

3) Sunt date elemente cu cheile: 19, 33, 40, 25, 12, 24, 31, 48, 23, 17.

Sortarea rapidă (quicksort)

Mai jos este explicată varianta recursivă:

1. **Se alege o valoare pivot**. Se ia valoarea elementului din mijloc ca valoare pivot, dar poate fi oricare altă valoare, care este în intervalul valorilor sortate, chiar dacă nu este prezentă în tablou.

2. **Partiţionare,** Se rearanjează elementele în aşa fel încât, toate elementele care sunt mai mari decât pivotul merg în partea dreaptă a tabloului. Valorile egale cu pivotul pot sta în orice parte a tabloului. În plus, tabloul poate fi împărţit în părţi care nu au aceeaşi dimensiune (nu sunt egale).

3. **Se sortează amândouă părţile**.se aplică recursiv algoritmul de sortare rapidă în partea stângă şi în partea dreaptă.

**Algoritmul de partiţie în detaliu.**

Există 2 indici i şi j, şi la începutul algoritmului de partiţionare i indică primul element al tabloului iar j indică ultimul element din tablou. La pasul următor algoritmul mută i înainte, până când un element cu o valoare mai mare sau egală cu pivotul este găsită. Indicele j este mutat înapoi, pâna când un element cu valoare mai mică sau egală cu pivotul este găsită. Dacă i<=j atunci i merge pe poziţia i+1 iar j merge pe poziţia j-1. Algoritmul se opreşte, când i devine mai mare decât j.

19, 33, 40, 25, 12, 24, 31, 48, 23, 17 - nesortat

19, 33, 40, 25, 12, 24, 31, 48, 23, 17 - valoarea pivot = 12  
19, 33, 40, 25, 12, 24, 31, 48, 23, 17 – interschimbam 12 cu 17.

Când marginea dreaptă traversează marginea stângă, toate elementele din stânga marginii stângi sunt mai mici decât pivotul și toate elementele din dreapta sunt mai mari sau egale cu pivotul.

19, 33, 40, 25, 17, 24, 31, 48, 23, 12 - interschimbam 19 cu 12.

12, 33, 40, 25, 17, 24, 31, 48, 23, 19 – selectam pivot – 24

12, 33, 40, 25, 17, 24, 31, 48, 23, 19 – interschimbam 24 cu 19

Deplasam marginea stângă către dreapta până când atinge o valoare mai mare sau egală cu pivotul.

12, 33, 40, 25, 17, 19, 31, 48, 23, 24 – interschimbam 33 cu 23

12, 23, 40, 25, 17, 19, 31, 48, 33, 24 -- interschimbam 40 cu 19

12, 23, 19, 25, 17, 40, 31, 48, 33, 24 – interschimbam 25 cu 17

12, 23, 19, 17, 25, 40, 31, 48, 33, 24 interschimbam 25 cu 24

Când marginea dreaptă traversează marginea stângă, toate elementele din stânga marginii stângi sunt mai mici decât pivotul și toate elementele din dreapta sunt mai mari sau egale cu pivotul.

12, 23, 19, 17, 24, 40, 31, 48, 33, 25 interschimbam 19 cu 17

12, 23, 17, 19, 24, 40, 31, 48, 33, 25 interschimbam 23 cu 17

12, 17, 23, 19, 24, 40, 31, 48, 33, 25 interschimbam 23 cu 19

12, 17, 19, 23, 24, 40, 31, 48, 33, 25 interschimbam 48 cu 25

12, 17, 19, 23, 24, 40, 31, 25, 33, 48 selectam pivot si interschimbam 31 cu 33

12, 17, 19, 23, 24, 40, 33, 25, 31, 48 interschimbam 40 cu 25

12, 17, 19, 23, 24, 25, 33, 40, 31, 48 interschimbam 33 cu 31

12, 17, 19, 23, 24, 25, 31, 40, 33, 48 selectam pivot si interschimbam 40 cu 33

12, 17, 19, 23, 24, 25, 31, 33, 40, 48 sortat.

Unul dintre dezavantajele sortării prin inserţie este faptul că la fiecare etapă un element al şirului se deplasează cu o singură poziţie. O variantă de reducere a numărului de operaţii efectuate este de a compara elemente aflate la o distanţă mai mare *(*h *≥* 1) şi de a realiza deplasarea acestor elemente peste mai multe poziţii. Un şir x[1..n]este considerat h-sortat dacă orice subşir x[i0],x[i0 + h],x[i0 + 2h]... este sortat (i0 ∈ {1,...,h}).Aceasta este ideea algoritmului cunoscut sub numele ”shell sort”.

Elementul cheie al algoritmului îl reprezintă alegerea valorilor pasului h*.* Pentru alegeri adecvate ale secvenţei hkse poate obţine un algoritm de complexitate *O*(n3/2)în loc de *O*(n2) cum este în cazul algoritmului clasic de sortare prin inserţie.

**Heap Sort**

Se numeşte ansamblu (heap) a secvenţă de chei h1, h2,..., hn care satisfac condiţiile: hi <= h2i si hi <= h2i+1 i=1,N/2.

Se aduce tabloul la forma unui ansamblu, adică pentru orice i,j, k din intervalul [1,N], unde j=2\*i si k=2\*i+1, să avem a[i]<=a[j] si a[i]<=a[k] (\*). Se observă că în acest caz a[1] este elementul cu cheia minimă în tablou. Se interschimbă elementele a[1] şi a[N] şi se aduce subtabloul a[1],...,a[N-1] la forma de ansamblu, apoi se interschimbă elementele a[1] si a[N-1] şi se aduce subtabloul a[1],...,a[N-2] la forma de ansamblu ş.a.m.d. În final rezultă tabloul ordonat invers. Dacă se schimbă sensul relaţiilor în condiţiile (\*) atunci se obţine o ordonare directă a tabloului (a[1] va fi elementul cu cheia maximă).

Aducerea unui tablou la forma de ansamblu se bazează pe faptul că subtabloul a[N/2+1],...,a[N] este deja un ansamblu (nu există indicii j si k definiţi ca mai sus). Acest subtablou se va extinde mereu spre stânga cu câte un element al tabloului, pâna când se ajunge la a[1]. Elementul adăugat va fi glisat astfel încât subtabloul extins să devină ansamblu.

Procedura Deplasare(s,d) realizează glisarea elementului a[s] astfel că subtabloul a[s],...,a[d] (s<d) să devină ansamblu. Această procedură este folosită mai întâi pentru aducerea întregului tablou la structura de ansamblu şi apoi pentru ordonarea tabloului conform metodei enunţate mai sus.

Timpul de execuţie al sortării este O(N\*log N).

19, 33, 40, 25, 12, 24, 31, 48,

Pasul 1: 19 este schimbat cu 48  
Pasul 2: 19 este deconectat de la heap  
ca 19 este în poziția corectă acum și. Pasul 3: Max-heap este creat și 40 este schimbat cu 31  
Pasul 4: 40 este deconectat de la heap.

Metoda insertiei:

Sortarea inserției se bazează pe ideea că un element din elementele de intrare este consumat în fiecare iterație pentru a-și găsi poziția corectă, adică poziția căreia îi aparține într-o matrice sortată.

Iterează elementele de intrare prin creșterea matricei sortate la fiecare iterație. Compară elementul curent cu cea mai mare valoare din matricea sortată. Dacă elementul curent este mai mare, atunci lasă elementul în locul său și trece la elementul următor, își găsește poziția corectă în matricea sortată și îl mută în acea poziție. Acest lucru se face prin deplasarea tuturor elementelor, care sunt mai mari decât elementul curent, în matrice sortate la o poziție înainte

19, 33, 40, 25, 17, 24, 31, 48, 23, 12

Deci verificam daca 19>33 schimbam locul cifrelor ,daca nu raman pe loc

Apoi verificam daca 33 cu 40 daca nu este mai mare il lasam pe pozitia corecta

Apoi 40 > 25 – da atunci schimbam locul 40 cu 25 si observam ca 25 este mai mic si ca 33 deci schimbam locul si cu 33

Apoi verificam daca 17 > 24 nu il lasam asa dar observam ca numere pana la el sunt mai mare il mutam pe prima pozitie

Deci avem sirul 17, 19,24,25,33,40 ,31,48,23,12

Si astfel verificam pana avem tot sirul sortat

Complexitate

*Sortarea prin inserţie cu pas variabil* este un algoritm de sortare, care în implementarea originală necesită un număr de O(n2) comparaţii şi înlocuiri de elemente, în cazul în care algoritmul se comportă ineficient. Algoritmul de sortare prin inserţie cu pas variabil este uşor de depanat dacă se doreşte a se vedea cum funcţioneză acest algoritm, însă este mai dificil să se examineze timpul de execuţie al algoritmului.

Dacă se aplică sortarea prin interclasare în cadrul unei liste de lungime n atunci timpul de sortare a acesteia va fi de T(n), unde relaţia T(n)=2T(n/2) + n provine de la ideea de bază a algoritmului . Adică, perioada de sortare T(n) a unei liste de dimensiune n depinde şi de perioada de sortare a celor două sub-liste: T(n)=2T(n/2) + n, plus număr de treceri pentru a sorta cele două sub-liste rezultate.

În cazul în care sortarea prin interclasare se comportă ineficient aceasta va efectua un număr de (*n***|**lg *n***|** - 2**|**lg *n***|** + 1) comparaţii, care se cuprind in intervalul (*n* lg *n- n*+1) şi (*n* lg *n + n* + O(lg*n*)). Astfel că în cazul în care sortarea prin interclasare se comportă ineficient, se efectuează cu 39% mai puţine comparaţii ale elementelor decât în cazul comportării mulţumitoare a sortării rapide, deoreace sortarea prin interclasare va efectua întotdeauna mai puţine comparaţii decât cel mai rapid algoritm de sortare - sortarea rapidă (quicksort), excepţie facând cazurile extrem de rare când comportarea ineficientă a sortării prin interclasare este mai performantă decât comportarea eficientă a sortării rapide.Sortarea prin interclasare cât şi sortarea rapidă pot sorta şi cazuri mai complexe, ambele finalizând sortarea intr-un timp definit prin relaţia O(*n* log *n*).

Implementările recursive ale algoritmului de sortare prin interclasare efectuează printr-un număr de 2n-1 treceri în cazul când acesta se comportă ineficient, care în comparaţie cu sortarea rapidă care efectuează un număr de n treceri, aceasta însemnând că sortarea prin interclasare este mult mai recursivă decât sortarea rapidă.

Implementările acestei sortări ocolesc metoda de efectuare a mai multor treceri decât este necesar, astfel fiind uşor de implementat.

Cele mai multe implementări ale sortării prin interclasare nu efectuează sortări "pe loc", astfel, capacitatea memoriei alocată la începutul procedeului de sortare are aceeaşi dimensiune ca şi capacitatea memoriei folosită pentru a stoca datele şi după terminarea procesului de sortare.Deasemeni este permisă utilizarea conceptului de sortare pe loc în cadrul algoritmului de sortare prin interclasare, însă va rezulta un algoritm foarte complicat care va avea o performanţă foarte scazută din puncrt de vedere practic, astfel că algoritmul nu va mai rula într-un timp de O(*n* log*n*).În acest caz metoda de sortare prin ansamble (heapsort) are o comportare mult mai eficientă, deoarece prezintă un algoritm mai uşor de implemetat.

*Sortarea prin inserare* este un algoritm simplu de sortare prin compararea unor elemente aflate în tablouri de date sortate, construind câte o intrare la un anumit timp.

Acest tip de algoritm este mai puţin eficient în lucrul cu liste mari de date, şi nu are performanţa algoritmilor de sortare avansaţi cum sunt algoritmul de sortare rapidă quick sort sau merge sort.

4) Metoda Fibonacci

Inserarea unui nod într-un heap deja existent urmează pașii de mai jos.

Creați un nod nou pentru element.

Verificați dacă grămada este goală.

Dacă heap-ul este gol, setați noul nod ca nod rădăcină și marcați-l

Altfel, inserați nodul în lista rădăcină și actualizați