TÜRKİYE CUMHURİYETİ
YILDIZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

GNOME SORT ALGORITHM (CÜCE SIRAMALA ALGORİTMASI)

16011706 – DUYGU ERDURAN

Yapısal Programlamaya Giriş Dersi 2019/2 Dönemi
FİNAL PROJESİ

Öğretim Üyesi
Dr. Öğr.Üyesi Hafiza İrem TÜRKMEN ÇİLİNGİR

Haziran,2020

İÇİNDEKİLER

Gnome Sort (Cüce Sıralama) Algoritmasının Tarifi

Aptal sıralama olarak adlandırılan Gnome sort Algoritması, 2000 yılında İranlı bir bilgisayar bilimcisi Hamid Sarbazi-Azad (Sharif Teknoloji Üniversitesi Bilgisayar Bilimi ve Mühendisliği profesörü) tarafından önerilen bir sıralama algoritmasıdır.

Algoritma ilk olarak aptal sıralama olarak adlandırıldı. (Bogosort ile karıştırılmamalıdır.) ve daha sonra Dick Grune tarafından tanımlanarak gnome sort olarak adlandırılmıştır. Algoritmanın amacı, dizideki sayıları sıralayarak, doğru aralığa taşımayı amaçlar.

Cüce Sıralaması adı algoritma yönteminin mitolojideki Hollanda cücelerinin (gnome) bir dizi çiçek saksısını sıraya diziş biçimine benzemesinden kaynaklanmaktadır. Dick Grune bu sıralamayı şöyle anlatmaktadır:

" Bir bahçe cücesi, saksıları aşağıdaki yöntemle sıralar:

1. Saksı hemen önce ve sonra doğru sıradaysa, bir adım ileri gider.
2. Doğru sırada değilse, saksıları değiştirir ve bir adım geri gider.
3. Ondan önce pot olmadığında başlangıçta öne doğru ilerler ve pot çizgisinin sonuna ulaştığında liste sıralanır. "

Algoritmanın Çalışma Prensibi

Bir Gnome Sıralaması yaparken akılda tutulması gereken ilk ilke, algoritma ilerledikçe, karşılaştırılan sayının solundaki tüm sayıların her zaman sıralı olmasıdır. Algoritmanın her yinelemesinde, karşılaştırılan sayı önceden sıralanan sayılara bakılarak doğru noktaya taşınır.

Yanyana duran ve sıralama kriterini bozan bir ikili bulana kadar dizide hareket edilir.Bu ikili bulununca düzeltilir ve dengeye ulaşana kadar gerekirse dizinin başına kadar geri dönülür. Ardından işlem kaldığı yerden devam eder. Dizi zaten sıralanmışsa, döngü her öğeye yalnızca bir kez bakarak çalışır. Sıralanmamışsa, her bir öğeye bakarak ve bir adım ileri veya geri hareket ederek normal prosedürü çalıştırır

Uygulama Alanları

Sıralama algoritmalarının çalışma hızları, sıralanacak verinin büyüklüğü, kısmen sıralı olması, tersten sıralı olması veya tümüyle karışık yapıda olmasına bağlı olarak farklılık göstermektedir. Seçim yapılırken verinin bu anlamda değerlendirilmesi ve uygun algoritmanın tercihi en iyi sonuç için faydalı olacaktır. Kullanılacak sıralama algoritması seçilirken uygulanacak veri yapısı detaylı olarak ele alınmalıdır.

Gnome sort küçük veri gruplarında kullanılır. Veri seti büyüdükçe algoritma verimsizleşir. Bu sebeple Gnome Sort'un hakim olduğu yer öğretmedir. Örnek vererek açıklarsak; algoritma mantığını anlamak veya anlatmada, herhangi bir programlama dillerini öğrenmek yada öğretmek

için en elverişli algoritmadır. Algoritmanın anlaşılması ve uygulanması diğer sıralama algoritmalarından basittir.

Algoritmanın Karmaşıklık Analizi

En İyi Durum (Best-Case): O (n)
Ortalama Durum (Average-Case): O (n^2)
En Kötü Durum (Worst-Case): O(n^2)
Memory: 1
Stable: var

Bu, kötü performans gösteren bir başka sıralama algoritması olan Bubble Sort ile aynı en kötü durumdur. Gnome sort iyileştirilemedi.

Algoritmanın Kısıtları:

Büyük veri setlerini sıralama da çalışma süresi katlanarak artar ve bu da onu verimsiz hale getirir. Bu yüzden kullanışlı değildir.

İyileştirme:

1. Sıralama yapılacak veri gruplarına göre başka efektif bir sıralama algoritması seçilebilir. Örneğin; Büyük veri gruplarını hızlı sıralamak için Merge Sort yada Quick Sort tercih edilebilir.

2. Yazdığınız kodu daha temiz kod yapamaya çalışarak bir miktarda olsa efektif hale getirebiliriz. Tabi bu algoritmanın dezavantajını ortadan kaldırmaz. Sadece öğrenmek veya öğretmek amaçlı faydalı olur. Örnek olsun diye aşağıda verilen C koduna iyileştirme yapılmıştır.

Algoritmanın Avantajları ve Dezavantajları

Gnome Sort düşük performanslı bir algoritmadır ancak değeri performansta değil öğretimde yatmaktadır. Algoritmanın anlaşılması ve uygulanması diğer sıralama algoritmalarından Kavramsal olarak basittir, iç içe döngü gerektirmez. Bu sebeple öğrenmek ve öğretmek amacıyla çok elverişli bir algoritmadır. En basit sıralama algoritması Bubble Sort veya Insertion Sort'a değil, Gnome Sort'tur.

Az miktarda veri için en basit sıralama algoritmasıdır. Daha büyük veri grupları için, çalışma süresi katlanarak artar ve bu da onu verimsiz hale getirir. Kullanışlı değildir. Tercih edilmez.

Diğer Sıralama Algoritmaları ile Karşılaştırılması

Stable: Sıralama yaparken bir adım önceki sırayı bozmamasıdır.

Gnome Sort, araya sokmalı sıralamaya benzer bir sıralama algoritmasıdır. Ara sokmalı sıralamadan farkı kabarcık sıralaması yönteminde olduğu gibi, bir elemanın sıralanan dizideki yerine birçok yer değiştirme yoluyla gelmesidir.
Kabarcık sıralaması iç içe döngüde gerçekleştirilir, ancak gnome sıralaması tek bir döngüde gerçekleştirilir. Dahası, kabarcık sıralaması bitişik elemanları liste boyunca ardışık geçişlerde karşılaştırırken, gnome sıralaması bitişik elemanları geriye doğru karşılaştırır ve indeksi ileri geri hareket ettirir.
Duck Grune dediği gibi:
“ En basit sıralama algoritması ne Insertion Sort’tur ne de Bubble Sort, Gnome Sort’tur..

Comparison sorts

Name	Best	Average	Worst	Memory	Stable	Method	Other notes
Quicksort	$n \log n$	$n \log n$	n^2	$\log n$	No	Partitioning	Quicksort is usually done in-place with $O(\log n)$ stack space.[5][6]
Merge sort	$n \log n$	$n \log n$	$n \log n$	n	Yes	Merging	Highly parallelizable (up to $O(\log n)$ using the Three Hungarians' Algorithm).[7]
In-place merge sort	—	—	$n \log^2 n$	1	Yes	Merging	Can be implemented as a stable sort based on stable in-place merging.[8]
Introsort	$n \log n$	$n \log n$	$n \log n$	$\log n$	No	Partitioning & Selection	Used in several STL implementations.
Heapsort	$n \log n$	$n \log n$	$n \log n$	1	No	Selection	
Insertion sort	n	n^2	n^2	1	Yes	Insertion	$O(n + d)$, in the worst case over sequences that have d inversions.
Block sort	n	$n \log n$	$n \log n$	1	Yes	Insertion & Merging	Combine a block-based $O(n)$ in-place merge algorithm[9] with a bottom-up merge sort.
Quadsort	n	$n \log n$	$n \log n$	n	Yes	Merging	Uses a 4-input sorting network.[10]
Timsort	n	$n \log n$	$n \log n$	n	Yes	Insertion & Merging	Makes n comparisons when the data is already sorted or reverse sorted.
Selection sort	n^2	n^2	n^2	1	No	Selection	Stable with $O(n)$ extra space or when using linked lists.[11]
Cubesort	n	$n \log n$	$n \log n$	n	Yes	Insertion	Makes n comparisons when the data is already sorted or reverse sorted.
Shellsort	$n \log n$	$n^{4/3}$	$n^{3/2}$	1	No	Insertion	Small code size.
Bubble sort	n	n^2	n^2	1	Yes	Exchanging	Tiny code size.
Tree sort	$n \log n$	$n \log n$	$n \log n$(balanced)	n	Yes	Insertion	When using a self-balancing binary search tree.
Cycle sort	n^2	n^2	n^2	1	No	Insertion	In-place with theoretically optimal number of writes.
Library sort	n	$n \log n$	n^2	n	Yes	Insertion	
Patience sorting	n	—	$n \log n$	n	No	Insertion & Selection	Finds all the longest increasing subsequences in $O(n \log n)$.
Smoothsort	n	$n \log n$	$n \log n$	1	No	Selection	An adaptive variant of heapsort based upon the Leonardo sequence rather than a traditional binary heap.
Strand sort	n	n^2	n^2	n	Yes	Selection	
Tournament sort	$n \log n$	$n \log n$	$n \log n$	n[12]	No	Selection	Variation of Heap Sort.
Cocktail shaker sort	n	n^2	n^2	1	Yes	Exchanging	
Comb sort	$n \log n$	n^2	n^2	1	No	Exchanging	Faster than bubble sort on average.
Gnome sort	n	n^2	n^2	1	Yes	Exchanging	Tiny code size.
UnShuffle Sort[13]	n	kn	kn	n	No	Distribution and Merge	No exchanges are performed. The parameter k is proportional to the entropy in the input. $k = 1$ for ordered or reverse ordered input.
Franceschini's method[14]	—	$n \log n$	$n \log n$	1	Yes	?	
Odd–even sort	n	n^2	n^2	1	Yes	Exchanging	Can be run on parallel processors easily.

Algoritmanın C dili ile yazılması

```
//Tumlesik kutuphaneler
#include <stdio.h>
#include <stdlib.h>
#include <time.h>  //clock fonksiyonunu kullanmamizi saglar
#include <locale.h>
#include <windows.h>
#define dMAX 10000
#define tmax 10

// Global Degisken tanimlari
int data_number[tmax]; // girilen sayi adedini tutuyor.
int times[tmax];
int iter = 0; //kacinci aramada oldugunun bilgisi

//Fonksiyon tanimlari
void gnomesort();
void printHistogram();

void gnomesort()
   {
      int i,k, temp, array[dMAX],n;
      printf("\t    Siralanacak sayi adedini giriniz:");
      scanf("%d", &n);
      printf("\t    Siralanacak sayilari bosluk birakarak giriniz. :\n");
      printf("\t    ");
      for (i = 0; i < n; i++)
         scanf(" %d", &array[i]);
      i = 0;
      clock_t baslangic = clock(); // gecen süreyi hesaplamak icin islem saatinin baslatiriz.
      while (i < n)  // Gnome Sort siralama yapilmasi
      {
         if (i == 0 || array[i - 1] <= array[i])
            i++;
         else
         {
            temp = array[i-1];
            array[i - 1] = array[i];
            array[i] = temp;
            i = i - 1;
         }
      }printf(" \t    Gnome Sorted :");
      for (i = 0;i < n;i++)
         printf("%d  ", array[i]); //siralanmis diziyi ekrana yazdirdik
      clock_t bitis= clock(); // islem saatini bitiririz.
      float saniye= ((double)(bitis - baslangic) / CLOCKS_PER_SEC); // Saniyeyi hesaplar degiskene atariz
      int ms= saniye *1000; // histogram için sadelestirme yaptik (optimizasyon). Milisaniyeyi bulup
degiskene atadik
      printf(" \n\t    %d milisaniyede islem yapýldý.\n", ms); // milisaniyeyi yazdiririz.
                  data_number[iter] = n; // tutulacak sayi adetini diziye attik
                  times[iter] = ms; // hesaplanan milisaniyeye diziye atildi.
                  iter++; // devamliligi sagladik
                  //Siralanan sayi adetlerini ve Siralanan dizinin sürelerini data_number ve times dizilerinden
tuttuk ve her islemde yazdirdik.
```

```
                for(k = 0; k < iter; k++){
                        printf("\n\t    Siralanan sayi adedi [%d] = %d\n", k, data_number[k]);
                        printf("\t    Siralanan dizinin süresi [%d] = %d\n", k, times[k]);
                }
   }

void printHistogram() {
        int k,n,t;
   printf("\n");
        printf(" %18s%20s%18s    \n\n", "  \t    Veri adeti (N)   "," \t Çalisma Süresi (MS) ","  \t
Histogram "); // histogram için kullanacagimiz veriler
        int i ;
        for(k=0;k<iter;k++){ //verileri ve  sürelerini sirayla diziden yazdirdik
                printf(" %20u%20d   \t     ---->      ",data_number[k],times[k]);
                //birinci arama isleminin sonucunu digerlerine oranla  *** kodun iyilestirilmesi***
                //(en küçük olan veriyi buluyorum, 1 yildiz veriyorum, digerlerini ona oranliyorum)
          int smallest = times[0];

           for ( i = 1; i < iter; i++) {
             if (times[i] < smallest) {
               smallest = times[i];
             }
           }
                int divide_result = times[k] / smallest;

                for(t=0; t < /* times[k] */ divide_result ; t++){
                        printf("%c",'*' );
                }
                        printf("\n");
        }
}

int main() {

        setlocale(LC_ALL, "Turkish");  //Turkcelestirme
   system(" color 0F "); // Tema renk ayari

        //Degisken tanimlari
        int islem = 0;
        char tekrar = 'e';
        while (tekrar == 'e') {
                //Islem secimi
                printf(" \n\n      ~~ YAPISAL PROGRAMLAMAYA GÝRÝÞ FÝNAL PROJESÝ --> GNOME
SORT ALGORITHM ~~ \n");
                printf( "\n             ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"
                        "             ~    1- GNOME SORT HESAPLAMA    ~ \n"
                        "             ~                              ~ \n"
                                      "             ~    2- HÝSTOGRAM OLUÞTURMA     ~  \n"
                                      "             ~                              ~ \n"
                                      "             ~    3- PROGRAMDAN ÇIKMA       ~   "
                                      "\n             ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ \n"
                                      "\n\t    Yapmak istediginiz iþlemin numarasýný giriniz: ");
                scanf("%d", &islem);
                switch (islem) {
                case (1):
              gnomesort();
```

```
                    break;
                case (2):
                    printHistogram();
                    break;
                case (3):
                    exit(0);
                    break;
                default:
                    printf("\t    Ýslem seçilmedi.");
                }
                //Tekrarlayim mi?
                tekrar = 'h';
                printf("\n\t    Tekrar islem yapmak istiyor musunuz (e/h)? ");
                scanf(" %c", &tekrar);

                //tekrar karakter kontrolu
                while(tekrar != 'e' && tekrar != 'h'){
                    printf("\"%c\" geçerli deðildir.\n"
                  "\t    Lütfen geçerli karakter giriniz (e/h): ", tekrar);
                    scanf(" %c", &tekrar);
                }
                printf("\n");
        }
        return 0;
}
```

Öneri Linkler:

http://panthema.net/2013/sound-of-sorting/

https://www.youtube.com/watch?v=EdcWAw4Hcu0&feature=emb_title

Kaynakça

URL:

http://www.kodyazan.com/MakaleDetay/1225/Algoritma-Nedir--Analizi-Nasil-Yapilir-

https://www.dickgrune.com/

https://en.wikipedia.org/wiki/Sorting_algorithm#Comparison_of_algorithms

https://stackoverflow.com/questions/2066541/what-is-the-average-big-%CE%9F-complexity-of-gnome-sort

https://en.wikipedia.org/wiki/Gnome_sort

https://exceptionnotfound.net/gnome-sort-csharp-the-sorting-algorithm-family-reunion/

http://sorting.at/

https://rastgele-sayi-hesaplama.hesabet.com/

KİTAP:

- Algoritmalar Dördüncü Basımdan Çeviri, Robert Sedgewick | Kevin Wayne, Çeviri Editörü: Şadi Evren Şeker

Youtube Video Linki: https://youtu.be/1-IgQckZJic