Problem LCS dengan Solusi Rekursif, Memoisasi, dan DP

Suryana Setiawan

Masalah 6: The Longest Common Subsequence

- Diberikan dua string, misalnya “BANDUNG” dan “BUNDAKANDUNG”.
  - Pencocokan **B**AND**UNG** dan **BUN**DAKANDUN**G** pada yang bold memberikan tingkat kesamaan 4 karakter (57.1% dari string BANDUNG)
  - Pencocokan **BANDUNG** dan **B**UNDAK**ANDUNG** memberikan tingkat kesamaan 7 karakter (100% dari string BANDUNG).
- Cari pencocokan dengan tingkat kesamaan terbesar.
- Masalah ini sering ditemukan dalam pembuatan *search engine* (mencari substring yang mirip dengan tingkat kemiripan > 80% dari panjang string), plagiarism detection, dll.

LCS Sebagai Masalah Rekursif

- **Definisi Input**: Dua string *a*[0..*n-1*] dan *b*[0..*m-1*], mencari *C*(*n*-1, *m*-1)
- **Sudut pandang rekursif**:
  - jika prefiks *a*[0..*i-1*] dan prefix *b*[0..*j*-1] memiliki LCS berharga len sementara a[i] == b[j] maka prefiks *a*[1..*i*] dan prefix *b*[1..*j*] memiliki LCS berharga len+1
  - Tapi jika a[i] != b[j] maka prefiks *a*[0..*i*] dan prefix *b*[0..*j*] memiliki LCS maksimum dari antara LCS prefiks *a*[0..*i-1*] dan prefix *b*[0..*j*] dengan LCS prefiks *a*[0..*i*] dan prefix *b*[0..*j-1*]
- **Rekurensi (formulasi rekursif)**:

$$C(i,j) = \begin{cases} 0, & \text{jika } i \text{ atau } j < 0 \\ C(i-1, j-1) + 1, & \text{jika } i, j \geq 0 \text{ dan } a[i] = b[j] \\ \max\{C(i-1, j), C(i, j-1)\}, & \text{jika } i, j \geq 0 \text{ dan } a[i] \neq b[j] \end{cases}$$

Solusi Rekursif Naif

```
int rekursifNaifLSC(int n,int m){
 if (n < 0 || m < 0) return 0;
    if (a[n] == b[m])
        return rekursifNaifLSC(n-1,m-1)+1;
    else
        return max(rekursifNaifLSC(n-1, m),
                   rekursifNaifLSC(n, m-1));
}
```

Solusi Rekursif dengan Memoisasi

```
int rekursifMemoLSC(int n,int m){

    if (n < 0 || m < 0) return 0;
    if (memo[n][m] == -1) {
        if (a[n] == b[m])
            memo[n][m] = rekursifMemoLSC(n-1,m-1)+1;
        else
            memo[n][m] = max(rekursifMemoLSC(n-1, m),
                             rekursifMemoLSC(n, m-1));
    }
    return memo[n][m];
}
```

Solusi Non-rekursif dengan DP

- *Baris 0 dan kolom 0 table DP akan digunakan Khusus sebagai sentinel (bemper) yang diinisialisasi 0 agar algoritma tidak mengakses indeks baris/kolom negatif. Jadi iterasi pada table dimulai dari 1 sementara data string tetap a[0..n-1] dan b[0..m-1].*

```
int DP_LSC(int n,int m){
    for (int i = 0; i <= n; i++) DP[i][0] = 0; //sentinel
    for (int i = 0; i <= m; i++) DP[0][j] = 0; //sentinel
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            if (a[i-1] == b[j-1])
                DP[i][j] = DP[i-1][j-1]+1;
            else
                DP[i][j] = max(DP[i-1][j],DP[i][j-1]);
    return DP[n][m];
}
```

Contoh Kasus

		B	U	N	D	A	K	A	N	D	U	N	G
	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	1	1	1	1	1	1	1	1	1	1	1	1
A	0	1	1	1	1	2	2	2	2	2	2	2	2
N	0	1	1	2	2	2	2	2	3	3	3	3	3
D	0	1	1	2	3	3	3	3	3	4	4	4	4
U	0	1	2	2	3	3	3	3	3	4	5	5	5
N	0	1	2	3	3	3	3	3	4	4	5	6	6
G	0	1	2	3	3	3	3	3	4	4	5	6	7