Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Tugas Besar 3 IF2211 Strategi Algoritma

Semester II tahun 2025/2026

Implementasi *Pattern Matching* pada *Chromium Browser Extension* untuk Deteksi Judi Online

Batas pengumpulan : Minggu, 30 Mei 2026 pukul 23.59

Arsip pengumpulan :

- Source program yang dapat dijalankan disertai README
- Laporan (soft copy)

Deskripsi Tugas

Gambar 1 : Luffy mengerjakan tugas

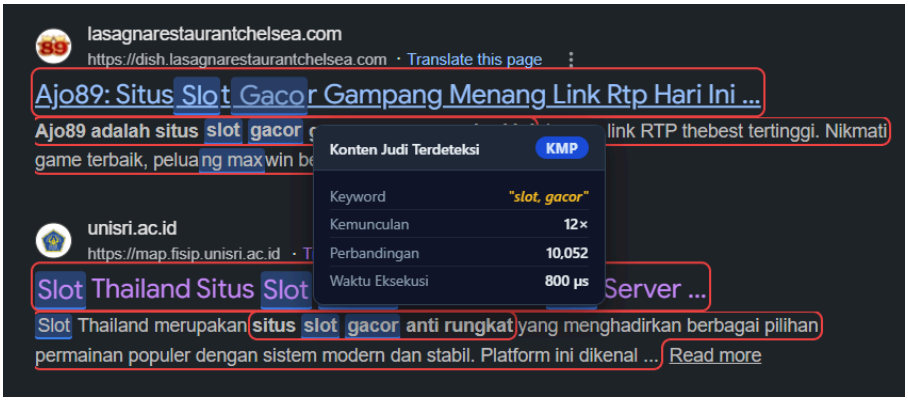
(Sumber: AI Generated Image)

Luffy adalah seorang mahasiswa Teknik Informatika yang sedang menjalani semester akhirnya bersama timnya. Setelah mempelajari berbagai materi selama masa perkuliahan, mereka mendapatkan tugas besar terakhir sebagai bentuk penerapan ilmu yang telah dipelajari. Luffy ingin membuat sebuah proyek yang

tidak hanya memenuhi kebutuhan akademik, tetapi juga memiliki hubungan dengan permasalahan yang sering ditemui di kehidupan sehari-hari.

Saat menggunakan internet untuk belajar, mencari referensi, maupun membuka berbagai website, Luffy sering menemukan kemunculan iklan dan konten perjudian online yang tersebar di berbagai halaman web. Tidak jarang, konten tersebut menggunakan variasi penulisan yang unik dan tersamarkan agar sulit dikenali secara langsung. Dari situ, Luffy mulai tertarik untuk mempelajari bagaimana sebuah sistem dapat mengenali pola-pola tertentu pada teks secara otomatis dan efisien.

Berangkat dari ide tersebut, Luffy dan timnya memutuskan untuk mengembangkan sebuah browser extension bernama **Judol Detector** sebagai proyek tugas besar mereka. Melalui proyek ini, mereka berharap dapat menggabungkan konsep algoritma, pemrosesan string, dan pengembangan perangkat lunak ke dalam sebuah aplikasi yang interaktif dan aplikatif.



Gambar 2 : Visualisasi Browser Extension

Spesifikasi Wajib

Pada tugas besar ini, buatlah sebuah ***chromium browser extension*** yang dapat mendeteksi informasi yang mengandung unsur judol. Sebagai catatan, seperti yang biasa kita temukan dalam website sekarang dengan format <kata><angka>. Contoh: GACOR99, MAXWIN88, MADU308, dan sebagainya. Sistem harus memiliki **minimal fitur** dengan detail seperti berikut.

1. Extension yang dibuat pada tugas besar ini dikembangkan dengan bahasa **Typescript**. Dalam proses pencocokan kata kunci, sistem wajib mengimplementasikan algoritma **Knuth-Morris-Pratt** dan **Boyer Moore** serta **RegEx**. Untuk algoritma selain RegEx, pencarian dilakukan menggunakan file keyword.txt (file ini boleh dimodifikasi sesuai kebutuhan) yang dipisahkan dengan endline untuk melakukan string matching secara iterative, sehingga pengguna tidak memasukkan input, tapi sistem yang membaca file tersebut secara langsung.

Untuk pencarian dengan regex, format pencarian **seminimal mungkin** dapat menghandle 2 atau 3 angka di belakang kata tanpa spasi. Contoh: MAXWIN234, SLOT99. Test case yang diberikan nanti akan sangat bervariasi, sehingga pastikan pencarian RegEx diimplementasikan dengan detail dan dapat **menghandle banyak edge case**. *Adapun juga algoritma Aho-Corasick dan Rabin Karp apabila ingin mengerjakan bonus, sesuai tertera di Spesifikasi Bonus dibawah.*

2. Perlu diketahui bahwa, Developer Judol pun juga membuat penamaan dari websitenya, seperti pada teknik *Phising,* jadi membuat nama domain yang serupa dengan pada umumnya, namun nyatanya berbeda. Ini terjadi oleh karena adanya perbedaan kode komputer (*unicode*), dan komputer mengiranya adalah karakter yang berbeda. Contoh:

Gacor999	Gɑcor999
HOKI88	H0KI88
dan lain sebagainya	

Maka dari itu, sistem harus bisa membaca kecocokan secara exact matching dan juga fuzzy matching (apabila tidak ditemukan kecocokan yang persis). Untuk setiap kata kunci yang tidak memiliki kemunculan sama sekali pada saat *exact matching*, lakukan pencarian kembali dengan perhitungan tingkat kemiripan menggunakan algoritma Weighted Levenshtein Distance. Penyesuaian bobot (*weight*) ini krusial karena pelaku sering kali menyiasati strategi dengan menggunakan huruf pengganti yang mirip secara visual (misalnya mengubah huruf 'O' menjadi angka '0', 'A' menjadi '4', atau 'I' menjadi '1'). Implementasi harus memperhatikan hal-hal berikut:

1. Weighted Cost: Berikan bobot penalti yang lebih kecil untuk substitusi karakter yang mirip secara visual dibandingkan dengan karakter yang benar-benar berbeda.
2. Thresholding: Tentukan nilai ambang batas (*threshold*) kemiripan yang tepat untuk memastikan sistem tetap memberikan hasil yang relevan namun tetap menghindari munculnya *false positive*.

Algoritma ini memungkinkan sistem untuk tetap menampilkan hasil pencarian yang relevan meskipun terdapat perbedaan minor, kesalahan ketik, atau upaya manipulasi teks oleh pengguna. *Perlu diperhatikan bahwa secara konsep, exact matching juga merupakan bagian dari fuzzy matching*.

$$\operatorname{lev}(a,b) = \begin{cases} |a| \\ |b| \\ \operatorname{lev}\left(\operatorname{tail}(a), \operatorname{tail}(b)\right) \\ 1 + \min \begin{cases} \operatorname{lev}\left(\operatorname{tail}(a), b\right) \\ \operatorname{lev}\left(a, \operatorname{tail}(b)\right) \\ \operatorname{lev}\left(\operatorname{tail}(a), \operatorname{tail}(b)\right) \end{cases} \end{cases}$$

Weighted Levenshtein Distance Formula

3. Jadi sistem menjalan pengecekan dengan hirarki sebagai berikut:
    a. Exact Matching: Algoritma *pattern matching* untuk mencari kata kunci dasar dari keyword.txt
    b. RegEx Matching: Secara paralel atau sekuensial, jalankan Regex untuk menangkap pola <kata><angka>
    c. Fuzzy Matching: Kalau string dalam DOM tidak lolos namun memiliki kemiripan, jalankan dengan Weighted Levenshtein Distance dalam upaya manipulasi karakter
4. Apabila ditemukan result keyword, maka elemen DOM terkait harus ditandai, dengan warna bebas. Namun highlight tidak boleh merusak layout, tepat sasaran dan dapat terhapus saat rescanning.
5. Saat user hover pada elemen yang terdeteksi, maka tampilkan tooltip dengan informasi berikut:
    a. Keyword yang terdeteksi
    b. Algoritma yang digunakan
    c. Jumlah kemunculan
    d. Waktu eksekusi

    Tooltip dibuat dengan custom DOM element.
6. Terdapat juga statistik secara realtime pada popup extension yang menampilkan:
    a. Total keyword yang ditemukan
    b. Perbandingan jumlah keyword yang satu dengan lainnya, menggunakan visualisasi
    c. Waktu eksekusi tiap algoritma
    d. Jumlah match tiap algoritma
7. Setiap kelompok harap mengisi nama kelompok dan anggotanya pada link berikut, paling lambat **Minggu, 17 Mei 2026 23.59**. Nama kelompok dilarang mengandung unsur SARA, penghinaan, provokasi, atau hal-hal lain yang bertentangan dengan norma hukum.

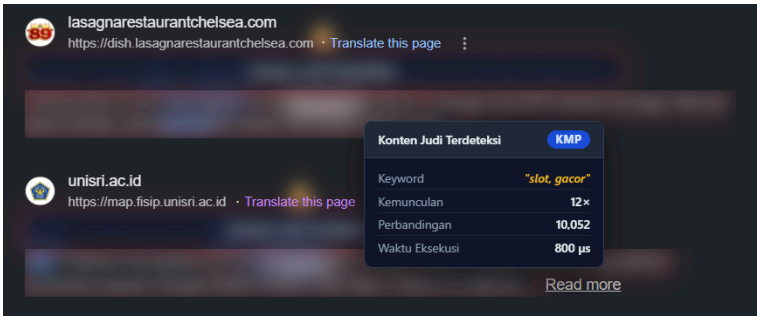
Perlu diperhatikan bahwa kelompok tubes untuk mata kuliah ini **tidak boleh sama**. Jadi jika berkelompok dengan X pada tubes ini, maka pada tubes selanjutnya anda tidak diperkenankan berkelompok dengan X.

Perhatian

a. Seluruh algoritma string matching wajib dibuat ***from scratch***
b. **Dilarang keras** menggunakan
    i. *string.includes()*
    ii. *string.indexOf()*
    iii. *Built-in search function*
    iv. *External string matching library*
c. Implementasi manual wajib mencakup:
    i. *Border function*
    ii. *Failure function*
    iii. *Last occurrence table*
    iv. *Shifting process*
    v. *Comparison Counting*
d. Regex **diperbolehkan** menggunakan regex engine dari JavaScript

Spesifikasi Bonus (Maks 20 poin)

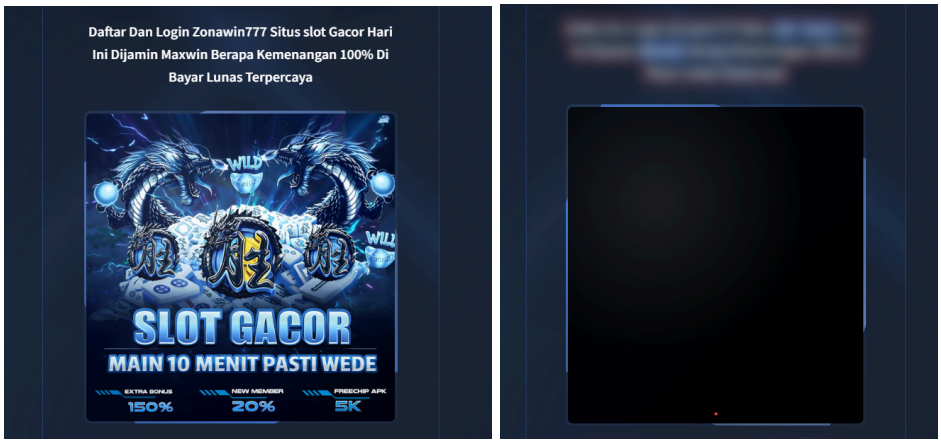
1. (6 poin) **[Video]** Membuat video tentang browser extension Judol Detector yang dikembangkan, kemudian mengunggahnya di Youtube. Video dibuat harus memiliki audio dan menampilkan wajah dari setiap anggota kelompok. Untuk contoh video tubes stima tahun-tahun sebelumnya dapat dilihat di Youtube dengan kata kunci "Tubes Stima", "strategi algoritma", "Tugas besar stima", dll. Semakin **menarik** dan **informatif** video, maka semakin banyak poin yang diberikan.
2. (4 poin) **[Algoritma Bonus]** Mengimplementasikan Algoritma Aho-Corasick dan Rabin Karp sebagai algoritma alternatif**.**
3. (5 poin) **[Censorship / Blur Teks]**
   Mengimplementasikan fitur blur visual secara otomatis pada elemen halaman web yang terdeteksi memuat konten judi online. Tidak hanya sekadar menghighlight teks, ekstensi langsung melakukan aksi blur pada elemen yang mengandung kata kunci judol, sehingga konten tersebut tidak terlihat oleh pengguna. Perlu diperhatikan, terdapat toggle on/off untuk ini.



Gambar 3 : Contoh Blur Teks

4. (5 poin) **[Optical Character Recognition Pada Gambar]**

   Mengimplementasikan fitur OCR untuk mendeteksi kata kunci judi online yang sengaja disembunyikan dalam bentuk gambar. Ekstensi akan mengekstrak teks dari elemen gambar yang ditemukan di halaman, kemudian mencocokkannya dengan daftar kata kunci judol, lalu melakukan blur atau penggantian pada gambar yang terdeteksi mengandung konten judol. Fitur ini dapat diimplementasikan dengan Tesseract.js



Gambar 4 : Contoh Hasil Sensor Gambar

***Catatan**: Gambar pengganti tidak harus hitam, bisa dikasih gambar apapun ^_^*

Petunjuk Development

Struktur projek kurang lebih akan berbentuk sebagai berikut (hanya contoh, tidak harus sama).

```
judol-detector/
│
├── public/
│   ├── manifest.json
│   ├── images/
│
├── src/
│   ├── algorithms/
│   ├── content/
│   ├── popup/
```

```
│   └── styles/
│
├── keywords/
│   └── keywords.txt
│
├── package.json
├── tsconfig.json
└── vite.config.ts
```

manifest.json merupakan file utama pada Browser Extension yang berfungsi sebagai konfigurasi extension. File ini berisi informasi penting mengenai extension seperti nama extension, versi, permission yang digunakan, file JavaScript yang dijalankan, popup extension, serta halaman mana saja yang dapat diakses oleh extension. Chrome akan membaca file manifest.json terlebih dahulu ketika extension di-load ke browser. File manifest.json sekurang-kurangnya akan berisi sebagai berikut (**ini hanya contoh, sesuaikan dengan kebutuhan projek**):

```
{
  "manifest_version": 3,
  "name": "Judol Detector",
  "version": "1.0",
  "permissions": ["storage"],
  "host_permissions": ["<all_urls>"],
  "action": {
    "default_popup": "popup.html"
  },
  "content_scripts": [
    {
      "matches": ["<all_urls>"],
      "js": ["content.js"]
    }
  ]
}
```

Untuk menggunakan project sebagai browser extension, project perlu di-build terlebih dahulu menggunakan perintah:

```
npm run build
```

Hasil build akan tersimpan pada folder dist/. Untuk mencoba extension yang telah dibuat:

1. Buka browser (misalnya Google Chrome).
2. Masuk ke halaman chrome://extensions/
3. Aktifkan opsi **Developer Mode**.
4. Tekan tombol **Load Unpacked**.
5. Pilih folder `dist/` hasil build sebelumnya.

Isi Laporan

- **Cover:** Cover laporan ada foto anggota kelompok (foto bertiga). Foto ini menggantikan logo "gajah" ganesha.
- **Bab 1:** Deskripsi tugas (dapat menyalin spesifikasi tugas ini).
- **Bab 2:** Landasan Teori.
    - Dasar teori algoritma Knuth-Morris-Pratt dan Boyer-Moore secara umum (serta algortima Aho-Corasick dan Rabin Karp jika mengerjakan bonus).
    - Penjelasan singkat mengenai browser extension yang dibangun.
- **Bab 3:** Analisis Pemecahan Masalah.
    - Langkah-langkah pemecahan masalah.
    - Proses pemetaan masalah menjadi elemen-elemen algoritma KMP dan BM.
    - Fitur fungsional dan arsitektur extension yang dibangun.
    - Contoh ilustrasi kasus.
- **Bab 4:** Implementasi dan pengujian.
    - Spesifikasi teknis program (struktur data, fungsi, dan prosedur yang dibangun).
    - Penjelasan tata cara penggunaan extension (interface, fitur-fitur yang disediakan).
    - Hasil pengujian **minimal 5 kasus** dengan variasi algoritma, dan jenis halaman yang berbeda.
    - Analisis hasil pengujian dan perbandingan performa algoritma.
- **Bab 5:** Kesimpulan dan saran.
- **Lampiran:**
    - Pernyataan Tidak Melakukan Kecurangan
    - Tautan *repository* Github
    - Tautan Video (jika mengerjakan bonus)
    - Tabel Checklist Spesifikasi
    - Tabel Pembagian Tugas

Keterangan laporan:

1. Laporan ditulis dalam bahasa Indonesia yang baik dan benar.
2. Laporan mengikuti format pada *section* Isi Laporan dengan baik dan benar.
3. Identitas per halaman harus jelas (misalnya : halaman, kode kuliah).
4. Pastikan memasukkan **Pernyataan Tidak Melakukan Kecurangan yang ditandatangani** pada laporan, dengan format sebagai berikut:

Tugas ini disusun sepenuhnya tanpa bantuan kecerdasan buatan (*Generative AI*), melainkan hasil pemikiran dan analisis mandiri.

[Tanda tangan mahasiswa]
[Nama mahasiswa]

[Tanda tangan mahasiswa]
[Nama mahasiswa]

[Tanda tangan mahasiswa]
[Nama mahasiswa]

Pengumpulan Tugas

1. Program disimpan dalam repository yang bernama **Tubes3_NamaKelompok** dengan nama kelompok sesuai dengan yang di sheets tertera. Berikut merupakan struktur dari isi repository tersebut:
    - Folder **src** berisi *source code* TypeScript yang dapat dijalankan
    - Folder **dist** berisi hasil build extension yang siap di-load di Chrome
    - Folder **doc** berisi laporan tugas besar dengan format NamaKelompok.pdf
    - **README** untuk tata cara penggunaan yang minimal berisi:
        - Penjelasan singkat algoritma KMP dan BM yang diimplementasikan
        - Requirement program dan instalasi
        - Langkah-langkah build dan cara load extension di Chrome
        - Author (identitas pembuat)
2. Sangat disarankan untuk menggunakan ***semantic commit***. Buatlah release dengan format **v1.x** dengan **x** adalah nomor revisi dimulai dari revisi 0. Contoh v1.0 untuk release pertama, v1.1 untuk revisi selanjutnya.
3. Pastikan untuk membuat repository bersifat **Public** paling lambat **H+1 deadline (1 hari setelah deadline)**. Sebelum deadline repository harus bersifat **Private**.
4. Laporan dikumpulkan hari **Minggu, 30 Mei 2026** pada alamat Google Form berikut paling lambat **pukul 23.59 WIB**:

   PERINGATAN: Keterlambatan akan mengurangi nilai sebanyak 1 poin untuk setiap menit keterlambatan.
5. Adapun pertanyaan terkait tugas besar ini bisa disampaikan melalui QnA berikut.

Penilaian

1. **Bagian 1: Desain dan analisis laporan (35%)**
    a. Pemahaman tugas besar (5%)
    b. Analisis proses preprocessing dan normalization teks (5%)
    c. Perancangan algoritma Pattern Matching menggunakan Knuth-Morris-Pratt, Boyer-Moore, dan RegEx (15%)
    d. Analisis penerapan fuzzy matching menggunakan Weighted Levenshtein Distance (5%)
    e. Analisis arsitektur browser extension, pengolahan DOM, serta efisiensi dan perbandingan performa algoritma (5%)
2. **Bagian 2: Implementasi program dan demo (65%)**
    a. Kesesuaian desain algoritma dengan implementasi program dan hasil demo (15%)
    b. Fungsionalitas keseluruhan berjalan sesuai spesifikasi (20%)
    c. Demo dan pemahaman program (30%)
3. **Bagian 3: Komponen bonus (Maksimal 20 Poin)**
    a. Membuat Video (6 Poin)
    b. Implementasi Algoritma Aho-Corasick dan Rabin Karp (4 Poin)
    c. Implementasi Censorship / Blur Teks (5 Poin)
    d. Implementasi Optical Character Recognition pada Gambar (5 Poin)

Perhatian

- Dilarang keras copy paste program dari internet, AI, repository lain, ataupun program milik teman. Program yang dikerjakan menggunakan agent (Opus, Sonnet, dan lain-lain) akan sangat terlihat dan akan memiliki kemiripan dengan mahasiswa yang menggunakan hal serupa. Segala bentuk kecurangan akan diberikan sanksi berat yaitu nilai tugas menjadi nol. Kami tidak ingin hal itu terjadi dan kami ingin Anda untuk dapat memikirkan solusi algoritma dari hasil pemikiran Anda sendiri.
- Pastikan program **dapat dikompilasi setidaknya** pada windows dan linux.
- Apabila program **tidak dapat dijalankan** maka tidak akan dinilai oleh asisten.
- Tambahkan tabel berikut yang diisi checklist (✓) pada bagian lampiran laporan dan readme Anda.

No	Poin	Ya	Tidak
1	Extension berhasil di-build dan di-load tanpa kesalahan pada chromium browser dan dikembangkan dengan TypeScript		

2	KMP dan Boyer-Moore diimplementasikan from scratch		
3	Regex menghandle format <kata><angka> dan berbagai edge case		
4	Pencarian KMP & BM membaca keyword.txt secara iteratif dan tidak menggunakan built-in search function atau library eksternal		
5	Exact matching dan fuzzy matching berjalan benar		
6	Elemen DOM terdeteksi diberi highlight dan terhapus saat rescanning		
7	Tooltip muncul saat hover dengan informasi keyword, algoritma, kemunculan, dan waktu eksekusi		
8	Popup menampilkan statistik realtime (total keyword, perbandingan, waktu eksekusi, jumlah match)		
9	[Bonus] Membuat Video		
10	[Bonus] Implementasi Algoritma Aho-Corasick dan Rabin Karp		
11	[Bonus] Implementasi Censorship / Blur Teks		
12	[Bonus] Implementasi Optical Character Recognition pada Gambar		

Referensi

- **Chrome Extension Documentation**

  https://developer.chrome.com/docs/extensions/
- **Tesseract.js**

  https://tesseract.projectnaptha.com/
- **Aho-Corasick**

  https://www.geeksforgeeks.org/dsa/aho-corasick-algorithm-pattern-searching/
- **Rabin Karp**

  https://www.geeksforgeeks.org/dsa/rabin-karp-algorithm-for-pattern-searching/

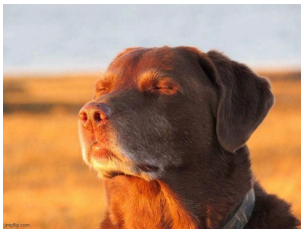
“Ditunggu ya nanti milis ‘[IF2120] Tugas Besar 4 Strategi Algoritma’”
--- Fariz---

“max win”
--- Hakim---

“Congrats udah di tubes terakhir, janlup belajar regex yah”
--- Radhi---

“”
--- Lukas---

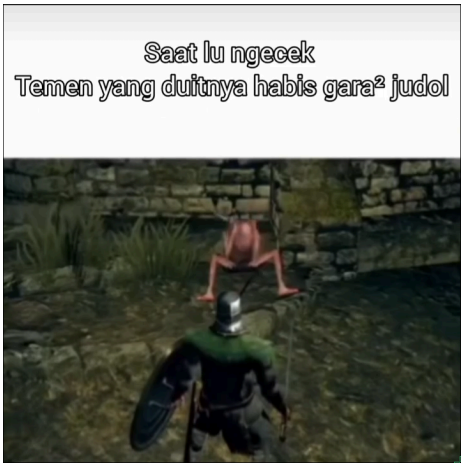
“War is Over (?)”



--- Carlo---

“all in semesta”
--- Orvin---

“Bars 🔥”
--- Rizal---



--- Theo---