RI203
Uvod u računarske algoritme

dr.sc. Edin Pjanić

Pregled predavanja

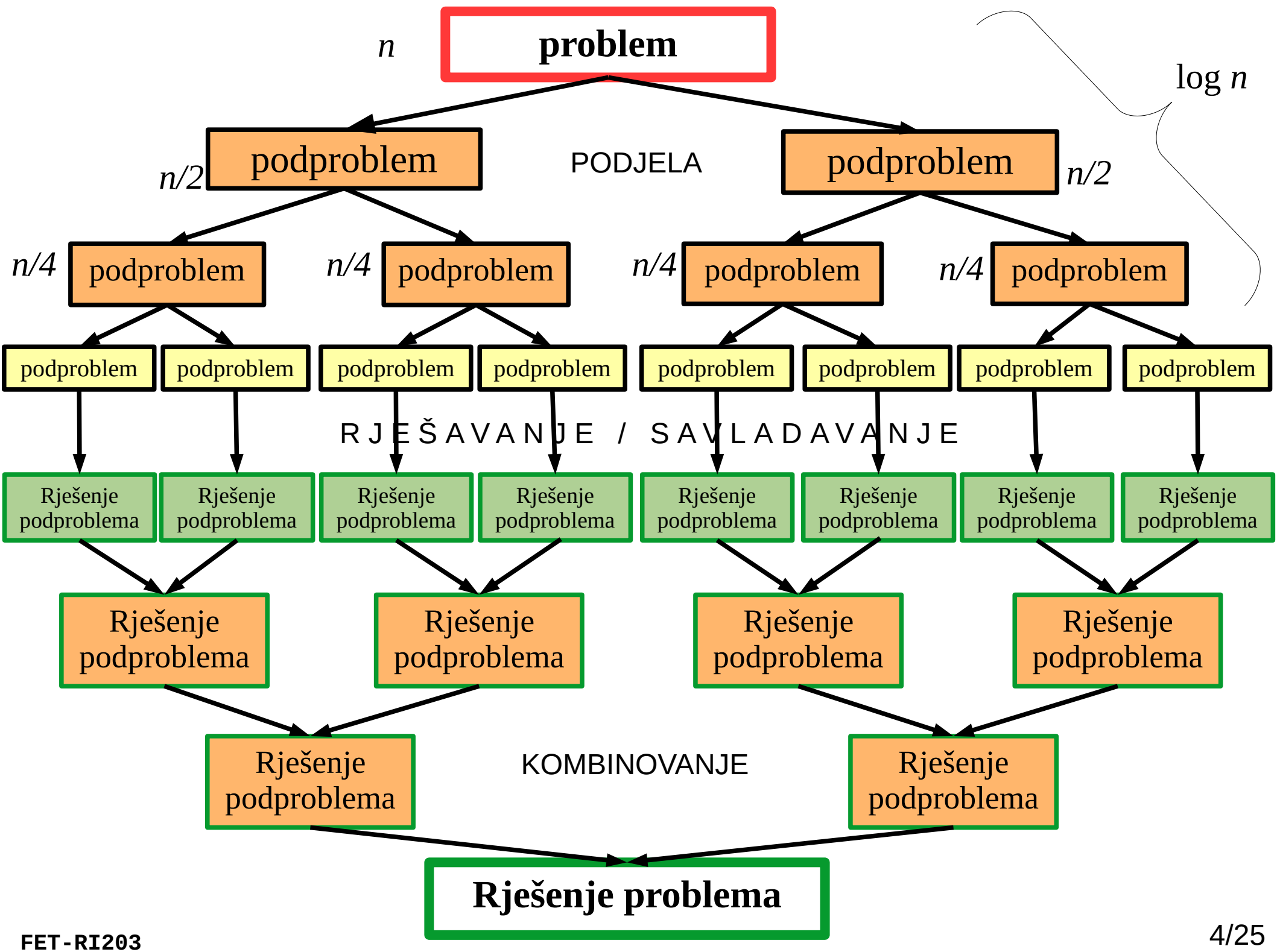
- Podijeli pa savladaj (“divide-and-conquer”) algoritmi
  - merge sort
  - quick sort
- Usporedba analiziranih metoda sortiranja
- Složenost “divide-and-conquer” algoritama

Podijeli pa savladaj – divide and conquer

Podijeli pa savladaj je poznati metod u dizajnu algoritama.

Metod ima tri koraka:

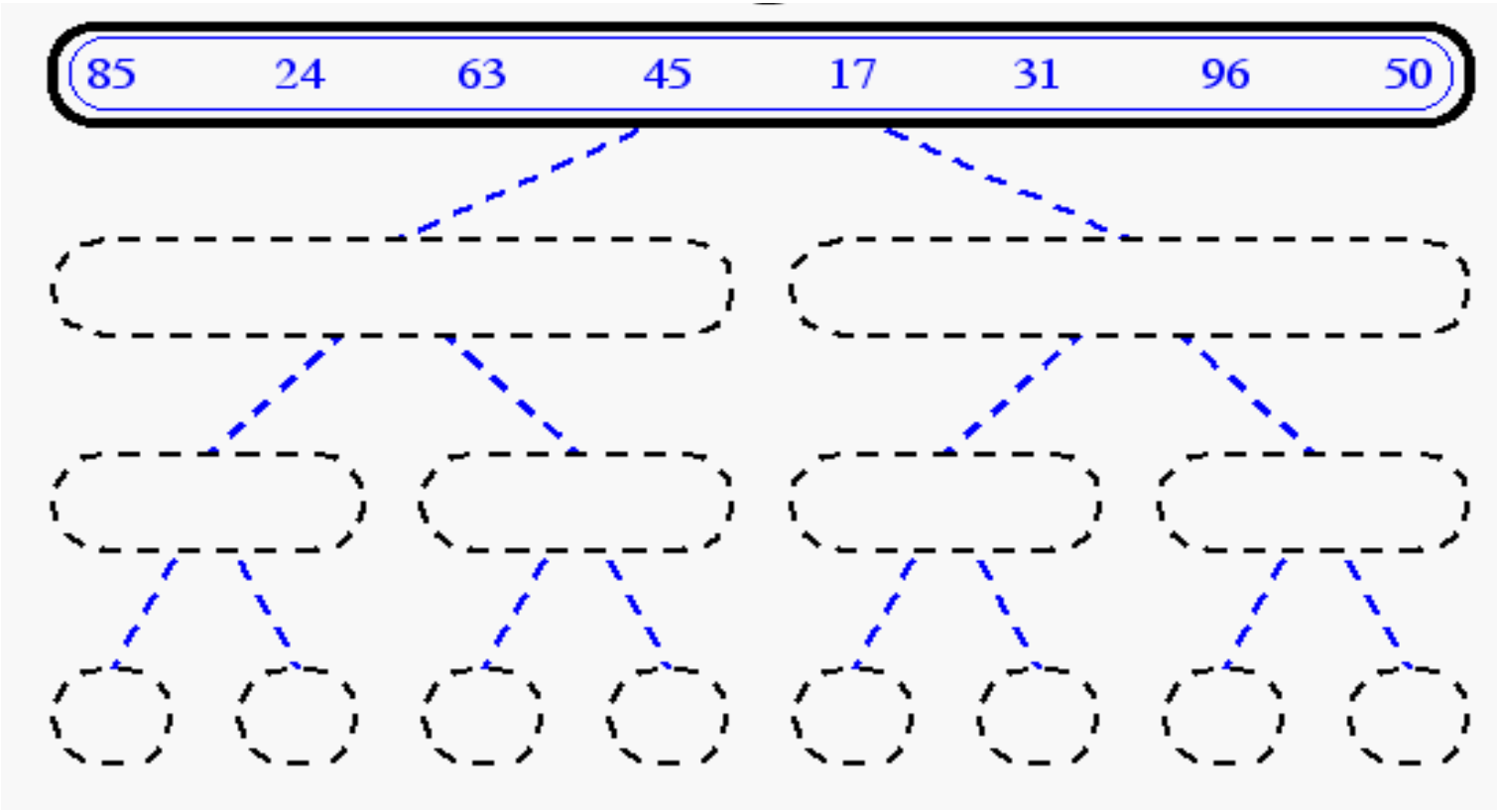
- **Podijeli**:
  - Ako imamo previše ulaznih podataka da bismo ih jednostavno obradili, podijeliti ih na dva ili više odvojenih podskupova.
- **Ponavljaj (podjelu)**:
  - Koristiti “podijeli pa savladaj” rekurzivno u cilju rješavanja podproblema na podskupovima.
- **Savladaj**:
  - Riješiti podproblem kad to bude moguće. Iskoristiti rješenja podproblema i spojiti ih u konačno rješenje originalnog problema.


n
problem
log n
n/2
podproblem
PODJELA
podproblem
n/2
n/4
podproblem
n/4
podproblem
n/4
podproblem
n/4
podproblem
podproblem
podproblem
podproblem
podproblem
podproblem
podproblem
podproblem
podproblem
RJEŠAVANJE / SAVLADAVANJE
Rješenje podproblema
Rješenje podproblema
Rješenje podproblema
Rješenje podproblema
Rješenje podproblema
Rješenje podproblema
Rješenje podproblema
Rješenje podproblema
Rješenje podproblema
Rješenje podproblema
Rješenje podproblema
Rješenje podproblema
Rješenje podproblema
KOMBINOVANJE
Rješenje podproblema
Rješenje problema

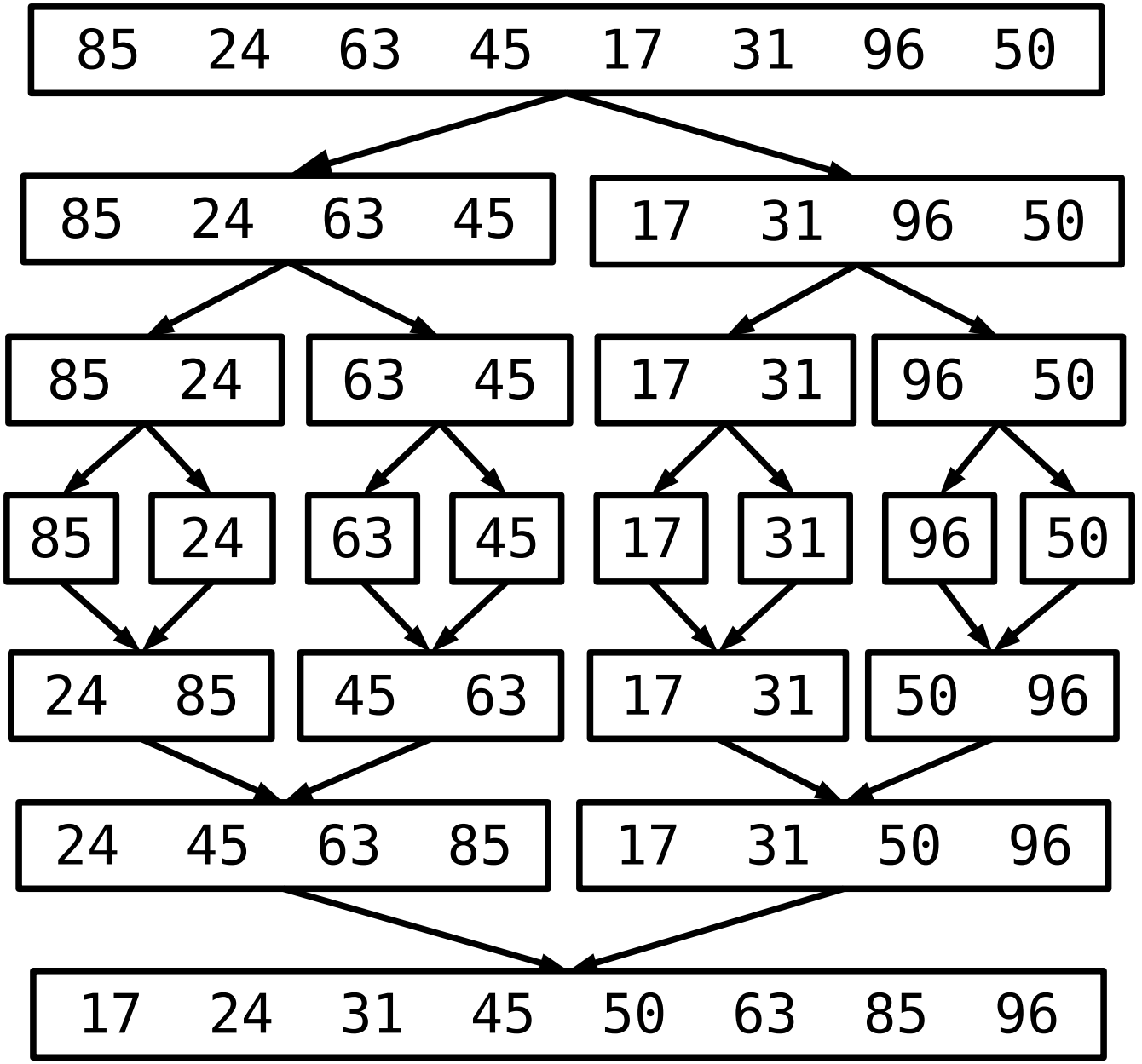
Merge sort – sortiranje spajanjem

- Baziran na principu "podijeli pa savladaj".
- Autor je John von Neumann, 1945.
- Princip je posve drugačiji od principa prethodnih metoda koji su se bazirali na poređenju.
- **Podijeli**:
  - nesortirani niz podijeli se na dva podniza jednake veličine.
- **Ponavljaj**:
  - svaki podniz na isti način se dijeli rekurzivno istom metodom, dok se ne dobije niz od 1 elementa - koji je sortiran.
- **Savladaj**:
  - dva sortirana podniza se spoje u treći niz tako da i on bude sortiran.

Merge sort



Merge sort


85 24 63 45 17 31 96 50
85 24 63 45
17 31 96 50
85 24
63 45
17 31
96 50
85
24
63
45
17
31
96
50
24 85
45 63
17 31
50 96
24 45 63 85
17 31 50 96
17 24 31 45 50 63 85 96

Merge sort – analiza

- Ako je ulazni niz već sortiran ovaj metod to ne prepoznaje (uvijek se isto ponaša). Postoje varijante koje prepoznaju sortirane dijelove niza.
- Koristi dodatnu memoriju pri spajanju podnizova.
- Uglavnom se koristi za sortiranje većih fajlova u vanjskoj memoriji pri čemu se koristi vanjska memorija kao pomoćna pri spajanju.
- Ovo je stabilan algoritam sortiranja.

- Složenost:
  - Za procesiranje jednog elementa: O(1). Kako na svakom nivou imamo n elemenata, to je O(n) za jedan nivo.
  - Nivoa imamo $log_2 \, n$, odnosno $lg \, n$.
  - Otuda je složenost $\boldsymbol{O(n \log n)}$.

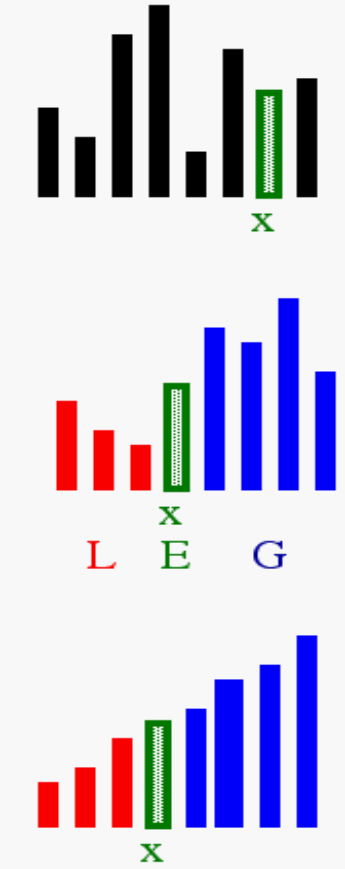
Quicksort

- Do sada najbrži algoritam za sortiranje O(n *log* n). U najgorem slučaju je O(n^2) ali se to vrlo rijetko javlja.
- Baziran na rekurzivnom "podijeli pa savladaj".
- Autor je Tony Hoare.
- Quicksort je poznat i kao "partitioning-exchange sort" – sortiranje particioniranjem (dijeljenjem) i razmjenom.

Ideja ovog metoda je da se u svakom koraku jedan element (tzv. pivot) postavi na svoje mjesto.

- svi elementi manji od tog elementa idu lijevo
- svi elementi veći od tog elementa idu desno

Postupak se rekurzivno ponovi za lijevi dio niza i za desni dio niza od pivota.



Quicksort - metodologija

- **Podijeli**:
  - Ako niz ima 2 ili više elemenata odabrati bilo koji element za tzv. **pivot** (E) te podijeliti niz na tri dijela (**particionisanje**):
    - L dio: dio sa elementima manjim od pivota **E**
    - pivot **E**
    - G dio: dio sa elementima većim od pivota **E**
- **Ponavljaj**:
  - rekurzivno sortirati L i G sve dok imaju više od 1 elementa
- **Savladaj**:
  - spojiti podnizove redom: {L, E, G}

Particionisanje

Quick sort

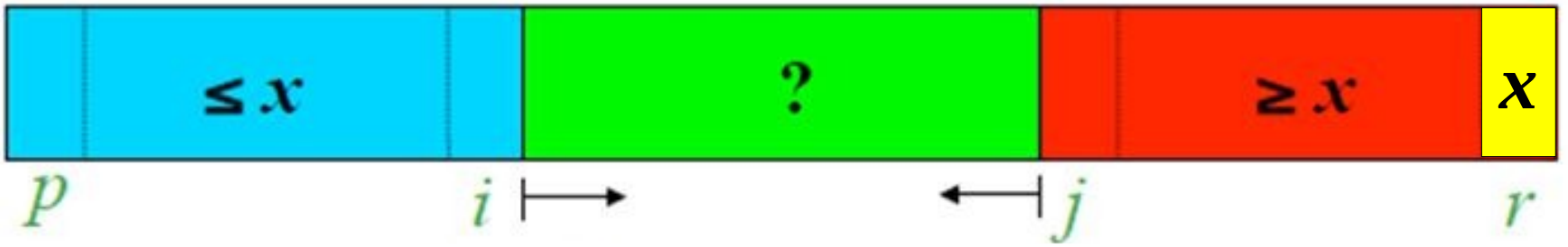
- Raspodjela elemenata niza (liste) na dva dijela tako da su elementi u prvom (lijevom) dijelu manji od elemenata u drugom (desnom) dijelu.

Generalno

- Raspodjela elemenata niza (liste) na dva dijela tako da elementi u prvom (lijevom) dijelu zadovoljavaju neki kriterij a elementi u drugom (desnom) dijelu ne zadovoljavaju taj kriterij.

Particioniranje: quick sort

- Hoare algoritam (Tony Hoare)



- Lomuto algoritam (Nico Lomuto)

Particioniranje: generalno

```
template <typename It, typename Pred>
  It partition(It first, It last, Pred pred);
```


```
i=first; j=last

...

while(pred(*i)) ++i;      // while(*i < pivot) ++i;

while(!pred(*j)) --j;     // while(pivot < *i) --j;

...
```

Quick sort algoritam - pseudokod

```
QuickSort(first, last)
  if first < last
      pivotPos = partition(first, last)
      QuickSort(first, pivotPos)
      QuickSort(pivotPos+1, last)
```

Quicksort - primjer

18 32 12 5 38 33 16 2 **25**

18 **32** 12 5 38 33 16 **2** **25**

18 2 12 5 **38** 33 **16** 32 **25**

18 2 12 5 16 **33** 38 32 **25**

18 2 12 5 16 25 38 32 33

18 2 12 5 **16** 38 32 **33**

18 2 12 **5** **16** **38** **32** **33**

5 2 12 18 **16** 32 38 **33**

5 2 12 **18** **16** 32 **38** **33**

5 2 12 16 18 32 33 38

5 2 **12**

5 2 12

5 **2**

2 5

2 5 12 16 18 25 32 33 38

Quicksort - analiza

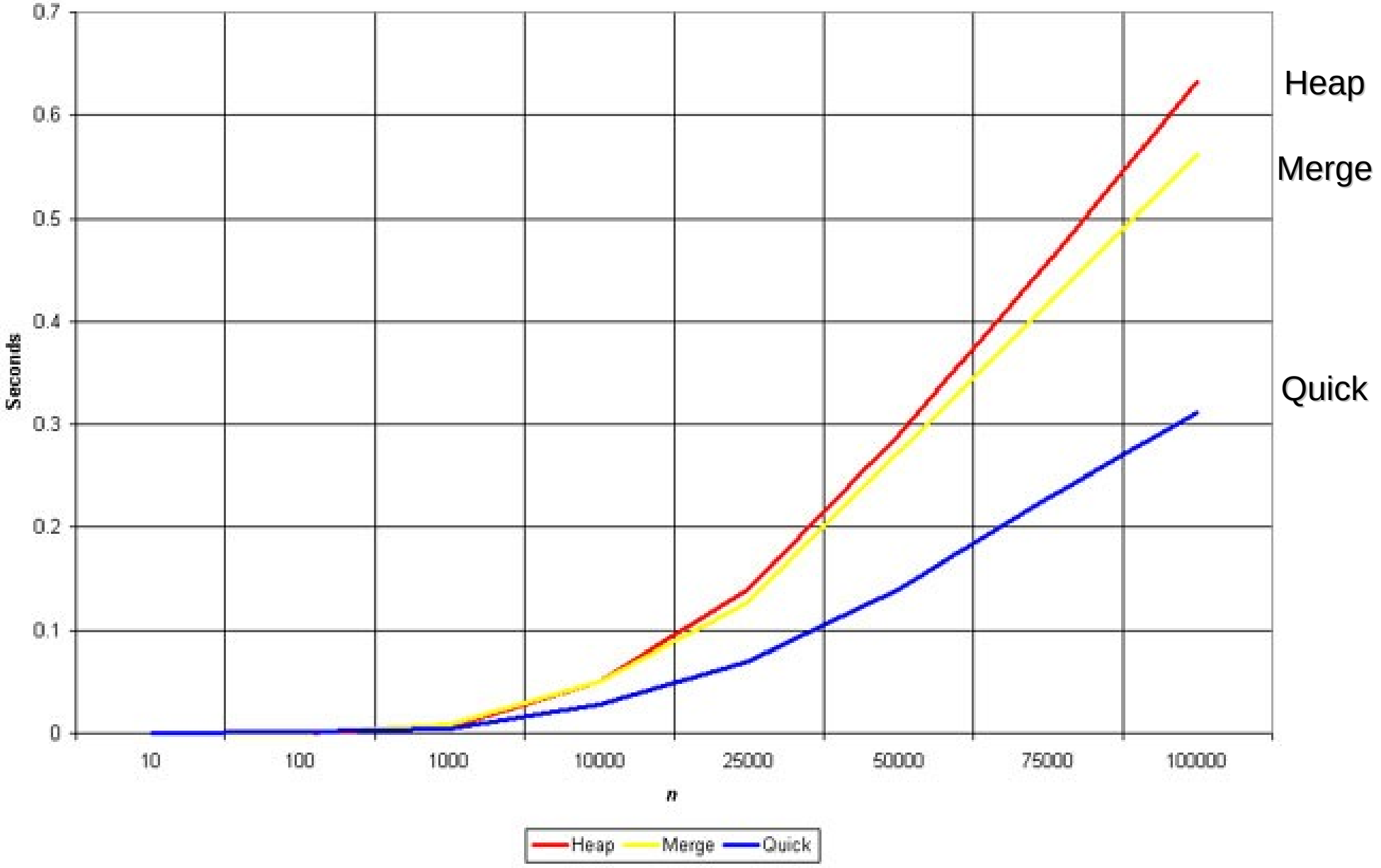
- Za pivot se može odabrati bilo koji element, mada postoje određene strategije pri kojim se nekad dobijaju bolji rezultati.
- U različitim izvedbama se različito tretiraju i elementi koji su jednaki pivot elementu.
- Unutrašnja petlja za particionisanje je krajnje jednostavna i brza, što je prednost ovog algoritma.
- Složenost: u najboljem slučaju se niz dijeli na dva jednaka dijela. U tom slučaju imamo pronalazak tačnog mjesta pivota – O(n) i dva dijela koja moramo rekurzivno sortirati:

$$T(n) = O(n) + T(n/2) + T(n/2)$$
$$\Rightarrow O(n \lg n)$$

- Najgori slučaj nastupa ako je pivot u svakom koraku najveći ili najmanji element:

$$\Rightarrow O(n^2)$$

Usporedba metoda sortiranja koji su O(n *log* n)



Pregled analiziranih metoda sortiranja

Naziv	Najbolje (bc)	Prosječno (ac)	Najgore (wc)	Stabilan	Metoda
selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	ne	biranje
insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$	da	umetanje
bubble sort	$O(n)$	$O(n^2)$	$O(n^2)$	da	zamjena
shell sort	-	-	$O(n^{3/2})$	ne	umetanje
merge sort	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	da	spajanje
quick sort	$O(n\log n)$	$O(n\log n)$	$O(n^2)$	ne	podjela

“Podijeli-pa-savladaj” algoritmi (divide-and-conquer)

- Najpoznatija i najpopularnija strategija u dizajnu algoritama.
- **Podijeli** (divide)
  - Problem veličine n razbije se (podijeli) na određeni broj manjih pod-problema.
- **Savladaj** (conquer)
  - Po istom principu se rekurzivno rješavaju pod-problemi (razbijaju na manje), dok se ne dođe do nivoa kad se rješenje može naći direktno.
- **Kombinuj** (combine)
  - Dobijena rješenja svih pod-problema se spoje u konačno rješenje originalnog problema.

“Podijeli-pa-savladaj” algoritmi 2

- Neka se problem rješava tako da se originalni problem veličine ***n*** podijeli na ***a*** problema koji su veličine ***n/b***.
- Rješavanje ***a*** takvih pod-problema bi zahtijevalo:

$$T_1(n/b) + T_2(n/b) + T_3(n/b) + \ldots + T_a(n/b)$$

- Rješavanje originalnog problema, osim rješavanja ***a*** navedenih pod-problema veličine ***n/b***, zahtijeva još i spajanje rješenja pod-problema u konačno rješenje čija veličina zavisi od veličine originalnog problema. Neka je ta zavisnost data funkcijom ***f(n)***.
- Vrijeme potrebno za rješavanje originalnog problema bismo sada mogli napisati kao:

$$T(n) = T_1(n/b) + T_2(n/b) + T_3(n/b) + \ldots + T_a(n/b) + f(n)$$

tj. $T(n) = a\,T(n/b) + f(n)$

Master teorema – uprošteno

- Teorema za izračunavanje asimptotske složenosti rekurzivnog divide-and-conquer algoritma.
- Odnosi se na algoritme čije se vrijeme izvršavanja (broj operacija, memorija i sl.) može opisati formulom:

$$T(n) = a\,T(n/b) + \Theta(n^c)$$
$$T(1) = \Theta(1)$$

gdje su $a \geq 1$, $b > 1$, $c > 0$ konstante.

$T(n)$ mora biti monotona funkcija, a funkcija $f(n)$, sa prethodnog slajda, mora biti polinomskog oblika.

Master teorema – uprošteno

$$T(n) = a\,T(n/b) + \Theta(n^c)$$
$$T(1) = \Theta(1)$$

1. slučaj: ako je $a < b^c$, odnosno $\log_b a < c$, tada je

$$T(n) = \Theta(n^c)$$

2. slučaj: ako je $a = b^c$, odnosno $\log_b a = c$, tada je

$$T(n) = \Theta(n^c \log n)$$

3. slučaj: ako je $a > b^c$, odnosno $\log_b a > c$, tada je

$$T(n) = \Theta(n^{\log_b a})$$

Primjer 1

$$T(n) = 3T(n/2) + 4n^2 + 3n$$

$$f(n) = 4n^2 + 3n \Rightarrow O(n^2)$$

$$\Rightarrow a = 3$$

$$b = 2$$

$$c = 2$$

Pošto je $3 < 2^2$ primjenjuje se prvi slučaj, pa je:

$$T(n) = \Theta(n^2)$$

1. slučaj: ako je $a < b^c$, odnosno $\log_b a < c$, tada je

$T(n) = \Theta(n^c)$

2. slučaj: ako je $a = b^c$, odnosno $\log_b a = c$, tada je

$T(n) = \Theta(n^c \log n)$

3. slučaj: ako je $a > b^c$, odnosno $\log_b a > c$, tada je

$T(n) = \Theta(n^{\log_b a})$

Primjer 2: Quick sort

$$T(n) = 2T(n/2) + \Theta(n)$$

Imamo particionisanje, koje je linearno ovisno o ***n***, i onda rekurzivno quick sort dvije polovice niza.

$$\Rightarrow a = 2$$
$$b = 2$$
$$c = 1$$

Pošto je $2 = 2^1$ primjenjuje se drugi slučaj, pa je:

$$T(n) = \Theta(n^c \log n)$$

tj., $T(n) = \Theta(n \log n)$

1. slučaj: ako je $a < b^c$, odnosno $\log_b a < c$, tada je
$T(n) = \Theta(n^c)$

2. slučaj: ako je $a = b^c$, odnosno $\log_b a = c$, tada je
$T(n) = \Theta(n^c \log n)$

3. slučaj: ako je $a > b^c$, odnosno $\log_b a > c$, tada je
$T(n) = \Theta(n^{\log_b a})$

Primjer: najveći podniz (maximum subarray)

- Za dati niz, potrebno je pronaći dio niza za koji je suma elemenata najveća. Taj dio niza nazivamo *najveći podniz*.
- Npr. za niz A[0 ... 12]:
  - A: {5, -4, 7, 2, -12, 6, -2, 5, 6, -1, -13, 7, 2}

Najveći podniz je A[5 ... 8] a suma iznosi 15.

Rješenje:

- Brute-force: $\Theta(n^2)$
- Divide and conquer:
  - Master teorema => $\Theta(n \log n)$