Laporan Praktikum Otomata

Implementasi Pengenal Bahasa Palindrom

Kelompok A12:

Nabil Julian Syah 5025231023

Dosen:

Victor Hariadi

Muhammad Hilmil Muchtar Aditya Pradana, S.Kom., M.Sc., Ph.D.

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

2024/2025

1. Pendahuluan

1.1. Latar Belakang

Pada Otomata, bahasa (language) didefinisikan sebagai himpunan string yang terbentuk dari simbol-simbol dalam suatu alfabet. Salah satu contoh bahasa klasik yang sering dipelajari adalah bahasa palindrom. Palindrom adalah sebuah string yang akan dibaca sama, baik dari arah depan maupun dari arah belakang.

Praktikum kali ini bertujuan untuk merancang dan mengimplementasikan sebuah program yang dapat mengenali atau memutuskan apakah sebuah string input termasuk dalam bahasa palindrom.

1.2. Definisi Bahasa

- Alphabet (Σ): Himpunman simbol yang diizinkan dalam string. Untuk kasus ini, alfabet adalah gabungan dari huruf kecil dan digit: Σ = {a, b, c, …, z, 0, 1, …, 9}.
- Bahasa (L): Bahasa palindrom L didefinisikan sebagai himpunan semua string w yang terbentuk dari simbol-simbol di Σ, di mana string tersebut sama dengan kebnalikannya (w^R). $L = \{w \in \Sigma^* \mid w = w^R\}$. Contoh string dalam bahasa L: “level”, “Kodok”, “12321”. “madam1madam”.

2. Dasar Teori

2.1. Hierarki Chomsky dan Palindrom

Dalam teori bahasa formal, terdapat sebuah sistem klasifikasi yang disebut *Hierarki Chomsky*, yang membagi bahasa ke dalam empat tingkatan berdasarkan kompleksitasnya. Bahasa palindrom, yaitu bahasa yang terdiri dari string yang dapat dibaca sama dari depan maupun belakang, bukanlah bagian dari bahasa reguler. Hal ini disebabkan oleh keterbatasan memori dari Finite Automata (FA), yaitu mesin pengenal untuk bahasa reguler. Untuk mengenali sebuah palindrom, mesin perlu mengingat paruh pertama string agar dapat mencocokkannya dengan paruh kedua secara terbalik. Karena FA hanya memiliki jumlah state yang terbatas dan tidak memiliki struktur penyimpanan seperti tumpukan, maka ia tidak dapat menyimpan seluruh informasi dari paruh pertama string yang panjangnya tidak terbatas. Oleh karena itu, pengenalan bahasa palindrom memerlukan mesin yang lebih kuat, yaitu *Pushdown Automata* yang digunakan untuk mengenali bahasa bebas konteks (*Context-Free Language* / CFL). Mesin ini memiliki sebuah tumpukan yang dapat digunakan untuk menyimpan karakter, sehingga mampu mencocokkan bagian-bagian string yang diperlukan untuk memastikan apakah sebuah string termasuk palindrom.

Dengan demikian, bahasa palindrom dikategorikan ke dalam bahasa bebas konteks dalam hierarki Chomsky.

2.2. Pushdown Automata (PDA)

Bahasa Bebas Konteks dapat dikenali oleh mesin yang lebih kuat dari FA, yaitu Pushdown Automata (PDA). PDA adalah sebuah Finite Automata yang dilengkapi dengan sebuah memori tambahan berupa stack.

Cara kerja PDA untuk mengenali bahasa palindrom adalah sebagai berikut:

1. Tahap Mendorong (Push**):** Mesin akan membaca paruh pertama dari string input. Setiap simbol yang dibaca akan didorong (push) ke dalam stack.
2. Tahap Transisi Tengah: Mesin secara non-deterministik menebak titik tengah string. Untuk palindrom ganjil, mesin akan melewati satu simbol tengah tanpa operasi stack. Untuk palindrom genap, tahap ini dilewati.
3. Tahap Membaca (Pop): Mesin membaca paruh kedua dari string. Untuk setiap simbol yang dibaca, mesin akan mengambil (pop) simbol teratas dari stack.
4. Pencocokan: Simbol yang dibaca dari input dicocokkan dengan simbol yang di-pop dari stack. Jika tidak cocok, string ditolak.
5. Penerimaan (Acceptance): Jika seluruh string input telah habis dibaca dan stack dalam keadaan kosong, maka string tersebut diterima sebagai palindrom.

3. Implementasi

3.1. Program

Kode ini mencerminkan cara kerja PDA menggunakan stack sebagai memori sementara untuk menyimpan paruh pertama, lalu mencocokkannya secara terbalik pada paruh kedua string.

```python
def is_palindrome_pda(s):
    stack = []
    n = len(s)
    mid = n // 2

    for i in range(mid):
        stack.append(s[i])

    start_pop = mid + 1 if (n % 2 == 1) else mid

    for i in range(start_pop, n):
        if not stack:
            return False
        top = stack.pop()
        if s[i] != top:
            return False

    return len(stack) == 0
```

Gambar 3.1 Algoritma Palindrom

3.2. Penjelasan Algoritma

1. Push
   Kita baca paruh pertama string (s[0] sampai s[mid-1]) dan push setiap karakter ke dalam stack.

2. Titik Tengah
   Jika panjang string ganjil, kita lewati satu karakter tengah (start_pop = mid + 1), karena karakter itu tidak perlu dicocokkan. Jika genap, langsung mulai pop dari posisi mid.

3. Pop dan Pencocokan
   Untuk setiap karakter sisa pada paruh kedua (s[start_pop] sampai akhir), kita pop satu karakter dari stack dan bandingkan. Jika ada yang tidak cocok, langsung tolak.

4. Acceptance
   Jika seluruh input diproses dan stack kembali kosong, maka string tersebut diterima sebagai palindrom.

4. Pengujian dan Hasil

Berikut adalah hasil pengujian program menggunakan beberapa contoh string untuk memverifikasi kebenarannya.

```python
if __name__ == "__main__":
    test_strings = [
        "level",
        "madam",
        "kodok",
        "12321",
        "robot",
        "palinilap",
        "abccba",
        "abca"
    ]

    for w in test_strings:
        result = is_palindrome_pda(w)
        print(f"'{w}': {'PALINDROM' if result else 'BUKAN palindrom'}")
```

Gambar 4.1 Pengujian

Hasil pengujian menunjukkan bahwa program berhasil menerima semua string yang termasuk dalam bahasa palindrom (termasuk yang mengandung spasi, tanda baca, dan perbedaan kapital) dan menolak string yang bukan palindrom.

```
'level': PALINDROM
'madam': PALINDROM
'kodok': PALINDROM
'12321': PALINDROM
'robot': BUKAN palindrom
'palinilap': PALINDROM
'abccba': PALINDROM
'abca': BUKAN palindrom
```

Gambar 4.2 Hasil Pengujian

5. Kesimpulan

Dalam laporan ini telah ditunjukkan bahwa bahasa palindrom, yakni himpunan string yang terbaca sama dari depan maupun belakang, termasuk dalam kelas Bahasa Bebas Konteks (Context-Free Language) dalam Hierarki Chomsky. Keterbatasan memori Finite Automata membuatnya tidak mampu mengenali palindrom, sedangkan Pushdown Automata dengan tambahan stack mampu "mengingat" paruh pertama string dan mencocokkannya secara terbalik pada paruh kedua. Implementasi simulasi PDA dalam Python berhasil mereplikasi logika pengenalan palindrom melalui operasi push–pop pada stack dan penanganan titik tengah string, meski tanpa membangun state machine secara eksplisit. Program tersebut telah berfungsi sebagai *decider* untuk alfabet $\Sigma = \{a,\ldots,z, 0,\ldots,9\}$, sesuai dengan tujuan awal, dan

membuktikan efektivitas pendekatan teori automata dalam aplikasi praktis pengenalan bahasa formal.