Поређење временске сложености алгоритама сортирања

Јован Стефановић

31. јануар 2026.

Садржај

1 Увод **2**

2 Основни појмови **2**

3 Алгоритми квадратне сложености **2**
- 3.1 Сортирање балончићима 2
- 3.2 Сортирање одабиром 2
- 3.3 Сортирање уметањем 3

4 Алгоритми логаритамске сложености **3**
- 4.1 Сортирање спајањем 3
- 4.2 Брзо сортирање 3

5 Теоријска доња граница **4**

6 Табеларно поређење **4**

7 Илустрација раста сложености **4**

8 Закључак **4**

1 Увод

Иако више алгоритама може успешно сортирати податке, њихова ефикасност може значајно да се разликује. Због тога је временска сложеност један од **најважнијих критеријума** при избору алгоритма сортирања.

У овом раду анализира се начин рада најчешћих алгоритама сортирања, као и њихова асимптотска временска сложеност.

2 Основни појмови

Дефиниција 1. Временска сложеност алгоритма је функција $T(n)$ која описује како број основних операција зависи од величине улаза n.

Код алгоритама сортирања, основна операција је најчешће **поређење елемената**. За упоређивање алгоритама користи се Big-O нотација:

$$T(n) = O(f(n))$$

3 Алгоритми квадратне сложености

3.1 Сортирање балончићима

Сортирање балончићима је једноставан алгоритам који више пута пролази кроз низ и упоређује суседне елементе, мењајући им места ако нису у исправном редоследу. Овај поступак се понавља све док низ не буде потпуно сортиран.

Број поређења у најгорем случају износи:

$$C(n) = \frac{n(n-1)}{2}$$

што доводи до временске сложености:

$$T(n) = O(n^2)$$

3.2 Сортирање одабиром

Сортирање одабиром у сваком кораку проналази најмањи елемент у несортираном делу низа и поставља га на одговарајућу позицију. Број поређења не зависи од почетног распореда елемената.

Због тога алгоритам увек има сложеност:

$$T(n) = O(n^2)$$

3.3 Сортирање уметањем

Сортирање уметањем гради сортирани део низа тако што сваки нови елемент убацује на одговарајуће место, слично начину на који се карте ређају у руци.

У најбољем случају (већ сортиран низ) важи:

$$T(n) = O(n)$$

док је у најгорем случају:

$$T(n) = O(n^2)$$

4 Алгоритми логаритамске сложености

4.1 Сортирање спајањем

Сортирање спајањем је алгоритам типа *подели па владај*. Низ се рекурзивно дели на две половине док се не добију низови дужине један, који се затим спајају у сортиран редослед.

Лема 1. *Временска сложеност сортирања спајањем алгоритма може се описати релацијом:*

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

Решавањем ове релације добија се:

$$T(n) = O(n \log n)$$

4.2 Брзо сортирање

Брзо сортирање такође користи приступ *подели па владај*. Алгоритам бира један елемент (пивот) и распоређује остале елементе тако да су мањи лево, а већи десно од пивота.

У просечном случају важи:

$$T(n) = O(n \log n)$$

Међутим, у најгорем случају (лош избор пивота):

$$T(n) = O(n^2)$$

5 Теоријска доња граница

Теорема 1. *Ниједан алгоритам сортирања заснован искључиво на поређењу не може имати временску сложеност бољу од* $O(n \log n)$ *у најгорем случају.*

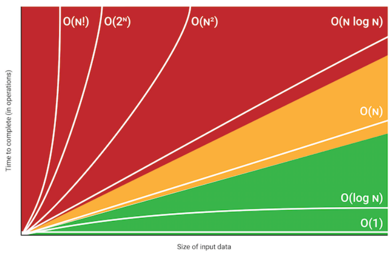
Ова теорема показује да алгоритми попут Merge sort-а имају асимптотски оптималну сложеност.

6 Табеларно поређење

Алгоритам	**Најбољи**	**Просечан**	**Најгори**
Сортирање балончићима	$O(n)$	$O(n^2)$	$O(n^2)$
Сортирање Уметањем	$O(n)$	$O(n^2)$	$O(n^2)$
Брзо сортирање	$O(n \log n)$	$O(n \log n)$	$O(n^2)$
Сортирање спајањем	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$

Табела 1: Поређење временске сложености алгоритама

7 Илустрација раста сложености



Слика 1: Поређење раста функција

Функције n^2 и $n \log n$ имају различито понашање за велике вредности n, што објашњава значај избора ефикасног алгоритма.

8 Закључак

Овај рад показује значајну разлику између различитих алгоритама за сортирање и придодаје значај одабиру истих.

Иако једноставни алгоритми имају образовну вредност, алгоритми сложености $O(n \log n)$ представљају практичан избор у реалним применама.