ALGORITHM DESIGN

QUICK SORT COUNTING SORT BINARY HEAPS

ANGGOTA

KELOMPOK 1

IKHSAN RIHAN FARLI G1A024083	ALI AKBAR BERMANO G1A024041	M IQBAL ZAFARULLAH G1A024007
NURMELIZAH G1A024013	M FIKRAN AL FARIZI G1A024025	ANJELIA FEBILIYANTI G1A024053

KELOMPOK 1

ALGORITHM DESIGN

Perancangan Algoritma mengacu pada proses pembuatan prosedur langkah demi langkah yang efisien dan efektif untuk memecahkan masalah komputasi. Proses ini melibatkan pemilihan model matematika, struktur data, dan teknik yang tepat untuk mengembangkan solusi algoritmik.

KELOMPOK 1

QUICK SORT

QuickSort merupakan algoritma pengurutan (sorting) yang bekerja dengan cara membagi Divide and Conquer, lalu memilih satu elemen sebagai pivot dan membagi array yang diberikan.i Kemudian menempatkan pivot tersebut pada posisi yang benar dalam array yang diurutkan. Quick sort dapat digunakan dalam urutan naik (ascending) maupun turun (descending).

FUNGSI

Quick sort diguanakan untuk mengurutkan data dengan cara membagi data tersebut berdasarkan elemen pivot, kemudian mengurutkan bagian-bagian tersebut secara rekrusif hingga semua data tersusun dengan benar.

KELEBIHAN

- Cepat dan Efisien untuk data besar
- Penggunaan Memori yang hemat sehingga tidak memerlukan ruang tambahan yang besar
- Cache-Friendly karena bekerja pada sub-array yang lebih kecil
- Mudah Diimplementasikan

KEKURANGAN

- Performa Terburuk
- Tidak Stabil
- Sensitif terhadap pemilihan pivot:
- Rekursi Mendalam:

KELOMPOK 1

CIRI-CIRI

Quick Sort adalah algoritma pengurutan yang memiliki ciri-ciri: menggunakan metode divide and conquer untuk membagi data menjadi dua bagian berdasarkan pivot dan mengurutkannya secara rekursif. Algoritma ini mengurutkan data di tempat tanpa membutuhkan banyak memori tambahan dan sangat bergantung pada pemilihan pivot yang tepat. Berbeda dengan Counting Sort dan Binary Heap Sort yang memiliki waktu pengurutan yang lebih konsisten dan lebih stabil, Quick Sort bisa sangat dipengaruhi oleh pemilihan pivot dan tidak stabil.

CONTOH

Seorang guru ingin menyusun daftar nilai ujian siswanya dari yang terendah ke tertinggi, agar bisa menentukan siapa saja yang perlu remedial. Guru itu mencatat nilai berikut:
[3,1,5,7,4,8,2,6]
Untuk mengurutkan nilai-nilai ini, guru tersebut menggunakan algoritma Quick Sort.

Jawab:

Langkah 1: Pilih pivot – misalnya kita pilih nilai 6 Bagi dua kelompok:

- Kurang dari 6: [3, 1, 5, 4, 2]
- Lebih dari 6: [7, 8]
- Urutan : [3,1,5,4,2,6,7,8]

Urutkan bagian kanan [7, 8]:

- Pivot = 8
- Tidak ada perubahan

Urutkan bagian kiri [3, 1, 5, 4, 2]:

- Pivot = 2
- Kurang dari 2: [1]
- Lebih dari 2: [3, 5, 4]
- Gabung: [1, 2, 3, 5, 4]
- bagian [3,5,4]
- pivot 4
- urutan [3,4,5]
- gabungan [1,2,3,4,5]

Gabungkan Semua:

- Kiri: [1, 2, 3, 4, 5]
- Pivot utama: 6
- Kanan: [7, 8]]

Hasil akhir: [1, 2, 3, 4, 5, 6, 7, 8]

KESIMPULAN

Quick Sort adalah algoritma pengurutan yang efisien, namun dapat melambat menjadi $O(n^2)$ pada kasus terburuk. Algoritma ini tidak stabil dan sensitif terhadap pemilihan pivot.

KELOMPOK 1

COUNTING SORT

Counting sort adalah teknik pengurutan yang bekerja dengan menghitung berapa kali setiap elemen muncul dalam array. Dalam algoritma ini, kita perlu membuat sebuah array penampung untuk menyimpan jumlah kemunculan setiap data, dan ukuran array tersebut harus sesuai dengan rentang angka yang bisa dimasukkan oleh pengguna. Algoritma ini dapat mengurutkan data dari yang terbesar ke terkecil (Ascending) maupun dari yang terkecil ke terbesar (Descending).

FUNGSI

Counting sort berfungsi untuk mengurutkan data bilangan bulat (non negatif) dengan wwaktu yang cepat ketika rentang nilai data tidak terlalu besar

KELEBIHAN

-Waktu eksekusi akan sebanding dengan jumlah data dan ukuran rentang nilainya, jadi prosesnya bisa sangat cepat jika datanya memiliki rentang yang kecil.
-Pengurutan stabil
-implementasinya sederhana
-Efektif untuk data besar dengan rentang nilai terbatas

KEKURANGAN

-Tidak efisien untuk data dengan rentang nilai besar
-Membutuhkan ruang ekstra yang besarnya bergantung pada jumlah data dan rentang nilainya.
-Hanya cccok untuk bilangan bulat non-negatif

KELOMPOK 1

CIRI CIRI

Counting Sort memiliki ciri-ciri bekerja dengan menghitung frekuensi kemunculan setiap elemen, lalu mengakumulasi frekuensi tersebut untuk menentukan posisi akhir elemen dalam array hasil. Berbeda dengan Quick Sort dan Heap Sort, yang fleksibel untuk berbagai tipe data dan rentang nilai, Counting Sort hanya cocok untuk data bilangan bulat non-negatif dengan rentang nilai kecil.

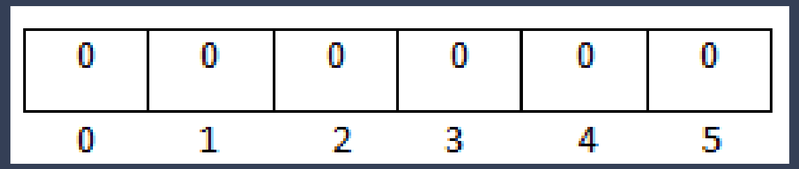
CONTOH

Misalkan sebuah toko ingin menghitung jumlah produk yang terjual dalam berbagai kategori. Setiap kategori diberi nomor ID. Toko memiliki 5 kategori produk dengan ID: Kategori 1 (elektronik), kategori 2 (alat tulis), kategori 3 (buku), kategori 4 (pakaian), kategori 5 (sepatu). Toko ingin menghitung jumlah produk yang terjual di setiap kategori dan menyusun hasilnya dalam urutan yang sesuai berdasarkan kategori produk.
Data Penjualan : [3, 1, 2, 3, 1, 2, 4, 5, 3, 1, 2, 4]

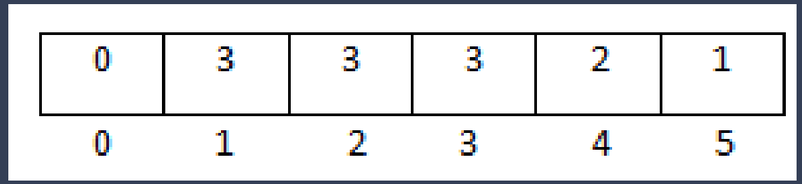
- Find the Range (Menentukan Rentang Nilai)
  Tentukan nilai maksimum dalam array. Dalam kasus ini, nilai maksimumnya adalah 5
  Array: [3, 1, 2, 3, 1, 2, 4, 5, 3, 1, 2, 4]
  Nilai maksimum: 5

KELOMPOK 1

- Initialize Count Array (Membuat Array Hitung)
  Buat array hitungan dengan ukuran nilai maksimum ditambah satu (untuk menyertakan nol). Inisialisasi semua elemen ke nol.



- Count Each Category (Menghitung Setiap Kategori)
  Setiap kategori yang ada dalam data penjualan akan dihitung dan nilainya akan ditambahkan ke posisi yang sesuai dalam array hitung.



- Cumulative Count (Penjumlahan Kumulatif)

  Setelah menghitung berapa kali kategori produk dijual selanjutnya jumlahkan angka-angka tersebut secara bertahap. Tujuannya untuk mengetahui posisi terakhir dari setiap angka dalam array yang sudah diurutkan.

  - untuk index 1 (kategori 1 - elektronik)

  Count array menjadi [0, 3, 3, 3, 2, 1] yang berarti berada di posisi 0 sampai 2 di array hasil nanti

  - Untuk index 2 (Kategori 2 - alat Tulis)

  Count array menjadi [0, 3, 6, 3, 2, 1] yang berarti berada di posisi 3 sampai 5

  - Untuk index 3 (Kategori 3 - buku)

  Count array menjadi [0, 3, 6, 9, 2, 1] yang berarti berada di posisi 6 sampai 8

  - Untuk index 4 (Kategori 4 - pakaian)

  Count array menjadi [0, 3, 6, 9, 11] yang berarti berada di posisi 9 sampai 10

  - untuk index 5 (Kategori 5 - sepatu)

  Count array menjadi [0, 3, 6, 9, 11, 12] yang berarti berada di posisi 11

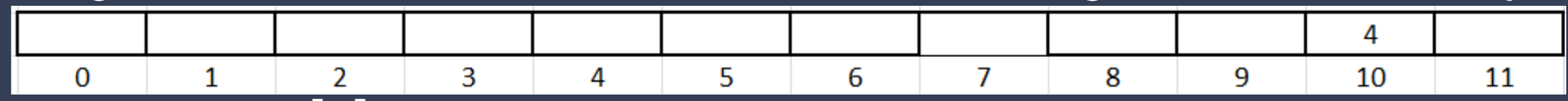
- Build the Sorted Array (Membangun Array Terurut)

  Setelah kita tahu posisi masing-masing angka, langkah selanjutnya adalah memasukkan angka-angka dari array asli ke dalam array hasil (output) sesuai dengan urutan dan posisi terakhir yang sudah kita ketahui dari penjumlahan kumulatif.

  Original array (Data Penjualan) [3, 1, 2, 3, 1, 2, 4, 5, 3, 1, 2, 4]

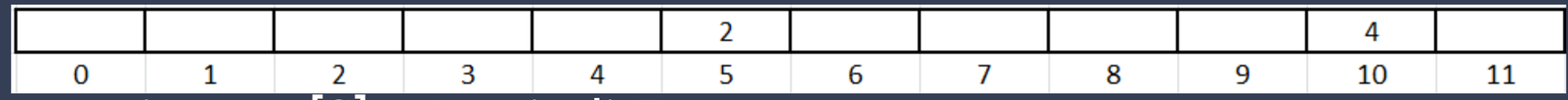
  Count array (Cumulative Count): [0, 3, 6, 9, 11, 12]

  - Ambil angka terakhir = 4, count[4] = 11, berarti angka 4 akan ditempati di index 10

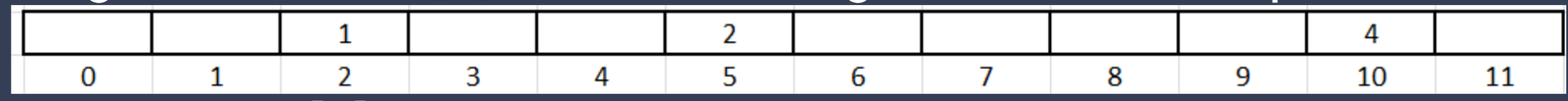
    lalu kurangi count[4] - 1 → jadi 10

  - Ambil angka = 2, count[2] = 6, berarti angka 2 akan ditempati di index 5

    lalu kurangi count[2] - 1 → jadi 5

  - Ambil angka = 1, count[1] = 3, berarti angka 1 akan ditempati di index 2

    lalu kurangi count[1] - 1 → jadi 1

- Ambil angka= 3, count[3] = 9, berarti angka 3 akan ditempati di index 8

		1			2			3		4	
0	1	2	3	4	5	6	7	8	9	10	11

lalu kurangi count[3] - 1 → jadi 8

- Ambil angka = 5, count[5] = 12, berarti angka 5 akan ditempati di index 11

		1			2			3		4	5
0	1	2	3	4	5	6	7	8	9	10	11

lalu kurangi count[5] -1 → jadi 11

- Ambil angka= 4, count[4] = 10, berarti angka 4 akan ditempati di index 9

		1			2			3	4	4	5
0	1	2	3	4	5	6	7	8	9	10	11

lalu kurangi count[4] - 1 → jadi 9

- Ambil angka= 2, count[2] = 5, berarti angka 2 akan ditempati di index 4

		1		2	2			3	4	4	5
0	1	2	3	4	5	6	7	8	9	10	11

lalu kurangi count[2] - 1 → jadi 4

- Ambil angka= 1, count[1] = 2, berarti angka 1 akan ditempati di index 1

	1	1		2	2			3	4	4	5
0	1	2	3	4	5	6	7	8	9	10	11

lalu kurangi count[1] - 1 → jadi 1

- Ambil angka= 3, count[3] = 8, berarti angka 3 akan ditempati di index 7

	1	1		2	2		3	3	4	4	5
0	1	2	3	4	5	6	7	8	9	10	11

  lalu kurangi count[3] - 1 → jadi 7

- Ambil angka= 2, count[2] = 4, berarti angka 2 akan ditempati di index 3

	1	1	2	2	2		3	3	4	4	5
0	1	2	3	4	5	6	7	8	9	10	11

  lalu kurangi count[2] - 1 → jadi 3

- Ambil angka= 1, count[1] = 1, berarti angka 1 akan ditempati di index 0

1	1	1	2	2	2		3	3	4	4	5
0	1	2	3	4	5	6	7	8	9	10	11

  lalu kurangi count[1] - 1 → jadi 0

- Ambil angka= 3, count[3] = 7, berarti angka 3 akan ditempati di index 6

1	1	1	2	2	2	3	3	3	4	4	5
0	1	2	3	4	5	6	7	8	9	10	11

  lalu kurangi count[3] - 1 → jadi 6

Hasil akhir : [1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 5]

KESIMPULAN

Counting sort ini cocok untuk data integer dengan rentang kecil, tetapi tidak cocok untuk data umum atau rentang besar

KELOMPOK 1

BINARY HEAPS

Binary Heap adalah pohon biner lengkap yang menyimpan data secara efisien, memungkinkan akses cepat ke elemen maksimum atau minimum, tergantung pada jenis tumpukan. Itu bisa berupa Min Heap atau Max Heap. Dalam Min Heap kunci di akar harus menjadi yang terkecil di antara semua kunci di tumpukan, dan properti ini harus benar secara rekursif untuk semua simpul. Demikian pula, Max Heap mengikuti prinsip yang sama, tetapi dengan kunci terbesar di akarnya.

KELOMPOK 1

CIRI-CIRI

Binary heap memiliki ciri-ciri utama seperti struktur pohon biner lengkap, memenuhi sifat tumpukan min/maks, dan dapat diimplementasikan dengan array. Pohon biner lengkap berarti semua level terisi kecuali level terakhir yang terisi dari kiri ke kanan. Sifat tumpukan min/maks berarti dalam tumpukan min, setiap simpul kurang dari atau sama dengan simpul anaknya, sedangkan dalam tu

FUNGSI

Binary heap adalah struktur data pohon biner yang memenuhi sifat heap. Fungsinya utama adalah untuk mengimplementasikan antrean prioritas dan juga digunakan dalam algoritma penyortiran heapsort. Dalam binary heap, setiap node memiliki paling banyak dua anak, dan setiap node memiliki prioritas yang memenuhi sifat heap (maksimum atau minimum).

KELEBIHAN

-Memiliki implementasi yang lebih mudah dan menggunakan lebih sedikit memori dibandingkan dengan Binary Search Tree (BST),
-Membangun heap biner memiliki waktu yang lebih singkat (O(N)) dibandingkan dengan BST (O(N log N)).

KEKURANGAN

-Penggunaan memori yang besar terutama untuk pohon biner yang besar-Tidak Menjaga Data Selalu Tersortir
-Kurang Fleksibel untuk Operasi Kompleks

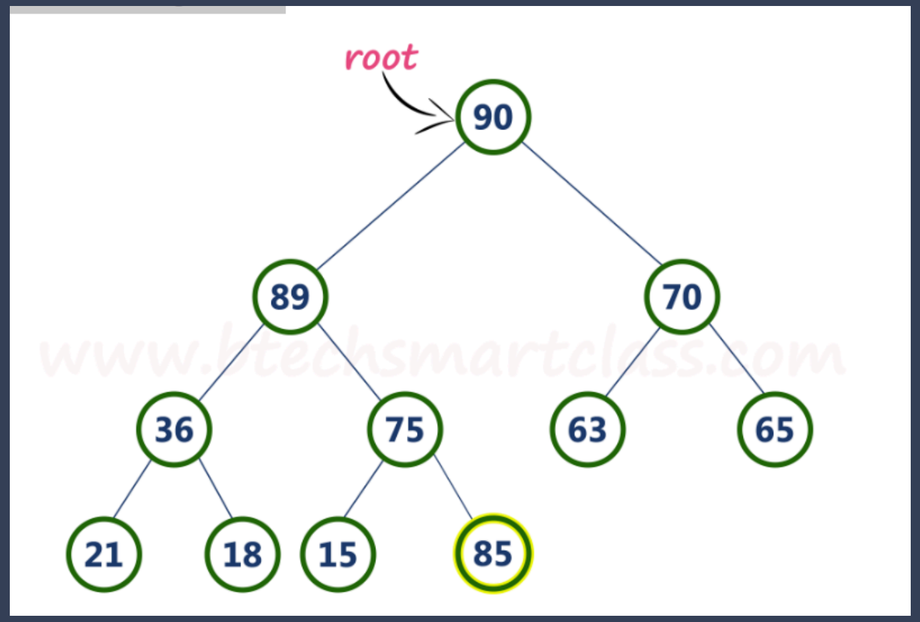
CONTOH

MENEMUKAN OPERASI NILAI MAKSIMUM DI MAX HEAP

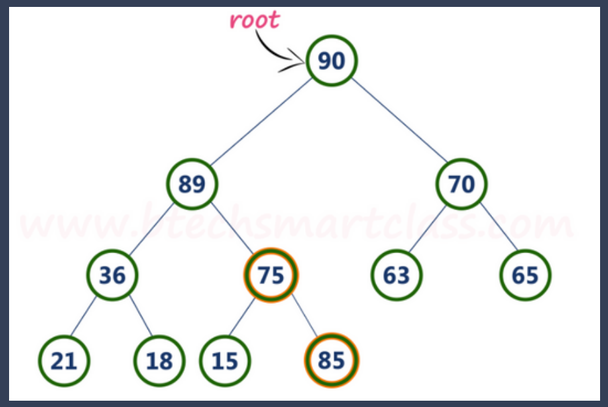
- Menemukan simpul yang memiliki nilai maksimum dalam tumpukan maksimum sangatlah mudah. Dalam tumpukan maksimum, simpul akar memiliki nilai maksimum dibandingkan semua simpul lainnya. Jadi, kita dapat langsung menampilkan nilai simpul akar sebagai nilai maksimum dalam tumpukan maksimum.

KELOMPOK 1

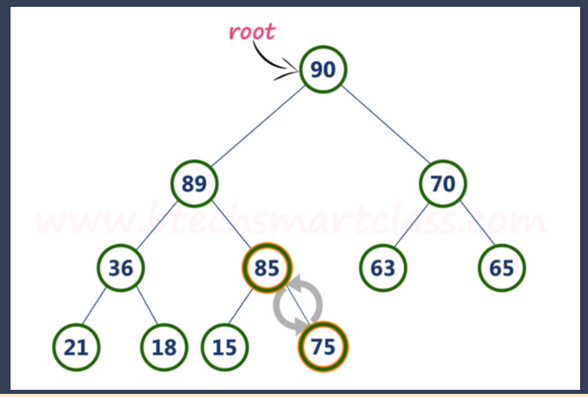
1. Masukkan newNode dengan nilai 85 sebagai daun terakhir dari kiri ke kanan. Itu berarti newNode ditambahkan sebagai anak kanan dari node dengan nilai 75. Setelah menambahkan heap maksimum adalah sebagai berikut...



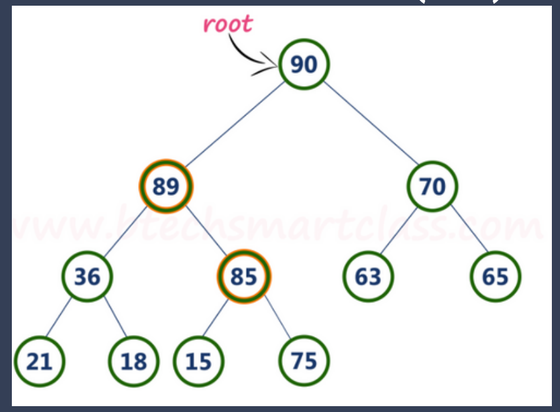
2. Bandingkan nilai newNode (85) dengan nilai simpul induknya (75) . Itu berarti 85 > 75



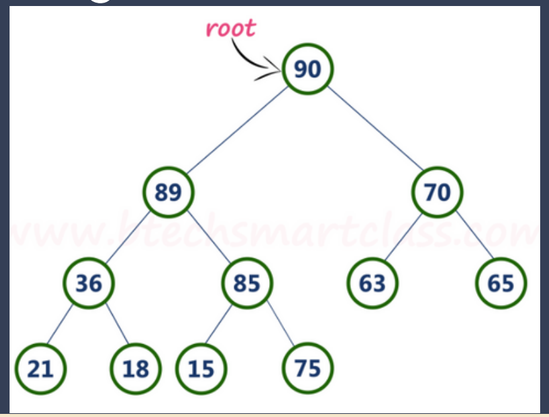
3. Di sini **nilai newNode (85) lebih besar** dari **nilai induknya (75)** , lalu **tukar** keduanya. Setelah ditukar, heap maksimum adalah sebagai berikut...



4. Sekarang, bandingkan lagi nilai newNode (85) dengan nilai node induknya (89).



5. Di sini, nilai newNode (85) lebih kecil dari nilai simpul induknya (89). Jadi, kita hentikan proses penyisipan. Akhirnya, tumpukan maksimum setelah penyisipan simpul baru dengan nilai 85 adalah sebagai berikut...



KESIMPULAN

Binary heap adalah struktur data efisien untuk operasi berbasis prioritas seperti insert dan extract. Strukturnya sederhana berbentuk array, namun kurang cepat untuk pencarian elemen tertentu.

THANK YOU