

Studi Kasus Algoritma Divide and Conquer: Mencari Titik Puncak (Peak Finding Problem) dalam Array 1D

Identitas Kelompok 10 Kelas IK23A

231011002	Andi Nurul Fitriah syahrir
231011088	Listyana
231011021	Fitri Ramadani



1. Analisis Masalah dan Alasan Pemilihan Divide and Conquer

1.1. Deskripsi Masalah

Studi kasus ini berfokus pada permasalahan menemukan titik puncak dalam array satu dimensi (1D Peak Finding problem). Titik puncak dalam array yang dicari yaitu [1, 3, 4, 7, 6, 2, 5], [10, 18, 9, 11, 8, 15, 1, 14, 9, 7], [1, 3, 5, 6, 8, 9, 10, 11, 13, 2, 4, 7]. Titik puncak adalah elemen array yang lebih besar atau sama dengan elemen di sebelahnya. Algoritma ini menggunakan pendekatan pencarian biner untuk meningkatkan efisiensi waktu pencarian. Dalam beberapa kasus, bisa terdapat lebih dari satu titik puncak dalam sebuah array.

1.2. Alasan Menggunakan Divide and Conquer

Metode Divide and Conquer dipilih dibandingkan metode lainnya karena:

- 1) Algoritma ini bekerja dalam $O(\log n)$, jauh lebih cepat dibandingkan brute Force yang memiliki kompleksitas $O(n)$, jadi efisiensinya lebih tinggi.
- 2) Pencarinya lebih optimal sehingga algoritma ini tidak harus memeriksa semua elemen, melainkan membagi array menjadi bagian yang lebih kecil.
- 3) Memudahkan dalam memahami serta menerapkan konsep algoritma rekursif, karena dapat diimplementasikan secara rekursif.

1.3. Contoh Penggunaan dalam kehidupan Nyata

Salah satu contoh penggunaan Divide and Conquer dalam kehidupan sehari-hari adalah untuk menganalisis pasar saham, khususnya untuk mencari harga saham tertinggi dalam periode waktu tertentu. Dalam dunia investasi, data harga saham biasanya sangat besar dan tersebar dalam jangka waktu panjang. Pendekatan ini cocok digunakan untuk analisis historis pasar saham yang kompleks, dengan menerapkan metode Divide and Conquer, proses pencarian harga tertinggi dapat dilakukan dengan lebih efisien dan cepat, sehingga sangat membantu analisis atau investor dalam pengambilan keputusan.

2. Penjelasan Algoritma dalam Bentuk Pseudocode atau Flowchart

2.1. Pseudocode Peak Finding (1D) dengan Divide and Conquer

```
Fungsi findPeakDC(array):
    Atur kiri ke 0
    Atur kanan ke panjang array - 1
    Selama kiri ≤ kanan:
        Hitung tengah = (kiri + kanan) dibagi 2

        .....
        Jika tengah > 0 DAN array[tengah] < array[tengah - 1]:
            Geser pencarian ke kiri (kanan = tengah - 1)
        Jika tengah < panjang array - 1 DAN array[tengah] < array[tengah + 1]:
            Geser pencarian ke kanan (kiri = tengah + 1)
        Jika tidak:
            Kembalikan array[tengah] sebagai puncak
        .....

Fungsi findPeakBF(array):
    Untuk setiap indeks i dari 0 sampai panjang array - 1:
        Jika i adalah indeks pertama DAN array[i] > array[i + 1]:
            Kembalikan array[i]
        Jika i adalah indeks terakhir DAN array[i] > array[i - 1]:
            Kembalikan array[i]
        Jika array[i] lebih besar dari kiri dan kanan:
            Kembalikan array[i]
```

2.2. Penjelasan Langkah-Langkah

Divide and Conquer

- 1) fungsi `findPeakDC(array)` akan dipanggil dan langsung menjalankan proses pencarian dengan pendekatan iteratif (tanpa rekursi).
- 2) Fungsi menginisialisasi dua indeks penanda, yaitu `left` (posisi paling kiri array) dan `right` (posisi paling kanan array).
- 3) Fungsi akan menjalankan perulangan selama `left` masih kurang dari atau sama dengan `right`.
- 4) Di dalam perulangan, ditentukan indeks tengah dari array yang sedang diproses.
- 5) Setelah indeks tengah ditentukan, dilakukan pemeriksaan apakah elemen `arr[mid]` merupakan titik puncak dengan kondisi:
 - o Jika elemen tengah (`arr[mid]`) lebih kecil dari elemen sebelumnya (`arr[mid - 1]`), maka pencarian diarahkan ke kiri.
 - o Jika elemen tengah lebih kecil dari elemen setelahnya (`arr[mid + 1]`), maka pencarian diarahkan ke kanan.
 - o Jika tidak memenuhi dua kondisi tersebut, maka `arr[mid]` dianggap sebagai titik puncak dan dikembalikan sebagai hasil.
- 6) Proses ini akan berulang secara rekursif hingga ditemukan satu elemen yang memenuhi syarat sebagai titik puncak.
- 7) Setelah titik puncak ditemukan, nilainya akan dikembalikan ke fungsi utama dan dicetak sebagai hasil akhir.

Brute Force

- 1) Fungsi findPeakBF(array) akan memeriksa elemen demi elemen dari indeks pertama hingga terakhir.
- 2) Pada setiap iterasi, fungsi akan memeriksa apakah elemen tersebut merupakan titik puncak dengan kondisi:
 - o Jika elemen berada di indeks pertama, cukup dibandingkan dengan elemen setelahnya.
 - o Jika elemen berada di indeks terakhir, cukup dibandingkan dengan elemen sebelumnya.
 - o Jika elemen berada di tengah-tengah, dibandingkan dengan elemen sebelum dan sesudahnya.
- 3) Jika ditemukan elemen yang lebih besar dari tetangganya, maka elemen tersebut dikembalikan sebagai titik puncak.
- 4) Pencarian akan berhenti segera setelah satu titik puncak ditemukan.

3. Implementasi Kode dan Uji Coba dengan Berbagai Input

4.1. Kode Program dalam Python

Input:

```
import time

# Metode Divide and Conquer
def cariPuncak_DC(data):
    kiri = 0
    kanan = len(data) - 1

    while kiri <= kanan:
        tengah = kiri + (kanan - kiri) // 2

        if tengah > 0 and data[tengah] < data[tengah - 1]:
            kanan = tengah - 1
        elif tengah < len(data) - 1 and data[tengah] < data[tengah + 1]:
            kiri = tengah + 1
        else:
            return data[tengah]

# Metode Brute Force
def cariPuncak_BF(data):
    for i in range(len(data)):
        if i == 0 and data[i] > data[i + 1]:
            return data[i]
        elif i == len(data) - 1 and data[i] > data[i - 1]:
            return data[i]
        elif 0 < i < len(data) - 1 and data[i] > data[i - 1] and data[i] > data[i + 1]:
            return data[i]

def main():
    kumpulan_array = [
        [1, 3, 4, 7, 6, 2, 5],
        [10, 18, 9, 11, 8, 15, 1, 14, 9, 7],
        [1, 3, 5, 6, 8, 9, 10, 11, 13, 2, 4, 7]
```

```

        [1, 3, 4, 7, 6, 2, 5]

    print("\nHasil pencarian nilai puncak:\n")
    print("Menggunakan Divide and Conquer:")
    for data in kumpulan_array:
        print(f"Data: {data}")
        mulai = time.perf_counter()
        hasil = cariPuncak_DC(data)
        selesai = time.perf_counter()

        durasi = selesai - mulai
        print(f"Puncak: {hasil}")
        print(f"lama Eksekusi: {durasi:.6f} detik\n")

    print("-" * 55)
    print("Menggunakan Brute Force:")
    for data in kumpulan_array:
        print(f"Data: {data}")
        mulai = time.perf_counter()
        hasil = cariPuncak_BF(data)
        selesai = time.perf_counter()

        durasi = selesai - mulai
        print(f"Puncak: {hasil}")
        print(f"lama Eksekusi: {durasi:.6f} detik\n")

main()

```

Output:

```

PS C:\Users\Asus\OneDrive\Documents> python IK23A_TK_DivideNConquer_Kelompok10.py

Hasil pencarian nilai puncak:

Menggunakan Divide and Conquer
Data: [1, 3, 4, 7, 6, 2, 5]
Puncak: 7
Lama Eksekusi: 0.000006 detik

Data: [10, 18, 9, 11, 8, 15, 1, 14, 9, 7]
Puncak: 18
Lama Eksekusi: 0.000006 detik

Data: [1, 3, 5, 6, 8, 9, 10, 11, 13, 2, 4, 7]
Puncak: 13
Lama Eksekusi: 0.000004 detik

-----
Menggunakan Brute Force
Data: [1, 3, 4, 7, 6, 2, 5]
Puncak: 7
Lama Eksekusi: 0.000013 detik

Data: [10, 18, 9, 11, 8, 15, 1, 14, 9, 7]
Puncak: 18
Lama Eksekusi: 0.000007 detik

Data: [1, 3, 5, 6, 8, 9, 10, 11, 13, 2, 4, 7]
Puncak: 13
Lama Eksekusi: 0.000009 detik

PS C:\Users\Asus\OneDrive\Documents>

```

4. Perbandingan Efisiensi dengan Brute Force

4.1. Perbandingan Efisiensi Divide and Conquer dan Brute Force.

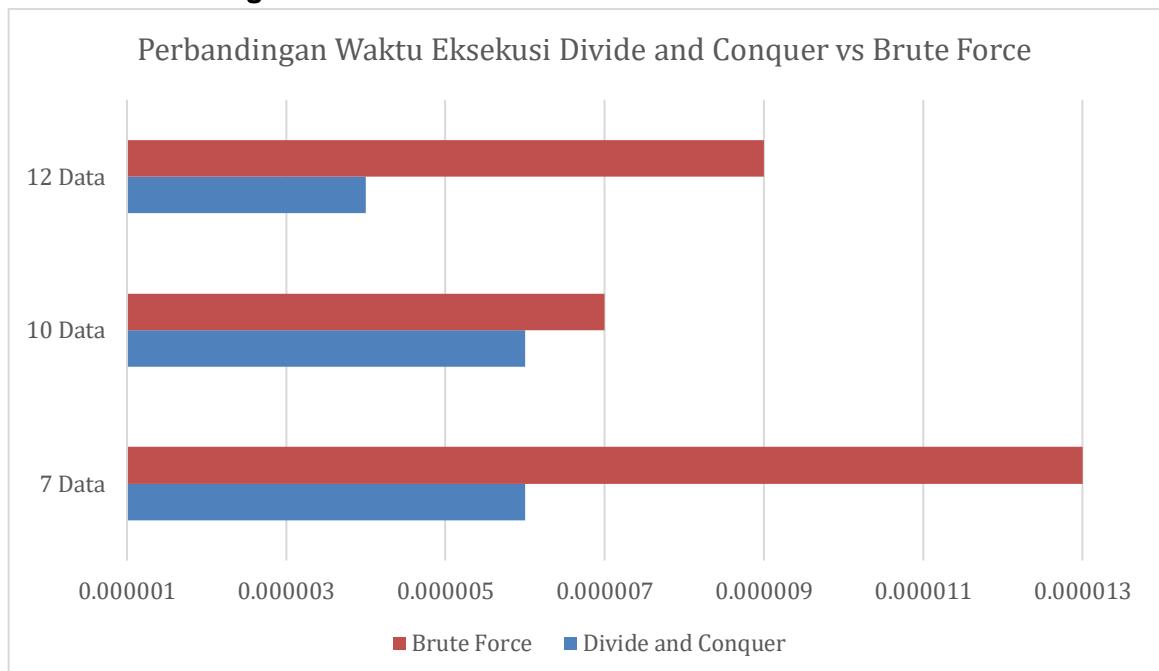
- Divide and Conquer

Algoritma *find_peak* menggunakan pendekatan Divide and Conquer, yang membagi array menjadi dua bagian di setiap langkah rekursi. Oleh karena itu, kompleksitas waktunya adalah $O(\log n)$.

- Brute Force

Pendekatan Brute Force mengecek satu persatu setiap elemen dan membandingkan dengan tetangganya, yang memerlukan $O(n)$ operasi

4.2. Grafik Perbandingan Waktu Eksekusi



4.3. Analisis Efisiensi dengan Divide and Conquer

Jumlah Data/ Array	Waktu Eksekusi	
	Divide and Conquer(s)	Brute Force(s)
7	0.000006	0.000013
10	0.000006	0.000007
12	0.000004	0.000009

5. Kesimpulan dan Tantangan yang Dihadapi

Berdasarkan hasil analisis dan implementasi algoritma Divide and Conquer dalam studi kasus pencarian titik puncak pada array satu dimensi, dapat disimpulkan bahwa metode ini memberikan solusi yang jauh lebih efisien dibandingkan dengan pendekatan Brute Force. Dengan kompleksitas waktu $O(\log n)$, algoritma ini mampu menemukan titik puncak tanpa harus memeriksa setiap elemen dalam array satu per satu, seperti yang dilakukan oleh metode Brute Force yang memiliki kompleksitas $O(n)$.

Selama proses implementasi algoritma ini, ada beberapa tantangan yang dihadapi, terutama dalam memahami cara kerja rekursi dalam Divide and Conquer. Bagi kami yang belum terbiasa dengan konsep rekursi, memahami bagaimana algoritma ini membagi array, mengevaluasi elemen tengah, dan kemudian memutuskan apakah harus mencari di bagian kiri atau kanan merupakan sesuatu yang cukup membingungkan pada awalnya.

Berdasarkan hasil studi kasus ini, dapat disimpulkan bahwa algoritma Divide and Conquer telah memberikan hasil yang sangat memuaskan dalam pencarian titik puncak pada array satu dimensi. Efisiensinya yang tinggi dan kompleksitas waktu yang lebih rendah dibandingkan metode Brute Force menunjukkan bahwa algoritma ini sudah sangat optimal untuk kasus ini. Oleh karena itu, tidak ada saran untuk pengembangan lebih lanjut karena metode yang digunakan telah mampu menyelesaikan permasalahan dengan efektif dan efisien.

Code of Conduct

Kami menyatakan bahwa Tugas Kelompok ini kami kerjakan dengan usaha kami sendiri. Kami tidak menyalin jawaban dari sumber mana pun. Kami bertanggung jawab menjaga agar jawaban Tugas Kelompok kami tidak disalin oleh peserta kelompok lain atau pihak lain yang tidak terlibat dalam penggerjaan tugas ini.

Proposi penggerjaan Tugas Kelompok adalah sebagai berikut:

NPM	Nama Anggota	Deskripsi Tugas	Kontribusi (%)
231011002	Andi Nurul Fitriah syahrir	Implementasi: Menentukan code algoritma (BF), susun isi laporan(Nomor 2, 3)	32%
231011088	Listyana	Implementasi: Menentukan code algoritma (BF dan DC), mengetik laporan, susun isi laporan(Nomor 1, 3, 4)	36%
231011021	Fitri Ramadani	Implementasi: susun isi laporan (Nomor 2, 5)	32%

Tanda tangan ketua kelompok:



Andi Nurul Fitriah Syahrir