ISSN 3060-4567

BUCKET SORT ALGORITMI

Onarqulov Maqsadjon Karimberdiyevich
Farg'ona davlat universiteti amaliy matematika va informatika kafedrasi dotsenti (PhD)
maxmaqsad@gmail.com
Abdulhafizov Ibrohim Husanjon o'g'li
Farg'ona davlat universiteti 2-kurs talabasi
ibrohimabdulhafizov33@gmail.com

Annotatsiya: *Ushbu maqola Bucket Sort (chelakli saralash) algoritmi haqida batafsil ma'lumot berishga qaratilgan. Bucket Sort - bu massiv elementlarini bir nechta bucket (chelak) yoki guruhlarga taqsimlash orqali ishlaydigan saralash algoritmidir. Har bir chelak alohida saralanadi, so'ngra chelaklardagi saralangan elementlar birlashtirilib, yakuniy tartiblangan massiv hosil qilinadi. Maqolada algoritmning asosiy tamoyillari, ishlash prinsipi, murakkablik tahlili, afzalliklari, kamchiliklari va amaliy qo'llanilish sohalari ko'rib chiqiladi. Shuningdek, algoritmning C# dasturlash tilidagi amaliy namunasi keltiriladi.*

Kalit so'zlar: *bucket sort, saralash algoritmi, taqsimlash orqali saralash, chiziqli vaqt murakkabligi, ma'lumotlar tuzilmalari, algoritm murakkabligi, bin sort, uniform taqsimot, vaqt murakkabligi, xotira murakkabligi.*

Аннотация: *Целью этой статьи является предоставление подробной информации об алгоритме сортировки сегментов. Bucket Sort — это алгоритм сортировки, который работает путем разделения элементов массива на несколько сегментов или групп. Каждый сегмент сортируется отдельно, затем отсортированные элементы в сегментах объединяются для формирования окончательного отсортированного массива. В статье рассмотрены основные принципы алгоритма, принцип работы, анализ сложности, преимущества, недостатки и области практического применения. Также представлен практический пример алгоритма на языке программирования C#.*

ISSN 3060-4567

Ключевые слова: *сегментная сортировка, алгоритм сортировки, сортировка по распределению, линейная временная сложность, структуры данных, сложность алгоритма, сортировка ячеек, равномерное распределение, временная сложность, сложность памяти.*

Abstract: *This article examines medical information about the Bucket Sorting Algorithm. Bucket Sort is an algorithmic control of array validation by partitioning it into multiple buckets or controls. Sorted by each bucket, combined to form a sorted array of buckets. The article considers the basics of the algorithm, the principle of operation, analysis of complexity, errors and areas of practical application. The implementation of the algorithm in the C# programming language is also in place.*

Key words: *bucket sort, sorting algorithm, sorting by distribution, linear time complexity, data structures, algorithm complexity, bin sort, uniform distribution, time complexity, memory complexity.*

Kirish

Bucket Sort (yoki Bin Sort, Chelakli Saralash) - bu massiv elementlarini bir nechta “chelak” (inglizcha: bucket) yoki “quti” (inglizcha: bin) deb ataladigan guruhlarga taqsimlash orqali ishlaydigan saralash algoritmidir. Bu algoritm taqsimlash orqali saralash (distribution sort) turkumiga kiradi va Pigeonhole Sort (kaptar uyasi saralashi) algoritmining umumlashtirilgan shakli hisoblanadi, chunki u har bir chelakka bir nechta kalit (element) tushishiga imkon beradi. Shuningdek, u Radix Sort (razryadli saralash) algoritmining eng yuqori razryaddan boshlab saralash turiga yaqin hisoblanadi.

Bucket Sort algoritmining asosiy g’oyasi shundan iboratki, agar kiruvchi ma’lumotlar ma’lum bir diapazonda bir tekis taqsimlangan bo’lsa, saralash jarayonini tezlashtirish mumkin. Algoritm elementlarni qiymatlari bo’yicha oldindan belgilangan diapazonlarga mos keladigan chelaklarga joylashtiradi. Har bir chelak ichidagi elementlar alohida saralanadi. Buning uchun boshqa bir saralash algoritmi (masalan, Insertion Sort - qo’shish orqali saralash) ishlatilishi yoki Bucket Sort algoritmining o’zi

ISSN 3060-4567

rekursiv tarzda qo'llanilishi mumkin. Nihoyat, barcha chelaklardagi saralangan elementlar ketma-ketlikda birlashtirilib, yakuniy tartiblangan massiv hosil qilinadi.

Bucket Sort taqqoslashga asoslangan saralash algoritmi sifatida ham qaralishi mumkin, chunki chelaklar ichidagi elementlarni saralash uchun ko'pincha taqqoslashga asoslangan algoritmlar (masalan, Insertion Sort, Merge Sort) ishlatiladi. Algoritmning samaradorligi (hisoblash murakkabligi) har bir chelakni saralash uchun ishlatiladigan algoritmga, ishlatiladigan chelaklar soniga va kiruvchi ma'lumotlarning qanchalik bir tekis taqsimlanganligiga bog'liq. ## Algoritmning qanday ishlashi

Bucket Sort algoritmining ishlash jarayoni odatda quyidagi qadamlardan iborat bo'ladi:

1. **Chelaklarni yaratish:** Dastlab, ma'lum bir `k` sonda bo'sh chelaklar (odatda ro'yxatlar yoki massivlar ro'yxati ko'rinishida) yaratiladi. Chelaklar soni `k` odatda kiruvchi massivdagi elementlar soni `n` ga teng yoki yaqin olinadi, ammo bu qiymat muammoning xususiyatiga qarab tanlanishi mumkin. Chelaklar kiruvchi ma'lumotlar diapazonini qamrab olishi kerak. Masalan, agar elementlar 0 dan 1 gacha bo'lgan haqiqiy sonlar bo'lsa, 10 ta chelak yaratilishi mumkin, har biri 0.1 uzunlikdagi intervalni (masalan, [0, 0.1), [0.1, 0.2), …, [0.9, 1.0)) ifodalaydi.

2. **Elementlarni taqsimlash (Scatter):** Kiruvchi massivdagi har bir element `arr[i]` olinadi va uning qiymatiga mos keladigan chelakka joylashtiriladi. Element qaysi chelakka tushishini aniqlash uchun odatda maxsus formula ishlatiladi. Masalan, 0 dan `M` gacha bo'lgan qiymatlar uchun `k` ta chelak bo'lsa, `arr[i]` elementi `floor(k * arr[i] / M)` indeksli chelakka joylashtirilishi mumkin (agar `M` maksimal qiymatdan biroz katta olingan bo'lsa). Agar elementlar 0 dan 1 gacha bo'lsa, `floor(k * arr[i])` formulasi ishlatilishi mumkin. Har bir element o'ziga tegishli chelakka qo'shiladi.

3. **Har bir chelakni saralash:** Barcha elementlar chelaklarga taqsimlangandan so'ng, har bir bo'sh bo'lmagan chelak ichidagi elementlar alohida saralanadi. Bu saralash uchun odatda Insertion Sort kabi oddiy va kichik hajmdagi

ma’lumotlar uchun samarali bo’lgan algoritm ishlatiladi. Chunki agar kiruvchi ma’lumotlar bir tekis taqsimlangan bo’lsa, har bir chelakdagi elementlar soni kam bo’lishi kutiladi. Biroq, boshqa saralash algoritmlari (Merge Sort, Quick Sort) yoki hatto Bucket Sortning o’zi rekursiv ravishda ishlatilishi ham mumkin.

4. **Chelaklarni birlashtirish (Gather):** Nihoyat, barcha chelaklardagi saralangan elementlar tartib bo’yicha (birinchi chelakdan boshlab oxirgisigacha) ketma-ket olinib, asl massivga yoki yangi natijaviy massivga yig’iladi. Bu jarayon natijasida to’liq saralangan massiv hosil bo’ladi. ## Vaqt va Xotira Murakkabliklari

Bucket Sort algoritmining samaradorligi kiruvchi ma’lumotlarning taqsimlanishiga va har bir chelakni saralash uchun ishlatiladigan algoritmga bog’liq.

Vaqt Murakkabligi:

1. **Eng Yaxshi Holat (Best Case):** O(n + k). Bu holat elementlar chelaklar bo’ylab bir tekis taqsimlanganda yuzaga keladi. Har bir chelakda taxminan bir xil va oz sonli elementlar bo’ladi. Agar chelaklarni saralash uchun Insertion Sort kabi chiziqli vaqtda ishlaydigan (kichik yoki deyarli saralangan ma’lumotlar uchun) algoritm ishlatilsa va chelaklar soni `k` massiv elementlari soni `n` ga proporsional bo’lsa (k ≈ n), umumiy vaqt murakkabligi O(n) ga yaqinlashadi. `O(n)` vaqt elementlarni chelaklarga taqsimlash uchun, `O(k)` vaqt esa (har bir chelak uchun o’rtacha O(1) yoki kichik konstant vaqt sarflansa) chelaklarni saralash va birlashtirish uchun sarflanadi.

2. **O’rtacha Holat (Average Case):** O(n + k). Agar kiruvchi ma’lumotlar tasodifiy va bir tekis taqsimlangan deb faraz qilinsa, Bucket Sort o’rtacha hisobda chiziqli vaqtda ishlaydi. Elementlar chelaklarga notekis taqsimlangan bo’lsa ham, agar chelaklardagi elementlar soni kvadratlarining yig’indisi umumiy elementlar soniga nisbatan chiziqli bo’lib qolsa, algoritm chiziqli vaqtda ishlashda davom etadi.

3. **Eng Yomon Holat (Worst Case):** $O(n^2)$. Eng yomon holat barcha (yoki deyarli barcha) elementlar bitta chelakka tushganda yuzaga keladi. Bu holatda algoritmning samaradorligi asosan shu bitta chelakni saralash uchun ishlatiladigan algoritmning samaradorligiga bog’liq bo’lib qoladi. Agar chelaklarni saralash uchun

ISSN 3060-4567

Insertion Sort yoki Bubble Sort kabi $O(n^2)$ murakkablikka ega algoritm ishlatilsa, Bucket Sortning umumiy eng yomon vaqt murakkabligi ham $O(n^2)$ bo'ladi. Agar chelaklarni saralash uchun Merge Sort yoki Heap Sort kabi $O(n*\log_2 n)$ murakkablikka ega algoritm ishlatilsa, eng yomon holat murakkabligi $O(n*\log_2 n)$ gacha yaxshilanishi mumkin.

Xotira Murakkabligi:

Bucket Sort algoritmi qo'shimcha xotira talab qiladi. Xotira murakkabligi **O(n + k)** dir. Bu yerda `O(k)` xotira `k` ta chelakni saqlash uchun, `O(n)` xotira esa eng yomon holatda barcha `n` element bitta chelakka tushganda yoki chelaklar ichida elementlarni saqlash uchun (masalan, bog'langan ro'yxatlar ishlatilganda) kerak bo'ladi.

Afzalliklari va Kamchiliklari:

Bucket Sort algoritmining o'ziga xos afzalliklari va kamchiliklari mavjud.

Afzalliklari:

1. **Yuqori Tezlik (O'rtacha Holatda):** Agar kiruvchi ma'lumotlar bir tekis taqsimlangan bo'lsa, Bucket Sort o'rtacha hisobda juda tez ishlaydi, vaqt murakkabligi $O(n + k)$ ga teng bo'ladi. Agar `k` `n` ga proporsional bo'lsa ($k = n$), bu deyarli chiziqli $O(n)$ vaqtni anglatadi. Bu uni Quick Sort, Merge Sort kabi $O(n \log_2 n)$ murakkablikdagi algoritmlardan tezroq qiladi.
2. **Tashqi Saralash uchun Yaroqlilik:** Bucket Sort katta hajmdagi ma'lumotlar to'plamini saralash uchun mos kelishi mumkin, chunki har bir chelak alohida qayta ishlanishi va saralanishi mumkin, bu esa xotiradan tashqarida joylashgan ma'lumotlar bilan ishlashga imkon beradi.
3. **Parallelizatsiya:** Chelaklarni saralash jarayoni osongina parallellashtirilishi mumkin, chunki har bir chelak mustaqil ravishda saralanadi. Bu ko'p yadroli protsessorlarda yoki taqsimlangan tizimlarda samaradorlikni oshirishi mumkin.

Kamchiliklari:

1. **Ma’lumotlarning Taqsimlanishiga Bog’liqlik:** Algoritmning samaradorligi kiruvchi ma’lumotlarning qanchalik bir tekis taqsimlanganligiga juda bog’liq. Agar ma’lumotlar notekis taqsimlangan bo’lsa va ko’pchilik elementlar bitta yoki bir nechta chelakka to’planib qolsa, algoritmning ishlash vaqti keskin yomonlashadi va eng yomon holatda $O(n^2)$ yoki O(n log n) ga yetishi mumkin (chelaklarni saralash uchun ishlatilgan algoritmga qarab).

2. **Qo’shimcha Xotira Talabi:** Bucket Sort chelaklarni va ularning ichidagi elementlarni saqlash uchun qo’shimcha xotira (O(n + k)) talab qiladi. Bu xotira sarfi ba’zi hollarda, ayniqsa xotira cheklangan muhitlarda muammo tug’dirishi mumkin.

3. **Chelak Sonini Tanlash:** Optimal chelaklar sonini k tanlash har doim ham oson emas va kiruvchi ma’lumotlarning xususiyatlariga bog’liq. Noto’g’ri tanlangan k qiymati samaradorlikka salbiy ta’sir ko’rsatishi mumkin.

4. **Faqat Muayyan Ma’lumot Turlariga Mosligi:** Asosiy Bucket Sort algoritmi odatda bir tekis taqsimlanishi mumkin bo’lgan raqamli ma’lumotlar (masalan, 0 dan 1 gacha bo’lgan haqiqiy sonlar yoki ma’lum diapazondagi butun sonlar) uchun mo’ljallangan. Boshqa turdagi ma’lumotlar (masalan, satrlar) uchun qo’shimcha ishlov berish (masalan, xeshlash) talab qilinishi mumkin.

C# tilidagi dasturi:

Quyida Bucket Sort algoritmining C# dasturlash tilida yozilgan namunasi keltirilgan. Ushbu kod massivdagi 0 dan 1 gacha bo’lgan haqiqiy sonlarni saralash uchun mo’ljallangan. Kodda har bir chelakni saralash uchun Insertion Sort algoritmidan foydalanilgan.

```
using System;
using System.Collections.Generic;

class BucketSortAlgorithm
{
   static void InsertionSort(List<int> bucket)
   {
```

ISSN 3060-4567

```
    for (int i = 1; i < bucket.Count; ++i)
    {
        int key = bucket[i];
        int j = i - 1;
        while (j >= 0 && bucket[j] > key)
        {
            bucket[j + 1] = bucket[j];
            j--;
        }
        bucket[j + 1] = key;
    }
}
public static void Sort(int[] arr)
{
    int n = arr.Length;
    if (n <= 0) return;
    List<int>[] buckets = new List<int>[n];
    for (int i = 0; i < n; i++)
    {
        buckets[i] = new List<int>();
    }
    for (int i = 0; i < n; i++)
    {
        int bucketIndex = (int)(n * arr[i]);
        if (bucketIndex >= n) bucketIndex = n - 1;
        if (bucketIndex < 0) bucketIndex = 0;
        buckets[bucketIndex].Add(arr[i]);
    }
    for (int i = 0; i < n; i++)
    {
```

ISSN 3060-4567

```
            InsertionSort(buckets[i]);
        }
        int index = 0;
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < buckets[i].Count; j++)
            {
                arr[index++] = buckets[i][j];
            }
        }
    }
    public static void Main(string[] args)
    {
        int[] arr = {7, 43, 36, 21, 44, 99};

        Console.WriteLine("Boshlang'ich massiv:");
        foreach (int num in arr)
        {
            Console.Write(num + " ");
        }
        Console.WriteLine("\n");

        Sort(arr);

        Console.WriteLine("Saralangan massiv:");
        foreach (int num in arr)
        {
            Console.Write(num + " ");
        }
        Console.WriteLine();
```

```
        Console.ReadKey();
    }
}
```

Qo’llanish sohalari

Bucket Sort algoritmi, ayniqsa, ma’lum shartlar bajarilganda samarali bo’lgani uchun o’ziga xos qo’llanilish sohalariga ega:

1. **Bir Tekis Taqsimlangan Ma’lumotlar:** Algoritmning eng kuchli tomoni uning bir tekis taqsimlangan ma’lumotlar bilan ishlashdagi yuqori samaradorligidir. Agar kiruvchi ma’lumotlar (masalan, 0 dan 1 gacha bo’lgan haqiqiy sonlar yoki ma’lum bir diapazondagi butun sonlar) taxminan bir tekis taqsimlangan bo’lsa, Bucket Sort chiziqli vaqtda (O(n)) ishlashi mumkin. Bu uni moliyaviy ma’lumotlar, statistik tahlillar yoki simulyatsiyalardan olingan natijalar kabi sohalarda qo’llash uchun qulay qiladi.

2. **Haqiqiy Sonlarni Saralash:** Bucket Sort ko’pincha katta hajmdagi haqiqiy (suzuvchi nuqtali) sonlarni saralash uchun ishlatiladi, ayniqsa ular ma’lum bir cheklangan diapazonda joylashgan bo’lsa. Chelaklar ushbu diapazonni kichikroq intervallarga bo’lish uchun ishlatiladi.

3. **Tashqi Saralash (External Sorting):** Katta hajmdagi ma’lumotlar to’plami asosiy xotiraga sig’maganda, Bucket Sort tashqi saralash uchun asos bo’lishi mumkin. Ma’lumotlar avval chelaklarga taqsimlanib, har bir chelak alohida fayl sifatida saqlanishi mumkin. Keyin har bir fayl (chelak) alohida saralanib, natijalar birlashtiriladi.

4. **Ma’lumotlar Bazalarida Indekslash:** Ba’zi ma’lumotlar bazasi tizimlarida yoki ma’lumotlarni qayta ishlash jarayonlarida ma’lumotlarni taxminiy joylashuviga qarab guruhlash uchun Bucket Sortga o’xshash algoritmlardan foydalanish mumkin.

5. **Grafika va Geometriya:** Kompyuter grafikasida yoki hisoblash geometriyasida nuqtalarni yoki obyektlarni koordinatalari bo’yicha guruhlash yoki saralash uchun Bucket Sort tamoyillari qo’llanilishi mumkin.

ISSN 3060-4567

Xolosa

Shuni ta'kidlash kerakki, Bucket Sortning samaradorligi ma'lumotlarning taqsimlanishiga bog'liq bo'lgani uchun u har qanday vaziyat uchun universal yechim emas. Ammo yuqorida sanab o'tilgan shartlar bajarilganda, u juda tez va samarali saralash usuli bo'la oladi.

FOYDALANILGAN ADABIYOTLAR:

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein - Introduction to Algorithms (CLRS)
2. Steven S. Skiena - The Algorithm Design Manual
3. Robert Sedgewick, Kevin Wayne - Algorithms
4. Aditya Y. Bhargava - Grokking Algorithms: An Illustrated Guide for Programmers and Other Curious People
5. Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser - Data Structures and Algorithms in Java (yoki C++/Python versiyalari)
6. **"Algoritmlar va ma'lumotlar tuzilmalari"** – D. Axmedov, TATU Darslik.
7. **"Dasturlash asoslari (C# dasturlash tili misolida)"** – B. Karimov, Toshkent 2021
8. **"Algoritmlarni loyihalash va tahlil qilish"** – M. Jo'rayev, Farg'ona Politexnika Instituti
9. **"Diskret matematika va algoritmik tafakkur"** – T. Yo'ldoshev
10. **"Kompyuter fanlariga kirish"** – U. Qosimov, TATU
11. **"Ma'lumotlar tuzilmasi va algoritmlar (praktikum)"** – Sh. Normuradov, Samarqand
12. **"Kompyuter dasturlashining nazariy asoslari"** – B. Ergashev
13. Medium (Jo'rayev Zarif): https://medium.com/@zarifjorayev/bucket-sort-93c5b4486926
14. Wikipedia (English): https://en.wikipedia.org/wiki/Bucket_sort
15. GeeksforGeeks: https://www.geeksforgeeks.org/bucket-sort-2/