Համակարգչային գիտությունում **տեսակավորման ալգորիթմ**ը [ալգորիթմ](https://hy.wikipedia.org/wiki/%D4%B1%D5%AC%D5%A3%D5%B8%D6%80%D5%AB%D5%A9%D5%B4) է, որը դասավորում է էլեմենտները [ցանկից](https://hy.wikipedia.org/w/index.php?title=%D5%91%D5%A1%D5%B6%D5%AF_(%D5%B0%D5%A1%D5%B4%D5%A1%D5%AF%D5%A1%D6%80%D5%A3%D5%A1%D5%B5%D5%AB%D5%B6)&action=edit&redlink=1) որոշակի [կարգով](https://hy.wikipedia.org/w/index.php?title=%D4%B8%D5%B6%D5%A4%D5%B0%D5%A1%D5%B6%D5%B8%D6%82%D6%80_%D5%AF%D5%A1%D6%80%D5%A3&action=edit&redlink=1)։

Տեսակավորման ալգորիթմները գնահատվում են կատարման արագությամբ և հիշողության օգտագործման արդյունավետությամբ։

**Պղպջակային տեսակավորում** (անգլ [bubble sort](https://en.wikipedia.org/wiki/bubble_sort)), տեսակավորման պարզ [ալգորիթմ](https://hy.wikipedia.org/wiki/%D4%B1%D5%AC%D5%A3%D5%B8%D6%80%D5%AB%D5%A9%D5%B4), որը կարգավորում է զանգվածը շարունակաբար անցնելով զանգվածի վրայով և տեղերով փոխանակելով սխալ հերթականթյամբ շարված հարևան էլեմենտները[[1]](https://hy.wikipedia.org/wiki/%D5%8A%D5%B2%D5%BA%D5%BB%D5%A1%D5%AF%D5%A1%D5%B5%D5%AB%D5%B6_%D5%BF%D5%A5%D5%BD%D5%A1%D5%AF%D5%A1%D5%BE%D5%B8%D6%80%D5%B8%D6%82%D5%B4#cite_note-1)։ Սա շարունակվում է այնքան, մինչև այլևս կարիք չլինի փոխանակել էլեմենտները, ինչը կնշանակի, որ զանգվածը կարգավորված է։

|  |  |
| --- | --- |
| **Վատագույն դեպքում կատարումը** O(n^2) | O(n^2) |
| **Լավագույն դեպքում կատարումը** O(n) | O(n) |

**Ընտրության տեսակավորումը**[տեսակավորուման ալգորիթմ է](https://hy.wikipedia.org/w/index.php?title=%D5%8F%D5%A5%D5%BD%D5%A1%D5%AF%D5%A1%D5%BE%D5%B8%D6%80%D5%B8%D6%82%D5%B4%D5%A1%D5%B6_%D5%A1%D5%AC%D5%A3%D5%B8%D6%80%D5%AB%D5%A9%D5%B4_%D5%A7&action=edit&redlink=1), մասնավորապես [տվյալների համեմատության տեսակավորում է](https://hy.wikipedia.org/w/index.php?title=%D5%8F%D5%BE%D5%B5%D5%A1%D5%AC%D5%B6%D5%A5%D6%80%D5%AB_%D5%B0%D5%A1%D5%B4%D5%A5%D5%B4%D5%A1%D5%BF%D5%B8%D6%82%D5%A9%D5%B5%D5%A1%D5%B6_%D5%BF%D5%A5%D5%BD%D5%A1%D5%AF%D5%A1%D5%BE%D5%B8%D6%80%D5%B8%D6%82%D5%B4_%D5%A7&action=edit&redlink=1)։ Այն ունի [O](https://hy.wikipedia.org/w/index.php?title=Big_O_notation&action=edit&redlink=1)(*n*2) բարդությունը, դարձնելով անարդյունավետ խոշոր ցուցակները, և ընդհանրապես այն որպես ալգորիթմ իրականացվում է ավելի վատ, քան համանման [ներդրման տեսակավորումը](https://hy.wikipedia.org/w/index.php?title=%D5%86%D5%A5%D6%80%D5%A4%D6%80%D5%B4%D5%A1%D5%B6_%D5%BF%D5%A5%D5%BD%D5%A1%D5%AF%D5%A1%D5%BE%D5%B8%D6%80%D5%B8%D6%82%D5%B4%D5%A8&action=edit&redlink=1)։ Ընտրությանը տեսակավոումը նշանակալի է իր պարզությամբ և ալգորիթմի առավելություն նրանում է, որ կարող է կատարել առավել բարդ խնդիրներ, մասնավորապես երբ օժանդակ հիշողությունը սահմանափակ է։

Ալգորիթմը աշխատում է հետևյալ կերպ `

1. Գտնել նվազագույն արժեքը ցուցակում,
2. Փոխանակել այն արժեքի հետ, որը գտնվում է առաջին տեղում,
3. Վերը նշված քայլերը կրկնել համար ցուցակի մնացած մասի համար (սկսած երկրորդ հորիզոնականը զբաղեցնողից մինչև հաջորդը և այդպես յուրաքանչյուր անգամ):

Ցուցակի արդյունավետ կերպով բաժանվում է երկու մասի. ենթացուցակ, որն արդեն տեսակավորված է, կառուցված ձախից աջ, և գտնվում է սկզբում, և ենթացուցակ, որը պետք է տեսակավորվի, զբաղեցնելով համախմբում մնացած մասը։

|  |  |
| --- | --- |
| [**Վատագույն դեպքում կատարումը**](https://hy.wikipedia.org/w/index.php?title=%D4%BC%D5%A1%D5%BE%D5%A1%D5%A3%D5%B8%D6%82%D5%B5%D5%B6,_%D5%BE%D5%A1%D5%BF%D5%A1%D5%A3%D5%B8%D6%82%D5%B5%D5%B6_%D6%87_%D5%B4%D5%AB%D5%BB%D5%AB%D5%B6_%D5%A4%D5%A5%D5%BA%D6%84%D5%A5%D6%80%D5%B8%D6%82%D5%B4&action=edit&redlink=1) | O(n^2) |
| [**Լավագույն դեպքում կատարումը**](https://hy.wikipedia.org/w/index.php?title=%D4%BC%D5%A1%D5%BE%D5%A1%D5%A3%D5%B8%D6%82%D5%B5%D5%B6,_%D5%BE%D5%A1%D5%BF%D5%A1%D5%A3%D5%B8%D6%82%D5%B5%D5%B6_%D6%87_%D5%B4%D5%AB%D5%BB%D5%AB%D5%B6_%D5%A4%D5%A5%D5%BA%D6%84%D5%A5%D6%80%D5%B8%D6%82%D5%B4&action=edit&redlink=1) | O(n^2)k |

**Ներդրմամբ տեսակավորումը** պարզ [տեսակավորման ալգորիթմ](https://hy.wikipedia.org/wiki/%D5%8F%D5%A5%D5%BD%D5%A1%D5%AF%D5%A1%D5%BE%D5%B8%D6%80%D5%B4%D5%A1%D5%B6_%D5%A1%D5%AC%D5%A3%D5%B8%D6%80%D5%AB%D5%A9%D5%B4) է, որը ստեղծում է վերջնական սորտավորված [զանգված](https://hy.wikipedia.org/w/index.php?title=%D4%B6%D5%A1%D5%B6%D5%A3%D5%BE%D5%A1%D5%AE_(%D5%AE%D6%80%D5%A1%D5%A3%D6%80%D5%A1%D5%BE%D5%B8%D6%80%D5%B8%D6%82%D5%B4)&action=edit&redlink=1) կամ ցանկ։

Ալգորիթմի յուրաքանչյուր քայլում մենք ընտրում ենք մուտքագրված տվյալներից մեկ էլեմենտ և տեղադրում ենք այն համապատասխան տեղը, որտեղ արդեն տեսակավորված են, այնքան ժամանակ քանի դեռ մուտքագրված տվյալների հավաքածուները ավարտված կլինեն։

**Արագ տեսակավորում** ([անգլ.](https://hy.wikipedia.org/wiki/%D4%B1%D5%B6%D5%A3%D5%AC%D5%A5%D6%80%D5%A5%D5%B6)՝ quicksort) հաճախ անվանում են qsort C լեզվի ստանդարտ գրադարանի իրականացման անունով։ Այն հայտնի դասակարգման [ալգորիթմ](https://hy.wikipedia.org/wiki/%D4%B1%D5%AC%D5%A3%D5%B8%D6%80%D5%AB%D5%A9%D5%B4) է, որը մշակվել է անգլիացի ինֆորմատիկ [Չարլզ Հոարի](https://hy.wikipedia.org/wiki/%D5%89%D5%A1%D6%80%D5%AC%D5%A6_%D4%B1%D5%B6%D5%BF%D5%B8%D5%B6%D5%AB_%D5%8C%D5%AB%D5%B9%D5%A1%D6%80%D5%A4_%D5%80%D5%B8%D5%A1%D6%80) կողմից 1960 թվականին։ Զանգվածի տեսակավորման առաջին արագ ունիվերսալ ալգորիթմն է։ *n* հատ տարրերի տեսակավորման համար կատարում է միջինը O (*n* log *n*) համեմատություն։ Ամենավատ դեպքում կատարում է O(*n*2) համեմատություն, չնայած այսպիսի բան պատահում է հազվադեպ։ Quicksort-ը հաճախ ավելի արագ է գործում, քան այլ O(*n* log *n*) ալգորիթմները[[2]](https://hy.wikipedia.org/wiki/%D4%B1%D6%80%D5%A1%D5%A3_%D5%BF%D5%A5%D5%BD%D5%A1%D5%AF%D5%A1%D5%BE%D5%B8%D6%80%D5%B8%D6%82%D5%B4#cite_note-2)։

Արագ տեսակավորումը օգտագործում է [«Բաժանիր և տիրիր»](https://hy.wikipedia.org/w/index.php?title=%D4%B2%D5%A1%D5%AA%D5%A1%D5%B6%D5%AB%D6%80_%D6%87_%D5%BF%D5%AB%D6%80%D5%AB%D6%80&action=edit&redlink=1) ստրատեգիան։ Ալգորիթմի քայլերի հաջորդականությունը հետևյալն է՝

1. Ընտրում ենք զանգվածից մի քանի էլեմենտ, որոնք պետք է անվանենք հենակետային էլեմենտ։ Ալգորիթմի կոռեկտության տեսանկյունից հենակետային էլեմենտի ընտրությունը անկարևոր է։ Ալգորիթմի էֆեկտիվության բարձրացման տեսանկյունից պետք է ընտրվի մեդիանան, բայց առանց լրացուցիչ տեսակավորված տվյալների հնարավոր չէ տեղեկություն ստանալ։ Հայտնի ստրատեգիա է՝ մշտապես ընտրել միևնույն էլեմենտը, օրինակ մեջտեղի կամ վերջին զետեղվածը, ընտրել պատահական ինդեքսով ընտրված էլեմենտ։
2. Զանգվածից բաժանման օպերացիան՝ վերակազմել զանգվածը այնպես, որպեսզի բոլոր էլեմենտները փոքր կամ հավասար գտնվեն հենակետային էլեմենտից ձախ, իսկ հենակետայինից մեծը՝ նրանից աջ։ Օպերացիայի ալգորիթմն է՝
   1. 2 ինդեքսներ - l և r, հավասարվում են բաժանվող զանգվածի մաքսիմալ և մինիմալ ինդեքսին։
   2. Առանձնացվում է ինդեքսի m հենակետային էլեմենտը։
   3. Ինդեքս l-ին մեծանում է մինչև m-ը այնքան ժամանակ, քանի դեռ l էլեմենտը չի գերազանցել հենակետայինին։
   4. Ինդեքս r-ը փոքրանում է մինչև m-ը այնքան ժամանակ, քանի դեռ r էլեմենտը չի թվացել փոքր կամ հավասար հենակետայինին։
   5. Եթե r = l - գտնվում է զանգվածի մեջտեղում - բաժանման օպերացիան ավարտված է, 2 ինդեքսներն էլ ցույց են տալիս հենակետային էլեմենտը։
   6. Եթե l < r - գտնված զույգ էլեմենտները հարկավոր է փոխել տեղերով և շարունակել բաժանման օպերացիան մինչև l և r հասանելի լինեն։ Հարկավոր է հաշվել, որ եթե ինչ-որ սահման (l կամ r) հասել է մինչև հենակետային էլեմենտի, ապա m-ի փոխանակության իմաստը փոխվում է r կամ l էլեմենտի համապատասխանաբար։
3. [Ռեկուրսիվ](https://hy.wikipedia.org/wiki/%D5%8C%D5%A5%D5%AF%D5%B8%D6%82%D6%80%D5%BD%D5%AB%D5%BE) կարգավորում ենք զանգվածները հենակետային էլեմենտների աջ և ձախ կողմերում։
4. Ռեկուրսիայի բազան համարվում է հավաքածու, որը բաղկացած է մեկ կամ երկու էլեմենտներից։ Առաջինը վերադառնում է նախնական դիրքի, իսկ երկրորդ՝ անհրաժեշտության դեպքում, տեսակավորումը հասնում է 2 էլեմենտների վերադասավորմանը։

Առայժմ մեր սովորած տեսակավորման ալգորիթմներն են [ընտրանքային տեսակավորումը](https://hy.khanacademy.org/computing/computer-science/algorithms/sorting-algorithms/a/sorting) և [մուտքային տեսակավորումը](https://hy.khanacademy.org/computing/computer-science/algorithms/insertion-sort/a/insertion-sort), որոնց վատագույն հաշվողական դժվարությունը \Theta(n^2)Θ(*n*2)\Theta, left parenthesis, n, squared, right parenthesis է։ Մուտքագրված մեծածավալ զանգվածն այս երկու ալգորիթմները կարող է շատ երկար ժամանակում տեսակավորել։ Այս և հաջորդ տեսանյութերում մենք կծանոթանանք ևս երկու ալգորիթմների հետ՝ միաձուլման տեսակավորում և արագ տեսակավորում, որոնք ավելի արագ են աշխատում։ Միաձուլման տեսակավորումը բոլոր դեպքերում ունի \Theta(n \lg n)Θ(*n*lg*n*)\Theta, left parenthesis, n, \lg, n, right parenthesis հաշվողական դժվարություն, մինչդեռ արագ տեսակավորման լավագույն հաշվողական դժվարությունը \Theta(n \lg n)Θ(*n*lg*n*)\Theta, left parenthesis, n, \lg, n, right parenthesis է, իսկ վատագույնը՝ \Theta(n^2)Θ(*n*2)\Theta, left parenthesis, n, squared, right parenthesis։ Այս աղյուսակը ցույց է տալիս մեր սովորած չորս տեսակավորման ալգորիթմները և դրանց հաշվողական դժվարությունները․

| **Ալգորիթմ** | **Վատագույն հաշվողական դժվարություն** | **Լավագույն հաշվողական դժվարություն** | **Միջին հաշվողական դժվարություն** |
| --- | --- | --- | --- |
| Ընտրանքային տեսակավորում | \Theta(n^2)Θ(*n*2)\Theta, left parenthesis, n, squared, right parenthesis | \Theta(n^2)Θ(*n*2)\Theta, left parenthesis, n, squared, right parenthesis | \Theta(n^2)Θ(*n*2)\Theta, left parenthesis, n, squared, right parenthesis |
| Մուտքային տեսակավորում | \Theta(n^2)Θ(*n*2)\Theta, left parenthesis, n, squared, right parenthesis | \Theta(n)Θ(*n*)\Theta, left parenthesis, n, right parenthesis | \Theta(n^2)Θ(*n*2)\Theta, left parenthesis, n, squared, right parenthesis |
| Միաձուլման տեսակավորում | \Theta(n \lg n)Θ(*n*lg*n*)\Theta, left parenthesis, n, \lg, n, right parenthesis | \Theta(n \lg n)Θ(*n*lg*n*)\Theta, left parenthesis, n, \lg, n, right parenthesis | \Theta(n \lg n)Θ(*n*lg*n*)\Theta, left parenthesis, n, \lg, n, right parenthesis |
| Արագ տեսակավորում | \Theta(n^2)Θ(*n*2)\Theta, left parenthesis, n, squared, right parenthesis | \Theta(n \lg n)Θ(*n*lg*n*)\Theta, left parenthesis, n, \lg, n, right parenthesis | \Theta(n \lg n)Θ(*n*lg*n*)\Theta, left parenthesis, n, \lg, n, right parenthesis |

\small{3}3\small{5}5\small{7}7\small{9}9\small{11}11\small{13}13\small{15}15\small{17}17\small{19}19\small{0}0\small{10}10\small{20}20n*n*\ n log \_{2} n *nlog*2​*n*\ n^2 *n*2\ n *n*

**Բաժանիր և տիրիր**

Միաձուլման տեսակավորումն ու արագ տեսակավորումն ունեն ռեկուրսիայի հիման վրա ստեղծված ալգորիթմական ստրատեգիա։ Այս ստրատեգիան՝ **Բաժանիր և տիրիր**, բաժանում է խնդիրը ենթախնդիրների, որոնք նման են դրան, ռեկուրսիվորեն լուծում է ենթախնդիրները և վերջապես միավորում է դրանք, որպեսզի դրանց օգնությամբ լուծի իրական խնդիրը։ Քանի որ «Բաժանիր և տիրիր» ռազմավարությունը ռեկուրսիվորեն է լուծում ենթախնդիրները, բոլոր ենթախնդիրները պետք է ավելի փոքր լինեն, քան խնդիրը, և ենթախնդիրների համար պետք է լինի առաջին քայլ։ Կարող ենք ասել, որ «Բաժանիր և տիրիր» ալգորիթմը երեք մաս ունի․

1. **Բաժանել** խնդիրն ինչ-որ քանակի ենթախնդիրների, որոնք խնդրի փոքր մասն են կազմում։
2. **Տիրել** ենթախնդիրներին՝ դրանք ռեկուրսիվորեն լուծելով։ Եթե դրանք բավականաչափ փոքր են, կարող ես լուծել որպես առաջին քայլ։
3. **Միավորել** ենթախնդիրների լուծումներն իրական խնդրին։

Դու հեշտությամբ կարող ես հիշել «Բաժանիր և տիրիր» ալգորիթմի քայլերը՝ որպես *բաժանել, տիրել, միավորել*։ Ահա թե ինչպես պատկերացնել այն՝ համարելով, որ մի բաժանումը ստեղծում է երկու ենթախնդիր (չնայած «Բաժանիր և տիրիր» ռազմավարության որոշ ալգորիթմներ ստեղծում են ավելին, քան երկուսը)․