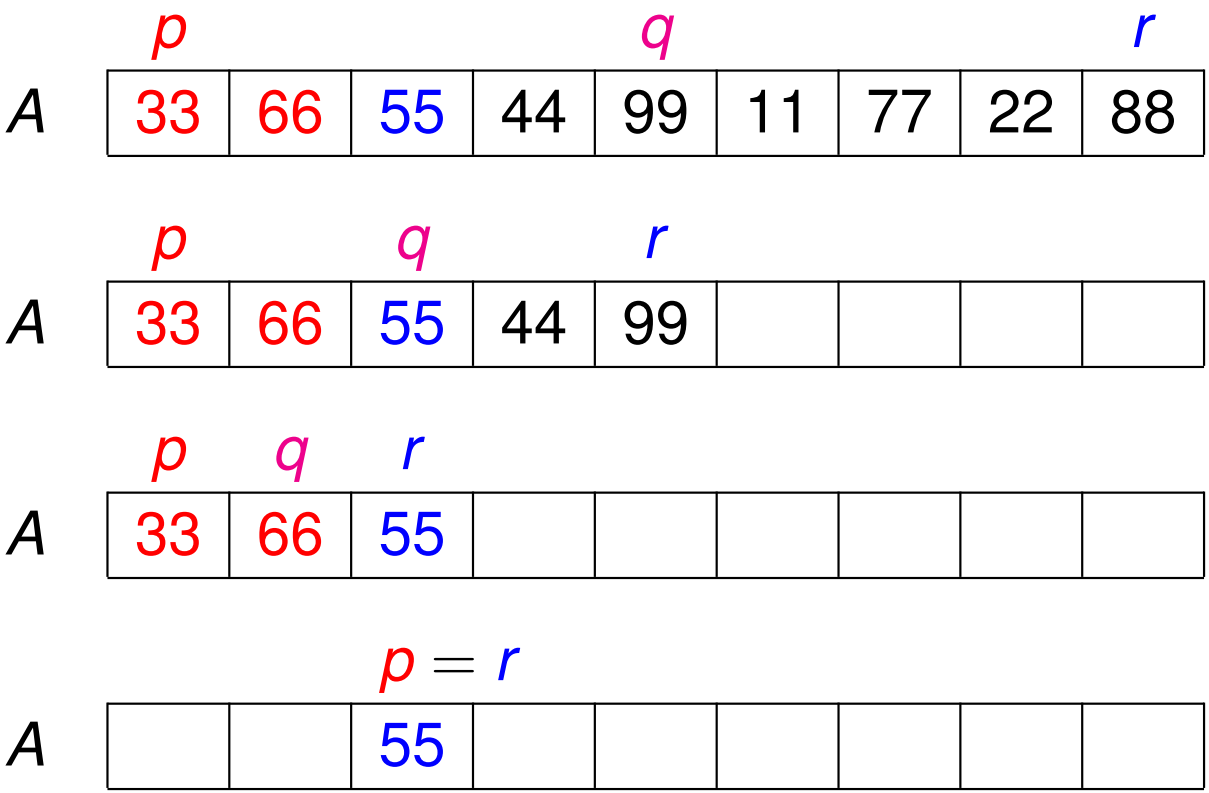
Mergesort


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

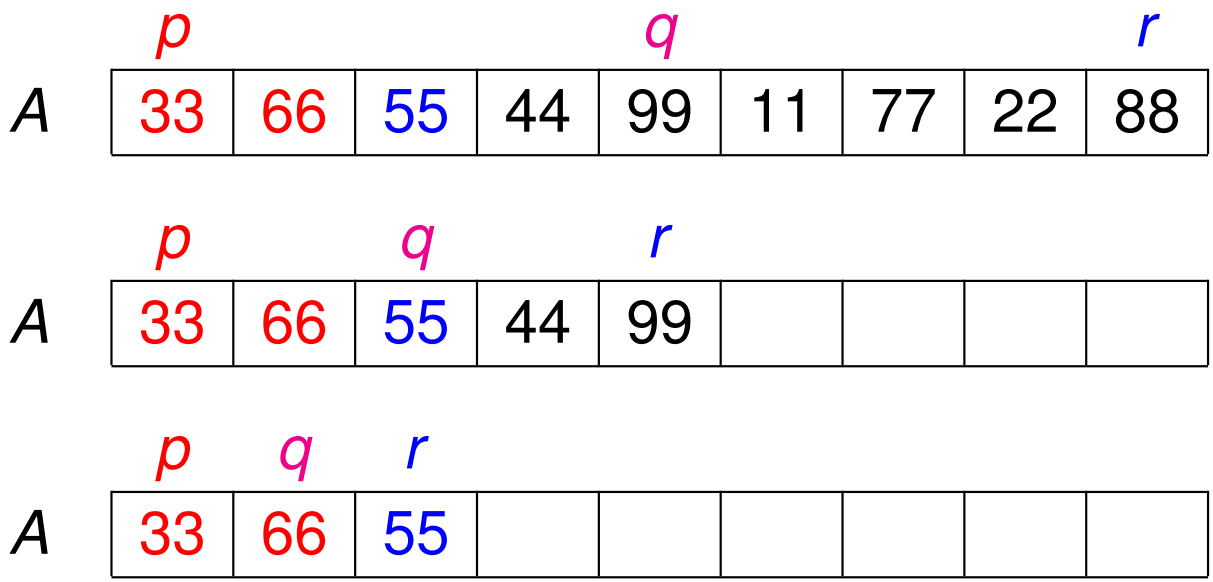
Mergesort



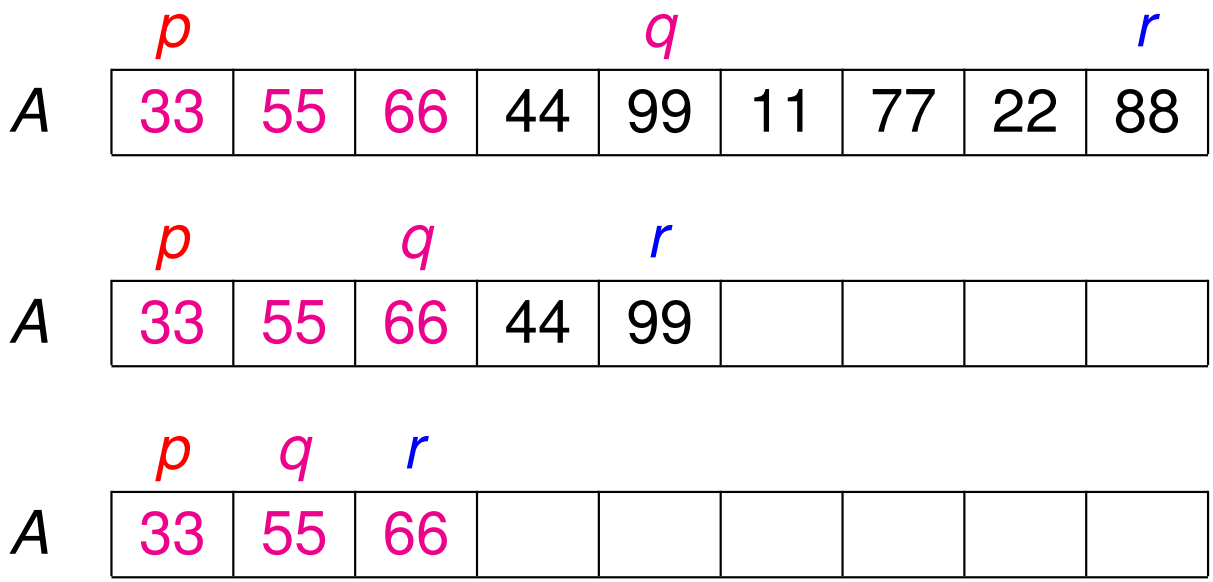
Mergesort


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

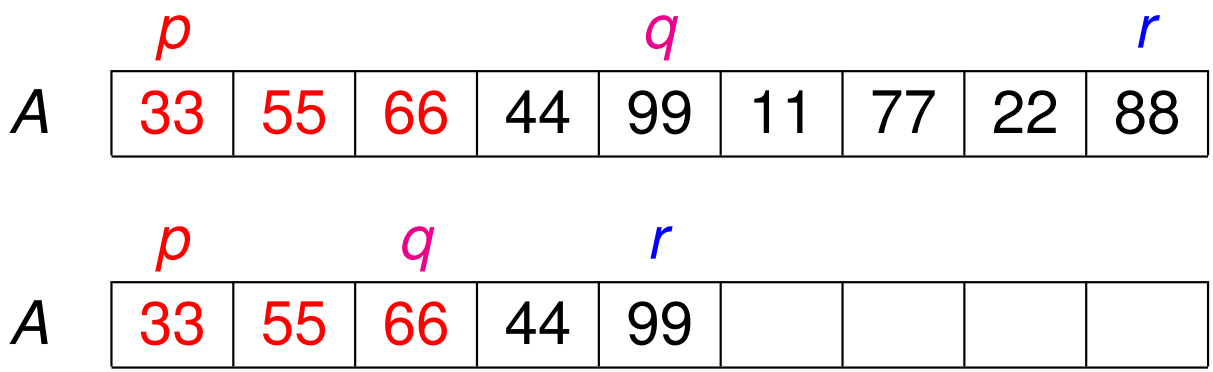
Mergesort



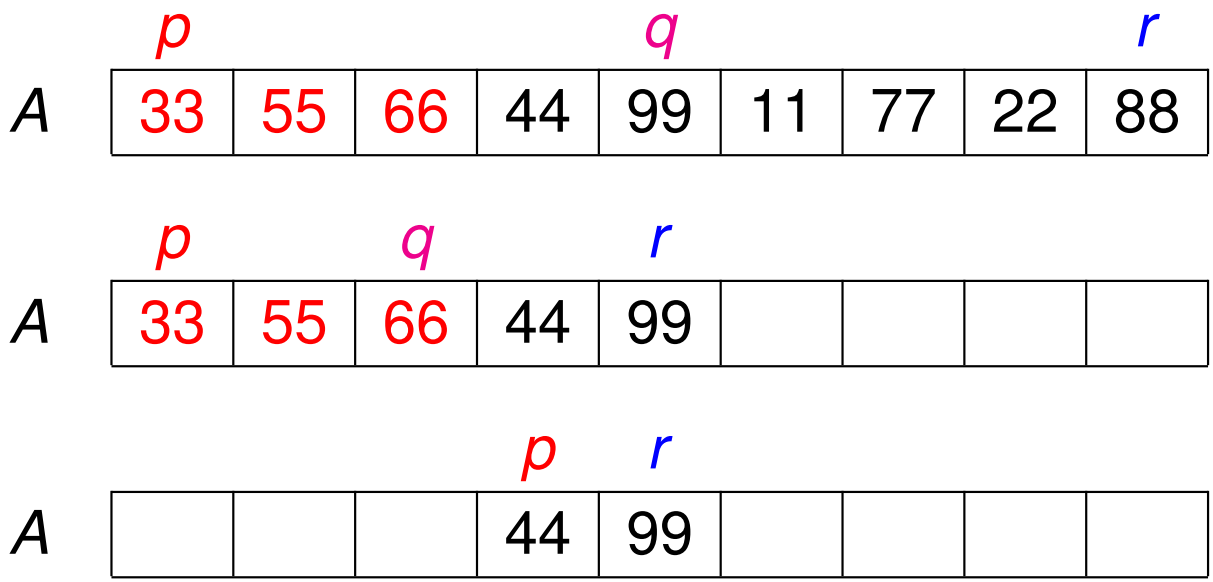
Mergesort



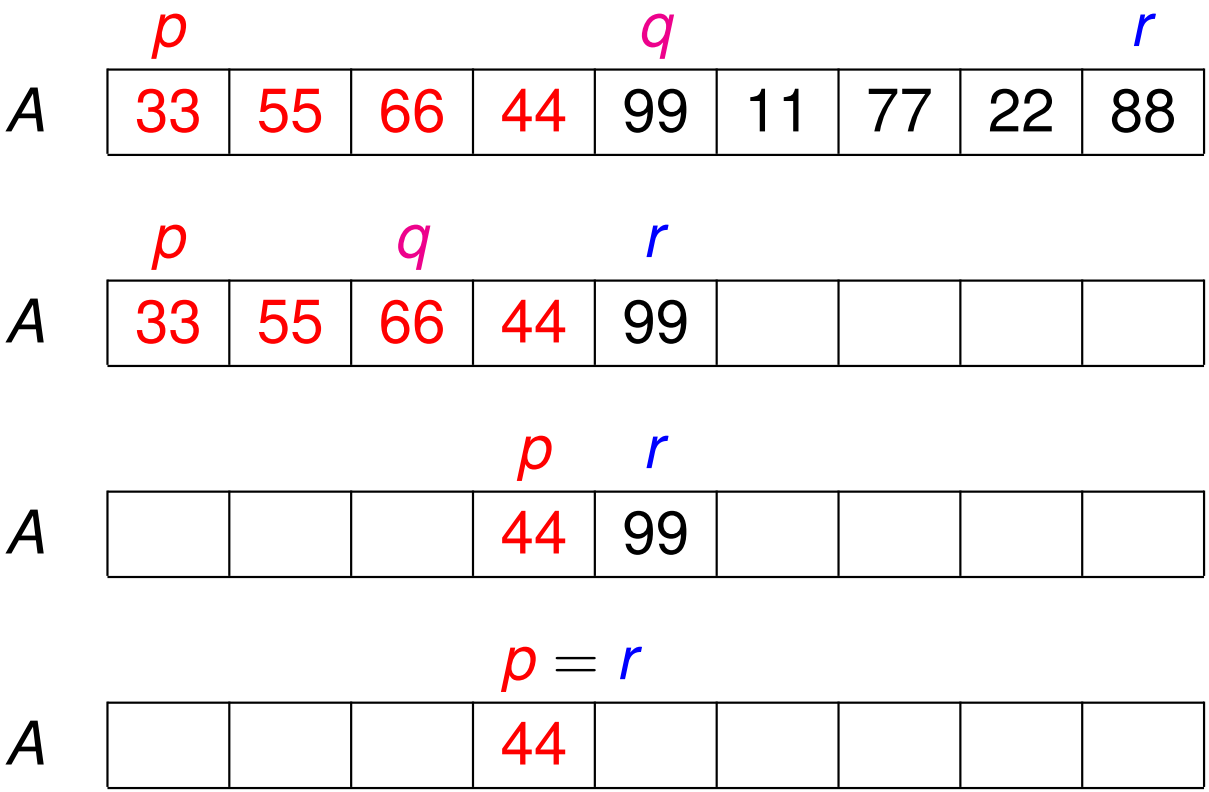
Mergesort



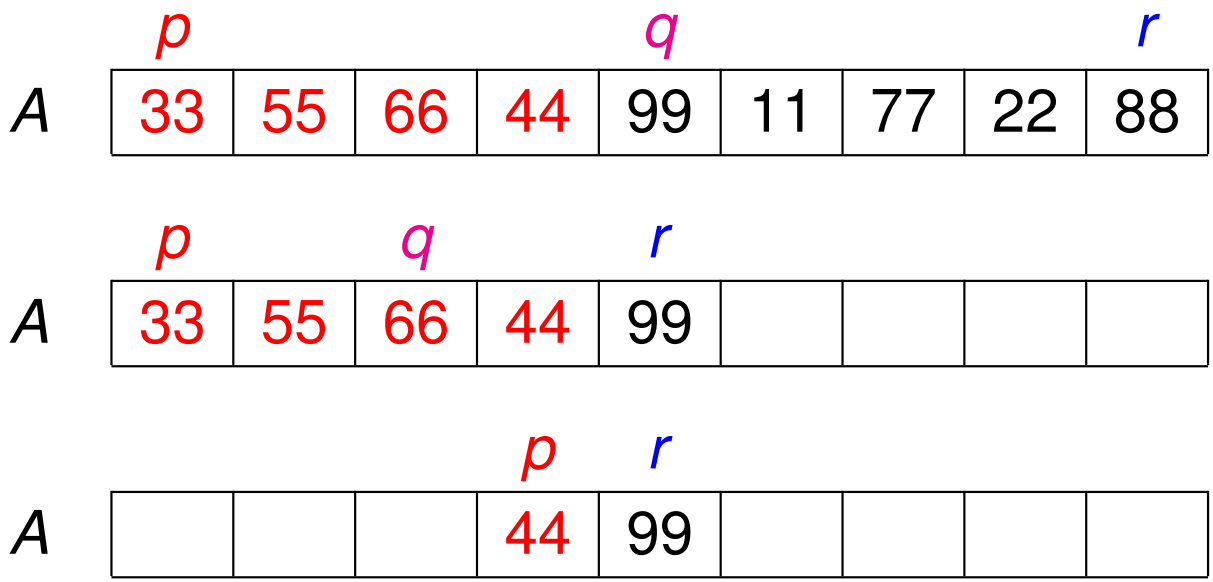
Mergesort



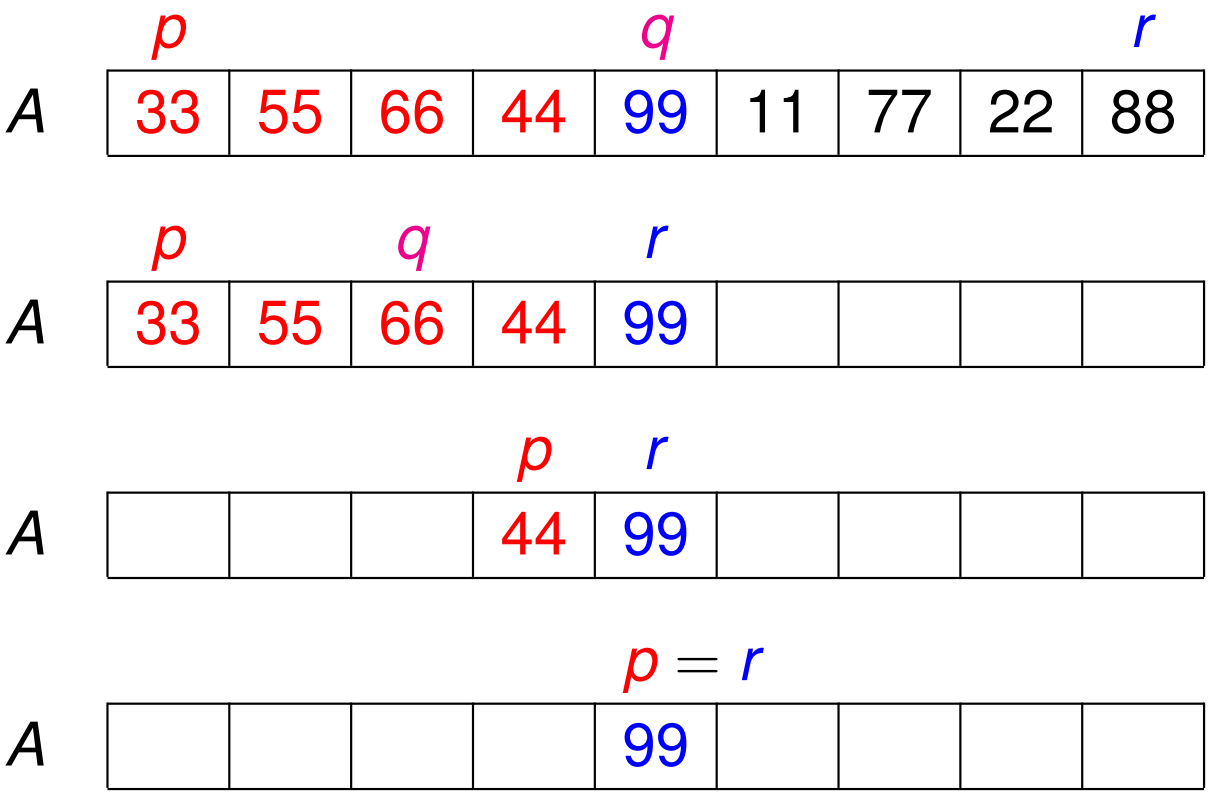
Mergesort



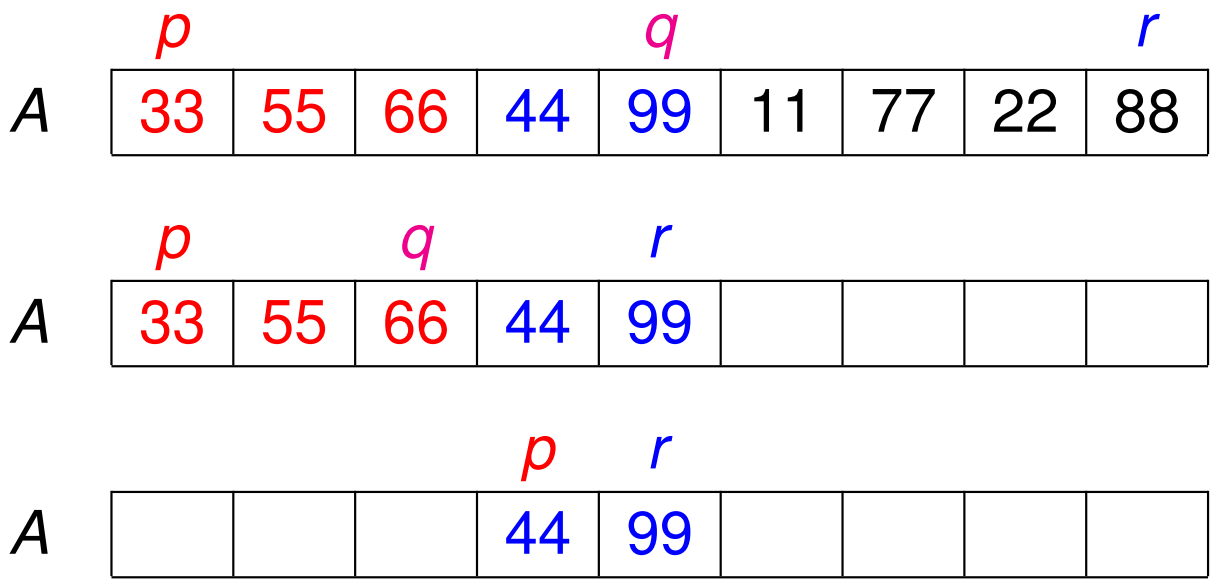
Mergesort



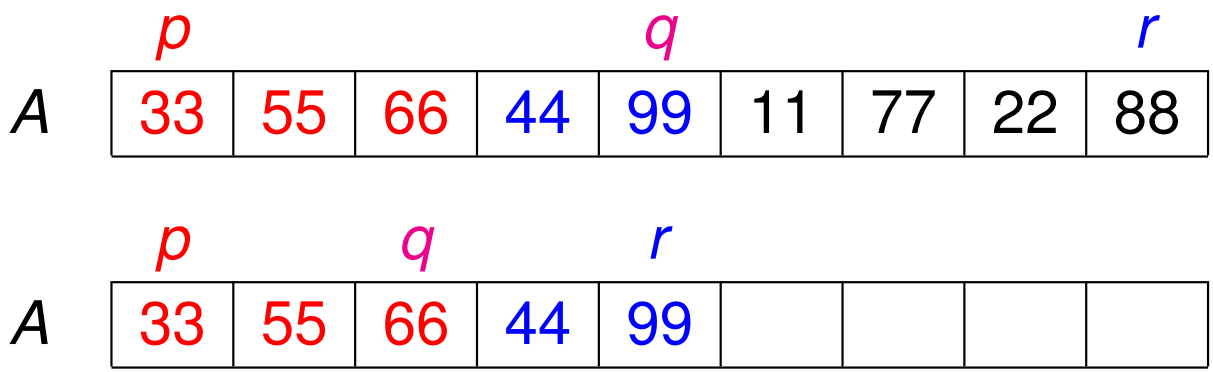
Mergesort



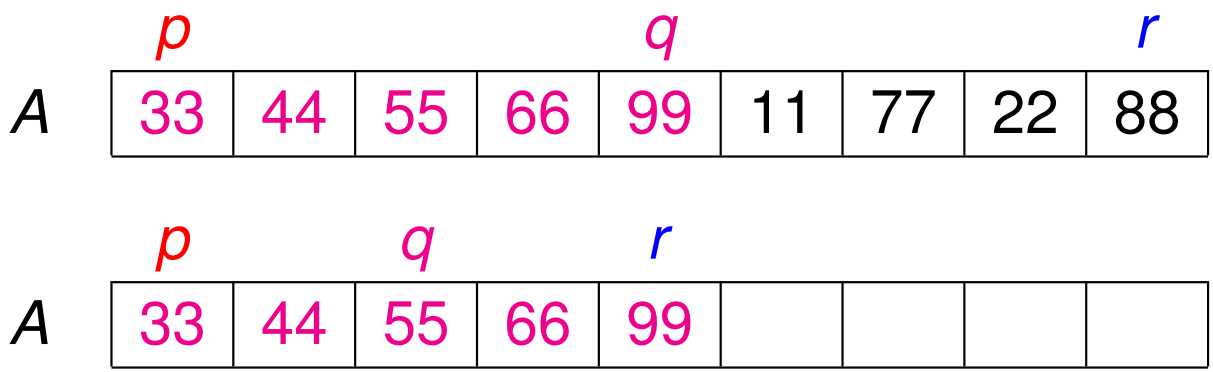
Mergesort



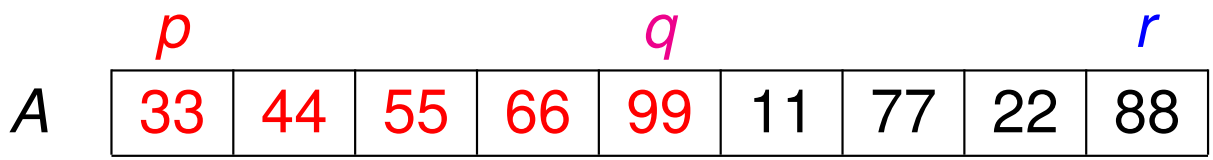
Mergesort



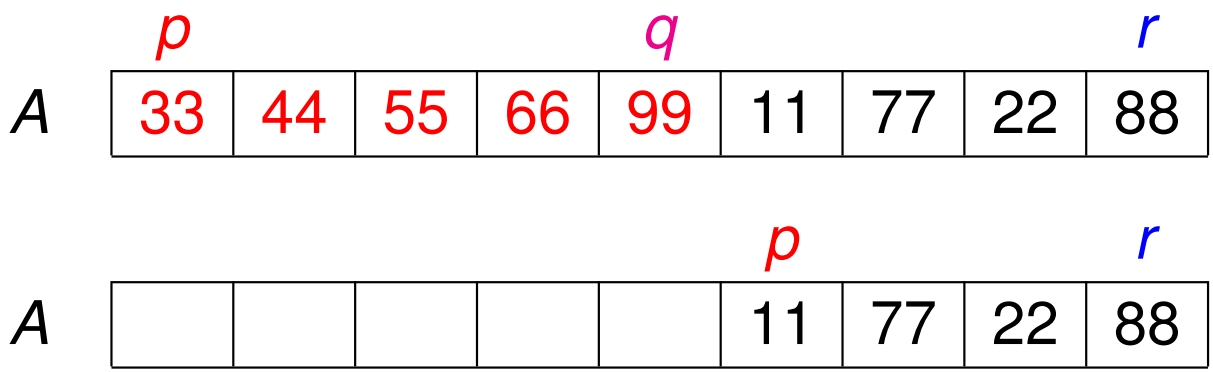
Mergesort



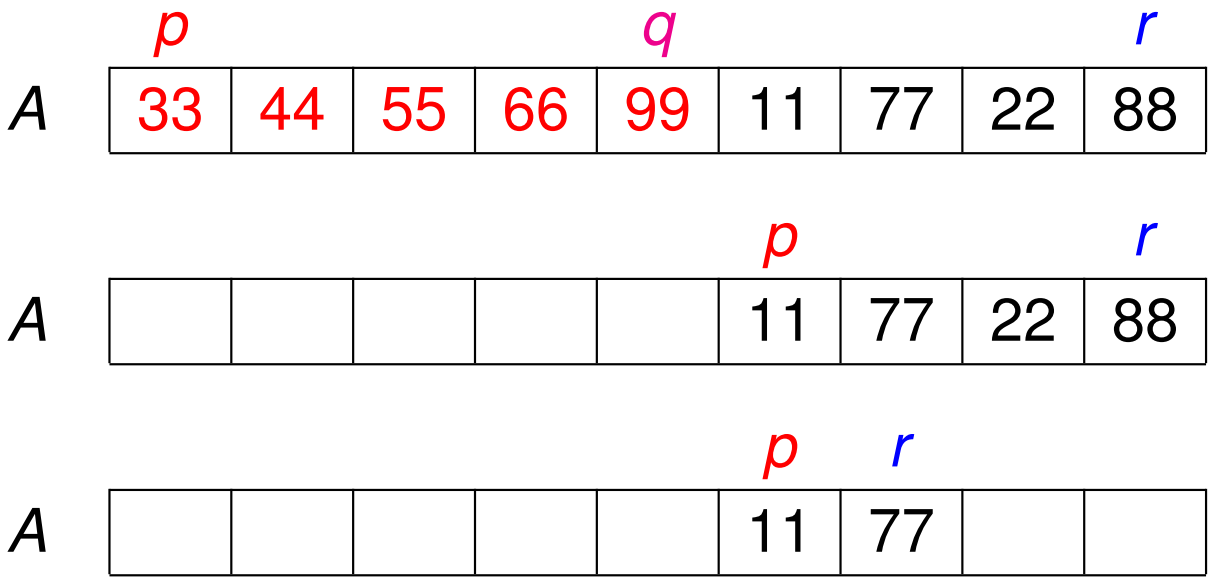
Mergesort



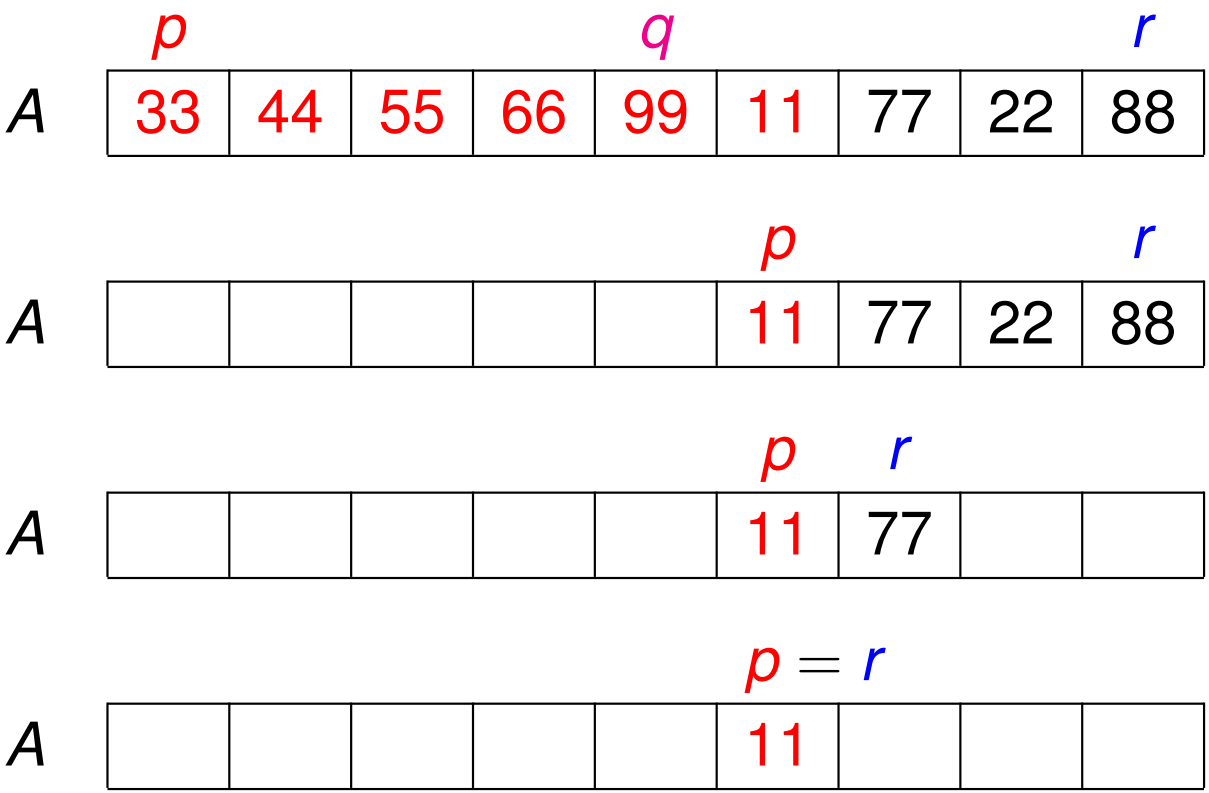
Mergesort



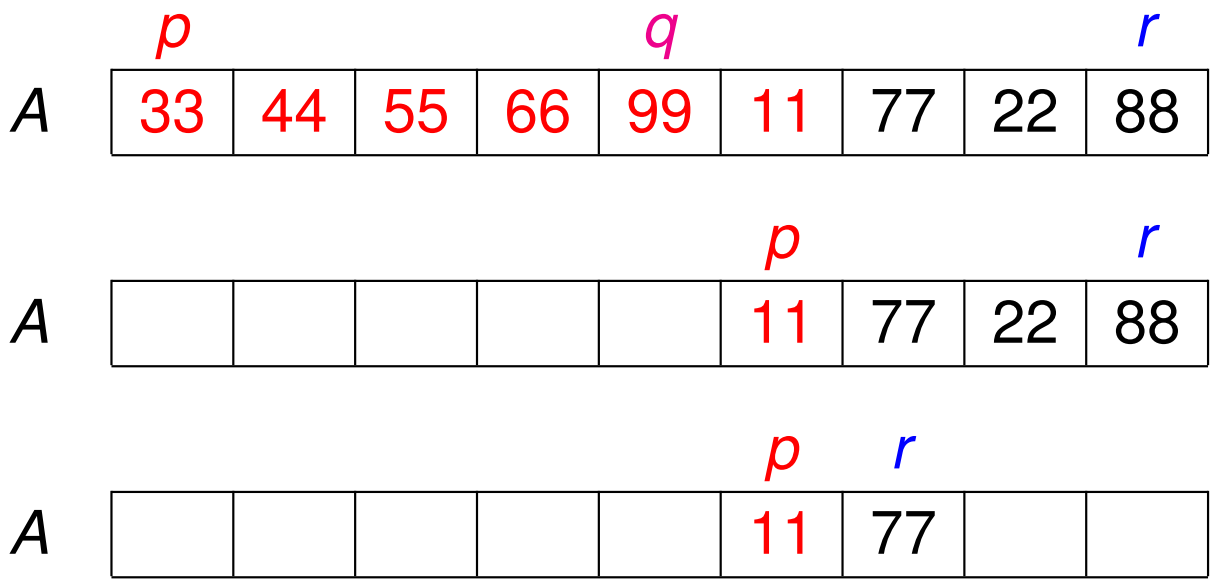
Mergesort



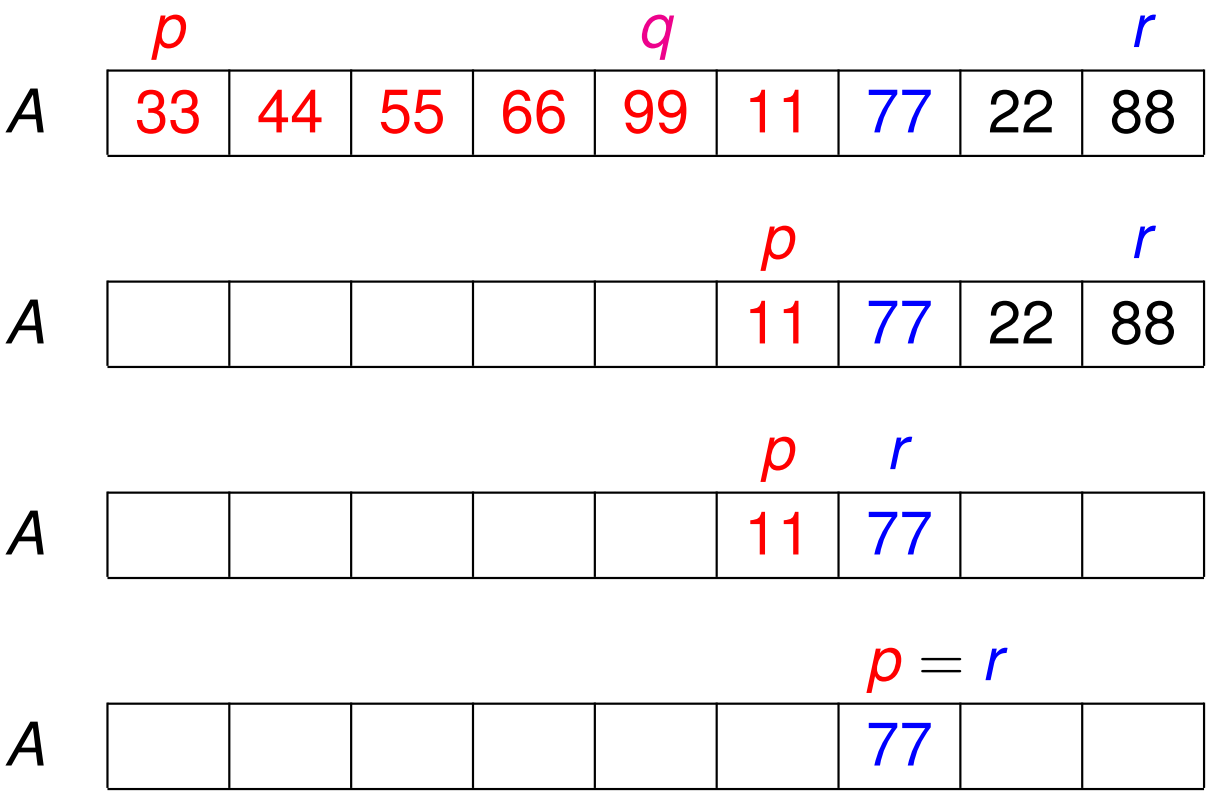
Mergesort



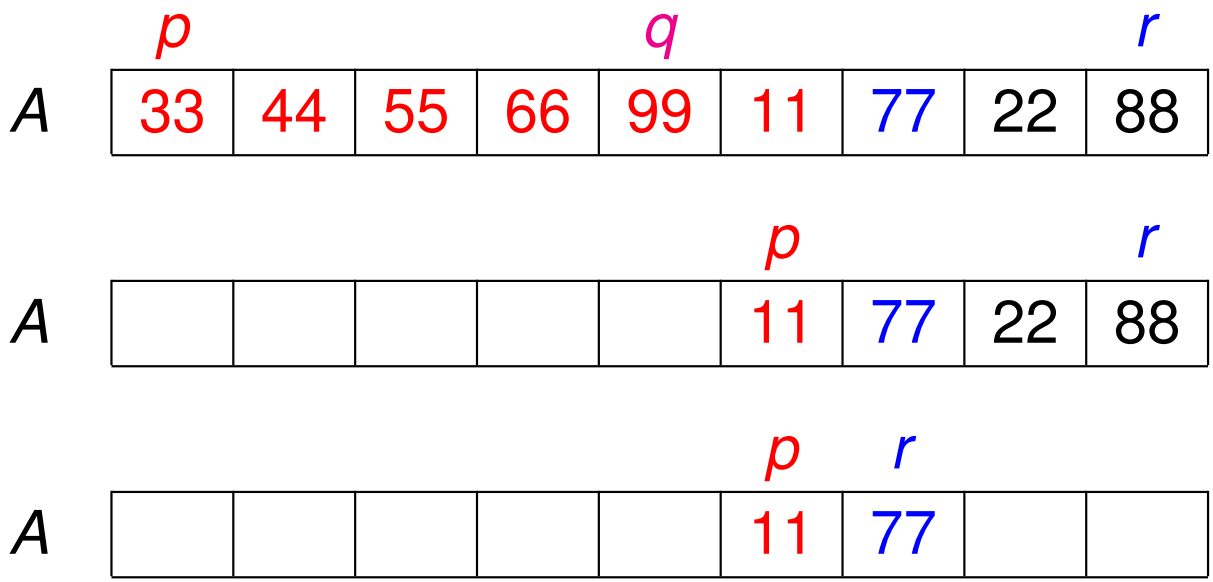
Mergesort



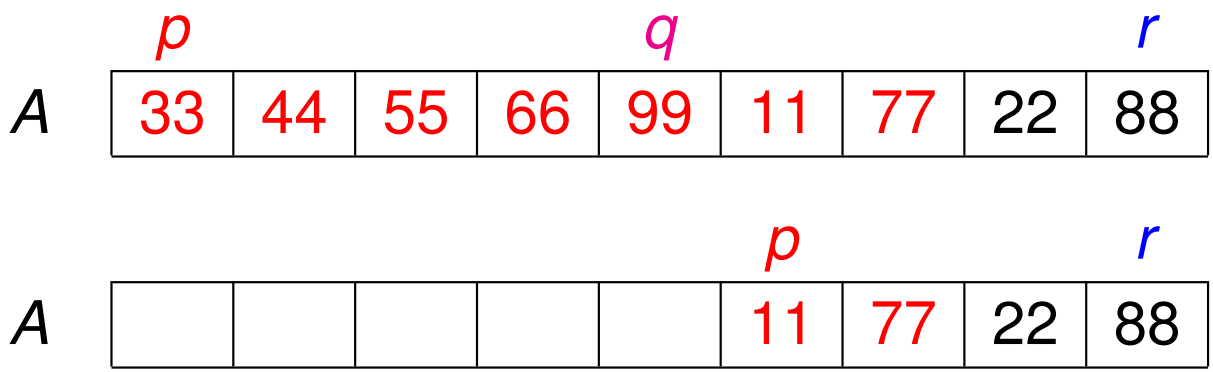
Mergesort



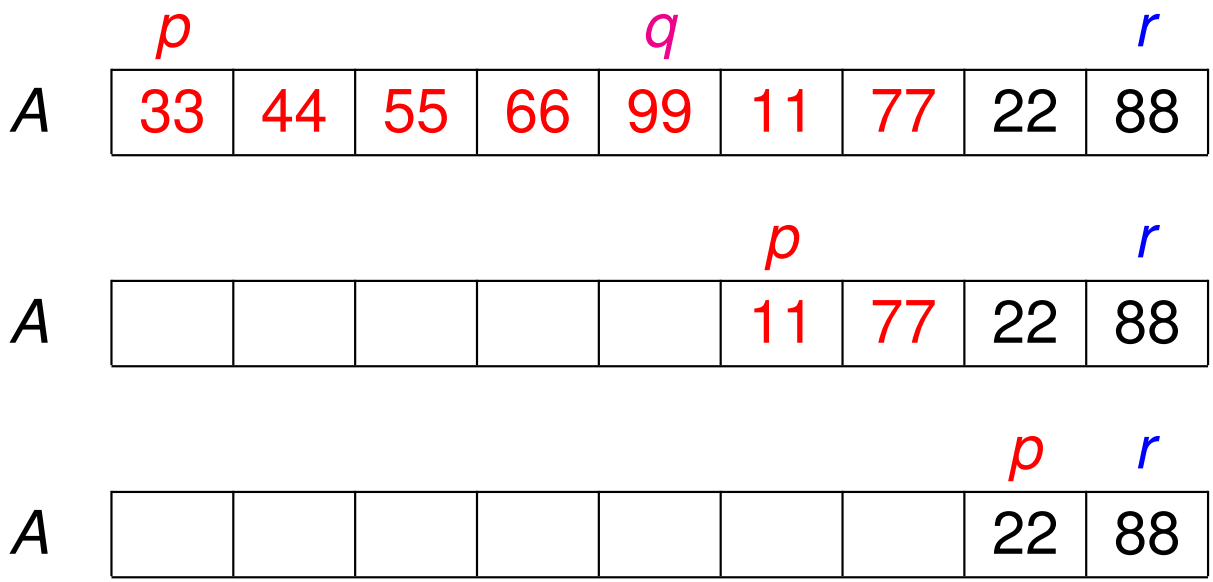
Mergesort



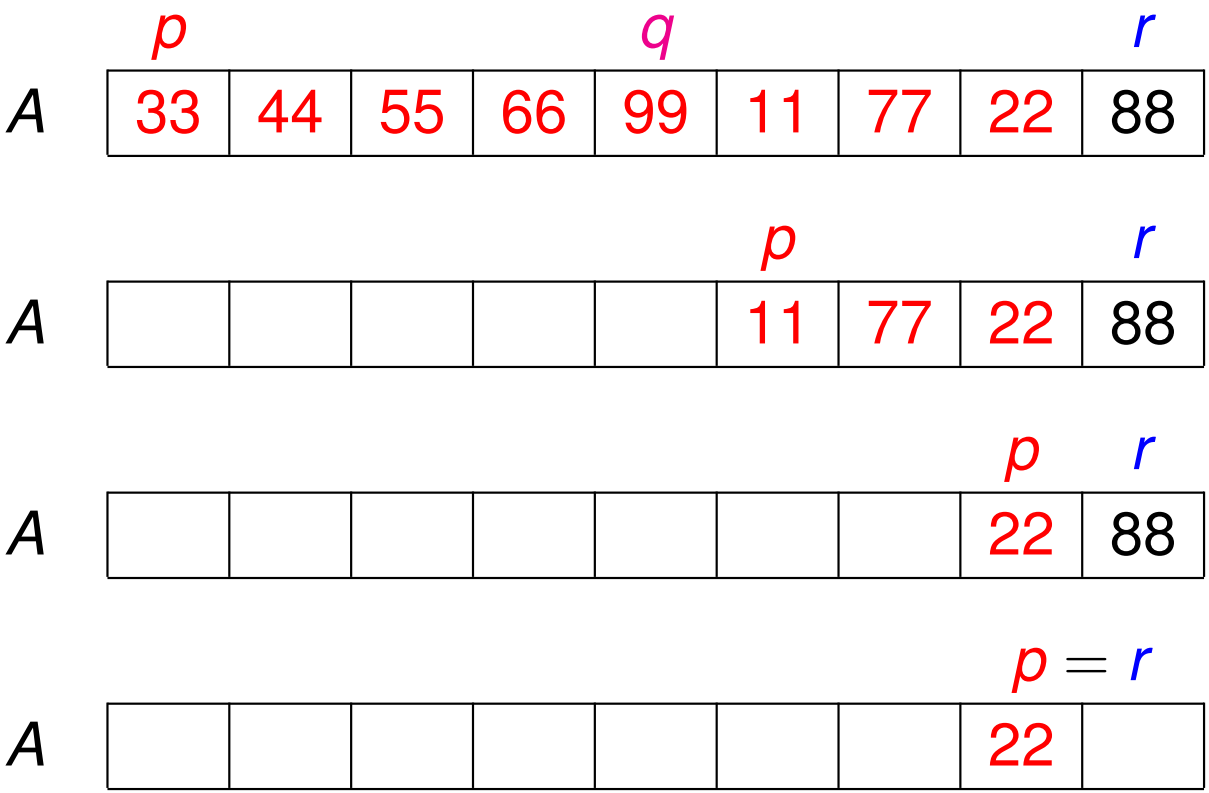
Mergesort



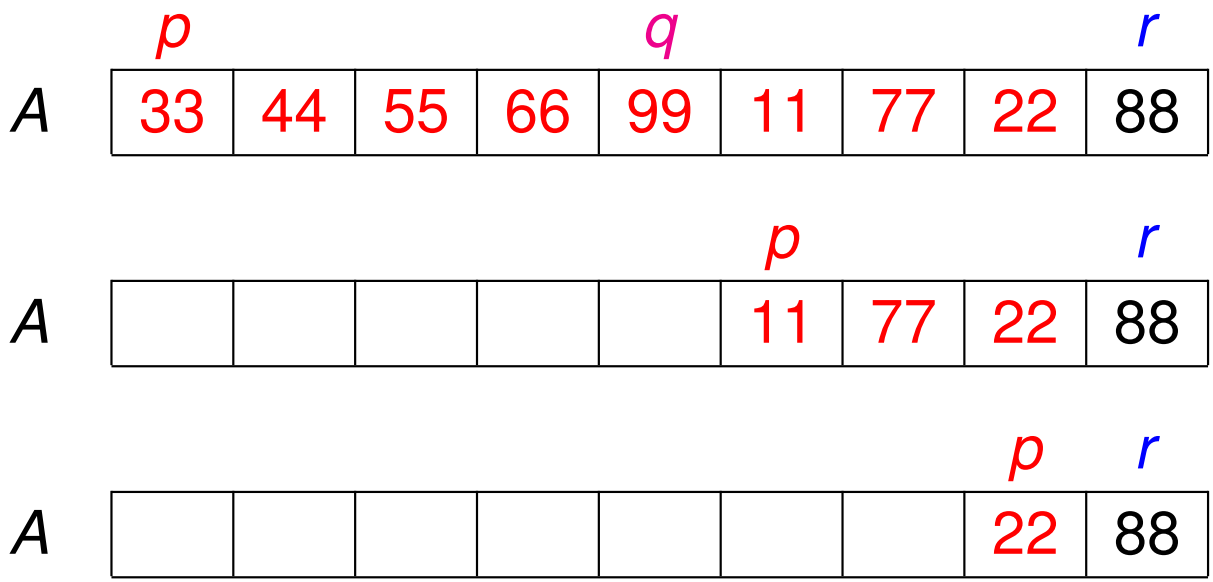
Mergesort



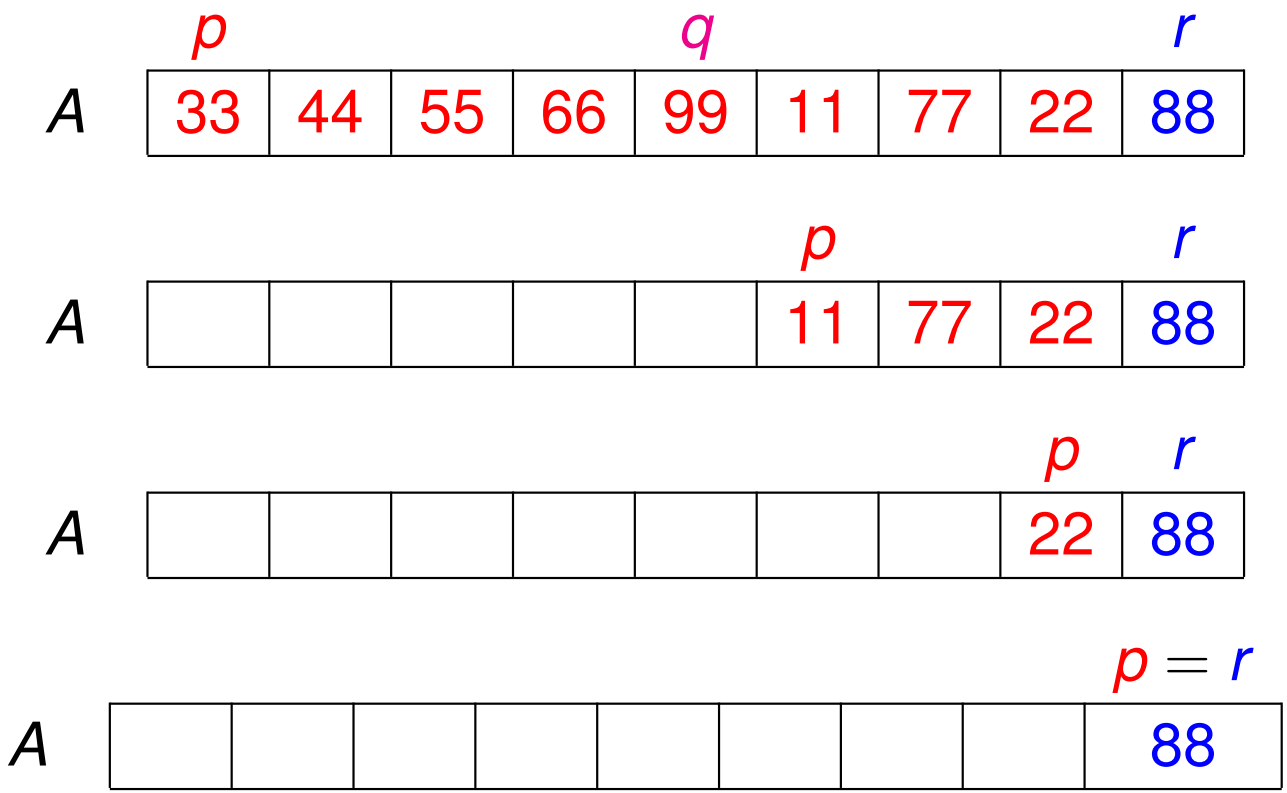
Mergesort



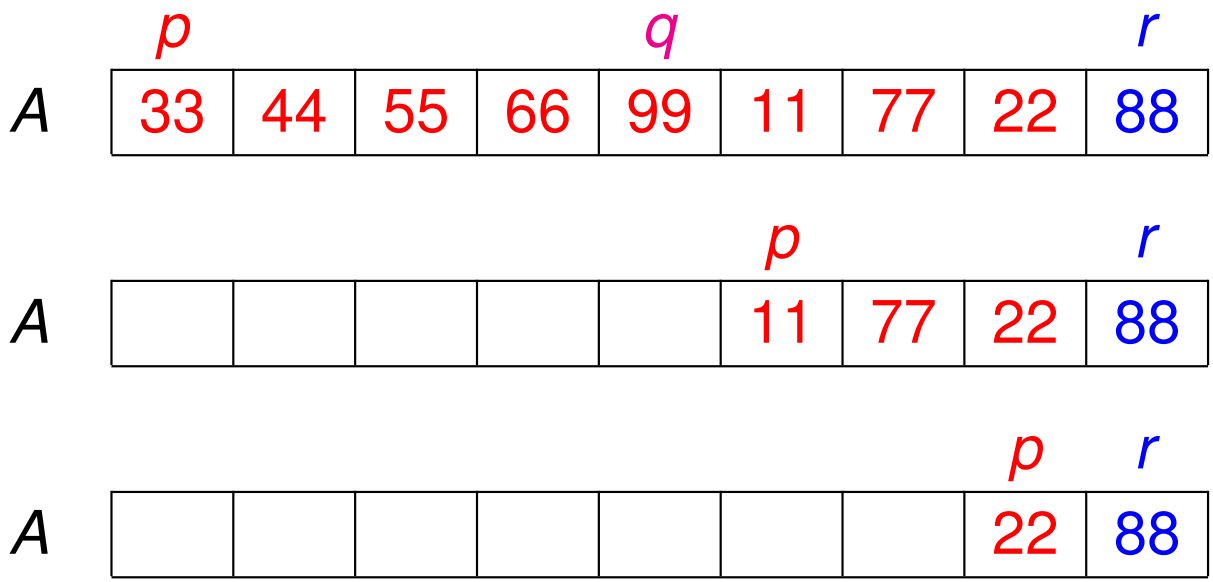
Mergesort



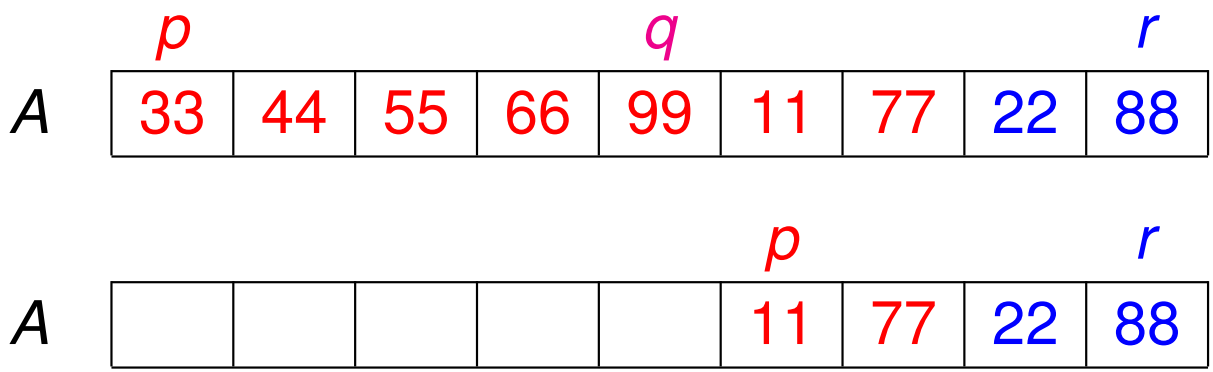
Mergesort



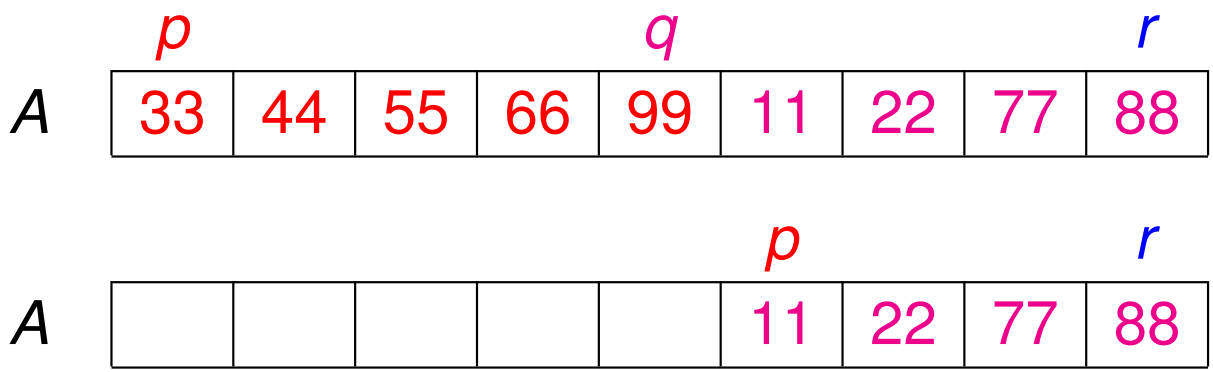
Mergesort



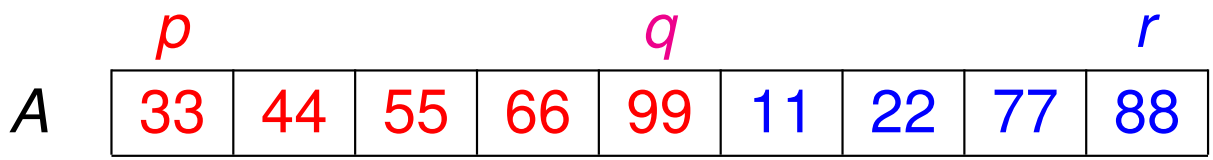
Mergesort



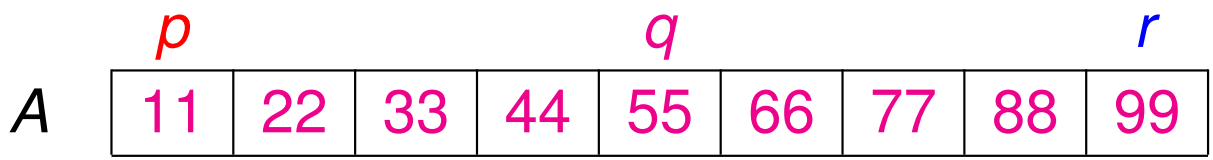
Mergesort



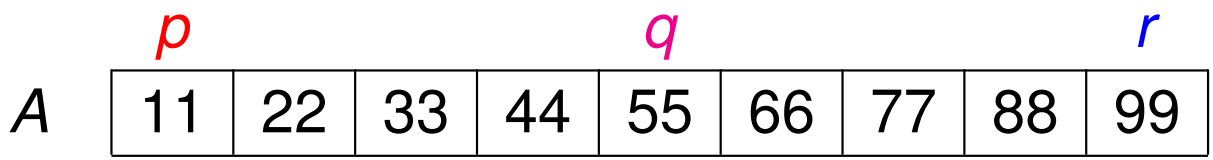
Mergesort



Mergesort



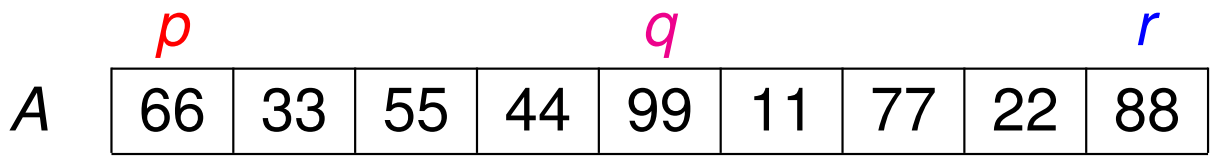
Mergesort



Mergesort

Relembrando: o objetivo é rearranjar $A[p \ldots r]$, com $p \leq r$, em ordem crescente.

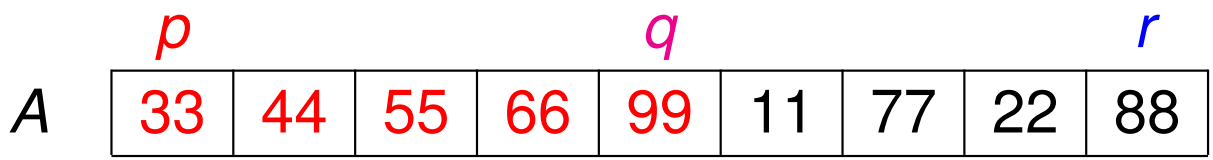
```
MERGESORT(A, p, r)
1   se p < r
2       então q ← ⌊(p + r)/2⌋
3             MERGESORT(A, p, q)
4             MERGESORT(A, q + 1, r)
5             INTERCALA(A, p, q, r)
```



Mergesort

Relembrando: o objetivo é rearranjar $A[p \ldots r]$, com $p \leq r$, em ordem crescente.

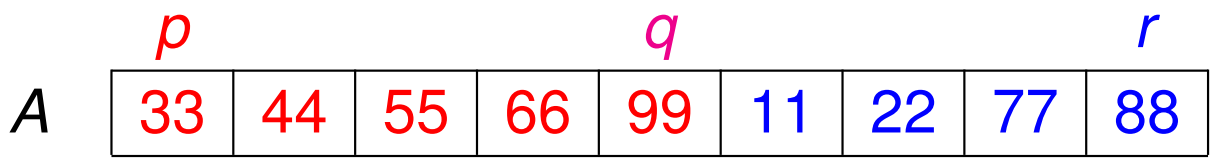
```
MERGESORT(A, p, r)
1   se p < r
2       então q ← ⌊(p + r)/2⌋
3             MERGESORT(A, p, q)
4             MERGESORT(A, q + 1, r)
5             INTERCALA(A, p, q, r)
```



Mergesort

Relembrando: o objetivo é rearranjar $A[p \ldots r]$, com $p \leq r$, em ordem crescente.

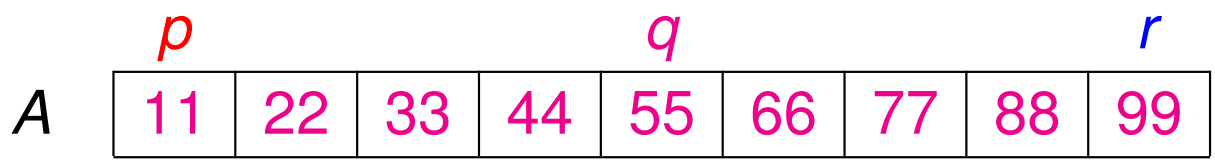
```
MERGESORT(A, p, r)
1   se p < r
2       então q ← ⌊(p + r)/2⌋
3             MERGESORT(A, p, q)
4             MERGESORT(A, q + 1, r)
5             INTERCALA(A, p, q, r)
```



Mergesort

Relembrando: o objetivo é rearranjar $A[p \ldots r]$, com $p \leq r$, em ordem crescente.

```
MERGESORT(A, p, r)
1   se p < r
2       então q ← ⌊(p + r)/2⌋
3             MERGESORT(A, p, q)
4             MERGESORT(A, q + 1, r)
5             INTERCALA(A, p, q, r)
```



Corretude do Mergesort

```
MERGESORT(A, p, r)
1   se p < r
2       então q ← ⌊(p + r)/2⌋
3             MERGESORT(A, p, q)
4             MERGESORT(A, q + 1, r)
5             INTERCALA(A, p, q, r)
```

O algoritmo está correto?

A corretude do algoritmo Mergesort apoia-se na corretude do algoritmo Intercala e segue facilmente **por indução** em $n := r - p + 1$.

Você consegue ver por quê? (Exercício)

Complexidade do Mergesort

```
MERGESORT(A, p, r)
1   se p < r
2       então q ← ⌊(p + r)/2⌋
3             MERGESORT(A, p, q)
4             MERGESORT(A, q + 1, r)
5             INTERCALA(A, p, q, r)
```

Qual é a complexidade de tempo do MERGESORT?

Seja $T(n) :=$ o consumo de tempo máximo (pior caso) em função de $n = r - p + 1$

Complexidade do Mergesort

```
MERGESORT(A, p, r)
1   se p < r
2       então q ← ⌊(p + r)/2⌋
3             MERGESORT(A, p, q)
4             MERGESORT(A, q + 1, r)
5             INTERCALA(A, p, q, r)
```

linha	consumo de tempo
1	?
2	?
3	?
4	?
5	?

$T(n) = ?$

Complexidade do Mergesort

```
MERGESORT(A, p, r)
1   se p < r
2       então q ← ⌊(p + r)/2⌋
3             MERGESORT(A, p, q)
4             MERGESORT(A, q + 1, r)
5             INTERCALA(A, p, q, r)
```

linha	consumo de tempo
1	$\Theta(1)$
2	$\Theta(1)$
3	$T(\lceil n/2 \rceil)$
4	$T(\lfloor n/2 \rfloor)$
5	$\Theta(n)$

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

Complexidade do Mergesort

Obtemos uma relação de recorrência (i.e., uma função definida em termos de si mesma).

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

Algoritmo Recursivo $\Rightarrow$ Relação de Recorrência

Em geral, algoritmos recursivos levam a complexidade $T(n)$ que é uma relação de recorrência.

- É necessário então resolver a recorrência!
- Ou seja, obter uma "fórmula não-recursiva" (ou "fórmula fechada") para $T(n)$.

No caso, $T(n) = \Theta(n \lg n)$. Assim, o consumo de tempo do Mergesort é $\Theta(n \lg n)$ (no pior ou melhor caso?).

Veremos em breve como resolver recorrências...