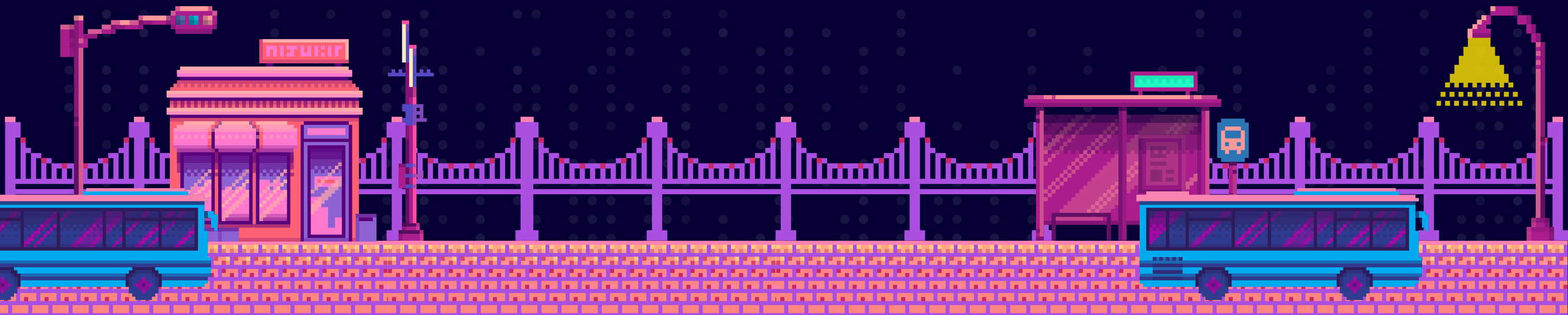

ALGORITMA
RADIX SORT
START

NAMA ANGGOTA BESERTA NIU

1. Alvito Ramadhany Rafli (538017)
2.Adam Abdillah Hanafi (544847)
3. Zaid Chandraditya Wiratama (543603)
4. Muhammad Fasha (535041)
5. Muhammad Nabil Fadhila (534448)
6. Nathanael Edwardo Sembiring (542155)
7. Marcell Albertino Panggabean(544902)
8. Qais Faqih Basyaev (542181)
9. Antonius Budi Wiryawan (545571)
10. Muhammad Tsaqif Putranda (545567)
11. Moh Rasendria Talenta Ridlo (535570)
12. Muhammad Zeva Hardi Seswanto (542780)
13. Farissa Fachrianur (478563)
14. Beltsazar Mega Rizky (540081)
15. Muhammad Rifki Arrajwa (534251)
16. Timoteus Juniorrichson Sibarani (535905)



PENGERTIAN

RADIX SORTING ALGORITHM

Secara *Etimologis* Radix berasal dari bahasa *Latin* yaitu akar, dasar, atau asal. Radix sort adalah algoritma pengurutan linear yang mengurutkan elemen dengan memprosesnya digit demi digit. Algoritma ini sangat efisien untuk pengurutan bilangan bulat (int) atau string dengan fixed-sized keys. Radix Sort tidak membandingkan masing-masing elemen secara langsung namun mendistribusikan elemen ke dalam keranjang berdasarkan nilai setiap digit.

Note: Algoritma ini hanya dapat digunakan untuk mengurutkan angka

VARIASI RADIX SORT

LSD RADIX SORT

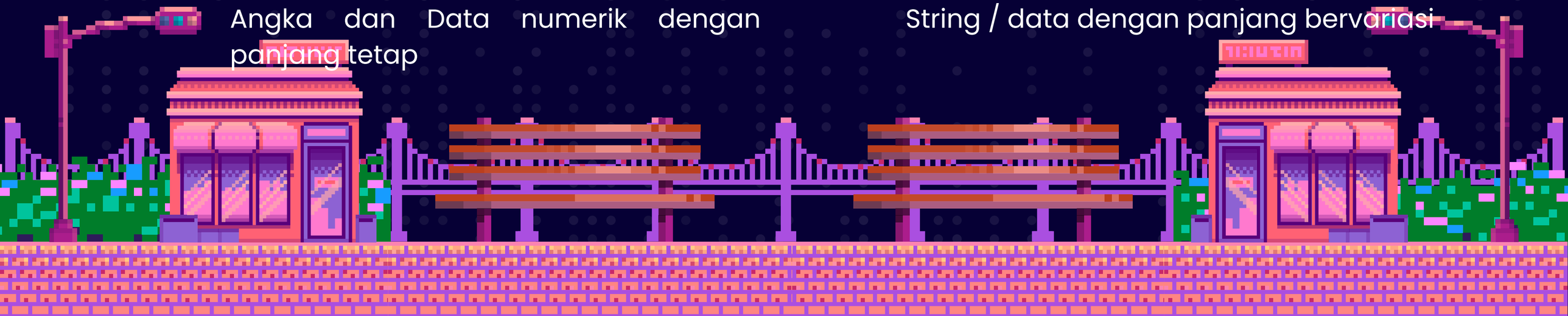
- Memulai pengurutan dari digit paling rendah (least significant digit), lalu bergerak ke digit yang lebih tinggi.

Aplikasi:
Angka dan Data numerik dengan panjang tetap

MSD RADIX SORT

- Memulai pengurutan dari digit paling tinggi (most significant digit), lalu turun ke digit yang lebih rendah.

Aplikasi:
String / data dengan panjang bervariasi



VARIASI RADIX SORT

Aspek	LSD Radix Sort	MSD Radix Sort
Teknik	Iteratif level–by–level	Rekursif divide–and–conquer
Momentum	Single-pass per digit	Multiple-pass per bucket
Stability	Stabil di setiap tahap	Harus dijaga agar tetap stabil
Penanganan Panjang	Otomatis menambahkan nol	Perlu penanganan khusus jika panjang angka berbeda

KOMPLEKSITAS VARIASI RADIX SORT

	LSD Radix Sort	MSD Radix Sort
Worst-case time	O(d·(n + k))	O(d·(n + k))
Average-case time	O(d·(n + k))	O(d·(n + k))
Ruang tambahan	O(n + k)	O(n + k·d) (karena rekursi)
Keunggulan khusus	Lebih sederhana, mudah di-implementasi	Lebih cepat jika banyak data berbagi prefix sama (early split)
Kelemahan	Proses setiap digit sama beban	Overhead rekursi tinggi jika data flat (banyak bucket kecil)

IMPLEMENTASI VARIASI RADIX SORT

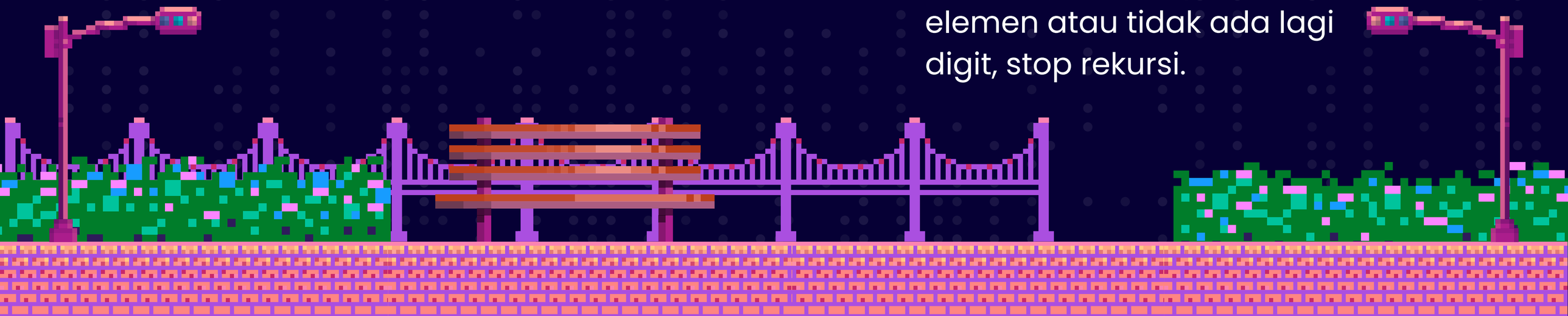
LSD RADIX SORT

Untuk setiap digit d = 0 … (maxDigits–1):

- Bagi elemen ke bucket 0–9 menurut digit ke-d.
- Gabungkan kembali semua bucket secara berurutan.

MSD RADIX SORT

1. Bagi elemen ke bucket 0–9 menurut digit paling tinggi.
2. Secara rekursif urutkan setiap bucket dengan cara yang sama, tapi untuk digit selanjutnya.
3. Saat bucket hanya satu elemen atau tidak ada lagi digit, stop rekursi.



CARA KERJA RADIX SORT

1. Temukan Elemen terbesar dalam array, pada contoh adalah 802. Pada elemen terbesar terdapay 3 digit, maka kita akan mengulanginya tiga kali, sekali proses dalam digit yang ditinjau (significant place).

2. Lakukan penyortiran pada array berdasarkan digit tempat satuan.

170	45	75	90	802	24	2	66

3. Lakukan penyortiran pada array berdasarkan digit tempat puluhan.

170	90	802	2	24	45	75	66

4. Lakukan penyortiran pada array berdasarkan digit tempat ratusan, dst.

802	2	24	45	66	170	75	90

2	24	45	66	75	90	170	802

Sorted Array

ANALISIS TIME COMPLEXITY

RADIX SORTING ALGORITHM

Radix sort adalah algoritma pengurutan integer non-komparatif (algoritma yang tidak membandingkan elemen satu per satu untuk menentukan urutan) dengan mengelompokkan kunci berdasarkan digit-digit yang memiliki posisi yang sama. Radix Sort memiliki kompleksitas waktu:

$O(nk)$ atau $O(k*(n+b))$

Nilai (n+b) adalah TC untuk Counting Sort (Pengurutan).

Dimana k = jumlah digit, n=jumlah elemen, dan b=basis sistem bilangan yang digunakan.

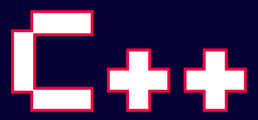
PERBANDINGAN DENGAN YANG LAIN

Algoritma	Kompleksitas Waktu Terbaik	Rata-rata	Terburuk
Radix Sort	O(nk)	O(nk)	O(nk)
Quick Sort	O(n log n)	O(n log n)	$O(n^2)$
Merge Sort	O(n log n)	O(n log n)	O(n log n)
Bubble Sort	O(n)	$O(n^2)$	$O(n^2)$

IMPLEMENTASI RADIX SORTING ALGORITHM

PSEUDOCODE

```
radixSortAlgo(arr as an array)
  Find the largest element in arr
  maximum=the element in arr that is the largest
  Find the number of digits in maximum
  k=the number of digits in maximum
  Create buckets of size 0-9 k times
for j -> 0 to k
  Acquire the j^th place of each element in arr. Here j=0 represents the least
significant digit.
  Use a stable sorting algorithm like counting sort to sort the elements in arr
according to the digits of the elements in the j^thplace
   arr = sorted elements
```



```cpp
// Implementasi Radix Sort dalam C++

#include <iostream>
using namespace std;

// Fungsi untuk mendapatkan nilai terbesar
// Nilai di dalam arr[]
int getMax(int arr[], int n)
{
    int mx = arr[0];
    for (int i = 1; i < n; i++)
        if (arr[i] > mx)
            mx = arr[i];
    return mx;
}

//Fungsi yang melakukan Counting Sort di dalam arr[]
//mengacu pada digit di representasikan dengan exp

void countSort(int arr[], int n, int exp)
{

    // Output Array
    int output[n];
    int i, count[10] = { 0 };


    //Menyimpan berapa kali perhitungan dalam count[]

    for (i = 0; i < n; i++)
        count[(arr[i] / exp) % 10]++;

    // Mengubah count[i] maka dari itu count[i]
    // sekarang memuat posisi yang sebenarnya
    // dari digit yang dimaksud dalam output[]
    for (i = 1; i < 10; i++)
        count[i] += count[i - 1];

    // Membuat output array
    for (i = n - 1; i >= 0; i--) {
        output[count[(arr[i] / exp) % 10] - 1] = arr[i];
        count[(arr[i] / exp) % 10]--;
    }

    // Menyalin output array ke arr[],
    // maka dari itu arr[] sekarang memuat
    // nilai terurut mengacu pada digit yang ditinjau
    for (i = 0; i < n; i++)
        arr[i] = output[i];
}

// Fungsi utama yang menyortir arr[]
// dengan size n menggunakan Radix Sort
void radixsort(int arr[], int n)
{

    // Mencari nilai maksimal untuk
    // mengetahui jumlah digit
    int m = getMax(arr, n);

    // Lakukan Counting Sort untuk setiap Digit
    for (int exp = 1; m / exp > 0; exp *= 10)
        countSort(arr, n, exp);
}

// Fungsi untuk mencetak hasil
void print(int arr[], int n)
{
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
}

int main()
{
    int arr[] = { 170, 45, 75, 90, 802, 24, 2, 66 };
    int n = sizeof(arr) / sizeof(arr[0]);

    // Memanggil Fungsi
    radixsort(arr, n);
    print(arr, n);
    return 0;
}
```

DAFTAR PUSTAKA

Guru99. (n.d.). Radix Sort in Data Structure: Algorithm (Pseudocode) & Example. Guru99. Retrieved May 19, 2025. from https://www.guru99.com/id/radix-sort.html#pseudocode-of-radix-sort-algorithm

Radix Sort: Algorithm, Time Complexity, Code, Example, WsCube Tech, 2025. https://www.wscubetech.com/resources/dsa/radix-sort

Sven Woltmann. Radix Sort – Algorithm, Source Code, Time Complexity. HappyCoders.eu. December 2, 2025. from https://www.happycoders.eu/algorithms/radix-sort/

C. Lee, Radix Sort Algorithm, Medium, Sep. 28, 2022. https://yuminlee2.medium.com/radix-sort-algorithm-764a260f2a04

Radix Sort Algorithm, HeyCoach Blog, Nov. 04, 2024. https://blog.heycoach.in/radix-sort-algorithm/


TERIMA
KASIH
ALGORITMA STRUKTUR DATA