Penerapan Binary Search dalam Mengidentifikasi Waktu Hilangnya Objek dalam Video

M. Hanief Fatkhan Nashrullah - 13522100
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13522100@std.stei.itb.ac.id

***Abstrak*— Video merupakan salah satu media elektronik yang dapat menampilkan suatu peristiwa secara detail. Video dapat digunakan untuk menangkap momen ketika suatu objek dicuri atau menghilang. Namun, untuk mengetahui kapan peristiwa tersebut terjadi, diperlukan analisis video yang dapat menghabiskan banyak waktu. Penggunaan algoritma binary search dapat membantu mengurangi langkah yang dibutuhkan sehingga waktu yang dibutuhkan untuk mencari *timestamp* dari peristiwa tersebut dapat diketahui dengan cepat.**

Keywords—video, objek, menghilang, timestamp, waktu, binary search

I. PENDAHULUAN

Video merupakan salah satu media elektronik yang dapat menampilkan suatu peristiwa secara detail dan dapat bergerak. Umumnya video digunakan sebagai media hiburan, edukasi, informasi, promosi, dokumentasi, komunikasi dan masih banyak lagi. Salah satu kegunaan dari video adalah untuk keamanan. Video dapat digunakan untuk keamanan seperti menangkap momen ketika suatu objek dicuri atau menghilang.

Analisis dari video untuk mendapatkan momen peristiwa suatu objek dicuri atau menghilang dapat menjadi aspek yang sangat penting dalam menanggapi peristiwa pencurian atau kehilangan tersebut. Namun, analisis dari video dengan skala besar dan durasi panjang dapat menguras tenaga dan waktu sehingga proses analisis menjadi sangat tidak efisien. Maka dari itu diperlukan cara yang lebih baik untuk melakukan analisis video dalam rangka menemukan waktu atau *timestamp* dari suatu objek spesifik menghilang dari video.

Algoritma *binary search* dapat diterapkan pada persoalan ini. Algoritma ini dapat mengurangi langkah yang diperlukan untuk mencari *timestamp* dari suatu peristiwa pencurian ataupun menghilangnya suatu objek dari video. Pengurangan langkah pemeriksaan akan sangat membantu dan mempercepat lama proses pencarian *timestamp* dari peristiwa pencurian objek ataupun hilangnya suatu objek.

II. LANDASAN TEORI

A. Algoritma Brute Force

Algoritma *brute force* merupakan algoritma yang memiliki pendekatan *straightforward* dalam menyelesaikan suatu persoalan. Algoritma ini mengeksplorasi seluruh kemungkinan yang ada untuk menemukan solusi dari masalah tersebut. Solusi yang diberikan oleh *brute force* dijamin akan memberikan solusi yang tepat dan optimal. Namun, dengan pendekatannya yang mengeksplorasi seluruh kemungkinan yang ada, diperlukan daya komputasi yang besar dan waktu yang lama dalam penyelesaian persoalan.

B. Algoritma Decrease and Conquer

Algoritma *decrease and conquer* merupakan sebuah algoritma untuk menyelesaikan suatu masalah dengan mereduksi masalah tersebut menjadi dua *sub-problem* yang lebih kecil dan kemudian hanya memproses salah satu dari dua *sub-problem* yang telah dibentuk. Algoritma ini berbeda dengan algoritma *divide and conquer* yang membagi persoalan menjadi dua persoalan yang berbeda dan kemudian tetap memproses kedua *sub-problem* yang telah dibentuk sebelumnya. Bagian yang diselesaikan inilah yang membedakan *decrease and conquer* dengan *divide and conquer*, *decrease and conquer* hanya menyelesaikan bagian dari *sub-problem* yang telah dibentuk, sementara *divide and conquer* menyelesaikan seluruh persoalan yang telah dibentuk.

Decrease and conquer terdiri atas dua tahapan, yaitu tahap *decrease* dan tahap *conquer*. Tahap *decrease* merupakan tahap membagi atau mereduksi permasalahan menjadi masalah yang lebih kecil. Tahap *conquer* merupakan tahap menyelesaikan dengan memilih salah satu *sub-problem* yang dibentuk.

Decrease and conquer memiliki tiga varian, yaitu *decrease by a constant*, *decrease by a constant factor*, dan *decrease by a variable size*. *Decrease by a constant* mereduksi permasalahan dengan suatu konstanta yang sama pada setiap iterasinya. *Decrease by a constant factor* mereduksi permasalahan dengan suatu faktor konstanta yang sama pada setiap iterasi. Terakhir, *decrease by a variable size* mereduksi permasalahan dengan suatu variable yang dapat memiliki nilai berbeda pada setiap iterasinya.

C. Algoritma Binary Search

Algoritma *binary search* merupakan salah satu contoh dari algoritma *decrease and conquer*. Algoritma ini adalah varian *decrease by a constant* factor dari algoritma *decrease and* conquer. Algoritma ini berfungsi untuk mencari suatu posisi dari nilai target yang ada pada suatu *array*.

Terdapat dua syarat yang harus dipenuhi oleh sebuah *array* agar algoritma *binary search* dapat diterapkan. Pertama, seluruh elemen dari *array* harus sudah terurut. Kedua, elemen dari *array* harus dapat diakses melalui posisinya dengan waktu yang konstan.

Secara garis besar, cara kerja dari algoritma ini adalah memilih elemen yang berada di posisi tengah, lalu ambil Keputusan yang bersesuaian dengan tujuan. Pertama, pilih elemen yang berada di posisi tengah *array*. Lalu, tentukan apakah elemen tersebut lebih besar, lebih kecil, atau merupakan elemen target itu sendiri. Jika target lebih besar dari elemen posisi tengah yang diperiksa, pindahkan batas bawah menjadi posisi tengah, kemudian pilih posisi tengah baru yaitu elemen yang berada di tengah batas bawah dan batas atas. Jika target lebih kecil dari elemen posisi tengah yang diperiksa, pindahkan batas bawah menjadi posisi tengah, kemudian pilih posisi tengah yang baru. Pemindahan batas atas – batas bawah dan posisi tengah ini dilakukan berulang kali sampai elemen target ditemukan.

Fig. 1. Contoh ilustrasi pencarian elemen bernilai 6 pada array terurut

Jumlah operasi yang ada pada *binary search* adalah sebagai berikut

$$T(n) = \begin{cases} 0 & , n = 0 \\ 1 + (n / 2), & n > 0 \end{cases}$$

Jika relasi rekurens tersebut diselesaikan secara iteratif, maka akan didapatkan

$$T(n) = k + (n / 2^k)$$

Dengan asumsi bahwa ukuran *array* adalah perpangkatan dari dua, maka akan didapatkan kompleksitas waktunya adalah O(log n).

D. Video Digital dan Hilangnya Objek

Video merupakan sebuah media elektronik yang mampu menampilkan media yang dapat bergerak. Video digital merupakan sebuah bentuk digital dari video. Dalam video digital, setiap *frame* dari video dapat divisualisasikan dengan sebuah gambar yang terpisah. Video digital merupakan bentuk video yang dapat dibaca dan diproses langsung oleh computer.

Pada dasarnya, video terdiri dari sekumpulan *frame* yang independen dari satu *frame* dengan *frame* yang lain. *Frame* ditampilkan dan diganti dengan *frame* yang lain secara sekuensial dan membentuk sebuah ilusi dari gerakan. Proses ini memungkinkan penonton seolah-olah melihat suatu peristiwa atau objek bergerak dari sekumpulan gambar statis.

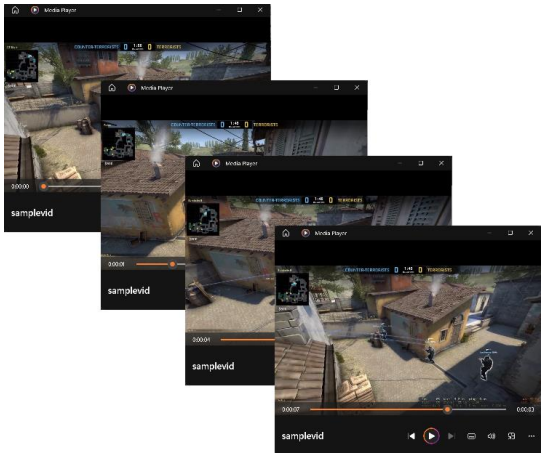


Fig. 2. Contoh ilustrasi sekumpulan *frame* dari suatu video

Menurut Kamus Besar Bahasa Indonesia (KBBI), objek merupakan "benda, hal, dan sebagainya yang dijadikan sasaran untuk diteliti, diperhatikan, dan sebagainya." Dalam pembahasan ini, objek dibatasi menjadi suatu material atau elemen yang dapat dilihat dan ditampilkan pada suatu video. Jadi, seluruh benda ataupun makhluk yang dapat dilihat dan sedang ditampilkan oleh video merupakan suatu objek pada permasalahan ini.

Hilang adalah suatu kondisi ketika sebuah objek tidak ada lagi, lenyap, atau tidak dapat dilihat. Dalam konteks permasalahan ini, hilang berarti sebuah objek tidak ditampilkan oleh video sehingga tidak dapat terlihat lagi. Jadi ketika objek yang sebelumnya ada pada video dan suatu saat objek tersebut tidak ditampilkan lagi, maka objek tersebut dianggap menghilang.

III. Metode Penyelesaian Masalah

A. Penyelesaian dengan Algoritma Brute Force

Untuk dapat menentukan waktu hilangnya objek dari *frame* video, diperlukan suatu representasi video dalam bentuk yang dapat diselesaikan oleh algoritma *brute force* dan algoritma *binary search*. Video dapat dianggap sebagai suatu *array* panjang yang memiliki nilai *boolean.* Nilai dari elemen pada *array* ditentukan oleh keberadaan objek dalam *frame*. Sehingga, video yang terdiri atas kumpulan *frame* dapat dianggap sebagai sebuah *array* dengan nilai elemen bergantung pada keberadaan dari suatu objek yang diamati pada setiap *frame* yang ada pada video.

Fig. 3. Contoh ilustrasi video dengan tiga frame

Contohnya, pada Fig. 3, terdapat video yang memiliki total tiga jumlah *frame*. Pada contoh ini, mobil *orange* akan menjadi objek yang akan diamati. Dengan demikian, video tersebut dapat direpresentasikan sebagai *array of boolean* dengan nilai *true* jika mobil *orange* ada pada dalam *frame* dan *false* jika mobil *orange* tidak ada pada dalam *frame*.

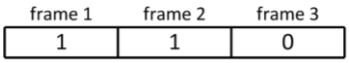


Fig. 4. Representasi *array of boolean* dari Fig. 3

Setelah permasalahan dan representasi yang tepat dapat diidentifikasi, pemilihan algoritma untuk menyelesaikan persoalan dapat dilakukan. Terdapat dua algoritma yang akan diterapkan, yaitu algoritma *brute force* dan algoritma *binary search*. Kedua algoritma ini akan dibandingkan mana yang lebih unggul dan seberapa besar perbedaan dari sisi waktu *runtime* untuk menyelesaikan persoalan ini.

B. Penyelesaian dengan Algoritma Brute Force

Pendekatan pertama adalah dengan menggunakan algoritma *brute force*. Penyelesaian dengan algoritma ini adalah dengan memeriksa satu per satu dari setiap *frame* yang ada pada video apakah objek yang diamati masih ada atau tidak pada *frame* video. Pemeriksaan ini dimulai dari *frame* pertama yang ada pada video dan dilanjutkan ke *frame* berikutnya sampai *frame* dari video habis atau objek tidak ditemukan pada *frame* dalam video.

C. Penyelesaian dengan Algoritma Binary Search

Pendekatan ini berbeda dengan pendekatan *brute force* yang memeriksa satu per satu *frame* yang ada pada video. *Binary search* hanya akan memeriksa *frame* yang berada di posisi tengah batas atas dan batas bawah dari video yang diperiksa. Sedikit berbeda dengan metode pencarian elemen pada *array of integer* dengan elemen yang sudah terurut, pemindahan batas atas atau batas bawah dilakukan berdasarkan ada atau tidaknya suatu objek dalam *frame* yang diperiksa.

Pemeriksaan dilakukan dengan membuat asumsi bahwa suatu video pasti memiliki elemen yang terurut. Terurut dalam persoalan ini adalah setiap objek yang diamati dalam video akan selalu ada pada *frame* sebelumnya jika objek ada pada *frame* yang sedang diperiksa. Jika objek tidak ada pada *frame* yang diperiksa, maka dapat dipastikan objek tidak akan ada pada *frame* setelahnya. Pemindahan batas atas dilakukan jika objek tidak ada pada *frame* dan pemindahan batas bawah dilakukan jika objek masih ada dalam frame. Pencarian akan dihentikan ketika batas bawah memiliki posisi yang sama dengan posisi tengah dari batas atas dan batas bawah. Hal inilah yang membuat pemeriksaan sedikit berbeda dengan *binary search* pada umumnya. Jika *binary search* pada umumnya mencari elemen dari *array* yang memiliki nilai sama dengan elemen target, *binary search* pada persoalan ini mencari kemunculan elemen bernilai *false* pada *array*.

D. Implementasi dengan Python dan OpenCV

Implementasi untuk melakukan pemeriksaan dan otomasi dilakukan dengan menggunakan Bahasa Python dan dengan menggunakan library OpenCV untuk mendeteksi keberadaan objek yang ingin diamati. Bahasa Python dipilih karena implementasi yang cukup mudah dan dukungan library OpenCV.

Implementasi dari program sedikit berbeda dengan penjelasan dari bagian sebelumnya. Pembangkitan *array boolean* berdasarkan keberadaan objek akan menjadi lebih berat jika seluruh *frame* diperiksa. Maka dari itu, pembangkitan nilai *boolean* mengenai keberadaan objek hanya akan dibangkitkan ketika *frame* diperiksa. Pembangkitan secara menyeluruh sama saja dengan membentuk *array of boolean* secara *brute force*.

Berikut adalah implementasi dari algoritma *brute force*

```
import cv2
import numpy as np
import time

start_time = time.time()
cap = cv2.VideoCapture('parking.mp4') # Sesuaikan dengan nama file video
object_detected = False
missing_frames = 0

while True:
    ret, frame = cap.read()
    if not ret:
        break

    frame_number = cap.get(cv2.CAP_PROP_POS_FRAMES)
    print(f'Processing frame {frame_number}')

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    lower_color = np.array([10, 100, 20])   # Sesuaikan dengan warna objek
    upper_color = np.array([25, 255, 255])  # Sesuaikan dengan warna objek

    mask = cv2.inRange(hsv, lower_color, upper_color)

    contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    if contours:
        object_detected = True
        missing_frames = 0
    else:
        if object_detected:
            missing_frames += 1

        if missing_frames == 5:
            object_detected = False
            break
    cv2.waitKey(1)
    small_frame = cv2.resize(frame, (320, 320))
    current_time_in_milliseconds = cap.get(cv2.CAP_PROP_POS_MSEC)
    cv2.imshow('Frame', small_frame)

current_time_in_seconds = current_time_in_milliseconds / 1000
hours, remainder = divmod(current_time_in_seconds, 3600)
minutes, seconds = divmod(remainder, 60)
print(f"{str(int(hours)).zfill(2)}:{str(int(minutes)).zfill(2)}:{str(int(seconds)).zfill(2)}")

end_time = time.time()
runtime = end_time - start_time
print(f'The script ran for {runtime} seconds')

cap.release()
cv2.destroyAllWindows()
```

Berikut adalah implementasi dari algoritma *binary search*

```
import cv2
import numpy as np
import time

start_time = time.time()
cap = cv2.VideoCapture('parking.mp4') # Sesuaikan dengan nama file video
object_detected = False
missing_frames = 0
total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
low = 0
mid = None
high = total_frames;
booleanValue = None;
while (low <= high) :
    mid = (low + high) // 2;
    cap.set(cv2.CAP_PROP_POS_FRAMES, mid)
    ret, frame = cap.read()
    if not ret:
        break
    frame_number = cap.get(cv2.CAP_PROP_POS_FRAMES)
    print(f'Processing frame {frame_number}')
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    lower_color = np.array([10, 100, 20])  # Sesuaikan dengan warna objek
    upper_color = np.array([25, 255, 255]) # Sesuaikan dengan warna objek

    mask = cv2.inRange(hsv, lower_color, upper_color)
    contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    if contours:
        booleanValue = True;
    else:
        booleanValue = False;
    if (booleanValue):
        low = mid + 1;
    elif (low == mid):
        print(mid);
        current_time_in_milliseconds = cap.get(cv2.CAP_PROP_POS_MSEC)
        break
    else :
        high = mid;
    current_time_in_milliseconds = cap.get(cv2.CAP_PROP_POS_MSEC)
    small_frame = cv2.resize(frame, (320, 320))
    cv2.imshow('Frame', small_frame)
    cv2.waitKey(1)
end_time = time.time()
current_time_in_seconds = current_time_in_milliseconds / 1000

hours, remainder = divmod(current_time_in_seconds, 3600)
minutes, seconds = divmod(remainder, 60)

print(f"{str(int(hours)).zfill(2)}:{str(int(minutes)).zfill(2)}:{str(int(seconds)).zfill(2)}")
runtime = end_time - start_time
print(f'The script ran for {runtime} seconds')
cap.release()
cv2.destroyAllWindows()
```

IV. Eksperimen dan Analisis

Berikut adalah eksperimen dan analisis dari percobaan dengan menggunakan algoritma *brute force* dan dengan menggunakan algoritma *binary search.*

A. Test Case 1 - 10 detik

Video ini merupakan ilustrasi dari sebuah tempat parkir dengan tiga mobil dan satu truk terparkir. Mobil merah yang terparkir akan berada di dalam video selama 7.5 detik pertama kemudian melaju keluar dari *frame* video.

Brute force :

Timestamp : 00:00:07
Runtime : 7.113 seconds
Frame checked : 229 frames

Binary search:

Timestamp : 00:00:07
Runtime : 0.896 seconds
Frame checked : 9 frames

B. Test Case 2 - 20 detik

Video ini merupakan ilustrasi dari sebuah tempat parkir dengan tiga mobil dan satu truk terparkir. Mobil merah yang terparkir akan berada di dalam video selama 10 detik pertama, kemudian mundur sebelum melaju keluar dari *frame* video.

Brute force :

Timestamp : 00:00:10
Runtime : 10.214 seconds
Frame checked : 331 frames

Binary search:

Timestamp : 00:00:11
Runtime : 1.077 seconds
Frame checked : 10 frames

C. Test Case 3 - 30 detik

Video ini merupakan ilustrasi dari sebuah restoran kosong dengan terdapat laptop berwarna biru di salah satu meja. Laptop pada meja menghilang setelah 20 detik pertama.

Brute force :

Timestamp : 00:00:19
Runtime : 15.406 seconds
Frame checked : 601 frames

Binary search:

Timestamp : 00:00:20
Runtime : 1.416 seconds
Frame checked : 11 frames

D. Test Case 4 - 60 detik

Video ini merupakan ilustrasi dari sebuah restoran kosong dengan terdapat *handphone* berwarna biru di salah satu meja. *Handphone* pada meja menghilang setelah 40 detik pertama.

Brute force :

Timestamp : 00:00:39
Runtime : 30.173 seconds
Frame checked : 1201 frames

Binary search:

Timestamp : 00:00:40
Runtime : 1.429 seconds
Frame checked : 12 frames

E. Test Case 5 - 2 menit

Video ini merupakan ilustrasi dari sekumpulan sepeda yang terparkir di suatu area parkir. Sepeda berwarna biru menghilang dari video setelah 1 menit pertama.

Brute force :

Timestamp *: 00:00:59*
Runtime *: 48.472 seconds*
Frame checked *: 1801 frames*

Binary search:

Timestamp *: 00:01:00*
Runtime *: 1.390 seconds*
Frame checked *: 12 frames*

F. Test Case 6 - 5 menit

Video ini merupakan ilustrasi dari bagian teras dari suatu rumah. Terdapat sebuah paket berwarna biru yang terletak di depan rumah tersebut. Paket ini akan menghilang di menit ke 3 lebih 20 detik.

Brute force :

Timestamp *: 00:03:19*
Runtime *: 103.394 seconds*
Frame checked *: 6001 frames*

Binary search:

Timestamp *: 00:03:20*
Runtime *: 1.579 seconds*
Frame checked *: 13 frames*

G. Test Case 7 - 10 menit

Video ini merupakan ilustrasi dari bagian teras dari suatu rumah. Terdapat sebuah paket berwarna biru yang terletak di depan rumah tersebut. Paket ini akan menghilang di menit ke 7 lebih 30 detik.

Brute force :

Timestamp *: 00:07:29*
Runtime *: 228.547 seconds*
Frame checked *: 13501 frames*

Binary search:

Timestamp *: 00:07:30*
Runtime *: 2.480 seconds*
Frame checked *: 15 frames*

H. Test Case 8 - 20 menit

Video ini merupakan gambar statis dari persegi merah yang ditampilkan di layar selama 10 menit pertama dan kemudian menghilang setelahnya.

Brute force :

Timestamp *: 00:09:59*
Runtime *: 398.599 seconds*
Frame checked *: 18001 frames*

Binary search:

Timestamp *: 00:10:00*
Runtime *: 2.068 seconds*
Frame checked *: 15 frames*

I. Test Case 9 - 30 Menit

Video ini merupakan gambar statis dari persegi merah yang ditampilkan di layar selama 20 menit pertama dan kemudian menghilang setelahnya.

Brute force :

Timestamp *: 00:19:59*
Runtime *: 581.074 seconds*
Frame checked *: 35982 frames*

Binary search:

Timestamp *: 00:19:59*
Runtime *: 1.737 seconds*
Frame checked *: 16 frames*

J. Analisis Hasil Eksperimen

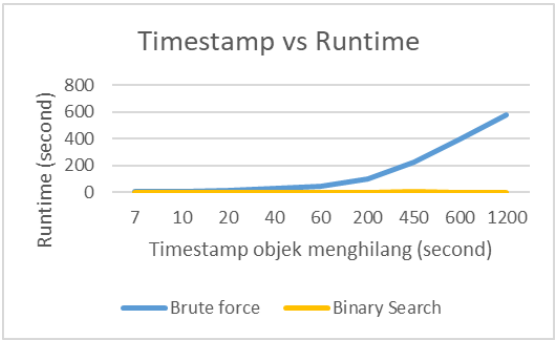


Fig. 5. Grafik perbandingan antara kapan objek menghilang dengan waktu yang dibutuhkan oleh masing masing algoritma

Berdasarkan hasil eksperimen, tidak terdapat peningkatan *runtime* yang signifikan pada algoritma *binary search* seiring meningkatnya *timestamp* atau waktu objek menghilang dari video. Sedangkan pada algoritma brute force, tampak bahwa peningkatan *runtime* pada algoritma *brute force* jauh lebih besar dibandingkan dengan algoritma *binary search*. Peningkatan *runtime* algoritma *brute force* konsisten naik meskipun besarnya fluktuatif.

Selain dari *runtime*, dapat diperhatikan juga bahwa jumlah pengecekan *frame* dari algoritma *binary search* tidak sebesar pengecekan *frame* dari algoritma *brute force*. Jumlah pengecekan *frame* algoritma *brute force* meningkat secara linear sebanding dengan lama waktu sampai objek menghilang. Sedangkan pada algoritma *binary search*, jumlah *frame* yang diperiksa hanya meningkat secara logaritmik sehingga jumlah *frame* yang diperiksa tidak sebanyak algoritma *brute force*.

V. KESIMPULAN

Algoritma *binary search* merupakan suatu algoritma yang efektif dan efisien untuk diterapkan pada persoalan pencarian element pada *array* yang terurut. Tidak hanya pada *array* yang memiliki elemen bernilai angka *real*, tetapi juga dapat diterapkan *array* yang memiliki nilai *boolean*. Dalam makalah ini, penulis menerapkan algoritma *binary search* pada persoalan pencarian waktu objek menghilang dari *frame* video. Persoalan ini dapat digeneralisasi menjadi persoalan *binary*

search dengan elemen terurut yang memiliki nilai *boolean*. Penerapan ini dapat mempersingkat waktu dalam melakukan pencarian *timestamp* dari suatu peristiwa objek menghilang dari *frame* video.

UCAPAN TERIMA KASIH

Penulis mengucapkan rasa syukur kepada Allah Subhanahu wa ta'ala karena telah memberikan kesehatan dan kelancaran kepada penulis dalam pembuatan makalah ini. Penulis juga ingin mengucapkan terima kasih kepada orangtua yang telah mendorong dan mendukung penulis untuk bisa berkuliah di Institut Teknologi Bandung. Terakhir, penulis juga ingin mengucapkan terima kasih kepada Dr. Nur Ulfa Maulidevi, S.T, M.Sc. selaku dosen dari mata kuliah IF2211 Strategi Algoritma yang telah membimbing selama keberjalanan mata kuliah ini.

REFERENSI

[1] Felton, J. (2023, January 26). Computer Scientist Attempts To Get The Police To Find His Stolen Bike Using Math. *IFLScience*. Diakses dari https://www.iflscience.com/computer-scientist-attempts-to-get-the-police-to-find-his-stolen-bike-using-math-67260

[2] Kementerian Pendidikan dan Kebudayaan Republik Indonesia. (2016). Kamus Besar Bahasa Indonesia. Diakses dari https://kbbi.kemdikbud.go.id/

[3] Munir, R. (2022). Algoritma Brute Force [PDF]. Diakses dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf

[4] Munir, R. (2024). Algoritma Decrease and Conquer [PDF]. Diakses dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Decrease-and-Conquer-2024-Bagian1.pdf

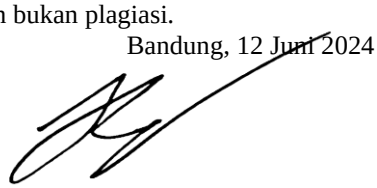
[5] Ravikishor. (2024, 18 Maret). Time and Space Complexity Analysis of Binary Search Algorithm. Diakses dari https://www.geeksforgeeks.org/complexity-analysis-of-binary-search/

[6] Saloni1297. (2023, 15 Februari). Decrease and Conquer. Diakses dari https://www.geeksforgeeks.org/decrease-and-conquer/

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Juni 2024



M. Hanief Fatkhan Nashrullah (13522100)