Алгоритми сортирања: анализа и поређење

Михајло Јанићијевић

март 2026.

Садржај

1 Увод

Сортирање представља један од **фундаменталних проблема** у рачунарству. Готово свака значајнија апликација — од претраживача до база података — ослања се на ефикасне алгоритме за уређивање података. *Циљ овог рада* је да прикаже и упореди неколико класичних алгоритама сортирања, анализира њихову временску и просторну сложеност, те истакне ситуације у којима је сваки од њих оптималан за примену.

У пракси се сортирање најчешће своди на уређивање низа од n елемената тако да важи:

$$a_1 \leq a_2 \leq \cdots \leq a_n$$

Ефикасност алгоритма мери се бројем *поређења* и *размена* елемената у зависности од величине улаза.

2 Основни појмови

2.1 Дефиниције

Дефиниција 2.1 (Алгоритам сортирања)**. Алгоритам сортирања** је поступак који, за дати низ елемената из уређеног скупа, производи перестановку тог низа у нерастућем или неопадајућем редоследу.

Дефиниција 2.2 (Стабилно сортирање)**.** Алгоритам сортирања је **стабилан** ако елементи са истим кључем задржавају свој међусобни редослед из оригиналног низа.

2.2 Временска сложеност

Лема 2.1. *Сваки алгоритам сортирања заснован искључиво на поређењима захтева у најгорем случају најмање* $\lceil \log_2(n!) \rceil$ *поређења.*

Теорема 2.1 (Доња граница сортирања поређењима)**.** *Временска сложеност у најгорем случају за сваки алгоритам сортирања заснован на поређењима износи* $\Omega(n \log n)$.

Из Стирлингове апроксимације следи:

$$\log_2(n!) \approx n \log_2 n - n \log_2 e = \Theta(n \log n)$$

3 Алгоритми и анализа

3.1 Merge сорт

Merge сорт (енгл. *Merge Sort*) је алгоритам заснован на принципу **завади па владај**. Рекурентна релација за временску сложеност је:

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$$

Применом Мастер теореме добијамо $T(n) = \Theta(n \log n)$.

3.2 Квик сорт

Квик сорт у просечном случају постиже сложеност $\Theta(n \log n)$, а у најгорем $O(n^2)$. Просечна вредност броја поређења износи:

$$C_{\text{avg}}(n) = 2(n+1)H_n - 4n \approx 2n \ln n$$

где је $H_n = \sum_{k=1}^{n} \frac{1}{k}$ n-ти хармонијски број.

3.3 Хип сорт

Хип сорт гарантује $O(n \log n)$ у свим случајевима и ради *in-place*, са просторном сложеношћу $O(1)$.

4 Поређење алгоритама

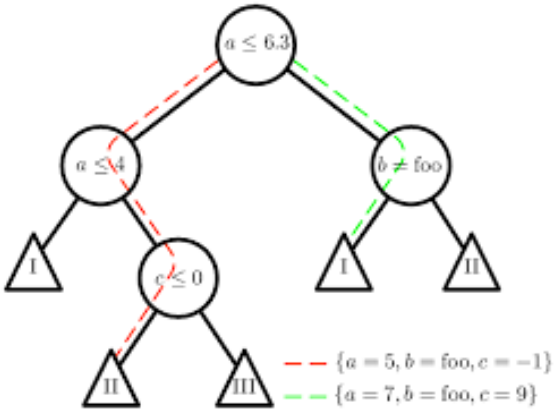
4.1 Табела сложености

У табели 1 приказана је упоредна анализа разматраних алгоритама.

Табела 1: Поређење алгоритама сортирања

Алгоритам	**Најбољи**	**Просечни**	**Најгори**	**Стабилан**
Мерге сорт	$\Omega(n \log n)$	$\Theta(n \log n)$	$O(n \log n)$	Да
Куик сорт	$\Omega(n \log n)$	$\Theta(n \log n)$	$O(n^2)$	Не
Хип сорт	$\Omega(n \log n)$	$\Theta(n \log n)$	$O(n \log n)$	Не
Bubble	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	Да
Уметање	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	Да

4.2 Илустрација — стабло одлука



Стабло одлука на слици илуструје све могуће путање извршавања алгоритма сортирања заснованог на поређењима за низ од три елемента. Сваки лист одговара јединственој пермутацији, а **висина стабла** директно одређује број поређења у најгорем случају.

За $n = 3$ постоји $3! = 6$ листова, па је потребно најмање $\lceil \log_2 6 \rceil = 3$ поређења. Ово *минималистичко стабло* чини основу доказа доње границе $\Omega(n \log n)$.

5 Практичне смернице

5.1 Када користити који алгоритам?

- **Merge сорт** — предност када је стабилност обавезна и меморија није ограничена.
- **Квик сорт** — у пракси најбржи за насумичне податке; употребити са случајним пивотом.
- **Хип сорт** — гарантована сложеност без додатне меморије.

5.2 Специјализовани алгоритми

За посебне улазе постоје алгоритми **линеарне** сложености:

1. *Counting sort* — погодан када је опсег вредности мали ($k = O(n)$), сложеност $O(n + k)$.
2. *Radix sort* — ради цифру по цифру; сложеност $O(d \cdot (n + b))$.
3. *Bucket sort* — ако су подаци равномерно распоређени; очекивано $O(n)$.

5.3 Правила доброг одабира

Мали низови ($n < 20$)

Употребити *Insertion sort* — мали константни фактор надмашује асимптотску предност.

Велики, насумични низови

Introsort (комбинација квик, хип и уметање) — стандардна имплементација у `std::sort`.

Готово сортирани низови

Timsort (Python, Java) — искоришћава постојеће уређене подсеквенце.

6 Закључак

У раду смо представили и анализирали пет класичних алгоритама сортирања. **Кључни закључак** је да не постоји универзално „најбољи"алгоритам — одабир зависи од величине улаза, расподеле података и ограничења меморије. За опште намене, *хибридни алгоритми* попут *Introsort*-а нуде оптималан баланс. Разумевање доње границе $\Omega(n \log n)$ подсећа нас да ниједан алгоритам заснован на поређењима не може бити асимптотски бржи у општем случају.