Penerapan dan Perbandingan Algoritma *Brute Force*, *Decrease and Conquer*, dan *Divide and Conquer* dalam Permainan "Where's Waldo?"

Mayla Yaffa Ludmilla - 13523050
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: maylayaffa@gmail.com , 13523050@std.stei.itb.ac.id

***Abstrak*— Permainan "Where's Waldo?" merupakan teka-teki visual yang menantang kemampuan manusia dalam menemukan objek tersembunyi di dalam gambar yang kompleks. Penelitian ini mengimplementasikan dan membandingkan tiga pendekatan algoritma berbeda untuk menyelesaikan permasalahan pencarian otomatis karakter Waldo menggunakan teknik pengolahan citra digital. Algoritma yang dibandingkan meliputi Brute Force yang melakukan pencarian menyeluruh pada setiap piksel gambar, Decrease and Conquer yang menggunakan pendekatan piramida resolusi untuk mempersempit area pencarian secara bertahap, dan Divide and Conquer yang membagi gambar menjadi subregion yang lebih kecil untuk diproses secara rekursif. Implementasi dilakukan menggunakan bahasa Java dengan library OpenCV untuk pengolahan citra, di mana gambar dikonversi ke grayscale dan menggunakan metode Mean Squared Error (MSE) untuk mengukur tingkat kemiripan antara template Waldo dengan potongan gambar. Hasil pengujian pada tiga test case menunjukkan bahwa algoritma Decrease and Conquer memiliki performa terbaik dengan waktu eksekusi 373-436 ms, diikuti oleh Brute Force dengan 3267-4146 ms, dan Divide and Conquer dengan 6342-11207 ms.**

Keywords—algoritma pencarian, brute force, divide and conquer, decrease and conquer, where's waldo

I. PENDAHULUAN

Permainan visual sudah sejak lama menjadi bagian dari kehidupan manusia. Teka-teki gambar tersembunyi menjadi salah satu bentuk permainan visual yang cukup digemari. Salah contoh paling ikonik dari permainan semacam ini adalah "Where's Waldo?", sebuah permainan mencari tokoh Waldo yang tersembunyi dalam peta besar yang ramai dan penuh warna." Where's Waldo?" pertama kali diperkenalkan pada tahun 1987 oleh Martin Handford, seorang ilustrator asal Inggris.

Pada permainan ini, pemain harus mencari tokoh Waldo yang khas dengan pakaian bergaris merah-putih, topi bundar, dan kacamata. Waldo biasanya "bersembunyi" di tengah kerumunan karakter dan objek, dengan detail ilustrasi yang sangat padat dan dibuat dengan membingungkan. Visual ini menjadi ciri khas permainan "Where's Waldo?" dan menjadikannya populer di berbagai kalangan di seluruh dunia.

Gambar 1.1 Karakter Waldo
Sumber: [5]

Setiap edisi "Where's Waldo?" hadir dengan variasi peta yang berbeda, mulai dari suasana kota yang padat, pantai, stadion, hingga lokasi fiktif seperti dunia fantasi atau zaman prasejarah. Pemain tidak hanya diminta untuk menemukan Waldo, tetapi juga karakter-karakter lain, objek tersembunyi, dan anomali visual yang menarik. Hal inilah yang menjadikan pengalaman membaca buku "Where's Waldo?" sebagai sebuah permainan yang interaktif dan menyenangkan, baik untuk anak-anak maupun orang dewasa.

Dalam beberapa tahun terakhir, konsep permainan semacam ini mulai dikaji dari sudut pandang teknologi, dimana kemampuan manusia dalam menemukan objek visual secara cepat dan tepat mulai dibandingkan dengan kemampuan komputer untuk menyelesaikan tugas serupa. Hal ini membuka peluang untuk mengembangkan sistem komputer yang dapat mengenali Waldo secara otomatis menggunakan pendekatan pengolahan citra digital.

Sebelum membahas lebih lanjut, perlu dikatakan bahwa pada implementasi di kajian ini, perlu ditentukan terlebih dahulu citra Waldo secara manual. Gambar ini akan digunakan sebagai acuan dalam proses pencocokan. Tanpa adanya sebuah acuan yang jelas, komputer tidak dapat melakukan pencocokan gambar dan menemukan posisi Waldo.

Dalam kajian ini, algoritma pencocokan seperti Brute Force dan pendekatan Decrease and Conquer digunakan untuk menemukan posisi Waldo pada gambar. Makalah ini bertujuan untuk membandingkan efektivitas kedua pendekatan tersebut dalam konteks kecepatan dan ketepatan pencarian. Dengan

demikian, “Where’s Waldo?” tidak hanya menjadi permainan masa kecil yang dikenang, tetapi juga contoh menarik bagi penerapan teknik pencarian visual dalam komputasi modern.

II. Dasar Teori

A. Algoritma

Dalam kehidupan sehari-hari, dapat ditemukan banyak permasalahan atau persoalan yang ingin diketahui jawabannya. Persoalan tersebut dapat berupa pertanyaan “Siapa orang dengan umur 18 tahun di antara sekumpulan orang orang ini?” atau “Bagaimana cara paling optimal untuk saya dapat memasukkan seluruh barang ke dalam ransel saya?” atau “Apakah kutipan yang saya cari ada ada di dalam artikel ini?”. Apabila banyak instans yang terlibat dalam persoalan meningkat, persoalan-persoalan tersebut, walaupun terdengar sederhana, dapat berkembang menjadi kompleks sehingga solusinya menjadi lebih sulit ditentukan. Untuk itu, algoritma dirancang sebagai prosesur umum yang berisi langkah-langkah penyelesaian persoalan.[2]

Ada banyak perbedaan pendapat mengenai definisi bagu algoritma. Menurut Levitin (2003) algoritma adalah urutan instruksi-instruksi yang tidak ambigu untuk menyelesaikan suatu masalah, yaitu untuk memperoleh keluaran (output) yang diinginkan dari setiap masukan (input) yang sah dalam rentang waktu yang terbatas. [1]

B. Brute Force

Seperti namanya, brute force (*brute* berarti kasar dan *force* berarti memaksa) adalah algoritma yang lurus atau terus terang (*straightforward*) untuk menyelesaikan sebuah persoalan. Biasanya, algoritma brute force langsung didasari oleh pernyataan persoalan dan konsep yang berhubungan dengan persoalan tersebut.

Sebagai contoh, tinjau persoalan mencari orang berumur 18 tahun dari sekelompok orang yang ada. Untuk sekelompok orang berjumlah n, persoalan ini dapat dinyatakan sebagai list berukuran n dengan isi list adalah umur orang-orang di kelompok itu. Untuk mencari orang dengan umur 18, berarti harus dicari indeks dari dari isi list tersebut yang bernilai 18. Menggunakan algoritma brute force yang *straightforward*, pendekatan yang dilakukan adalah mengiterasi seluruh isi list dan sampai menemukan isi list yang bernilai 18.

```
Procedure bruteForceUmur(ageList: array of
integer) -> integer
Input:
ageList: array berisi daftar umur orang (tidak
perlu terurut)
Output:
indeks dari orang yang berumur 18, atau -1 jika
tidak ditemukan

Deklarasi:
i: integer

Algoritma:
for i from 0 to length(ageList) - 1 do
   if ageList[i] = 18 then
       return i      // index ditemukan
   endif
endfor
return -1      // umur 18 tidak ditemukan
endprocedure
```

Walaupun brute force seringkali tidak efisien, brute force merupakan algoritma yang penting karena algoritma ini dapat diaplikasikan ke jenis persoalan yang sangat luas. Sulit untuk menemukan persoalan yang tidak dapat diselesaikan oleh brute force. Algoritma brute force juga bisa berperan sebagai alat bantu pembelajaran atau pembanding teoritis untuk mengukur seberapa baik solusi alternatif yang lebih efisien. Selain itu, algoritma brute force memastikan penemuan solusi yang tepat dengan cara mencari semua kemungkinan solusi.

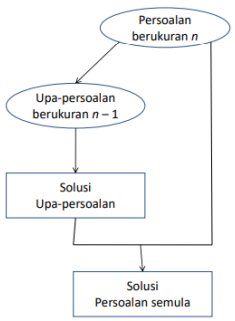
C. Decrease and Conquer

Dalam literatur terdahulu, algoritma yang membagi suatu permasalahan menjadi dua submasalah yang lebih kecil umumnya diklasifikasikan sebagai bagian dari pendekatan divide and conquer. Namun, apabila hanya satu submasalah yang diselesaikan dan sisanya diabaikan, pendekatan tersebut tidak lagi dianggap sebagai divide and conquer, melainkan masuk dalam kategori decrease and conquer.

Decrease and conquer merupakan metode perancangan algoritma yang bekerja dengan menyederhanakan permasalahan menjadi submasalah yang lebih kecil, namun hanya satu submasalah yang diproses lebih lanjut untuk memperoleh solusi. Teknik decrease-and-conquer didasarkan pada pemanfaatan hubungan antara solusi dari suatu permasalahan dengan solusi dari versi yang lebih kecil dari permasalahan tersebut. Setelah hubungan ini ditemukan, pendekatan ini dapat diterapkan secara *top-down* maupun *bottom-up*. Pendekatan *top-down* biasanya menghasilkan implementasi yang bersifat rekursif, meskipun pada akhirnya beberapa algoritma bisa diubah menjadi bentuk non-rekursif. Sebaliknya, pendekatan bottom-up biasanya diimplementasikan secara iteratif, dimulai dari solusi untuk kasus terkecil dari masalah tersebut. Pendekatan ini sering disebut juga sebagai metode inkremental. [1]

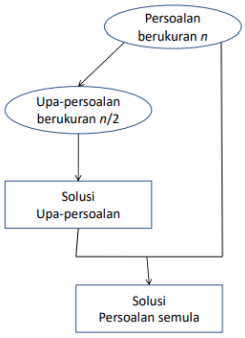
Ada tiga variasi dari decrease and conquer, yaitu decrease by a constant, decrease by a constant factor, dan variable size decrease.

- Decrease by a constant
  Ukuran persoalan dikurangi secara tetap di setiap langkah oleh konstanta tertentu, biasanya 1. Contoh persoalannya adalah perpangkatan bilangan, selection sort, dan insertion sort.



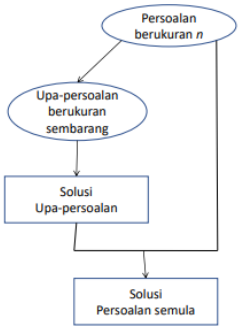
Gambar 2.1 Diagram Decrease by a Constant
Sumber: [3]

- Decrease by a constant factor
  Ukuran persoalan dikurangi berdasarkan rasio/faktor konstanta tertentu. Biasanya, faktor konstanta yang digunakan adalah 2. Contoh persoalannya adalah binary search dan mencari koin palsu.



Gambar 2.2 Diagram Decrease by a Constant Factor
Sumber: [3]

- Decrease by a variable size
  Ukuran masalah berkurang dalam jumlah yang tidak tetap, tergantung pada kondisi tertentu. Contoh persoalannya adalah interpolation search dan mencari nilai median.



Gambar 2.3 Diagram Decrease by a Variable Size
Sumber: [4]

Sebagai perbandingan dengan brute force, tinjau kembali persoalan mencari orang berumur 18 tahun dari sekelompok orang yang ada. Sama seperti brute force sebelumnya, untuk sekelompok orang berjumlah n, persoalan ini dapat dinyatakan sebagai list berukuran n dengan isi list adalah umur orang-orang di kelompok itu. Untuk mencari orang dengan umur 18, berarti harus dicari indeks dari dari isi list tersebut yang bernilai 18.

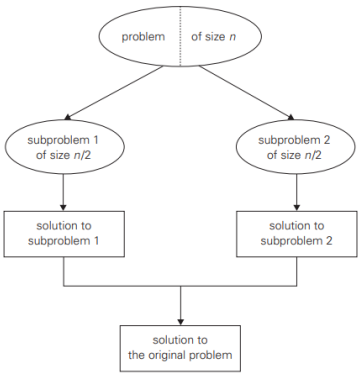
Menggunakan algoritma decrease and conquer, interpolation search dapat digunakan untuk menyelesaikan persoalan ini. Perbedaannya adalah, list dari umur tadi harus sudah terurut, dapat dari kecil ke besar maupun sebaliknya karena interpolation search memanfaatkan nilai setiap elemen list itu sendiri untuk menebak posisi indeksnya

```
Procedure InterpolationSearch(ages: array of integer, K: integer) -> integer
Input:
ages: array terurut berisi umur orang
K: umur yang ingin dicari (misalnya 18)
Output:
indeks dari umur K, atau -1 jika tidak ditemukan

Deklarasi:
low ← 0
high ← length(ages) - 1

Algoritma:
while low ≤ high and K ≥ ages[low] and K ≤ ages[high] do

   pos ← low + ((high - low) * (K - ages[low])) / (ages[high] - ages[low])

   if ages[pos] = K then
        return pos
   else if ages[pos] < K then
        low ← pos + 1
   else
        high ← pos - 1
   endif
endwhile

return -1

endprocedure
```

D. Divide and Conquer

Divide and Conquer bekerja dengan membagi permasalahan utama menjadi beberapa submasalah yang lebih kecil dan serupa dengan bentuk awal. Proses ini dilakukan melalui tiga tahap utama:

1. Divide
   Permasalahan dibagi menjadi beberapa submasalah yang memiliki tipe atau bentuk yang sama dengan permasalahan awal. Idealnya, pembagian ini menghasilkan submasalah dengan ukuran yang kurang lebih sama.
2. Conquer
   Setiap submasalah diselesaikan secara rekursif. Dalam beberapa kasus, jika ukuran submasalah sudah cukup kecil, penyelesaian dapat dilakukan menggunakan metode langsung atau algoritma lain yang lebih efisien untuk skala kecil.
3. Combine
   Jika diperlukan, hasil dari submasalah yang telah diselesaikan kemudian digabungkan untuk membentuk solusi akhir dari permasalahan awal.



Gambar 2.4 Diagram Teknik Divide and Conquer
Sumber: [1]

E. *Mean Squared Error*

Mean Squared Error (MSE) adalah salah satu cara yang digunakan untuk mengukur seberapa besar perbedaan antara nilai prediksi dengan nilai aktual. MSE dihitung dengan mengambil rata-rata dari kuadrat selisih antara nilai prediksi dan nilai sebenarnya. MSE sering digunakan dalam berbagai bidang, terutama untuk mengukur seberapa akurat suatu model prediksi atau regresi. Rumus MSE adalah sebagai berikut

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - y'_i)$$

Keterangan:

n = jumlah data

y_i = nilai aktual

y'_i = nilai prediksi

Dalam konteks *image matching*, MSE digunakan untuk mengukur seberapa mirip dua gambar (biasanya antara gambar asli atau dan gambar hasil predicted atau potongan gambar). Perhitungan dilakukan dengan membandingkan nilai piksel dari kedua gambar secara langsung. Semakin kecil nilai MSE, semakin mirip kedua gambar tersebut secara piksel per piksel.

Penggunaan MSE dalam *image matching* banyak dipilih karena perhitungan ini sederhana, mudah dihitung, dan efisien secara komputasi. Meskipun demikian, MSE memiliki keterbatasan dalam hal sensitivitas terhadap perubahan kecil pada gambar seperti rotasi, pergeseran posisi, atau pencahayaan.

F. *OpenCV*

Implementasi dari program ini memanfaatkan library OpenCV. OpenCV (Open Source Computer Vision) merupakan sebuah pustaka pemrograman sumber terbuka yang menyediakan berbagai fungsi untuk keperluan *computer vision*, khususnya dalam pengolahan citra secara real-time. OpenCV banyak digunakan dalam berbagai bidang, seperti pengolahan citra 2D dan 3D, sistem pengenalan wajah, pengenalan gestur, interaksi manusia dan komputer, deteksi objek, robotika bergerak, serta berbagai aplikasi lainnya yang berkaitan dengan analisis visual.

Pada program ini, OpenCV dimanfaatkan untuk membaca dan menyimpan gambar, mengubah citra berwarna menjadi citra grayscale agar lebih mudah dianalisis, serta mengambil potongan citra dari gambar utama untuk dibandingkan dengan gambar acuan. Selain itu, pustaka ini juga digunakan untuk menghitung tingkat kemiripan antara gambar acuan dan potongan gambar melalui perhitungan selisih nilai piksel, serta menandai lokasi hasil pencocokan terbaik pada gambar asli.

III. IMPLEMENTASI

Program dibuat dalam Bahasa Java dan untuk lebih lengkapnya dapat dilihat di tautan GitHub penulis di bagian lampiran. Untuk kedua algoritma, program akan menerima dua path gambar menuju gambar utama persoalan dan gambar target (potongan gambar Waldo dalam gambar utama tersebut). Kemudian kedua gambar tersebut diubah menjadi berwarna hitam putih untuk mengurangi kerja CPU dan mempercepat proses pencarian. Setelah itu dibuat sebuah kotak bernama ROI (*region of interest*) yang akan menjadi sliding window untuk mengambil sebagian kecil dari gambar besar dan dicocokkan dengan gambar target. Akan dibuat gambar baru yang menandai lokasi Waldo (gambar target) dengan kotak berwarna merah setelah pencarian gambar target dilakukan.

A. *Brute Force*

Pertama tama, fungsi brute force dalam kode ini akan menginisialisasi best_score untuk nilai MSE terendah dengan nilai tak hingga agar nilai MSE pertama yang lebih kecil langsung menggantikannya dan best_location untuk titik koordinat terbaik. Lalu, algoritma akan menggeser ROI ke seluruh bagian gambar utama hitam putih per piksel dan membuat patch, matriks seukuran ROI untuk dibandingkan Setelah itu, MSE dari patch dan gambar target hitam putih akan dihitung berdasarkan rumus. Jika MSE yang dihitung lebih kecil dari best score, maka best score diperbarui, begitu pula dengan best_location.

```
function brute_force_template_matching
(source_image, template_image) -> best_location
… // proses gambar

    convert source_color_image to grayscale → 
source_gray
    convert template_image to grayscale → 
template_gray

    initialize best_score ← ∞
    initialize best_location ← (0, 0)

    define roi as rectangle with same size as 
template_gray

    for y [0..(source_gray.height - 
template_gray.height)]
        for x [0..(source_gray.width - 
template_gray.width)]
            roi.x ← x
            roi.y ← y

            extract patch from source_gray using 
roi

            difference ← abs(template_gray - 
patch)
            convert difference to float
            square each element in difference
            sum all values in difference → sum_sq
            compute mse ← sum_sq / 
(template_gray.width * template_gray.height)

            if mse < best_score then
                best_score ← mse
                best_location ← (x, y)

    print best_location
```

B. *Decrease and Conquer*

Pertama-tama, gambar utama dan template akan diperkecil ukurannya beberapa kali menggunakan metode pyrDown, sehingga membentuk piramida resolusi. Tujuannya adalah agar pencocokan bisa dimulai dari gambar dengan resolusi paling

rendah (paling kasar), lalu secara bertahap ditingkatkan ke resolusi asli.

Pada level paling kasar ini, sistem akan melakukan pencocokan secara menyeluruh ke seluruh gambar, mirip seperti brute force, tapi lebih cepat karena ukuran gambarnya kecil. Setelah ditemukan posisi terbaik (dengan nilai MSE terkecil), posisi tersebut disimpan sebagai hasil sementara (currentBest).

Setelah itu, sistem turun ke level yang lebih tinggi (lebih detail). Karena ukuran gambar di level ini dua kali lebih besar, maka koordinat hasil sebelumnya juga ikut diperbesar 2x agar tetap sesuai. Tapi, di level ini sistem tidak lagi mencocokkan seluruh gambar. Sebagai gantinya, hanya area kecil di sekitar lokasi terbaik sebelumnya yang diperiksa. Di area inilah proses pencocokan dilakukan lagi, dan hasil terbaik pada level tersebut akan menggantikan currentBest. Proses ini akan terus diulang dari level ke level, sampai mencapai level 0, yaitu gambar dengan resolusi penuh (ukuran asli).

```
function decrease_and_conquer(source_image,
template_image, max_levels)-> current_best

… // proses gambar

    source_pyramid ← build_pyramid(source_image,
max_levels)
    template_pyramid ←
build_pyramid(template_image, max_levels)

    level ← max_levels - 1
    current_best ←
search_at_level(source_pyramid[level],
template_pyramid[level],full_image_region, level)

    for level [(max_levels - 2)…0]:
        scaled_location ← current_best.location*2

        search_region ←
create_search_region(scaled_location, radius,
source_width, source_height,template_width,
template_height)

        current_best ←
search_at_level(source_pyramid[level],
template_pyramid[level], search_region, level)

    return current_best
```

C. *Divide and Conquer*

Pertama-tama, algoritma akan menganggap seluruh gambar sebagai satu region besar yang akan dianalisis. Region ini kemudian akan dibagi menjadi beberapa bagian lebih kecil, empat subregion yang saling tumpang tindih sedikit. Tujuannya adalah agar tidak ada kecocokan yang terlewat hanya karena berada di batas antarregion. Setiap subregion tersebut akan diproses secara rekursif. Artinya, setiap subregion akan dianggap sebagai masalah baru yang lebih kecil, dan akan dibagi lagi menjadi sub-subregion, begitu terus hingga ukuran region sudah cukup kecil untuk dilakukan pencocokan secara langsung. Pada titik ini, sistem akan melakukan pencocokan gambar secara menyeluruh pada region kecil tersebut menggunakan metode brute force. Dari hasil ini, didapatkan lokasi terbaik dan nilai kesalahan (biasanya MSE) yang paling rendah.

Setelah semua subregion selesai diproses, hasil dari masing-masing subregion akan dibandingkan satu sama lain. Sistem kemudian memilih hasil terbaik (yang memiliki nilai MSE paling kecil) sebagai solusi untuk level tersebut. Nilai ini kemudian dibawa naik sebagai hasil untuk parent region di level sebelumnya.

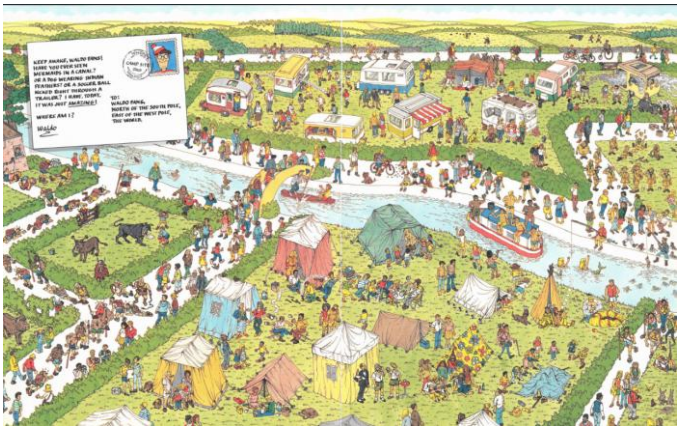
```
function divideAndConquer(source_image, template,
region, depth) -> bestResult

… // proses gambar

   Tampilkan info region dan level rekursi

   if region < threshold:
       Tampilkan info base case
       return bruteForceSearch(source_image,
template, region, depth)
   endif

   subRegions ←
divideRegion(region.template.width,
template.height)
   subResults ← list kosong

   for subRegion in subRegions do
       subResult ← divideAndConquer(source_image,
template, subRegion, depth + 1)
       tambahkan subResult ke subResults
   endfor

   bestResult ← combineResults(subResults, depth)
   return bestResult
```

IV. Hasil dan Analisis

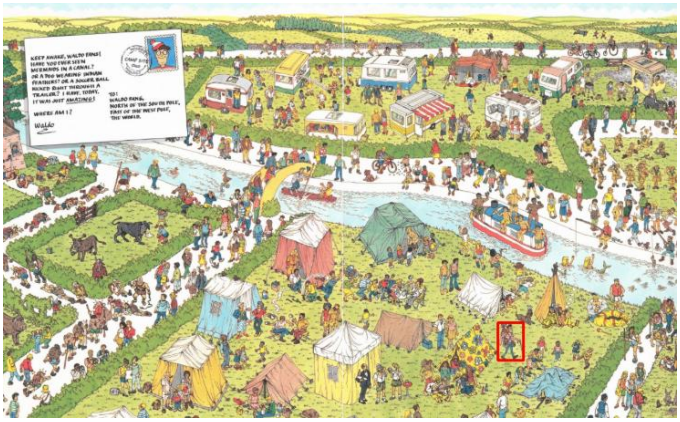
A. *Hasil*

a) *Test Case 1*

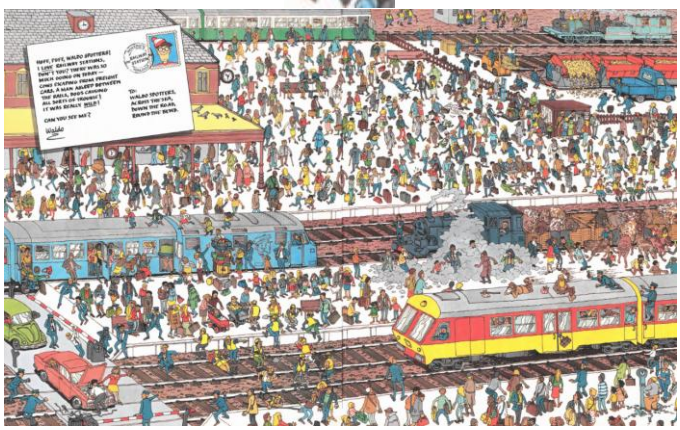
Hasil:



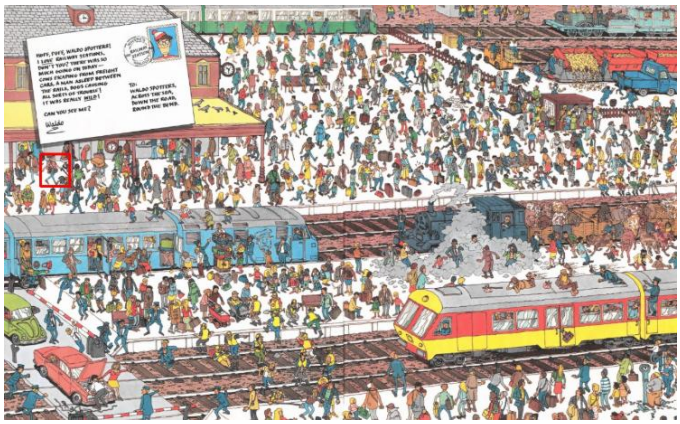
Hasil:



b) Test Case 2



Hasil:



c) Test Case 3.

Tabel 4.1 Waktu eksekusi setiap test case dengan 3 algoritma berbeda

Test Case	Runtime Execution(ms)		
	Brute Force	Decrease and Conquer	Divide and Conquer
1	3267	373	11207
2	4146	375	10077
3	4037	436	6342

B. Analisis

Berdasarkan hasil implementasi, dapat dilihat bahwa brute force sangat mahal secara waktu karena setiap posisi dalam gambar harus diuji secara eksplisit. Oleh karena itu, waktu eksekusinya cukup tinggi (3267–4146 ms), seperti terlihat pada tabel. Meskipun sederhana, metode ini tidak efisien untuk gambar besar.

Algoritma decrease and conquer adalah pendekatan yang paling efisien di antara yang lain karena tidak mencari di seluruh gambar di resolusi tinggi. Hanya sebagian kecil area yang dianalisis secara detail. Oleh karena itu, waktu eksekusi jauh lebih cepat (373–436 ms), menunjukkan efisiensi tinggi karena kompleksitasnya jauh lebih rendah dibanding brute force.

Terakhir, meskipun divide and conquer dapat mengecilkan cakupan pencarian per level, ia tetap memproses semua bagian gambar secara rekursif. Hal ini membuat waktu totalnya bisa lebih tinggi dari brute force, terutama jika ukuran gambar besar dan struktur rekursinya dalam. Tabel menunjukkan waktu eksekusi 6342–11207 ms, yang artinya bisa jauh lebih lambat daripada brute force.

V. KESIMPULAN

Dari hasil program, dapat disimpulkan bahwa algoritma brute force, decrease and conquer, dan divide and conquer dapat menemukan solusi tepat di permasalahan "Where's Waldo". Akan tetapi, ketiga algoritma tersebut memiliki waktu eksekusi yang berbeda, dengan urutan dari paling cepat ke paling lambat adalah decrease and conquer, brute force, dan divide and conquer.

VIDEO LINK DI YOUTUBE

https://youtu.be/8FIVeujvXS4?si=xtu9EKIuiT0oJWoX

GITHUB KODE

https://github.com/maymilla/wheres-waldo-solver

UCAPAN TERIMA KASIH

Penulis menyampaikan ucapan terima kasih kepada:
1. Tuhan Yang Maha Esa, yang atas berkat-Nya penulis dapat menyelesaikan makalah ini,
2. Kedua orang tua penulis yang selalu memberikan dukungan kepada penulis,
3. Ibu Dr. Nur Ulfa Maulidevi selaku dosen pengampu mata kuliah IF2211 Strategi Algoritma kelas K1 yang telah membimbing penulis dan memberikan banyak ilmu yang bermanfaat selama perkuliahan.

Akhir kata, penulis mengharapkan makalah ini dapat bermanfaat bagi pihak yang membacanya.

DAFTAR PUSTAKA

[1] Levitin, Anany. 2012. *Introduction to the Design and Analysis of Algorithms*. Edisi ketiga. United States of America: Pearson.

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/02-Algoritma-Brute-Force-(2025)-Bag1.pdf (Diakses 24 Juni 2025)

[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/11-Algoritma-Decrease-and-Conquer-2025-Bagian1.pdf (Diakses 24 Juni 2025)

[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/12-Algoritma-Decrease-and-Conquer-2025-Bagian2.pdf (Diakses 24 Juni 2025).

[5] https://www.lifesizecustomcutouts.com/SC758-Wheres-Waldo-Cardboard-Cutout-Standup-Movie-Hollywood-Legends-Waldo (diakses 24 Juni 2025)

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, 24 Juni 2025

Mayla Yaffa Ludmilla 13523050