Penerapan Algoritma Program Dinamis pada Pendeteksian Plagiarisme Pekerjaan Mahasiswa

Grace Claudia - 13520078
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520078@std.stei.itb.ac.id

***Abstrak*—Pandemi COVID-19 memaksa masyarakat dunia untuk beradaptasi terhadap keadaan, tak terkecuali kegiatan pembelajaran baik sekolah maupun universitas. Untuk menekan laju penyebaran virus yang ada, kegiatan pembelajaran baik pengumpulan tugas dan ujian di universitas maupun sekolah dilaksanakan secara daring. Karena dilaksanakan secara daring, banyak mahasiswa yang melakukan tindakan kecurangan. Tindakan kecurangan yang biasanya dilakukan diantaranya menyalin pekerjaan rumah, ataupun ada juga yang nekad untuk menyalin jawaban pada saat kuis maupun ujian. Mencontek merupakan tindakan yang tidak terpuji dan mencerminkan pengkhianatan baik kepada diri sendiri maupun orang lain. Untuk meminimalisir terjadinya tindakan pelanggaran ini, perlu adanya tindakan penanganan serius untuk memberikan konsekuensi terhadap pihak terkait. Untuk mendeteksi terjadinya kecurangan ini, kita dapat memanfaatkan konsep program dinamis dalam membangun program yang dapat mendeteksi kemiripan pekerjaan mahasiswa. Pemanfaatan algoritma program dinamis akan dilakukan dengan menghitung longest common subsequence antar pekerjaan mahasiswa terkait.**

***Kata kunci*—plagiarisme, program dinamis, longest common subsequence**

I. PENDAHULUAN

Menurut Peraturn Menteri Pendidikan Nasional Republik Indonesia Nomor 17 Tahun 2020, tindakan plagiarisme diartikan sebagai perbuatan secara sengaja maupun tidak sengaja untuk memperoleh kredit atas pekerjaan orang lain yang diakui sebagai karyanya sendiri dengan mengutip sebagian atau seluruh karya sumber. Masalah terkait plagiarisme merupakan masalah yang cukup sering terjadi di dunia akademik. Tindakan ini memang memberikan kemudahan dan kepraktisan untuk melakukan sesuatu dalam waktu yang singkat. Namun, selain berdampak buruk ke orang lain yang menjadi sumber yang diplagiasi, hal ini juga berdampak buruk bagi pencontek itu sendiri karena dapat membuat reputasi diri buruk, menurunkan percaya diri, menanamkan sifat pemalas, menghambat kreativitas, dan dapat juga tersandung kasus hukum.

Untuk mengurangi tindakan kecurangan tersebut, perlu adanya konsekuensi yang dibuat kepada para pihak yang terkait. Langkah yang dapat dilakukan adalah membandingkan kemiripan sebuah karya dengan karya lainnya. Apabila kemiripan merupakan 70-80% maka dapat dianggap karya tersebut merupakan kutipan orang lain. Pendeteksian ini dapat dilakukan secara manual, tetapi tentunya sangat memakan waktu. Bagaimana cara kita agar dapat dengan mudah dan cepat mendeteksi plagiarisme seseorang? Untuk idenya adalah dengan melakukan pencocokan karya yang satu dengan karya yang lain dengan mengubah semua karya tersebut menjadi string yang panjang lalu dilakukan perbandingan antara keduanya. Nantinya, akan dihitung tingkat kemiripannya berdasarkan algoritma penyelesaian tertentu, lalu dalam suatu tingkaAda banyak cara penyelesaian untuk mendeteksi plagiarisme ini, bisa dengan menggunakan konsep string matching, program dinamis, dan lain lain. Algoritma string matching yang terkenal untuk melakukan perbandingan pencocokan antar dua string diantaranya adalah Boyer Moore, dan Knuth-Morris-Pratt.

Makalah kali ini akan difokuskan untuk membahas penyelesaian pendeteksian plagiarisme dengan menggunakan konsep program dinamis. Untuk penyelesaian dengan program dinamis, kita akan memanfaatkan penyelesaian persoalan terkenal yaitu Longest Common Subsequence (LCS). LCS ini sebenarnya dapat diselesaikan dengan cara naïve, tapi tentunya memakan waktu yang sangat lama. Dengan program dinamis, program LCS yang dibuat dapat bekerja dengan lebih efektif dan efisien.

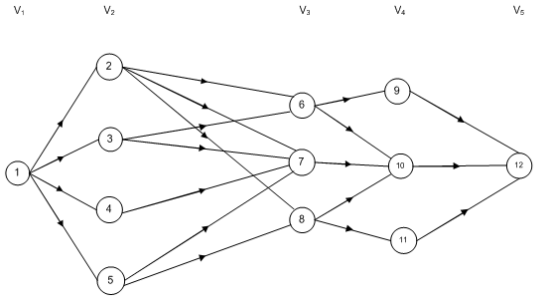
II. DASAR TEORI

A. *Program dinamis*

Program Dinamis (*dynamic programming*) merupakan suatu metode penyelesaian masalah dengan memecahkan solusi menjadi sebuah kumpulan tahap sedemikian rupa sehingga solusi persoalan akan dipandang sebagai serangkaian keputusan yang saling berkaitan. Pada program dinamis, masalah akan diselesaikan dengan membagi persoalan menjadi beberapa upa persoalan lain yang saling berhubungan yang nantinya akan disimpan dalam suatu larik. Istilah program di pemrograman dinamis ini tidak ada kaitannya dengan pemrograman, sedangkan untuk dinamis sendiri didapatkan dari tahap penyelesaian masalah yang menggunakan tabel yang dapat berkembang. Biasanya, Program dinamis ini digunakan untuk menyelesaikan persoalan-persoalan optimasi berupa maksimasi atau minimasi. Algoritma program dinamis memiliki kesamaan dengan algoritma greedy karena menyelesaikan persoalan optimasi, tetapi ada perbedaan antara algoritma greedy dengan program dinamis yaitu greedy hanya satu rangkaian keputusan yang dihasilkan, sedangkan program dinamis memiliki lebih dari satu keputusan.

Program dinamis ini berpedoman pada prinsip optimalitas dimana "jika solusi total optimal, maka bagian solusi sampai tahap ke-k juga optimal". Ini berarti, apabila kita berada di tahap k ke k+1, kita dapat menggunakan hasil optimal tahap k tanpa harus kembali ke awal. Ongkos pada tahap k+1 dihitung sebagai ongkos yang dihasilkan pada tahap k ditambah dengan ongkos dari tahap k ke tahap k+1 atau Ck,k+1. Ada beberapa karakteristik Persoalan yang dapat dikategorikan sebagai persoalan program dinamis, diantaranya sebagai berikut:

1. Persoalan dapat dibagi menjadi tahapan yang pada masing-masing tahapannya dapat diambil satu keputusan
2. Masing-masing tahapan yang ada terdiri dari sejumlah kemungkinan masukan yang ada pada suatu tahap atau yang disebut juga status. Penggambaran status dan simpul dapat digambarkan dengan graf multitahap. Tiap simpul pada graf menyatakan status, sedangkan v1, v2, … vn menyatakan tahap.



Gambar 1. Graf multitahap

(Sumber: http://ryonnatalius.com/)

3. Hasil keputusan pada setiap tahap ditransformasikan dari status terkait ke status selanjutnya di tahap selanjutnya.
4. Pertambahan cost pada suatu tahap meningkat secara teratur
5. Ongkos suatu tahap bergantung kepada cost sebelumnya yang telah dilalui
6. Pengidentifikasian keputusan terbaik dilakukan secara rekursif untuk setiap status
7. Berlakunya prinsip optimasi baik minimasi maupun maksimasi pada persoalan tersebut

Dalam melakukan pemecahan persoalan dengan algoritma program dinamis, kita dapat memecahkannya menjadi beberapa tahap:

1. Penentuan karakteristik dari persoalan struktur solusi yang optimal
2. Menentukan suatu nilai sebagai solusi optimal secara rekursif
3. Perhitungan nilai solusi optimal menggunakan pendekatan yang dipilih. Pendekatan ini dapat berupa pendekatan bottom-up atau top-down
4. Rekonstruksi solusi optimal secara mundur (opsional)

Ada dua pendekatan dalam penyelesaian program dinamis yaitu:

- Top-down approach

  Algoritma ini umumnya dilakukan secara rekursif dan lebih intuitif untuk diimplementasikan karena umumnya persoalannya terletak pada pengidentifikasian pola dalam suatu algoritma dan mengubahnya menjadi pola solusi program dinamis. Top-down approach disebut juga memoization

- Bottom-up approach

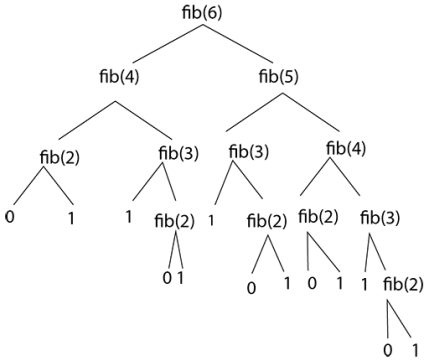
  Algoritma ini umumnya dilakukan secara iterative tetapi kurang intuitif, sehingga memerlukan pemahaman lebih di upa persoalan yang lebih kecil terlebih dahulu lalu digabungkan dengan nilai yang lebih kecil untuk solusi gabungan. Bottom-up approach ini disebut juga tabulation method

  Pada penyelesaian masalah plagiarisme kali ini akan dilakukan dengan menggunakan top-down approach(memoization).

B. *Top-down approach (Memoization)*

Memoization merupakan Teknik pemrograman yang mempercepat performa dengan caching return value sebagai hasil dari suatu fungsi. Fungsi yang "memoized" akan langsung mengeluarkan output berupa value solusi dari tahapan tersebut dari hasil yang pernah dicapai sebelumnya.

Memoization adalah bentuk spesifik dari chacing yang sangat berguna untuk fungsi yang dilakukan pemanggilan berkali-kali dengan argument yang sama. Teknik ini sangat menaikan efisiensi dan juga mengurangi kerja CPU. Kita dapat melihat contohnya dari sebuah pemanggilan fungsi fibonaci.



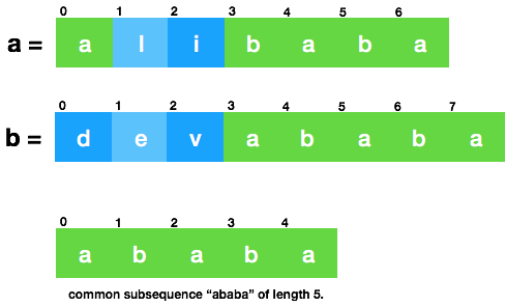
Gambar 2. Graf pemanggilan fungsi Fibonacci

(Sumber: https://paulmouzas.github.io/)

Apabila kita membuat solusi naif dari algoritma fibonacci, maka akan memanggil fungsi fibonacci secara berulang kali, dapat dilihat dalam menghitung fib(6) kita memerlukan pemanggilan fib(5) satu kali, fib(4) 2 kali, fib(3) 3 kali, dan fib(2) 4 kali. Ini merupakan ide dasar dari penyelesaian secara memoization, karena pemanggilan fungsi fib(4), fib(3), fib(2) dilakukan secara berulang maka idenya adalah menaruh hasil tersebut ke suatu tempat penyimpanan dimana nanti pada saat pemanggilan fungsi tersebut tidak perlu dikalkulasi ulang, melainkan tinggal membaca nilai yang menunjuk ke hasil dari kalkulasi fungsi tersebut. Untuk algoritma fibonacci naif, fibonaci kompleksitasnya memakan $O(2^n)$ karena berupa naik secara eksponensial. Algoritma fibonaci yang menggunakan Teknik memorization memakan waktu hanya O(2n+1) untuk pemanggilan fungsi 2n, dan 1 untuk pamanggilan pertama kali. Algoritma ini tentu sangat menaikkan efisiensi dalam pemanggilan fungsi yang dilakukan berulang kali.

C. *Longest common subsequence problem*

Longest common subsequence problem merupakan masalah yang cukup popular di dunia ilmu computer. Problem statement dari masalah ini adalah *"*Diberikan 2 sequence, carilah ukuran paling Panjang dari subsequence yang ada di kedua sequence tersebut.*"* Subsequence merupakan sebuah sequence yang muncul dengan urutan yang sama tetapi tidak harus secara berurut. Misalnya, LCS untuk "AEDFHR" dengan "ABCDGH" adalah "ADH" dengan panjang 3, LCS untuk "GXTXAYB" dan "AGGTAB" adalah "GTAB" dengan panjang 4.



Gambar 3. Contoh Persoalan LCS

(sumber: https://kodebinary.com/)

D. *Plagiarisme dan Algoritma Pendeteksi Plagiarisme*

Plagiarisme dapat dideteksi dengan beragam konsep salah satunya memanfaatkan memoization approach dari dynamic programming. Banyak juga cara untuk menentukan tingkat plagiarisme, tetapi pada kali ini penulis akan memberikan langkah salah satu cara penyelesaian metode pendeteksi plagiarisme.

Misalkan ada pekerjaan orang A dan pekerjaan orang B, maka untuk pendeteksian plagiarismenya akan dilakukan tahapan sebagai berikut:

1. Pencarian Longest Common Subsequence untuk kedua pekerjaan
2. Dari hasil LCS, akan diketahui hasil string LCS kedua pekerjaan tersebut dan akan dihitung pula panjang LCS yang telah dihasilkan
3. Panjang LCS dibagi panjang string pekerjaan orang A akan menggambarkan persentase dari pekerjaan orang A yang juga ditemukan di pekerjaan orang B.
4. Jika persentase cukup tinggi, kita dapat mengasumsikan identifikasi adanya plagiarisme. Standar persentase ini berbeda-beda ini mengikuti kebijakan setiap universitas atau pekerjaan yang diberikan. Menurut Ristekdikti maksimum angka batass untuk plagiarisme adalah 25%

   Perlu diingat pendeteksian ini hanya bersifat dugaan, jika ingin hasil sangat akurat maka perlu adanya investigasi mendalam terkait pekerjaan antar mahasiswa.

III. IMPLEMENTASI DAN EKSPERIMEN

A. *Ide penyelesaian dengan naive solution*

Sebenarnya penyelesaian LCS problem ini dapat kita selesaikan tanpa dynamic programming. Idenya adalah untuk mengenerate semua subsequence dari kedua string yang sedang dibandingkan lalu mencari mana yang menghasilkan subsequence yang sama. Memang seperti yang kita tahu brute force selalu menghasilkan solusi, tetapi tentu akan memakan waktu yang sangat banyak karena harus mengenerate semua subsequence string dari kedua string yang dibandingkan, selajutnya harus membandingkan satu satu antar subsequence kedua string. Kompleksitas waktu dengan naïve solution memakan waktu yang eksponensial untuk mengenerate semua subsequence yaitu 2^n. Kita akan mencoba pendekatan lain untuk menyelesaikan problem ini.

B. *Pengidentifikasian pemecahan masalah pada longest common subsequence problem*

Kita akan melakukan penyelesaian masalah LCS ini dengan menggunakan pendekatan recursive. Hanya sebagai contoh, kita akan melakukan perbandingan antara string pendek, "AGAB" dengan "ACBA". Kita akan mempunyai fungsi yang

menerima parameter string1 dan string2. Kita dapat membaginya menjadi 2 kasus:

1. Kasus 1 (base case): string 1 atau string 2 ada yang sudah habis diiterasi. Ditandai dengan panjang string salah satunya adalah 0.
2. Kasus 2: kedua huruf alfabet didepan sama

   Untuk kasus ini, maka huruf alfabet tersebut pasti merupakan jawaban.
3. Kasus 3: kedua huruf alfabet didepan berbeda

   Untuk kasus ini, maka akan dibandingkan pemanggilan fungsi LCS:

   1. LCS string1 dari index 0 dan string2 dari index 1 sampai terakhir
   2. LCS string1 dari index 1 dan string2 dari index 0 sampai terakhir

      Selanjutnya maka akan dicari yang lebih besar, yang lebih besar akan menjadi hasil untuk iterasi tersebut

C. *Implementasi penyelesaian longest common subsequence problem dengan rekursif biasa*

Karena kita sudah mengidentifikasinya diatas, maka kita akan mengidentifikasi returnnya berdasarkan kasus-kasus yang ada. Return untuk setiap kasus:

- Kasus 1: karena basecase, maka akan direturn string kosong yaitu “”
- Kasus 2: karena alfabet sama dan pasti itu merupakan jawaban maka akan mereturn s[0] + LCS(s1[1:], s2[1:]) yaitu alfabet yang sama dan pencarian LCS dilanjutkan dengan menggeser 1 index
- Kasus 3: karena alfabet berbeda, maka akan dilakukan perbandingan. Akan ditaruh di variable baru yang bernama resultA dan resultB.

  resultA = LCS(s1[1:], s2)

  resultB = LCS(s1, s2[1:])

  Jika panjang resultA lebih besar dari result maka akan direturn resultA, begitupun sebaliknya, contoh code dalam pythonnya adalah sebagai berikut:

```python
def LCSver1(s1,s2):
    if len(s1)==0 or len(s2)==0:
        return ""
    if (s1[0] == s2[0]):
        return s1[0] + LCSver1(s1[1:], s2[1:])
    resultA = LCSver1(s1[1:], s2)
    resultB = LCSver1(s1, s2[1:])
    if len(resultA) > len(resultB):
        return resultA
    else:
        return resultB
```

Gambar 4. Penyelesaian LCS dengan algoritma rekursi biasa

(sumber: dokumen pribadi penulis)

D. *Implementasi penyelesaian longest common subsequence problem dengan optimasi rekursif biasa*

Setelah berhasil membangun program untuk menyeleesaikan LCS, ternyata masih ada hal yang dapat dioptimasi. Hal ini merupakan penanganan saat terjadinya pemanggilan fungsi. Karena parameter saat memanggilnya merupakan string hasil indexing dari string tersebut, (contoh: s1[1:]) string tersebut diulang-ulang pengkonstruksiannya pada saat pemanggilan fungsi. Ide dari penyelesaian masalah ini merupakan penambahan parameter pada fungsi agar men-*track* index yang sekarang sedang di cek, untuk itu akan ditambahkan di parameter yaitu argument i1 untuk index string1 dan i2 untuk index string2. Tujuan dari penambahan index pada fungsi adalah agar tidak perlu meng-*construct* string dari awal.

```python
def LCSver2(s1,s2, i1,i2):
    if i1 == len(s1) or i2 == len(s2):
        return ""
    if (s1[i1] == s2[i2]):
        return s1[i1] +
LCSver2(s1,s2, i1+1, i2+1)
    resultA = LCSver2(s1, s2, i1+1, i2)
    resultB = LCSver2(s1, s2, i1, i2+1)
    if len(resultA) > len(resultB):
        return resultA
    else:
        return resultB
```

Gambar 5. Penyelesaian LCS dengan algoritma rekursi biasa dengan optimasi

(sumber: dokumen pribadi penulis)

Untuk pemanggilan fungsi pada pertama kali maka akan bernilai 0 untuk argumen i1 dan i2.

```python
res2 = LCSver2(string1,string2, 0,0)
```

Gambar 6. Pemanggilan LCS untuk penyelesaian dengan rekursi yang telah dioptimasi

(sumber: dokumen pribadi penulis)

E. *Implementasi penyelesaian longest common subsequence problem dengan program dinamis*

Walaupun sudah mengoptimasi program rekursi penyelesaian masalah LCS, ternyata masih ada hal lagi yang dapat dioptimasi. Jika diperhatikan, algoritma LCS rekursi yang sebelumnya memanggil fungsi dengan argumen yang sama secara berulang. Hal ini sama dengan yang telah dipaparkan sebelumnya pada gambar 2 kasus Fibonacci. Untuk itu, kita mengimplementasikan memoization yaitu konsep dari program dinamis. Konsep ini membuat program tidak perlu mengkomputasi ulang jika argumen yang sama memanggil sebuah fungsi melainkan hanya perlu membaca dari suatu tempat penyimpanan. Untuk penyimpanannya sendiri akan

digunakan matrix yang memiliki ukuran panjang string 1 x ukuran panjang string 2. Perubahan yang terjadi diantaranya adalah:

1. Penambahan argument pada fungsi yaitu memo sebagai matrix tempat penyimpanan yang diinisasi terlebih dahulu dengan undefined (atau None pada python) yang mengindikasi pemanggilan fugsi dengan argument tersebut belum pernah dilakukan
2. Pada line 4 ditambahkan pembacaan nilai matrix apabila sebelumnya fungsi telah dikomputasi dengan argumen yang sama sehingga akan mereturn pembacaannya
3. Pada line 7, ada juga perubahan yang tadinya langsung mereturn hasil, sekarang menyimpan dulu di memo yang nantinya tinggal dibaca jika fungsi dipanggil kembali. Yang direturn adalah memonya.
4. Pada line 15, result untuk setiap opsi masih disimpan di variable baru yaitu resultA dan resultB, tapi selanjutnya result terpilih akan distore di memo, baru mereturn memo index tersebut

```
1  def LCSDP(s1,s2, i1, i2, memo):
2      if len(s1) == i1 or len(s2) == i2:
3          return ""
4      if memo[i1][i2] is not None:
5          return memo[i1][i2]
6      if s1[i1] == s2[i2]:
7          memo[i1][i2] = s1[i1] +
   LCSDP(s1,s2,i1+1, i2+1, memo)
8          return memo[i1][i2]
9      resultA = LCSDP(s1,s2,i1+1
   , i2, memo)
10     resultB = LCSDP(s1,s2,i1, i2+1
   , memo)
11     if len(resultA) > len(resultB):
12         result = resultA
13     else:
14         result = resultB
15     memo[i1][i2] = result
16     return memo[i1][i2]
```

Gambar 7. Penyelesaian LCS dengan konsep dynamic programming yaitu memoization

(sumber: dokumen pribadi penulis)

Untuk pemanggilan fungsi, terlebih dahulu harus dibuat matrix berelemen none dengan ukuran baris x kolom berupa panjang string1 x string2. None ini menyatakan belum pernah ada fungsi yang terpanggil dengan argument tersebut. Nantinya matrix yang berupa memo ini akan di pass di parameter fungsi saat pemanggilan

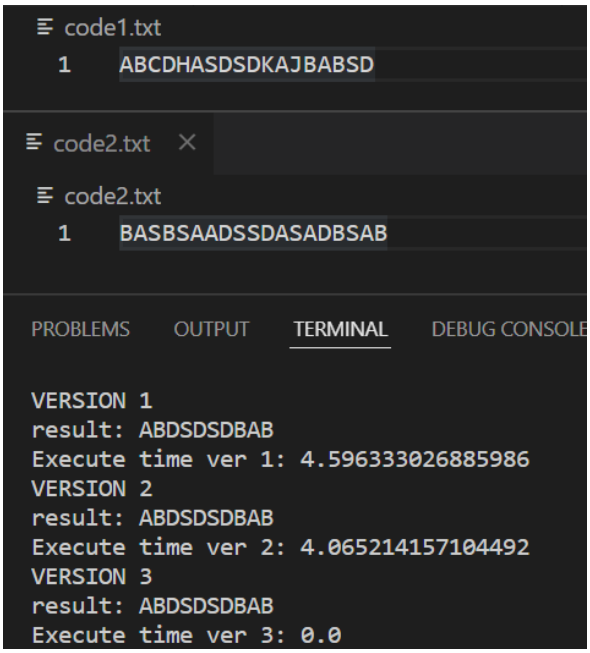
```
memo = [[None for i in range(cols)] for j in range(rows)]
res3 = LCSDP(code1,code2,0,0,memo)
```

Gambar 8. Pemanggilan LCS untuk konsep dynamic programming

(sumber: dokumen pribadi penulis)

F. Perbandingan kompleksitas penyelesaian persoalan LCS dengan rekursi biasa dan program dinamis

Misalkan panjang string satu adalah p dan panjang string kedua adalah q, maka untuk LCS rekursif biasa akan memakan waktu $O(2^{n+m})$ ini terjadi karena perlu adanya iterasi ke semua kemungkinan index. Di sisi lain, untuk LCS program dinamis hanya memerlukan waktu O(n+m), jauh lebih cepat disbanding LCS rekursif biasa. Hal ini dikarenakan adanya konsep memoization yang menyebabkan tidak perlu komputasi ulang saat pemanggilan fungsi LCS.



Gambar 9. Pembuktian Pengoptimasian program menggunakan Program dinamis jauh lebih cepat dibandingkan dengan yang lainnya

(Sumber: dokumen pribadi penulis)

Terbukti pada gambar bahwa optimasi code mempercepat execute time. Program dengan dynamic programming jauh lebih cepat dibandingkan yang lain.

G. Deteksi Plagiarisme dengan program dinamis longest common subsequence

Pendeteksian plagiarisme akan dilakukan dengan panjang LCS / dengan stringA yang akan menghasilkan tingkat kesamaan stringA pada stringB. Untuk thresholdnya sendiri akan penulis tentukan yaitu >70% akan dikategorikan melakukan plagiarisme. Untuk membaca file maka akan dibandingkan 2 file txt lalu dibaca pada program. Program pembacaan file txt adalah sebagai berikut:

```python
#menginput nama file
file1 = input("Input first filename: ")
#open text file di mode read
string1_file = open("./"+file1, "r")
#read whole file to a string
string1 = string1_file.read()
#close file setelah dibaca
string1_file.close()

file2 = input("Input second filename: ")
string2_file = open("./"+file2, "r")
string2 = string2_file.read()
string2_file.close()
```

Gambar 10. Program input file untuk kedua pekerjaan yang akan dibandingkan

(Sumber: dokumen pribadi penulis)

Untuk plagiarism checker, maka dibuat fungsi berikut:

```python
def plagiarism_checker(LCS, string, maximumRate):
    plagiarismSuspect = len(LCS)/len(string)
    if (plagiarismSuspect>maximumRate):
        print("Ada indikasi plagiarisme")
    else:
        print("Tidak ada indikasi plagiarisme ditemukan")
```

Gambar 11. Program input file untuk pengecekan plagiarisme

(Sumber: dokumen pribadi penulis)

LCS menerima string yang berupa longest subsequence dari kedua string yang dibandingkan, string menerima salah satu string yang dibandingkan, dan maximumRate menerima maximumRate of palgiarism yang ditoleransi karena tidak ada batas pasti plagiarisme.

H. Pengujian program

PENGUJIAN I

Untuk pengujian program LCS, maka akan diambil contoh untuk pengujian string yang pendek terlebih dahulu

Text1.txt STONE
Text2.txt LONGEST

Akan dilakukan pengisian tabel dengan cara sebagai berikut:

1. Mengambil character pertama pada string1 dan string2 lalu apabila sama, maka akan menambah 1 dari value sebelumnya ke tabel
2. Jika tidak ada kecocokan maka ambil maksimum dari baris atau kolom sebelumnya
3. Ulang langkah ke dua untuk setiap string dengan penambahan secara bergantian sampai semua kolom teriterasi
4. Untuk mengetahui hasilnya, maka akan backtrack sampai value 0(tidak ikut sebagai hasil) dari table yang dihasilkan dengan cara mulai dari pojok kanan bawah kita akan melakukan pengecekan ke baris diatasnya terlebih dahulu, apabila sama maka backtrack ke atas, lalu dilakukan pengecekan ke kiri, jika sama maka akan backtrack ke kiri dengan mengganti value sebelumnya, apabila kita tidak menemui value yang sama untuk baris maupunkolom sebelumnya maka akan melakukan iterasi secara diagonal ke kiri atas baris dan kolom sebelumnya

	K0	K1	K2	K3	K4	K5	K6	K7
B0	-	L	O	N	G	E	S	T
B1	S	0	0	0	0	0	1	1
B2	T	0	0	0	0	0	1	2
B3	O	0	1	1	1	1	1	2
B4	N	0	1	2	2	2	2	2
B5	E	0	1	2	2	3	3	3

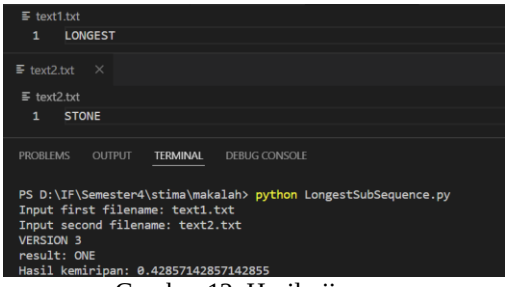
Contoh Step 2:

- Untuk block B1 K6, dilakukan perbandingan terhadap string “S” dengan “LONGES” yang memiliki kesamaan 1 yaitu “S” sehingga ditambahkan 1 untuk blok tersebut
- Untuk block B1 K7, dilakukan perbandingan terhadap string “S” dengan “LONGEST” yang memiliki kesaaman tetap 1 yaitu “S” sehingga ditambahkan 1 untuk blok tersebut
- Untuk block B5 K5, dilakukan perbandingan terhadap string “STONE” dengan “LONGEST” yang memiliki kesamaan 3 yaitu “ONE” maka blok tersebut akan berisi 3

Backtrack yang dilakukan:

1. T
2. S
3. E
4. GE
5. NE
6. ONE

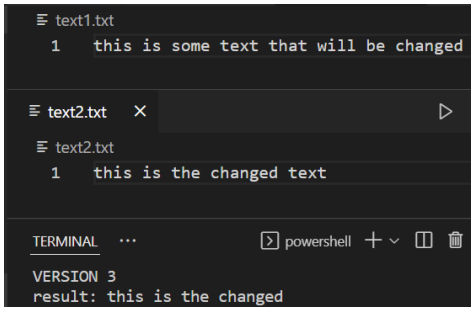
Hasil ini bersesuaian dengan yang diperoleh di program yaitu “ONE”



Gambar 12. Hasil uji program

(Sumber: dokumen pribadi penulis)

PENGUJIAN II



Gambar 13. Hasil uji program

(Sumber: dokumen pribadi penulis)

Untuk pengerjaan tabel:

	K0	K1	K2	K3	K4	K5	K6	K7	K8
B0	-	this	is	some	text	that	will	be	changed
B1	this	1	1	1	1	1	1	1	1
B2	is	1	2	2	2	2	2	2	2
B3	the	1	2	2	2	2	2	2	2
B4	changed	1	2	2	2	2	2	2	3
B5	text	1	2	2	3	3	3	3	3

Padahal pada tabel manual kita menemukan bahwa jawabannya adalah this is changed. Mengapa bisa berbeda?
Setelah melakukan analisis lebih lanjut, yang terjadi adalah timbulnya “the” pada program kita karena pembacaan dilakukan per karakter, sehingga the di text2 dibaca di text1 sebagai t dalam text, h dalam that dan e di dalam be. Oleh karena itu program ini cocok hanya untuk perbandingan karakter pada string. Sekarang akan didevelop algo untuk perbandingan kata.

Yang kita perlukan adalah pengecekan apakah kata tersebut berada di kedua string dan juga keterurutannya yang benar, apabila benar dan terurut maka dipastikan, hasil yang telah diperoleh dari program merupakan hasilnya.

Penambahan yang dilakukan:
Akan ditmabahkan fungsi intersect untuk pengiris hasil dari string1 dan string2.

```python
def intersection(l1, l2):
    # menghasilkan list3 berupa hasil intersect list1 dan list2
    l3 = [value for value in l1 if value in l2]
    return l3

```

Gambar 14. Fungsi tambahan intersection

(Sumber: dokumen pribadi penulis)

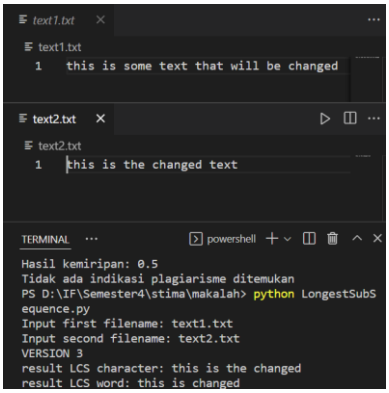
Selanjutnya ada fungsi result_check untuk mengecek apakah setiap string memiliki kata tersebut dan kebenaran urutan kemunculannya.

```python
def result_check(res, string1, string2):
    # split string menjadi array
    res = res.split(" ")
    s1 = string1.split(" ")
    s2 = string2.split(" ")

    # tempat menampung hasil
    trueRes = []

    # check index agar penemuan kata terurut
    idxS1 = []
    # hasil untuk kalimat pertama
    resS1 = []
    # membuat copy S1 yang akan dimodifikasi
    checkS1 = s1
    # melakukan iterasi sebesar ukuran result
    for i in range (len(res)):
        # jika result ada di string1
        if (res[i] in checkS1):
            # mencari letak index dalam kalimat
            idxFound = checkS1.index(res[i])
            # jika idxS1 masih kosong atau idx yang terakhir
            # disimpan pada string lebih kecil dari penemuan sekarang
            if (len(idxS1) == 0 or idxFound>idxS1[-1]):
                # index akan diappend untuk pengecekan berikutnya
                idxS1.append(idxFound)
                # result akan diappend
                resS1.append(res[i])
                # tempat index yang sudah dialkukan pengecekan
                # diganti menjadi string kosong
                checkS1[idxFound] = ""

    idxS2 = []
    resS2 = []
    checkS2 = s2
    for i in range (len(res)):
        if (res[i] in checkS2):
            idxFound = checkS2.index(res[i])
            if (len(idxS2) == 0 or idxFound>idxS2[-1]):
                idxS2.append(idxFound)
                resS2.append(res[i])
                checkS2[idxFound] = ""

    # intersect hasil dari l1 dan l2
    trueRes = " ".join(intersection(resS1, resS2))
    return trueRes

```

Gambar 15. Fungsi tambahan result_check

(Sumber: dokumen pribadi penulis)

Setelah dilakukan perubahan, hasilnya menjadi sesuai dengan yang kita harapkan. (Lihat LCS word)



Gambar 16. Hasil setelah adanya fungsi tambahan

(Sumber: dokumen pribadi penulis)

Tidak lupa juga untuk mengecek plagiarisme, perhitungan panjang string akan menjadi panjang kata dalam sebuah kalimat, perubahan fungsinya adalah sebagai berikut:

```
def word_plagiarism_checker(LCS, string, maximumRate):
    plagiarismSuspect = len(LCS.split(" "))/len(string.split(" "))*100
    print("Hasil kemiripan:", str(plagiarismSuspect)+"%")
    if (plagiarismSuspect>maximumRate):
        print("Ada indikasi plagiarisme")
    else:
        print("Tidak ada indikasi plagiarisme ditemukan")
```

Gambar 17. Perubahan perhitungan plagiarisme

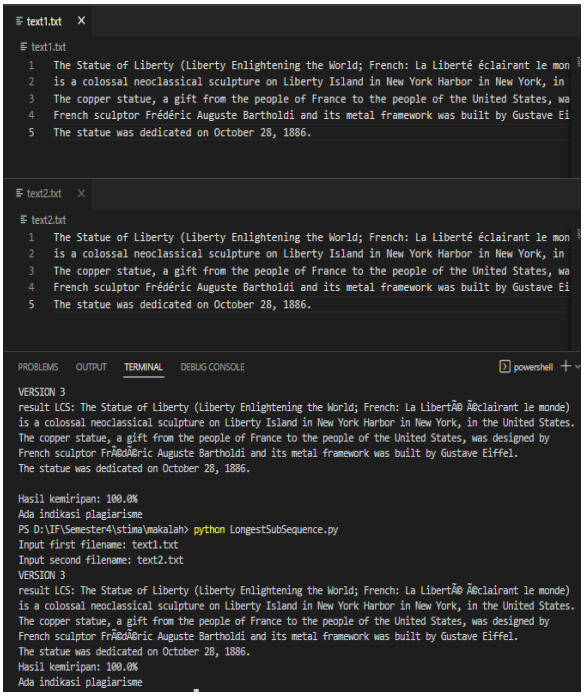
(Sumber: dokumen pribadi penulis)

PENGUJIAN AKHIR

TC 1: Kedua kata jawaban sama persis, akan menghasilkan kemiripan 100% :

Text1.txt The Statue of Liberty (Liberty Enlightening the World; French: La Liberté éclairant le monde) is a colossal neoclassical sculpture on Liberty Island in New York Harbor in New York, in the United States. The copper statue, a gift from the people of France to the people of the United States, was designed by French sculptor Frédéric Auguste Bartholdi and its metal framework was built by Gustave Eiffel. The statue was dedicated on October 28, 1886.
Text2.txt The Statue of Liberty (Liberty Enlightening the World; French: La Liberté éclairant le monde) is a colossal neoclassical sculpture on Liberty Island in New York Harbor in New York, in the United States. The copper statue, a gift from the people of France to the people of the United States, was designed by French sculptor Frédéric Auguste Bartholdi and its metal framework was built by Gustave Eiffel. The statue was dedicated on October 28, 1886.

Hasil pengujian:



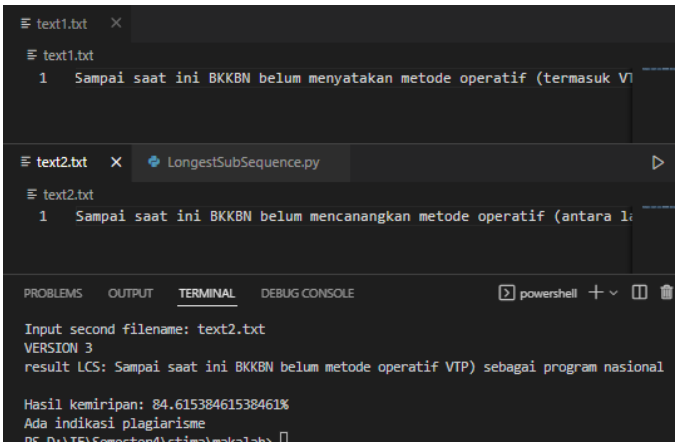
Gambar 18. Hasil Uji Dua text sama

(Sumber: dokumen pribadi penulis)

TC 2: Kedua kalimat beda beberapa, plagiarisme diasumsikan apabila ada >30% kata yang sama:

Text1.txt Sampai saat ini BKKBN belum menyatakan metode operatif (termasuk VTP) sebagai program nasional
Text2.txt Sampai saat ini BKKBN belum mencanangkan metode operatif (antara lain VTP) sebagai program nasional

Hasil pengujian:



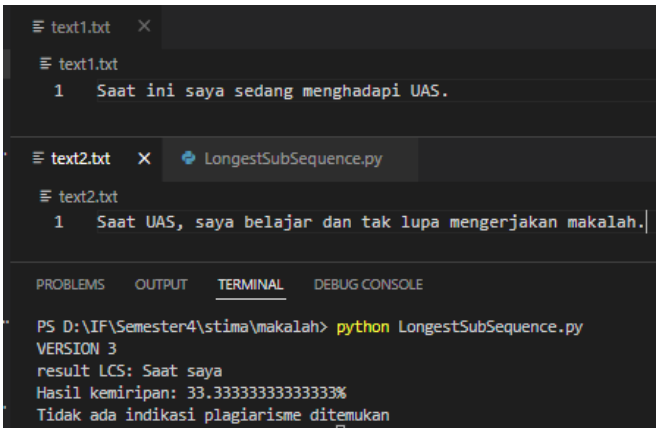
Gambar 19. Hasil uji dua text mirip

(Sumber: dokumen pribadi penulis)

TC 3: Kedua kalimat berbeda

Text1.txt Saat ini saya sedang menghadapi UAS.
Text2.txt Saat UAS, saya belajar dan tak lupa mengerjakan makalah.

Hasil pengujian:



Gambar 20. Hasil uji dua text berbeda

(Sumber: dokumen pribadi penulis)

IV. PENUTUP

A. *Kesimpulan*

Pemanfaatan Algoritma Program Dinamis (Dynamic Programming) sangat luas di dunia computer. Pada program dinamis, masalah akan diselesaikan dengan membagi persoalan menjadi beberapa upa persoalan lain yang saling berhubungan yang nantinya akan disimpan dalam suatu larik. Ada 2 approach dari dynamic programming, yaitu top-down approach dan bottom-up approach. Program dinamis ini dapat menyelesaikan persoalan LCS yang sangat populer. LCS merupakan persoalan longest common subsequence dimana persoalan ini mencari subsequence terpanjang dari kedua string yang telah diberikan. LCS dapat diselesaikan dengan berbagai cara, penyelesaian LCS dengan program dinamis merupakan salah satu yang paling efektif karena tidak perlu mengkomputasi saat pemanggilan fungsi secara rekursif dilakukan. Pemanfaaatan LCS sendiri ada banyak diantaranya untuk melakukan perbandingan, penghitung plagiarisme, dan lain lain. Perhitungan plagiarisme yang penulis lakukan disini adalah dengan mencari panjang LCS dengan panjang string pekerjaan seseorang yang akan menghasilkan tingkat plagiarisme seseorang dengan orang lain yang dibandingkan. Untuk tingkat plagiarismenya sendiri karena setiap universitas maupun tugas memiliki tingkat yang beragam maka penulis mengkategorikan plagiarisme apabila sudah lebih dari 25%.

B. *Saran*

Penulis menyarankan agar para pembaca dapat menjadikan makalah ini sebagai referensi untuk salah satu cara pendeteksian plagiarisme antar dua pekerjaan. Berikutnya, Penulis juga menyarankan agar setelah adanya indikasi plagiarisme maupun tidak adanya indikasi plagiarisme yang telah dinyatakan dari program ini, apabila ingin sangat akurat maka perlu langsung melakukan investigasi lebih dalam karena tidak mungkin program yang dibuat akurat 100%. Program hanya bersifat membantu saja. Penulis juga menyarankan agar pembaca tidak melakukan plagiarisme terhadap apapun tugas maupun ujian karena dampaknya akan merugikan diri sendiri dan orang lain.

VIDEO LINK YOUTUBE

https://youtu.be/94JOO2sVs1k

UCAPAN TERIMA KASIH

Pertama-tama, penulis mengucapkan terima kasih dan puji syukur kepada Tuhan Yang Maha Esa atas berkat dan rahmatnya, penulis dapat menyelesaikan makalah ini dengan baik dan tepat waktu. Tak lupa juga, penulis mengucapkan terima kasih kepada orang tua yang selalu mendukung penulis dalam segala kegiatan pendidikan yang dijalankan penulis. Terima kasih sebesar-besarnya juga dipanjatkan penulis kepada Pak Rinaldi Munir selaku dosen yang membimbing penulis dalam pembelajaran strategi algoritma selama satu semester ini. Terimakasih juga kepada Ibu Nur Ulfa Maulidevi dan Ibu Masayu Leylia Khodra yang sudah bekerjasama dalam keterlaksanaannya perkuliahan IF2211 dengan baik

bagi mahasiswa IF Angkatan 2020. Tak lupa juga, penulis mengucapkan terima kasih kepada teman-teman penulis yang selalu mendukung, memotivasi, dan memberi masukan dalam setiap pembelajaran pada mata kuliah IF2211 ini

REFERENSI

[1] Faizti, N. (2021, June 22). *Inilah 5 dampak plagiarisme Yang Harus Diwaspadai*. Dunia Dosen. Retrieved May 9, 2022, from https://www.duniadosen.com/dampak-plagiarisme/

[2] Informatika. (n.d.). Retrieved May 9, 2022, from https://informatika.stei.itb.ac.id/

[3] Jarednielsen. (n.d.). *What is dynamic programming? memoization and tabulation*. jarednielsencom RSS. Retrieved May 12, 2022, from https://jarednielsen.com/dynamic-programming-memoization-tabulation/

[4] *Grokking dynamic programming patterns for coding interviews - learn interactively*. Educative. (n.d.). Retrieved May 12, 2022, from https://www.educative.io/courses/grokking-dynamic-programming-patterns-for-coding-interviews

[5] *Dynamic Programming*. GeeksforGeeks. (n.d.). Retrieved May 12, 2022, from https://www.geeksforgeeks.org/dynamic-programming/

[6] *Longest common subsequence: DP-4*. GeeksforGeeks. (2022, May 5). Retrieved May 13, 2022, from https://www.geeksforgeeks.org/longest-common-subsequence-dp-4/

[7] *Longest common subsequence*. Programiz. (n.d.). Retrieved May 13, 2022, from https://www.programiz.com/dsa/longest-common-subsequence

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, 20 Mei 2022

Grace Claudia 13520078