



TUGAS AKHIR - KI141502

**DESAIN, ANALISIS DAN IMPLEMENTASI ALGORITMA
KOMPUTASI STRING DENGAN METODE DYNAMIC
PROGRAMMING DAN MEET IN THE MIDDLE PADA
PERMASALAHAN KLASIK SPOJ 9967 PLAYING WITH
WORDS**

**DEWANGGA WINASFORCEPTA WINARDI
NRP 5113 100 098**

**Dosen Pembimbing 1
Rully Soelaiman, S.Kom., M.Kom.**

**Dosen Pembimbing 2
Ir. F.X. Arunanto, M.Sc.**

**DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2017**

Halaman ini sengaja dikosongkan



TUGAS AKHIR - KI141502

**DESAIN, ANALISIS DAN IMPLEMENTASI ALGORITMA
KOMPUTASI STRING DENGAN METODE DYNAMIC
PROGRAMMING DAN MEET IN THE MIDDLE PADA
PERMASALAHAN KLASIK SPOJ 9967 PLAYING WITH
WORDS**

DEWANGGA WINASFORCEPTA WINARDI
NRP 5113 100 098

Dosen Pembimbing 1
Rully Soelaiman, S.Kom., M.Kom.

Dosen Pembimbing 2
Ir. F.X. Arunanto, M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2017

Halaman ini sengaja dikosongkan



UNDERGRADUATE THESES - KI141502

**ALGORITHM DESIGN, ANALYSIS AND IMPLEMENTATION
FOR STRING COMPUTATION USING DYNAMIC
PROGRAMMING AND MEET IN THE MIDDLE TECHNIQUE IN
SPOJ CLASSIC PROBLEM 9967 PLAYING WITH WORDS**

DEWANGGA WINASFORCEPTA WINARDI
NRP 5113 100 098

Supervisor 1
Rully Soelaiman, S.Kom., M.Kom.

Supervisor 2
Ir. F.X. Arunanto, M.Sc.

INFORMATICS DEPARTMENT
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya, 2017

Halaman ini sengaja dikosongkan

**DESAIN, ANALISIS DAN IMPLEMENTASI ALGORITMA
KOMPUTASI STRING DENGAN METODE DYNAMIC
PROGRAMMING DAN MEET IN THE MIDDLE PADA
PERMASALAHAN KLASIK SPOJ 9967 PLAYING WITH
WORDS**

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Algoritma Pemrograman
Program Studi S-1 Departemen Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:

Dewangga Winasforcepta Winardi
NRP: 5113 100 098

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Rully Soelaiman, S.Kom., M.Kom.
NIP: 197002131994021001

.....
(Pembimbing 1)

Ir. F.X. Arunanto, M.Sc.
NIP: 195701011983031004

.....
(Pembimbing 2)

**SURABAYA
JULI 2017**

Halaman ini sengaja dikosongkan

DESAIN, ANALISIS DAN IMPLEMENTASI ALGORITMA KOMPUTASI STRING DENGAN METODE DYNAMIC PROGRAMMING DAN MEET IN THE MIDDLE PADA PERMASALAHAN KLASIK SPOJ 9967 PLAYING WITH WORDS

Nama : DEWANGGA WINASFORCEPTA WINARDI
NRP : 5113 100 098
Departemen : Teknik Informatika FTIF-ITS
Pembimbing I : Rully Soelaiman, S.Kom., M.Kom.
Pembimbing II : Ir. F.X. Arunanto, M.Sc.

Abstrak

Diberikan dua buah string orig1 dan orig2. Diberikan tiga tahapan proses enkripsi untuk menghasilkan string ad1 dan ad2. Tahap pertama adalah string orig1 diacak susunan karakter-karakternya. Tahap kedua adalah string orig2 diacak susunan karakter-karakternya. Tahap terakhir adalah salah satu karakter dari string orig1 atau orig2 diganti dengan karakter sebelum atau sesudahnya dalam alfabet. Jarak dua buah string didefinisikan sebagai jumlah dari selisih mutlak dari karakter-karakter pada posisi yang sama. Diberikan sebuah bilangan bulat X yang merupakan jarak dari string orig1 dan string ad1 dijumlahkan dengan jarak dari string orig2 dan string ad2. Tentukan jumlah kemungkinan kombinasi string orig1 dan orig2 jika diberikan string ad1, ad2 dan nilai X.

Meet in the middle adalah sebuah teknik pencarian dengan paradigma divide and conquer yang membagi permasalahan menjadi dua, lalu menyelesaikannya masing-masing, lalu menggabungkannya kembali.

Dynamic programming adalah sebuah paradigma untuk mendapatkan nilai optimal dari beberapa kemungkinan jawaban, dimana permasalahan tersebut memiliki submasalah tumpang

tindih dan struktur optimal.

Pada tugas akhir ini akan dirancang penyelesaian masalah yang disampaikan pada paragraf pertama dengan menggunakan teknik meet in the middle dan pendekatan dynamic programming.

*Solusi yang dikembangkan berjalan dengan kompleksitas waktu $\mathcal{O}(2^{|S|} * MAX_DIST^2 * T)$, dimana $|S|$ adalah panjang string yang diberikan dan MAX_DIST adalah jarak antar string maksimal.*

Kata Kunci: string, divide and conquer, meet in the middle, dynamic programming

ALGORITHM DESIGN, ANALYSIS AND IMPLEMENTATION FOR STRING COMPUTATION USING DYNAMIC PROGRAMMING AND MEET IN THE MIDDLE TECHNIQUE IN SPOJ CLASSIC PROBLEM 9967 PLAYING WITH WORDS

Name : DEWANGGA WINASFORCEPTA WINARDI
NRP : 5113 100 098
Major : Informatics Department Faculty of IT-ITS
Supervisor I : Rully Soelaiman, S.Kom., M.Kom.
Supervisor II : Ir. F.X. Arunanto, M.Sc.

Abstract

Given two strings orig1 and orig2. Given three steps to encrypt orig1 and orig2 to become ad1 and ad2. First step is shuffle the letters position of string orig1. The Second step is shuffle the letters position of string orig2. The last step is replace one letter from string orig1 or orig2 with its next or previous letter in the alphabet. Distance between two strings is defined as the sum of absolute difference between letter in same position. Given an integer X which is $distance(orig1, ad1) + distance(orig2, ad2)$. Find the number of possible string orig1 and orig2 from given string ad1, ad2 and integer X.

Meet in the middle is a searching technique with divide and conquer paradigm that divide problem into two or more subproblems, solve each, and combine all the subproblem's answers to get main answer.

Dynamic programming is a paradigm to get optimal solution from some possible answer, which each of subproblem has overlapping subproblems and optimal structure.

In this final project the writer will design and analyse an algorithm to solve the problem that stated in the first paragraph using meet in

the middle and dynamic programming.

*The solution will run in $\mathcal{O}(2^{|S|} * MAX_DIST^2 * T)$ time complexity where $|S|$ is the given string ad1 and ad2 and MAX_DIST is the maximum distance between given strings.*

Keywords: string, divide and conquer, meet in the middle, dynamic programming

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT. atas pimpinan, penyertaan, dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul :

DESAIN, ANALISIS DAN IMPLEMENTASI ALGORITMA KOMPUTASI STRING DENGAN METODE DYNAMIC PROGRAMMING DAN MEET IN THE MIDDLE PADA PERMASALAHAN KLASIK SPOJ 9967 PLAYING WITH WORDS.

Penelitian Tugas Akhir ini dilakukan untuk memenuhi salah satu syarat meraih gelar Sarjana di Jurusan Teknik Informatika Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember.

Dengan selesaiannya Tugas Akhir ini diharapkan apa yang telah dikerjakan penulis dapat memberikan manfaat bagi perkembangan ilmu pengetahuan terutama di bidang teknologi informasi serta bagi diri penulis sendiri selaku peneliti.

Penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan baik secara langsung maupun tidak langsung selama penulis mengerjakan Tugas Akhir maupun selama menempuh masa studi antara lain:

- Almh. Mama yang selalu memberi perhatian, kasih sayang serta dukungan atas apapun keputusan dan kegiatan yang dilakukan oleh penulis dan menjadi semangat utama bagi diri penulis baik selama penulis menempuh masa kuliah maupun penggerjaan Tugas Akhir ini.
- Bapak, Ibu dan keluarga penulis yang selalu memberikan perhatian, dorongan dan kasih sayang.

- Bapak Rully Soelaiman, S.Kom., M.Kom. selaku Dosen Pembimbing yang telah banyak meluangkan waktu untuk memberikan ilmu, nasihat, motivasi, pandangan dan bimbingan kepada penulis baik selama penulis menempuh masa kuliah maupun selama penggerjaan Tugas Akhir ini.
- Bapak Ir. F.X. Arunanto, M.Sc. selaku dosen pembimbing yang telah memberikan ilmu, dan masukan kepada penulis.
- Teman-teman, Kakak-kakak dan Adik-adik administrator Laboratorium Pemrograman yang selalu menjadi teman untuk berbagi ilmu.
- Seluruh tenaga pengajar dan karyawan Jurusan Teknik Informatika ITS yang telah memberikan ilmu dan waktunya demi berlangsungnya kegiatan belajar mengajar di Jurusan Teknik Informatika ITS.
- Seluruh teman penulis di Jurusan Teknik Informatika ITS yang telah memberikan dukungan dan semangat kepada penulis selama penulis menyelesaikan Tugas Akhir ini.

Penulis mohon maaf apabila masih ada kekurangan pada Tugas Akhir ini. Penulis juga mengharapkan kritik dan saran yang membangun untuk pembelajaran dan perbaikan di kemudian hari. Semoga melalui Tugas Akhir ini penulis dapat memberikan kontribusi dan manfaat yang sebaik-baiknya.

Surabaya, Juli 2017

Dewangga Winasforcepta Winardi

DAFTAR ISI

SAMPUL	i
LEMBAR PENGESAHAN	vii
ABSTRAK	ix
ABSTRACT	xi
KATA PENGANTAR	xiii
DAFTAR ISI	xv
DAFTAR TABEL	xix
DAFTAR GAMBAR	xxxiii
DAFTAR KODE SUMBER	xxxv
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Batasan Masalah	2
1.4 Tujuan	3
1.5 Manfaat	3
1.6 Metodologi	3
1.7 Sistematika Penulisan	4
BAB II DASAR TEORI	7
2.1 Deskripsi Permasalahan	7
2.2 Deskripsi Umum	8
2.2.1 <i>String</i>	8
2.2.2 Rekurens	10
2.2.3 <i>Divide and Conquer</i>	10
2.2.4 <i>Meet In The Middle</i>	10
2.2.5 <i>Dynamic Programming</i>	10
2.2.6 <i>State</i>	11
2.2.7 <i>Bitmask</i>	11
2.3 Analisa Submasalah Optimal	11

2.3.1	Membagi Permasalah Menjadi Submasalah yang <i>Independent</i>	11
2.3.2	Submasalah Optimal untuk Menghitung Jumlah Kombinasi <i>String Orig</i> dari <i>String Ad</i> Tanpa Operasi <i>Replace</i> dengan Jarak <i>D</i>	17
2.3.3	Submasalah Optimal untuk Menghitung Jumlah Kombinasi <i>String Orig</i> dari <i>String Ad</i> dengan Operasi <i>Replace</i> dengan Jarak <i>D</i>	19
2.4	Pemodelan Relasi Rekurens	23
2.4.1	Pemodelan Relasi Rekurens Submasalah Optimal untuk Menghitung Jumlah Kemungkinan <i>String</i> Awal Tanpa Operasi <i>Replace</i> dengan Jarak $X - dist$	28
2.4.2	Pemodelan Relasi Rekurens Submasalah Optimal untuk Menghitung Jumlah Kemungkinan <i>String</i> Awal dengan Sekali Operasi <i>Replace</i> dengan Jarak $X - dist$	35
BAB III DESAIN	49
3.1	Desain Umum Sistem	49
3.2	Desain Fungsi Preprocess	49
3.3	Desain Fungsi Init	50
3.4	Desain Fungsi Solve	53
3.4.1	Desain Fungsi F	54
3.4.2	Desain Fungsi G	57
BAB IV IMPLEMENTASI	69
4.1	Lingkungan Implementasi	69
4.2	Rancangan Data	69
4.2.1	Data Masukan	70
4.2.2	Data Keluaran	70
4.3	Implementasi Algoritma	70
4.3.1	Header-Header yang Diperlukan	70
4.3.2	Variabel Global	71
4.3.3	Implementasi Fungsi Main	72

4.3.4	Implementasi Fungsi Preprocess	72
4.3.5	Implementasi Fungsi ReadInput	73
4.3.6	Implementasi Fungsi Init	73
4.3.7	Implementasi Fungsi Solve	74
4.3.8	Implementasi Fungsi F	75
4.3.9	Implementasi Fungsi F1	76
4.3.10	Implementasi Fungsi G	77
4.3.11	Implementasi Fungsi G1	77
4.3.12	Implementasi Fungsi G2	78
4.3.13	Implementasi Fungsi G3	79
4.3.14	Implementasi Fungsi Duplicate Rule 1 . . .	79
4.3.15	Implementasi Fungsi Duplicate Rule 2 . . .	80
4.3.16	Implementasi Fungsi Duplicate Rule 3 . . .	81
BAB V	UJI COBA DAN EVALUASI	83
5.1	Lingkungan Uji Coba	83
5.2	Uji Coba Kebenaran	83
5.3	Analisa Kompleksitas Waktu	89
BAB VI	KESIMPULAN	91
6.1	Kesimpulan	91
6.2	Saran	91
DAFTAR PUSTAKA		93
Lampiran A	Tabel Himpunan <i>setBit</i> Setelah Fungsi Preprocess Dijalankan	95
Lampiran B	Hasil Uji Coba Kebenaran pada Situs SPOJ	131
Lampiran C	Tabel Simulasi Perhitungan Jumlah Kemungkinan <i>String orig1</i> dan <i>orig2</i> pada Kasus <i>String ad1=c, String ad2=n</i> dan <i>X=1</i>	133
Lampiran D	Tabel Simulasi Perhitungan Jumlah Kemungkinan <i>String orig1</i> dan <i>orig2</i> pada Kasus <i>String ad1=kbenh, String ad2=kbenh</i> dan <i>X=5</i>	137
BIODATA PENULIS		197

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Tabel 2.1	Kombinasi <i>string orig1</i> dan <i>orig2</i> dengan <i>ad1</i> = <i>c</i> , <i>string ad2</i> = <i>n</i> dan <i>X</i> = 1	8
Tabel 2.2	Kombinasi <i>string orig1</i> dan <i>orig2</i> dengan <i>ad1</i> = <i>bd</i> , <i>string ad2</i> = <i>gj</i> dan <i>X</i> = 5	9
Tabel 2.3	Kombinasi <i>string orig1</i> dengan nilai <i>string</i> <i>ad1</i> = <i>bd</i> tanpa operasi <i>replace</i>	15
Tabel 2.4	Kombinasi <i>string orig2</i> dengan nilai <i>string</i> <i>ad2</i> = <i>gj</i> dengan operasi <i>replace</i>	15
Tabel 2.5	Kombinasi <i>string orig1</i> dengan nilai <i>string</i> <i>ad1</i> = <i>bd</i> tanpa operasi <i>replace</i> dan <i>string</i> <i>orig2</i> dengan nilai <i>string ad2</i> = <i>gj</i> dengan operasi <i>replace</i> dengan <i>X</i> = 5	16
Tabel 2.6	Kombinasi <i>string orig1</i> dengan nilai <i>string</i> <i>ad1</i> = <i>bd</i> dengan operasi <i>replace</i>	16
Tabel 2.7	Kombinasi <i>string orig2</i> dengan nilai <i>string</i> <i>ad2</i> = <i>gj</i> tanpa operasi <i>replace</i>	16
Tabel 2.8	Kombinasi <i>string orig1</i> dengan nilai <i>string</i> <i>ad1</i> = <i>bd</i> dengan operasi <i>replace</i> dan <i>string</i> <i>orig2</i> dengan nilai <i>string ad2</i> = <i>gj</i> tanpa operasi <i>replace</i> dengan <i>X</i> = 5	17
Tabel 2.9	Daftar notasi persamaan 2.4.1	28
Tabel 2.10	Daftar notasi persamaan 2.4.2, 2.4.3 dan 2.4.4	29
Tabel 2.11	Daftar notasi persamaan 2.4.5, 2.4.6, 2.4.7, 2.4.8, 2.4.9 dan 2.4.10 (1)	36
Tabel 2.12	Daftar notasi persamaan 2.4.5, 2.4.6, 2.4.7, 2.4.8, 2.4.9 dan 2.4.10 (2)	37
Tabel 3.1	Hasil $charFirstPos_{(S,C)}$ dan $charLastPos_{(S,C)}$ dengan <i>string</i> <i>S</i> = "inicontoh" setelah fungsi init dijalankan	52

Tabel 3.2	Simulasi fungsi F dengan $S = "kbenh"$, $X = 5$ dan $dist = 5$	55
Tabel 3.3	Simulasi fungsi G dengan $S = "kbenh"$, $X = 5$ dan $dist = 0$ (1)	59
Tabel 3.4	Simulasi fungsi G dengan $S = "kbenh"$, $X = 5$ dan $dist = 0$ (2)	60
Tabel 3.5	Simulasi fungsi G dengan $S = "kbenh"$, $X = 5$ dan $dist = 0$ (3)	61
Tabel 3.6	Simulasi fungsi G dengan $S = "kbenh"$, $X = 5$ dan $dist = 0$ (4)	62
Tabel 3.7	Simulasi fungsi G dengan $S = "kbenh"$, $X = 5$ dan $dist = 0$ (5)	63
Tabel 3.8	Simulasi fungsi G dengan $S = "kbenh"$, $X = 5$ dan $dist = 0$ (6)	64
Tabel 5.1	Kecepatan maksimal, minimal dan rata-rata dari hasil uji coba pengumpulan 30 kali pada situs pengujian daring SPOJ	84
Tabel A.1	Tabel himpunan setBit setelah fungsi preprocess dijalankan (1)	95
Tabel A.2	Tabel himpunan setBit setelah fungsi preprocess dijalankan (2)	96
Tabel A.3	Tabel himpunan setBit setelah fungsi preprocess dijalankan (3)	97
Tabel A.4	Tabel himpunan setBit setelah fungsi preprocess dijalankan (4)	98
Tabel A.5	Tabel himpunan setBit setelah fungsi preprocess dijalankan (5)	99
Tabel A.6	Tabel himpunan setBit setelah fungsi preprocess dijalankan (6)	100
Tabel A.7	Tabel himpunan setBit setelah fungsi preprocess dijalankan (7)	101
Tabel A.8	Tabel himpunan setBit setelah fungsi preprocess dijalankan (8)	102

Tabel A.9 Tabel himpunan setBit setelah fungsi preprocess dijalankan (9)	103
Tabel A.10 Tabel himpunan setBit setelah fungsi preprocess dijalankan (10)	104
Tabel A.11 Tabel himpunan setBit setelah fungsi preprocess dijalankan (11)	105
Tabel A.12 Tabel himpunan setBit setelah fungsi preprocess dijalankan (12)	106
Tabel A.13 Tabel himpunan setBit setelah fungsi preprocess dijalankan (13)	107
Tabel A.14 Tabel himpunan setBit setelah fungsi preprocess dijalankan (14)	108
Tabel A.15 Tabel himpunan setBit setelah fungsi preprocess dijalankan (15)	109
Tabel A.16 Tabel himpunan setBit setelah fungsi preprocess dijalankan (16)	110
Tabel A.17 Tabel himpunan setBit setelah fungsi preprocess dijalankan (17)	111
Tabel A.18 Tabel himpunan setBit setelah fungsi preprocess dijalankan (18)	112
Tabel A.19 Tabel himpunan setBit setelah fungsi preprocess dijalankan (19)	113
Tabel A.20 Tabel himpunan setBit setelah fungsi preprocess dijalankan (20)	114
Tabel A.21 Tabel himpunan setBit setelah fungsi preprocess dijalankan (21)	115
Tabel A.22 Tabel himpunan setBit setelah fungsi preprocess dijalankan (22)	116
Tabel A.23 Tabel himpunan setBit setelah fungsi preprocess dijalankan (23)	117
Tabel A.24 Tabel himpunan setBit setelah fungsi preprocess dijalankan (24)	118

Tabel A.25 Tabel himpunan setBit setelah fungsi preprocess dijalankan (25)	119
Tabel A.26 Tabel himpunan setBit setelah fungsi preprocess dijalankan (26)	120
Tabel A.27 Tabel himpunan setBit setelah fungsi preprocess dijalankan (27)	121
Tabel A.28 Tabel himpunan setBit setelah fungsi preprocess dijalankan (28)	122
Tabel A.29 Tabel himpunan setBit setelah fungsi preprocess dijalankan (29)	123
Tabel A.30 Tabel himpunan setBit setelah fungsi preprocess dijalankan (30)	124
Tabel A.31 Tabel himpunan setBit setelah fungsi preprocess dijalankan (31)	125
Tabel A.32 Tabel himpunan setBit setelah fungsi preprocess dijalankan (32)	126
Tabel A.33 Tabel himpunan setBit setelah fungsi preprocess dijalankan (33)	127
Tabel A.34 Tabel himpunan setBit setelah fungsi preprocess dijalankan (34)	128
Tabel A.35 Tabel himpunan setBit setelah fungsi preprocess dijalankan (35)	129
Tabel C.1 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> tanpa operasi <i>replace</i> dengan <i>dist = 0</i> pada kasus <i>string ad1 = c, string ad2 = n</i> dan <i>X = 1</i>	133
Tabel C.2 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> dengan operasi <i>replace</i> dengan <i>dist = 0</i> pada kasus <i>string ad1 = c, string ad2 = n</i> dan <i>X = 1</i>	133

Tabel C.3	Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 0 pada kasus <i>string ad1</i> = <i>c</i> , <i>string ad2</i> = <i>n</i> dan <i>X</i> = 1	134
Tabel C.4	Simulasi perhitungan jumlah kombinasi <i>string orig2</i> tanpa operasi <i>replace</i> dengan <i>dist</i> = 0 pada kasus <i>string ad1</i> = <i>c</i> , <i>string ad2</i> = <i>n</i> dan <i>X</i> = 1	134
Tabel C.5	Simulasi perhitungan jumlah kombinasi <i>string orig1</i> tanpa operasi <i>replace</i> dengan <i>dist</i> = 1 pada kasus <i>string ad1</i> = <i>c</i> , <i>string ad2</i> = <i>n</i> dan <i>X</i> = 1	134
Tabel C.6	Simulasi perhitungan jumlah kombinasi <i>string orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 1 pada kasus <i>string ad1</i> = <i>c</i> , <i>string ad2</i> = <i>n</i> dan <i>X</i> = 1	135
Tabel C.7	Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 1 pada kasus <i>string ad1</i> = <i>c</i> , <i>string ad2</i> = <i>n</i> dan <i>X</i> = 1	135
Tabel C.8	Simulasi perhitungan jumlah kombinasi <i>string orig2</i> tanpa operasi <i>replace</i> dengan <i>dist</i> = 1 pada kasus <i>string ad1</i> = <i>c</i> , <i>string ad2</i> = <i>n</i> dan <i>X</i> = 1	135
Tabel D.1	Simulasi perhitungan jumlah kombinasi <i>string orig1</i> tanpa operasi <i>replace</i> dengan <i>dist</i> = 0 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5	137
Tabel D.2	Simulasi perhitungan jumlah kombinasi <i>string orig1</i> tanpa operasi <i>replace</i> dengan <i>dist</i> = 0 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5	138

Tabel D.3 Simulasi perhitungan jumlah kombinasi string <i>orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 0 pada kasus string <i>ad1</i> = <i>kbenh</i> , string <i>ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (1)	139
Tabel D.4 Simulasi perhitungan jumlah kombinasi string <i>orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 0 pada kasus string <i>ad1</i> = <i>kbenh</i> , string <i>ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (2)	140
Tabel D.5 Simulasi perhitungan jumlah kombinasi string <i>orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 0 pada kasus string <i>ad1</i> = <i>kbenh</i> , string <i>ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (3)	141
Tabel D.6 Simulasi perhitungan jumlah kombinasi string <i>orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 0 pada kasus string <i>ad1</i> = <i>kbenh</i> , string <i>ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (4)	142
Tabel D.7 Simulasi perhitungan jumlah kombinasi string <i>orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 0 pada kasus string <i>ad1</i> = <i>kbenh</i> , string <i>ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (5)	143
Tabel D.8 Simulasi perhitungan jumlah kombinasi string <i>orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 0 pada kasus string <i>ad1</i> = <i>kbenh</i> , string <i>ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (6)	144
Tabel D.9 Simulasi perhitungan jumlah kombinasi string <i>orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 0 pada kasus string <i>ad1</i> = <i>kbenh</i> , string <i>ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (1)	145
Tabel D.10 Simulasi perhitungan jumlah kombinasi string <i>orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 0 pada kasus string <i>ad1</i> = <i>kbenh</i> , string <i>ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (2)	146

Tabel D.11 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 0 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (3)	147
Tabel D.12 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> tanpa operasi <i>replace</i> dengan <i>dist</i> = 0 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (1)	148
Tabel D.13 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> tanpa operasi <i>replace</i> dengan <i>dist</i> = 0 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (2)	149
Tabel D.14 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> tanpa operasi <i>replace</i> dengan <i>dist</i> = 1 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5	150
Tabel D.15 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 1 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (1)	151
Tabel D.16 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 1 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (2)	152
Tabel D.17 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 1 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (3)	153
Tabel D.18 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 1 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (4)	154

Tabel D.19 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 1 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (5)	155
Tabel D.20 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 1 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (1)	156
Tabel D.21 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 1 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (2)	157
Tabel D.22 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 1 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (3)	158
Tabel D.23 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> tanpa operasi <i>replace</i> dengan <i>dist</i> = 1 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5	158
Tabel D.24 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> tanpa operasi <i>replace</i> dengan <i>dist</i> = 2 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5	159
Tabel D.25 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 2 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (1)	160
Tabel D.26 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 2 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (2)	161

Tabel D.27 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 2 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (3)	162
Tabel D.28 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 2 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (4)	163
Tabel D.29 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 2 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (5)	164
Tabel D.30 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 2 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (1)	165
Tabel D.31 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 2 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (2)	166
Tabel D.32 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 2 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (3)	167
Tabel D.33 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> tanpa operasi <i>replace</i> dengan <i>dist</i> = 2 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5	168
Tabel D.34 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> tanpa operasi <i>replace</i> dengan <i>dist</i> = 3 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5	169

Tabel D.35 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 3 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (1)	170
Tabel D.36 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 3 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (2)	171
Tabel D.37 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 3 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (3)	172
Tabel D.38 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 3 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (1)	173
Tabel D.39 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 3 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (2)	174
Tabel D.40 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 3 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (3)	175
Tabel D.41 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 3 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (4)	176
Tabel D.42 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 3 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (5)	177

Tabel D.43 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> tanpa operasi <i>replace</i> dengan <i>dist</i> = 3 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5	177
Tabel D.44 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> tanpa operasi <i>replace</i> dengan <i>dist</i> = 4 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5	178
Tabel D.45 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 4 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (1)	179
Tabel D.46 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 4 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (2)	180
Tabel D.47 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 4 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (3)	181
Tabel D.48 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 4 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (1)	182
Tabel D.49 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 4 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (2)	183
Tabel D.50 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 4 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (3)	184

Tabel D.51 Simulasi perhitungan jumlah kombinasi string <i>orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 4 pada kasus string <i>ad1</i> = <i>kbenh</i> , string <i>ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (4)	185
Tabel D.52 Simulasi perhitungan jumlah kombinasi string <i>orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 4 pada kasus string <i>ad1</i> = <i>kbenh</i> , string <i>ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (5)	186
Tabel D.53 Simulasi perhitungan jumlah kombinasi string <i>orig2</i> tanpa operasi <i>replace</i> dengan <i>dist</i> = 4 pada kasus string <i>ad1</i> = <i>kbenh</i> , string <i>ad2</i> = <i>kbenh</i> dan <i>X</i> = 5	186
Tabel D.54 Simulasi perhitungan jumlah kombinasi string <i>orig1</i> tanpa operasi <i>replace</i> dengan <i>dist</i> = 5 pada kasus string <i>ad1</i> = <i>kbenh</i> , string <i>ad2</i> = <i>kbenh</i> dan <i>X</i> = 5	187
Tabel D.55 Simulasi perhitungan jumlah kombinasi string <i>orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 5 pada kasus string <i>ad1</i> = <i>kbenh</i> , string <i>ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (1)	188
Tabel D.56 Simulasi perhitungan jumlah kombinasi string <i>orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 5 pada kasus string <i>ad1</i> = <i>kbenh</i> , string <i>ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (2)	189
Tabel D.57 Simulasi perhitungan jumlah kombinasi string <i>orig2</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 5 pada kasus string <i>ad1</i> = <i>kbenh</i> , string <i>ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (3)	190
Tabel D.58 Simulasi perhitungan jumlah kombinasi string <i>orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 5 pada kasus string <i>ad1</i> = <i>kbenh</i> , string <i>ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (1)	191

Tabel D.59 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 5 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (2)	192
Tabel D.60 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 5 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (3)	193
Tabel D.61 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 5 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (4)	194
Tabel D.62 Simulasi perhitungan jumlah kombinasi <i>string orig1</i> dengan operasi <i>replace</i> dengan <i>dist</i> = 5 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5 (5)	195
Tabel D.63 Simulasi perhitungan jumlah kombinasi <i>string orig2</i> tanpa operasi <i>replace</i> dengan <i>dist</i> = 5 pada kasus <i>string ad1</i> = <i>kbenh</i> , <i>string ad2</i> = <i>kbenh</i> dan <i>X</i> = 5	196

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Gambar 2.1	Ilustrasi umum penyelesaian permasalahan dengan metode <i>meet in the middle</i> tanpa operasi <i>replace</i>	12
Gambar 2.2	Ilustrasi umum penyelesaian permasalahan dengan metode <i>meet in the middle</i> dengan operasi <i>replace</i>	14
Gambar 2.3	Ilustrasi umum penyelesaian permasalahan dengan metode <i>meet in the middle</i> dengan operasi <i>replace</i> tanpa mempedulikan kombinasi <i>string</i> yang dihasilkan	18
Gambar 2.4	Ilustrasi perhitungan jumlah kombinasi <i>string orig</i> dari <i>string ad</i> tanpa operasi <i>replace</i> dengan nilai <i>string ad = bcd</i> dan $D = 4$	20
Gambar 2.5	Contoh kasus tumpang tindih pada perhitungan kombinasi <i>string orig</i> dari <i>string ad</i> tanpa operasi <i>replace</i> dengan nilai <i>string ad = bcd</i> dan $D = 4$	21
Gambar 2.6	Ilustrasi perhitungan jumlah kombinasi <i>string orig</i> dari <i>string ad</i> tanpa operasi <i>replace</i> dengan nilai <i>string ad = bcd</i> dan $D = 4$ tanpa menghitung kasus yang tumpang tindih	22
Gambar 2.7	Ilustrasi perhitungan jumlah kombinasi <i>string orig</i> dari <i>string ad</i> dengan operasi <i>replace</i> dengan nilai <i>string ad = be</i> dan $D = 1$	24

Gambar 2.8 Contoh kasus tumpang tindih pada perhitungan kombinasi <i>string orig</i> dari <i>string ad</i> dengan operasi <i>replace</i> dengan nilai <i>string ad = be</i> dan $D = 1$	25
Gambar 2.9 Submasalah perhitungan jumlah kombinasi <i>string orig</i> terhadap <i>string ad</i> tanpa operasi <i>replace</i> pada submasalah perhitungan jumlah kombinasi <i>string orig</i> terhadap <i>string ad</i> dengan operasi <i>replace</i>	26
Gambar 3.1 Pseudocode Fungsi Main	50
Gambar 3.2 Pseudocode Fungsi Preprocess	50
Gambar 3.3 Pseudocode Fungsi Init	53
Gambar 3.4 Pseudocode Fungsi Solve	54
Gambar 3.5 Pseudocode Fungsi F	56
Gambar 3.6 Pseudocode Fungsi F1	57
Gambar 3.7 Pseudocode Fungsi duplicate_rule1	57
Gambar 3.8 Pseudocode Fungsi G	65
Gambar 3.9 Pseudocode Fungsi G1	65
Gambar 3.10 Pseudocode Fungsi G2	66
Gambar 3.11 Pseudocode Fungsi G3	66
Gambar 3.12 Pseudocode Fungsi duplicate_rule2	66
Gambar 3.13 Pseudocode Fungsi duplicate_rule3	67
Gambar B.1 Hasil uji coba pada situs penilaian SPOJ	131
Gambar B.2 Hasil pengujian sebanyak 30 kali pada situs penilaian daring SPOJ (1)	131
Gambar B.3 Hasil pengujian sebanyak 30 kali pada situs penilaian daring SPOJ (2)	132
Gambar B.4 Grafik hasil uji coba pada situs SPOJ sebanyak 30 kali	132

DAFTAR KODE SUMBER

Kode Sumber 4.3.1	Header yang diperlukan	71
Kode Sumber 4.3.2	Variabel global	72
Kode Sumber 4.3.3	Fungsi main	72
Kode Sumber 4.3.4	Fungsi preprocess	73
Kode Sumber 4.3.5	Fungsi readInput	73
Kode Sumber 4.3.6	Fungsi init	74
Kode Sumber 4.3.7	Fungsi solve	75
Kode Sumber 4.3.8	Fungsi F (1)	75
Kode Sumber 4.3.9	Fungsi F (2)	76
Kode Sumber 4.3.10	Fungsi F1	76
Kode Sumber 4.3.11	Fungsi G	77
Kode Sumber 4.3.12	Fungsi G1	78
Kode Sumber 4.3.13	Fungsi G2 (1)	78
Kode Sumber 4.3.14	Fungsi G2 (2)	79
Kode Sumber 4.3.15	Fungsi G3	79
Kode Sumber 4.3.16	Fungsi duplicate_rule1	80
Kode Sumber 4.3.17	Fungsi duplicate_rule2	80
Kode Sumber 4.3.18	Fungsi duplicate_rule3	81

Halaman ini sengaja dikosongkan

BAB I

PENDAHULUAN

Pada bab ini akan dijelaskan latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi dan sistematika penulisan Tugas Akhir.

1.1 Latar Belakang

Tugas Akhir ini mengacu pada permasalahan klasik SPOJ 9967 *Playing With Words*. Diberikan dua buah *string* *orig1* dan *orig2*. Diberikan tiga tahapan proses enkripsi untuk menghasilkan *string* *ad1* dan *ad2* sebagai berikut:

1. *String* *orig1* diacak urutan karakter-karakternya.
2. *String* *orig2* diacak urutan karakter-karakternya.
3. Salah satu karakter dari *string* *orig1* atau *orig2* diganti dengan karakter sebelum atau sesudahnya dalam alfabet.

Jarak dua buah *string* didefinisikan sebagai jumlah dari selisih mutlak dari karakter-karakter pada posisi yang sama. Diberikan sebuah bilangan bulat *X* yang merupakan jarak dari *string* *orig1* dan *string* *ad1* dijumlahkan dengan jarak dari *string* *orig2* dan *string* *ad2*. Berapakah jumlah kemungkinan kombinasi *string* *orig1* dan *orig2* jika diberikan *string* *ad1*, *ad2* dan nilai *X*.

Untuk menyelesaikan permasalahan di atas, penulis akan menggunakan pendekatan solusi dengan teknik *dynamic programming* dan *meet in the middle*. Selain dapat menjawab pertanyaan dengan benar, waktu juga menjadi salah satu faktor penting untuk memberikan gambaran tentang performa dari algoritma yang dirancang.

Hasil dari Tugas Akhir ini diharapkan dapat memberikan gambaran mengenai performa algoritma dengan teknik *dynamic programming* dan *meet in the middle*.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Perancangan algoritma yang sesuai untuk menyelesaikan permasalahan klasik SPOJ 9967 *Playing With Words* yang didasari oleh teknik *dynamic programming* dan *meet in the middle*.
2. Implementasi algoritma untuk menyelesaikan permasalahan klasik SPOJ 9967 *Playing With Words*.
3. Analisis performa algoritma yang telah dirancang untuk menyelesaikan permasalahan klasik SPOJ 9967 *Playing With Words*.

1.3 Batasan Masalah

Permasalahan yang dibahas pada Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Implementasi algoritma menggunakan bahasa pemrograman C++.
2. Uji coba kebenaran dilakukan dengan uji *submission* ke situs penilaian daring SPOJ.
3. Panjang *string* masukan *ad1* dan *ad2* maksimal bernilai 10.
4. Karakter pada *string* masukan *ad1* dan *ad2* berada dalam rentang '*b*' ≤ *ad1_i*, *ad2_i* ≤ '*y*'.
5. Nilai masukan *X* tidak melebihi 100000.
6. Batas waktu eksekusi program adalah 6,459 detik.

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah sebagai berikut:

1. Menyelesaikan permasalahan klasik SPOJ 9967 *Playing With Words* dengan algoritma yang telah dirancang dan diimplementasikan menggunakan teknik *dynamic programming* dan *meet in the middle*.
2. Menguji kebenaran algoritma dengan melakukan uji kebenaran terhadap algoritma yang telah dirancang dan diimplementasikan.
3. Mengetahui performa algoritma yang dibangun dengan menganalisa hasil uji coba.

1.5 Manfaat

Manfaat dari Tugas Akhir ini adalah sebagai berikut:

1. Mengetahui pemanfaatan metode *dynamic programming* dan *meet in the middle* dalam menyelesaikan suatu permasalahan.
2. Melatih kemampuan analisis karakteristik permasalahan yang dapat diselesaikan dengan metode *dynamic programming* dan *meet in the middle*.

1.6 Metodologi

Metodologi yang digunakan dalam penggerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir
Pada tahap ini dilakukan penyusunan proposal Tugas Akhir yang berisi permasalahan dan gagasan solusi yang akan diteliti pada permasalahan klasik SPOJ 9967 *Playing With Words*.
2. Studi literatur
Pada tahap ini dilakukan pencarian informasi dan studi

literatur mengenai pengetahuan atau metode yang dapat digunakan dalam penyelesaian masalah. Informasi didapatkan dari materi-materi yang berhubungan dengan algoritma yang digunakan untuk penyelesaian permasalahan ini, materi-materi tersebut didapatkan dari buku, jurnal, maupun internet.

3. Desain

Pada tahap ini dilakukan desain rancangan algoritma yang digunakan dalam solusi untuk pemecahan permasalahan klasik SPOJ 9967 *Playing With Words*.

4. Implementasi perangkat lunak

Pada tahap ini dilakukan implementasi atau realiasi dari rancangan desain algoritma yang telah dibangun pada tahap desain ke dalam bentuk program.

5. Uji coba dan evaluasi

Pada tahap ini dilakukan uji coba kebenaran implementasi. Pengujian kebenaran dilakukan pada sistem penilaian daring SPOJ sesuai dengan masalah yang dikerjakan untuk diuji apakah luaran dari program telah sesuai.

6. Penyusunan buku Tugas Akhir

Pada tahap ini dilakukan penyusunan buku Tugas Akhir yang berisi dokumentasi hasil penggerjaan Tugas Akhir.

1.7 Sistematika Penulisan

Berikut adalah sistematika penulisan buku Tugas Akhir ini:

1. BABI: PENDAHULUAN

Bab ini berisi latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi dan sistematika penulisan Tugas Akhir.

2. BAB II: DASAR TEORI

Bab ini berisi dasar teori mengenai permasalahan dan algoritma penyelesaian yang digunakan dalam Tugas Akhir

3. BAB III: DESAIN

Bab ini berisi desain algoritma dan struktur data yang digunakan dalam penyelesaian permasalahan.

4. BAB IV: IMPLEMENTASI

Bab ini berisi implementasi berdasarkan desain algortima yang telah dilakukan pada tahap desain.

5. BAB V: UJI COBA DAN EVALUASI

Bab ini berisi uji coba dan evaluasi dari hasil implementasi yang telah dilakukan pada tahap implementasi.

6. BAB VI: KESIMPULAN

Bab ini berisi kesimpulan yang didapat dari hasil uji coba yang telah dilakukan.

Halaman ini sengaja dikosongkan

BAB II

DASAR TEORI

Pada bab ini akan dijelaskan mengenai dasar teori yang menjadi dasar pengerjaan Tugas Akhir ini.

2.1 Deskripsi Permasalahan

Amr M. sangat curiga dengan sebuah iklan yang terdiri dari 2 buah *string* $ad1$ dan $ad2$. Ia menduga kedua *string* tersebut menyimpan pesan tersembunyi. Setelah menghubungi sumber yang terpercaya, ia menemukan proses untuk menyembunyikan pesan tersebut. Pesan asli yang dibawa selalu berupa dua buah *string* $orig1$ dan $orig2$. Berikut adalah langkah-langkah transformasi pesan asli menjadi pesan pada iklan:

1. Karakter-karakter pada *string* $orig1$ diacak urutannya.
2. Karakter-karakter pada *string* $orig2$ diacak urutannya.
3. Salah satu karakter dari *string* $orig1$ atau $orig2$ diganti dengan karakter sebelum atau sesudahnya dalam alfabet.

Langkah-langkah di atas akan menghasilkan *string* $ad1$ dan $ad2$ dari $orig1$ dan $orig2$ secara berurutan. Contohnya untuk *string* $orig1 = bcd$ dan $orig2 = wcy$ dapat menghasilkan *string* $ad1 = dcw$ dan $ad2 = cwy$ di mana cwy berasal dari wcy yang diacak menjadi cwy dan karakter w digantikan dengan karakter x .

Setelah melakukan riset, Amr menemukan sebuah jarak X , di mana X adalah $jarak(orig1 + ad1)$ ditambah dengan $jarak(orig2 + ad2)$. Jarak antara dua *string* didefinisikan sebagai jumlah dari selisih absolut dari karakter-karakter pada posisi yang sama. Contohnya $jarak(ab, cd) = |'a' - 'c'| + |'b' - 'd'| = 4$. Diberikan

Tabel 2.1 Kombinasi *string orig1* dan *orig2* dengan $ad1 = c$, *string ad2 = n* dan $X = 1$

orig1	orig2	X	Validitas
b	n	1	Valid
d	n	1	Valid
c	m	1	Valid
c	o	1	Valid

string ad1, *ad2* dan sebuah bilangan bulat X . Hitung jumlah kemungkinan *string orig1* dan *orig2* yang mungkin.

Sebagai contoh, Tabel 2.1 adalah kombinasi dari *string orig1* dan *orig2* dengan $string ad1 = c$, *string ad2 = n* dan $X = 1$. Jawaban dari kasus tersebut adalah 4 karena terdapat 4 kombinasi dengan $X = 1$. Contoh berikutnya adalah kasus ketika $ad1 = bd$, *string ad2 = gj* dan $X = 5$. Berdasarkan daftar kombinasi *string orig1* dan *orig2* pada Tabel 2.2, terdapat 8 kombinasi *string orig1* dan *string orig2* yang memenuhi kriteria $X = 5$ sehingga jawaban akhir dari kasus tersebut adalah 8.

2.2 Deskripsi Umum

Pada subbab ini akan dijelaskan mengenai deskripsi-deskripsi umum yang terdapat pada Tugas Akhir ini.

2.2.1 *String*

Pada dunia ilmu komputer, *string* didefinisikan sebagai sebuah rangkaian karakter. *String* pada umumnya dipahami sebagai sebuah struktur data dan diimplementasi menggunakan struktur data *array[1]*.

Tabel 2.2 Kombinasi string $orig1$ dan $orig2$ dengan $ad1 = bd$, string $ad2 = gj$ dan $X = 5$

orig1	orig2	X	Validitas
ad	gj	1	Tidak valid
cd	gj	1	Tidak valid
bc	gj	1	Tidak valid
be	gj	1	Tidak valid
bd	fj	1	Tidak valid
bd	hj	1	Tidak valid
bd	gi	1	Tidak valid
bd	gk	1	Tidak valid
ad	jg	7	Tidak valid
cd	jg	7	Tidak valid
bc	jg	7	Tidak valid
be	jg	7	Tidak valid
bd	ig	5	Valid
bd	kg	7	Tidak valid
bd	jf	7	Tidak valid
bd	jh	5	Valid
cb	gj	3	Tidak valid
eb	gj	5	Valid
da	gj	5	Valid
dc	gj	3	Tidak valid
db	fj	5	Valid
db	hj	5	Valid
db	gi	5	Valid
db	gk	5	Valid
cb	jg	9	Tidak valid
eb	jg	11	Tidak valid
da	jg	11	Tidak valid
dc	jg	9	Tidak valid
db	ig	9	Tidak valid
db	kg	11	Tidak valid
db	jf	11	Tidak valid
db	jh	9	Tidak valid

2.2.2 Rekurens

Ketika sebuah algoritma mengandung sebuah persamaan rekursif yang memanggil dirinya sendiri, waktu prosesnya dapat dikatakan sebagai rekurens. Rekurens adalah sebuah persamaan atau pertidaksamaan yang mendeskripsikan sebuah fungsi dalam hal nilai pada masukan yang lebih kecil[2].

2.2.3 *Divide and Conquer*

Dalam ilmu komputer, *divide and conquer* (D&C) adalah paradigma perancangan algoritma yang bekerja dengan memecah permasalahan menjadi dua atau lebih submasalah dengan karakteristik yang sama atau berkaitan hingga cukup sederhana untuk diselesaikan secara langsung. Solusi dari masing-masing submasalah akan dikombinasikan untuk mendapatkan solusi dari permasalahan utama. Pada umumnya *divide and conquer* (D&C) merajuk pada aplikasi algoritma yang mereduksi setiap permasalahan menjadi hanya satu submasalah[3].

2.2.4 *Meet In The Middle*

Dalam dunia pemrograman komputer, *meet in the middle* adalah sebuah teknik pencarian dua arah dengan membagi dua permasalahan, lalu menyelesaiakannya secara terpisah, lalu menggabungkan keduanya untuk mendapatkan hasil yang diinginkan[3].

2.2.5 *Dynamic Programming*

Dalam dunia ilmu komputer, *dynamic programming* adalah sebuah metode penyelesaian masalah yang memecah sebuah permasalahan yang rumit menjadi submasalah-submasalah yang lebih sederhana. *dynamic programming* bersifat efektif ketika submasalah dari permasalahan yang diberikan mungkin berasal dari lebih dari

satu pilihan. Teknik kunci dari *dynamic programming* adalah menyimpan solusi untuk setiap submasalah untuk digunakan jika submasalah tersebut muncul kembali[2].

2.2.6 State

State atau *state variable* adalah himpunan variabel parameter dari sebuah submasalah dari permasalahan yang diberikan[4].

2.2.7 Bitmask

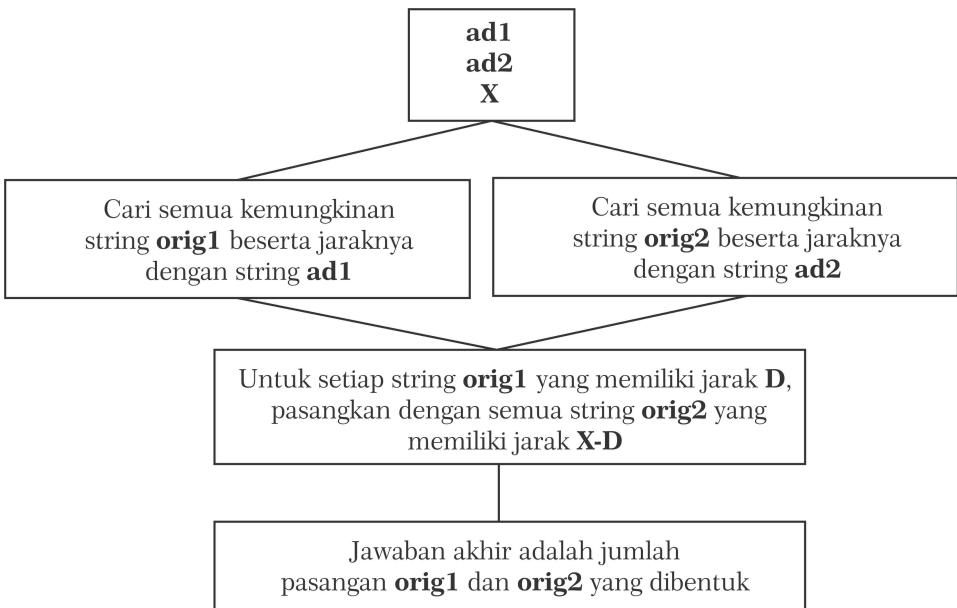
Bitmask adalah sebuah bilangan bulat yang disimpan dan direpresentasikan sebagai himpunan dari nilai *boolean*. Salah satu contoh pemanfaatan teknik *bitmasking* adalah penggunaan *bitmask* sebagai salah satu index pada tabel memo pada teknik *dynamic programming*[3].

2.3 Analisa Submasalah Optimal

Pada subbab ini akan dijelaskan mengenai submasalah-submasalah yang jawabannya dapat membangun jawaban akhir. Tujuan utama dari permasalahan yang diberikan adalah untuk mencari jumlah kemungkinan *string* *orig1* dan *orig2* dari *string* *ad1* dan *ad2* yang memiliki jarak $dist(orig1, ad1) + dist(orig2, ad2) = X$ yang berikutnya disebut dengan jawaban akhir.

2.3.1 Membagi Permasalah Menjadi Submasalah yang Independent

Pada permasalahan klasik SPOJ 9967 *Playing With Words*, jawaban akhir merupakan banyak kombinasi *string* *orig1* dan *string* *orig2* yang mungkin. Dapat dilihat bahwa perhitungan jumlah kombinasi *string* *orig1* dari *string* *ad1* dan perhitungan jumlah kombinasi *string* *orig2* dari *string* *ad2* tidak memiliki keterkaitan satu sama lain. Artinya adalah dapat dihitung jumlah kombinasi *string* *orig1*



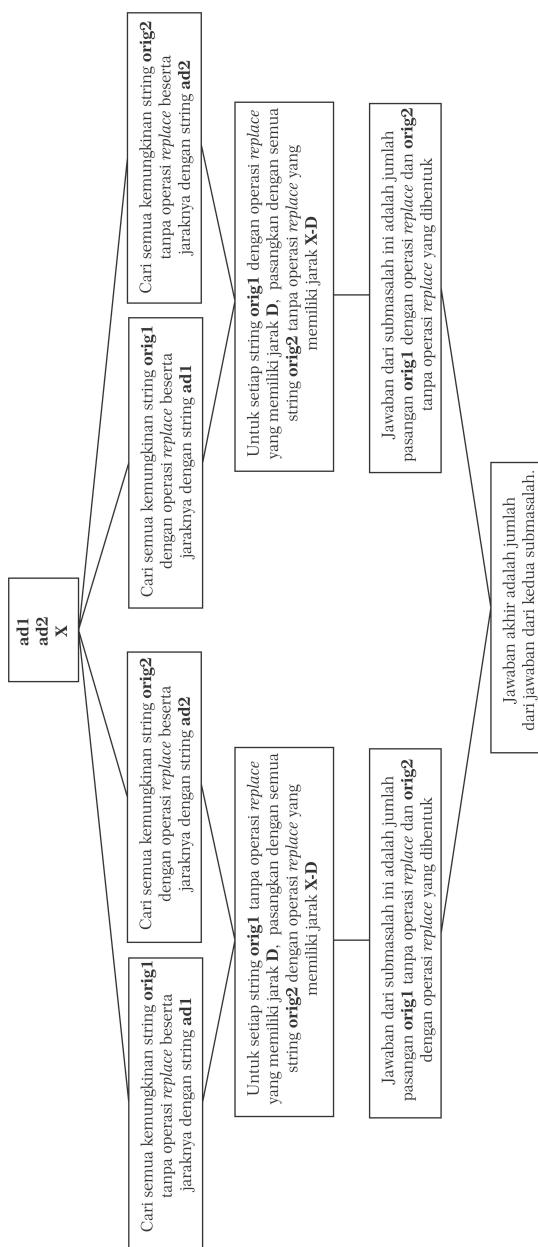
Gambar 2.1 Ilustrasi umum penyelesaian permasalahan dengan metode *meet in the middle* tanpa operasi *replace*

dari *string ad1* tanpa mempedulikan kondisi *string orig2*. Begitu juga sebaliknya pada perhitungan jumlah kombinasi *string orig2*. Dengan kata lain perhitungan jumlah kombinasi *string orig1* dan *orig2* dapat dilakukan secara terpisah. Teknik tersebut disebut dengan *meet in the middle*.

Apabila operasi *replace* pada permasalahan klasik SPOJ 9967 *Playing With Words* diabaikan, maka jawaban akhir dari dapat dihitung dengan alur secara umum pada Gambar 2.1. Karena operasi *replace* diabaikan, maka jawaban akhir dapat dibentuk dengan mencari seluruh kemungkinan pasangan *string orig1* yang memiliki jarak *D* terhadap *string ad1* dengan *string orig2* yang memiliki jarak *X - D* terhadap *string ad2*.

Ketika terdapat operasi *replace*, di mana operasi *replace* adalah operasi di mana salah satu karakter pada *string orig1* atau *orig2* diganti dengan karakter sebelumnya atau sesudahnya secara alfabetis, maka perlu dilakukan penyesuaian pada proses perhitungan jawaban. Selain perhitungan kombinasi *string orig1* dan *string orig2*, perlu juga dilakukan perhitungan untuk kombinasi *string orig1* dan *string orig2* setelah dilakukan satu kali operasi *replace*. Sehingga jawaban akhir dari permasalahan klasik SPOJ 9967 *Playing With Words* adalah total dari jumlah kemungkinan pasangan *string orig1* tanpa operasi *replace* yang memiliki jarak D terhadap *string ad1* dengan *string orig2* dengan operasi *replace* yang memiliki jarak $X - D$ terhadap *string ad2* dan jumlah kemungkinan pasangan *string orig1* dengan operasi *replace* yang memiliki jarak D terhadap *string ad1* dengan *string orig2* tanpa operasi *replace* yang memiliki jarak $X - D$ terhadap *string ad2*. Gambar 2.2 adalah ilustrasi umum penyelesaian permasalahan klasik SPOJ 9967 *Playing With Words* dengan metode *meet in the middle*.

Sebagai contoh kasus ketika *string orig1* = *bd*, *string orig2* = *gj* dan $X = 5$. Langkah pertama untuk menyelesaikan kasus ini adalah dengan mencari semua kemungkinan kombinasi *string orig1* tanpa operasi *replace* dan semua kemungkinan *string orig2* dengan operasi *replace*. Tabel 2.3 menunjukkan seluruh kombinasi *string orig1* tanpa operasi *replace* beserta jarak masing-masing dengan *string ad1* dan Tabel 2.4 menunjukkan seluruh kombinasi *string orig2* dengan operasi *replace* beserta jarak masing-masing dengan *string ad2*. Berikutnya adalah mencari seluruh pasangan kombinasi *string orig1* tanpa operasi *replace* dengan jarak terhadap *string ad1* sebesar D dan kombinasi *string orig2* dengan operasi *replace* dengan jarak terhadap *string ad2* sebesar $X - D$ yang hasilnya dapat dilihat pada Tabel 2.5. Berikutnya hal yang sama juga dilakukan untuk kombinasi *string orig1* dengan operasi *replace* dan *string orig2* tanpa operasi *replace*. Tabel 2.6 menunjukkan



Gambar 2.2 Ilustrasi umum penyelesaian permasalahan dengan metode *meet in the middle* dengan operasi *replace*

Tabel 2.3 Kombinasi *string orig1* dengan nilai *string ad1 = bd* tanpa operasi *replace*

<i>orig1</i>	Jarak dengan <i>ad1</i>
<i>bd</i>	0
<i>db</i>	4

Tabel 2.4 Kombinasi *string orig2* dengan nilai *string ad2 = gj* dengan operasi *replace*

<i>orig2</i>	Jarak dengan <i>ad2</i>
<i>jj</i>	1
<i>hj</i>	1
<i>gi</i>	1
<i>gk</i>	1
<i>ig</i>	5
<i>kg</i>	7
<i>jf</i>	7
<i>jh</i>	5

seluruh kombinasi *string orig1* dengan operasi *replace* beserta jarak masing-masing dengan *string ad1* dan Tabel 2.7 menunjukkan seluruh kombinasi *string orig2* tanpa operasi *replace* beserta jarak masing-masing dengan *string ad2*. Berikutnya adalah mencari seluruh pasangan kombinasi *string orig1* dengan operasi *replace* dengan jarak terhadap *string ad1* sebesar D dan kombinasi *string orig2* dengan operasi *replace* tanpa jarak terhadap *string ad2* sebesar $X - D$ yang hasilnya dapat dilihat pada Tabel 2.8. Jawaban akhir dari kasus ketika *string orig1 = bd*, *string orig2 = gj* dan $X = 5$ adalah jumlah dari kombinasi pasangan *string orig1* dan *string orig2* pada Tabel 2.5 dan Tabel 2.8.

Pada deskripsi permasalahan klasik SPOJ 9967 *Playing With Words* jawaban akhir adalah banyak kemungkinan *string orig1*

Tabel 2.5 Kombinasi *string orig1* dengan nilai *string ad1 = bd* tanpa operasi *replace* dan *string orig2* dengan nilai *string ad2 = gj* dengan operasi *replace* dengan $X = 5$

<i>orig1</i>	Jarak dengan <i>ad1</i>	<i>orig2</i>	Jarak dengan <i>ad2</i>
<i>bd</i>	0	<i>ig</i>	5
<i>bd</i>	0	<i>jh</i>	5
<i>db</i>	4	<i>fj</i>	1
<i>db</i>	4	<i>hj</i>	1
<i>db</i>	4	<i>gi</i>	1
<i>db</i>	4	<i>gk</i>	1

Tabel 2.6 Kombinasi *string orig1* dengan nilai *string ad1 = bd* dengan operasi *replace*

<i>orig1</i>	Jarak dengan <i>ad1</i>
<i>ad</i>	1
<i>cd</i>	1
<i>bc</i>	1
<i>be</i>	1
<i>cb</i>	3
<i>eb</i>	5
<i>da</i>	5
<i>dc</i>	3

Tabel 2.7 Kombinasi *string orig2* dengan nilai *string ad2 = gj* tanpa operasi *replace*

<i>orig2</i>	Jarak dengan <i>ad2</i>
<i>gj</i>	0
<i>jk</i>	6

Tabel 2.8 Kombinasi *string orig1* dengan nilai *string ad1 = bd* dengan operasi *replace* dan *string orig2* dengan nilai *string ad2 = gj* tanpa operasi *replace* dengan $X = 5$

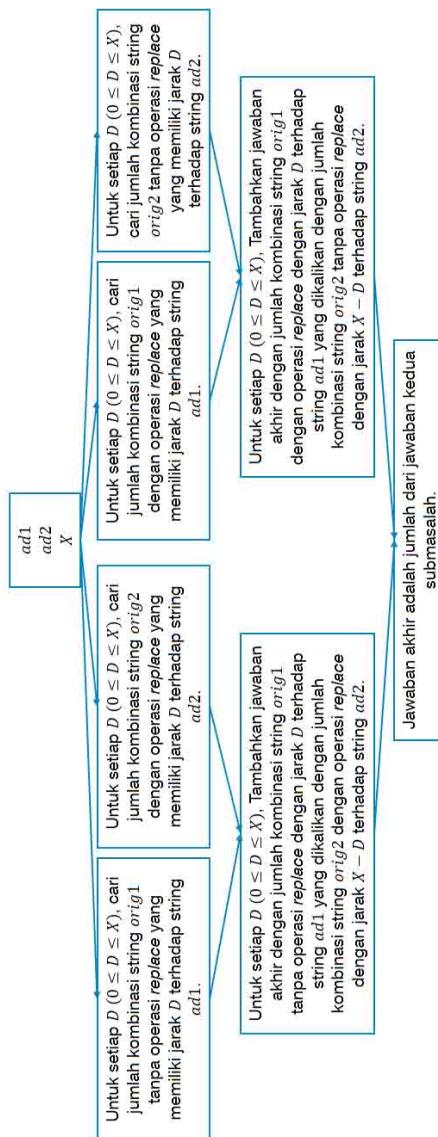
<i>orig1</i>	Jarak dengan <i>ad1</i>	<i>orig2</i>	Jarak dengan <i>ad2</i>
<i>eb</i>	5	<i>gj</i>	0
<i>da</i>	5	<i>gj</i>	0

dan *orig2* tanpa perlu menyertakan daftar *string orig1* dan *orig2* yang mungkin. Maka dari itu, proses perhitungan jawaban akhir dapat disederhanakan agar algoritma yang dibangun lebih optimal. Seperti yang terlihat pada ilustrasi umum penyelesaian pada Gambar 2.3, proses pencarian kombinasi *string orig1* dan *orig2* dapat diganti dengan hanya menghitung jumlah kemungkinan kombinasi *string orig1* dengan atau tanpa operasi *replace* dengan jarak terhadap *string ad1* sebesar D dan jumlah kemungkinan kombinasi *string orig2* dengan atau tanpa operasi *replace* dengan jarak terhadap *string ad2* sebesar D dengan $0 \leq D \leq X$.

2.3.2 Submasalah Optimal untuk Menghitung Jumlah Kombinasi *String Orig* dari *String Ad* Tanpa Operasi *Replace* dengan Jarak D

Ilustrasi pada Gambar 2.3 menunjukkan bahwa untuk mendapatkan jawaban akhir dari permasalahan klasik SPOJ 9967 *Playing With Words* salah satu langkah yang harus dilakukan adalah menghitung jumlah kombinasi *string orig* dari *string ad* tanpa operasi *replace* dengan jarak D . Terdapat dua kali perhitungan untuk proses ini yaitu perhitungan untuk mencari jumlah kombinasi *string orig1* dan *string orig2* tanpa operasi *replace* dengan jarak D .

Perhitungan jumlah kombinasi *string orig* tanpa operasi *replace* dapat dilakukan dengan memanfaatkan teknik *bitmasking*. Gambar 2.4 adalah ilustrasi perhitungan kombinasi *string orig* dari *string*



Gambar 2.3 Ilustrasi umum penyelesaian permasalahan dengan metode *meet in the middle* dengan operasi *replace* tanpa mempedulikan kombinasi string yang dihasilkan

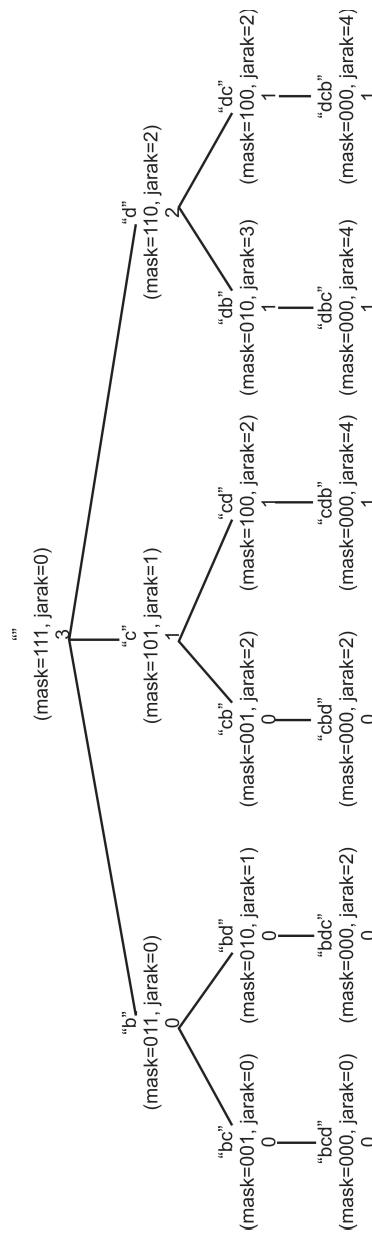
ad dengan nilai *string ad = bcd* dan $D = 4$ tanpa operasi *replace*. Nilai dari suatu *state* adalah jumlah dari nilai seluruh *state* yang merupakan *child* dari *state* tersebut dengan kasus dasar pada *state* dengan *mask* = 0, apabila *jarak* = D maka *state* tersebut akan bernilai 1, jika tidak maka akan bernilai 0.

Namun, seperti yang dapat dilihat pada Gambar 2.5, terdapat beberapa kasus di mana pada suatu *state*, *state* dengan nilai tersebut bersifat tumpang tindih dengan suatu *state* lain. Pada Gambar 2.5, *state* yang saling tumpang tindih ditandai dengan tulisan berwarna merah. Dengan adanya kasus *state* yang tumpang tindih tersebut dapat dimanfaatkan untuk melakukan optimasi pada algoritma yang dirancang dengan tidak melakukan perhitungan ulang pada *state* yang sudah pernah muncul sebelumnya. Sehingga proses perhitungan yang sebelumnya seperti dengan ilustrasi pada Gambar 2.4 dapat disederhanakan menjadi seperti yang terdapat pada Gambar 2.6. Teknik tersebut dikenal dengan teknik *dynamic programming*.

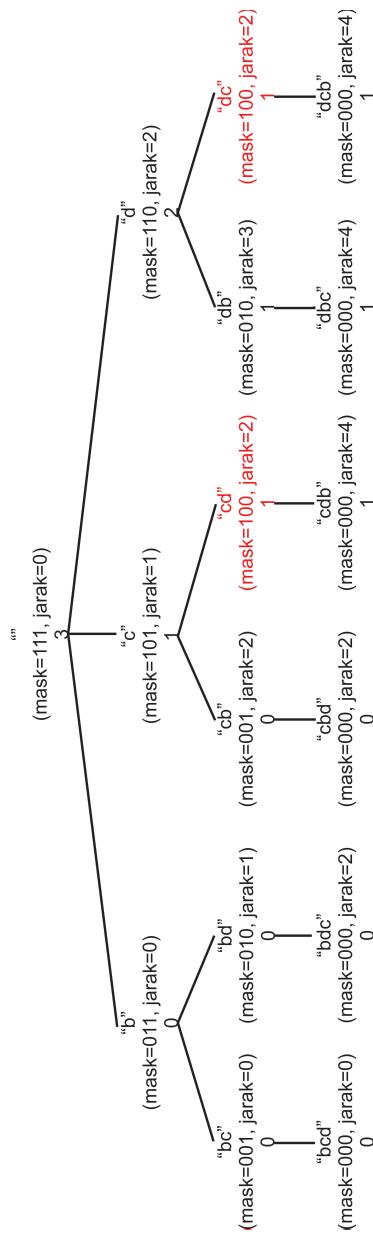
2.3.3 Submasalah Optimal untuk Menghitung Jumlah Kombinasi *String Orig* dari *String Ad* dengan Operasi *Replace* dengan Jarak D

Ilustrasi pada Gambar 2.3 menunjukkan bahwa untuk mendapatkan jawaban akhir dari permasalahan klasik SPOJ 9967 *Playing With Words* salah satu langkah yang harus dilakukan adalah menghitung jumlah kombinasi *string orig* dari *string ad* dengan operasi *replace* dengan jarak D . Terdapat dua kali perhitungan untuk proses ini yaitu perhitungan untuk mencari jumlah kombinasi *string orig1* dan *string orig2* tanpa operasi *replace* dengan jarak D .

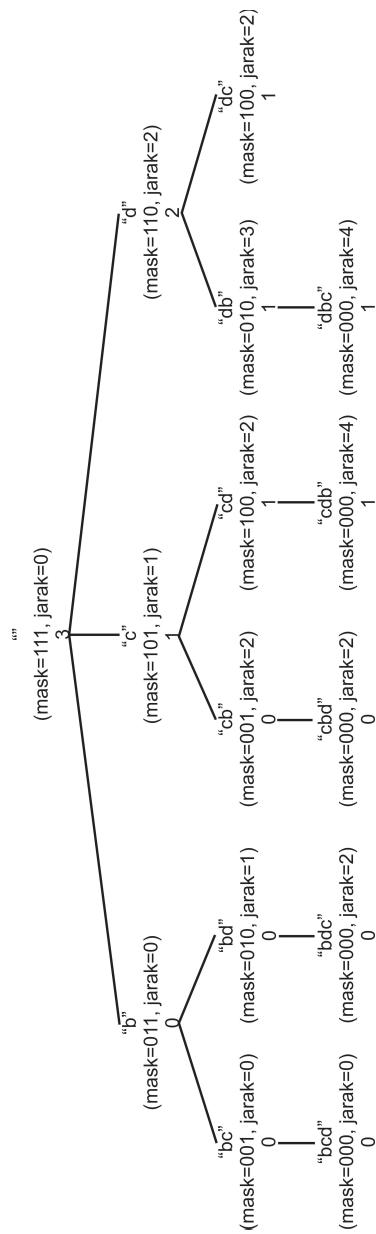
Cara untuk mendapatkan jawaban dari submasalah ini hampir sama dengan cara mencari jawaban pada submasalah perhitungan jumlah kombinasi *string orig* dari *string ad* tanpa operasi *replace*. Gambar 2.7 merupakan ilustrasi dari penyelesaian submasalah perhitungan



Gambar 2.4 Ilustrasi perhitungan jumlah kombinasi *string orig* dari *string ad* tanpa operasi *replace* dengan nilai *string ad = bcd* dan *D = 4*



Gambar 2.5 Contoh kasus tumpang tindih pada perhitungan kombinasi *string orig* dari *string ad* tanpa operasi *replace* dengan nilai *string ad* = *bcd* dan *D* = 4



Gambar 2.6 Ilustrasi perhitungan jumlah kombinasi *string orig* dari *string ad* tanpa operasi *replace* dengan nilai *string ad* = *bcd* dan *D* = 4 tanpa menghitung kasus yang tumpang tindih

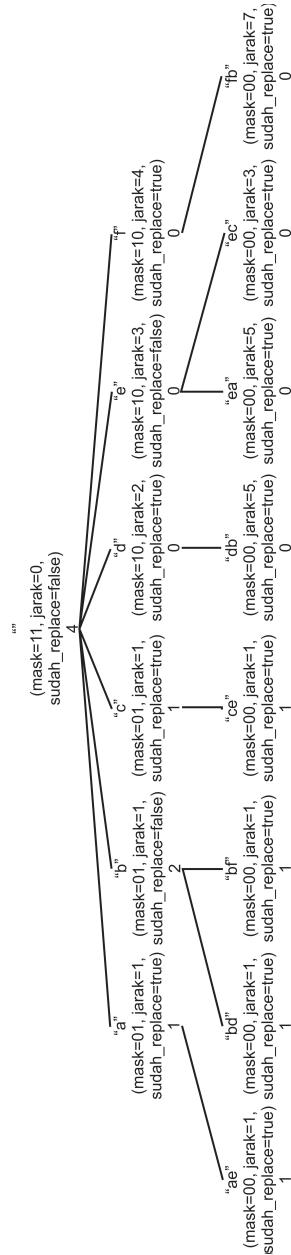
jumlah kombinasi *string orig* dari *string ad* dengan operasi *replace* dengan nilai *ad* = *be* dan *D* = 1. Sedikit berbeda dengan penyelesaian submasalah perhitungan jumlah kombinasi *string orig* dari *string ad* tanpa operasi *replace*, pada penyelesaian submasalah ini terdapat satu parameter lagi pada setiap *state*, yaitu penanda bahwa pada *state* tersebut sudah pernah melakukan operasi *replace* atau belum.

Sama seperti pada penyelesaian submasalah perhitungan jumlah kombinasi *string orig* dari *string ad* tanpa operasi *replace*, pada submasalah perhitungan jumlah kombinasi *string orig* dari *string ad* dengan operasi *replace* juga memiliki *state* yang saling tumpang tindih seperti yang terlihat pada Gambar 2.8 sehingga dapat dilakukan optimasi menggunakan teknik *dynamic programming* untuk meningkatkan efisiensi algoritma yang dibangun. Pada Gambar 2.9 dapat dilihat bahwa terdapat beberapa *state* yang ternyata memiliki kondisi yang mampu diselesaikan dengan metode yang sama dengan metode penyelesaian submasalah perhitungan jumlah kombinasi *string orig* dari *string ad* tanpa operasi *replace*. Sehingga penyelesaian submasalah perhitungan jumlah kombinasi *string orig* dari *string ad* dengan operasi *replace* dapat disederhanakan dengan memanfaatkan penyelesaian submasalah perhitungan jumlah kombinasi *string orig* dari *string ad* tanpa operasi *replace*.

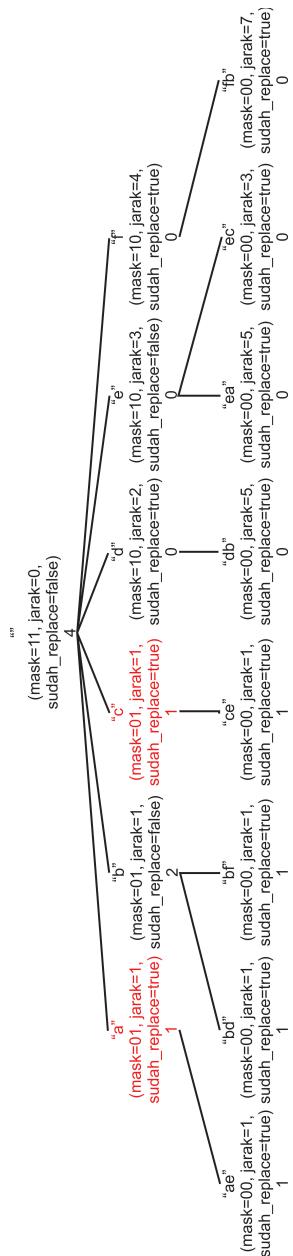
2.4 Pemodelan Relasi Rekurens

$$\begin{aligned} answer = \sum_{dist=0}^{dist=\min(X,250)} & ((F_{(S_0, 2^{|S_0|}, bound-dist)} \\ & G_{(S_1, 2^{|S_1|}, bound-X+dist)}) + (G_{(S_0, 2^{|S_0|}, bound-dist)} \\ & F_{(S_1, 2^{|S_1|}, bound-X+dist)})) \end{aligned} \quad * \quad * \quad (2.4.1)$$

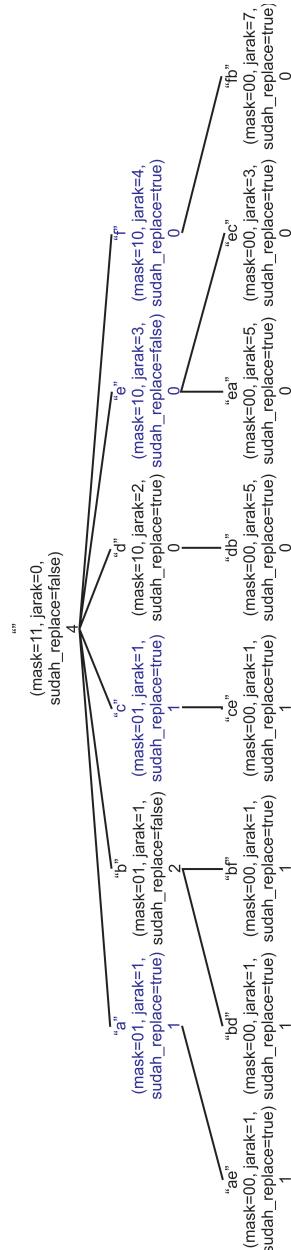
Pada subbab ini akan dijelaskan tentang relasi rekurens berdasarkan analisis pada subbab 2.3. Pada subbagian 2.3.1, dijelaskan bahwa



Gambar 2.7 Ilustrasi perhitungan jumlah kombinasi *string orig* dari *string ad* dengan operasi *replace* dengan nilai *string ad = be* dan *D = 1*



Gambar 2.8 Contoh kasus tumpang tindih pada perhitungan kombinasi *string orig* dari *string ad* dengan operasi *replace* dengan nilai *string ad = be* dan *D = 1*



Gambar 2.9 Submasalah perhitungan jumlah kombinasi *string orig* terhadap *string ad* tanpa operasi *replace* pada submasalah perhitungan jumlah kombinasi *string orig* terhadap *string ad* dengan operasi *replace*

permasalahan dapat dipecah menjadi dua submasalah yang dapat diselesaikan tanpa bergantung satu sama lain dengan memecah permasalahan berdasarkan masing-masing *string ad*. Karena terdapat sebuah operasi *replace* yang dilakukan, maka untuk menyelesaikan masing-masing submasalah harus dilakukan dua jenis perhitungan, yaitu operasi perhitungan jumlah kemungkinan *string orig* tanpa operasi *replace* dan operasi perhitungan jumlah kemungkinan *string orig* dengan operasi *replace*. Kedua operasi tersebut didefinisikan dalam bentuk fungsi sebagai berikut:

1. $F_{(S,mask,dist)}$, yaitu fungsi untuk menghitung jumlah kemungkinan *string* awal dari *string S* tanpa operasi *replace* di mana *S* adalah *string* awal yang akan dihitung, *mask* adalah nilai *bitmask* dan *dist* adalah jarak *string* awal dengan *string* yang dibentuk pada *state* tersebut.
2. $G_{(S,mask,dist)}$, yaitu fungsi untuk menghitung jumlah kemungkinan *string* awal dari *string S* dengan sekali operasi *replace* di mana *S* adalah *string* awal yang akan dihitung, *mask* adalah nilai *bitmask* dan *dist* adalah jarak *string* awal dengan *string* yang dibentuk pada *state* tersebut.

Jawaban permasalahan klasik SPOJ 9967 *Playing With Words* dapat dihitung dengan memanfaatkan kedua fungsi di atas. Persamaan 2.4.1 merupakan persamaan untuk menghitung jawaban utama dari permasalahan klasik SPOJ 9967 *Playing With Words* di mana S_0 adalah *string ad1*, S_1 adalah *string ad2*, X adalah jumlah jarak(*ad1, orig1*) dengan jarak(*ad2, orig2*) dan $bound = \min(250, X)$ dengan Tabel 2.9 adalah daftar notasi yang digunakan pada persamaan tersebut. Nilai $bound = \min(250, X)$ memiliki arti batas atas variabel *bound* adalah 250 karena panjang *string* masukan dijamin tidak lebih dari 10 karakter yang artinya jarak antar dua *string* pada permasalahan klasik SPOJ 9967 *Playing With Words* tidak mungkin melebihi angka 250. Sehingga apabila nilai masukan $X > 250$ dapat diasumsikan bahwa hal tersebut tidak

mungkin.

Tabel 2.9 Daftar notasi persamaan 2.4.1

Notasi	Deskripsi
$dist$	Nilai jarak yang diiterasi dari 0 hingga $\min(X, 250)$
X	Masukan yang merepresentasikan jumlah jarak string $orig1$ dengan string $ad1$ dan jarak string $orig2$ dengan string $ad2$
S_0	String masukan yang merepresentasikan $ad1$
S_1	String masukan yang merepresentasikan $ad2$
$ S_0 $	Panjang string $ad1$
$ S_1 $	Panjang string $ad2$
$bound$	Nilai batas jarak maksimal yang bernilai $\min(X, 250)$

2.4.1 Pemodelan Relasi Rekurens Submasalah Optimal untuk Menghitung Jumlah Kemungkinan **String** Awal Tanpa Operasi **Replace** dengan Jarak $X - dist$

$$F_{(S,mask,dist)} = \begin{cases} 0, & \text{if } dist > bound, \\ & \text{or } (mask = 0 \text{ and } dist \neq bound) \\ 1, & \text{if } mask = 0 \text{ and,} \\ & dist = bound \\ \sum_{i=0}^{i=NSB_{(mask)}} F1_{(S,mask,\text{set_bit}(mask)_i,dist)}, & \text{otherwise} \end{cases} \quad (2.4.2)$$

Tabel 2.10 Daftar notasi persamaan 2.4.2, 2.4.3 dan 2.4.4

Notasi	Deskripsi
S	$String$ yang akan dicari kemungkinan $string$ awalnya.
$mask$	Sebuah bilangan bulat yang bertugas sebagai <i>bitmask</i> yang merepresentasikan kondisi karakter mana saja yang sudah diambil pada kondisi (<i>state</i>) tersebut.
$dist$	Jarak $string$ yang sudah terbentuk pada kondisi tersebut dengan $string$ S dari $bound$ atau secara matematis dapat dituliskan dengan $bound - distance(currentString, S)$.
$bound$	Nilai batas jarak maksimal yang bernilai $\min(X, 250)$
idx	Index karakter pada $string$ S yang akan diambil atau digunakan.
$NSB_{(mask)}$	Mengembalikan jumlah angka 1 pada $mask$ apabila direpresentasikan dalam basis biner
$set_bit_{(mask)}$	Himpunan index bilangan bernilai satu dari $mask$ apabila direpresentasikan dalam basis biner.
$is_on_{(mask,idx)}$	Mengembalikan nilai <i>true</i> apabila bilangan pada index idx pada $mask$ bernilai 1 apabila direpresentasikan dalam basis biner.

$$F1_{(S,mask,idx,dist)} = \begin{cases} F_{(S,mask-2^{idx},dist+|S_{idx}-S_{curIdx}|)}, & \text{idx } = |S| - 1 \text{ or} \\ & \text{duplicate_rule1}_{(S,mask,idx)} = \\ & \text{True} \\ 0, & \text{otherwise} \end{cases} \quad (2.4.3)$$

$$\text{duplicate_rule1}_{(S,mask,idx)} = \begin{cases} \text{True}, & \text{if } idx < |S| - 1 \text{ and} \\ & (S_{idx} \neq S_{idx+1} \\ & \text{or } ((S_{idx} = S_{idx+1}) \text{ and} \\ & (\text{is_on}_{(mask,idx+1)} = \\ & \text{False})) \\ \text{False}, & \text{otherwise} \end{cases} \quad (2.4.4)$$

$$\text{is_on}_{(mask,idx)} = \begin{cases} \text{True}, & (mask \& 2^{idx}) = 1 \\ \text{False}, & \text{otherwise} \end{cases}$$

Pada bagian ini akan dijelaskan beberapa persamaan rekurens dengan daftar notasi seperti yang terdapat pada Tabel 2.10. Pada persamaan 2.4.1 terdapat fungsi $F_{(S,mask,dist)}$ yang merupakan fungsi untuk menghitung jumlah kemungkinan *string orig* dari *string S* tanpa operasi *replace* dengan jarak $X - dist$. Nilai dari fungsi $F_{(S,mask,dist)}$ adalah hasil penjumlahan seluruh *state* yang berhubungan, yaitu *state* $F_{(S,mask-2^{idx},dist+|S_{idx}-S_{curIdx}|)}$ di mana *curIdx* adalah jumlah *string* yang sudah dipilih pada *state* tersebut yang direpresentasikan dengan jumlah bit tidak menyala pada *mask*. Tidak semua *state* $F_{(S,mask-2^{idx},dist+|S_{idx}-S_{curIdx}|)}$ dijumlahkan untuk mendapatkan nilai dari fungsi $F_{(S,mask,dist)}$.

Hanya *state* yang valid yang nilainya dijumlahkan untuk membentuk nilai dari fungsi $F_{(S,mask,dist)}$. Persamaan 2.4.3 adalah persamaan rekurens untuk menentukan apakah *state* $F_{(S,mask-2^{idx},dist+|S_{idx}-S_{curIdx}|)}$ merupakan *state* yang valid dari sebuah *state* $F_{(S,mask,dist)}$. Persamaan 2.4.2 adalah relasi rekurens dari submasalah perhitungan jumlah kemungkinan *string orig* dari *string S* tanpa operasi *replace* dengan jarak $X - dist$.

Fungsi $duplicate_rule1_{(S,mask,idx)}$ adalah fungsi yang mencegah terjadinya perhitungan kombinasi *string* yang sama secara berulang. Contohnya pada kasus *string S = bcc*. Pada dasarnya, fungsi F akan melakukan perhitungan seluruh kombinasi *string S* yang mungkin sehingga hasil dari *string S* yang memiliki panjang 3 karakter adalah 6. Berikut adalah *string* yang merupakan kombinasi dari *string S* yang memiliki panjang 3 karakter:

1. $S_0S_1S_2$
2. $S_0S_2S_1$
3. $S_1S_0S_2$
4. $S_1S_2S_0$
5. $S_2S_0S_1$
6. $S_2S_1S_0$

Sehingga apabila *string S = bcc*, maka kombinasi *string* yang terbentuk adalah sebagai berikut:

1. *bcc*
2. *bcc*
3. *cbc*
4. *ccb*
5. *cbc*
6. *ccb*

Terdapat beberapa *string* yang bersifat duplikat sehingga tidak dapat dikatakan sebagai *string* yang berbeda. Sehingga banyak kombinasi *string* berbeda dari *S = bcc* adalah 3 dengan rincian

sebagai berikut:

1. *bcc*
2. *cbc*
3. *ccb*

Konsep dasar dari fungsi $duplicate_rule1_{(S,mask,idx)}$ adalah dengan menerapkan aturan hanya boleh memilih karakter S_{idx} apabila $S_{idx} = S_{idx+1}$ dan karakter tersebut telah dipilih sebelumnya, yang secara matematis didefinisikan dengan $mask \& 2^{idx+1} = 0$.

Untuk penjelasan fungsi $F_{(S,mask,dist)}$ yang lebih jelas, akan disimulasikan contoh pemanggilan fungsi $F_{(bcc,7,3)}$ pada kasus $S = bcc$ dan $X = 5$. Berikut adalah penjelasan rinci dari parameter fungsi yang dipanggil:

1. Parameter pertama yang bernilai *bcc* memiliki arti *string* masukan yang akan dicari jumlah kombinasi *string* awalnya tanpa operasi *replace* adalah $S = bcc$.
2. Parameter kedua yang bernilai 7 merepresentasikan bahwa pada *state* tersebut nilai *mask* = 7 atau apabila direpresentasikan dalam basis biner bernilai $mask = 111_{(2)}$ yang artinya pada *state* tersebut belum ada karakter pada S yang dipilih untuk melengkapi kombinasi *string* yang akan dicari.
3. parameter ketiga yang bernilai 3 merepresentasikan bahwa pada *state* tersebut nilai *dist* = 3 yang artinya pada *state* tersebut membutuhkan jarak sebesar *bound* – 3 untuk mencapai kondisi valid sebuah kombinasi *string* awal di mana *bound* = 5.

Karena himpunan $set_bit_{(7)} = \{0, 1, 2\}$, maka hasil dari pemanggilan fungsi $F_{(bcc,7,3)}$ adalah hasil dari penjumlahan hasil fungsi-fungsi yang akan dipanggil pada fungsi tersebut. Berikut adalah fungsi-fungsi yang dipanggil pada pemanggilan fungsi

$F_{(bcc,7,3)}$:

1. $F1_{(bcc,7,0,3)}$ yang akan memanggil fungsi $F_{(bcc,6,3)}$ karena memenuhi syarat $duplicate_rule1_{(bcc,7,0)} = True$.
2. $F1_{(bcc,7,1,3)}$ yang akan mengembalikan nilai 0 karena nilai $idx \neq |S| - 1$ dan nilai $duplicate_rule1_{(bcc,7,1)} \neq True$ bukan merupakan kondisi yang memenuhi syarat terpanggilnya fungsi $F_{(bcc,5,4)}$.
3. $F1_{(bcc,7,2,3)}$ yang akan memanggil fungsi $F_{(bcc,3,4)}$ karena memenuhi syarat $idx = |S| - 1$ di mana $|S| = 3$ sehingga $|S| - 1 = 2$ dan $idx = 2$.

Proses dilanjutkan secara rekursif, yaitu dengan pemanggilan fungsi $F_{(bcc,6,3)}$. Karena himpunan $set_bit_{(6)} = \{1, 2\}$, maka hasil dari pemanggilan fungsi $F_{(bcc,6,3)}$ adalah hasil dari penjumlahan hasil fungsi-fungsi yang akan dipanggil pada fungsi tersebut. Berikut adalah fungsi-fungsi yang dipanggil pada pemanggilan fungsi $F_{(bcc,6,3)}$:

1. $F1_{(bcc,6,1,3)}$ yang akan mengembalikan nilai 0 karena nilai $idx \neq |S| - 1$ dan nilai $duplicate_rule1_{(bcc,6,0)} \neq True$ bukan merupakan kondisi yang memenuhi syarat terpanggilnya fungsi $F_{(bcc,4,3)}$.
2. $F1_{(bcc,6,2,3)}$ yang akan memanggil fungsi $F_{(bcc,2,3)}$ karena memenuhi syarat $idx = |S| - 1$ di mana $|S| = 3$ sehingga $|S| - 1 = 2$ dan $idx = 2$.

Proses berikutnya adalah pemanggilan fungsi $F_{(bcc,2,3)}$. Karena himpunan $set_bit_{(2)} = \{1\}$, maka nilai dari fungsi $F_{(bcc,2,3)}$ sama dengan nilai dari satu-satunya fungsi yang dipanggil pada fungsi tersebut, yaitu $F1_{(bcc,2,1,3)}$. Fungsi $F1_{(bcc,2,1,3)}$ akan memanggil fungsi $F_{(bcc,0,3)}$ karena memenuhi kondisi nilai $duplicate_rule1_{(bcc,2,1)} = True$. Fungsi $F_{(bcc,0,3)}$ sendiri akan mengembalikan nilai 0 karena kondisi ketika $mask = 0$ adalah kondisi dasar (*base case*) dan $dist \neq bound$ di mana $dist = 3$

dan $bound = 5$. Sehingga nilai dari fungsi $F_{(bcc,2,3)} = 0$, fungsi $F_{(bcc,6,3)} = 0$ dan nilai dari fungsi $F_{(bcc,7,3)} = 0$.

Berikutnya adalah perhitungan fungsi $F_{(bcc,3,4)}$. Karena himpunan $set_bit_{(3)} = \{0, 1\}$, maka hasil dari pemanggilan fungsi $F_{(bcc,3,4)}$ adalah hasil dari penjumlahan hasil fungsi-fungsi yang akan dipanggil pada fungsi tersebut. Berikut adalah fungsi-fungsi yang dipanggil pada pemanggilan fungsi $F_{(bcc,3,4)}$:

1. $F1_{(bcc,3,0,4)}$ yang akan memanggil fungsi $F_{(bcc,2,5)}$ karena memenuhi syarat $duplicate_rule1_{(bcc,3,0)} = True$.
2. $F1_{(bcc,3,1,4)}$ yang akan memanggil fungsi $F_{(bcc,1,4)}$ karena memenuhi syarat $duplicate_rule1_{(bcc,3,1)} = True$.

Proses berikutnya adalah pemanggilan fungsi $F_{(bcc,2,5)}$. Karena himpunan $set_bit_{(2)} = \{1\}$, maka nilai dari fungsi $F_{(bcc,2,5)}$ sama dengan nilai dari satu-satunya fungsi yang dipanggil pada fungsi tersebut, yaitu $F1_{(bcc,2,1,5)}$. Fungsi $F1_{(bcc,2,1,5)}$ akan memanggil fungsi $F_{(bcc,0,5)}$ karena memenuhi kondisi nilai $duplicate_rule1_{(bcc,2,1)} = True$. Fungsi $F_{(bcc,0,5)}$ sendiri akan mengembalikan nilai 1 karena kondisi ketika $mask = 0$ adalah kondisi dasar (*base case*) dan $dist = bound$ di mana $dist = 5$ dan $bound = 5$. Sehingga nilai dari fungsi $F_{(bcc,2,5)} = 1$.

Berikutnya adalah pemanggilan fungsi $F_{(bcc,1,4)}$. Karena himpunan $set_bit_{(1)} = \{0\}$, maka nilai dari fungsi $F_{(bcc,1,4)}$ sama dengan nilai dari satu-satunya fungsi yang dipanggil pada fungsi tersebut, yaitu $F1_{(bcc,1,0,4)}$. Fungsi $F1_{(bcc,1,0,4)}$ akan memanggil fungsi $F_{(bcc,0,5)}$ karena memenuhi kondisi nilai $duplicate_rule1_{(bcc,1,0)} = True$. Fungsi $F_{(bcc,0,5)}$ sendiri akan mengembalikan nilai 1 karena kondisi ketika $mask = 0$ adalah kondisi dasar (*base case*) dan $dist = bound$ di mana $dist = 5$ dan $bound = 5$. Sehingga nilai dari fungsi $F_{(bcc,1,4)} = 1$, fungsi $F_{(bcc,3,4)} = 2$ dan nilai akhir dari fungsi $F_{(bcc,7,3)} = 2$.

2.4.2 Pemodelan Relasi Rekurens Submasalah Optimal untuk Menghitung Jumlah Kemungkinan String Awal dengan Sekali Operasi *Replace* dengan Jarak $X - dist$

$$G_{(S, mask, dist)} = \begin{cases} 0, & dist > bound \text{ or} \\ & mask = bound \\ \sum_{i=0}^{i=NSB(mask)} & \\ (G1_{(S, mask,} & \\ set_bit_{(mask)i, dist}) & \\ + G2_{(S, mask,} & \\ set_bit_{(mask)i, dist}) & \\ + G3_{(S, mask,} & \\ set_bit_{(mask)i, dist})), & \text{otherwise} \end{cases} \quad (2.4.5)$$

$$G1_{(S, mask, idx, dist)} = \begin{cases} G_{(S, mask} & \\ - 2^{idx}, dist & \\ + |S_{idx}|, & idx = |S| - 1 \text{ or} \\ - S_{curIdx}|), & duplicate_rule1_{(S, mask, idx)} = \\ True & \\ 0, & \text{otherwise} \end{cases} \quad (2.4.6)$$

Tabel 2.11 Daftar notasi persamaan 2.4.5, 2.4.6, 2.4.7, 2.4.8, 2.4.9 dan 2.4.10 (1)

Notasi	Deskripsi
S	$String$ yang akan dicari kemungkinan $string$ awalnya.
$mask$	Sebuah bilangan bulat yang bertugas sebagai $bitmask$ yang merepresentasikan kondisi karakter mana saja yang sudah diambil pada kondisi ($state$) tersebut.
$dist$	Jarak $string$ yang sudah terbentuk pada kondisi tersebut dengan $string S$ dari $bound$ atau secara matematis dapat dituliskan dengan $bound - distance(currentString, S)$.
$bound$	Nilai batas jarak maksimal yang bernilai $\min(X, 250)$
idx	Index karakter pada $string S$ yang akan diambil atau digunakan.
$NSB_{(mask)}$	Mengembalikan jumlah angka 1 pada $mask$ apabila direpresentasikan dalam basis biner
$set_bit_{(mask)}$	Himpunan index bilangan bernilai satu dari $mask$ apabila direpresentasikan dalam basis biner.
$is_on_{(mask, idx)}$	Mengembalikan nilai $true$ apabila bilangan pada index idx pada $mask$ dalam basis biner bernilai 1.

Tabel 2.12 Daftar notasi persamaan 2.4.5, 2.4.6, 2.4.7, 2.4.8, 2.4.9 dan 2.4.10 (2)

Notasi	Deskripsi
$charLastPos_{(S,C)}$	Index terbesar dari karakter C pada string S .
$charFirstPos_{(S,C)}$	Index terkecil dari karakter C pada string S .
$curIdx$	Angka yang merepresentasikan panjang string $orig$ pada state tersebut. Nilai $curIdx$ adalah jumlah bit yang bernilai 0 pada $mask$.

$$G2_{(S,mask,idx,dist)} = \begin{cases} F_{(S,mask \\ -2^{idx},dist \\ +|S_{idx}|+1 \\ -|S_{curIdx}|)}, & idx = |S| - 1 \text{ or} \\ & (duplicate_rule1_{(S,mask,idx)} = \\ & True \quad \quad \quad \text{and} \\ & duplicate_rule2_{(S,mask,idx)} = \\ & True) \\ 0, & \text{otherwise} \end{cases} \quad (2.4.7)$$

$$G3_{(S,mask,idx,dist)} = \begin{cases} F_{(S,mask \\ -2^{idx},dist \\ +|S_{idx}-1 \\ -S_{curIdx}|)}, & (idx = |S| - 1 \text{ or} \\ & \text{duplicate_rule1}_{(S,mask,idx)} = \\ & \text{True}) \text{ and } (idx = 0 \text{ or} \\ & \text{duplicate_rule3}_{(S,mask,idx)} = \\ & \text{True}) \\ 0, & \text{otherwise} \end{cases} \quad (2.4.8)$$

$$\text{duplicate_rule2}_{(S,mask,idx)} = \begin{cases} \text{True}, & \text{if } idx < |S| - 1 \text{ and} \\ & (\text{charFirstPos}_{(S,S_{idx}+1)} = \\ & -1 \quad \quad \quad \text{or} \\ & (\text{charFirstPos}_{(S,S_{idx}+1)} \neq \\ & -1 \text{ and } \text{is_on}_{(mask,} \\ & \text{charFirstPos}_{(S,S_{idx}+1)}) = \\ & \text{False} \\ \text{False}, & \text{otherwise} \end{cases} \quad (2.4.9)$$

Pada bagian ini akan dijelaskan beberapa persamaan rekurens dengan daftar notasi seperti yang terdapat pada Tabel 2.11 dan Tabel 2.12. Pada persamaan 2.4.1 terdapat fungsi $G_{(S,mask,dist)}$ yang merupakan fungsi untuk menghitung jumlah kemungkinan *string orig* dari *string S* dengan sekali operasi *replace* dengan jarak $X - dist$. Sama halnya dengan fungsi $F_{(S,mask,dist)}$, nilai dari fungsi $G_{(S,mask,dist)}$ adalah hasil penjumlahan dari seluruh *state* yang berhubungan dan valid. Terdapat tiga kasus *state* yang

mungkin, yaitu:

$$\text{duplicate_rule3}(S, \text{mask}, \text{idx}) = \begin{cases} \text{True}, & \text{if } \text{idx} > 0 - 1 \text{ and} \\ & (\text{charLastPos}(S, S_{\text{idx}-1}) = -1 \text{ or} \\ & (\text{charLastPos}(S, S_{\text{idx}-1}) \neq -1 \text{ and} \\ & \text{is_on}(\text{mask}, \\ & \text{charLastPos}_{(S, S_{\text{idx}-1})}) = \\ & \text{True} \\ \text{False}, & \text{otherwise} \end{cases} \quad (2.4.10)$$

1. State $G_{(S, \text{mask}-2^{\text{idx}}, \text{dist}+|\text{S}_{\text{idx}}-\text{S}_{\text{curIdx}}|)}$ dengan kasus ketika memilih S_{idx} sebagai $\text{orig}_{\text{curIdx}}$ tanpa melakukan *replace*.
2. State $F_{(S, \text{mask}-2^{\text{idx}}, \text{dist}+|\text{S}_{\text{idx}}+1-\text{S}_{\text{curIdx}}|)}$ dengan kasus ketika memilih $S_{\text{idx}} + 1$, dengan kata lain mengganti karakter S_{idx} dengan karakter setelahnya dalam alfabet sebagai $\text{orig}_{\text{curIdx}}$.
3. State $F_{(S, \text{mask}-2^{\text{idx}}, \text{dist}+|\text{S}_{\text{idx}}-1-\text{S}_{\text{curIdx}}|)}$ dengan kasus ketika memilih $S_{\text{idx}} - 1$, dengan kata lain mengganti karakter S_{idx} dengan karakter sebelumnya dalam alfabet, sebagai $\text{orig}_{\text{curIdx}}$.

Masing-masing jenis *state* yang berhubungan langsung dengan *state* $G_{(S, \text{mask}, \text{dist})}$ memiliki syarat tersendiri untuk menjadi sebuah *state* yang valid. Berikut adalah syarat dari masing-masing jenis *state* yang dapat dibentuk dari *state* $G_{(S, \text{mask}, \text{dist})}$:

1. Persamaan 2.4.6 adalah persamaan yang menentukan apakah *state* $G_{(S, \text{mask}-2^{\text{idx}}, \text{dist}+|\text{S}_{\text{idx}}-\text{S}_{\text{curIdx}}|)}$ merupakan sebuah *state* yang valid dari *state* $G_{(S, \text{mask}, \text{dist})}$.
2. Persamaan 2.4.7 adalah persamaan yang menentukan apakah *state* $F_{(S, \text{mask}-2^{\text{idx}}, \text{dist}+|\text{S}_{\text{idx}}+1-\text{S}_{\text{curIdx}}|)}$ merupakan sebuah

state yang valid dari *state* $G_{(S,mask,dist)}$.

3. Persamaan 2.4.8 adalah persamaan yang menentukan apakah *state* $F_{(S,mask-2^{idx},dist+|S_{idx}-1-S_{curIdx}|)}$ merupakan sebuah *state* yang valid dari *state* $G_{(S,mask,dist)}$.

Fungsi $duplicate_rule2_{(S,mask,idx)}$ dan fungsi $duplicate_rule3_{(S,mask,idx)}$ adalah fungsi yang mencegah terjadinya perhitungan kombinasi *string* yang sama secara berulang setelah operasi *replace* dilakukan. Contohnya pada kasus *string* $S = bcc$. Pada dasarnya, fungsi $G_{(S,mask,dist)}$ akan melakukan perhitungan seluruh kombinasi *string* S yang mungkin dengan sekali operasi *replace* sehingga hasil dari *string* S yang memiliki panjang 3 karakter adalah 36. Berikut adalah *string* yang merupakan kombinasi dari *string* S yang memiliki panjang 3 karakter dengan sekali operasi *replace*:

1. $S_0 + 1S_1S_2$
2. $S_0S_1 + 1S_2$
3. $S_0S_1S_2 + 1$
4. $S_0 - 1S_1S_2$
5. $S_0S_1 - 1S_2$
6. $S_0S_1S_2 - 1$
7. $S_0 + 1S_2S_1$
8. $S_0S_2 + 1S_1$
9. $S_0S_2S_1 + 1$
10. $S_0 - 1S_2S_1$
11. $S_0S_2 - 1S_1$
12. $S_0S_2S_1 - 1$
13. $S_1 + 1S_0S_2$
14. $S_1S_0 + 1S_2$
15. $S_1S_0S_2 + 1$
16. $S_1 - 1S_0S_2$
17. $S_1S_0 - 1S_2$
18. $S_1S_0S_2 - 1$

19. $S_1 + 1S_2S_0$
20. $S_1S_2 + 1S_0$
21. $S_1S_2S_0 + 1$
22. $S_1 - 1S_2S_0$
23. $S_1S_2 - 1S_0$
24. $S_1S_2S_0 - 1$
25. $S_2 + 1S_0S_1$
26. $S_2S_0 + 1S_1$
27. $S_2S_0S_1 + 1$
28. $S_2 - 1S_0S_1$
29. $S_2S_0 - 1S_1$
30. $S_2S_0S_1 - 1$
31. $S_2 + 1S_1S_0$
32. $S_2S_1 + 1S_0$
33. $S_2S_1S_0 + 1$
34. $S_2 - 1S_1S_0$
35. $S_2S_1 - 1S_0$
36. $S_2S_1S_0 - 1$

Sehingga apabila *string* $S = bcc$, maka kombinasi *string* yang terbentuk adalah sebagai berikut:

1. *ccc*
2. *bdc*
3. *bcd*
4. *acc*
5. *bbc*
6. *bcb*
7. *ccc*
8. *bdc*
9. *bcd*
10. *acc*
11. *bbc*
12. *bcb*

13. *dbc*
14. *ccc*
15. *cbd*
16. *bbc*
17. *cac*
18. *cbb*
19. *dcb*
20. *cdb*
21. *ccc*
22. *bcb*
23. *cbb*
24. *cca*
25. *dbc*
26. *ccc*
27. *cbd*
28. *bbc*
29. *cac*
30. *cbb*
31. *dcb*
32. *cdb*
33. *ccc*
34. *bcb*
35. *cbb*
36. *cca*

Terdapat beberapa *string* yang bersifat duplikat sehingga tidak dapat dikatakan sebagai *string* yang berbeda. Sehingga banyak kombinasi *string* berbeda dari $S = bcc$ dengan sekali operasi *replace* adalah 13 dengan rincian sebagai berikut:

1. *ccc*
2. *bdc*
3. *bcd*
4. *acc*

5. *bbc*
6. *bcb*
7. *dbc*
8. *cbd*
9. *cac*
10. *cbb*
11. *dcb*
12. *cdb*
13. *cca*

Konsep dasar dari fungsi $duplicate_rule2_{(S,mask,idx)}$ mirip dengan konsep dasar dari fungsi $duplicate_rule1_{(S,mask,idx)}$. Hanya saja karena pada fungsi $G_{(S,mask,dist)}$ terdapat kondisi di mana karakter S_i yang dipilih diganti dengan karakter $S_i + 1$ yang merupakan karakter berikutnya dalam alfabet, aturan yang diterapkan berbeda dengan fungsi $duplicate_rule1_{(S,mask,idx)}$. Pada fungsi $duplicate_rule1_{(S,mask,idx)}$ diterapkan aturan hanya boleh memilih sebuah karakter S_{idx} apabila tidak ada karakter $S_{idx} + 1$ yang muncul pada *string S* atau karakter $S_{idx} + 1$ pertama yang muncul atau dengan kata lain karakter $S_{idx} + 1$ dengan index terkecil pada *string S* telah dipilih sebelumnya.

Fungsi $duplicate_rule3_{(S,mask,idx)}$ memiliki konsep dasar yang mirip dengan fungsi $duplicate_rule2_{(S,mask,idx)}$. Hanya saja pada fungsi $duplicate_rule3_{(S,mask,idx)}$ bertujuan untuk mencegah duplikasi pada kondisi fungsi $G_{(S,mask,dist)}$ memilih sebuah karakter S_{idx} yang berikutnya digantikan dengan karakter $S_{idx} - 1$ yang merupakan karakter sebelumnya dalam alfabet. Sehingga aturan yang diterapkan adalah hanya boleh memilih karakter S_{idx} apabila karakter $S_{idx} - 1$ yang merupakan karakter sebelumnya pada alfabet tidak muncul pada *string S* atau karakter $S_{idx} - 1$ yang terakhir muncul atau dengan kata lain karakter $S_{idx} - 1$ dengan index terbesar pada *string S* belum dipilih sebelumnya.

Untuk penjelasan fungsi $G_{(S,mask,dist)}$ yang lebih jelas, akan

disimulasikan contoh pemanggilan fungsi $G_{(bcc,7,2)}$ pada kasus $S = bcc$ dan $X = 3$. Berikut adalah penjelasan rinci dari parameter fungsi yang dipanggil:

1. Parameter pertama yang bernilai bcc memiliki arti *string* masukan yang akan dicari jumlah kombinasi *string* awalnya tanpa operasi *replace* adalah $S = bcc$.
2. Parameter kedua yang bernilai 7 merepresentasikan bahwa pada *state* tersebut nilai *mask* = 7 atau apabila direpresentasikan dalam basis biner bernilai $mask = 111_{(2)}$ yang artinya pada *state* tersebut belum ada karakter pada S yang dipilih untuk melengkapi kombinasi *string* yang akan dicari.
3. parameter ketiga yang bernilai 2 merepresentasikan bahwa pada *state* tersebut nilai *dist* = 2 yang artinya pada *state* tersebut membutuhkan jarak sebesar *bound* – 2 untuk mencapai kondisi valid sebuah kombinasi *string* awal di mana *bound* = 3.

Karena himpunan $set_bit_{(7)} = \{0, 1, 2\}$, maka hasil dari pemanggilan fungsi $G_{(bcc,7,2)}$ adalah hasil dari penjumlahan hasil fungsi-fungsi yang akan dipanggil pada fungsi tersebut. Berikut adalah fungsi-fungsi yang dipanggil pada pemanggilan fungsi $G_{(bcc,7,2)}$:

1. Fungsi $G1_{(bcc,7,0,2)}$ yang akan memanggil fungsi $G_{(bcc,6,2)}$ karena memenuhi kondisi $duplicate_rule1_{(bcc,7,0)} = True$.
2. Fungsi $G2_{(bcc,7,0,2)}$ yang mana karena $idx \neq |S| - 1$ dan $duplicate_rule2_{(bcc,7,0)} = False$ tidak akan memanggil fungsi $F_{(bcc,6,3)}$ dan akan mengembalikan nilai 0.
3. Fungsi $G3_{(bcc,7,0,2)}$ yang akan memanggil fungsi $F_{(bcc,6,3)}$ karena memenuhi kondisi $duplicate_rule1_{(bcc,7,0)} = True$ dan $duplicate_rule3_{(bcc,7,0)} = True$. Fungsi $F_{(bcc,6,3)}$ akan mengembalikan nilai 1.
4. Fungsi $G1_{(bcc,7,1,2)}$ yang mana karena $idx \neq |S| - 1$ dan

$duplicate_rule1_{(bcc,7,1)} = False$ tidak akan memanggil fungsi $G_{(bcc,5,3)}$ dan akan mengembalikan nilai 0.

5. Fungsi $G2_{(bcc,7,1,2)}$ yang mana karena $idx \neq |S| - 1$ dan $duplicate_rule1_{(bcc,7,1)} = False$ tidak akan memanggil fungsi $F_{(bcc,5,4)}$ dan akan mengembalikan nilai 0.
6. Fungsi $G3_{(bcc,7,1,2)}$ yang mana karena $idx \neq |S| - 1$ dan $duplicate_rule1_{(bcc,7,1)} = False$ tidak akan memanggil fungsi $F_{(bcc,5,2)}$ dan akan mengembalikan nilai 0.
7. Fungsi $G1_{(bcc,7,2,2)}$ yang akan memanggil fungsi $G_{(bcc,3,3)}$ karena memenuhi kondisi $idx = |S| - 1$.
8. Fungsi $G2_{(bcc,7,2,2)}$ yang akan memanggil fungsi $F_{(bcc,3,4)}$ karena memenuhi kondisi $idx = |S| - 1$. Fungsi $F_{(bcc,3,4)}$ akan mengembalikan nilai 0.
9. Fungsi $G3_{(bcc,7,2,2)}$ yang akan memanggil fungsi $F_{(bcc,3,2)}$ karena memenuhi kondisi $idx = |S| - 1$. Fungsi $F_{(bcc,3,2)}$ akan mengembalikan nilai 2.

Berikutnya adalah pemanggilan fungsi $G_{(bcc,6,2)}$. Karena himpunan $set_bit_{(6)} = \{1, 2\}$, maka hasil dari pemanggilan fungsi $G_{(bcc,6,2)}$ adalah hasil dari penjumlahan hasil fungsi-fungsi yang akan dipanggil pada fungsi tersebut. Berikut adalah fungsi-fungsi yang dipanggil pada pemanggilan fungsi $G_{(bcc,6,2)}$:

1. Fungsi $G1_{(bcc,6,1,2)}$ yang mana karena $idx \neq |S| - 1$ dan $duplicate_rule1_{(bcc,6,1)} = False$ tidak akan memanggil fungsi $G_{(bcc,4,2)}$ dan akan mengembalikan nilai 0.
2. Fungsi $G2_{(bcc,6,1,2)}$ yang mana karena $idx \neq |S| - 1$ dan $duplicate_rule1_{(bcc,6,1)} = False$ tidak akan memanggil fungsi $F_{(bcc,4,3)}$ dan akan mengembalikan nilai 0.
3. Fungsi $G3_{(bcc,6,1,2)}$ yang mana karena $idx \neq |S| - 1$ dan $duplicate_rule1_{(bcc,6,1)} = False$ tidak akan memanggil fungsi $F_{(bcc,4,3)}$ dan akan mengembalikan nilai 0.
4. Fungsi $G1_{(bcc,6,2,2)}$ yang akan memanggil fungsi $G_{(bcc,2,2)}$ karena memenuhi kondisi $idx = |S| - 1$.

5. Fungsi $G2_{(bcc,6,2,2)}$ yang akan memanggil fungsi $F_{(bcc,2,3)}$ karena memenuhi kondisi $idx = |S| - 1$. Fungsi $F_{(bcc,2,3)}$ akan mengembalikan nilai 1.
6. Fungsi $G3_{(bcc,6,2,2)}$ yang mana karena $idx \neq 0$ dan $duplicate_rule3_{(bcc,6,1)} = False$ tidak akan memanggil fungsi $F_{(bcc,2,3)}$ dan akan mengembalikan nilai 0.

Berikutnya adalah pemanggilan fungsi $G_{(bcc,2,2)}$. Karena himpunan $set_bit_{(2)} = \{1\}$, maka hasil dari pemanggilan fungsi $G_{(bcc,2,2)}$ adalah hasil dari penjumlahan hasil fungsi-fungsi yang akan dipanggil pada fungsi tersebut. Berikut adalah fungsi-fungsi yang dipanggil pada pemanggilan fungsi $G_{(bcc,2,2)}$:

1. Fungsi $G1_{(bcc,2,1,2)}$ yang akan memanggil fungsi $G_{(bcc,0,2)}$ karena memenuhi kondisi $duplicate_rule1_{(bcc,2,1)} = True$. Fungsi $G_{(bcc,0,2)}$ akan mengembalikan nilai 0 karena pada kasus dasar $mask = 0$, fungsi $G_{(S,mask,dist)}$ akan selalu mengembalikan nilai 0.
2. Fungsi $G2_{(bcc,2,1,2)}$ yang akan memanggil fungsi $F_{(bcc,0,3)}$ karena memenuhi kondisi $duplicate_rule1_{(bcc,2,1)} = True$ dan $duplicate_rule2_{(bcc,2,1)} = True$. Fungsi $F_{(bcc,0,3)}$ akan mengembalikan nilai 1.
3. Fungsi $G3_{(bcc,2,1,2)}$ yang mana karena $idx \neq |S| - 1$ dan $duplicate_rule3_{(bcc,2,1)} = False$ tidak akan memanggil fungsi $F_{(bcc,0,3)}$ dan akan mengembalikan nilai 0.

Sehingga nilai fungsi $G_{(bcc,2,2)} = 1$ dan nilai fungsi $G_{(bcc,6,2)} = 2$. Berikutnya adalah pemanggilan fungsi $G_{(bcc,3,3)}$. Karena himpunan $set_bit_{(3)} = \{0, 1\}$, maka hasil dari pemanggilan fungsi $G_{(bcc,3,3)}$ adalah hasil dari penjumlahan hasil fungsi-fungsi yang akan dipanggil pada fungsi tersebut. Berikut adalah fungsi-fungsi yang dipanggil pada pemanggilan fungsi $G_{(bcc,3,3)}$:

1. Fungsi $G1_{(bcc,3,0,3)}$ yang akan memanggil fungsi $G_{(bcc,2,4)}$ karena memenuhi kondisi $duplicate_rule1_{(bcc,3,0)} = True$.

Nilai dari fungsi $G_{(bcc,2,4)}$ adalah 0 karena kasus dasar dari fungsi $G_{(S,mask,dist)}$ ketika $dist > bound$ akan mengembalikan nilai 0.

2. Fungsi $G2_{(bcc,3,0,3)}$ yang mana karena $idx \neq |S| - 1$ dan $duplicate_rule2_{(bcc,3,0)} = False$ tidak akan memanggil fungsi $F_{(bcc,2,3)}$ dan akan mengembalikan nilai 0.
3. Fungsi $G3_{(bcc,3,0,3)}$ yang akan memanggil fungsi $F_{(bcc,2,5)}$ karena memenuhi kondisi $duplicate_rule1_{(bcc,3,0)} = True$ dan $idx = 0$. Nilai dari $F_{(bcc,2,5)}$ adalah 0.
4. Fungsi $G1_{(bcc,3,1,3)}$ yang akan memanggil fungsi $G_{(bcc,1,3)}$ karena memenuhi kondisi $duplicate_rule1_{(bcc,3,0)} = True$.
5. Fungsi $G2_{(bcc,3,1,3)}$ yang akan memanggil fungsi $F_{(bcc,1,4)}$ karena memenuhi kondisi $duplicate_rule1_{(bcc,3,1)} = True$ dan $duplicate_rule2_{(bcc,3,1)} = True$. Nilai dari $F_{(bcc,1,4)}$ adalah 0.
6. Fungsi $G3_{(bcc,3,1,3)}$ yang akan memanggil fungsi $F_{(bcc,1,4)}$ karena memenuhi kondisi $duplicate_rule1_{(bcc,3,1)} = True$ dan $duplicate_rule3_{(bcc,3,1)} = True$. Nilai dari $F_{(bcc,1,4)}$ adalah 0.

Berikutnya adalah pemanggilan fungsi $G_{(bcc,1,3)}$. Karena himpunan $set_bit_{(1)} = \{0\}$, maka hasil dari pemanggilan fungsi $G_{(bcc,1,3)}$ adalah hasil dari penjumlahan hasil fungsi-fungsi yang akan dipanggil pada fungsi tersebut. Berikut adalah fungsi-fungsi yang dipanggil pada pemanggilan fungsi $G_{(bcc,1,3)}$:

1. Fungsi $G1_{(bcc,1,0,3)}$ yang akan memanggil fungsi $G_{(bcc,0,4)}$ karena memenuhi kondisi $duplicate_rule1_{(bcc,1,0)} = True$. Nilai dari fungsi $G_{(bcc,0,4)}$ adalah 0 karena pada kasus dasar fungsi $G_{(S,mask,dist)}$, ketika $mask = 0$ fungsi $G_{(S,mask,dist)}$ akan mengembalikan nilai 0.
2. Fungsi $G2_{(bcc,1,0,3)}$ yang akan memanggil fungsi $F_{(bcc,0,3)}$ karena memenuhi kondisi $duplicate_rule1_{(bcc,1,0)} = True$ dan $duplicate_rule2_{(bcc,1,0)} = True$. Nilai dari fungsi

$F_{(bcc,0,3)}$ adalah 1.

3. Fungsi $G3_{(bcc,1,0,3)}$ yang akan memanggil fungsi $F_{(bcc,0,5)}$ karena memenuhi kondisi $duplicate_rule1_{(bcc,1,0)} = True$ dan $idx = 0$. Nilai dari fungsi $F_{(bcc,0,5)}$ adalah 0.

Sehingga hasil dari fungsi $G_{(bcc,1,3)} = 1$, fungsi $G_{(bcc,3,3)} = 1$ dan fungsi $G_{(bcc,7,2)} = 6$.

BAB III

DESAIN

Pada bab ini akan dibahas tentang desain algoritma untuk menyelesaikan permasalahan klasik SPOJ 9967 *Playing With Words*.

3.1 Desain Umum Sistem

Pada subbab ini akan dijelaskan mengenai gambaran secara umum dari algoritma yang dirancang.

Program akan diawali dengan melakukan *preprocess* lalu dilanjutkan dengan menerima masukan berupa banyak data uji. Untuk setiap data uji berupa sebuah baris yang terdiri dari tiga data masukan yang dipisahkan oleh sebuah spasi, yaitu *string ad1*, *string ad2* dan bilangan bulat X . *String ad1* dan *ad2* adalah *string* hasil enkripsi sesuai dengan deskripsi permasalahan klasik SPOJ 9967 *Playing With Words* dan $X = \text{dist}(\text{ad1}, \text{orig1}) + \text{dist}(\text{ad2}, \text{orig2})$ di mana $\text{dist}(st1, st2)$ adalah total jarak absolut masing-masing karakter *st1* dan *st2* pada posisi yang sama. Setelah menerima masukan, maka masukan tersebut diolah dan hasilnya ditampilkan di layar. Secara garis besar seperti yang terlihat pada Gambar 3.1.

3.2 Desain Fungsi Preprocess

Fungsi *preprocess* merupakan fungsi yang bertujuan agar algoritma yang menyelesaikan permasalahan dapat berjalan dengan benar dan efisien. Pada fungsi ini akan dilakukan perhitungan daftar bit yang bernilai 1 pada setiap bilangan bulat dengan konstanta rentang

```

Main()
1 preprocess()
2 TC = Input()
3 for T = 0 to TC - 1
4     readInput()
5     init()
6     solveProblem()
7     writeOutput()

```

Gambar 3.1 Pseudocode Fungsi Main

```

preprocess()
1 for num = 0 to  $2^{10} - 1$ 
2     for bitPos = 0 to 10 - 1
3         if isBitOn(num, bitPos)
4             setBit(powerNum).push(bitPos)

```

Gambar 3.2 Pseudocode Fungsi Preprocess

bilangan yang telah ditentukan. Konstanta rentang bilangan yang digunakan adalah 0 hingga 2^{10} di mana 10 merupakan panjang maksimal *string ad1* dan *ad2* yang mungkin. Tabel A.1 sampai dengan Tabel A.35 adalah nilai dari himpunan *setBit* setelah fungsi *preprocess* dijalankan. Gambar 3.2 adalah *pseudocode* untuk fungsi *preprocess*.

3.3 Desain Fungsi Init

Fungsi *init* merupakan fungsi yang bertujuan untuk melakukan inisialisasi nilai awal dan perhitungan data-data yang diperlukan

untuk menyelesaikan permasalahan klasik SPOJ 9967 *Playing With Words* untuk setiap kasus uji.

Karena algoritma yang dibangun menggunakan pendekatan paradigma *dynamic programming* yang menggunakan teknik memoisasi, maka algoritma yang dibangun harus melakukan inisialisasi nilai untuk setiap memo yang digunakan. Terdapat dua variabel memo yang digunakan pada algoritma yang dibangun, yaitu $\text{memoF}_{(idx,mask,dist)}$ untuk mencatat hasil perhitungan fungsi $F_{(S,mask,dist)}$ dan $\text{memoG}_{(idx,mask,dist)}$ untuk mencatat hasil perhitungan fungsi $G_{(S,mask,d)}$.

Untuk mempermudah menyelesaikan permasalahan klasik SPOJ 9967 *Playing With Words*, algoritma yang dibangun membutuhkan *string* masukan *ad1* dan *ad2* dalam keadaan yang sudah terurut *ascending* secara alfabetis.

Pada bagian berikutnya adalah perhitungan data-data yang dibutuhkan untuk perhitungan jawaban akhir dari permasalahan klasik SPOJ 9967 *Playing With Words*. Data-data yang diperlukan adalah sebagai berikut:

1. $\text{maxMask}_{(S)}$ yaitu nilai maksimal *mask* untuk *string* *S*. Nilai maksimal *mask* dari *string* *S* adalah $2^{|S|} - 1$.
2. $\text{charFirstPos}_{(S,C)}$ yaitu posisi pertama karakter *C* pada *string* *S*.
3. $\text{charLastPos}_{(S,C)}$ yaitu posisi terakhir karakter *C* pada *string* *S*.

Sebagai contoh, ketika *string* masukan *S* = "inicontoh", maka *string* *S* setelah diurutkan secara alfabetis akan menjadi "chiinnoot". Tabel 3.1 adalah nilai dari $\text{charFirstPos}_{(S,C)}$ dan $\text{charLastPos}_{(S,C)}$. Nilai dari $\text{maxMask}_{(S)}$ adalah 511. Gambar 3.3 adalah *pseudocode* dari fungsi ini.

Tabel 3.1 Hasil $charFirstPos_{(S,C)}$ dan $charLastPos_{(S,C)}$ dengan string $S = "inicontoh"$ setelah fungsi init dijalankan

C	$charFirstPos_{(S,C)}$	$charLastPos_{(S,C)}$
a	\emptyset	\emptyset
b	\emptyset	\emptyset
c	0	0
d	\emptyset	\emptyset
e	\emptyset	\emptyset
f	\emptyset	\emptyset
g	\emptyset	\emptyset
h	1	1
i	2	3
j	\emptyset	\emptyset
k	\emptyset	\emptyset
l	\emptyset	\emptyset
m	\emptyset	\emptyset
n	4	5
o	6	7
p	\emptyset	\emptyset
q	\emptyset	\emptyset
r	\emptyset	\emptyset
s	\emptyset	\emptyset
t	8	8
u	\emptyset	\emptyset
v	\emptyset	\emptyset
w	\emptyset	\emptyset
x	\emptyset	\emptyset
y	\emptyset	\emptyset
z	\emptyset	\emptyset

```

init()
1  memoF = ∅
2  memoG = ∅
3  charFirstPos = ∅
4  charLastPos = ∅
5  maxMask(ad1) = 2|ad1| - 1
6  sort(ad1)
7  for i = 0 to |ad1| - 1
8      charLastPos(ad1,ad1i) = i
9      if charFirstPos(ad1,ad1i) = ∅
10         charFirstPos(ad1,ad1i) = i
11  maxMask(ad2) = 2|ad2| - 1
12  sort(ad2)
13 for i = 0 to |ad2| - 1
14     charLastPos(ad2,ad2i) = i
15     if charFirstPos(ad2,ad2i) = ∅
16         charFirstPos(ad2,ad2i) = i

```

Gambar 3.3 Pseudocode Fungsi Init

3.4 Desain Fungsi Solve

Fungsi solve adalah fungsi yang bertujuan untuk menyelesaikan permasalahan sesuai dengan deskripsi permasalahan untuk setiap masukan yang diberikan. Setelah melalui proses *preprocessing*, membaca masukan dan inisialisasi, masukan akan diolah untuk menghasilkan jawaban dari permasalahan klasik SPOJ 9967 *Playing With Words*. Algoritma fungsi solve yang dibangun akan didasari oleh persamaan-persamaan yang terdapat pada subbab 2.4.

Fungsi solve merupakan fungsi yang mengimplementasi persamaan

```

solve(ad1, ad2, X)
1  ret = 0
2  bound = min(X, 250)
3  for dist = 0 to min(250, X)
4      rem = X - dist
5      if rem > 250
6          continue
7      if rem < 0
8          break
9      ret = ret + F(ad1,maxMaskad1,bound-dist)
10     + G(ad2,maxMaskad2,bound-rem)
10     ret = ret + G(ad1,maxMaskad1,bound-dist)
11     + F(ad2,maxMaskad2,bound-rem)
11  return ret

```

Gambar 3.4 Pseudocode Fungsi Solve

2.4.1. Fungsi solve sendiri akan membutuhkan beberapa fungsi-fungsi lain untuk membantu. Fungsi-fungsi tersebut antara lain fungsi $F_{(S,mask,dist)}$ dan fungsi $G_{(S,mask,dist)}$. Gambar 3.4 adalah pseudocode dari fungsi solve.

3.4.1 Desain Fungsi F

Pada pseudocode pada Gambar 3.4, terdapat perhitungan dengan menggunakan fungsi $F_{(S,mask,dist)}$ pada baris 9 dan 10. Seperti yang telah dijelaskan pada bagian 2.4.1, fungsi $F_{(S,mask,dist)}$ adalah fungsi untuk menghitung jumlah kemungkinan *string orig* dari *string S* tanpa operasi *replace* dengan jarak *dist*. Nilai dari fungsi $F_{(S,mask,dist)}$ adalah hasil penjumlahan seluruh *state* yang berhubungan, yaitu $state F_{(S,mask-2^{idx},dist+|S_{idx}-S_{curIdx}|)}$ di mana *curIdx* adalah jumlah bit tidak menyala pada *mask* dan *idx* adalah

Tabel 3.2 Simulasi fungsi F dengan $S = "kbenh"$, $X = 5$ dan $dist = 5$

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,31,5)}$	$F_{(behkn,30,5)} + F_{(behkn,29,8)} + F_{(behkn,27,11)} + F_{(behkn,23,14)} + F_{(behkn,15,17)}$	1
$F_{(behkn,15,17)}$	<i>base case</i>	0
$F_{(behkn,23,14)}$	<i>base case</i>	0
$F_{(behkn,27,11)}$	<i>base case</i>	0
$F_{(behkn,29,8)}$	<i>base case</i>	0
$F_{(behkn,30,5)}$	$F_{(behkn,28,5)} + F_{(behkn,26,8)} + F_{(behkn,22,11)} + F_{(behkn,14,14)}$	1
$F_{(behkn,14,14)}$	<i>base case</i>	0
$F_{(behkn,22,11)}$	<i>base case</i>	0
$F_{(behkn,26,8)}$	<i>base case</i>	0
$F_{(behkn,28,5)}$	$F_{(behkn,24,5)} + F_{(behkn,20,8)} + F_{(behkn,12,11)}$	1
$F_{(behkn,12,11)}$	<i>base case</i>	0
$F_{(behkn,20,8)}$	<i>base case</i>	0
$F_{(behkn,24,5)}$	$F_{(behkn,16,5)} + F_{(behkn,8,8)}$	1
$F_{(behkn,8,8)}$	<i>base case</i>	0
$F_{(behkn,16,5)}$	$F_{(behkn,0,5)}$	1
$F_{(behkn,0,5)}$	<i>base case</i>	1

$set_bit(mask)_i$ untuk setiap i di mana $0 \leq i \leq NSB_{mask}$ dengan $set_bit(mask)$ adalah Himpunan index bit menyala pada $mask$ dan NSB_{mask} adalah jumlah bit menyala pada $mask$. Algoritma pada fungsi $F_{(S,mask,dist)}$ akan didasari oleh persamaan 2.4.2. Gambar 3.5 adalah pseudocode dari fungsi $F_{(S,mask,dist)}$.

```

F( $S, mask, dist$ )
1 if  $dist > bound \vee (mask = 0 \wedge dist \neq bound)$ 
2     return 0
3 if  $mask = 0 \wedge dist = bound$ 
4     return 1
5 if  $memoF_{(S,mask,dist)} \neq \emptyset$ 
6     return  $memoF_{(S,mask,dist)}$ 
7  $numberOfSetBit = size_{(setBit_{(mask)})}$ 
8  $retVal = 0$ 
9 for  $i = 0$  to  $numberOfSetBit - 1$ 
10    if  $setBit_{(mask)i} \geq length_{(S)}$ 
11        break
12     $ret = ret + F1_{(S,mask,setBit_{(mask)i},dist)}$ 
13 return  $memoF_{(S,mask,dist)} = ret$ 

```

Gambar 3.5 Pseudocode Fungsi F

Karena tidak semua *state* yang terhubung dengan *state* $F_{(S,mask,dist)}$ valid, maka diperlukan sebuah fungsi $F1_{(S,mask,idx,dist)}$ untuk menentukan valid atau tidaknya sebuah *state* $F_{(S,mask-2^{idx},dist+|S_{idx}-S_{curIdx}|)}$ yang terbentuk dari *state* $F_{(S,mask,dist)}$. Perancangan algoritma fungsi $F1_{(S,mask,idx,dist)}$ akan didasari oleh persamaan 2.4.3. Gambar 3.6 adalah pseudocode dari fungsi $F1_{(S,mask,idx,dist)}$ dan Gambar 3.7 adalah pseudocode dari fungsi *duplicate_rule1*($S, mask, idx$). Tabel 3.2 adalah simulasi fungsi F dengan $S = "kbenh"$, $X = 5$ dan $dist = 5$.

```

F1( $S, mask, idx, dist$ )
1    $curIdx = \text{length}_{(S)} - \text{size}_{(\text{setBit}_{(mask)})}$ 
2   if  $idx = \text{length}_{(S)} - 1 \vee \text{duplicateRule1}_{(S, mask, idx)}$ 
3       return  $F_{(S, mask - 2^{idx}, dist + |S_{idx} - S_{curIdx}|)}$ 

```

Gambar 3.6 Pseudocode Fungsi F1

```

duplicate_rule1( $S, mask, idx$ )
1   return  $idx < \text{length}_{(S)} - 1 \wedge (S_{idx} \neq S_{idx+1}) \vee$ 
      ( $S_{idx} = S_{idx+1} \wedge \neg \text{isBitOn}_{(mask, idx+1)}$ )

```

Gambar 3.7 Pseudocode Fungsi duplicate_rule1

3.4.2 Desain Fungsi G

Pada *pseudocode* pada Gambar 3.4, terdapat perhitungan dengan menggunakan fungsi $G_{(S, mask, dist)}$ pada baris 9 dan 10. Seperti yang telah dijelaskan pada bagian 2.4.1, fungsi $G_{(S, mask, dist)}$ merupakan fungsi untuk menghitung jumlah kemungkinan *string orig* dari *string S* dengan sekali operasi *replace* dengan jarak $X - dist$. Nilai dari fungsi $G_{(S, mask, dist)}$ adalah hasil penjumlahan dari seluruh *state* yang berhubungan dengan dengan *state* $G_{(S, mask, dist)}$ yang valid. Gambar 3.8 adalah *pseudocode* dari fungsi $G_{(S, mask, dist)}$. Terdapat tiga kasus *state* yang mungkin, yaitu:

1. *State* $G_{(S, mask - 2^{idx}, dist + |S_{idx} - S_{curIdx}|)}$ dengan kasus ketika mengambil karakter posisi idx pada *string S* sebagai karakter posisi $curIdx$ pada *string orig* tanpa melakukan *replace*. Fungsi $G1_{(S, mask, idx, dist)}$ adalah fungsi yang melakukan

validasi terhadap *state* jenis pertama. Gambar 3.9 adalah *pseudocode* dari fungsi $G1_{(S,mask,idx,dist)}$.

2. *State* $F_{(S,mask-2^{idx},dist+|S_{idx}+1-S_{curIdx}|)}$ dengan kasus ketika mengambil karakter posisi idx pada *string* S sebagai karakter posisi $curIdx$ pada *string* *orig* dengan melakukan *replace* dengan karakter setelahnya secara alfabetis. Fungsi $G2_{(S,mask,idx,dist)}$ adalah fungsi yang melakukan validasi terhadap *state* jenis kedua. Gambar 3.10 adalah *pseudocode* dari fungsi $G2_{(S,mask,idx,dist)}$ dan Gambar 3.12 adalah *pseudocode* dari fungsi *duplicate_rule2*($S, mask, idx$).
3. *State* $F_{(S,mask-2^{idx},dist+|S_{idx}-1-S_{curIdx}|)}$ dengan kasus ketika mengambil karakter posisi idx pada *string* S sebagai karakter posisi $curIdx$ pada *string* *orig* dengan melakukan *replace* dengan karakter sebelumnya secara alfabetis. Fungsi $G3_{(S,mask,idx,dist)}$ adalah fungsi yang melakukan validasi terhadap *state* jenis ketiga. Gambar 3.11 adalah *pseudocode* dari fungsi $G3_{(S,mask,idx,dist)}$ dan Gambar 3.13 adalah *pseudocode* dari fungsi *duplicate_rule3*($S, mask, idx$).

Tabel 3.3 sampai dengan 3.8 adalah simulasi fungsi G dengan $S = "kbenh"$, $X = 5$ dan $dist = 0$.

Tabel 3.3 Simulasi fungsi G dengan $S = "kbenh"$, $X = 5$ dan $dist = 0$
(1)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,31,0)}$	$G_{(behkn,30,0)} + F_{(behkn,30,1)} + F_{(behkn,30,1)} + G_{(behkn,29,3)} + F_{(behkn,29,4)} + F_{(behkn,29,2)} + G_{(behkn,27,6)} + F_{(behkn,27,7)} + F_{(behkn,27,5)} + G_{(behkn,23,9)} + F_{(behkn,23,10)} + F_{(behkn,23,8)} + G_{(behkn,15,12)} + F_{(behkn,15,13)} + F_{(behkn,15,11)}$	8
$F_{(behkn,15,11)}$	base case	0
$F_{(behkn,15,13)}$	base case	0
$G_{(behkn,15,12)}$	base case	0
$F_{(behkn,23,8)}$	base case	0
$F_{(behkn,23,10)}$	base case	0
$G_{(behkn,23,9)}$	base case	0
$F_{(behkn,27,5)}$	$F_{(behkn,26,8)} + F_{(behkn,25,5)} + F_{(behkn,19,11)} + F_{(behkn,11,14)}$	0
$F_{(behkn,11,14)}$	base case	0
$F_{(behkn,19,11)}$	base case	0
$F_{(behkn,25,5)}$	$memoF_{(behkn,25,5)}$	0
$F_{(behkn,26,8)}$	base case	0
$F_{(behkn,27,7)}$	base case	0
$G_{(behkn,27,6)}$	base case	0
$F_{(behkn,29,2)}$	$F_{(behkn,28,5)} + F_{(behkn,25,5)} + F_{(behkn,21,8)} + F_{(behkn,13,11)}$	1
$F_{(behkn,13,11)}$	base case	0
$F_{(behkn,21,8)}$	base case	0
$F_{(behkn,25,5)}$	$memoF_{(behkn,25,5)}$	0
$F_{(behkn,28,5)}$	$memoF_{(behkn,28,5)}$	1
$F_{(behkn,29,4)}$	$F_{(behkn,28,7)} + F_{(behkn,25,7)} + F_{(behkn,21,10)} + F_{(behkn,13,13)}$	0

Tabel 3.4 Simulasi fungsi G dengan $S = "kbenh"$, $X = 5$ dan $dist = 0$
(2)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,13,13)}$	<i>base case</i>	0
$F_{(behkn,21,10)}$	<i>base case</i>	0
$F_{(behkn,25,7)}$	<i>base case</i>	0
$F_{(behkn,28,7)}$	<i>base case</i>	0
$G_{(behkn,29,3)}$	$G_{(behkn,28,6)} + F_{(behkn,28,5)} +$ $F_{(behkn,28,7)} + G_{(behkn,25,6)} +$ $F_{(behkn,25,7)} + F_{(behkn,25,5)} +$ $G_{(behkn,21,9)} + F_{(behkn,21,10)} +$ $F_{(behkn,21,8)} + G_{(behkn,13,12)} +$ $F_{(behkn,13,13)} + F_{(behkn,13,11)}$	1
$F_{(behkn,13,11)}$	<i>base case</i>	0
$F_{(behkn,13,13)}$	<i>base case</i>	0
$G_{(behkn,13,12)}$	<i>base case</i>	0
$F_{(behkn,21,8)}$	<i>base case</i>	0
$F_{(behkn,21,10)}$	<i>base case</i>	0
$G_{(behkn,21,9)}$	<i>base case</i>	0
$F_{(behkn,25,5)}$	$F_{(behkn,24,11)} + F_{(behkn,17,8)} +$ $F_{(behkn,9,11)}$	0
$F_{(behkn,9,11)}$	<i>base case</i>	0
$F_{(behkn,17,8)}$	<i>base case</i>	0
$F_{(behkn,24,11)}$	<i>base case</i>	0
$F_{(behkn,25,7)}$	<i>base case</i>	0
$G_{(behkn,25,6)}$	<i>base case</i>	0
$F_{(behkn,28,7)}$	<i>base case</i>	0
$F_{(behkn,28,5)}$	$F_{(behkn,24,5)} + F_{(behkn,20,8)} +$ $F_{(behkn,12,11)}$	1
$F_{(behkn,12,11)}$	<i>base case</i>	0
$F_{(behkn,20,8)}$	<i>base case</i>	0

Tabel 3.5 Simulasi fungsi G dengan $S = "kbenh"$, $X = 5$ dan $dist = 0$
(3)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,24,5)}$	$memoF_{(behkn,24,5)}$	1
$G_{(behkn,28,6)}$	<i>base case</i>	0
$F_{(behkn,30,1)}$	$memoF_{(behkn,30,1)}$	0
$F_{(behkn,30,1)}$	$F_{(behkn,28,1)} + F_{(behkn,26,4)} + F_{(behkn,22,7)} + F_{(behkn,14,10)}$	0
$F_{(behkn,14,10)}$	<i>base case</i>	0
$F_{(behkn,22,7)}$	<i>base case</i>	0
$F_{(behkn,26,4)}$	$memoF_{(behkn,26,4)}$	0
$F_{(behkn,28,1)}$	$memoF_{(behkn,28,1)}$	0
$G_{(behkn,30,0)}$	$G_{(behkn,28,0)} + F_{(behkn,28,1)} + F_{(behkn,28,1)} + G_{(behkn,26,3)} + F_{(behkn,26,4)} + F_{(behkn,26,2)} + G_{(behkn,22,6)} + F_{(behkn,22,7)} + F_{(behkn,22,5)} + G_{(behkn,14,9)} + F_{(behkn,14,10)} + F_{(behkn,14,8)}$	6
$F_{(behkn,14,8)}$	<i>base case</i>	0
$F_{(behkn,14,10)}$	<i>base case</i>	0
$G_{(behkn,14,9)}$	<i>base case</i>	0
$F_{(behkn,22,5)}$	$F_{(behkn,20,8)} + F_{(behkn,18,5)} + F_{(behkn,6,11)}$	0
$F_{(behkn,6,11)}$	<i>base case</i>	0
$F_{(behkn,18,5)}$	$memoF_{(behkn,18,5)}$	0
$F_{(behkn,20,8)}$	<i>base case</i>	0
$F_{(behkn,22,7)}$	<i>base case</i>	0
$G_{(behkn,22,6)}$	<i>base case</i>	0
$F_{(behkn,26,2)}$	$F_{(behkn,24,5)} + F_{(behkn,18,5)} + F_{(behkn,10,8)}$	1
$F_{(behkn,10,8)}$	<i>base case</i>	0
$F_{(behkn,18,5)}$	$memoF_{(behkn,18,5)}$	0

Tabel 3.6 Simulasi fungsi G dengan $S = "kbenh"$, $X = 5$ dan $dist = 0$
(4)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,24,5)}$	$memoF_{(behkn,24,5)}$	1
$F_{(behkn,26,4)}$	$F_{(behkn,24,7)} + F_{(behkn,18,7)} + F_{(behkn,10,10)}$	0
$F_{(behkn,10,10)}$	<i>base case</i>	0
$F_{(behkn,18,7)}$	<i>base case</i>	0
$F_{(behkn,24,7)}$	<i>base case</i>	0
$G_{(behkn,26,3)}$	$G_{(behkn,24,6)} + F_{(behkn,24,5)} + F_{(behkn,24,7)} + G_{(behkn,18,6)} + F_{(behkn,18,7)} + F_{(behkn,18,5)} + G_{(behkn,10,9)} + F_{(behkn,10,10)} + F_{(behkn,10,8)}$	1
$F_{(behkn,10,8)}$	<i>base case</i>	0
$F_{(behkn,10,10)}$	<i>base case</i>	0
$G_{(behkn,10,9)}$	<i>base case</i>	0
$F_{(behkn,18,5)}$	$F_{(behkn,16,11)} + F_{(behkn,2,8)}$	0
$F_{(behkn,2,8)}$	<i>base case</i>	0
$F_{(behkn,16,11)}$	<i>base case</i>	0
$F_{(behkn,18,7)}$	<i>base case</i>	0
$G_{(behkn,18,6)}$	<i>base case</i>	0
$F_{(behkn,24,7)}$	<i>base case</i>	0
$F_{(behkn,24,5)}$	$F_{(behkn,16,5)} + F_{(behkn,8,8)}$	1
$F_{(behkn,8,8)}$	<i>base case</i>	0
$F_{(behkn,16,5)}$	$memoF_{(behkn,16,5)}$	1
$G_{(behkn,24,6)}$	<i>base case</i>	0
$F_{(behkn,28,1)}$	$memoF_{(behkn,28,1)}$	0
$F_{(behkn,28,1)}$	$F_{(behkn,24,1)} + F_{(behkn,20,4)} + F_{(behkn,12,7)}$	0

Tabel 3.7 Simulasi fungsi G dengan $S = "kbenh"$, $X = 5$ dan $dist = 0$
(5)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,12,7)}$	$base\ case$	0
$F_{(behkn,20,4)}$	$memoF_{(behkn,20,4)}$	0
$F_{(behkn,24,1)}$	$memoF_{(behkn,24,1)}$	0
$G_{(behkn,28,0)}$	$G_{(behkn,24,0)} + F_{(behkn,24,1)} +$ $F_{(behkn,24,1)} + G_{(behkn,20,3)} +$ $F_{(behkn,20,4)} + F_{(behkn,20,2)} +$ $G_{(behkn,12,6)} + F_{(behkn,12,7)} +$ $F_{(behkn,12,5)}$	4
$F_{(behkn,12,5)}$	$F_{(behkn,8,8)} + F_{(behkn,4,5)}$	0
$F_{(behkn,4,5)}$	$memoF_{(behkn,4,5)}$	0
$F_{(behkn,8,8)}$	$base\ case$	0
$F_{(behkn,12,7)}$	$base\ case$	0
$G_{(behkn,12,6)}$	$base\ case$	0
$F_{(behkn,20,2)}$	$F_{(behkn,16,5)} + F_{(behkn,4,5)}$	1
$F_{(behkn,4,5)}$	$memoF_{(behkn,4,5)}$	0
$F_{(behkn,16,5)}$	$memoF_{(behkn,16,5)}$	1
$F_{(behkn,20,4)}$	$F_{(behkn,16,7)} + F_{(behkn,4,7)}$	0
$F_{(behkn,4,7)}$	$base\ case$	0
$F_{(behkn,16,7)}$	$base\ case$	0
$G_{(behkn,20,3)}$	$G_{(behkn,16,6)} + F_{(behkn,16,5)} +$ $F_{(behkn,16,7)} + G_{(behkn,4,6)} +$ $F_{(behkn,4,7)} + F_{(behkn,4,5)}$	1
$F_{(behkn,4,5)}$	$F_{(behkn,0,11)}$	0
$F_{(behkn,0,11)}$	$base\ case$	0
$F_{(behkn,4,7)}$	$base\ case$	0
$G_{(behkn,4,6)}$	$base\ case$	0
$F_{(behkn,16,7)}$	$base\ case$	0
$F_{(behkn,16,5)}$	$F_{(behkn,0,5)}$	1

Tabel 3.8 Simulasi fungsi G dengan $S = "kbenh"$, $X = 5$ dan $dist = 0$
(6)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,0,5)}$	<i>base case</i>	1
$G_{(behkn,16,6)}$	<i>base case</i>	0
$F_{(behkn,24,1)}$	<i>memoF</i> $_{(behkn,24,1)}$	0
$F_{(behkn,24,1)}$	$F_{(behkn,16,1)} + F_{(behkn,8,4)}$	0
$F_{(behkn,8,4)}$	<i>memoF</i> $_{(behkn,8,4)}$	0
$F_{(behkn,16,1)}$	<i>memoF</i> $_{(behkn,16,1)}$	0
$G_{(behkn,24,0)}$	$G_{(behkn,16,0)} + F_{(behkn,16,1)} +$ $F_{(behkn,16,1)} + G_{(behkn,8,3)} +$ $F_{(behkn,8,4)} + F_{(behkn,8,2)}$	2
$F_{(behkn,8,2)}$	$F_{(behkn,0,5)}$	1
$F_{(behkn,0,5)}$	<i>base case</i>	1
$F_{(behkn,8,4)}$	$F_{(behkn,0,7)}$	0
$F_{(behkn,0,7)}$	<i>base case</i>	0
$G_{(behkn,8,3)}$	$G_{(behkn,0,6)} + F_{(behkn,0,5)} +$ $F_{(behkn,0,7)}$	1
$F_{(behkn,0,7)}$	<i>base case</i>	0
$F_{(behkn,0,5)}$	<i>base case</i>	1
$G_{(behkn,0,6)}$	<i>base case</i>	0
$F_{(behkn,16,1)}$	<i>memoF</i> $_{(behkn,16,1)}$	0
$F_{(behkn,16,1)}$	$F_{(behkn,0,1)}$	0
$F_{(behkn,0,1)}$	<i>base case</i>	0
$G_{(behkn,16,0)}$	$G_{(behkn,0,0)} + F_{(behkn,0,1)} +$ $F_{(behkn,0,1)}$	0
$F_{(behkn,0,1)}$	<i>base case</i>	0
$F_{(behkn,0,1)}$	<i>base case</i>	0
$G_{(behkn,0,0)}$	<i>base case</i>	0

```

G( $S, mask, dist$ )
1 if  $dist > bound \vee mask = 0$ 
2   return 0
3 if  $memoG_{(S,mask,dist)} \neq \emptyset$ 
4   return  $memoG_{(S,mask,dist)}$ 
5  $numberOfSetBit = \text{size}_{(setBit}_{(mask))}$ 
6  $retVal = 0$ 
7 for  $i = 0$  to  $numberOfSetBit - 1$ 
8   if  $setBit_{(mask)i} \geq length_{(S)}$ 
9     break
10   $ret = ret + G1_{(S,mask, setBit_{(mask)i}, dist)}$ 
11   $ret = ret + G2_{(S,mask, setBit_{(mask)i}, dist)}$ 
12   $ret = ret + G3_{(S,mask, setBit_{(mask)i}, dist)}$ 
13 return  $memoF_{(S,mask,dist)} = ret$ 

```

Gambar 3.8 Pseudocode Fungsi G

```

G1( $S, mask, idx, dist$ )
1  $curIdx = length_{(S)} - \text{size}_{(setBit}_{(mask))}$ 
2 if  $idx = length_{(S)} - 1 \vee \text{duplicateRule1}_{(S,mask,idx)}$ 
3   return  $G_{(S,mask-2^{idx},dist+|S_{idx}-S_{curIdx}|)}$ 

```

Gambar 3.9 Pseudocode Fungsi G1

```

G2( $S, mask, idx, dist$ )
1    $curIdx = \text{length}_{(S)} - \text{size}_{(\text{setBit}_{(mask)})}$ 
2   if  $idx = \text{length}_{(S)} - 1 \vee (\text{duplicateRule2}_{(S, mask, idx)}) \wedge$ 
       $(\text{duplicateRule1}_{(S, mask, idx)})$ 
3     return  $F_{(S, mask - 2^{idx}, dist + |(S_{idx+1}) - S_{curIdx}|)}$ 

```

Gambar 3.10 Pseudocode Fungsi G2

```

G3( $S, mask, idx, dist$ )
1    $curIdx = \text{length}_{(S)} - \text{size}_{(\text{setBit}_{(mask)})}$ 
2   if  $(idx = \text{length}_{(S)} - 1 \vee \text{duplicateRule1}_{(S, mask, idx)}) \wedge$ 
       $(idx = 0 \vee \text{duplicateRule3}_{(S, mask, idx)})$ 
3     return  $F_{(S, mask - 2^{idx}, dist + |(S_{idx-1}) - S_{curIdx}|)}$ 

```

Gambar 3.11 Pseudocode Fungsi G3

```

duplicate_rule2( $S, mask, idx$ )
1   return  $idx < \text{length}_{(S)} - 1 \wedge (\text{charFirstPos}_{(S, S_{idx+1})}$ 
     $= \emptyset \vee (\text{charFirstPos}_{(S, S_{idx+1})} \neq \emptyset$ 
     $\wedge \neg \text{isBitOn}_{(mask, \text{charFirstPos}_{(S, S_{idx+1})})}))$ 

```

Gambar 3.12 Pseudocode Fungsi `duplicate_rule2`

```
duplicate_rule3( $S, mask, idx$ )
1   return  $idx > 0 \wedge (charLastPos_{(S, S_{idx}-1)}$ 
       $= \emptyset \vee (charLastPos_{(S, S_{idx}-1)} \neq \emptyset$ 
       $\wedge isBitOn_{(mask, charLastPos_{(S, S_{idx}-1)})}))$ 
```

Gambar 3.13 Pseudocode Fungsi duplicate_rule3

Halaman ini sengaja dikosongkan

BAB IV

IMPLEMENTASI

Pada bab ini dijelaskan mengenai implementasi dari desain algoritma penyelesaian permasalahan klasik SPOJ 9967 *Playing With Words*.

4.1 Lingkungan Implementasi

Lingkungan implementasi dalam pembuatan Tugas Akhir ini meliputi perangkat keras dan perangkat lunak yang digunakan untuk melakukan proses pendekatan paradigma *dynamic programming* dan teknik *meet in the middle* untuk permasalahan klasik SPOJ 9967 *Playing With Words* adalah sebagai berikut:

1. Perangkat Keras.
 - Processor Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz.
 - Memori 8 GB.
2. Perangkat Lunak.
 - Sistem operasi Linux Mint 17.1 Rebecca 64 bit.
 - *Text editor* vim
 - *Compiler* g++ versi 4.8.4.

4.2 Rancangan Data

Pada subbab ini dijelaskan mengenai desain data masukan yang diperlukan untuk melakukan proses algoritma, dan data keluaran yang dihasilkan oleh program.

4.2.1 Data Masukan

Data masukan adalah data yang akan diproses oleh program sebagai masukan menggunakan algoritma yang telah dirancang dalam tugas akhir ini.

Data masukan berupa berkas teks yang berisi data dengan format yang telah ditentukan pada deskripsi permasalahan klasik SPOJ 9967 *Playing With Words*. Pada masing-masing berkas data masukan, baris pertama berupa sebuah bilangan bulat yang merepresentasikan jumlah kasus uji yang ada pada berkas tersebut. Untuk setiap kasus uji, masukan berupa sebuah baris masukan yang terdiri dari dua buah *string* *ad1* dan *ad2* diikuti oleh sebuah bilangan bulat *X* yang merepresentasikan $dist(ad1, orig1) + dist(ad2, orig2)$.

4.2.2 Data Keluaran

Data keluaran yang dihasilkan oleh program hanya berupa satu nilai, yaitu jumlah kemungkinan *string* *orig1* dan *orig2* yang mungkin membentuk *string* *ad1* dan *ad2*.

4.3 Implementasi Algoritma

Pada subbab ini akan dijelaskan tentang implementasi proses algoritma secara keseluruhan berdasarkan desain yang telah dijelaskan pada bab III.

4.3.1 Header-Header yang Diperlukan

Implementasi algoritma dengan teknik *meet in the middle* dan *dynamic programming* untuk menyelesaikan permasalahan klasik SPOJ 9967 *Playing With Words* membutuhkan lima buah header yaitu *cstdio*, *algorithm*, *vector*, *string* dan *cstring*, seperti yang terlihat pada Kode Sumber 4.3.1.

```
1 #include <cstdio>
2 #include <algorithm>
3 #include <vector>
4 #include <string>
5 #include <cstring>
```

Kode Sumber 4.3.1 Header yang diperlukan

Header *cstdio* berisi modul untuk menerima masukan dan memberikan keluaran. Header *vector* berisi struktur data yang digunakan untuk menyimpan data himpunan index bit menyalah dari sebuah bilangan bulat. Header *algorithm* berisi modul yang memiliki fungsi-fungsi yang sangat berguna dalam membantu mengimplementasi algortima yang telah dibangun. Contohnya adalah fungsi *max* dan *sort*. Header *string* berisi modul untuk menyimpan data berupa text. Header *cstring* berisi modul yang memiliki fungsi-fungsi untuk melakukan pemrosesan string. Contoh fungsi yang membantu mengimplementasikan algoritma yang dibangun adalah fungsi *memset*.

4.3.2 Variabel Global

Variabel global digunakan untuk memudahkan dalam mengakses data yang digunakan lintas fungsi. Kode sumber implementasi variabel global dapat dilihat pada Kode Sumber 4.3.2.

```

1  using namespace std;
2
3  vector<int> set_bit[(1 << 11)];
4  string S[2];
5  int charLastPos[2][50];
6  int charFirstPos[2][50];
7  int X, bound;
8  int memoF[2][(1 << 10) + 2][250 + 2];
9  int memoG[2][(1 << 10) + 2][250 + 2];
10 int maxMask[2];

```

Kode Sumber 4.3.2 Variabel global

4.3.3 Implementasi Fungsi Main

Fungsi main adalah implementasi algoritma yang dirancang pada Gambar 3.1. Implementasi fungsi main dapat dilihat pada Kode Sumber 4.3.3.

```

1  int main() {
2      preprocess();
3      int t;
4      scanf("%d", &t);
5      for (int tc=1; tc<=t; tc++) {
6          readInput();
7          init();
8          long long ans = solveProblem();
9          writeOutput(tc, ans);
10     }
11     return 0;
12 }

```

Kode Sumber 4.3.3 Fungsi main

4.3.4 Implementasi Fungsi Preprocess

Fungsi preprocess adalah implementasi dari hasil perancangan pada pseudocode pada Gambar 3.2. Implementasi dari fungsi preprocess

dapat dilihat pada Kode Sumber 4.3.4.

```

1 void preprocess() {
2     for (int i = 0; i < (1 << 10); i++) {
3         for (int j = 0; j < 10; j++) {
4             if (isBitOn(i, j))
5                 set_bit[i].
6                     push_back(j);
7         }
8     }
9 }
```

Kode Sumber 4.3.4 Fungsi preprocess

4.3.5 Implementasi Fungsi ReadInput

Fungsi readInput akan membaca masukan dari berkas uji untuk setiap kasus ujinya. Pada awalnya, fungsi akan membaca masukan *string ad1*, lalu dilanjutkan dengan membaca *string ad2* dan diakhiri dengan membaca sebuah bilangan bulat *X*. Implementasi fungsi readInput dapat dilihat pada Kode Sumber 4.3.5.

```

1 void readInput() {
2     char dummySt[30];
3     scanf("%s", dummySt);
4     S[0] = dummySt;
5     scanf("%s", dummySt);
6     S[1] = dummySt;
7     scanf("%d", &X);
8 }
```

Kode Sumber 4.3.5 Fungsi readInput

4.3.6 Implementasi Fungsi Init

Implementasi dari fungsi init dapat dilihat pada Kode Sumber 4.3.6.

```

1 void init() {
2     memset(charLastPos, -1, sizeof
3             charLastPos);
4     memset(charFirstPos, -1, sizeof
5             charFirstPos);
6     memset(memoF, -1, sizeof memoF);
7     memset(memoG, -1, sizeof memoG);
8     for (int idx = 0; idx < 2; idx++) {
9         sort(S[idx].begin(), S[idx].end()
10             );
11        maxMask[idx] = (1 << S[idx].
12                     length()) - 1;
13        for (int i = 0; i < S[idx].length
14             (); i++) {
15            charLastPos[idx][S[idx][i
16                      ] - 'a'] = i;
17            if (charFirstPos[idx][S[
18                        idx][i] - 'a'] ==
19                            -1) {
20                charFirstPos[idx]
21                    [S[idx][i
22                      ] - 'a'] = i
23                    ;
24            }
25        }
26    }
27    bound = min(X, 250);
28 }
```

Kode Sumber 4.3.6 Fungsi init

4.3.7 Implementasi Fungsi Solve

Fungsi solve adalah implementasi dari desain algoritma pada Gambar 3.4 dimana algoritma tersebut adalah hasil perancangan berdasarkan persamaan 2.4.1. Implementasi dari fungsi solve dapat dilihat pada Kode Sumber 4.3.7.

```

1 long long solveProblem() {
2     long long ret = 0;
3     for (int dist = 0; dist <= min(250, X);
4         dist++) {
5         int rem = X - dist;
6         if (rem > 250) continue;
7         if (rem < 0) break;
8         ret += F(0, maxMask[0], bound -
9                 dist) * G(1, maxMask[1],
10                bound - rem);
11        ret += G(0, maxMask[0], bound -
12                dist) * F(1, maxMask[1],
13                bound - rem);
14    }
15    return ret;
16 }
```

Kode Sumber 4.3.7 Fungsi solve

4.3.8 Implementasi Fungsi F

terdapat fungsi F yang telah dirancang algoritmanya pada subbab 3.4.1. Implementasi dari fungsi F dapat dilihat pada Kode Sumber 4.3.8 dan 4.3.9.

```

1 long long F(int idx, int mask, int dist) {
2     if (dist > bound || (mask == 0 && dist
3             != bound)) return 0;
4     if (mask == 0 && dist == bound)
5         return 1;
6     if (memoF[idx][mask][dist] != -1)
7         return memoF[idx][mask][dist];
8     int NSB = set_bit[mask].size();
```

Kode Sumber 4.3.8 Fungsi F (1)

```

1      long long ret = 0;
2      for (int i=0; i<NSB; i++) {
3          if (set_bit[mask][i] >=
4              S[idx].length()) break;
5          ret += F1(idx, mask,
6                     set_bit[mask][i], dist);
7      }
8      return memoF[idx][mask][dist] = ret;
9  }

```

Kode Sumber 4.3.9 Fungsi F (2)

4.3.9 Implementasi Fungsi F1

Fungsi F1 adalah implementasi dari perancangan pada *pseudocode* pada Gambar 3.6 yang dirancang berdasarkan persamaan 2.4.3. Implementasi dari fungsi F1 dapat dilihat pada Kode Sumber 4.3.10.

```

1  long long F1(int idx, int mask, int charIdx, int
2                  dist) {
3      int curIdx = S[idx].length() -
4          __builtin_popcount(mask);
5      if (charIdx == S[idx].length() - 1 ||
6          duplicate_rule1(idx, mask, charIdx)
7      ) {
8          return F(idx, mask - (1 <<
9                  charIdx), dist + abs(S[idx]
10                 [charIdx] - S[idx][curIdx
11                 ]));
12     }
13     return 0;
14 }

```

Kode Sumber 4.3.10 Fungsi F1

4.3.10 Implementasi Fungsi G

Implementasi dari fungsi G dapat dilihat pada Kode Sumber 4.3.11. Algoritma pada fungsi F menggunakan pendekatan *dynamic programming*.

```

1  long long G(int idx, int mask, int dist) {
2      if (dist > bound || mask == 0) return 0;
3      if (memoG[idx][mask][dist] != -1) return
4          memoG[idx][mask][dist];
5      int NSB = set_bit[mask].size();
6      long long ret = 0;
7      for (int i=0; i<NSB; i++) {
8          if (set_bit[mask][i] >= S[idx].
9              length()) break;
10         ret += G1(idx, mask, set_bit[mask
11             ][i], dist);
12         ret += G2(idx, mask, set_bit[mask
13             ][i], dist);
14         ret += G3(idx, mask, set_bit[mask
15             ][i], dist);
16     }
17     return memoG[idx][mask][dist] = ret;
18 }
```

Kode Sumber 4.3.11 Fungsi G

4.3.11 Implementasi Fungsi G1

Fungsi G1 adalah implementasi dari perancangan pada *pseudocode* pada Gambar 3.9 yang dirancang berdasarkan persamaan 2.4.6. Implementasi dari fungsi G1 dapat dilihat pada Kode Sumber 4.3.12.

```

1  long long G1(int idx, int mask, int charIdx, int
2      dist) {
3      int curIdx = S[idx].length() -
4          __builtin_popcount(mask);
5      if (charIdx == S[idx].length() - 1 ||
6          duplicate_rule1(idx, mask,
7              charIdx)) {
8          return G(idx, mask - (1 <<
9                  charIdx), dist + abs(
10                 S[idx][charIdx] -
11                 S[idx][curIdx]));
12     }
13     return 0;
14 }
```

Kode Sumber 4.3.12 Fungsi G1

4.3.12 Implementasi Fungsi G2

Fungsi G2 adalah implementasi dari perancangan pada *pseudocode* pada Gambar 3.10. Implementasi dari fungsi G2 dapat dilihat pada Kode Sumber 4.3.13 dan Kode Sumber 4.3.14.

```

1  long long G2(int idx, int mask, int charIdx, int
2      dist) {
3      int curIdx = S[idx].length() -
4          __builtin_popcount(mask);
5      if (charIdx == S[idx].length() - 1 ||
6          (duplicate_rule2(idx, mask,
7              charIdx) && duplicate_rule1(idx,
8                  mask, charIdx))) {
9          return F(idx, mask - (1 <<
10                 charIdx), dist + abs((
11                 S[idx][charIdx] + 1) -
12                 S[idx][curIdx]));
13 }
```

Kode Sumber 4.3.13 Fungsi G2 (1)

```

5
6         }
7 }
```

Kode Sumber 4.3.14 Fungsi G2 (2)

4.3.13 Implementasi Fungsi G3

Fungsi G3 adalah implementasi dari perancangan pada *pseudocode* pada Gambar 3.11 yang dirancang berdasarkan persamaan 2.4.8. Implementasi dari fungsi G3 dapat dilihat pada Kode Sumber 4.3.15.

```

1 long long G3(int idx, int mask, int charIdx, int
2             dist) {
3     int curIdx = S[idx].length() -
4             __builtin_popcount(mask);
5     if ((charIdx == S[idx].length() - 1 ||
6          duplicate_rule1(idx, mask,
7          charIdx)) && (charIdx == 0 ||
8          duplicate_rule3(idx, mask,
9          charIdx)) ) {
10        return F(idx, mask - (1 <<
11                  charIdx), dist + abs(
12                  S[idx][charIdx] - 1) -
13                  S[idx][curIdx]));
14    }
15    return 0;
16 }
```

Kode Sumber 4.3.15 Fungsi G3

4.3.14 Implementasi Fungsi Duplicate Rule 1

Fungsi `duplicate_rule1` adalah implementasi dari perancangan pada *pseudocode* pada Gambar 3.7 yang dirancang berdasarkan

persamaan 2.4.4. Implementasi dari fungsi `duplicate_rule1` dapat dilihat pada Kode Sumber 4.3.16.

```

1 bool duplicate_rule1(int idx, int mask, int
2           charIdx) {
3     return (charIdx < S[idx].length() - 1
4             && (S[idx][charIdx] != S[idx][
5                 charIdx + 1]
6             || (S[idx][charIdx] == S[idx][
7                 charIdx + 1]
8             && !isBitOn(mask, charIdx + 1))
9             ));
10 }
```

Kode Sumber 4.3.16 Fungsi `duplicate_rule1`

4.3.15 Implementasi Fungsi Duplicate Rule 2

Fungsi `duplicate_rule2` adalah implementasi dari perancangan pada *pseudocode* pada Gambar 3.12 yang dirancang berdasarkan persamaan 2.4.9. Implementasi dari fungsi `duplicate_rule2` dapat dilihat pada Kode Sumber 4.3.17.

```

1 bool duplicate_rule2(int idx, int mask, int
2           charIdx) {
3     return (charIdx < S[idx].length() - 1
4             && (charFirstPos[idx][(S[idx][
5                 charIdx] + 1) - 'a'] == -1
6             || (charFirstPos[idx][(S[idx][
7                 charIdx]) + 1 - 'a'] != -1
8             && !isBitOn(mask, charFirstPos[
9                 idx][S[idx][charIdx] + 1 -
10                'a'])));
11 }
```

Kode Sumber 4.3.17 Fungsi `duplicate_rule2`

4.3.16 Implementasi Fungsi Duplicate Rule 3

Fungsi `duplicate_rule3` adalah implementasi dari perancangan pada *pseudocode* pada Gambar 3.13 yang dirancang berdasarkan persamaan 2.4.10. Implementasi dari fungsi `duplicate_rule1` dapat dilihat pada Kode Sumber 4.3.18.

```
1 bool duplicate_rule3(int idx, int mask, int
2           charIdx) {
3     return (charIdx > 0
4             && (charLastPos[idx][(S[idx][
5               charIdx] - 1) - 'a'] == -1
6             || (charLastPos[idx][(S[idx][
7               charIdx] - 1) - 'a'] != -1
8             && isBitOn(mask, charLastPos[idx]
9                     [(S[idx][charIdx] - 1) -
10                      'a']))));
11 }
```

Kode Sumber 4.3.18 Fungsi `duplicate_rule3`

Halaman ini sengaja dikosongkan

BAB V

UJI COBA DAN EVALUASI

Pada bab ini dijelaskan tentang uji coba dan evaluasi dari implementasi yang telah dilakukan pada tugas akhir ini.

5.1 Lingkungan Uji Coba

Linkungan uji coba yang digunakan adalah salah satu sistem yang digunakan situs penilaian daring SPOJ, yaitu kluster *Cube* dengan spesifikasi sebagai berikut:

1. Perangkat Keras.
 - Processor Intel(R) Pentium G860 CPU @ 3GHz.
 - Memory 1536 MB.
2. Perangkat Lunak.
 - Compiler clang 4.0.

5.2 Uji Coba Kebenaran

Uji coba kebenaran dilakukan dengan mengirimkan kode sumber program ke dalam situs penilaian daring SPOJ dan melakukan hasil uji coba kasus sederhana dengan langkah-langkah sesuai dengan algoritma yang telah dirancang dengan keluaran sistem. Permasalahan yang diselesaikan adalah permasalahan klasik SPOJ 9967 *Playing With Words*. Hasil uji coba dengan waktu terbaik pada situs SPOJ ditunjukkan pada Gambar B.1.

Selain itu, dilakukan pengujian sebanyak 30 kali pada situs penilaian daring SPOJ untuk melihat variasi waktu dan memori yang dibutuhkan program. Hasil uji coba sebanyak 30 kali dapat dilihat pada Gambar B.2, B.3 dan B.4.

Tabel 5.1 Kecepatan maksimal, minimal dan rata-rata dari hasil uji coba pengumpulan 30 kali pada situs pengujian daring SPOJ

Waktu Maksimal	1,02 detik
Waktu Minimal	0,98 detik
Waktu Rata-Rata	1,004 detik
Memori Maksimal	19 MB
Memori Minimal	19 MB
Memori Rata-Rata	19 MB

Dari hasil uji coba pada Gambar B.2, B.3 dan B.4 dapat kita tarik beberapa informasi seperti yang tertera pada Tabel 5.1.

Berdasarkan Tabel 5.1, dari percobaan yang dilakukan, didapatkan waktu eksekusi rata-rata 1,004 detik dan waktu maksimal 1,02 detik. Waktu eksekusi tersebut 6 kali lebih cepat dari batas waktu eksekusi yang tertera pada deskripsi permasalahan, yaitu 6,459 detik.

Selanjutnya akan dilakukan uji coba menggunakan kasus uji yang diberikan pada deskripsi permasalahan klasik SPOJ 9967 *Playing With Words*. Pada deskripsi permasalahan, terdapat dua kasus. Kasus pertama yaitu kasus di mana nilai dari *string ad1 = c*, *string ad2 = n* dan *X = 1*.

Sesuai dengan algoritma yang telah dirancang, berdasarkan persamaan 2.4.1 yang sudah ditransformasikan ke dalam bentuk *pseudocode* pada Gambar 3.1, algoritma akan melakukan iterasi variabel *dist* dari 0 hingga *X*.

Pada awalnya jawaban akhir diinisialisasi dengan nilai 0. Proses penyelesaian diawali dengan iterasi *dist = 0*. pada iterasi *dist = 0*, nilai jawaban akhir akan ditambahkan dengan $F_{(c,1,1)}$, yang merupakan jumlah kombinasi *string orig1* tanpa operasi *replace* dengan jarak 0 terhadap *string ad1*, yang berdasarkan ilustrasi

pada Tabel C.1, bernilai 1 yang dikalikan dengan $G_{(n,1,0)}$, yang merupakan jumlah kombinasi *string orig2* dengan operasi *replace* yang memiliki jarak 1 terhadap *string ad2*, yang berdasarkan ilustrasi pada Tabel C.2, bernilai 2. Lalu nilai jawaban akhir akan ditambahkan dengan $G_{(c,1,1)}$, yang merupakan jumlah kombinasi *string orig1* dengan operasi *replace* dengan jarak 0 terhadap *string ad1*, yang berdasarkan ilustrasi pada Tabel C.3, bernilai 0 yang dikalikan dengan $F_{(n,1,0)}$, yang merupakan jumlah kombinasi *string orig2* tanpa operasi *replace* yang memiliki jarak 1 terhadap *string ad2*, yang berdasarkan ilustrasi pada Tabel C.4, bernilai 0. Sehingga nilai jawaban dari iterasi $dist = 0$ adalah 2 dan nilai jawaban akhir sementara adalah 2.

Berikutnya, pada iterasi $dist = 1$, nilai jawaban akhir akan ditambahkan dengan $F_{(c,1,0)}$, yang merupakan jumlah kombinasi *string orig1* tanpa operasi *replace* dengan jarak 1 terhadap *string ad1*, yang berdasarkan ilustrasi pada Tabel C.5, bernilai 0 yang dikalikan dengan $G_{(n,1,1)}$, yang merupakan jumlah kombinasi *string orig2* dengan operasi *replace* yang memiliki jarak 0 terhadap *string ad2*, yang berdasarkan ilustrasi pada Tabel C.6, bernilai 0. Lalu nilai jawaban akhir akan ditambahkan dengan $G_{(c,1,0)}$, yang merupakan jumlah kombinasi *string orig1* dengan operasi *replace* dengan jarak 1 terhadap *string ad1*, yang berdasarkan ilustrasi pada Tabel C.7, bernilai 2 yang dikalikan dengan $F_{(n,1,1)}$, yang merupakan jumlah kombinasi *string orig2* tanpa operasi *replace* yang memiliki jarak 0 terhadap *string ad2*, yang berdasarkan ilustrasi pada Tabel C.8, bernilai 1. Sehingga nilai jawaban dari iterasi 1 adalah 2 dan nilai jawaban akhir pada kasus di mana *string ad1 = c*, *string ad2 = n* dan *X = 1* adalah 4.

Kasus berikutnya adalah kasus di mana *string ad1 = kbenh*, *string ad2 = kbenh* dan *X = 5*. Proses yang dilakukan sama seperti pada kasus sebelumnya, yaitu dimulai dengan mengiterasi variable *dist* dari 0 hingga *X*.

Pada awalnya nilai jawaban akhir diinisialisasi dengan nilai 0. Pada iterasi $dist = 0$, nilai jawaban akhir akan ditambahkan dengan $F_{(behkn,31,5)}$, yang merupakan jumlah kombinasi *string orig1* tanpa operasi *replace* dengan jarak 0 terhadap *string ad1*, yang berdasarkan ilustrasi pada Tabel D.1, bernilai 1 yang dikalikan dengan $G_{(behkn,31,0)}$, yang merupakan jumlah kombinasi *string orig2* dengan operasi *replace* yang memiliki jarak 5 terhadap *string ad2*, yang berdasarkan ilustrasi pada Tabel D.3 sampai dengan Tabel D.8, bernilai 8. Lalu nilai jawaban akhir akan ditambahkan dengan $G_{(behkn,31,5)}$, yang merupakan jumlah kombinasi *string orig1* dengan operasi *replace* dengan jarak 0 terhadap *string ad1*, yang berdasarkan ilustrasi pada Tabel D.9 sampai dengan Tabel D.11, bernilai 0 yang dikalikan dengan $F_{(behkn,31,0)}$, yang merupakan jumlah kombinasi *string orig2* tanpa operasi *replace* yang memiliki jarak 5 terhadap *string ad2*, yang berdasarkan ilustrasi pada Tabel D.12 sampai dengan Tabel D.13, bernilai 0. Sehingga nilai jawaban dari iterasi 0 adalah 8 dan nilai jawaban akhir hingga pada iterasi 0 adalah 8.

Berikutnya, pada iterasi $dist = 1$, nilai jawaban akhir akan ditambahkan dengan $F_{(behkn,31,4)}$, yang merupakan jumlah kombinasi *string orig1* tanpa operasi *replace* dengan jarak 1 terhadap *string ad1*, yang berdasarkan ilustrasi pada Tabel D.14, bernilai 0 yang dikalikan dengan $G_{(behkn,31,1)}$, yang merupakan jumlah kombinasi *string orig2* dengan operasi *replace* yang memiliki jarak 4 terhadap *string ad2*, yang berdasarkan ilustrasi pada Tabel D.15 sampai dengan Tabel D.19, bernilai 0. Lalu nilai jawaban akhir akan ditambahkan dengan $G_{(behkn,31,4)}$, yang merupakan jumlah kombinasi *string orig1* dengan operasi *replace* dengan jarak 1 terhadap *string ad1*, yang berdasarkan ilustrasi pada Tabel D.20 sampai dengan Tabel D.22, bernilai 10 yang dikalikan dengan $F_{(behkn,31,1)}$, yang merupakan jumlah kombinasi *string orig2* tanpa operasi *replace* yang memiliki jarak 4 terhadap *string ad2*, yang berdasarkan ilustrasi pada Tabel D.23 sampai dengan

Tabel D.23, bernilai 0. Sehingga nilai jawaban dari iterasi 1 adalah 0 dan nilai jawaban akhir hingga pada iterasi 1 adalah 8.

Berikutnya, pada iterasi $dist = 2$, nilai jawaban akhir akan ditambahkan dengan $F_{(behkn,31,3)}$, yang merupakan jumlah kombinasi *string orig1* tanpa operasi *replace* dengan jarak 2 terhadap *string ad1*, yang berdasarkan ilustrasi pada Tabel D.24, bernilai 0 yang dikalikan dengan $G_{(behkn,31,2)}$, yang merupakan jumlah kombinasi *string orig2* dengan operasi *replace* yang memiliki jarak 3 terhadap *string ad2*, yang berdasarkan ilustrasi pada Tabel D.25 sampai dengan Tabel D.29, bernilai 0. Lalu nilai jawaban akhir akan ditambahkan dengan $G_{(behkn,31,3)}$, yang merupakan jumlah kombinasi *string orig1* dengan operasi *replace* dengan jarak 2 terhadap *string ad1*, yang berdasarkan ilustrasi pada Tabel D.30 sampai dengan Tabel D.32, bernilai 0 yang dikalikan dengan $F_{(behkn,31,2)}$, yang merupakan jumlah kombinasi *string orig2* tanpa operasi *replace* yang memiliki jarak 3 terhadap *string ad2*, yang berdasarkan ilustrasi pada Tabel D.33 sampai dengan Tabel D.33, bernilai 0. Sehingga nilai jawaban dari iterasi 2 adalah 0 dan nilai jawaban akhir hingga pada iterasi 2 adalah 8.

Berikutnya, pada iterasi $dist = 3$, nilai jawaban akhir akan ditambahkan dengan $F_{(behkn,31,2)}$, yang merupakan jumlah kombinasi *string orig1* tanpa operasi *replace* dengan jarak 3 terhadap *string ad1*, yang berdasarkan ilustrasi pada Tabel D.34, bernilai 0 yang dikalikan dengan $G_{(behkn,31,3)}$, yang merupakan jumlah kombinasi *string orig2* dengan operasi *replace* yang memiliki jarak 2 terhadap *string ad2*, yang berdasarkan ilustrasi pada Tabel D.35 sampai dengan Tabel D.37, bernilai 0. Lalu nilai jawaban akhir akan ditambahkan dengan $G_{(behkn,31,2)}$, yang merupakan jumlah kombinasi *string orig1* dengan operasi *replace* dengan jarak 3 terhadap *string ad1*, yang berdasarkan ilustrasi pada Tabel D.38 sampai dengan Tabel D.42, bernilai 0 yang dikalikan dengan $F_{(behkn,31,3)}$, yang merupakan jumlah kombinasi *string*

orig2 tanpa operasi *replace* yang memiliki jarak 2 terhadap *string ad2*, yang berdasarkan ilustrasi pada Tabel D.43 sampai dengan Tabel D.43, bernilai 0. Sehingga nilai jawaban dari iterasi 3 adalah 0 dan nilai jawaban akhir hingga pada iterasi 3 adalah 8.

Berikutnya, pada iterasi $dist = 4$, nilai jawaban akhir akan ditambahkan dengan $F_{(behkn,31,1)}$, yang merupakan jumlah kombinasi *string orig1* tanpa operasi *replace* dengan jarak 4 terhadap *string ad1*, yang berdasarkan ilustrasi pada Tabel D.44, bernilai 0 yang dikalikan dengan $G_{(behkn,31,4)}$, yang merupakan jumlah kombinasi *string orig2* dengan operasi *replace* yang memiliki jarak 1 terhadap *string ad2*, yang berdasarkan ilustrasi pada Tabel D.45 sampai dengan Tabel D.47, bernilai 10. Lalu nilai jawaban akhir akan ditambahkan dengan $G_{(behkn,31,1)}$, yang merupakan jumlah kombinasi *string orig1* dengan operasi *replace* dengan jarak 4 terhadap *string ad1*, yang berdasarkan ilustrasi pada Tabel D.48 sampai dengan Tabel D.52, bernilai 0 yang dikalikan dengan $F_{(behkn,31,4)}$, yang merupakan jumlah kombinasi *string orig2* tanpa operasi *replace* yang memiliki jarak 1 terhadap *string ad2*, yang berdasarkan ilustrasi pada Tabel D.53 sampai dengan Tabel D.53, bernilai 0. Sehingga nilai jawaban dari iterasi 4 adalah 0 dan nilai jawaban akhir hingga pada iterasi 4 adalah 8.

Berikutnya, pada iterasi $dist = 5$, nilai jawaban akhir akan ditambahkan dengan $F_{(behkn,31,0)}$, yang merupakan jumlah kombinasi *string orig1* tanpa operasi *replace* dengan jarak 5 terhadap *string ad1*, yang berdasarkan ilustrasi pada Tabel D.54, bernilai 0 yang dikalikan dengan $G_{(behkn,31,5)}$, yang merupakan jumlah kombinasi *string orig2* dengan operasi *replace* yang memiliki jarak 0 terhadap *string ad2*, yang berdasarkan ilustrasi pada Tabel D.55 sampai dengan Tabel D.57, bernilai 0. Lalu nilai jawaban akhir akan ditambahkan dengan $G_{(behkn,31,0)}$, yang merupakan jumlah kombinasi *string orig1* dengan operasi *replace* dengan jarak 5 terhadap *string ad1*, yang berdasarkan ilustrasi pada

Tabel D.58 sampai dengan Tabel D.62, bernilai 8 yang dikalikan dengan $F_{(behkn,31,5)}$, yang merupakan jumlah kombinasi *string orig2* tanpa operasi *replace* yang memiliki jarak 0 terhadap *string ad2*, yang berdasarkan ilustrasi pada Tabel D.63 sampai dengan Tabel D.63, bernilai 1. Sehingga nilai jawaban dari iterasi 5 adalah 8 dan nilai jawaban akhir untuk kasus di mana *string ad1 = kbenh*, *string ad2 = kbenh* dan $X = 5$ adalah 16.

5.3 Analisa Kompleksitas Waktu

Pada *pseudocode* pada Gambar 3.1, terdapat fungsi preprocess. Kompleksitas waktu dari fungsi preprocess adalah $\mathcal{O}(2^{|S|} * |S|)$ di mana *S* adalah *string* masukan.

Berikutnya untuk setiap kasus uji terdapat empat fungsi utama. Dengan menggunakan *master theorem*, fungsi readInput memiliki kompleksitas $\mathcal{O}(1)$. Berdasarkan *pseudocode* fungsi init() pada Gambar 3.3, fungsi init dapat dipecah menjadi dua bagian utama, yaitu inisialisasi *memoF* dan *memoG* dan inisialisasi *charFirstPos* dan *charLastPos*. Inisialisasi *memoF* dan *memoG* masing-masing memiliki kompleksitas waktu $\mathcal{O}(2 * 2^{|S|} * MAX_DIST)$ di mana *MAX_DIST* adalah jarak maksimum antar *string* yang mungkin. Sedangkan inisialisasi *charFirstPos* dan *charLastPos* memiliki kompleksitas waktu $\mathcal{O}(|S| \log |S| + |S|)$ atau dapat disederhanakan menjadi $\mathcal{O}(|S| \log |S|)$. Sehingga fungsi ini memiliki kompleksitas $\mathcal{O}(2 * 2^{|S|} * MAX_DIST + |S| \log |S|)$ dan dapat disederhanakan menjadi $\mathcal{O}(2^{|S|} * MAX_DIST)$.

Fungsi berikutnya adalah fungsi solve. Pada fungsi solve terdapat sebuah iterasi sebanyak $\min(MAX_DIST, X)$ di mana pada kasus terburuk, $\min(MAX_DIST, X)$ bisa mencapai *MAX_DIST*. Di dalam iterasi tersebut, fungsi solve memanggil fungsi F dan fungsi G masing-masing sebanyak dua kali yang memiliki kompleksitas waktu $\mathcal{O}(2 * 2^{|S|} * MAX_DIST)$ atau

dapat disederhanakan menjadi $\mathcal{O}(2^{|S|} * MAX_DIST)$. Sehingga kompleksitas dari fungsi solve secara keseluruhan adalah $\mathcal{O}(2^{|S|} * MAX_DIST^2)$.

Terakhir adalah fungsi writeOutput. Kompleksitas waktu dari fungsi writeOutput adalah $\mathcal{O}(1)$. Sehingga secara keseluruhan, kompleksitas waktu dari algoritma yang telah dirancang pada Tugas Akhir ini adalah $\mathcal{O}(2^{|S|} * MAX_DIST^2 * T)$.

Pada umumnya, eksekusi program pada situs penilaian daring SPOJ adalah 1 detik untuk setiap 100.000.000 proses. Pada kasus terburuk, yaitu ketika $|S| = 10$, $MAX_DIST = 250$ dan $T = 10$, eksekusi program dengan kompleksitas waktu $\mathcal{O}(2^{|S|} * MAX_DIST^2 * T)$ akan melakukan 640.000.000 proses dimana jika dengan menggunakan standar berupa 100.000.000 proses per detik, program akan membutuhkan waktu eksekusi sebesar 6,4 detik. Sehingga Algoritma dengan kompleksitas waktu $\mathcal{O}(2^{|S|} * MAX_DIST^2 * T)$ cukup untuk menyelesaikan permasalahan klasik SPOJ 9967 *Playing With Words*.

BAB VI

KESIMPULAN

Pada bab ini dijelaskan mengenai kesimpulan dari hasil uji coba yang telah dilakukan.

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan terhadap perancangan dan implementasi algoritma untuk menyelesaikan permasalahan klasik SPOJ 9967 *Playing With Words* dapat diambil kesimpulan sebagai berikut:

1. Implementasi algoritma dengan menggunakan pendekatan *dynamic programming* dan teknik *meet in the middle* dapat menyelesaikan permasalahan permasalahan klasik SPOJ 9967 *Playing With Words* dengan benar.
2. Kompleksitas waktu sebesar $\mathcal{O}(2^{|S|} * MAX_DIST^2 * T)$ dapat menyelesaikan permasalahan klasik SPOJ 9967 *Playing With Words*.
3. Waktu yang dibutuhkan program untuk menyelesaikan permasalahan klasik SPOJ 9967 *Playing With Words* minimum 0.98 detik, maksimum 1.02 detik dan rata-rata 1.004 detik. Memori yang dibutuhkan adalah sebesar 19 MB.

6.2 Saran

Pada Tugas Akhir kali ini tentunya terdapat kekurangan serta nilai-nilai yang dapat penulis ambil. Berikut adalah saran-saran yang dapat diambil melalui Tugas Akhir ini:

1. Paradigma *dynamic programming* adalah pendekatan yang

sesuai untuk menyelesaikan permasalahan yang memiliki submasalah yang bersifat tumpang tindih dengan submasalah lainnya.

2. Teknik *meet in the middle* adalah teknik yang sesuai untuk menyelesaikan permasalahan apabila permasalahan tersebut dapat dibagi menjadi dua atau lebih submasalah yang tidak memiliki ketergantungan satu sama lain.

DAFTAR PUSTAKA

- [1] **Introduction To Java - MFC 158 G.** [Online]. Available: <http://www.acsu.buffalo.edu/fineberg/mfc158/week10lecture.htm>. [Accessed 24-May-2017].
- [2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, **Introduction To Algorithm**, 2nd ed. Cambridge, Massachusetts London, England: The MIT Press, 2001.
- [3] S. Halim and F. Halim, **Competitive Programming** 3. Singapore, 2013.
- [4] E. Elmaghiraby, **Journal of Mathematical Analysis and Applications**, vol. 29, no. 3, pp. 523–557, Mar. 1970.

Halaman ini sengaja dikosongkan

LAMPIRAN A

TABEL HIMPUNAN *SETBIT* SETELAH FUNGSI PREPROCESS DIJALANKAN

Tabel A.1 Tabel himpunan setBit setelah fungsi preprocess dijalankan (1)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
0	-	-	-	-	-	-	-	-	-	-	-
1	0	-	-	-	-	-	-	-	-	-	-
2	1	-	-	-	-	-	-	-	-	-	-
3	0	1	-	-	-	-	-	-	-	-	-
4	2	-	-	-	-	-	-	-	-	-	-
5	0	2	-	-	-	-	-	-	-	-	-
6	1	2	-	-	-	-	-	-	-	-	-
7	0	1	2	-	-	-	-	-	-	-	-
8	3	-	-	-	-	-	-	-	-	-	-
9	0	3	-	-	-	-	-	-	-	-	-
10	1	3	-	-	-	-	-	-	-	-	-
11	0	1	3	-	-	-	-	-	-	-	-
12	2	3	-	-	-	-	-	-	-	-	-
13	0	2	3	-	-	-	-	-	-	-	-
14	1	2	3	-	-	-	-	-	-	-	-
15	0	1	2	3	-	-	-	-	-	-	-
16	4	-	-	-	-	-	-	-	-	-	-
17	0	4	-	-	-	-	-	-	-	-	-
18	1	4	-	-	-	-	-	-	-	-	-
19	0	1	4	-	-	-	-	-	-	-	-
20	2	4	-	-	-	-	-	-	-	-	-

Tabel A.2 Tabel himpunan setBit setelah fungsi preprocess dijalankan (2)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
21		0	2	4	-	-	-	-	-	-	-
22		1	2	4	-	-	-	-	-	-	-
23		0	1	2	4	-	-	-	-	-	-
24		3	4	-	-	-	-	-	-	-	-
25		0	3	4	-	-	-	-	-	-	-
26		1	3	4	-	-	-	-	-	-	-
27		0	1	3	4	-	-	-	-	-	-
28		2	3	4	-	-	-	-	-	-	-
29		0	2	3	4	-	-	-	-	-	-
30		1	2	3	4	-	-	-	-	-	-
31		0	1	2	3	4	-	-	-	-	-
32		5	-	-	-	-	-	-	-	-	-
33		0	5	-	-	-	-	-	-	-	-
34		1	5	-	-	-	-	-	-	-	-
35		0	1	5	-	-	-	-	-	-	-
36		2	5	-	-	-	-	-	-	-	-
37		0	2	5	-	-	-	-	-	-	-
38		1	2	5	-	-	-	-	-	-	-
39		0	1	2	5	-	-	-	-	-	-
40		3	5	-	-	-	-	-	-	-	-
41		0	3	5	-	-	-	-	-	-	-
42		1	3	5	-	-	-	-	-	-	-
43		0	1	3	5	-	-	-	-	-	-
44		2	3	5	-	-	-	-	-	-	-
45		0	2	3	5	-	-	-	-	-	-
46		1	2	3	5	-	-	-	-	-	-
47		0	1	2	3	5	-	-	-	-	-
48		4	5	-	-	-	-	-	-	-	-
49		0	4	5	-	-	-	-	-	-	-
50		1	4	5	-	-	-	-	-	-	-
51		0	1	4	5	-	-	-	-	-	-

Tabel A.3 Tabel himpunan setBit setelah fungsi preprocess dijalankan (3)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
52		2	4	5	-	-	-	-	-	-	-
53		0	2	4	5	-	-	-	-	-	-
54		1	2	4	5	-	-	-	-	-	-
55		0	1	2	4	5	-	-	-	-	-
56		3	4	5	-	-	-	-	-	-	-
57		0	3	4	5	-	-	-	-	-	-
58		1	3	4	5	-	-	-	-	-	-
59		0	1	3	4	5	-	-	-	-	-
60		2	3	4	5	-	-	-	-	-	-
61		0	2	3	4	5	-	-	-	-	-
62		1	2	3	4	5	-	-	-	-	-
63		0	1	2	3	4	5	-	-	-	-
64		6	-	-	-	-	-	-	-	-	-
65		0	6	-	-	-	-	-	-	-	-
66		1	6	-	-	-	-	-	-	-	-
67		0	1	6	-	-	-	-	-	-	-
68		2	6	-	-	-	-	-	-	-	-
69		0	2	6	-	-	-	-	-	-	-
70		1	2	6	-	-	-	-	-	-	-
71		0	1	2	6	-	-	-	-	-	-
72		3	6	-	-	-	-	-	-	-	-
73		0	3	6	-	-	-	-	-	-	-
74		1	3	6	-	-	-	-	-	-	-
75		0	1	3	6	-	-	-	-	-	-
76		2	3	6	-	-	-	-	-	-	-
77		0	2	3	6	-	-	-	-	-	-
78		1	2	3	6	-	-	-	-	-	-
79		0	1	2	3	6	-	-	-	-	-
80		4	6	-	-	-	-	-	-	-	-
81		0	4	6	-	-	-	-	-	-	-
82		1	4	6	-	-	-	-	-	-	-

Tabel A.4 Tabel himpunan setBit setelah fungsi preprocess dijalankan (4)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
83		0	1	4	6	-	-	-	-	-	-
84		2	4	6	-	-	-	-	-	-	-
85		0	2	4	6	-	-	-	-	-	-
86		1	2	4	6	-	-	-	-	-	-
87		0	1	2	4	6	-	-	-	-	-
88		3	4	6	-	-	-	-	-	-	-
89		0	3	4	6	-	-	-	-	-	-
90		1	3	4	6	-	-	-	-	-	-
91		0	1	3	4	6	-	-	-	-	-
92		2	3	4	6	-	-	-	-	-	-
93		0	2	3	4	6	-	-	-	-	-
94		1	2	3	4	6	-	-	-	-	-
95		0	1	2	3	4	6	-	-	-	-
96		5	6	-	-	-	-	-	-	-	-
97		0	5	6	-	-	-	-	-	-	-
98		1	5	6	-	-	-	-	-	-	-
99		0	1	5	6	-	-	-	-	-	-
100		2	5	6	-	-	-	-	-	-	-
101		0	2	5	6	-	-	-	-	-	-
102		1	2	5	6	-	-	-	-	-	-
103		0	1	2	5	6	-	-	-	-	-
104		3	5	6	-	-	-	-	-	-	-
105		0	3	5	6	-	-	-	-	-	-
106		1	3	5	6	-	-	-	-	-	-
107		0	1	3	5	6	-	-	-	-	-
108		2	3	5	6	-	-	-	-	-	-
109		0	2	3	5	6	-	-	-	-	-
110		1	2	3	5	6	-	-	-	-	-
111		0	1	2	3	5	6	-	-	-	-
112		4	5	6	-	-	-	-	-	-	-

Tabel A.5 Tabel himpunan setBit setelah fungsi preprocess dijalankan (5)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
113	0	4	5	6	-	-	-	-	-	-	-
114	1	4	5	6	-	-	-	-	-	-	-
115	0	1	4	5	6	-	-	-	-	-	-
116	2	4	5	6	-	-	-	-	-	-	-
117	0	2	4	5	6	-	-	-	-	-	-
118	1	2	4	5	6	-	-	-	-	-	-
119	0	1	2	4	5	6	-	-	-	-	-
120	3	4	5	6	-	-	-	-	-	-	-
121	0	3	4	5	6	-	-	-	-	-	-
122	1	3	4	5	6	-	-	-	-	-	-
123	0	1	3	4	5	6	-	-	-	-	-
124	2	3	4	5	6	-	-	-	-	-	-
125	0	2	3	4	5	6	-	-	-	-	-
126	1	2	3	4	5	6	-	-	-	-	-
127	0	1	2	3	4	5	6	-	-	-	-
128	7	-	-	-	-	-	-	-	-	-	-
129	0	7	-	-	-	-	-	-	-	-	-
130	1	7	-	-	-	-	-	-	-	-	-
131	0	1	7	-	-	-	-	-	-	-	-
132	2	7	-	-	-	-	-	-	-	-	-
133	0	2	7	-	-	-	-	-	-	-	-
134	1	2	7	-	-	-	-	-	-	-	-
135	0	1	2	7	-	-	-	-	-	-	-
136	3	7	-	-	-	-	-	-	-	-	-
137	0	3	7	-	-	-	-	-	-	-	-
138	1	3	7	-	-	-	-	-	-	-	-
139	0	1	3	7	-	-	-	-	-	-	-
140	2	3	7	-	-	-	-	-	-	-	-
141	0	2	3	7	-	-	-	-	-	-	-
142	1	2	3	7	-	-	-	-	-	-	-

Tabel A.6 Tabel himpunan setBit setelah fungsi preprocess dijalankan (6)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
143		0	1	2	3	7	-	-	-	-	-
144		4	7	-	-	-	-	-	-	-	-
145		0	4	7	-	-	-	-	-	-	-
146		1	4	7	-	-	-	-	-	-	-
147		0	1	4	7	-	-	-	-	-	-
148		2	4	7	-	-	-	-	-	-	-
149		0	2	4	7	-	-	-	-	-	-
150		1	2	4	7	-	-	-	-	-	-
151		0	1	2	4	7	-	-	-	-	-
152		3	4	7	-	-	-	-	-	-	-
153		0	3	4	7	-	-	-	-	-	-
154		1	3	4	7	-	-	-	-	-	-
155		0	1	3	4	7	-	-	-	-	-
156		2	3	4	7	-	-	-	-	-	-
157		0	2	3	4	7	-	-	-	-	-
158		1	2	3	4	7	-	-	-	-	-
159		0	1	2	3	4	7	-	-	-	-
160		5	7	-	-	-	-	-	-	-	-
161		0	5	7	-	-	-	-	-	-	-
162		1	5	7	-	-	-	-	-	-	-
163		0	1	5	7	-	-	-	-	-	-
164		2	5	7	-	-	-	-	-	-	-
165		0	2	5	7	-	-	-	-	-	-
166		1	2	5	7	-	-	-	-	-	-
167		0	1	2	5	7	-	-	-	-	-
168		3	5	7	-	-	-	-	-	-	-
169		0	3	5	7	-	-	-	-	-	-
170		1	3	5	7	-	-	-	-	-	-
171		0	1	3	5	7	-	-	-	-	-
172		2	3	5	7	-	-	-	-	-	-

Tabel A.7 Tabel himpunan setBit setelah fungsi preprocess dijalankan (7)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
173	0	2	3	5	7	-	-	-	-	-	-
174	1	2	3	5	7	-	-	-	-	-	-
175	0	1	2	3	5	7	-	-	-	-	-
176	4	5	7	-	-	-	-	-	-	-	-
177	0	4	5	7	-	-	-	-	-	-	-
178	1	4	5	7	-	-	-	-	-	-	-
179	0	1	4	5	7	-	-	-	-	-	-
180	2	4	5	7	-	-	-	-	-	-	-
181	0	2	4	5	7	-	-	-	-	-	-
182	1	2	4	5	7	-	-	-	-	-	-
183	0	1	2	4	5	7	-	-	-	-	-
184	3	4	5	7	-	-	-	-	-	-	-
185	0	3	4	5	7	-	-	-	-	-	-
186	1	3	4	5	7	-	-	-	-	-	-
187	0	1	3	4	5	7	-	-	-	-	-
188	2	3	4	5	7	-	-	-	-	-	-
189	0	2	3	4	5	7	-	-	-	-	-
190	1	2	3	4	5	7	-	-	-	-	-
191	0	1	2	3	4	5	7	-	-	-	-
192	6	7	-	-	-	-	-	-	-	-	-
193	0	6	7	-	-	-	-	-	-	-	-
194	1	6	7	-	-	-	-	-	-	-	-
195	0	1	6	7	-	-	-	-	-	-	-
196	2	6	7	-	-	-	-	-	-	-	-
197	0	2	6	7	-	-	-	-	-	-	-
198	1	2	6	7	-	-	-	-	-	-	-
199	0	1	2	6	7	-	-	-	-	-	-
200	3	6	7	-	-	-	-	-	-	-	-
201	0	3	6	7	-	-	-	-	-	-	-
202	1	3	6	7	-	-	-	-	-	-	-

Tabel A.8 Tabel himpunan setBit setelah fungsi preprocess dijalankan (8)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
203		0	1	3	6	7	-	-	-	-	-
204		2	3	6	7	-	-	-	-	-	-
205		0	2	3	6	7	-	-	-	-	-
206		1	2	3	6	7	-	-	-	-	-
207		0	1	2	3	6	7	-	-	-	-
208		4	6	7	-	-	-	-	-	-	-
209		0	4	6	7	-	-	-	-	-	-
210		1	4	6	7	-	-	-	-	-	-
211		0	1	4	6	7	-	-	-	-	-
212		2	4	6	7	-	-	-	-	-	-
213		0	2	4	6	7	-	-	-	-	-
214		1	2	4	6	7	-	-	-	-	-
215		0	1	2	4	6	7	-	-	-	-
216		3	4	6	7	-	-	-	-	-	-
217		0	3	4	6	7	-	-	-	-	-
218		1	3	4	6	7	-	-	-	-	-
219		0	1	3	4	6	7	-	-	-	-
220		2	3	4	6	7	-	-	-	-	-
221		0	2	3	4	6	7	-	-	-	-
222		1	2	3	4	6	7	-	-	-	-
223		0	1	2	3	4	6	7	-	-	-
224		5	6	7	-	-	-	-	-	-	-
225		0	5	6	7	-	-	-	-	-	-
226		1	5	6	7	-	-	-	-	-	-
227		0	1	5	6	7	-	-	-	-	-
228		2	5	6	7	-	-	-	-	-	-
229		0	2	5	6	7	-	-	-	-	-
230		1	2	5	6	7	-	-	-	-	-
231		0	1	2	5	6	7	-	-	-	-
232		3	5	6	7	-	-	-	-	-	-

Tabel A.9 Tabel himpunan setBit setelah fungsi preprocess dijalankan (9)

Tabel A.10 Tabel himpunan setBit setelah fungsi preprocess dijalankan (10)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
263		0	1	2	8	-	-	-	-	-	-
264		3	8	-	-	-	-	-	-	-	-
265		0	3	8	-	-	-	-	-	-	-
266		1	3	8	-	-	-	-	-	-	-
267		0	1	3	8	-	-	-	-	-	-
268		2	3	8	-	-	-	-	-	-	-
269		0	2	3	8	-	-	-	-	-	-
270		1	2	3	8	-	-	-	-	-	-
271		0	1	2	3	8	-	-	-	-	-
272		4	8	-	-	-	-	-	-	-	-
273		0	4	8	-	-	-	-	-	-	-
274		1	4	8	-	-	-	-	-	-	-
275		0	1	4	8	-	-	-	-	-	-
276		2	4	8	-	-	-	-	-	-	-
277		0	2	4	8	-	-	-	-	-	-
278		1	2	4	8	-	-	-	-	-	-
279		0	1	2	4	8	-	-	-	-	-
280		3	4	8	-	-	-	-	-	-	-
281		0	3	4	8	-	-	-	-	-	-
282		1	3	4	8	-	-	-	-	-	-
283		0	1	3	4	8	-	-	-	-	-
284		2	3	4	8	-	-	-	-	-	-
285		0	2	3	4	8	-	-	-	-	-
286		1	2	3	4	8	-	-	-	-	-
287		0	1	2	3	4	8	-	-	-	-
288		5	8	-	-	-	-	-	-	-	-
289		0	5	8	-	-	-	-	-	-	-
290		1	5	8	-	-	-	-	-	-	-
291		0	1	5	8	-	-	-	-	-	-
292		2	5	8	-	-	-	-	-	-	-

Tabel A.11 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(11)

<i>index</i> <i>Num</i>	0	1	2	3	4	5	6	7	8	9
293	0	2	5	8	-	-	-	-	-	-
294	1	2	5	8	-	-	-	-	-	-
295	0	1	2	5	8	-	-	-	-	-
296	3	5	8	-	-	-	-	-	-	-
297	0	3	5	8	-	-	-	-	-	-
298	1	3	5	8	-	-	-	-	-	-
299	0	1	3	5	8	-	-	-	-	-
300	2	3	5	8	-	-	-	-	-	-
301	0	2	3	5	8	-	-	-	-	-
302	1	2	3	5	8	-	-	-	-	-
303	0	1	2	3	5	8	-	-	-	-
304	4	5	8	-	-	-	-	-	-	-
305	0	4	5	8	-	-	-	-	-	-
306	1	4	5	8	-	-	-	-	-	-
307	0	1	4	5	8	-	-	-	-	-
308	2	4	5	8	-	-	-	-	-	-
309	0	2	4	5	8	-	-	-	-	-
310	1	2	4	5	8	-	-	-	-	-
311	0	1	2	4	5	8	-	-	-	-
312	3	4	5	8	-	-	-	-	-	-
313	0	3	4	5	8	-	-	-	-	-
314	1	3	4	5	8	-	-	-	-	-
315	0	1	3	4	5	8	-	-	-	-
316	2	3	4	5	8	-	-	-	-	-
317	0	2	3	4	5	8	-	-	-	-
318	1	2	3	4	5	8	-	-	-	-
319	0	1	2	3	4	5	8	-	-	-
320	6	8	-	-	-	-	-	-	-	-
321	0	6	8	-	-	-	-	-	-	-
322	1	6	8	-	-	-	-	-	-	-

Tabel A.12 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(12)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
323		0	1	6	8	-	-	-	-	-	-
324		2	6	8	-	-	-	-	-	-	-
325		0	2	6	8	-	-	-	-	-	-
326		1	2	6	8	-	-	-	-	-	-
327		0	1	2	6	8	-	-	-	-	-
328		3	6	8	-	-	-	-	-	-	-
329		0	3	6	8	-	-	-	-	-	-
330		1	3	6	8	-	-	-	-	-	-
331		0	1	3	6	8	-	-	-	-	-
332		2	3	6	8	-	-	-	-	-	-
333		0	2	3	6	8	-	-	-	-	-
334		1	2	3	6	8	-	-	-	-	-
335		0	1	2	3	6	8	-	-	-	-
336		4	6	8	-	-	-	-	-	-	-
337		0	4	6	8	-	-	-	-	-	-
338		1	4	6	8	-	-	-	-	-	-
339		0	1	4	6	8	-	-	-	-	-
340		2	4	6	8	-	-	-	-	-	-
341		0	2	4	6	8	-	-	-	-	-
342		1	2	4	6	8	-	-	-	-	-
343		0	1	2	4	6	8	-	-	-	-
344		3	4	6	8	-	-	-	-	-	-
345		0	3	4	6	8	-	-	-	-	-
346		1	3	4	6	8	-	-	-	-	-
347		0	1	3	4	6	8	-	-	-	-
348		2	3	4	6	8	-	-	-	-	-
349		0	2	3	4	6	8	-	-	-	-
350		1	2	3	4	6	8	-	-	-	-
351		0	1	2	3	4	6	8	-	-	-
352		5	6	8	-	-	-	-	-	-	-

Tabel A.13 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(13)

<i>index</i> <i>Num</i>	0	1	2	3	4	5	6	7	8	9
353	0	5	6	8	-	-	-	-	-	-
354	1	5	6	8	-	-	-	-	-	-
355	0	1	5	6	8	-	-	-	-	-
356	2	5	6	8	-	-	-	-	-	-
357	0	2	5	6	8	-	-	-	-	-
358	1	2	5	6	8	-	-	-	-	-
359	0	1	2	5	6	8	-	-	-	-
360	3	5	6	8	-	-	-	-	-	-
361	0	3	5	6	8	-	-	-	-	-
362	1	3	5	6	8	-	-	-	-	-
363	0	1	3	5	6	8	-	-	-	-
364	2	3	5	6	8	-	-	-	-	-
365	0	2	3	5	6	8	-	-	-	-
366	1	2	3	5	6	8	-	-	-	-
367	0	1	2	3	5	6	8	-	-	-
368	4	5	6	8	-	-	-	-	-	-
369	0	4	5	6	8	-	-	-	-	-
370	1	4	5	6	8	-	-	-	-	-
371	0	1	4	5	6	8	-	-	-	-
372	2	4	5	6	8	-	-	-	-	-
373	0	2	4	5	6	8	-	-	-	-
374	1	2	4	5	6	8	-	-	-	-
375	0	1	2	4	5	6	8	-	-	-
376	3	4	5	6	8	-	-	-	-	-
377	0	3	4	5	6	8	-	-	-	-
378	1	3	4	5	6	8	-	-	-	-
379	0	1	3	4	5	6	8	-	-	-
380	2	3	4	5	6	8	-	-	-	-
381	0	2	3	4	5	6	8	-	-	-
382	1	2	3	4	5	6	8	-	-	-

Tabel A.14 Tabel himpunan setBit setelah fungsi preprocess dijalankan
 (14)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
383		0	1	2	3	4	5	6	8	-	-
384		7	8	-	-	-	-	-	-	-	-
385		0	7	8	-	-	-	-	-	-	-
386		1	7	8	-	-	-	-	-	-	-
387		0	1	7	8	-	-	-	-	-	-
388		2	7	8	-	-	-	-	-	-	-
389		0	2	7	8	-	-	-	-	-	-
390		1	2	7	8	-	-	-	-	-	-
391		0	1	2	7	8	-	-	-	-	-
392		3	7	8	-	-	-	-	-	-	-
393		0	3	7	8	-	-	-	-	-	-
394		1	3	7	8	-	-	-	-	-	-
395		0	1	3	7	8	-	-	-	-	-
396		2	3	7	8	-	-	-	-	-	-
397		0	2	3	7	8	-	-	-	-	-
398		1	2	3	7	8	-	-	-	-	-
399		0	1	2	3	7	8	-	-	-	-
400		4	7	8	-	-	-	-	-	-	-
401		0	4	7	8	-	-	-	-	-	-
402		1	4	7	8	-	-	-	-	-	-
403		0	1	4	7	8	-	-	-	-	-
404		2	4	7	8	-	-	-	-	-	-
405		0	2	4	7	8	-	-	-	-	-
406		1	2	4	7	8	-	-	-	-	-
407		0	1	2	4	7	8	-	-	-	-
408		3	4	7	8	-	-	-	-	-	-
409		0	3	4	7	8	-	-	-	-	-
410		1	3	4	7	8	-	-	-	-	-
411		0	1	3	4	7	8	-	-	-	-
412		2	3	4	7	8	-	-	-	-	-

Tabel A.15 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(15)

<i>index</i> <i>Num</i>	0	1	2	3	4	5	6	7	8	9
413	0	2	3	4	7	8	-	-	-	-
414	1	2	3	4	7	8	-	-	-	-
415	0	1	2	3	4	7	8	-	-	-
416	5	7	8	-	-	-	-	-	-	-
417	0	5	7	8	-	-	-	-	-	-
418	1	5	7	8	-	-	-	-	-	-
419	0	1	5	7	8	-	-	-	-	-
420	2	5	7	8	-	-	-	-	-	-
421	0	2	5	7	8	-	-	-	-	-
422	1	2	5	7	8	-	-	-	-	-
423	0	1	2	5	7	8	-	-	-	-
424	3	5	7	8	-	-	-	-	-	-
425	0	3	5	7	8	-	-	-	-	-
426	1	3	5	7	8	-	-	-	-	-
427	0	1	3	5	7	8	-	-	-	-
428	2	3	5	7	8	-	-	-	-	-
429	0	2	3	5	7	8	-	-	-	-
430	1	2	3	5	7	8	-	-	-	-
431	0	1	2	3	5	7	8	-	-	-
432	4	5	7	8	-	-	-	-	-	-
433	0	4	5	7	8	-	-	-	-	-
434	1	4	5	7	8	-	-	-	-	-
435	0	1	4	5	7	8	-	-	-	-
436	2	4	5	7	8	-	-	-	-	-
437	0	2	4	5	7	8	-	-	-	-
438	1	2	4	5	7	8	-	-	-	-
439	0	1	2	4	5	7	8	-	-	-
440	3	4	5	7	8	-	-	-	-	-
441	0	3	4	5	7	8	-	-	-	-
442	1	3	4	5	7	8	-	-	-	-

Tabel A.16 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(16)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
443		0	1	3	4	5	7	8	-	-	-
444		2	3	4	5	7	8	-	-	-	-
445		0	2	3	4	5	7	8	-	-	-
446		1	2	3	4	5	7	8	-	-	-
447		0	1	2	3	4	5	7	8	-	-
448		6	7	8	-	-	-	-	-	-	-
449		0	6	7	8	-	-	-	-	-	-
450		1	6	7	8	-	-	-	-	-	-
451		0	1	6	7	8	-	-	-	-	-
452		2	6	7	8	-	-	-	-	-	-
453		0	2	6	7	8	-	-	-	-	-
454		1	2	6	7	8	-	-	-	-	-
455		0	1	2	6	7	8	-	-	-	-
456		3	6	7	8	-	-	-	-	-	-
457		0	3	6	7	8	-	-	-	-	-
458		1	3	6	7	8	-	-	-	-	-
459		0	1	3	6	7	8	-	-	-	-
460		2	3	6	7	8	-	-	-	-	-
461		0	2	3	6	7	8	-	-	-	-
462		1	2	3	6	7	8	-	-	-	-
463		0	1	2	3	6	7	8	-	-	-
464		4	6	7	8	-	-	-	-	-	-
465		0	4	6	7	8	-	-	-	-	-
466		1	4	6	7	8	-	-	-	-	-
467		0	1	4	6	7	8	-	-	-	-
468		2	4	6	7	8	-	-	-	-	-
469		0	2	4	6	7	8	-	-	-	-
470		1	2	4	6	7	8	-	-	-	-
471		0	1	2	4	6	7	8	-	-	-
472		3	4	6	7	8	-	-	-	-	-

Tabel A.17 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(17)

<i>index</i> <i>Num</i>	0	1	2	3	4	5	6	7	8	9
473	0	3	4	6	7	8	-	-	-	-
474	1	3	4	6	7	8	-	-	-	-
475	0	1	3	4	6	7	8	-	-	-
476	2	3	4	6	7	8	-	-	-	-
477	0	2	3	4	6	7	8	-	-	-
478	1	2	3	4	6	7	8	-	-	-
479	0	1	2	3	4	6	7	8	-	-
480	5	6	7	8	-	-	-	-	-	-
481	0	5	6	7	8	-	-	-	-	-
482	1	5	6	7	8	-	-	-	-	-
483	0	1	5	6	7	8	-	-	-	-
484	2	5	6	7	8	-	-	-	-	-
485	0	2	5	6	7	8	-	-	-	-
486	1	2	5	6	7	8	-	-	-	-
487	0	1	2	5	6	7	8	-	-	-
488	3	5	6	7	8	-	-	-	-	-
489	0	3	5	6	7	8	-	-	-	-
490	1	3	5	6	7	8	-	-	-	-
491	0	1	3	5	6	7	8	-	-	-
492	2	3	5	6	7	8	-	-	-	-
493	0	2	3	5	6	7	8	-	-	-
494	1	2	3	5	6	7	8	-	-	-
495	0	1	2	3	5	6	7	8	-	-
496	4	5	6	7	8	-	-	-	-	-
497	0	4	5	6	7	8	-	-	-	-
498	1	4	5	6	7	8	-	-	-	-
499	0	1	4	5	6	7	8	-	-	-
500	2	4	5	6	7	8	-	-	-	-
501	0	2	4	5	6	7	8	-	-	-
502	1	2	4	5	6	7	8	-	-	-

Tabel A.18 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(18)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
503		0	1	2	4	5	6	7	8	-	-
504		3	4	5	6	7	8	-	-	-	-
505		0	3	4	5	6	7	8	-	-	-
506		1	3	4	5	6	7	8	-	-	-
507		0	1	3	4	5	6	7	8	-	-
508		2	3	4	5	6	7	8	-	-	-
509		0	2	3	4	5	6	7	8	-	-
510		1	2	3	4	5	6	7	8	-	-
511		0	1	2	3	4	5	6	7	8	-
512		9	-	-	-	-	-	-	-	-	-
513		0	9	-	-	-	-	-	-	-	-
514		1	9	-	-	-	-	-	-	-	-
515		0	1	9	-	-	-	-	-	-	-
516		2	9	-	-	-	-	-	-	-	-
517		0	2	9	-	-	-	-	-	-	-
518		1	2	9	-	-	-	-	-	-	-
519		0	1	2	9	-	-	-	-	-	-
520		3	9	-	-	-	-	-	-	-	-
521		0	3	9	-	-	-	-	-	-	-
522		1	3	9	-	-	-	-	-	-	-
523		0	1	3	9	-	-	-	-	-	-
524		2	3	9	-	-	-	-	-	-	-
525		0	2	3	9	-	-	-	-	-	-
526		1	2	3	9	-	-	-	-	-	-
527		0	1	2	3	9	-	-	-	-	-
528		4	9	-	-	-	-	-	-	-	-
529		0	4	9	-	-	-	-	-	-	-
530		1	4	9	-	-	-	-	-	-	-
531		0	1	4	9	-	-	-	-	-	-
532		2	4	9	-	-	-	-	-	-	-

Tabel A.19 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(19)

<i>index</i> <i>Num</i>	0	1	2	3	4	5	6	7	8	9
533	0	2	4	9	-	-	-	-	-	-
534	1	2	4	9	-	-	-	-	-	-
535	0	1	2	4	9	-	-	-	-	-
536	3	4	9	-	-	-	-	-	-	-
537	0	3	4	9	-	-	-	-	-	-
538	1	3	4	9	-	-	-	-	-	-
539	0	1	3	4	9	-	-	-	-	-
540	2	3	4	9	-	-	-	-	-	-
541	0	2	3	4	9	-	-	-	-	-
542	1	2	3	4	9	-	-	-	-	-
543	0	1	2	3	4	9	-	-	-	-
544	5	9	-	-	-	-	-	-	-	-
545	0	5	9	-	-	-	-	-	-	-
546	1	5	9	-	-	-	-	-	-	-
547	0	1	5	9	-	-	-	-	-	-
548	2	5	9	-	-	-	-	-	-	-
549	0	2	5	9	-	-	-	-	-	-
550	1	2	5	9	-	-	-	-	-	-
551	0	1	2	5	9	-	-	-	-	-
552	3	5	9	-	-	-	-	-	-	-
553	0	3	5	9	-	-	-	-	-	-
554	1	3	5	9	-	-	-	-	-	-
555	0	1	3	5	9	-	-	-	-	-
556	2	3	5	9	-	-	-	-	-	-
557	0	2	3	5	9	-	-	-	-	-
558	1	2	3	5	9	-	-	-	-	-
559	0	1	2	3	5	9	-	-	-	-
560	4	5	9	-	-	-	-	-	-	-
561	0	4	5	9	-	-	-	-	-	-
562	1	4	5	9	-	-	-	-	-	-

Tabel A.20 Tabel himpunan setBit setelah fungsi preprocess dijalankan (20)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
563		0	1	4	5	9	-	-	-	-	-
564		2	4	5	9	-	-	-	-	-	-
565		0	2	4	5	9	-	-	-	-	-
566		1	2	4	5	9	-	-	-	-	-
567		0	1	2	4	5	9	-	-	-	-
568		3	4	5	9	-	-	-	-	-	-
569		0	3	4	5	9	-	-	-	-	-
570		1	3	4	5	9	-	-	-	-	-
571		0	1	3	4	5	9	-	-	-	-
572		2	3	4	5	9	-	-	-	-	-
573		0	2	3	4	5	9	-	-	-	-
574		1	2	3	4	5	9	-	-	-	-
575		0	1	2	3	4	5	9	-	-	-
576		6	9	-	-	-	-	-	-	-	-
577		0	6	9	-	-	-	-	-	-	-
578		1	6	9	-	-	-	-	-	-	-
579		0	1	6	9	-	-	-	-	-	-
580		2	6	9	-	-	-	-	-	-	-
581		0	2	6	9	-	-	-	-	-	-
582		1	2	6	9	-	-	-	-	-	-
583		0	1	2	6	9	-	-	-	-	-
584		3	6	9	-	-	-	-	-	-	-
585		0	3	6	9	-	-	-	-	-	-
586		1	3	6	9	-	-	-	-	-	-
587		0	1	3	6	9	-	-	-	-	-
588		2	3	6	9	-	-	-	-	-	-
589		0	2	3	6	9	-	-	-	-	-
590		1	2	3	6	9	-	-	-	-	-
591		0	1	2	3	6	9	-	-	-	-
592		4	6	9	-	-	-	-	-	-	-

Tabel A.21 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(21)

<i>index</i> <i>Num</i>	0	1	2	3	4	5	6	7	8	9
593	0	4	6	9	-	-	-	-	-	-
594	1	4	6	9	-	-	-	-	-	-
595	0	1	4	6	9	-	-	-	-	-
596	2	4	6	9	-	-	-	-	-	-
597	0	2	4	6	9	-	-	-	-	-
598	1	2	4	6	9	-	-	-	-	-
599	0	1	2	4	6	9	-	-	-	-
600	3	4	6	9	-	-	-	-	-	-
601	0	3	4	6	9	-	-	-	-	-
602	1	3	4	6	9	-	-	-	-	-
603	0	1	3	4	6	9	-	-	-	-
604	2	3	4	6	9	-	-	-	-	-
605	0	2	3	4	6	9	-	-	-	-
606	1	2	3	4	6	9	-	-	-	-
607	0	1	2	3	4	6	9	-	-	-
608	5	6	9	-	-	-	-	-	-	-
609	0	5	6	9	-	-	-	-	-	-
610	1	5	6	9	-	-	-	-	-	-
611	0	1	5	6	9	-	-	-	-	-
612	2	5	6	9	-	-	-	-	-	-
613	0	2	5	6	9	-	-	-	-	-
614	1	2	5	6	9	-	-	-	-	-
615	0	1	2	5	6	9	-	-	-	-
616	3	5	6	9	-	-	-	-	-	-
617	0	3	5	6	9	-	-	-	-	-
618	1	3	5	6	9	-	-	-	-	-
619	0	1	3	5	6	9	-	-	-	-
620	2	3	5	6	9	-	-	-	-	-
621	0	2	3	5	6	9	-	-	-	-
622	1	2	3	5	6	9	-	-	-	-

Tabel A.22 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(22)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
623		0	1	2	3	5	6	9	-	-	-
624		4	5	6	9	-	-	-	-	-	-
625		0	4	5	6	9	-	-	-	-	-
626		1	4	5	6	9	-	-	-	-	-
627		0	1	4	5	6	9	-	-	-	-
628		2	4	5	6	9	-	-	-	-	-
629		0	2	4	5	6	9	-	-	-	-
630		1	2	4	5	6	9	-	-	-	-
631		0	1	2	4	5	6	9	-	-	-
632		3	4	5	6	9	-	-	-	-	-
633		0	3	4	5	6	9	-	-	-	-
634		1	3	4	5	6	9	-	-	-	-
635		0	1	3	4	5	6	9	-	-	-
636		2	3	4	5	6	9	-	-	-	-
637		0	2	3	4	5	6	9	-	-	-
638		1	2	3	4	5	6	9	-	-	-
639		0	1	2	3	4	5	6	9	-	-
640		7	9	-	-	-	-	-	-	-	-
641		0	7	9	-	-	-	-	-	-	-
642		1	7	9	-	-	-	-	-	-	-
643		0	1	7	9	-	-	-	-	-	-
644		2	7	9	-	-	-	-	-	-	-
645		0	2	7	9	-	-	-	-	-	-
646		1	2	7	9	-	-	-	-	-	-
647		0	1	2	7	9	-	-	-	-	-
648		3	7	9	-	-	-	-	-	-	-
649		0	3	7	9	-	-	-	-	-	-
650		1	3	7	9	-	-	-	-	-	-
651		0	1	3	7	9	-	-	-	-	-
652		2	3	7	9	-	-	-	-	-	-

Tabel A.23 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(23)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
653		0	2	3	7	9	-	-	-	-	-
654		1	2	3	7	9	-	-	-	-	-
655		0	1	2	3	7	9	-	-	-	-
656		4	7	9	-	-	-	-	-	-	-
657		0	4	7	9	-	-	-	-	-	-
658		1	4	7	9	-	-	-	-	-	-
659		0	1	4	7	9	-	-	-	-	-
660		2	4	7	9	-	-	-	-	-	-
661		0	2	4	7	9	-	-	-	-	-
662		1	2	4	7	9	-	-	-	-	-
663		0	1	2	4	7	9	-	-	-	-
664		3	4	7	9	-	-	-	-	-	-
665		0	3	4	7	9	-	-	-	-	-
666		1	3	4	7	9	-	-	-	-	-
667		0	1	3	4	7	9	-	-	-	-
668		2	3	4	7	9	-	-	-	-	-
669		0	2	3	4	7	9	-	-	-	-
670		1	2	3	4	7	9	-	-	-	-
671		0	1	2	3	4	7	9	-	-	-
672		5	7	9	-	-	-	-	-	-	-
673		0	5	7	9	-	-	-	-	-	-
674		1	5	7	9	-	-	-	-	-	-
675		0	1	5	7	9	-	-	-	-	-
676		2	5	7	9	-	-	-	-	-	-
677		0	2	5	7	9	-	-	-	-	-
678		1	2	5	7	9	-	-	-	-	-
679		0	1	2	5	7	9	-	-	-	-
680		3	5	7	9	-	-	-	-	-	-
681		0	3	5	7	9	-	-	-	-	-
682		1	3	5	7	9	-	-	-	-	-

Tabel A.24 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(24)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
683		0	1	3	5	7	9	-	-	-	-
684		2	3	5	7	9	-	-	-	-	-
685		0	2	3	5	7	9	-	-	-	-
686		1	2	3	5	7	9	-	-	-	-
687		0	1	2	3	5	7	9	-	-	-
688		4	5	7	9	-	-	-	-	-	-
689		0	4	5	7	9	-	-	-	-	-
690		1	4	5	7	9	-	-	-	-	-
691		0	1	4	5	7	9	-	-	-	-
692		2	4	5	7	9	-	-	-	-	-
693		0	2	4	5	7	9	-	-	-	-
694		1	2	4	5	7	9	-	-	-	-
695		0	1	2	4	5	7	9	-	-	-
696		3	4	5	7	9	-	-	-	-	-
697		0	3	4	5	7	9	-	-	-	-
698		1	3	4	5	7	9	-	-	-	-
699		0	1	3	4	5	7	9	-	-	-
700		2	3	4	5	7	9	-	-	-	-
701		0	2	3	4	5	7	9	-	-	-
702		1	2	3	4	5	7	9	-	-	-
703		0	1	2	3	4	5	7	9	-	-
704		6	7	9	-	-	-	-	-	-	-
705		0	6	7	9	-	-	-	-	-	-
706		1	6	7	9	-	-	-	-	-	-
707		0	1	6	7	9	-	-	-	-	-
708		2	6	7	9	-	-	-	-	-	-
709		0	2	6	7	9	-	-	-	-	-
710		1	2	6	7	9	-	-	-	-	-
711		0	1	2	6	7	9	-	-	-	-
712		3	6	7	9	-	-	-	-	-	-

Tabel A.25 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(25)

<i>index</i> <i>Num</i>	0	1	2	3	4	5	6	7	8	9
713	0	3	6	7	9	-	-	-	-	-
714	1	3	6	7	9	-	-	-	-	-
715	0	1	3	6	7	9	-	-	-	-
716	2	3	6	7	9	-	-	-	-	-
717	0	2	3	6	7	9	-	-	-	-
718	1	2	3	6	7	9	-	-	-	-
719	0	1	2	3	6	7	9	-	-	-
720	4	6	7	9	-	-	-	-	-	-
721	0	4	6	7	9	-	-	-	-	-
722	1	4	6	7	9	-	-	-	-	-
723	0	1	4	6	7	9	-	-	-	-
724	2	4	6	7	9	-	-	-	-	-
725	0	2	4	6	7	9	-	-	-	-
726	1	2	4	6	7	9	-	-	-	-
727	0	1	2	4	6	7	9	-	-	-
728	3	4	6	7	9	-	-	-	-	-
729	0	3	4	6	7	9	-	-	-	-
730	1	3	4	6	7	9	-	-	-	-
731	0	1	3	4	6	7	9	-	-	-
732	2	3	4	6	7	9	-	-	-	-
733	0	2	3	4	6	7	9	-	-	-
734	1	2	3	4	6	7	9	-	-	-
735	0	1	2	3	4	6	7	9	-	-
736	5	6	7	9	-	-	-	-	-	-
737	0	5	6	7	9	-	-	-	-	-
738	1	5	6	7	9	-	-	-	-	-
739	0	1	5	6	7	9	-	-	-	-
740	2	5	6	7	9	-	-	-	-	-
741	0	2	5	6	7	9	-	-	-	-
742	1	2	5	6	7	9	-	-	-	-

Tabel A.26 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(26)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
743		0	1	2	5	6	7	9	-	-	-
744		3	5	6	7	9	-	-	-	-	-
745		0	3	5	6	7	9	-	-	-	-
746		1	3	5	6	7	9	-	-	-	-
747		0	1	3	5	6	7	9	-	-	-
748		2	3	5	6	7	9	-	-	-	-
749		0	2	3	5	6	7	9	-	-	-
750		1	2	3	5	6	7	9	-	-	-
751		0	1	2	3	5	6	7	9	-	-
752		4	5	6	7	9	-	-	-	-	-
753		0	4	5	6	7	9	-	-	-	-
754		1	4	5	6	7	9	-	-	-	-
755		0	1	4	5	6	7	9	-	-	-
756		2	4	5	6	7	9	-	-	-	-
757		0	2	4	5	6	7	9	-	-	-
758		1	2	4	5	6	7	9	-	-	-
759		0	1	2	4	5	6	7	9	-	-
760		3	4	5	6	7	9	-	-	-	-
761		0	3	4	5	6	7	9	-	-	-
762		1	3	4	5	6	7	9	-	-	-
763		0	1	3	4	5	6	7	9	-	-
764		2	3	4	5	6	7	9	-	-	-
765		0	2	3	4	5	6	7	9	-	-
766		1	2	3	4	5	6	7	9	-	-
767		0	1	2	3	4	5	6	7	9	-
768		8	9	-	-	-	-	-	-	-	-
769		0	8	9	-	-	-	-	-	-	-
770		1	8	9	-	-	-	-	-	-	-
771		0	1	8	9	-	-	-	-	-	-
772		2	8	9	-	-	-	-	-	-	-

Tabel A.27 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(27)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
773	0	2	8	9	-	-	-	-	-	-	-
774	1	2	8	9	-	-	-	-	-	-	-
775	0	1	2	8	9	-	-	-	-	-	-
776	3	8	9	-	-	-	-	-	-	-	-
777	0	3	8	9	-	-	-	-	-	-	-
778	1	3	8	9	-	-	-	-	-	-	-
779	0	1	3	8	9	-	-	-	-	-	-
780	2	3	8	9	-	-	-	-	-	-	-
781	0	2	3	8	9	-	-	-	-	-	-
782	1	2	3	8	9	-	-	-	-	-	-
783	0	1	2	3	8	9	-	-	-	-	-
784	4	8	9	-	-	-	-	-	-	-	-
785	0	4	8	9	-	-	-	-	-	-	-
786	1	4	8	9	-	-	-	-	-	-	-
787	0	1	4	8	9	-	-	-	-	-	-
788	2	4	8	9	-	-	-	-	-	-	-
789	0	2	4	8	9	-	-	-	-	-	-
790	1	2	4	8	9	-	-	-	-	-	-
791	0	1	2	4	8	9	-	-	-	-	-
792	3	4	8	9	-	-	-	-	-	-	-
793	0	3	4	8	9	-	-	-	-	-	-
794	1	3	4	8	9	-	-	-	-	-	-
795	0	1	3	4	8	9	-	-	-	-	-
796	2	3	4	8	9	-	-	-	-	-	-
797	0	2	3	4	8	9	-	-	-	-	-
798	1	2	3	4	8	9	-	-	-	-	-
799	0	1	2	3	4	8	9	-	-	-	-
800	5	8	9	-	-	-	-	-	-	-	-
801	0	5	8	9	-	-	-	-	-	-	-
802	1	5	8	9	-	-	-	-	-	-	-

Tabel A.28 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(28)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
803		0	1	5	8	9	-	-	-	-	-
804		2	5	8	9	-	-	-	-	-	-
805		0	2	5	8	9	-	-	-	-	-
806		1	2	5	8	9	-	-	-	-	-
807		0	1	2	5	8	9	-	-	-	-
808		3	5	8	9	-	-	-	-	-	-
809		0	3	5	8	9	-	-	-	-	-
810		1	3	5	8	9	-	-	-	-	-
811		0	1	3	5	8	9	-	-	-	-
812		2	3	5	8	9	-	-	-	-	-
813		0	2	3	5	8	9	-	-	-	-
814		1	2	3	5	8	9	-	-	-	-
815		0	1	2	3	5	8	9	-	-	-
816		4	5	8	9	-	-	-	-	-	-
817		0	4	5	8	9	-	-	-	-	-
818		1	4	5	8	9	-	-	-	-	-
819		0	1	4	5	8	9	-	-	-	-
820		2	4	5	8	9	-	-	-	-	-
821		0	2	4	5	8	9	-	-	-	-
822		1	2	4	5	8	9	-	-	-	-
823		0	1	2	4	5	8	9	-	-	-
824		3	4	5	8	9	-	-	-	-	-
825		0	3	4	5	8	9	-	-	-	-
826		1	3	4	5	8	9	-	-	-	-
827		0	1	3	4	5	8	9	-	-	-
828		2	3	4	5	8	9	-	-	-	-
829		0	2	3	4	5	8	9	-	-	-
830		1	2	3	4	5	8	9	-	-	-
831		0	1	2	3	4	5	8	9	-	-
832		6	8	9	-	-	-	-	-	-	-

Tabel A.29 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(29)

<i>index</i> <i>Num</i>	0	1	2	3	4	5	6	7	8	9
833	0	6	8	9	-	-	-	-	-	-
834	1	6	8	9	-	-	-	-	-	-
835	0	1	6	8	9	-	-	-	-	-
836	2	6	8	9	-	-	-	-	-	-
837	0	2	6	8	9	-	-	-	-	-
838	1	2	6	8	9	-	-	-	-	-
839	0	1	2	6	8	9	-	-	-	-
840	3	6	8	9	-	-	-	-	-	-
841	0	3	6	8	9	-	-	-	-	-
842	1	3	6	8	9	-	-	-	-	-
843	0	1	3	6	8	9	-	-	-	-
844	2	3	6	8	9	-	-	-	-	-
845	0	2	3	6	8	9	-	-	-	-
846	1	2	3	6	8	9	-	-	-	-
847	0	1	2	3	6	8	9	-	-	-
848	4	6	8	9	-	-	-	-	-	-
849	0	4	6	8	9	-	-	-	-	-
850	1	4	6	8	9	-	-	-	-	-
851	0	1	4	6	8	9	-	-	-	-
852	2	4	6	8	9	-	-	-	-	-
853	0	2	4	6	8	9	-	-	-	-
854	1	2	4	6	8	9	-	-	-	-
855	0	1	2	4	6	8	9	-	-	-
856	3	4	6	8	9	-	-	-	-	-
857	0	3	4	6	8	9	-	-	-	-
858	1	3	4	6	8	9	-	-	-	-
859	0	1	3	4	6	8	9	-	-	-
860	2	3	4	6	8	9	-	-	-	-
861	0	2	3	4	6	8	9	-	-	-
862	1	2	3	4	6	8	9	-	-	-

Tabel A.30 Tabel himpunan setBit setelah fungsi preprocess dijalankan (30)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
863		0	1	2	3	4	6	8	9	-	-
864		5	6	8	9	-	-	-	-	-	-
865		0	5	6	8	9	-	-	-	-	-
866		1	5	6	8	9	-	-	-	-	-
867		0	1	5	6	8	9	-	-	-	-
868		2	5	6	8	9	-	-	-	-	-
869		0	2	5	6	8	9	-	-	-	-
870		1	2	5	6	8	9	-	-	-	-
871		0	1	2	5	6	8	9	-	-	-
872		3	5	6	8	9	-	-	-	-	-
873		0	3	5	6	8	9	-	-	-	-
874		1	3	5	6	8	9	-	-	-	-
875		0	1	3	5	6	8	9	-	-	-
876		2	3	5	6	8	9	-	-	-	-
877		0	2	3	5	6	8	9	-	-	-
878		1	2	3	5	6	8	9	-	-	-
879		0	1	2	3	5	6	8	9	-	-
880		4	5	6	8	9	-	-	-	-	-
881		0	4	5	6	8	9	-	-	-	-
882		1	4	5	6	8	9	-	-	-	-
883		0	1	4	5	6	8	9	-	-	-
884		2	4	5	6	8	9	-	-	-	-
885		0	2	4	5	6	8	9	-	-	-
886		1	2	4	5	6	8	9	-	-	-
887		0	1	2	4	5	6	8	9	-	-
888		3	4	5	6	8	9	-	-	-	-
889		0	3	4	5	6	8	9	-	-	-
890		1	3	4	5	6	8	9	-	-	-
891		0	1	3	4	5	6	8	9	-	-
892		2	3	4	5	6	8	9	-	-	-

Tabel A.31 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(31)

<i>index</i> <i>Num</i>	0	1	2	3	4	5	6	7	8	9
893	0	2	3	4	5	6	8	9	-	-
894	1	2	3	4	5	6	8	9	-	-
895	0	1	2	3	4	5	6	8	9	-
896	7	8	9	-	-	-	-	-	-	-
897	0	7	8	9	-	-	-	-	-	-
898	1	7	8	9	-	-	-	-	-	-
899	0	1	7	8	9	-	-	-	-	-
900	2	7	8	9	-	-	-	-	-	-
901	0	2	7	8	9	-	-	-	-	-
902	1	2	7	8	9	-	-	-	-	-
903	0	1	2	7	8	9	-	-	-	-
904	3	7	8	9	-	-	-	-	-	-
905	0	3	7	8	9	-	-	-	-	-
906	1	3	7	8	9	-	-	-	-	-
907	0	1	3	7	8	9	-	-	-	-
908	2	3	7	8	9	-	-	-	-	-
909	0	2	3	7	8	9	-	-	-	-
910	1	2	3	7	8	9	-	-	-	-
911	0	1	2	3	7	8	9	-	-	-
912	4	7	8	9	-	-	-	-	-	-
913	0	4	7	8	9	-	-	-	-	-
914	1	4	7	8	9	-	-	-	-	-
915	0	1	4	7	8	9	-	-	-	-
916	2	4	7	8	9	-	-	-	-	-
917	0	2	4	7	8	9	-	-	-	-
918	1	2	4	7	8	9	-	-	-	-
919	0	1	2	4	7	8	9	-	-	-
920	3	4	7	8	9	-	-	-	-	-
921	0	3	4	7	8	9	-	-	-	-
922	1	3	4	7	8	9	-	-	-	-

Tabel A.32 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(32)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
923		0	1	3	4	7	8	9	-	-	-
924		2	3	4	7	8	9	-	-	-	-
925		0	2	3	4	7	8	9	-	-	-
926		1	2	3	4	7	8	9	-	-	-
927		0	1	2	3	4	7	8	9	-	-
928		5	7	8	9	-	-	-	-	-	-
929		0	5	7	8	9	-	-	-	-	-
930		1	5	7	8	9	-	-	-	-	-
931		0	1	5	7	8	9	-	-	-	-
932		2	5	7	8	9	-	-	-	-	-
933		0	2	5	7	8	9	-	-	-	-
934		1	2	5	7	8	9	-	-	-	-
935		0	1	2	5	7	8	9	-	-	-
936		3	5	7	8	9	-	-	-	-	-
937		0	3	5	7	8	9	-	-	-	-
938		1	3	5	7	8	9	-	-	-	-
939		0	1	3	5	7	8	9	-	-	-
940		2	3	5	7	8	9	-	-	-	-
941		0	2	3	5	7	8	9	-	-	-
942		1	2	3	5	7	8	9	-	-	-
943		0	1	2	3	5	7	8	9	-	-
944		4	5	7	8	9	-	-	-	-	-
945		0	4	5	7	8	9	-	-	-	-
946		1	4	5	7	8	9	-	-	-	-
947		0	1	4	5	7	8	9	-	-	-
948		2	4	5	7	8	9	-	-	-	-
949		0	2	4	5	7	8	9	-	-	-
950		1	2	4	5	7	8	9	-	-	-
951		0	1	2	4	5	7	8	9	-	-
952		3	4	5	7	8	9	-	-	-	-

Tabel A.33 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(33)

<i>index</i> <i>Num</i>	0	1	2	3	4	5	6	7	8	9
953	0	3	4	5	7	8	9	-	-	-
954	1	3	4	5	7	8	9	-	-	-
955	0	1	3	4	5	7	8	9	-	-
956	2	3	4	5	7	8	9	-	-	-
957	0	2	3	4	5	7	8	9	-	-
958	1	2	3	4	5	7	8	9	-	-
959	0	1	2	3	4	5	7	8	9	-
960	6	7	8	9	-	-	-	-	-	-
961	0	6	7	8	9	-	-	-	-	-
962	1	6	7	8	9	-	-	-	-	-
963	0	1	6	7	8	9	-	-	-	-
964	2	6	7	8	9	-	-	-	-	-
965	0	2	6	7	8	9	-	-	-	-
966	1	2	6	7	8	9	-	-	-	-
967	0	1	2	6	7	8	9	-	-	-
968	3	6	7	8	9	-	-	-	-	-
969	0	3	6	7	8	9	-	-	-	-
970	1	3	6	7	8	9	-	-	-	-
971	0	1	3	6	7	8	9	-	-	-
972	2	3	6	7	8	9	-	-	-	-
973	0	2	3	6	7	8	9	-	-	-
974	1	2	3	6	7	8	9	-	-	-
975	0	1	2	3	6	7	8	9	-	-
976	4	6	7	8	9	-	-	-	-	-
977	0	4	6	7	8	9	-	-	-	-
978	1	4	6	7	8	9	-	-	-	-
979	0	1	4	6	7	8	9	-	-	-
980	2	4	6	7	8	9	-	-	-	-
981	0	2	4	6	7	8	9	-	-	-
982	1	2	4	6	7	8	9	-	-	-

Tabel A.34 Tabel himpunan setBit setelah fungsi preprocess dijalankan
 (34)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
983		0	1	2	4	6	7	8	9	-	-
984		3	4	6	7	8	9	-	-	-	-
985		0	3	4	6	7	8	9	-	-	-
986		1	3	4	6	7	8	9	-	-	-
987		0	1	3	4	6	7	8	9	-	-
988		2	3	4	6	7	8	9	-	-	-
989		0	2	3	4	6	7	8	9	-	-
990		1	2	3	4	6	7	8	9	-	-
991		0	1	2	3	4	6	7	8	9	-
992		5	6	7	8	9	-	-	-	-	-
993		0	5	6	7	8	9	-	-	-	-
994		1	5	6	7	8	9	-	-	-	-
995		0	1	5	6	7	8	9	-	-	-
996		2	5	6	7	8	9	-	-	-	-
997		0	2	5	6	7	8	9	-	-	-
998		1	2	5	6	7	8	9	-	-	-
999		0	1	2	5	6	7	8	9	-	-
1000		3	5	6	7	8	9	-	-	-	-
1001		0	3	5	6	7	8	9	-	-	-
1002		1	3	5	6	7	8	9	-	-	-
1003		0	1	3	5	6	7	8	9	-	-
1004		2	3	5	6	7	8	9	-	-	-
1005		0	2	3	5	6	7	8	9	-	-
1006		1	2	3	5	6	7	8	9	-	-
1007		0	1	2	3	5	6	7	8	9	-
1008		4	5	6	7	8	9	-	-	-	-
1009		0	4	5	6	7	8	9	-	-	-
1010		1	4	5	6	7	8	9	-	-	-
1011		0	1	4	5	6	7	8	9	-	-
1012		2	4	5	6	7	8	9	-	-	-

Tabel A.35 Tabel himpunan setBit setelah fungsi preprocess dijalankan
(35)

<i>Num</i>	<i>index</i>	0	1	2	3	4	5	6	7	8	9
1013		0	2	4	5	6	7	8	9	-	-
1014		1	2	4	5	6	7	8	9	-	-
1015		0	1	2	4	5	6	7	8	9	-
1016		3	4	5	6	7	8	9	-	-	-
1017		0	3	4	5	6	7	8	9	-	-
1018		1	3	4	5	6	7	8	9	-	-
1019		0	1	3	4	5	6	7	8	9	-
1020		2	3	4	5	6	7	8	9	-	-
1021		0	2	3	4	5	6	7	8	9	-
1022		1	2	3	4	5	6	7	8	9	-
1023		0	1	2	3	4	5	6	7	8	9

Halaman ini sengaja dikosongkan

LAMPIRAN B

HASIL UJI COBA KEBENARAN PADA SITUS SPOJ

198552011	<input checked="" type="checkbox"/> 2017-06-04 16:43:41	Playing with Words	accepted <small>2011 - 1840000-0</small>	0.08	19M	CPP14-CLANG
-----------	--	--------------------	---	------	-----	-------------

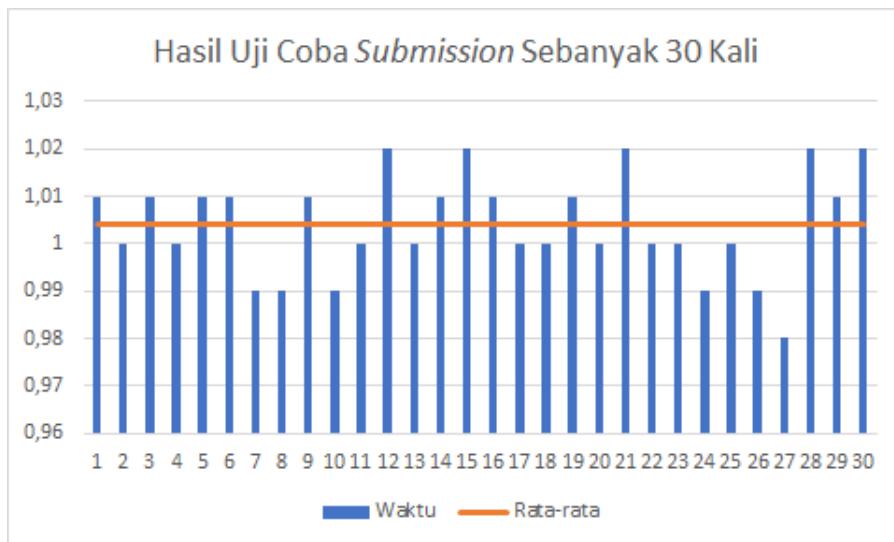
Gambar B.1 Hasil uji coba pada situs penilaian SPOJ

ID	DATE	PROBLEM	RESULT	TIME	MEM	LANG
19811567	2017-07-17 13:28:30	Playing with Words	accepted	1.01	19M	CPP14-CLANG
19811422	2017-07-17 12:44:25	Playing with Words	accepted	1.00	19M	CPP14-CLANG
19811418	2017-07-17 12:14:02	Playing with Words	accepted	1.01	19M	CPP14-CLANG
19810874	2017-07-17 11:04:31	Playing with Words	accepted	1.00	19M	CPP14-CLANG
19810580	2017-07-17 09:41:01	Playing with Words	accepted	1.01	19M	CPP14-CLANG
19810492	2017-07-17 09:05:43	Playing with Words	accepted	1.01	19M	CPP14-CLANG
19810435	2017-07-17 08:44:12	Playing with Words	accepted	0.99	19M	CPP14-CLANG
19810221	2017-07-17 07:37:13	Playing with Words	accepted	0.99	19M	CPP14-CLANG
19809996	2017-07-17 06:03:17	Playing with Words	accepted	1.01	19M	CPP14-CLANG
19809642	2017-07-17 04:00:22	Playing with Words	accepted	0.99	19M	CPP14-CLANG
19809514	2017-07-17 02:35:27	Playing with Words	accepted	1.00	19M	CPP14-CLANG
19809350	2017-07-17 00:49:10	Playing with Words	accepted	1.02	19M	CPP14-CLANG
19809202	2017-07-16 23:32:10	Playing with Words	accepted	1.00	19M	CPP14-CLANG
19808679	2017-07-16 20:40:29	Playing with Words	accepted	1.01	19M	CPP14-CLANG
19808238	2017-07-16 19:01:45	Playing with Words	accepted	1.02	19M	CPP14-CLANG
19807636	2017-07-16 18:50:09	Playing with Words	accepted	1.01	19M	CPP14-CLANG
19807366	2017-07-16 16:02:35	Playing with Words	accepted	1.00	19M	CPP14-CLANG
19807080	2017-07-16 14:50:10	Playing with Words	accepted	1.00	19M	CPP14-CLANG
19806799	2017-07-16 13:34:05	Playing with Words	accepted	1.01	19M	CPP14-CLANG
19806351	2017-07-16 12:23:12	Playing with Words	accepted	1.00	19M	CPP14-CLANG

Gambar B.2 Hasil pengujian sebanyak 30 kali pada situs penilaian daring SPOJ (1)

ID	DATE	PROBLEM	RESULT	TIME	MEM	LANG
19806027	2017-07-16 11:18:05	Playing with Words	accepted	1.02	19M	CPP14-CLANG
19805542	2017-07-16 09:30:37	Playing with Words	accepted	1.00	19M	CPP14-CLANG
19805357	2017-07-16 08:33:09	Playing with Words	accepted	1.00	19M	CPP14-CLANG
19805239	2017-07-16 07:49:25	Playing with Words	accepted	0.99	19M	CPP14-CLANG
19805075	2017-07-16 06:51:17	Playing with Words	accepted	1.00	19M	CPP14-CLANG
19804751	2017-07-16 05:20:36	Playing with Words	accepted	0.99	19M	CPP14-CLANG
19804489	2017-07-16 04:05:31	Playing with Words	accepted	0.98	19M	CPP14-CLANG
19804250	2017-07-16 02:08:38	Playing with Words	accepted	1.02	19M	CPP14-CLANG
19802308	2017-07-15 17:28:24	Playing with Words	accepted	1.01	19M	CPP14-CLANG
19801800	2017-07-15 16:02:44	Playing with Words	accepted	1.02	19M	CPP14-CLANG

Gambar B.3 Hasil pengujian sebanyak 30 kali pada situs penilaian daring SPOJ (2)



Gambar B.4 Grafik hasil uji coba pada situs SPOJ sebanyak 30 kali

LAMPIRAN C

TABEL SIMULASI PERHITUNGAN JUMLAH KEMUNGKINAN *STRING* ORIG1 DAN ORIG2 PADA KASUS *STRING* AD1=C, *STRING* AD2=N DAN X=1

Tabel C.1 Simulasi perhitungan jumlah kombinasi *string orig1* tanpa operasi *replace* dengan *dist* = 0 pada kasus *string ad1 = c, string ad2 = n* dan *X = 1*

Fungsi	Perhitungan Nilai	Nilai
$F_{(c,0,1)}$	<i>base case</i>	1
$F_{(c,1,1)}$	$F_{(c,0,1)}$	1

Tabel C.2 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist* = 0 pada kasus *string ad1 = c, string ad2 = n* dan *X = 1*

Fungsi	Perhitungan Nilai	Nilai
$G_{(n,0,0)}$	<i>base case</i>	0
$F_{(n,0,1)}$	<i>base case</i>	1
$F_{(n,0,1)}$	<i>base case</i>	1
$G_{(n,1,0)}$	$G_{(n,0,0)} + F_{(n,0,1)} + F_{(n,0,1)}$	2

Tabel C.3 Simulasi perhitungan jumlah kombinasi $string\ orig1$ dengan operasi $replace$ dengan $dist = 0$ pada kasus $string\ ad1 = c$, $string\ ad2 = n$ dan $X = 1$

Fungsi	Perhitungan Nilai	Nilai
$G_{(c,0,1)}$	<i>base case</i>	0
$F_{(c,0,2)}$	<i>base case</i>	0
$F_{(c,0,2)}$	<i>base case</i>	0
$G_{(c,1,1)}$	$G_{(c,0,1)} + F_{(c,0,2)} + F_{(c,0,2)}$	0

Tabel C.4 Simulasi perhitungan jumlah kombinasi $string\ orig2$ tanpa operasi $replace$ dengan $dist = 0$ pada kasus $string\ ad1 = c$, $string\ ad2 = n$ dan $X = 1$

Fungsi	Perhitungan Nilai	Nilai
$F_{(n,0,0)}$	<i>base case</i>	0
$F_{(n,1,0)}$	$F_{(n,0,0)}$	0

Tabel C.5 Simulasi perhitungan jumlah kombinasi $string\ orig1$ tanpa operasi $replace$ dengan $dist = 1$ pada kasus $string\ ad1 = c$, $string\ ad2 = n$ dan $X = 1$

Fungsi	Perhitungan Nilai	Nilai
$F_{(c,0,0)}$	<i>base case</i>	0
$F_{(c,1,0)}$	$F_{(c,0,0)}$	0

Tabel C.6 Simulasi perhitungan jumlah kombinasi $string\ orig2$ dengan operasi $replace$ dengan $dist = 1$ pada kasus $string\ ad1 = c$, $string\ ad2 = n$ dan $X = 1$

Fungsi	Perhitungan Nilai	Nilai
$G_{(n,0,1)}$	<i>base case</i>	0
$F_{(n,0,2)}$	<i>base case</i>	0
$F_{(n,0,2)}$	<i>base case</i>	0
$G_{(n,1,1)}$	$G_{(n,0,1)} + F_{(n,0,2)} + F_{(n,0,2)}$	0

Tabel C.7 Simulasi perhitungan jumlah kombinasi $string\ orig1$ dengan operasi $replace$ dengan $dist = 1$ pada kasus $string\ ad1 = c$, $string\ ad2 = n$ dan $X = 1$

Fungsi	Perhitungan Nilai	Nilai
$G_{(c,0,0)}$	<i>base case</i>	0
$F_{(c,0,1)}$	<i>base case</i>	1
$F_{(c,0,1)}$	<i>base case</i>	1
$G_{(c,1,0)}$	$G_{(c,0,0)} + F_{(c,0,1)} + F_{(c,0,1)}$	2

Tabel C.8 Simulasi perhitungan jumlah kombinasi $string\ orig2$ tanpa operasi $replace$ dengan $dist = 1$ pada kasus $string\ ad1 = c$, $string\ ad2 = n$ dan $X = 1$

Fungsi	Perhitungan Nilai	Nilai
$F_{(n,0,1)}$	<i>base case</i>	1
$F_{(n,1,1)}$	$F_{(n,0,1)}$	1

Halaman ini sengaja dikosongkan

LAMPIRAN D

**TABEL SIMULASI PERHITUNGAN JUMLAH
KEMUNGKINAN *STRING* ORIG1 DAN ORIG2 PADA
KASUS *STRING* AD1=KBENH, *STRING* AD2=KBENH DAN
 $X=5$**

Tabel D.1 Simulasi perhitungan jumlah kombinasi *string orig1* tanpa operasi *replace* dengan $dist = 0$ pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan $X = 5$

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,0,5)}$	<i>base case</i>	1
$F_{(behkn,16,5)}$	$F_{(behkn,0,5)}$	1
$F_{(behkn,8,8)}$	<i>base case</i>	0
$F_{(behkn,24,5)}$	$F_{(behkn,16,5)} + F_{(behkn,8,8)}$	1
$F_{(behkn,20,8)}$	<i>base case</i>	0
$F_{(behkn,12,11)}$	<i>base case</i>	0
$F_{(behkn,28,5)}$	$F_{(behkn,24,5)} + F_{(behkn,20,8)} + F_{(behkn,12,11)}$	1
$F_{(behkn,26,8)}$	<i>base case</i>	0
$F_{(behkn,22,11)}$	<i>base case</i>	0
$F_{(behkn,14,14)}$	<i>base case</i>	0
$F_{(behkn,30,5)}$	$F_{(behkn,28,5)} + F_{(behkn,26,8)} + F_{(behkn,22,11)} + F_{(behkn,14,14)}$	1
$F_{(behkn,29,8)}$	<i>base case</i>	0
$F_{(behkn,27,11)}$	<i>base case</i>	0
$F_{(behkn,23,14)}$	<i>base case</i>	0

Tabel D.2 Simulasi perhitungan jumlah kombinasi *string orig1* tanpa operasi *replace* dengan *dist* = 0 pada kasus *string ad1 = kbenh, string ad2 = kbenh* dan *X* = 5

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,15,17)}$	<i>base case</i>	0
$F_{(behkn,31,5)}$	$F_{(behkn,30,5)} + F_{(behkn,29,8)} + F_{(behkn,27,11)} + F_{(behkn,23,14)} + F_{(behkn,15,17)}$	1

Tabel D.3 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist* = 0 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (1)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,0,0)}$	<i>base case</i>	0
$F_{(behkn,0,1)}$	<i>base case</i>	0
$F_{(behkn,0,1)}$	<i>base case</i>	0
$G_{(behkn,16,0)}$	$G_{(behkn,0,0)} + F_{(behkn,0,1)} + F_{(behkn,0,1)}$	0
$F_{(behkn,0,1)}$	<i>base case</i>	0
$F_{(behkn,16,1)}$	$F_{(behkn,0,1)}$	0
$F_{(behkn,16,1)}$	<i>memoF</i> $_{(behkn,16,1)}$	0
$G_{(behkn,0,6)}$	<i>base case</i>	0
$F_{(behkn,0,5)}$	<i>base case</i>	1
$F_{(behkn,0,7)}$	<i>base case</i>	0
$G_{(behkn,8,3)}$	$G_{(behkn,0,6)} + F_{(behkn,0,5)} + F_{(behkn,0,7)}$	1
$F_{(behkn,0,7)}$	<i>base case</i>	0
$F_{(behkn,8,4)}$	$F_{(behkn,0,7)}$	0
$F_{(behkn,0,5)}$	<i>base case</i>	1
$F_{(behkn,8,2)}$	$F_{(behkn,0,5)}$	1
$G_{(behkn,24,0)}$	$G_{(behkn,16,0)} + F_{(behkn,16,1)} + F_{(behkn,16,1)} + G_{(behkn,8,3)} + F_{(behkn,8,4)} + F_{(behkn,8,2)}$	2
$F_{(behkn,16,1)}$	<i>memoF</i> $_{(behkn,16,1)}$	0
$F_{(behkn,8,4)}$	<i>memoF</i> $_{(behkn,8,4)}$	0
$F_{(behkn,24,1)}$	$F_{(behkn,16,1)} + F_{(behkn,8,4)}$	0
$F_{(behkn,24,1)}$	<i>memoF</i> $_{(behkn,24,1)}$	0
$G_{(behkn,16,6)}$	<i>base case</i>	0
$F_{(behkn,0,5)}$	<i>base case</i>	1
$F_{(behkn,16,5)}$	$F_{(behkn,0,5)}$	1
$F_{(behkn,16,7)}$	<i>base case</i>	0

Tabel D.4 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist* = 0 pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan *X* = 5 (2)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,4,6)}$	<i>base case</i>	0
$F_{(behkn,4,7)}$	<i>base case</i>	0
$F_{(behkn,0,11)}$	<i>base case</i>	0
$F_{(behkn,4,5)}$	$F_{(behkn,0,11)}$	0
$G_{(behkn,20,3)}$	$G_{(behkn,16,6)} + F_{(behkn,16,5)} + F_{(behkn,16,7)} + G_{(behkn,4,6)} + F_{(behkn,4,7)} + F_{(behkn,4,5)}$	1
$F_{(behkn,16,7)}$	<i>base case</i>	0
$F_{(behkn,4,7)}$	<i>base case</i>	0
$F_{(behkn,20,4)}$	$F_{(behkn,16,7)} + F_{(behkn,4,7)}$	0
$F_{(behkn,16,5)}$	<i>memoF</i> $_{(behkn,16,5)}$	1
$F_{(behkn,4,5)}$	<i>memoF</i> $_{(behkn,4,5)}$	0
$F_{(behkn,20,2)}$	$F_{(behkn,16,5)} + F_{(behkn,4,5)}$	1
$G_{(behkn,12,6)}$	<i>base case</i>	0
$F_{(behkn,12,7)}$	<i>base case</i>	0
$F_{(behkn,8,8)}$	<i>base case</i>	0
$F_{(behkn,4,5)}$	<i>memoF</i> $_{(behkn,4,5)}$	0
$F_{(behkn,12,5)}$	$F_{(behkn,8,8)} + F_{(behkn,4,5)}$	0
$G_{(behkn,28,0)}$	$G_{(behkn,24,0)} + F_{(behkn,24,1)} + F_{(behkn,24,1)} + G_{(behkn,20,3)} + F_{(behkn,20,4)} + F_{(behkn,20,2)} + G_{(behkn,12,6)} + F_{(behkn,12,7)} + F_{(behkn,12,5)}$	4
$F_{(behkn,24,1)}$	<i>memoF</i> $_{(behkn,24,1)}$	0
$F_{(behkn,20,4)}$	<i>memoF</i> $_{(behkn,20,4)}$	0
$F_{(behkn,12,7)}$	<i>base case</i>	0
$F_{(behkn,28,1)}$	$F_{(behkn,24,1)} + F_{(behkn,20,4)} + F_{(behkn,12,7)}$	0

Tabel D.5 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist* = 0 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (3)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,28,1)}$	$memoF_{(behkn,28,1)}$	0
$G_{(behkn,24,6)}$	<i>base case</i>	0
$F_{(behkn,16,5)}$	$memoF_{(behkn,16,5)}$	1
$F_{(behkn,8,8)}$	<i>base case</i>	0
$F_{(behkn,24,5)}$	$F_{(behkn,16,5)} + F_{(behkn,8,8)}$	1
$F_{(behkn,24,7)}$	<i>base case</i>	0
$G_{(behkn,18,6)}$	<i>base case</i>	0
$F_{(behkn,18,7)}$	<i>base case</i>	0
$F_{(behkn,16,11)}$	<i>base case</i>	0
$F_{(behkn,2,8)}$	<i>base case</i>	0
$F_{(behkn,18,5)}$	$F_{(behkn,16,11)} + F_{(behkn,2,8)}$	0
$G_{(behkn,10,9)}$	<i>base case</i>	0
$F_{(behkn,10,10)}$	<i>base case</i>	0
$F_{(behkn,10,8)}$	<i>base case</i>	0
$G_{(behkn,26,3)}$	$G_{(behkn,24,6)} + F_{(behkn,24,5)} +$ $F_{(behkn,24,7)} + G_{(behkn,18,6)} +$ $F_{(behkn,18,7)} + F_{(behkn,18,5)} +$ $G_{(behkn,10,9)} + F_{(behkn,10,10)} +$ $F_{(behkn,10,8)}$	1
$F_{(behkn,24,7)}$	<i>base case</i>	0
$F_{(behkn,18,7)}$	<i>base case</i>	0
$F_{(behkn,10,10)}$	<i>base case</i>	0
$F_{(behkn,26,4)}$	$F_{(behkn,24,7)} + F_{(behkn,18,7)} +$ $F_{(behkn,10,10)}$	0
$F_{(behkn,24,5)}$	$memoF_{(behkn,24,5)}$	1
$F_{(behkn,18,5)}$	$memoF_{(behkn,18,5)}$	0
$F_{(behkn,10,8)}$	<i>base case</i>	0
$F_{(behkn,26,2)}$	$F_{(behkn,24,5)} + F_{(behkn,18,5)} +$ $F_{(behkn,10,8)}$	1

Tabel D.6 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist* = 0 pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan *X* = 5 (4)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,22,6)}$	<i>base case</i>	0
$F_{(behkn,22,7)}$	<i>base case</i>	0
$F_{(behkn,20,8)}$	<i>base case</i>	0
$F_{(behkn,18,5)}$	$memoF_{(behkn,18,5)}$	0
$F_{(behkn,6,11)}$	<i>base case</i>	0
$F_{(behkn,22,5)}$	$F_{(behkn,20,8)} + F_{(behkn,18,5)} + F_{(behkn,6,11)}$	0
$G_{(behkn,14,9)}$	<i>base case</i>	0
$F_{(behkn,14,10)}$	<i>base case</i>	0
$F_{(behkn,14,8)}$	<i>base case</i>	0
$G_{(behkn,30,0)}$	$G_{(behkn,28,0)} + F_{(behkn,28,1)} + F_{(behkn,28,1)} + G_{(behkn,26,3)} + F_{(behkn,26,4)} + F_{(behkn,26,2)} + G_{(behkn,22,6)} + F_{(behkn,22,7)} + F_{(behkn,22,5)} + G_{(behkn,14,9)} + F_{(behkn,14,10)} + F_{(behkn,14,8)}$	6
$F_{(behkn,28,1)}$	$memoF_{(behkn,28,1)}$	0
$F_{(behkn,26,4)}$	$memoF_{(behkn,26,4)}$	0
$F_{(behkn,22,7)}$	<i>base case</i>	0
$F_{(behkn,14,10)}$	<i>base case</i>	0
$F_{(behkn,30,1)}$	$F_{(behkn,28,1)} + F_{(behkn,26,4)} + F_{(behkn,22,7)} + F_{(behkn,14,10)}$	0
$F_{(behkn,30,1)}$	$memoF_{(behkn,30,1)}$	0
$G_{(behkn,28,6)}$	<i>base case</i>	0
$F_{(behkn,24,5)}$	$memoF_{(behkn,24,5)}$	1
$F_{(behkn,20,8)}$	<i>base case</i>	0
$F_{(behkn,12,11)}$	<i>base case</i>	0
$F_{(behkn,28,5)}$	$F_{(behkn,24,5)} + F_{(behkn,20,8)} + F_{(behkn,12,11)}$	1

Tabel D.7 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist = 0* pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan *X = 5 (5)*

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,28,7)}$	<i>base case</i>	0
$G_{(behkn,25,6)}$	<i>base case</i>	0
$F_{(behkn,25,7)}$	<i>base case</i>	0
$F_{(behkn,24,11)}$	<i>base case</i>	0
$F_{(behkn,17,8)}$	<i>base case</i>	0
$F_{(behkn,9,11)}$	<i>base case</i>	0
$F_{(behkn,25,5)}$	$F_{(behkn,24,11)} + F_{(behkn,17,8)} + F_{(behkn,9,11)}$	0
$G_{(behkn,21,9)}$	<i>base case</i>	0
$F_{(behkn,21,10)}$	<i>base case</i>	0
$F_{(behkn,21,8)}$	<i>base case</i>	0
$G_{(behkn,13,12)}$	<i>base case</i>	0
$F_{(behkn,13,13)}$	<i>base case</i>	0
$F_{(behkn,13,11)}$	<i>base case</i>	0
$G_{(behkn,29,3)}$	$G_{(behkn,28,6)} + F_{(behkn,28,5)} + F_{(behkn,28,7)} + G_{(behkn,25,6)} + F_{(behkn,25,7)} + F_{(behkn,25,5)} + G_{(behkn,21,9)} + F_{(behkn,21,10)} + F_{(behkn,21,8)} + G_{(behkn,13,12)} + F_{(behkn,13,13)} + F_{(behkn,13,11)}$	1
$F_{(behkn,28,7)}$	<i>base case</i>	0
$F_{(behkn,25,7)}$	<i>base case</i>	0
$F_{(behkn,21,10)}$	<i>base case</i>	0
$F_{(behkn,13,13)}$	<i>base case</i>	0
$F_{(behkn,29,4)}$	$F_{(behkn,28,7)} + F_{(behkn,25,7)} + F_{(behkn,21,10)} + F_{(behkn,13,13)}$	0
$F_{(behkn,28,5)}$	<i>memo</i> $F_{(behkn,28,5)}$	1
$F_{(behkn,25,5)}$	<i>memo</i> $F_{(behkn,25,5)}$	0
$F_{(behkn,21,8)}$	<i>base case</i>	0
$F_{(behkn,13,11)}$	<i>base case</i>	0

Tabel D.8 Simulasi perhitungan jumlah kombinasi $string\ orig2$ dengan operasi $replace$ dengan $dist = 0$ pada kasus $string\ ad1 = kbenh$, $string\ ad2 = kbenh$ dan $X = 5$ (6)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,27,7)}$	<i>base case</i>	0
$F_{(behkn,26,8)}$	<i>base case</i>	0
$F_{(behkn,25,5)}$	$memoF_{(behkn,25,5)}$	0
$F_{(behkn,19,11)}$	<i>base case</i>	0
$F_{(behkn,11,14)}$	<i>base case</i>	0
$F_{(behkn,27,5)}$	$F_{(behkn,26,8)} + F_{(behkn,25,5)} + F_{(behkn,19,11)} + F_{(behkn,11,14)}$	0
$G_{(behkn,23,9)}$	<i>base case</i>	0
$F_{(behkn,23,10)}$	<i>base case</i>	0
$F_{(behkn,23,8)}$	<i>base case</i>	0
$G_{(behkn,15,12)}$	<i>base case</i>	0
$F_{(behkn,15,13)}$	<i>base case</i>	0
$F_{(behkn,15,11)}$	<i>base case</i>	0
$G_{(behkn,31,0)}$	$G_{(behkn,30,0)} + F_{(behkn,30,1)} + F_{(behkn,30,1)} + G_{(behkn,29,3)} + F_{(behkn,29,4)} + F_{(behkn,29,2)} + G_{(behkn,27,6)} + F_{(behkn,27,7)} + F_{(behkn,27,5)} + G_{(behkn,23,9)} + F_{(behkn,23,10)} + F_{(behkn,23,8)} + G_{(behkn,15,12)} + F_{(behkn,15,13)} + F_{(behkn,15,11)}$	8

Tabel D.9 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist = 0* pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan *X = 5 (1)*

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,0,5)}$	<i>base case</i>	0
$F_{(behkn,0,6)}$	<i>base case</i>	0
$F_{(behkn,0,6)}$	<i>base case</i>	0
$G_{(behkn,16,5)}$	$G_{(behkn,0,5)} + F_{(behkn,0,6)} + F_{(behkn,0,6)}$	0
$F_{(behkn,16,6)}$	<i>base case</i>	0
$F_{(behkn,16,6)}$	<i>base case</i>	0
$G_{(behkn,8,8)}$	<i>base case</i>	0
$F_{(behkn,8,9)}$	<i>base case</i>	0
$F_{(behkn,8,7)}$	<i>base case</i>	0
$G_{(behkn,24,5)}$	$G_{(behkn,16,5)} + F_{(behkn,16,6)} + F_{(behkn,16,6)} + G_{(behkn,8,8)} + F_{(behkn,8,9)} + F_{(behkn,8,7)}$	0
$F_{(behkn,24,6)}$	<i>base case</i>	0
$F_{(behkn,24,6)}$	<i>base case</i>	0
$G_{(behkn,20,8)}$	<i>base case</i>	0
$F_{(behkn,20,9)}$	<i>base case</i>	0
$F_{(behkn,20,7)}$	<i>base case</i>	0
$G_{(behkn,12,11)}$	<i>base case</i>	0
$F_{(behkn,12,12)}$	<i>base case</i>	0
$F_{(behkn,12,10)}$	<i>base case</i>	0
$G_{(behkn,28,5)}$	$G_{(behkn,24,5)} + F_{(behkn,24,6)} + F_{(behkn,24,6)} + G_{(behkn,20,8)} + F_{(behkn,20,9)} + F_{(behkn,20,7)} + G_{(behkn,12,11)} + F_{(behkn,12,12)} + F_{(behkn,12,10)}$	0
$F_{(behkn,28,6)}$	<i>base case</i>	0
$F_{(behkn,28,6)}$	<i>base case</i>	0

Tabel D.10 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist* = 0 pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan *X* = 5 (2)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,26,8)}$	<i>base case</i>	0
$F_{(behkn,26,9)}$	<i>base case</i>	0
$F_{(behkn,26,7)}$	<i>base case</i>	0
$G_{(behkn,22,11)}$	<i>base case</i>	0
$F_{(behkn,22,12)}$	<i>base case</i>	0
$F_{(behkn,22,10)}$	<i>base case</i>	0
$G_{(behkn,14,14)}$	<i>base case</i>	0
$F_{(behkn,14,15)}$	<i>base case</i>	0
$F_{(behkn,14,13)}$	<i>base case</i>	0
$G_{(behkn,30,5)}$	$G_{(behkn,28,5)} + F_{(behkn,28,6)} +$ $F_{(behkn,28,6)} + G_{(behkn,26,8)} +$ $F_{(behkn,26,9)} + F_{(behkn,26,7)} +$ $G_{(behkn,22,11)} + F_{(behkn,22,12)} +$ $F_{(behkn,22,10)} + G_{(behkn,14,14)} +$ $F_{(behkn,14,15)} + F_{(behkn,14,13)}$	0
$F_{(behkn,30,6)}$	<i>base case</i>	0
$F_{(behkn,30,6)}$	<i>base case</i>	0
$G_{(behkn,29,8)}$	<i>base case</i>	0
$F_{(behkn,29,9)}$	<i>base case</i>	0
$F_{(behkn,29,7)}$	<i>base case</i>	0
$G_{(behkn,27,11)}$	<i>base case</i>	0
$F_{(behkn,27,12)}$	<i>base case</i>	0
$F_{(behkn,27,10)}$	<i>base case</i>	0
$G_{(behkn,23,14)}$	<i>base case</i>	0
$F_{(behkn,23,15)}$	<i>base case</i>	0
$F_{(behkn,23,13)}$	<i>base case</i>	0
$G_{(behkn,15,17)}$	<i>base case</i>	0

Tabel D.11 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist* = 0 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (3)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,15,18)}$	<i>base case</i>	0
$F_{(behkn,15,16)}$	<i>base case</i>	0
$G_{(behkn,31,5)}$	$G_{(behkn,30,5)} + F_{(behkn,30,6)} +$ $F_{(behkn,30,6)} + G_{(behkn,29,8)} +$ $F_{(behkn,29,9)} + F_{(behkn,29,7)} +$ $G_{(behkn,27,11)} + F_{(behkn,27,12)} +$ $F_{(behkn,27,10)} + G_{(behkn,23,14)} +$ $F_{(behkn,23,15)} + F_{(behkn,23,13)} +$ $G_{(behkn,15,17)} + F_{(behkn,15,18)} +$ $F_{(behkn,15,16)}$	0

Tabel D.12 Simulasi perhitungan jumlah kombinasi *string orig2* tanpa operasi *replace* dengan *dist* = 0 pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan *X* = 5 (1)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,0,0)}$	<i>base case</i>	0
$F_{(behkn,16,0)}$	$F_{(behkn,0,0)}$	0
$F_{(behkn,0,6)}$	<i>base case</i>	0
$F_{(behkn,8,3)}$	$F_{(behkn,0,6)}$	0
$F_{(behkn,24,0)}$	$F_{(behkn,16,0)} + F_{(behkn,8,3)}$	0
$F_{(behkn,16,6)}$	<i>base case</i>	0
$F_{(behkn,4,6)}$	<i>base case</i>	0
$F_{(behkn,20,3)}$	$F_{(behkn,16,6)} + F_{(behkn,4,6)}$	0
$F_{(behkn,12,6)}$	<i>base case</i>	0
$F_{(behkn,28,0)}$	$F_{(behkn,24,0)} + F_{(behkn,20,3)} + F_{(behkn,12,6)}$	0
$F_{(behkn,24,6)}$	<i>base case</i>	0
$F_{(behkn,18,6)}$	<i>base case</i>	0
$F_{(behkn,10,9)}$	<i>base case</i>	0
$F_{(behkn,26,3)}$	$F_{(behkn,24,6)} + F_{(behkn,18,6)} + F_{(behkn,10,9)}$	0
$F_{(behkn,22,6)}$	<i>base case</i>	0
$F_{(behkn,14,9)}$	<i>base case</i>	0
$F_{(behkn,30,0)}$	$F_{(behkn,28,0)} + F_{(behkn,26,3)} + F_{(behkn,22,6)} + F_{(behkn,14,9)}$	0
$F_{(behkn,28,6)}$	<i>base case</i>	0
$F_{(behkn,25,6)}$	<i>base case</i>	0
$F_{(behkn,21,9)}$	<i>base case</i>	0
$F_{(behkn,13,12)}$	<i>base case</i>	0
$F_{(behkn,29,3)}$	$F_{(behkn,28,6)} + F_{(behkn,25,6)} + F_{(behkn,21,9)} + F_{(behkn,13,12)}$	0

Tabel D.13 Simulasi perhitungan jumlah kombinasi *string orig2* tanpa operasi *replace* dengan *dist* = 0 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (2)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,27,6)}$	<i>base case</i>	0
$F_{(behkn,23,9)}$	<i>base case</i>	0
$F_{(behkn,15,12)}$	<i>base case</i>	0
$F_{(behkn,31,0)}$	$F_{(behkn,30,0)} + F_{(behkn,29,3)} +$ $F_{(behkn,27,6)} + F_{(behkn,23,9)} +$ $F_{(behkn,15,12)}$	0

Tabel D.14 Simulasi perhitungan jumlah kombinasi *string orig1* tanpa operasi *replace* dengan *dist* = 1 pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan *X* = 5

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,0,4)}$	<i>base case</i>	0
$F_{(behkn,16,4)}$	$F_{(behkn,0,4)}$	0
$F_{(behkn,8,7)}$	<i>base case</i>	0
$F_{(behkn,24,4)}$	$F_{(behkn,16,4)} + F_{(behkn,8,7)}$	0
$F_{(behkn,20,7)}$	<i>base case</i>	0
$F_{(behkn,12,10)}$	<i>base case</i>	0
$F_{(behkn,28,4)}$	$F_{(behkn,24,4)} + F_{(behkn,20,7)} + F_{(behkn,12,10)}$	0
$F_{(behkn,26,7)}$	<i>base case</i>	0
$F_{(behkn,22,10)}$	<i>base case</i>	0
$F_{(behkn,14,13)}$	<i>base case</i>	0
$F_{(behkn,30,4)}$	$F_{(behkn,28,4)} + F_{(behkn,26,7)} + F_{(behkn,22,10)} + F_{(behkn,14,13)}$	0
$F_{(behkn,29,7)}$	<i>base case</i>	0
$F_{(behkn,27,10)}$	<i>base case</i>	0
$F_{(behkn,23,13)}$	<i>base case</i>	0
$F_{(behkn,15,16)}$	<i>base case</i>	0
$F_{(behkn,31,4)}$	$F_{(behkn,30,4)} + F_{(behkn,29,7)} + F_{(behkn,27,10)} + F_{(behkn,23,13)} + F_{(behkn,15,16)}$	0

Tabel D.15 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist* = 1 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (1)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,0,1)}$	<i>base case</i>	0
$F_{(behkn,0,2)}$	<i>base case</i>	0
$F_{(behkn,0,2)}$	<i>base case</i>	0
$G_{(behkn,16,1)}$	$G_{(behkn,0,1)} + F_{(behkn,0,2)} + F_{(behkn,0,2)}$	0
$F_{(behkn,0,2)}$	<i>base case</i>	0
$F_{(behkn,16,2)}$	$F_{(behkn,0,2)}$	0
$F_{(behkn,16,2)}$	<i>memoF</i> $_{(behkn,16,2)}$	0
$G_{(behkn,0,7)}$	<i>base case</i>	0
$F_{(behkn,0,6)}$	<i>base case</i>	0
$F_{(behkn,0,8)}$	<i>base case</i>	0
$G_{(behkn,8,4)}$	$G_{(behkn,0,7)} + F_{(behkn,0,6)} + F_{(behkn,0,8)}$	0
$F_{(behkn,0,8)}$	<i>base case</i>	0
$F_{(behkn,8,5)}$	$F_{(behkn,0,8)}$	0
$F_{(behkn,8,3)}$	<i>memoF</i> $_{(behkn,8,3)}$	0
$G_{(behkn,24,1)}$	$G_{(behkn,16,1)} + F_{(behkn,16,2)} + F_{(behkn,16,2)} + G_{(behkn,8,4)} + F_{(behkn,8,5)} + F_{(behkn,8,3)}$	0
$F_{(behkn,16,2)}$	<i>memoF</i> $_{(behkn,16,2)}$	0
$F_{(behkn,8,5)}$	<i>memoF</i> $_{(behkn,8,5)}$	0
$F_{(behkn,24,2)}$	$F_{(behkn,16,2)} + F_{(behkn,8,5)}$	0
$F_{(behkn,24,2)}$	<i>memoF</i> $_{(behkn,24,2)}$	0
$G_{(behkn,16,7)}$	<i>base case</i>	0
$F_{(behkn,16,6)}$	<i>base case</i>	0
$F_{(behkn,16,8)}$	<i>base case</i>	0

Tabel D.16 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist = 1* pada kasus *string ad1 = kbenh, string ad2 = kbenh* dan *X = 5 (2)*

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,4,7)}$	<i>base case</i>	0
$F_{(behkn,4,8)}$	<i>base case</i>	0
$F_{(behkn,4,6)}$	<i>base case</i>	0
$G_{(behkn,20,4)}$	$G_{(behkn,16,7)} + F_{(behkn,16,6)} + F_{(behkn,16,8)} + G_{(behkn,4,7)} + F_{(behkn,4,8)} + F_{(behkn,4,6)}$	0
$F_{(behkn,16,8)}$	<i>base case</i>	0
$F_{(behkn,4,8)}$	<i>base case</i>	0
$F_{(behkn,20,5)}$	$F_{(behkn,16,8)} + F_{(behkn,4,8)}$	0
$F_{(behkn,20,3)}$	<i>memoF</i> $_{(behkn,20,3)}$	0
$G_{(behkn,12,7)}$	<i>base case</i>	0
$F_{(behkn,12,8)}$	<i>base case</i>	0
$F_{(behkn,12,6)}$	<i>base case</i>	0
$G_{(behkn,28,1)}$	$G_{(behkn,24,1)} + F_{(behkn,24,2)} + F_{(behkn,24,2)} + G_{(behkn,20,4)} + F_{(behkn,20,5)} + F_{(behkn,20,3)} + G_{(behkn,12,7)} + F_{(behkn,12,8)} + F_{(behkn,12,6)}$	0
$F_{(behkn,24,2)}$	<i>memoF</i> $_{(behkn,24,2)}$	0
$F_{(behkn,20,5)}$	<i>memoF</i> $_{(behkn,20,5)}$	0
$F_{(behkn,12,8)}$	<i>base case</i>	0
$F_{(behkn,28,2)}$	$F_{(behkn,24,2)} + F_{(behkn,20,5)} + F_{(behkn,12,8)}$	0
$F_{(behkn,28,2)}$	<i>memoF</i> $_{(behkn,28,2)}$	0
$G_{(behkn,24,7)}$	<i>base case</i>	0
$F_{(behkn,24,6)}$	<i>base case</i>	0
$F_{(behkn,24,8)}$	<i>base case</i>	0
$G_{(behkn,18,7)}$	<i>base case</i>	0

Tabel D.17 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist = 1* pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan *X = 5 (3)*

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,18,8)}$	<i>base case</i>	0
$F_{(behkn,18,6)}$	<i>base case</i>	0
$G_{(behkn,10,10)}$	<i>base case</i>	0
$F_{(behkn,10,11)}$	<i>base case</i>	0
$F_{(behkn,10,9)}$	<i>base case</i>	0
$G_{(behkn,26,4)}$	$G_{(behkn,24,7)} + F_{(behkn,24,6)} +$ $F_{(behkn,24,8)} + G_{(behkn,18,7)} +$ $F_{(behkn,18,8)} + F_{(behkn,18,6)} +$ $G_{(behkn,10,10)} + F_{(behkn,10,11)} +$ $F_{(behkn,10,9)}$	0
$F_{(behkn,24,8)}$	<i>base case</i>	0
$F_{(behkn,18,8)}$	<i>base case</i>	0
$F_{(behkn,10,11)}$	<i>base case</i>	0
$F_{(behkn,26,5)}$	$F_{(behkn,24,8)} + F_{(behkn,18,8)} +$ $F_{(behkn,10,11)}$	0
$F_{(behkn,26,3)}$	<i>memoF_(behkn,26,3)</i>	0
$G_{(behkn,22,7)}$	<i>base case</i>	0
$F_{(behkn,22,8)}$	<i>base case</i>	0
$F_{(behkn,22,6)}$	<i>base case</i>	0
$G_{(behkn,14,10)}$	<i>base case</i>	0
$F_{(behkn,14,11)}$	<i>base case</i>	0
$F_{(behkn,14,9)}$	<i>base case</i>	0
$G_{(behkn,30,1)}$	$G_{(behkn,28,1)} + F_{(behkn,28,2)} +$ $F_{(behkn,28,2)} + G_{(behkn,26,4)} +$ $F_{(behkn,26,5)} + F_{(behkn,26,3)} +$ $G_{(behkn,22,7)} + F_{(behkn,22,8)} +$ $F_{(behkn,22,6)} + G_{(behkn,14,10)} +$ $F_{(behkn,14,11)} + F_{(behkn,14,9)}$	0

Tabel D.18 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist* = 1 pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan *X* = 5 (4)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,28,2)}$	$memoF_{(behkn,28,2)}$	0
$F_{(behkn,26,5)}$	$memoF_{(behkn,26,5)}$	0
$F_{(behkn,22,8)}$	<i>base case</i>	0
$F_{(behkn,14,11)}$	<i>base case</i>	0
$F_{(behkn,30,2)}$	$F_{(behkn,28,2)} + F_{(behkn,26,5)} + F_{(behkn,22,8)} + F_{(behkn,14,11)}$	0
$F_{(behkn,30,2)}$	$memoF_{(behkn,30,2)}$	0
$G_{(behkn,28,7)}$	<i>base case</i>	0
$F_{(behkn,28,6)}$	<i>base case</i>	0
$F_{(behkn,28,8)}$	<i>base case</i>	0
$G_{(behkn,25,7)}$	<i>base case</i>	0
$F_{(behkn,25,8)}$	<i>base case</i>	0
$F_{(behkn,25,6)}$	<i>base case</i>	0
$G_{(behkn,21,10)}$	<i>base case</i>	0
$F_{(behkn,21,11)}$	<i>base case</i>	0
$F_{(behkn,21,9)}$	<i>base case</i>	0
$G_{(behkn,13,13)}$	<i>base case</i>	0
$F_{(behkn,13,14)}$	<i>base case</i>	0
$F_{(behkn,13,12)}$	<i>base case</i>	0
$G_{(behkn,29,4)}$	$G_{(behkn,28,7)} + F_{(behkn,28,6)} + F_{(behkn,28,8)} + G_{(behkn,25,7)} + F_{(behkn,25,8)} + F_{(behkn,25,6)} + G_{(behkn,21,10)} + F_{(behkn,21,11)} + F_{(behkn,21,9)} + G_{(behkn,13,13)} + F_{(behkn,13,14)} + F_{(behkn,13,12)}$	0
$F_{(behkn,28,8)}$	<i>base case</i>	0
$F_{(behkn,25,8)}$	<i>base case</i>	0
$F_{(behkn,21,11)}$	<i>base case</i>	0

Tabel D.19 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist* = 1 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (5)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,13,14)}$	<i>base case</i>	0
$F_{(behkn,29,5)}$	$F_{(behkn,28,8)} + F_{(behkn,25,8)} + F_{(behkn,21,11)} + F_{(behkn,13,14)}$	0
$F_{(behkn,29,3)}$	<i>memo</i> $F_{(behkn,29,3)}$	0
$G_{(behkn,27,7)}$	<i>base case</i>	0
$F_{(behkn,27,8)}$	<i>base case</i>	0
$F_{(behkn,27,6)}$	<i>base case</i>	0
$G_{(behkn,23,10)}$	<i>base case</i>	0
$F_{(behkn,23,11)}$	<i>base case</i>	0
$F_{(behkn,23,9)}$	<i>base case</i>	0
$G_{(behkn,15,13)}$	<i>base case</i>	0
$F_{(behkn,15,14)}$	<i>base case</i>	0
$F_{(behkn,15,12)}$	<i>base case</i>	0
$G_{(behkn,31,1)}$	$G_{(behkn,30,1)} + F_{(behkn,30,2)} + F_{(behkn,30,2)} + G_{(behkn,29,4)} + F_{(behkn,29,5)} + F_{(behkn,29,3)} + G_{(behkn,27,7)} + F_{(behkn,27,8)} + F_{(behkn,27,6)} + G_{(behkn,23,10)} + F_{(behkn,23,11)} + F_{(behkn,23,9)} + G_{(behkn,15,13)} + F_{(behkn,15,14)} + F_{(behkn,15,12)}$	0

Tabel D.20 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist = 1* pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan *X = 5 (1)*

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,0,4)}$	<i>base case</i>	0
$F_{(behkn,0,5)}$	<i>base case</i>	1
$F_{(behkn,0,5)}$	<i>base case</i>	1
$G_{(behkn,16,4)}$	$G_{(behkn,0,4)} + F_{(behkn,0,5)} + F_{(behkn,0,5)}$	2
$F_{(behkn,16,5)}$	$memoF_{(behkn,16,5)}$	1
$F_{(behkn,16,5)}$	$memoF_{(behkn,16,5)}$	1
$G_{(behkn,8,7)}$	<i>base case</i>	0
$F_{(behkn,8,8)}$	<i>base case</i>	0
$F_{(behkn,8,6)}$	<i>base case</i>	0
$G_{(behkn,24,4)}$	$G_{(behkn,16,4)} + F_{(behkn,16,5)} + F_{(behkn,16,5)} + G_{(behkn,8,7)} + F_{(behkn,8,8)} + F_{(behkn,8,6)}$	4
$F_{(behkn,24,5)}$	$memoF_{(behkn,24,5)}$	1
$F_{(behkn,24,5)}$	$memoF_{(behkn,24,5)}$	1
$G_{(behkn,20,7)}$	<i>base case</i>	0
$F_{(behkn,20,8)}$	<i>base case</i>	0
$F_{(behkn,20,6)}$	<i>base case</i>	0
$G_{(behkn,12,10)}$	<i>base case</i>	0
$F_{(behkn,12,11)}$	<i>base case</i>	0
$F_{(behkn,12,9)}$	<i>base case</i>	0
$G_{(behkn,28,4)}$	$G_{(behkn,24,4)} + F_{(behkn,24,5)} + F_{(behkn,24,5)} + G_{(behkn,20,7)} + F_{(behkn,20,8)} + F_{(behkn,20,6)} + G_{(behkn,12,10)} + F_{(behkn,12,11)} + F_{(behkn,12,9)}$	6
$F_{(behkn,28,5)}$	$memoF_{(behkn,28,5)}$	1
$F_{(behkn,28,5)}$	$memoF_{(behkn,28,5)}$	1

Tabel D.21 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist* = 1 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (2)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,26,7)}$	<i>base case</i>	0
$F_{(behkn,26,8)}$	<i>base case</i>	0
$F_{(behkn,26,6)}$	<i>base case</i>	0
$G_{(behkn,22,10)}$	<i>base case</i>	0
$F_{(behkn,22,11)}$	<i>base case</i>	0
$F_{(behkn,22,9)}$	<i>base case</i>	0
$G_{(behkn,14,13)}$	<i>base case</i>	0
$F_{(behkn,14,14)}$	<i>base case</i>	0
$F_{(behkn,14,12)}$	<i>base case</i>	0
$G_{(behkn,30,4)}$	$G_{(behkn,28,4)} + F_{(behkn,28,5)} +$ $F_{(behkn,28,5)} + G_{(behkn,26,7)} +$ $F_{(behkn,26,8)} + F_{(behkn,26,6)} +$ $G_{(behkn,22,10)} + F_{(behkn,22,11)} +$ $F_{(behkn,22,9)} + G_{(behkn,14,13)} +$ $F_{(behkn,14,14)} + F_{(behkn,14,12)}$	8
$F_{(behkn,30,5)}$	<i>memoF</i> $_{(behkn,30,5)}$	1
$F_{(behkn,30,5)}$	<i>memoF</i> $_{(behkn,30,5)}$	1
$G_{(behkn,29,7)}$	<i>base case</i>	0
$F_{(behkn,29,8)}$	<i>base case</i>	0
$F_{(behkn,29,6)}$	<i>base case</i>	0
$G_{(behkn,27,10)}$	<i>base case</i>	0
$F_{(behkn,27,11)}$	<i>base case</i>	0
$F_{(behkn,27,9)}$	<i>base case</i>	0
$G_{(behkn,23,13)}$	<i>base case</i>	0
$F_{(behkn,23,14)}$	<i>base case</i>	0
$F_{(behkn,23,12)}$	<i>base case</i>	0
$G_{(behkn,15,16)}$	<i>base case</i>	0
$F_{(behkn,15,17)}$	<i>base case</i>	0

Tabel D.22 Simulasi perhitungan jumlah kombinasi $string\ orig1$ dengan operasi $replace$ dengan $dist = 1$ pada kasus $string\ ad1 = kbenh$, $string\ ad2 = kbenh$ dan $X = 5$ (3)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,15,15)}$	<i>base case</i>	0
$G_{(behkn,31,4)}$	$G_{(behkn,30,4)} + F_{(behkn,30,5)} +$ $F_{(behkn,30,5)} + G_{(behkn,29,7)} +$ $F_{(behkn,29,8)} + F_{(behkn,29,6)} +$ $G_{(behkn,27,10)} + F_{(behkn,27,11)} +$ $F_{(behkn,27,9)} + G_{(behkn,23,13)} +$ $F_{(behkn,23,14)} + F_{(behkn,23,12)} +$ $G_{(behkn,15,16)} + F_{(behkn,15,17)} +$ $F_{(behkn,15,15)}$	10

Tabel D.23 Simulasi perhitungan jumlah kombinasi $string\ orig2$ tanpa operasi $replace$ dengan $dist = 1$ pada kasus $string\ ad1 = kbenh$, $string\ ad2 = kbenh$ dan $X = 5$

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,30,1)}$	$memoF_{(behkn,30,1)}$	0
$F_{(behkn,29,4)}$	$memoF_{(behkn,29,4)}$	0
$F_{(behkn,27,7)}$	<i>base case</i>	0
$F_{(behkn,23,10)}$	<i>base case</i>	0
$F_{(behkn,15,13)}$	<i>base case</i>	0
$F_{(behkn,31,1)}$	$F_{(behkn,30,1)} + F_{(behkn,29,4)} +$ $F_{(behkn,27,7)} + F_{(behkn,23,10)} +$ $F_{(behkn,15,13)}$	0

Tabel D.24 Simulasi perhitungan jumlah kombinasi *string orig1* tanpa operasi *replace* dengan *dist* = 2 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,0,3)}$	<i>base case</i>	0
$F_{(behkn,16,3)}$	$F_{(behkn,0,3)}$	0
$F_{(behkn,8,6)}$	<i>base case</i>	0
$F_{(behkn,24,3)}$	$F_{(behkn,16,3)} + F_{(behkn,8,6)}$	0
$F_{(behkn,20,6)}$	<i>base case</i>	0
$F_{(behkn,12,9)}$	<i>base case</i>	0
$F_{(behkn,28,3)}$	$F_{(behkn,24,3)} + F_{(behkn,20,6)} + F_{(behkn,12,9)}$	0
$F_{(behkn,26,6)}$	<i>base case</i>	0
$F_{(behkn,22,9)}$	<i>base case</i>	0
$F_{(behkn,14,12)}$	<i>base case</i>	0
$F_{(behkn,30,3)}$	$F_{(behkn,28,3)} + F_{(behkn,26,6)} + F_{(behkn,22,9)} + F_{(behkn,14,12)}$	0
$F_{(behkn,29,6)}$	<i>base case</i>	0
$F_{(behkn,27,9)}$	<i>base case</i>	0
$F_{(behkn,23,12)}$	<i>base case</i>	0
$F_{(behkn,15,15)}$	<i>base case</i>	0
$F_{(behkn,31,3)}$	$F_{(behkn,30,3)} + F_{(behkn,29,6)} + F_{(behkn,27,9)} + F_{(behkn,23,12)} + F_{(behkn,15,15)}$	0

Tabel D.25 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist* = 2 pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan *X* = 5 (1)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,0,2)}$	<i>base case</i>	0
$F_{(behkn,0,3)}$	<i>base case</i>	0
$F_{(behkn,0,3)}$	<i>base case</i>	0
$G_{(behkn,16,2)}$	$G_{(behkn,0,2)} + F_{(behkn,0,3)} + F_{(behkn,0,3)}$	0
$F_{(behkn,0,3)}$	<i>base case</i>	0
$F_{(behkn,16,3)}$	$F_{(behkn,0,3)}$	0
$F_{(behkn,16,3)}$	<i>memoF</i> $_{(behkn,16,3)}$	0
$G_{(behkn,0,8)}$	<i>base case</i>	0
$F_{(behkn,0,7)}$	<i>base case</i>	0
$F_{(behkn,0,9)}$	<i>base case</i>	0
$G_{(behkn,8,5)}$	$G_{(behkn,0,8)} + F_{(behkn,0,7)} + F_{(behkn,0,9)}$	0
$F_{(behkn,8,6)}$	<i>base case</i>	0
$F_{(behkn,8,4)}$	<i>memoF</i> $_{(behkn,8,4)}$	0
$G_{(behkn,24,2)}$	$G_{(behkn,16,2)} + F_{(behkn,16,3)} + F_{(behkn,16,3)} + G_{(behkn,8,5)} + F_{(behkn,8,6)} + F_{(behkn,8,4)}$	0
$F_{(behkn,16,3)}$	<i>memoF</i> $_{(behkn,16,3)}$	0
$F_{(behkn,8,6)}$	<i>base case</i>	0
$F_{(behkn,24,3)}$	$F_{(behkn,16,3)} + F_{(behkn,8,6)}$	0
$F_{(behkn,24,3)}$	<i>memoF</i> $_{(behkn,24,3)}$	0
$G_{(behkn,16,8)}$	<i>base case</i>	0
$F_{(behkn,16,7)}$	<i>base case</i>	0
$F_{(behkn,16,9)}$	<i>base case</i>	0
$G_{(behkn,4,8)}$	<i>base case</i>	0

Tabel D.26 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist = 2* pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan *X = 5* (2)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,4,9)}$	<i>base case</i>	0
$F_{(behkn,4,7)}$	<i>base case</i>	0
$G_{(behkn,20,5)}$	$G_{(behkn,16,8)} + F_{(behkn,16,7)} + F_{(behkn,16,9)} + G_{(behkn,4,8)} + F_{(behkn,4,9)} + F_{(behkn,4,7)}$	0
$F_{(behkn,20,6)}$	<i>base case</i>	0
$F_{(behkn,20,4)}$	<i>memo</i> $F_{(behkn,20,4)}$	0
$G_{(behkn,12,8)}$	<i>base case</i>	0
$F_{(behkn,12,9)}$	<i>base case</i>	0
$F_{(behkn,12,7)}$	<i>base case</i>	0
$G_{(behkn,28,2)}$	$G_{(behkn,24,2)} + F_{(behkn,24,3)} + F_{(behkn,24,3)} + G_{(behkn,20,5)} + F_{(behkn,20,6)} + F_{(behkn,20,4)} + G_{(behkn,12,8)} + F_{(behkn,12,9)} + F_{(behkn,12,7)}$	0
$F_{(behkn,24,3)}$	<i>memo</i> $F_{(behkn,24,3)}$	0
$F_{(behkn,20,6)}$	<i>base case</i>	0
$F_{(behkn,12,9)}$	<i>base case</i>	0
$F_{(behkn,28,3)}$	$F_{(behkn,24,3)} + F_{(behkn,20,6)} + F_{(behkn,12,9)}$	0
$F_{(behkn,28,3)}$	<i>memo</i> $F_{(behkn,28,3)}$	0
$G_{(behkn,24,8)}$	<i>base case</i>	0
$F_{(behkn,24,7)}$	<i>base case</i>	0
$F_{(behkn,24,9)}$	<i>base case</i>	0
$G_{(behkn,18,8)}$	<i>base case</i>	0
$F_{(behkn,18,9)}$	<i>base case</i>	0
$F_{(behkn,18,7)}$	<i>base case</i>	0

Tabel D.27 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist = 2* pada kasus *string ad1 = kbenh, string ad2 = kbenh* dan *X = 5 (3)*

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,10,11)}$	<i>base case</i>	0
$F_{(behkn,10,12)}$	<i>base case</i>	0
$F_{(behkn,10,10)}$	<i>base case</i>	0
$G_{(behkn,26,5)}$	$G_{(behkn,24,8)} + F_{(behkn,24,7)} +$ $F_{(behkn,24,9)} + G_{(behkn,18,8)} +$ $F_{(behkn,18,9)} + F_{(behkn,18,7)} +$ $G_{(behkn,10,11)} + F_{(behkn,10,12)} +$ $F_{(behkn,10,10)}$	0
$F_{(behkn,26,6)}$	<i>base case</i>	0
$F_{(behkn,26,4)}$	$memoF_{(behkn,26,4)}$	0
$G_{(behkn,22,8)}$	<i>base case</i>	0
$F_{(behkn,22,9)}$	<i>base case</i>	0
$F_{(behkn,22,7)}$	<i>base case</i>	0
$G_{(behkn,14,11)}$	<i>base case</i>	0
$F_{(behkn,14,12)}$	<i>base case</i>	0
$F_{(behkn,14,10)}$	<i>base case</i>	0
$G_{(behkn,30,2)}$	$G_{(behkn,28,2)} + F_{(behkn,28,3)} +$ $F_{(behkn,28,3)} + G_{(behkn,26,5)} +$ $F_{(behkn,26,6)} + F_{(behkn,26,4)} +$ $G_{(behkn,22,8)} + F_{(behkn,22,9)} +$ $F_{(behkn,22,7)} + G_{(behkn,14,11)} +$ $F_{(behkn,14,12)} + F_{(behkn,14,10)}$	0
$F_{(behkn,28,3)}$	$memoF_{(behkn,28,3)}$	0
$F_{(behkn,26,6)}$	<i>base case</i>	0
$F_{(behkn,22,9)}$	<i>base case</i>	0
$F_{(behkn,14,12)}$	<i>base case</i>	0
$F_{(behkn,30,3)}$	$F_{(behkn,28,3)} + F_{(behkn,26,6)} +$ $F_{(behkn,22,9)} + F_{(behkn,14,12)}$	0

Tabel D.28 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist* = 2 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (4)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,30,3)}$	$memoF_{(behkn,30,3)}$	0
$G_{(behkn,28,8)}$	<i>base case</i>	0
$F_{(behkn,28,7)}$	<i>base case</i>	0
$F_{(behkn,28,9)}$	<i>base case</i>	0
$G_{(behkn,25,8)}$	<i>base case</i>	0
$F_{(behkn,25,9)}$	<i>base case</i>	0
$F_{(behkn,25,7)}$	<i>base case</i>	0
$G_{(behkn,21,11)}$	<i>base case</i>	0
$F_{(behkn,21,12)}$	<i>base case</i>	0
$F_{(behkn,21,10)}$	<i>base case</i>	0
$G_{(behkn,13,14)}$	<i>base case</i>	0
$F_{(behkn,13,15)}$	<i>base case</i>	0
$F_{(behkn,13,13)}$	<i>base case</i>	0
$G_{(behkn,29,5)}$	$G_{(behkn,28,8)} + F_{(behkn,28,7)} +$ $F_{(behkn,28,9)} + G_{(behkn,25,8)} +$ $F_{(behkn,25,9)} + F_{(behkn,25,7)} +$ $G_{(behkn,21,11)} + F_{(behkn,21,12)} +$ $F_{(behkn,21,10)} + G_{(behkn,13,14)} +$ $F_{(behkn,13,15)} + F_{(behkn,13,13)}$	0
$F_{(behkn,29,6)}$	<i>base case</i>	0
$F_{(behkn,29,4)}$	$memoF_{(behkn,29,4)}$	0
$G_{(behkn,27,8)}$	<i>base case</i>	0
$F_{(behkn,27,9)}$	<i>base case</i>	0
$F_{(behkn,27,7)}$	<i>base case</i>	0
$G_{(behkn,23,11)}$	<i>base case</i>	0
$F_{(behkn,23,12)}$	<i>base case</i>	0
$F_{(behkn,23,10)}$	<i>base case</i>	0
$G_{(behkn,15,14)}$	<i>base case</i>	0

Tabel D.29 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist* = 2 pada kasus *string ad1 = kbenh, string ad2 = kbenh* dan *X* = 5 (5)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,15,15)}$	<i>base case</i>	0
$F_{(behkn,15,13)}$	<i>base case</i>	0
$G_{(behkn,31,2)}$	$G_{(behkn,30,2)} + F_{(behkn,30,3)} +$ $F_{(behkn,30,3)} + G_{(behkn,29,5)} +$ $F_{(behkn,29,6)} + F_{(behkn,29,4)} +$ $G_{(behkn,27,8)} + F_{(behkn,27,9)} +$ $F_{(behkn,27,7)} + G_{(behkn,23,11)} +$ $F_{(behkn,23,12)} + F_{(behkn,23,10)} +$ $G_{(behkn,15,14)} + F_{(behkn,15,15)} +$ $F_{(behkn,15,13)}$	0

Tabel D.30 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist = 2* pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan *X = 5 (1)*

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,0,3)}$	<i>base case</i>	0
$F_{(behkn,0,4)}$	<i>base case</i>	0
$F_{(behkn,0,4)}$	<i>base case</i>	0
$G_{(behkn,16,3)}$	$G_{(behkn,0,3)} + F_{(behkn,0,4)} + F_{(behkn,0,4)}$	0
$F_{(behkn,16,4)}$	<i>memoF_(behkn,16,4)</i>	0
$F_{(behkn,16,4)}$	<i>memoF_(behkn,16,4)</i>	0
$G_{(behkn,8,6)}$	<i>base case</i>	0
$F_{(behkn,8,7)}$	<i>base case</i>	0
$F_{(behkn,0,8)}$	<i>base case</i>	0
$F_{(behkn,8,5)}$	$F_{(behkn,0,8)}$	0
$G_{(behkn,24,3)}$	$G_{(behkn,16,3)} + F_{(behkn,16,4)} + F_{(behkn,16,4)} + G_{(behkn,8,6)} + F_{(behkn,8,7)} + F_{(behkn,8,5)}$	0
$F_{(behkn,24,4)}$	<i>memoF_(behkn,24,4)</i>	0
$F_{(behkn,24,4)}$	<i>memoF_(behkn,24,4)</i>	0
$G_{(behkn,20,6)}$	<i>base case</i>	0
$F_{(behkn,20,7)}$	<i>base case</i>	0
$F_{(behkn,16,8)}$	<i>base case</i>	0
$F_{(behkn,4,8)}$	<i>base case</i>	0
$F_{(behkn,20,5)}$	$F_{(behkn,16,8)} + F_{(behkn,4,8)}$	0
$G_{(behkn,12,9)}$	<i>base case</i>	0
$F_{(behkn,12,10)}$	<i>base case</i>	0
$F_{(behkn,12,8)}$	<i>base case</i>	0
$G_{(behkn,28,3)}$	$G_{(behkn,24,3)} + F_{(behkn,24,4)} + F_{(behkn,24,4)} + G_{(behkn,20,6)} + F_{(behkn,20,7)} + F_{(behkn,20,5)} + G_{(behkn,12,9)} + F_{(behkn,12,10)} + F_{(behkn,12,8)}$	0

Tabel D.31 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist* = 2 pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan *X* = 5 (2)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,28,4)}$	$memoF_{(behkn,28,4)}$	0
$F_{(behkn,28,4)}$	$memoF_{(behkn,28,4)}$	0
$G_{(behkn,26,6)}$	<i>base case</i>	0
$F_{(behkn,26,7)}$	<i>base case</i>	0
$F_{(behkn,24,8)}$	<i>base case</i>	0
$F_{(behkn,18,8)}$	<i>base case</i>	0
$F_{(behkn,10,11)}$	<i>base case</i>	0
$F_{(behkn,26,5)}$	$F_{(behkn,24,8)} + F_{(behkn,18,8)} + F_{(behkn,10,11)}$	0
$G_{(behkn,22,9)}$	<i>base case</i>	0
$F_{(behkn,22,10)}$	<i>base case</i>	0
$F_{(behkn,22,8)}$	<i>base case</i>	0
$G_{(behkn,14,12)}$	<i>base case</i>	0
$F_{(behkn,14,13)}$	<i>base case</i>	0
$F_{(behkn,14,11)}$	<i>base case</i>	0
$G_{(behkn,30,3)}$	$G_{(behkn,28,3)} + F_{(behkn,28,4)} + F_{(behkn,28,4)} + G_{(behkn,26,6)} + F_{(behkn,26,7)} + F_{(behkn,26,5)} + G_{(behkn,22,9)} + F_{(behkn,22,10)} + F_{(behkn,22,8)} + G_{(behkn,14,12)} + F_{(behkn,14,13)} + F_{(behkn,14,11)}$	0
$F_{(behkn,30,4)}$	$memoF_{(behkn,30,4)}$	0
$F_{(behkn,30,4)}$	$memoF_{(behkn,30,4)}$	0
$G_{(behkn,29,6)}$	<i>base case</i>	0
$F_{(behkn,29,7)}$	<i>base case</i>	0
$F_{(behkn,28,8)}$	<i>base case</i>	0
$F_{(behkn,25,8)}$	<i>base case</i>	0

Tabel D.32 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist* = 2 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (3)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,21,11)}$	<i>base case</i>	0
$F_{(behkn,13,14)}$	<i>base case</i>	0
$F_{(behkn,29,5)}$	$F_{(behkn,28,8)} + F_{(behkn,25,8)} + F_{(behkn,21,11)} + F_{(behkn,13,14)}$	0
$G_{(behkn,27,9)}$	<i>base case</i>	0
$F_{(behkn,27,10)}$	<i>base case</i>	0
$F_{(behkn,27,8)}$	<i>base case</i>	0
$G_{(behkn,23,12)}$	<i>base case</i>	0
$F_{(behkn,23,13)}$	<i>base case</i>	0
$F_{(behkn,23,11)}$	<i>base case</i>	0
$G_{(behkn,15,15)}$	<i>base case</i>	0
$F_{(behkn,15,16)}$	<i>base case</i>	0
$F_{(behkn,15,14)}$	<i>base case</i>	0
$G_{(behkn,31,3)}$	$G_{(behkn,30,3)} + F_{(behkn,30,4)} + F_{(behkn,30,4)} + G_{(behkn,29,6)} + F_{(behkn,29,7)} + F_{(behkn,29,5)} + G_{(behkn,27,9)} + F_{(behkn,27,10)} + F_{(behkn,27,8)} + G_{(behkn,23,12)} + F_{(behkn,23,13)} + F_{(behkn,23,11)} + G_{(behkn,15,15)} + F_{(behkn,15,16)} + F_{(behkn,15,14)}$	0

Tabel D.33 Simulasi perhitungan jumlah kombinasi *string orig2* tanpa operasi *replace* dengan *dist* = 2 pada kasus *string ad1 = kbenh, string ad2 = kbenh* dan *X* = 5

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,30,2)}$	$memoF_{(behkn,30,2)}$	0
$F_{(behkn,29,5)}$	$memoF_{(behkn,29,5)}$	0
$F_{(behkn,27,8)}$	<i>base case</i>	0
$F_{(behkn,23,11)}$	<i>base case</i>	0
$F_{(behkn,15,14)}$	<i>base case</i>	0
$F_{(behkn,31,2)}$	$F_{(behkn,30,2)} + F_{(behkn,29,5)} + F_{(behkn,27,8)} + F_{(behkn,23,11)} + F_{(behkn,15,14)}$	0

Tabel D.34 Simulasi perhitungan jumlah kombinasi *string orig1* tanpa operasi *replace* dengan *dist* = 3 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,0,2)}$	<i>base case</i>	0
$F_{(behkn,16,2)}$	$F_{(behkn,0,2)}$	0
$F_{(behkn,8,5)}$	<i>memo</i> $F_{(behkn,8,5)}$	0
$F_{(behkn,24,2)}$	$F_{(behkn,16,2)} + F_{(behkn,8,5)}$	0
$F_{(behkn,20,5)}$	<i>memo</i> $F_{(behkn,20,5)}$	0
$F_{(behkn,12,8)}$	<i>base case</i>	0
$F_{(behkn,28,2)}$	$F_{(behkn,24,2)} + F_{(behkn,20,5)} + F_{(behkn,12,8)}$	0
$F_{(behkn,26,5)}$	<i>memo</i> $F_{(behkn,26,5)}$	0
$F_{(behkn,22,8)}$	<i>base case</i>	0
$F_{(behkn,14,11)}$	<i>base case</i>	0
$F_{(behkn,30,2)}$	$F_{(behkn,28,2)} + F_{(behkn,26,5)} + F_{(behkn,22,8)} + F_{(behkn,14,11)}$	0
$F_{(behkn,29,5)}$	<i>memo</i> $F_{(behkn,29,5)}$	0
$F_{(behkn,27,8)}$	<i>base case</i>	0
$F_{(behkn,23,11)}$	<i>base case</i>	0
$F_{(behkn,15,14)}$	<i>base case</i>	0
$F_{(behkn,31,2)}$	$F_{(behkn,30,2)} + F_{(behkn,29,5)} + F_{(behkn,27,8)} + F_{(behkn,23,11)} + F_{(behkn,15,14)}$	0

Tabel D.35 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist* = 3 pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan *X* = 5 (1)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,0,3)}$	<i>base case</i>	0
$F_{(behkn,0,4)}$	<i>base case</i>	0
$F_{(behkn,0,4)}$	<i>base case</i>	0
$G_{(behkn,16,3)}$	$G_{(behkn,0,3)} + F_{(behkn,0,4)} + F_{(behkn,0,4)}$	0
$F_{(behkn,0,4)}$	<i>base case</i>	0
$F_{(behkn,16,4)}$	$F_{(behkn,0,4)}$	0
$F_{(behkn,16,4)}$	<i>memoF</i> $_{(behkn,16,4)}$	0
$G_{(behkn,8,6)}$	<i>base case</i>	0
$F_{(behkn,8,7)}$	<i>base case</i>	0
$F_{(behkn,8,5)}$	<i>memoF</i> $_{(behkn,8,5)}$	0
$G_{(behkn,24,3)}$	$G_{(behkn,16,3)} + F_{(behkn,16,4)} + F_{(behkn,16,4)} + G_{(behkn,8,6)} + F_{(behkn,8,7)} + F_{(behkn,8,5)}$	0
$F_{(behkn,16,4)}$	<i>memoF</i> $_{(behkn,16,4)}$	0
$F_{(behkn,8,7)}$	<i>base case</i>	0
$F_{(behkn,24,4)}$	$F_{(behkn,16,4)} + F_{(behkn,8,7)}$	0
$F_{(behkn,24,4)}$	<i>memoF</i> $_{(behkn,24,4)}$	0
$G_{(behkn,20,6)}$	<i>base case</i>	0
$F_{(behkn,20,7)}$	<i>base case</i>	0
$F_{(behkn,20,5)}$	<i>memoF</i> $_{(behkn,20,5)}$	0
$G_{(behkn,12,9)}$	<i>base case</i>	0
$F_{(behkn,12,10)}$	<i>base case</i>	0
$F_{(behkn,12,8)}$	<i>base case</i>	0

Tabel D.36 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist* = 3 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (2)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,28,3)}$	$G_{(behkn,24,3)} + F_{(behkn,24,4)} + F_{(behkn,24,4)} + G_{(behkn,20,6)} + F_{(behkn,20,7)} + F_{(behkn,20,5)} + G_{(behkn,12,9)} + F_{(behkn,12,10)} + F_{(behkn,12,8)}$	0
$F_{(behkn,24,4)}$	<i>memoF</i> $_{(behkn,24,4)}$	0
$F_{(behkn,20,7)}$	<i>base case</i>	0
$F_{(behkn,12,10)}$	<i>base case</i>	0
$F_{(behkn,28,4)}$	$F_{(behkn,24,4)} + F_{(behkn,20,7)} + F_{(behkn,12,10)}$	0
$F_{(behkn,28,4)}$	<i>memoF</i> $_{(behkn,28,4)}$	0
$G_{(behkn,26,6)}$	<i>base case</i>	0
$F_{(behkn,26,7)}$	<i>base case</i>	0
$F_{(behkn,26,5)}$	<i>memoF</i> $_{(behkn,26,5)}$	0
$G_{(behkn,22,9)}$	<i>base case</i>	0
$F_{(behkn,22,10)}$	<i>base case</i>	0
$F_{(behkn,22,8)}$	<i>base case</i>	0
$G_{(behkn,14,12)}$	<i>base case</i>	0
$F_{(behkn,14,13)}$	<i>base case</i>	0
$F_{(behkn,14,11)}$	<i>base case</i>	0
$G_{(behkn,30,3)}$	$G_{(behkn,28,3)} + F_{(behkn,28,4)} + F_{(behkn,28,4)} + G_{(behkn,26,6)} + F_{(behkn,26,7)} + F_{(behkn,26,5)} + G_{(behkn,22,9)} + F_{(behkn,22,10)} + F_{(behkn,22,8)} + G_{(behkn,14,12)} + F_{(behkn,14,13)} + F_{(behkn,14,11)}$	0
$F_{(behkn,28,4)}$	<i>memoF</i> $_{(behkn,28,4)}$	0
$F_{(behkn,26,7)}$	<i>base case</i>	0

Tabel D.37 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist = 3* pada kasus *string ad1 = kbenh, string ad2 = kbenh* dan *X = 5 (3)*

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,22,10)}$	<i>base case</i>	0
$F_{(behkn,14,13)}$	<i>base case</i>	0
$F_{(behkn,30,4)}$	$F_{(behkn,28,4)} + F_{(behkn,26,7)} + F_{(behkn,22,10)} + F_{(behkn,14,13)}$	0
$F_{(behkn,30,4)}$	<i>memoF_(behkn,30,4)</i>	0
$G_{(behkn,29,6)}$	<i>base case</i>	0
$F_{(behkn,29,7)}$	<i>base case</i>	0
$F_{(behkn,29,5)}$	<i>memoF_(behkn,29,5)</i>	0
$G_{(behkn,27,9)}$	<i>base case</i>	0
$F_{(behkn,27,10)}$	<i>base case</i>	0
$F_{(behkn,27,8)}$	<i>base case</i>	0
$G_{(behkn,23,12)}$	<i>base case</i>	0
$F_{(behkn,23,13)}$	<i>base case</i>	0
$F_{(behkn,23,11)}$	<i>base case</i>	0
$G_{(behkn,15,15)}$	<i>base case</i>	0
$F_{(behkn,15,16)}$	<i>base case</i>	0
$F_{(behkn,15,14)}$	<i>base case</i>	0
$G_{(behkn,31,3)}$	$G_{(behkn,30,3)} + F_{(behkn,30,4)} + F_{(behkn,30,4)} + G_{(behkn,29,6)} + F_{(behkn,29,7)} + F_{(behkn,29,5)} + G_{(behkn,27,9)} + F_{(behkn,27,10)} + F_{(behkn,27,8)} + G_{(behkn,23,12)} + F_{(behkn,23,13)} + F_{(behkn,23,11)} + G_{(behkn,15,15)} + F_{(behkn,15,16)} + F_{(behkn,15,14)}$	0

Tabel D.38 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist* = 3 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (1)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,0,2)}$	<i>base case</i>	0
$F_{(behkn,0,3)}$	<i>base case</i>	0
$F_{(behkn,0,3)}$	<i>base case</i>	0
$G_{(behkn,16,2)}$	$G_{(behkn,0,2)} + F_{(behkn,0,3)} + F_{(behkn,0,3)}$	0
$F_{(behkn,16,3)}$	<i>memoF</i> _(<i>behkn,16,3</i>)	0
$F_{(behkn,16,3)}$	<i>memoF</i> _(<i>behkn,16,3</i>)	0
$G_{(behkn,0,8)}$	<i>base case</i>	0
$F_{(behkn,0,7)}$	<i>base case</i>	0
$F_{(behkn,0,9)}$	<i>base case</i>	0
$G_{(behkn,8,5)}$	$G_{(behkn,0,8)} + F_{(behkn,0,7)} + F_{(behkn,0,9)}$	0
$F_{(behkn,8,6)}$	<i>base case</i>	0
$F_{(behkn,0,7)}$	<i>base case</i>	0
$F_{(behkn,8,4)}$	$F_{(behkn,0,7)}$	0
$G_{(behkn,24,2)}$	$G_{(behkn,16,2)} + F_{(behkn,16,3)} + F_{(behkn,16,3)} + G_{(behkn,8,5)} + F_{(behkn,8,6)} + F_{(behkn,8,4)}$	0
$F_{(behkn,24,3)}$	<i>memoF</i> _(<i>behkn,24,3</i>)	0
$F_{(behkn,24,3)}$	<i>memoF</i> _(<i>behkn,24,3</i>)	0
$G_{(behkn,16,8)}$	<i>base case</i>	0
$F_{(behkn,16,7)}$	<i>base case</i>	0
$F_{(behkn,16,9)}$	<i>base case</i>	0
$G_{(behkn,4,8)}$	<i>base case</i>	0
$F_{(behkn,4,9)}$	<i>base case</i>	0
$F_{(behkn,4,7)}$	<i>base case</i>	0

Tabel D.39 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist* = 3 pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan *X* = 5 (2)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,20,5)}$	$G_{(behkn,16,8)} + F_{(behkn,16,7)} + F_{(behkn,16,9)} + G_{(behkn,4,8)} + F_{(behkn,4,9)} + F_{(behkn,4,7)}$	0
$F_{(behkn,20,6)}$	<i>base case</i>	0
$F_{(behkn,16,7)}$	<i>base case</i>	0
$F_{(behkn,4,7)}$	<i>base case</i>	0
$F_{(behkn,20,4)}$	$F_{(behkn,16,7)} + F_{(behkn,4,7)}$	0
$G_{(behkn,12,8)}$	<i>base case</i>	0
$F_{(behkn,12,9)}$	<i>base case</i>	0
$F_{(behkn,12,7)}$	<i>base case</i>	0
$G_{(behkn,28,2)}$	$G_{(behkn,24,2)} + F_{(behkn,24,3)} + F_{(behkn,24,3)} + G_{(behkn,20,5)} + F_{(behkn,20,6)} + F_{(behkn,20,4)} + G_{(behkn,12,8)} + F_{(behkn,12,9)} + F_{(behkn,12,7)}$	0
$F_{(behkn,28,3)}$	<i>memoF</i> $_{(behkn,28,3)}$	0
$F_{(behkn,28,3)}$	<i>memoF</i> $_{(behkn,28,3)}$	0
$G_{(behkn,24,8)}$	<i>base case</i>	0
$F_{(behkn,24,7)}$	<i>base case</i>	0
$F_{(behkn,24,9)}$	<i>base case</i>	0
$G_{(behkn,18,8)}$	<i>base case</i>	0
$F_{(behkn,18,9)}$	<i>base case</i>	0
$F_{(behkn,18,7)}$	<i>base case</i>	0
$G_{(behkn,10,11)}$	<i>base case</i>	0
$F_{(behkn,10,12)}$	<i>base case</i>	0
$F_{(behkn,10,10)}$	<i>base case</i>	0

Tabel D.40 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist* = 3 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (3)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,26,5)}$	$G_{(behkn,24,8)} + F_{(behkn,24,7)} + F_{(behkn,24,9)} + G_{(behkn,18,8)} + F_{(behkn,18,9)} + F_{(behkn,18,7)} + G_{(behkn,10,11)} + F_{(behkn,10,12)} + F_{(behkn,10,10)}$	0
$F_{(behkn,26,6)}$	<i>base case</i>	0
$F_{(behkn,24,7)}$	<i>base case</i>	0
$F_{(behkn,18,7)}$	<i>base case</i>	0
$F_{(behkn,10,10)}$	<i>base case</i>	0
$F_{(behkn,26,4)}$	$F_{(behkn,24,7)} + F_{(behkn,18,7)} + F_{(behkn,10,10)}$	0
$G_{(behkn,22,8)}$	<i>base case</i>	0
$F_{(behkn,22,9)}$	<i>base case</i>	0
$F_{(behkn,22,7)}$	<i>base case</i>	0
$G_{(behkn,14,11)}$	<i>base case</i>	0
$F_{(behkn,14,12)}$	<i>base case</i>	0
$F_{(behkn,14,10)}$	<i>base case</i>	0
$G_{(behkn,30,2)}$	$G_{(behkn,28,2)} + F_{(behkn,28,3)} + F_{(behkn,28,3)} + G_{(behkn,26,5)} + F_{(behkn,26,6)} + F_{(behkn,26,4)} + G_{(behkn,22,8)} + F_{(behkn,22,9)} + F_{(behkn,22,7)} + G_{(behkn,14,11)} + F_{(behkn,14,12)} + F_{(behkn,14,10)}$	0
$F_{(behkn,30,3)}$	<i>memoF</i> $_{(behkn,30,3)}$	0
$F_{(behkn,30,3)}$	<i>memoF</i> $_{(behkn,30,3)}$	0
$G_{(behkn,28,8)}$	<i>base case</i>	0
$F_{(behkn,28,7)}$	<i>base case</i>	0
$F_{(behkn,28,9)}$	<i>base case</i>	0

Tabel D.41 Simulasi perhitungan jumlah kombinasi $string\ orig1$ dengan operasi $replace$ dengan $dist = 3$ pada kasus $string\ ad1 = kbenh$, $string\ ad2 = kbenh$ dan $X = 5$ (4)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,25,8)}$	<i>base case</i>	0
$F_{(behkn,25,9)}$	<i>base case</i>	0
$F_{(behkn,25,7)}$	<i>base case</i>	0
$G_{(behkn,21,11)}$	<i>base case</i>	0
$F_{(behkn,21,12)}$	<i>base case</i>	0
$F_{(behkn,21,10)}$	<i>base case</i>	0
$G_{(behkn,13,14)}$	<i>base case</i>	0
$F_{(behkn,13,15)}$	<i>base case</i>	0
$F_{(behkn,13,13)}$	<i>base case</i>	0
$G_{(behkn,29,5)}$	$G_{(behkn,28,8)} + F_{(behkn,28,7)} +$ $F_{(behkn,28,9)} + G_{(behkn,25,8)} +$ $F_{(behkn,25,9)} + F_{(behkn,25,7)} +$ $G_{(behkn,21,11)} + F_{(behkn,21,12)} +$ $F_{(behkn,21,10)} + G_{(behkn,13,14)} +$ $F_{(behkn,13,15)} + F_{(behkn,13,13)}$	0
$F_{(behkn,29,6)}$	<i>base case</i>	0
$F_{(behkn,28,7)}$	<i>base case</i>	0
$F_{(behkn,25,7)}$	<i>base case</i>	0
$F_{(behkn,21,10)}$	<i>base case</i>	0
$F_{(behkn,13,13)}$	<i>base case</i>	0
$F_{(behkn,29,4)}$	$F_{(behkn,28,7)} + F_{(behkn,25,7)} +$ $F_{(behkn,21,10)} + F_{(behkn,13,13)}$	0
$G_{(behkn,27,8)}$	<i>base case</i>	0
$F_{(behkn,27,9)}$	<i>base case</i>	0
$F_{(behkn,27,7)}$	<i>base case</i>	0
$G_{(behkn,23,11)}$	<i>base case</i>	0
$F_{(behkn,23,12)}$	<i>base case</i>	0

Tabel D.42 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist* = 3 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (5)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,23,10)}$	<i>base case</i>	0
$G_{(behkn,15,14)}$	<i>base case</i>	0
$F_{(behkn,15,15)}$	<i>base case</i>	0
$F_{(behkn,15,13)}$	<i>base case</i>	0
$G_{(behkn,31,2)}$	$G_{(behkn,30,2)} + F_{(behkn,30,3)} +$ $F_{(behkn,30,3)} + G_{(behkn,29,5)} +$ $F_{(behkn,29,6)} + F_{(behkn,29,4)} +$ $G_{(behkn,27,8)} + F_{(behkn,27,9)} +$ $F_{(behkn,27,7)} + G_{(behkn,23,11)} +$ $F_{(behkn,23,12)} + F_{(behkn,23,10)} +$ $G_{(behkn,15,14)} + F_{(behkn,15,15)} +$ $F_{(behkn,15,13)}$	0

Tabel D.43 Simulasi perhitungan jumlah kombinasi *string orig2* tanpa operasi *replace* dengan *dist* = 3 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,30,3)}$	<i>memo</i> $F_{(behkn,30,3)}$	0
$F_{(behkn,29,6)}$	<i>base case</i>	0
$F_{(behkn,27,9)}$	<i>base case</i>	0
$F_{(behkn,23,12)}$	<i>base case</i>	0
$F_{(behkn,15,15)}$	<i>base case</i>	0
$F_{(behkn,31,3)}$	$F_{(behkn,30,3)} + F_{(behkn,29,6)} +$ $F_{(behkn,27,9)} + F_{(behkn,23,12)} +$ $F_{(behkn,15,15)}$	0

Tabel D.44 Simulasi perhitungan jumlah kombinasi *string orig1* tanpa operasi *replace* dengan *dist = 4* pada kasus *string ad1 = kbenh, string ad2 = kbenh* dan *X = 5*

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,0,1)}$	<i>base case</i>	0
$F_{(behkn,16,1)}$	$F_{(behkn,0,1)}$	0
$F_{(behkn,8,4)}$	<i>memo</i> $F_{(behkn,8,4)}$	0
$F_{(behkn,24,1)}$	$F_{(behkn,16,1)} + F_{(behkn,8,4)}$	0
$F_{(behkn,20,4)}$	<i>memo</i> $F_{(behkn,20,4)}$	0
$F_{(behkn,12,7)}$	<i>base case</i>	0
$F_{(behkn,28,1)}$	$F_{(behkn,24,1)} + F_{(behkn,20,4)} + F_{(behkn,12,7)}$	0
$F_{(behkn,26,4)}$	<i>memo</i> $F_{(behkn,26,4)}$	0
$F_{(behkn,22,7)}$	<i>base case</i>	0
$F_{(behkn,14,10)}$	<i>base case</i>	0
$F_{(behkn,30,1)}$	$F_{(behkn,28,1)} + F_{(behkn,26,4)} + F_{(behkn,22,7)} + F_{(behkn,14,10)}$	0
$F_{(behkn,29,4)}$	<i>memo</i> $F_{(behkn,29,4)}$	0
$F_{(behkn,27,7)}$	<i>base case</i>	0
$F_{(behkn,23,10)}$	<i>base case</i>	0
$F_{(behkn,15,13)}$	<i>base case</i>	0
$F_{(behkn,31,1)}$	$F_{(behkn,30,1)} + F_{(behkn,29,4)} + F_{(behkn,27,7)} + F_{(behkn,23,10)} + F_{(behkn,15,13)}$	0

Tabel D.45 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist* = 4 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (1)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,0,4)}$	<i>base case</i>	0
$F_{(behkn,0,5)}$	<i>base case</i>	1
$F_{(behkn,0,5)}$	<i>base case</i>	1
$G_{(behkn,16,4)}$	$G_{(behkn,0,4)} + F_{(behkn,0,5)} + F_{(behkn,0,5)}$	2
$F_{(behkn,16,5)}$	<i>memoF</i> $_{(behkn,16,5)}$	1
$F_{(behkn,16,5)}$	<i>memoF</i> $_{(behkn,16,5)}$	1
$G_{(behkn,8,7)}$	<i>base case</i>	0
$F_{(behkn,8,8)}$	<i>base case</i>	0
$F_{(behkn,8,6)}$	<i>base case</i>	0
$G_{(behkn,24,4)}$	$G_{(behkn,16,4)} + F_{(behkn,16,5)} + F_{(behkn,16,5)} + G_{(behkn,8,7)} + F_{(behkn,8,8)} + F_{(behkn,8,6)}$	4
$F_{(behkn,24,5)}$	<i>memoF</i> $_{(behkn,24,5)}$	1
$F_{(behkn,24,5)}$	<i>memoF</i> $_{(behkn,24,5)}$	1
$G_{(behkn,20,7)}$	<i>base case</i>	0
$F_{(behkn,20,8)}$	<i>base case</i>	0
$F_{(behkn,20,6)}$	<i>base case</i>	0
$G_{(behkn,12,10)}$	<i>base case</i>	0
$F_{(behkn,12,11)}$	<i>base case</i>	0
$F_{(behkn,12,9)}$	<i>base case</i>	0
$G_{(behkn,28,4)}$	$G_{(behkn,24,4)} + F_{(behkn,24,5)} + F_{(behkn,24,5)} + G_{(behkn,20,7)} + F_{(behkn,20,8)} + F_{(behkn,20,6)} + G_{(behkn,12,10)} + F_{(behkn,12,11)} + F_{(behkn,12,9)}$	6
$F_{(behkn,28,5)}$	<i>memoF</i> $_{(behkn,28,5)}$	1
$F_{(behkn,28,5)}$	<i>memoF</i> $_{(behkn,28,5)}$	1

Tabel D.46 Simulasi perhitungan jumlah kombinasi $string\ orig2$ dengan operasi $replace$ dengan $dist = 4$ pada kasus $string\ ad1 = kbenh$, $string\ ad2 = kbenh$ dan $X = 5$ (2)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,26,7)}$	<i>base case</i>	0
$F_{(behkn,26,8)}$	<i>base case</i>	0
$F_{(behkn,26,6)}$	<i>base case</i>	0
$G_{(behkn,22,10)}$	<i>base case</i>	0
$F_{(behkn,22,11)}$	<i>base case</i>	0
$F_{(behkn,22,9)}$	<i>base case</i>	0
$G_{(behkn,14,13)}$	<i>base case</i>	0
$F_{(behkn,14,14)}$	<i>base case</i>	0
$F_{(behkn,14,12)}$	<i>base case</i>	0
$G_{(behkn,30,4)}$	$G_{(behkn,28,4)} + F_{(behkn,28,5)} +$ $F_{(behkn,28,5)} + G_{(behkn,26,7)} +$ $F_{(behkn,26,8)} + F_{(behkn,26,6)} +$ $G_{(behkn,22,10)} + F_{(behkn,22,11)} +$ $F_{(behkn,22,9)} + G_{(behkn,14,13)} +$ $F_{(behkn,14,14)} + F_{(behkn,14,12)}$	8
$F_{(behkn,28,5)}$	$memoF_{(behkn,28,5)}$	1
$F_{(behkn,26,8)}$	<i>base case</i>	0
$F_{(behkn,22,11)}$	<i>base case</i>	0
$F_{(behkn,14,14)}$	<i>base case</i>	0
$F_{(behkn,30,5)}$	$F_{(behkn,28,5)} + F_{(behkn,26,8)} +$ $F_{(behkn,22,11)} + F_{(behkn,14,14)}$	1
$F_{(behkn,30,5)}$	$memoF_{(behkn,30,5)}$	1
$G_{(behkn,29,7)}$	<i>base case</i>	0
$F_{(behkn,29,8)}$	<i>base case</i>	0

Tabel D.47 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist* = 4 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (3)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,29,6)}$	<i>base case</i>	0
$G_{(behkn,27,10)}$	<i>base case</i>	0
$F_{(behkn,27,11)}$	<i>base case</i>	0
$F_{(behkn,27,9)}$	<i>base case</i>	0
$G_{(behkn,23,13)}$	<i>base case</i>	0
$F_{(behkn,23,14)}$	<i>base case</i>	0
$F_{(behkn,23,12)}$	<i>base case</i>	0
$G_{(behkn,15,16)}$	<i>base case</i>	0
$F_{(behkn,15,17)}$	<i>base case</i>	0
$F_{(behkn,15,15)}$	<i>base case</i>	0
$G_{(behkn,31,4)}$	$G_{(behkn,30,4)} + F_{(behkn,30,5)} +$ $F_{(behkn,30,5)} + G_{(behkn,29,7)} +$ $F_{(behkn,29,8)} + F_{(behkn,29,6)} +$ $G_{(behkn,27,10)} + F_{(behkn,27,11)} +$ $F_{(behkn,27,9)} + G_{(behkn,23,13)} +$ $F_{(behkn,23,14)} + F_{(behkn,23,12)} +$ $G_{(behkn,15,16)} + F_{(behkn,15,17)} +$ $F_{(behkn,15,15)}$	10

Tabel D.48 Simulasi perhitungan jumlah kombinasi $string\ orig1$ dengan operasi $replace$ dengan $dist = 4$ pada kasus $string\ ad1 = kbenh$, $string\ ad2 = kbenh$ dan $X = 5$ (1)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,0,1)}$	<i>base case</i>	0
$F_{(behkn,0,2)}$	<i>base case</i>	0
$F_{(behkn,0,2)}$	<i>base case</i>	0
$G_{(behkn,16,1)}$	$G_{(behkn,0,1)} + F_{(behkn,0,2)} + F_{(behkn,0,2)}$	0
$F_{(behkn,16,2)}$	$memoF_{(behkn,16,2)}$	0
$F_{(behkn,16,2)}$	$memoF_{(behkn,16,2)}$	0
$G_{(behkn,0,7)}$	<i>base case</i>	0
$F_{(behkn,0,6)}$	<i>base case</i>	0
$F_{(behkn,0,8)}$	<i>base case</i>	0
$G_{(behkn,8,4)}$	$G_{(behkn,0,7)} + F_{(behkn,0,6)} + F_{(behkn,0,8)}$	0
$F_{(behkn,8,5)}$	$memoF_{(behkn,8,5)}$	0
$F_{(behkn,0,6)}$	<i>base case</i>	0
$F_{(behkn,8,3)}$	$F_{(behkn,0,6)}$	0
$G_{(behkn,24,1)}$	$G_{(behkn,16,1)} + F_{(behkn,16,2)} + F_{(behkn,16,2)} + G_{(behkn,8,4)} + F_{(behkn,8,5)} + F_{(behkn,8,3)}$	0
$F_{(behkn,24,2)}$	$memoF_{(behkn,24,2)}$	0
$F_{(behkn,24,2)}$	$memoF_{(behkn,24,2)}$	0
$G_{(behkn,16,7)}$	<i>base case</i>	0
$F_{(behkn,16,6)}$	<i>base case</i>	0
$F_{(behkn,16,8)}$	<i>base case</i>	0
$G_{(behkn,4,7)}$	<i>base case</i>	0
$F_{(behkn,4,8)}$	<i>base case</i>	0
$F_{(behkn,4,6)}$	<i>base case</i>	0

Tabel D.49 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist* = 4 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (2)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,20,4)}$	$G_{(behkn,16,7)} + F_{(behkn,16,6)} + F_{(behkn,16,8)} + G_{(behkn,4,7)} + F_{(behkn,4,8)} + F_{(behkn,4,6)}$	0
$F_{(behkn,20,5)}$	$memoF_{(behkn,20,5)}$	0
$F_{(behkn,16,6)}$	<i>base case</i>	0
$F_{(behkn,4,6)}$	<i>base case</i>	0
$F_{(behkn,20,3)}$	$F_{(behkn,16,6)} + F_{(behkn,4,6)}$	0
$G_{(behkn,12,7)}$	<i>base case</i>	0
$F_{(behkn,12,8)}$	<i>base case</i>	0
$F_{(behkn,12,6)}$	<i>base case</i>	0
$G_{(behkn,28,1)}$	$G_{(behkn,24,1)} + F_{(behkn,24,2)} + F_{(behkn,24,2)} + G_{(behkn,20,4)} + F_{(behkn,20,5)} + F_{(behkn,20,3)} + G_{(behkn,12,7)} + F_{(behkn,12,8)} + F_{(behkn,12,6)}$	0
$F_{(behkn,28,2)}$	$memoF_{(behkn,28,2)}$	0
$F_{(behkn,28,2)}$	$memoF_{(behkn,28,2)}$	0
$G_{(behkn,24,7)}$	<i>base case</i>	0
$F_{(behkn,24,6)}$	<i>base case</i>	0
$F_{(behkn,24,8)}$	<i>base case</i>	0
$G_{(behkn,18,7)}$	<i>base case</i>	0
$F_{(behkn,18,8)}$	<i>base case</i>	0
$F_{(behkn,18,6)}$	<i>base case</i>	0
$G_{(behkn,10,10)}$	<i>base case</i>	0
$F_{(behkn,10,11)}$	<i>base case</i>	0
$F_{(behkn,10,9)}$	<i>base case</i>	0

Tabel D.50 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist = 4* pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan *X = 5 (3)*

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,26,4)}$	$G_{(behkn,24,7)} + F_{(behkn,24,6)} + F_{(behkn,24,8)} + G_{(behkn,18,7)} + F_{(behkn,18,8)} + F_{(behkn,18,6)} + G_{(behkn,10,10)} + F_{(behkn,10,11)} + F_{(behkn,10,9)}$	0
$F_{(behkn,26,5)}$	<i>memoF</i> $_{(behkn,26,5)}$	0
$F_{(behkn,24,6)}$	<i>base case</i>	0
$F_{(behkn,18,6)}$	<i>base case</i>	0
$F_{(behkn,10,9)}$	<i>base case</i>	0
$F_{(behkn,26,3)}$	$F_{(behkn,24,6)} + F_{(behkn,18,6)} + F_{(behkn,10,9)}$	0
$G_{(behkn,22,7)}$	<i>base case</i>	0
$F_{(behkn,22,8)}$	<i>base case</i>	0
$F_{(behkn,22,6)}$	<i>base case</i>	0
$G_{(behkn,14,10)}$	<i>base case</i>	0
$F_{(behkn,14,11)}$	<i>base case</i>	0
$F_{(behkn,14,9)}$	<i>base case</i>	0
$G_{(behkn,30,1)}$	$G_{(behkn,28,1)} + F_{(behkn,28,2)} + F_{(behkn,28,2)} + G_{(behkn,26,4)} + F_{(behkn,26,5)} + F_{(behkn,26,3)} + G_{(behkn,22,7)} + F_{(behkn,22,8)} + F_{(behkn,22,6)} + G_{(behkn,14,10)} + F_{(behkn,14,11)} + F_{(behkn,14,9)}$	0
$F_{(behkn,30,2)}$	<i>memoF</i> $_{(behkn,30,2)}$	0
$F_{(behkn,30,2)}$	<i>memoF</i> $_{(behkn,30,2)}$	0
$G_{(behkn,28,7)}$	<i>base case</i>	0
$F_{(behkn,28,6)}$	<i>base case</i>	0
$F_{(behkn,28,8)}$	<i>base case</i>	0
$G_{(behkn,25,7)}$	<i>base case</i>	0

Tabel D.51 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist* = 4 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (4)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,25,8)}$	<i>base case</i>	0
$F_{(behkn,25,6)}$	<i>base case</i>	0
$G_{(behkn,21,10)}$	<i>base case</i>	0
$F_{(behkn,21,11)}$	<i>base case</i>	0
$F_{(behkn,21,9)}$	<i>base case</i>	0
$G_{(behkn,13,13)}$	<i>base case</i>	0
$F_{(behkn,13,14)}$	<i>base case</i>	0
$F_{(behkn,13,12)}$	<i>base case</i>	0
$G_{(behkn,29,4)}$	$G_{(behkn,28,7)} + F_{(behkn,28,6)} +$ $F_{(behkn,28,8)} + G_{(behkn,25,7)} +$ $F_{(behkn,25,8)} + F_{(behkn,25,6)} +$ $G_{(behkn,21,10)} + F_{(behkn,21,11)} +$ $F_{(behkn,21,9)} + G_{(behkn,13,13)} +$ $F_{(behkn,13,14)} + F_{(behkn,13,12)}$	0
$F_{(behkn,29,5)}$	<i>memo</i> $F_{(behkn,29,5)}$	0
$F_{(behkn,28,6)}$	<i>base case</i>	0
$F_{(behkn,25,6)}$	<i>base case</i>	0
$F_{(behkn,21,9)}$	<i>base case</i>	0
$F_{(behkn,13,12)}$	<i>base case</i>	0
$F_{(behkn,29,3)}$	$F_{(behkn,28,6)} + F_{(behkn,25,6)} +$ $F_{(behkn,21,9)} + F_{(behkn,13,12)}$	0
$G_{(behkn,27,7)}$	<i>base case</i>	0
$F_{(behkn,27,8)}$	<i>base case</i>	0
$F_{(behkn,27,6)}$	<i>base case</i>	0
$G_{(behkn,23,10)}$	<i>base case</i>	0
$F_{(behkn,23,11)}$	<i>base case</i>	0
$F_{(behkn,23,9)}$	<i>base case</i>	0

Tabel D.52 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist* = 4 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (5)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,15,13)}$	<i>base case</i>	0
$F_{(behkn,15,14)}$	<i>base case</i>	0
$F_{(behkn,15,12)}$	<i>base case</i>	0
$G_{(behkn,31,1)}$	$G_{(behkn,30,1)} + F_{(behkn,30,2)} +$ $F_{(behkn,30,2)} + G_{(behkn,29,4)} +$ $F_{(behkn,29,5)} + F_{(behkn,29,3)} +$ $G_{(behkn,27,7)} + F_{(behkn,27,8)} +$ $F_{(behkn,27,6)} + G_{(behkn,23,10)} +$ $F_{(behkn,23,11)} + F_{(behkn,23,9)} +$ $G_{(behkn,15,13)} + F_{(behkn,15,14)} +$ $F_{(behkn,15,12)}$	0

Tabel D.53 Simulasi perhitungan jumlah kombinasi *string orig2* tanpa operasi *replace* dengan *dist* = 4 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,30,4)}$	<i>memoF</i> $_{(behkn,30,4)}$	0
$F_{(behkn,29,7)}$	<i>base case</i>	0
$F_{(behkn,27,10)}$	<i>base case</i>	0
$F_{(behkn,23,13)}$	<i>base case</i>	0
$F_{(behkn,15,16)}$	<i>base case</i>	0
$F_{(behkn,31,4)}$	$F_{(behkn,30,4)} + F_{(behkn,29,7)} +$ $F_{(behkn,27,10)} + F_{(behkn,23,13)} +$ $F_{(behkn,15,16)}$	0

Tabel D.54 Simulasi perhitungan jumlah kombinasi *string orig1* tanpa operasi *replace* dengan *dist* = 5 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,0,0)}$	<i>base case</i>	0
$F_{(behkn,16,0)}$	$F_{(behkn,0,0)}$	0
$F_{(behkn,8,3)}$	<i>memo</i> $F_{(behkn,8,3)}$	0
$F_{(behkn,24,0)}$	$F_{(behkn,16,0)} + F_{(behkn,8,3)}$	0
$F_{(behkn,20,3)}$	<i>memo</i> $F_{(behkn,20,3)}$	0
$F_{(behkn,12,6)}$	<i>base case</i>	0
$F_{(behkn,28,0)}$	$F_{(behkn,24,0)} + F_{(behkn,20,3)} + F_{(behkn,12,6)}$	0
$F_{(behkn,26,3)}$	<i>memo</i> $F_{(behkn,26,3)}$	0
$F_{(behkn,22,6)}$	<i>base case</i>	0
$F_{(behkn,14,9)}$	<i>base case</i>	0
$F_{(behkn,30,0)}$	$F_{(behkn,28,0)} + F_{(behkn,26,3)} + F_{(behkn,22,6)} + F_{(behkn,14,9)}$	0
$F_{(behkn,29,3)}$	<i>memo</i> $F_{(behkn,29,3)}$	0
$F_{(behkn,27,6)}$	<i>base case</i>	0
$F_{(behkn,23,9)}$	<i>base case</i>	0
$F_{(behkn,15,12)}$	<i>base case</i>	0
$F_{(behkn,31,0)}$	$F_{(behkn,30,0)} + F_{(behkn,29,3)} + F_{(behkn,27,6)} + F_{(behkn,23,9)} + F_{(behkn,15,12)}$	0

Tabel D.55 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist* = 5 pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan *X* = 5 (1)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,0,5)}$	<i>base case</i>	0
$F_{(behkn,0,6)}$	<i>base case</i>	0
$F_{(behkn,0,6)}$	<i>base case</i>	0
$G_{(behkn,16,5)}$	$G_{(behkn,0,5)} + F_{(behkn,0,6)} + F_{(behkn,0,6)}$	0
$F_{(behkn,16,6)}$	<i>base case</i>	0
$F_{(behkn,16,6)}$	<i>base case</i>	0
$G_{(behkn,8,8)}$	<i>base case</i>	0
$F_{(behkn,8,9)}$	<i>base case</i>	0
$F_{(behkn,8,7)}$	<i>base case</i>	0
$G_{(behkn,24,5)}$	$G_{(behkn,16,5)} + F_{(behkn,16,6)} + F_{(behkn,16,6)} + G_{(behkn,8,8)} + F_{(behkn,8,9)} + F_{(behkn,8,7)}$	0
$F_{(behkn,24,6)}$	<i>base case</i>	0
$F_{(behkn,24,6)}$	<i>base case</i>	0
$G_{(behkn,20,8)}$	<i>base case</i>	0
$F_{(behkn,20,9)}$	<i>base case</i>	0
$F_{(behkn,20,7)}$	<i>base case</i>	0
$G_{(behkn,12,11)}$	<i>base case</i>	0
$F_{(behkn,12,12)}$	<i>base case</i>	0
$F_{(behkn,12,10)}$	<i>base case</i>	0
$G_{(behkn,28,5)}$	$G_{(behkn,24,5)} + F_{(behkn,24,6)} + F_{(behkn,24,6)} + G_{(behkn,20,8)} + F_{(behkn,20,9)} + F_{(behkn,20,7)} + G_{(behkn,12,11)} + F_{(behkn,12,12)} + F_{(behkn,12,10)}$	0
$F_{(behkn,28,6)}$	<i>base case</i>	0
$F_{(behkn,28,6)}$	<i>base case</i>	0

Tabel D.56 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist* = 5 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (2)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,26,8)}$	<i>base case</i>	0
$F_{(behkn,26,9)}$	<i>base case</i>	0
$F_{(behkn,26,7)}$	<i>base case</i>	0
$G_{(behkn,22,11)}$	<i>base case</i>	0
$F_{(behkn,22,12)}$	<i>base case</i>	0
$F_{(behkn,22,10)}$	<i>base case</i>	0
$G_{(behkn,14,14)}$	<i>base case</i>	0
$F_{(behkn,14,15)}$	<i>base case</i>	0
$F_{(behkn,14,13)}$	<i>base case</i>	0
$G_{(behkn,30,5)}$	$G_{(behkn,28,5)} + F_{(behkn,28,6)} +$ $F_{(behkn,28,6)} + G_{(behkn,26,8)} +$ $F_{(behkn,26,9)} + F_{(behkn,26,7)} +$ $G_{(behkn,22,11)} + F_{(behkn,22,12)} +$ $F_{(behkn,22,10)} + G_{(behkn,14,14)} +$ $F_{(behkn,14,15)} + F_{(behkn,14,13)}$	0
$F_{(behkn,30,6)}$	<i>base case</i>	0
$F_{(behkn,30,6)}$	<i>base case</i>	0
$G_{(behkn,29,8)}$	<i>base case</i>	0
$F_{(behkn,29,9)}$	<i>base case</i>	0
$F_{(behkn,29,7)}$	<i>base case</i>	0
$G_{(behkn,27,11)}$	<i>base case</i>	0
$F_{(behkn,27,12)}$	<i>base case</i>	0
$F_{(behkn,27,10)}$	<i>base case</i>	0
$G_{(behkn,23,14)}$	<i>base case</i>	0
$F_{(behkn,23,15)}$	<i>base case</i>	0
$F_{(behkn,23,13)}$	<i>base case</i>	0
$G_{(behkn,15,17)}$	<i>base case</i>	0

Tabel D.57 Simulasi perhitungan jumlah kombinasi *string orig2* dengan operasi *replace* dengan *dist* = 5 pada kasus *string ad1 = kbenh, string ad2 = kbenh* dan *X* = 5 (3)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,15,18)}$	<i>base case</i>	0
$F_{(behkn,15,16)}$	<i>base case</i>	0
$G_{(behkn,31,5)}$	$G_{(behkn,30,5)} + F_{(behkn,30,6)} +$ $F_{(behkn,30,6)} + G_{(behkn,29,8)} +$ $F_{(behkn,29,9)} + F_{(behkn,29,7)} +$ $G_{(behkn,27,11)} + F_{(behkn,27,12)} +$ $F_{(behkn,27,10)} + G_{(behkn,23,14)} +$ $F_{(behkn,23,15)} + F_{(behkn,23,13)} +$ $G_{(behkn,15,17)} + F_{(behkn,15,18)} +$ $F_{(behkn,15,16)}$	0

Tabel D.58 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist* = 5 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (1)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,0,0)}$	<i>base case</i>	0
$F_{(behkn,0,1)}$	<i>base case</i>	0
$F_{(behkn,0,1)}$	<i>base case</i>	0
$G_{(behkn,16,0)}$	$G_{(behkn,0,0)} + F_{(behkn,0,1)} + F_{(behkn,0,1)}$	0
$F_{(behkn,16,1)}$	<i>memoF</i> $_{(behkn,16,1)}$	0
$F_{(behkn,16,1)}$	<i>memoF</i> $_{(behkn,16,1)}$	0
$G_{(behkn,0,6)}$	<i>base case</i>	0
$F_{(behkn,0,5)}$	<i>base case</i>	1
$F_{(behkn,0,7)}$	<i>base case</i>	0
$G_{(behkn,8,3)}$	$G_{(behkn,0,6)} + F_{(behkn,0,5)} + F_{(behkn,0,7)}$	1
$F_{(behkn,8,4)}$	<i>memoF</i> $_{(behkn,8,4)}$	0
$F_{(behkn,0,5)}$	<i>base case</i>	1
$F_{(behkn,8,2)}$	$F_{(behkn,0,5)}$	1
$G_{(behkn,24,0)}$	$G_{(behkn,16,0)} + F_{(behkn,16,1)} + F_{(behkn,16,1)} + G_{(behkn,8,3)} + F_{(behkn,8,4)} + F_{(behkn,8,2)}$	2
$F_{(behkn,24,1)}$	<i>memoF</i> $_{(behkn,24,1)}$	0
$F_{(behkn,24,1)}$	<i>memoF</i> $_{(behkn,24,1)}$	0
$G_{(behkn,16,6)}$	<i>base case</i>	0
$F_{(behkn,16,5)}$	<i>memoF</i> $_{(behkn,16,5)}$	1
$F_{(behkn,16,7)}$	<i>base case</i>	0
$G_{(behkn,4,6)}$	<i>base case</i>	0
$F_{(behkn,4,7)}$	<i>base case</i>	0
$F_{(behkn,0,11)}$	<i>base case</i>	0
$F_{(behkn,4,5)}$	$F_{(behkn,0,11)}$	0

Tabel D.59 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist = 5* pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan *X = 5* (2)

Fungsi	Perhitungan Nilai	Nilai
$G_{(behkn,20,3)}$	$G_{(behkn,16,6)} + F_{(behkn,16,5)} + F_{(behkn,16,7)} + G_{(behkn,4,6)} + F_{(behkn,4,7)} + F_{(behkn,4,5)}$	1
$F_{(behkn,20,4)}$	$memoF_{(behkn,20,4)}$	0
$F_{(behkn,16,5)}$	$memoF_{(behkn,16,5)}$	1
$F_{(behkn,4,5)}$	$memoF_{(behkn,4,5)}$	0
$F_{(behkn,20,2)}$	$F_{(behkn,16,5)} + F_{(behkn,4,5)}$	1
$G_{(behkn,12,6)}$	<i>base case</i>	0
$F_{(behkn,12,7)}$	<i>base case</i>	0
$F_{(behkn,8,8)}$	<i>base case</i>	0
$F_{(behkn,4,5)}$	$memoF_{(behkn,4,5)}$	0
$F_{(behkn,12,5)}$	$F_{(behkn,8,8)} + F_{(behkn,4,5)}$	0
$G_{(behkn,28,0)}$	$G_{(behkn,24,0)} + F_{(behkn,24,1)} + F_{(behkn,24,1)} + G_{(behkn,20,3)} + F_{(behkn,20,4)} + F_{(behkn,20,2)} + G_{(behkn,12,6)} + F_{(behkn,12,7)} + F_{(behkn,12,5)}$	4
$F_{(behkn,28,1)}$	$memoF_{(behkn,28,1)}$	0
$F_{(behkn,28,1)}$	$memoF_{(behkn,28,1)}$	0
$G_{(behkn,24,6)}$	<i>base case</i>	0
$F_{(behkn,24,5)}$	$memoF_{(behkn,24,5)}$	1
$F_{(behkn,24,7)}$	<i>base case</i>	0
$G_{(behkn,18,6)}$	<i>base case</i>	0
$F_{(behkn,18,7)}$	<i>base case</i>	0
$F_{(behkn,16,11)}$	<i>base case</i>	0
$F_{(behkn,2,8)}$	<i>base case</i>	0
$F_{(behkn,18,5)}$	$F_{(behkn,16,11)} + F_{(behkn,2,8)}$	0
$G_{(behkn,10,9)}$	<i>base case</i>	0
$F_{(behkn,10,10)}$	<i>base case</i>	0

Tabel D.60 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist* = 5 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (3)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,10,8)}$	<i>base case</i>	0
$G_{(behkn,26,3)}$	$G_{(behkn,24,6)} + F_{(behkn,24,5)} + F_{(behkn,24,7)} + G_{(behkn,18,6)} + F_{(behkn,18,7)} + F_{(behkn,18,5)} + G_{(behkn,10,9)} + F_{(behkn,10,10)} + F_{(behkn,10,8)}$	1
$F_{(behkn,26,4)}$	<i>memoF</i> ($behkn,26,4$)	0
$F_{(behkn,24,5)}$	<i>memoF</i> ($behkn,24,5$)	1
$F_{(behkn,18,5)}$	<i>memoF</i> ($behkn,18,5$)	0
$F_{(behkn,10,8)}$	<i>base case</i>	0
$F_{(behkn,26,2)}$	$F_{(behkn,24,5)} + F_{(behkn,18,5)} + F_{(behkn,10,8)}$	1
$G_{(behkn,22,6)}$	<i>base case</i>	0
$F_{(behkn,22,7)}$	<i>base case</i>	0
$F_{(behkn,20,8)}$	<i>base case</i>	0
$F_{(behkn,18,5)}$	<i>memoF</i> ($behkn,18,5$)	0
$F_{(behkn,6,11)}$	<i>base case</i>	0
$F_{(behkn,22,5)}$	$F_{(behkn,20,8)} + F_{(behkn,18,5)} + F_{(behkn,6,11)}$	0
$G_{(behkn,14,9)}$	<i>base case</i>	0
$F_{(behkn,14,10)}$	<i>base case</i>	0
$F_{(behkn,14,8)}$	<i>base case</i>	0
$G_{(behkn,30,0)}$	$G_{(behkn,28,0)} + F_{(behkn,28,1)} + F_{(behkn,28,1)} + G_{(behkn,26,3)} + F_{(behkn,26,4)} + F_{(behkn,26,2)} + G_{(behkn,22,6)} + F_{(behkn,22,7)} + F_{(behkn,22,5)} + G_{(behkn,14,9)} + F_{(behkn,14,10)} + F_{(behkn,14,8)}$	6
$F_{(behkn,30,1)}$	<i>memoF</i> ($behkn,30,1$)	0

Tabel D.61 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist* = 5 pada kasus *string ad1 = kbenh*, *string ad2 = kbenh* dan *X* = 5 (4)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,30,1)}$	$memoF_{(behkn,30,1)}$	0
$G_{(behkn,28,6)}$	<i>base case</i>	0
$F_{(behkn,28,5)}$	$memoF_{(behkn,28,5)}$	1
$F_{(behkn,28,7)}$	<i>base case</i>	0
$G_{(behkn,25,6)}$	<i>base case</i>	0
$F_{(behkn,25,7)}$	<i>base case</i>	0
$F_{(behkn,24,11)}$	<i>base case</i>	0
$F_{(behkn,17,8)}$	<i>base case</i>	0
$F_{(behkn,9,11)}$	<i>base case</i>	0
$F_{(behkn,25,5)}$	$F_{(behkn,24,11)} + F_{(behkn,17,8)} + F_{(behkn,9,11)}$	0
$G_{(behkn,21,9)}$	<i>base case</i>	0
$F_{(behkn,21,10)}$	<i>base case</i>	0
$F_{(behkn,21,8)}$	<i>base case</i>	0
$G_{(behkn,13,12)}$	<i>base case</i>	0
$F_{(behkn,13,13)}$	<i>base case</i>	0
$F_{(behkn,13,11)}$	<i>base case</i>	0
$G_{(behkn,29,3)}$	$G_{(behkn,28,6)} + F_{(behkn,28,5)} + F_{(behkn,28,7)} + G_{(behkn,25,6)} + F_{(behkn,25,7)} + F_{(behkn,25,5)} + G_{(behkn,21,9)} + F_{(behkn,21,10)} + F_{(behkn,21,8)} + G_{(behkn,13,12)} + F_{(behkn,13,13)} + F_{(behkn,13,11)}$	1
$F_{(behkn,29,4)}$	$memoF_{(behkn,29,4)}$	0

Tabel D.62 Simulasi perhitungan jumlah kombinasi *string orig1* dengan operasi *replace* dengan *dist* = 5 pada kasus *string ad1* = *kbenh*, *string ad2* = *kbenh* dan *X* = 5 (5)

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,28,5)}$	$memoF_{(behkn,28,5)}$	1
$F_{(behkn,25,5)}$	$memoF_{(behkn,25,5)}$	0
$F_{(behkn,21,8)}$	<i>base case</i>	0
$F_{(behkn,13,11)}$	<i>base case</i>	0
$F_{(behkn,29,2)}$	$F_{(behkn,28,5)} + F_{(behkn,25,5)} + F_{(behkn,21,8)} + F_{(behkn,13,11)}$	1
$G_{(behkn,27,6)}$	<i>base case</i>	0
$F_{(behkn,27,7)}$	<i>base case</i>	0
$F_{(behkn,26,8)}$	<i>base case</i>	0
$F_{(behkn,25,5)}$	$memoF_{(behkn,25,5)}$	0
$F_{(behkn,19,11)}$	<i>base case</i>	0
$F_{(behkn,11,14)}$	<i>base case</i>	0
$F_{(behkn,27,5)}$	$F_{(behkn,26,8)} + F_{(behkn,25,5)} + F_{(behkn,19,11)} + F_{(behkn,11,14)}$	0
$G_{(behkn,23,9)}$	<i>base case</i>	0
$F_{(behkn,23,10)}$	<i>base case</i>	0
$F_{(behkn,23,8)}$	<i>base case</i>	0
$G_{(behkn,15,12)}$	<i>base case</i>	0
$F_{(behkn,15,13)}$	<i>base case</i>	0
$F_{(behkn,15,11)}$	<i>base case</i>	0
$G_{(behkn,31,0)}$	$G_{(behkn,30,0)} + F_{(behkn,30,1)} + F_{(behkn,30,1)} + G_{(behkn,29,3)} + F_{(behkn,29,4)} + F_{(behkn,29,2)} + G_{(behkn,27,6)} + F_{(behkn,27,7)} + F_{(behkn,27,5)} + G_{(behkn,23,9)} + F_{(behkn,23,10)} + F_{(behkn,23,8)} + G_{(behkn,15,12)} + F_{(behkn,15,13)} + F_{(behkn,15,11)}$	8

Tabel D.63 Simulasi perhitungan jumlah kombinasi *string orig2* tanpa operasi *replace* dengan *dist* = 5 pada kasus *string ad1 = kbenh, string ad2 = kbenh* dan *X* = 5

Fungsi	Perhitungan Nilai	Nilai
$F_{(behkn,30,5)}$	$memoF_{(behkn,30,5)}$	1
$F_{(behkn,29,8)}$	<i>base case</i>	0
$F_{(behkn,27,11)}$	<i>base case</i>	0
$F_{(behkn,23,14)}$	<i>base case</i>	0
$F_{(behkn,15,17)}$	<i>base case</i>	0
$F_{(behkn,31,5)}$	$F_{(behkn,30,5)} + F_{(behkn,29,8)} + F_{(behkn,27,11)} + F_{(behkn,23,14)} + F_{(behkn,15,17)}$	1

BIODATA PENULIS



Dewangga Winasforcepta Winardi, lahir di Surabaya tanggal 18 Mei 1995. Penulis merupakan anak kedua dari 4 bersaudara. Penulis telah menempuh pendidikan formal TK Aisyiyah Bustanul Athfal Denpasar, SD Negeri 5 Ubung (2001-2007), SMP Negeri 5 Denpasar (2007-2010) dan SMA Negeri 4 Denpasar (2010-2013). Penulis melanjutkan studi kuliah program sarjana di Jurusan Teknik Informatika ITS.

Selama kuliah di Teknik Informatika ITS, penulis mengambil bidang minat Algoritma Pemrograman (AP). Penulis pernah menjadi asisten dosen dan praktikum untuk mata kuliah Dasar Pemrograman (2015 dan 2016), Struktur data (2015 dan 2016). Selama menempuh perkuliahan penulis juga aktif mengikuti kompetisi pemrograman tingkat nasional dan menjadi Juara 2 kategori pemrograman pada lomba COMPFEST Universitas Indonesia 2014. Selain itu penulis juga aktif di kegiatan organisasi dan kepanitiaan diantaranya menjadi staff Departemen Riset dan Teknologi HMTC ITS, wakil ketua National Programming Contest Schematic 2014, ketua National Programming Contest 2014, panitia Pemusatan Latihan Nasional 2 TOKI 2014, 2015, 2016 dan 2017 di ITS dan technical committee Olimpiade Sains Nasional 2015 di Jogjakarta. Penulis dapat dihubungi melalui surel di dewangga.winardi@gmail.com.