Nama : Muvidha Fatmawati Putri

Nim : 21091397011

Kelas : A

Laporan individu (radix sort)

1. Buat kode c++ untuk setiap insertion sort (radix sort)

    - Fungsi void Radix sort (bagian pertama)

```cpp
//implementasi program C++ Radix sort
#include <iostream>

using namespace std;

// mendapatkan nilai maksimum dari array
int getMax(int arr[], int n)
{
    int max = arr[0];
    for (int i = 1; i < n; i++)
        if (arr[i] > max)
            max = arr[i];
    return max;
}
```

    - Fungsi cetak array (bagian kedua)

```cpp
// mencetak array[].
//penyelesaian dengan menggunakan metode pemanggilan
void countSort(int arr[], int n, int exp)
{
    // Count[i] array akan menghitung jumlah nilai array yang memiliki digit 'i'.
    int output[n], i, count[10] = {0};

    // menghitung berapa kali setiap digit muncul di setiap input.
    for (i = 0; i < n; i++)
        count[(arr[i] / exp) % 10]++;

    // Menghitung jumlah kumulatif array.
    for (i = 1; i < 10; i++)
        count[i] += count[i-1];

    // Menyisipkan nilai sesuai dengan digit.
    for (i = n - 1; i >= 0; i--)
    {
        output[count[(arr[i] / exp) % 10] - 1] = arr[i];
        count[(arr[i] / exp) % 10]--;
    }
    // Menetapkan hasil ke pointer array.
    for (i = 0; i < n; i++)
        arr[i] = output[i];
}
// mengurutkan array dengan variabel n menggunakan Radix Sort.
void radixsort(int arr[], int n)
{
    int exp, m;
    m = getMax(arr, n);
    // Memanggil countSort() untuk digit di setiap input.
    for (exp = 1; m/exp > 0; exp *= 10)
        countSort(arr, n, exp);
}
```

    - Fungsi main output program (bagian ketiga)

```
int main()
{
    // inisialisasi variabel
    int n, i;
    cout<<"Masukkan jumlah data dalam array : ";
    cin>>n;
    cout<<endl;

    cout << "Masukan " << n << " Nilai Array :"<<endl;
    int array[n];
    int arr[n];
    for(i = 0; i < n; i++)
    {
        cin>>arr[i];
    }
    cout<<endl;

    radixsort(arr, n);

    //----Proses sorting berakhir----
    // keluaran data yang sudah menggunakan radixsort
    cout<<"Array yang sudah disorting : \n";
    for(i = 0; i < n; i++)
    {
        cout<<"["<<arr[i]<<"] ";
    }

    return 0;
}
```

- Hasil output program

```
Masukkan jumlah data dalam array : 5

Masukan 5 Nilai Array :
4
20
3
9
13

Array yang sudah disorting :
[3] [4] [9] [13] [20]
--------------------------------
Process exited after 47.85 seconds with return value 0
Press any key to continue . . .
```

A. Pengertian radix sort

Radix Sort merupakan salah satu algoritma Non-Comparasion Sort (pengurutan tanpa pembandingan). Proses yang dilakukan dalam metode ini adalah mengklasifikasikan data array sesuai dengan kategori terurut yang tertentu, dan tiap kategori array dilakukan pengklasifikasian lagi, dan seterusnya sesuai kebutuhan, lalu subkategori-kategori array tersebut digabungkan kembali.

Secara harfiah Radix dapat diartikan sebagai posisi dalam angka, karena metode ini pertama kalinya mengurutkan nilai-nilai input berdasarkan radix pertamanya, lalu pengurutan dilakukan berdasarkan radix keduanya, dan begitu seterusnya.

Untuk kasus bilangan bulat (integer), algoritma ini akan mengurutkan data dengan mengelompokkan data array berdasarkan digit yang memiliki significant position dan value yang sama. Kelompok digit ini ditampung dalam suatu variable “bucket”. Struktur datanya direpresentasikan dengan array. Algoritma ini pertama kali diperkenalkan pada tahun 1887 oleh Herman Hollerith pada mesin tabulasi.

Ada dua cara radix sort saat ini :

- LSD (*least significant digit*) *radix sort*, yaitu *radix sort* yang mengurutkan data dimulai dari digit terkecil. Algoritma ini cenderung stabil karena tetap mengikuti urutan awal data-data sebelum diurutkan.
  Misalkan terdapat sebuah array dengan elemen “34 , 52 , 124 , 8 , 4 , 10”:
  – **Pass pertama :**
  Urutkan array berdasarkan digit terkecil (satuan) menjadi :

10 , 52 , 34 , 124 , 4 , 8

Catatan : 124 ditulis sebelum 4 karena pada urutan awal, 124 terletak sebelum 4.

Pada *pass* ini, terbentuk *array of bucket size* berikut:

1 (ukuran *bucket* digit 0 : 010)

1 (ukuran *bucket* digit 2 : 052)

3 (ukuran *bucket* digit 4 : 034 , 124 , 004)

1 (ukuran *bucket* digit 8 : 008)

- **Pass kedua :**

  Setelah itu, urutkan *array* berdasarkan digit selanjutnya (puluhan) menjadi :

  4 , 8 , 10 , 124 , 34 , 52

  Catatan : Bilangan yang tidak memiliki puluhan dianggap memiliki digit puluhan "0".

  Pada *pass* ini, terbentuk *array of bucket size* berikut:

  2 (ukuran *bucket* digit 0 : 004 , 008)

  1 (ukuran *bucket* digit 1 : 010)

  1 (ukuran *bucket* digit 2 : 124)

  1 (ukuran *bucket* digit 3 : 034)

  1 (ukuran *bucket* digit 5 : 052)

- **Pass ketiga :**

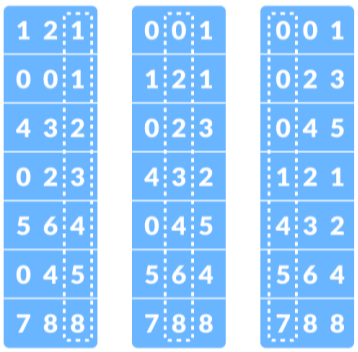
  Terakhir, urutkan *array* berdasarkan digit terbesar (ratusan) menjadi :

  4 , 8 , 10 , 34 , 52 , 124

  Pada *pass* ini, terbentuk *array of bucket size* berikut:

  5 (ukuran *bucket* digit 0 : 004 , 008 , 010 , 034 , 052)

  1 (ukuran *bucket* digit 1 : 124)



Pengurutan radix sort

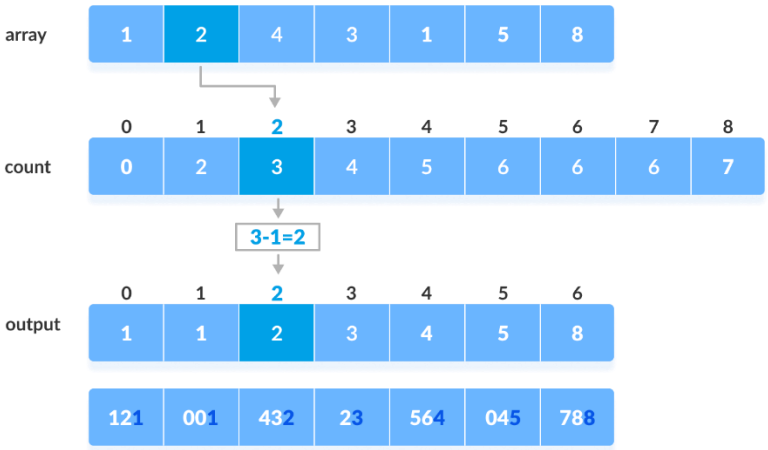
Pada contoh di atas, setiap pengurutan digit dilakukan dalam sekali *pass*. Oleh karena itu, pada contoh di atas, terjadi tiga kali *pass*.

- MSD (*most significant digit*) *radix sort*, yaitu *radix sort* yang mengurutkan data dimulai dari digit terbesar. Algoritma ini lebih susah untuk direalisasikan

daripada LSD *radix sort*, dan biasanya tidak mengikuti urutan awal data-data yang akan diurutkan

Cara kerja dari radix sort

- Temukan elemen terbesar dalam array, yaitu maksimal. Membiarkan X menjadi jumlah digit di max. X dihitung karena kita harus melewati semua tempat penting dari semua elemen.
  Dalam array ini [121, 432, 564, 23, 1, 45, 788], kami memiliki angka terbesar 788. Ini memiliki 3 digit. Oleh karena itu, loop harus naik ke tempat ratusan (3 kali).
- Sekarang, telusuri setiap tempat penting satu per satu.
  Gunakan teknik penyortiran yang stabil untuk mengurutkan angka di setiap tempat penting. Kami telah menggunakan pengurutan penghitungan untuk ini.
  Urutkan elemen berdasarkan angka tempat satuan (X=0).



Menggunakan pengurutan berdasarkan tempat unit

- Sekarang, urutkan elemen berdasarkan angka di tempat puluhan

Urutkan elemen berdasarkan tempat puluhan

- Terakhir, urutkan elemen berdasarkan ratusan

Urutkan elemen berdasarkan tempat ratusan

2. Hitung jenis big O nya, jelaskan kenapa kompleksitasnya adalah yang Anda temukan.
   Jawab :
   Kompleksitas pengurutan radix

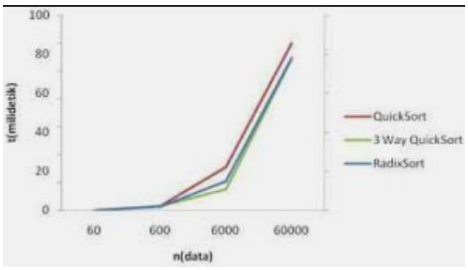
Kompleksitas ruang	O(n+k)
Kompleksitas waktu	O(nk)

Karena, pengurutan radix adalah algoritma non-komparatif, ia memiliki keunggulan dibandingkan algoritme pengurutan komparatif.

Untuk pengurutan radix yang menggunakan pengurutan penghitungan sebagai pengurutan stabil menengah, kompleksitas waktunya adalah O(nk).

Di sini, dadalah siklus bilangan dan O(nk)merupakan kompleksitas waktu pengurutan penghitungan. Dengan demikian, pengurutan radix memiliki kompleksitas waktu linier yang lebih baik daripada O(nlog n)algoritma pengurutan komparatif.

Jika kita mengambil angka digit yang sangat besar atau jumlah basis lain seperti angka 32-bit dan 64-bit maka itu dapat bekerja dalam waktu linier namun pengurutan menengah membutuhkan ruang yang besar. Ini membuat ruang sortir radix tidak efisien. Inilah alasan mengapa jenis ini tidak digunakan di perpustakaan perangkat lunak.



Grafik perbandingan beberapa sort

3. Jelaskan kelebihan dan kekurangan sorting (radix sort) yang kalian buat dibandingkan yang dibuat oleh teman kalian
   Jawab :
   Algoritma pengurutan radix memiliki beberapa kelebihan dan kekurangan dibandingkan dengan algoritma pengurutan lainnya. Beberapa keuntungan dari algoritma radix sort adalah:
   - Algoritma ini sangat efisien. Hal ini terlihat dari kompleksitas waktu asimtotik (O(nk)) yang sangat kecil. Ini membuat algoritma pengurutan radix sangat efektif bahkan untuk jumlah data yang sangat besar.
   - Konsep algoritmanya mudah dipahami. Algoritme pengurutan Radix mengurutkan data berdasarkan angka daripada proses perbandingan, yang cenderung sulit dipahami.

   Meskipun memiliki banyak keunggulan dibandingkan algoritma pengurutan lainnya, ia memiliki kekurangan.

   - Implementasi program menjadi kompleks dan kurang fleksibel jika diterapkan pada tipe data lain.
   - Menerapkan program untuk algoritma pengurutan radix tidak semudah memahami konsep dasarnya. Pada umumnya algoritma ini membutuhkan sebuah bucket untuk mengelompokkan data yang akan diurutkan. Menginisialisasi hal ini tidak mudah.

Awalnya, pengurutan radix hanya tersedia untuk data bit dan desimal. Namun, seiring waktu, radix sort telah dikembangkan untuk jenis data lainnya. Pengurutan radix sekarang tersedia untuk tipe data angka desimal dan negatif. Namun, perubahan pada algoritma ini penting. Secara umum pengembangan algoritma radix sort untuk tipe data lain dibantu dengan penggunaan count sorting, yaitu algoritma pengurutan yang tidak menggunakan perbandingan.

Aplikasi Sortir Radix

Pengurutan radix diimplementasikan di algoritma DC3 (Kärkkäinen-Sanders-Burkhardt) saat membuat array sufiks yaitu tempat-tempat di mana ada angka dalam rentang besar.