КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: АЛГОРИТМИ СТРУКТУРИ ОТ ДАННИ
ДИСЦИПЛИНА: СИНТЕЗ И АНАЛИЗ НА АЛГОРИТМИ

ЛАБОРАТОРНО УПРАЖНЕНИЕ № 7

ТЕМА: Алгоритми за бързо сортиране

ЦЕЛ:

Целта на упражнението е студентите да се запознаят с бързите алгоритми за сортиране. След упражнението студентите би следвало да могат да прилагат съответните алгоритми върху различни структури от данни.

I. ТЕОРЕТИЧНА ЧАСТ

В предното упражнение се запознахме с някои по бавни алгоритми за сортиране. В това ще разгледаме други алгоритми които са по бързи спрямо предходните. Това са:

- *Бързо сортиране (Quick Sort)*

Това е ефективен алгоритъм за сортиране, и един от често използваните такива. Когато се изпълнява добре той може да бъде до два или 3-пъти по бърз от основните му конкуренти, сортиране чрез сливане и пирамидално сортиране. Алгоритъмът за бързо сортиране е на принципа разделяй и владей. Той работи чрез избиране на „разделящ“ елемент от списъка и разделяне на останалите елементи на два под списъка, в зависимост от това дали те са по малки или по големи от разделящия. След това под – списъците се сортират рекурсивно.

Математическият анализ на бързата сортировка показва, средно алгоритъма има сложност O(n.log n), за да сортира n елемента. В най – лошия случай има сложност $O(n^2)$, въпреки че, това поведение, за този случай е изключително рядко.

Описанието на алгоритъма е следното. Първо разделя входния списък на два по – малки под – списъка: ниските елементи и високите елементи. След това рекурсивно сортира под – списъците. Стъпките на алгоритъма са следните:

- Избира се „разделящ“ елемент;
- Пренарежда се списъка така че всички елементи със стойности по – малки от разделящия да се намират преди него, а всички елементи с по – големи стойности да са след него.
- Рекурсивно се прилагат горните стъпки върху под – списъците.

- *Сортиране чрез сливане (merge sort)*

Сортирането чрез сливане е алгоритъм за сортиране, базиран на сравняване, който винаги има сложност O(nlog(n)). Алгоритъмът отново се гради на принципа „разделяй и владей“. Принципът на действие за алгоритъма е следния:

- Несортираният списък по произволен начин се разделя на два подсписъка в приблизително еднаква дължина (за линейно време).
- Рекурсивно се разделят списъците докато не се достигне до списъци с единична дължина (списъци от по 1 елемент)
- Сравняват се и се обединяват елементите от два по два съседни подсписъка.

- Обединяването се извършва рекурсивно докато не се стигне един списък който ще е сортиран.

- *Пирамидално сортиране (Heapsort)*

Този вид сортиране е на принципа на определяне на най-малкия и най-големия елемент в списъка и поставянето им в двата края, след което прави същото и за останалата част на несортирания списък. За да се реализира този алгоритъм, несортираният списък се представя като специална структура от данни heap, наречена Двоично дърво. Веднъж представен списъкът по този начин се гарантира, че коренът на дървото има най-големия (или най-малкия) елемент от списъка. Когато този елемент се вземе и постави в началото на сортирания списък, останалите елементи се пренареждат отново така че да коренът отново да бъде най-големият (или най-малкият) елемент от останалия несортиран списък. Използвайки heap-ът намирането на най-големия елемент отнема O(log n) време, вместо O(n) при линейно търсене. Това позволява пирамидалното сортиране да има времева ефективност O(n log n), както и при най-лош сценарий.

II. ПРАКТИЧЕСКА ЧАСТ

В практическата част са показани примери, които показват действието на разглежданите алгоритми за сортиране.

ЗАДАЧА1: *Да се сортира следния масив от числа по методите на разгледаните бързи алгоритми за сортиране в теоретичната част.*

{47, 34, 43, 59, 32, 58, 79, 84, 72, 61, 34, 77}

РЕШЕНИЕ:

1. Бързо сортиране (Quick sort).

*Разделящ елемент е **47***

1) *{**47**, 34, 43, 59, 32, 58, 79, 84, 72, 61, **34**, 77} – от дясно на ляво се пропускат всички по големи от него и се разменя с първия по малък*
2) *{34, 34, 43, **59**, 32, 58, 79, 84, 72, 61, **47**, [77]} – от ляво на дясно (от несортираната част) се пропускат всички по – малки от него и се разменя с първия по – голям*
3) *{[34, 34, 43], **47**, **32**, 58, 79, 84, 72, 61, 59, [77]} – от дясно на ляво (от несортираната част) се пропускат всички по – големи от него и се разменя с първия по – малък*
4) *{[34, 34, 43, 32],[**47**],[58, 79, 84, 72, 61, 59, 77]} – приключва първото разделяне на списъка.*
5) *…… // Стъпките се повтарят за отделните под-списъци докато не се достигне подсписъци с по 1 елемент.*
6) *{**32**, **34**, **34**, **43**, **47**, **58**, **59**, **61**, **72**, **77**, **79**, **84**} – край на алгоритъма*

2. Сортиране чрез сливане (merge sort)

1) *{47, 34, 43, 59, 32, 58} и {79, 84, 72, 61, 34, 77} - разделяне на два подсписъка*
2) *{47, 34, 43} и {59, 32, 58} и {79, 84, 72} и {61, 34, 77}*
3) *{47, 34, 43} и {59, 32, 58} и {79, 84, 72} и {61, 34, 77}*
4) *……*

5) *{**47**},{**34**},{**43**},{**59**},{**32**},{**58**},{**79**},{**84**},{**72**},{**61**},{**34**},{**77**}* - *крайно разделяне на списъците*

//Следва обединение на съседните списъци два по два

6) *{**34**, 47},{**43**, 59},{**32**, 58},{**79**, 84},{**61**, 72},{**34**, 77}*

*6.1) {**34**, 47} и {**43**, 59}-> {**34**},{**47**} и {**43**, 59}->{**34**, **43**},{**47**} и {**59**} -> {**34**, **43**, **47**, **59**}*

*6.2) {**32**, 58} и {**79**, 84} -> {**32**},{**58**}и {**79**, 84}->{**32**, **58**}, {} и {**79**, 84} ->{**32**, **58**, **79**, **84**}*

*6.3) {**61**, 72} и {**34**, 77} -> {**34**},{**61**,72} и {**77**} -> {**34**, **61**},{**72**} и {77}->{**34**, **61**, **72**, **77**}*

7) *{34, 43, 47, 59}, {32, 58, 79, 84}, {34, 61, 72, 77}*

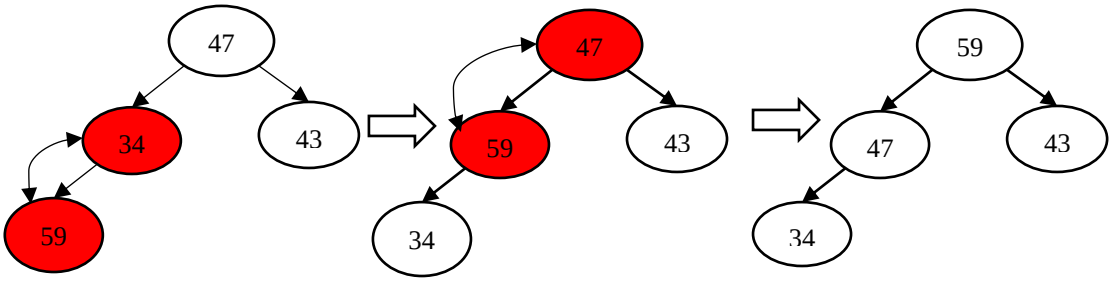
……

//следва повтаряне на същите стъпки докато се получи един списък със сортирани елементи

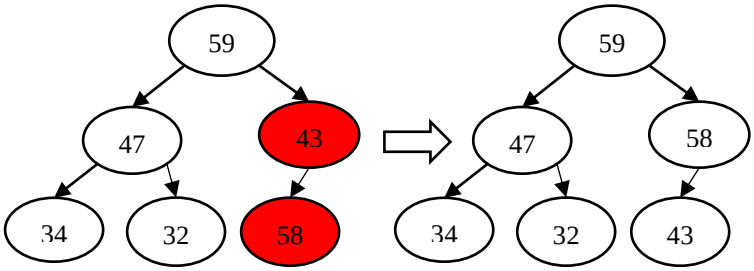
*{**32**, **34**, **34**, **43**, **47**, **58**, **59**, **61**, **72**, **77**, **79**, **84**}*

3. *Пирамидално сортиране(heapsort)*

1) *{**47, 34, 43, 59**, 32, 58, 79, 84, 72, 61, 34, 77} – маркираните елементи се добавени в дървото. Те принципно се премахват от списъка. Затова в следващите стъпки те добавените елементи са премахнати от списъка*

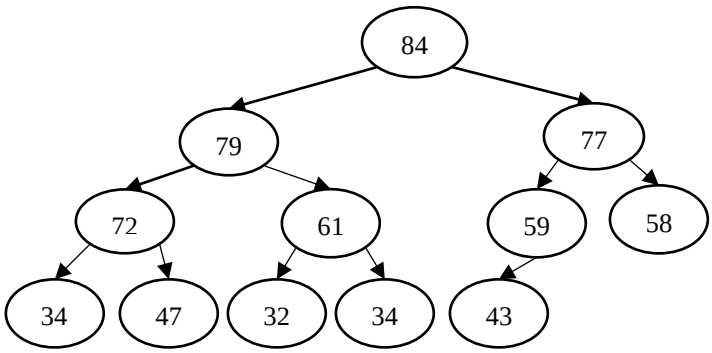


2) ***{ 32, 58**, 79, 84, 72, 61, 34, 77}*



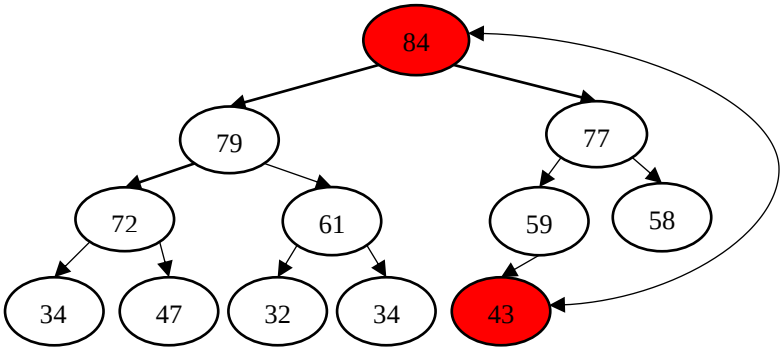
3) *……*

// Добавянето на елементи към двоичното дърво продължава докато списъка остане празен.

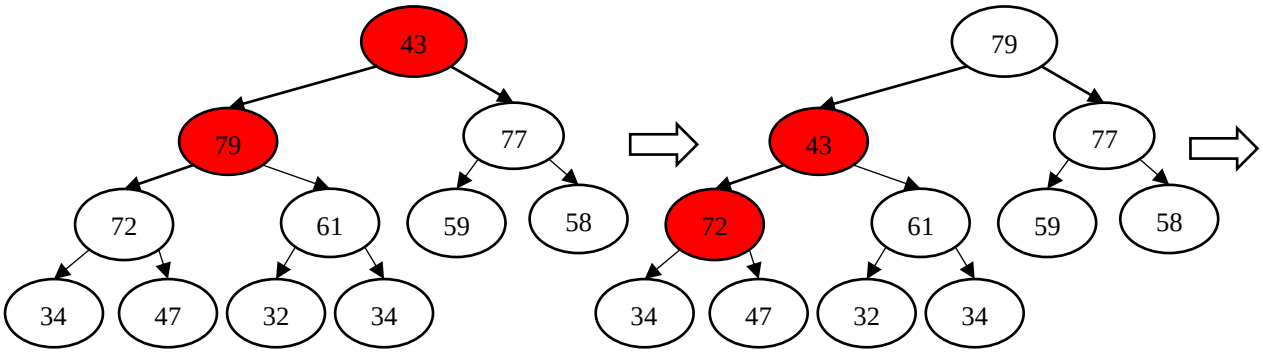


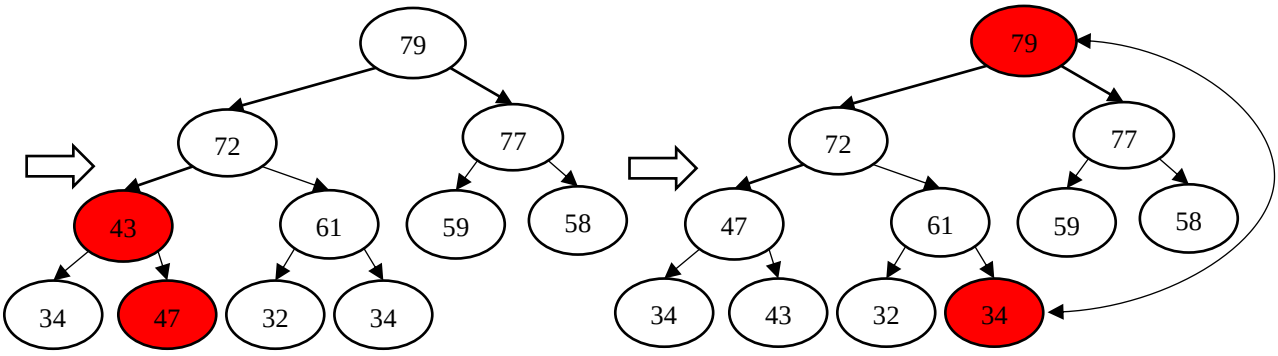
4) *{**84**, 79, 77, 72, 61, 59, 58, 34, 47, 32, 34, **43**} – Добавяме елементите към списъка последователно следвайки структурата на дървото. Върха на дървото със сигурност е най големия елемент и го разменяме със последното листо (т.е. се разменят първия и последния елемент от списъка). По този начин най – големия елемент вече ще си е на мястото подреден.*

*2*к+1 2*к+2*



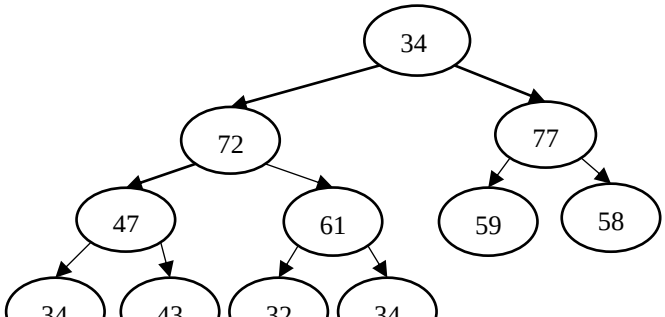
5) *{**43**, 79, 77, 72, 61, 59, 58, 34, 47, 32, 34, **84**} – Подредения елемент вече не се добавя към дървото. Следват стъпки подреждащи елементите така че родителите отново да са по големи от синовете.*





*{**79**, 72, 77, 47, 61, 59, 58, 34, 43, 32, **34**, **84**} – списъка с елементи до този момент.*

6) *{**34**, 72, 77, 47, 61, 59, 58, 34, 43, 32, **79**, **84**} – вече последните два елемента са си на местата. Дървото изглежда по следния начин:*



……
……..
// Продължава повтарянето на операциите от предната стъпка докато дървото не остане празно. Това ще означава че всички елементи от списъка са подредени.
……
{32, 34, 34, 43, 47, 58, 59, 61, 72, 77, 79, 84}

ПОЯСНЕНИЕ

1. ***Бързо сортиране:***

За разделящ елемент е избран първия елемент от списъка (със стойност 47). Обхожда се списъка като от дясно на ляво се пропускат всички по – големи елементи чиито стойности са по големи от 47. При срещането на първия по – малък елемент от стойността на разделящия, то те си разменят местата. Така в началото на втора стъпка вече имаме два или повече елемента които се намират от правилната страна спрямо разделящия елемент. В тази стъпка се обхожда списъка от ляво на дясно като се пропускат всички елементи чиито стойности са по малки от 47. При срещането на по – голям елемент то той си разменя мястото със разделящия. В случая това е елемента 59. В трета стъпка, отново списъка се обхожда от дясно наляво, като се пропускат всички по големи елементи и се търси по – малък елемент от стойността на разделящия елемент с който да си разменят местата. Тези обхождания продължават да се редуват, от дясно на ляво и от ляво на дясно, докато всички елементи чиито стойности са по малки от разделящия не се отзоват от ляво на него и съответно елементите със стойности по големи да са от дясно. За конкретния списък в стъпка 4-ри вече се е получило това разделяне. По този начин елементът със стойност 47 си е на мястото и списъка е разделен на два подсписъка. Продължава повтарянето на същите стъпки отначало върху двата под списъка. Тези стъпки се повтарят докато не се получат списъци с по един елемент. В този момент елементите ще са сортирани поради факта че сам елемент в един списък си е сортиран сам по себе си.

2. ***Сортиране чрез сливане***

В първа стъпка списъка се разделя на два подсписъка с еднаква дължина. Върху ново-получените подсписъци продължава да се прилага същото разделяне. Това се повтаря за всеки подсписък докато не се достигне до списък чиито брой елементи е един. Това се вижда на стъпка 5. Следва обединяване на всеки два съседни списъка чиито първи елементи се сравняват. По малкия елемент се премахва от конкретния подсписък и се добавя в нов. Останалите първи елементи от двата обединяващи подсписъци продължават да се сравняват и по малкия отново се добавя към новия списък на следващата свободна позиция. Това се повтаря докато елементите от двата обединяващи подсписъка се обединят. Това е отразено в стъпка 6, където в 6.1 , 6.2 и 6.3 е демонстрирано отделните подсписъци как се обединяват. Така в следващата стъпка ще са се получили на половина по малко на брой списъци чиито елементи са сортирани. Обединяване продължава по същия начин докато накрая не се получи само един списък. Това ще означава че елементите в този списък ще са сортирани.

3. Пирамидално сортиране

В този метод на сортиране се създава едно двоично дърво, в което трябва да се спазва образуването на пирамида. Това ще рече че всеки родител в дървото трябва да има по голяма стойност от сина си. В стъпка 1 и 2 е показано как се образува двоичното дърво, като се спазва това условие. В стъпка 3 е показано дървото готово. В стъпка 4 елементите се добавят в празен масив следвайки структурата на дървото. Спазвайки условието за пирамида на дървото се гарантира, че върха на му съдържа елемента, чиято стойност е най – голяма. В тази стъпка елементите намиращи се на върха на дървото и на последното листо си разменят местата, след което това последно листо се премахва от дървото (в този момент то съдържа елемента с най – голяма стойност). Разменящите се елементи на дървото разменят своите места и в списъка. По този начин най – големия елемент вече се намира на позицията която трябва да заема в сортирания списък. В стъпка 5 е демонстрирано как дървото придобива отново вид на пирамида, като не трябва да се забравя че при всяка размяна на елементи на дървото, същите си разменят местата и в списъка. След като пирамидата е наредена отново върха и последното листо си разменят местата (заедно и същите елементи в списъка) и последното листо се премахва от дървото. Тези стъпки, за пренареждане на елементите на дървото да образува пирамида и пращането върха на дървото на мястото на последното листо, заедно с премахването му, се повтарят докато дървото остане без елементи. Това ще означава че алгоритъмът приключва своята работа и списъка е сортиран.

III. Задача за самостоятелна работа.

1. Демонстрирайте работата на горните алгоритми за следния масив:

   {23, 18, 77, 32, 80, 18, 92, 43, 86, 69, 29, 41, 22, 95, 19, 90}
2. Напишете програма сортираща масив с 1000 елемента по методите QuickSort, MergeSort и HeapSort, като може да си помогнете с функциите от лекциите.
3. Сравнете алгоритмите по бързодействие. Тествайте работата на алгоритмите при сортиран и обратно сортиран масив. Анализирайте резултатите.