

TUGAS AKHIR - KI141502

**DESAIN, ANALISIS DAN IMPLEMENTASI ALGORITMA KOM- PUTASI STRING DENGAN METODE DYNAMIC PROGRAM- MING 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

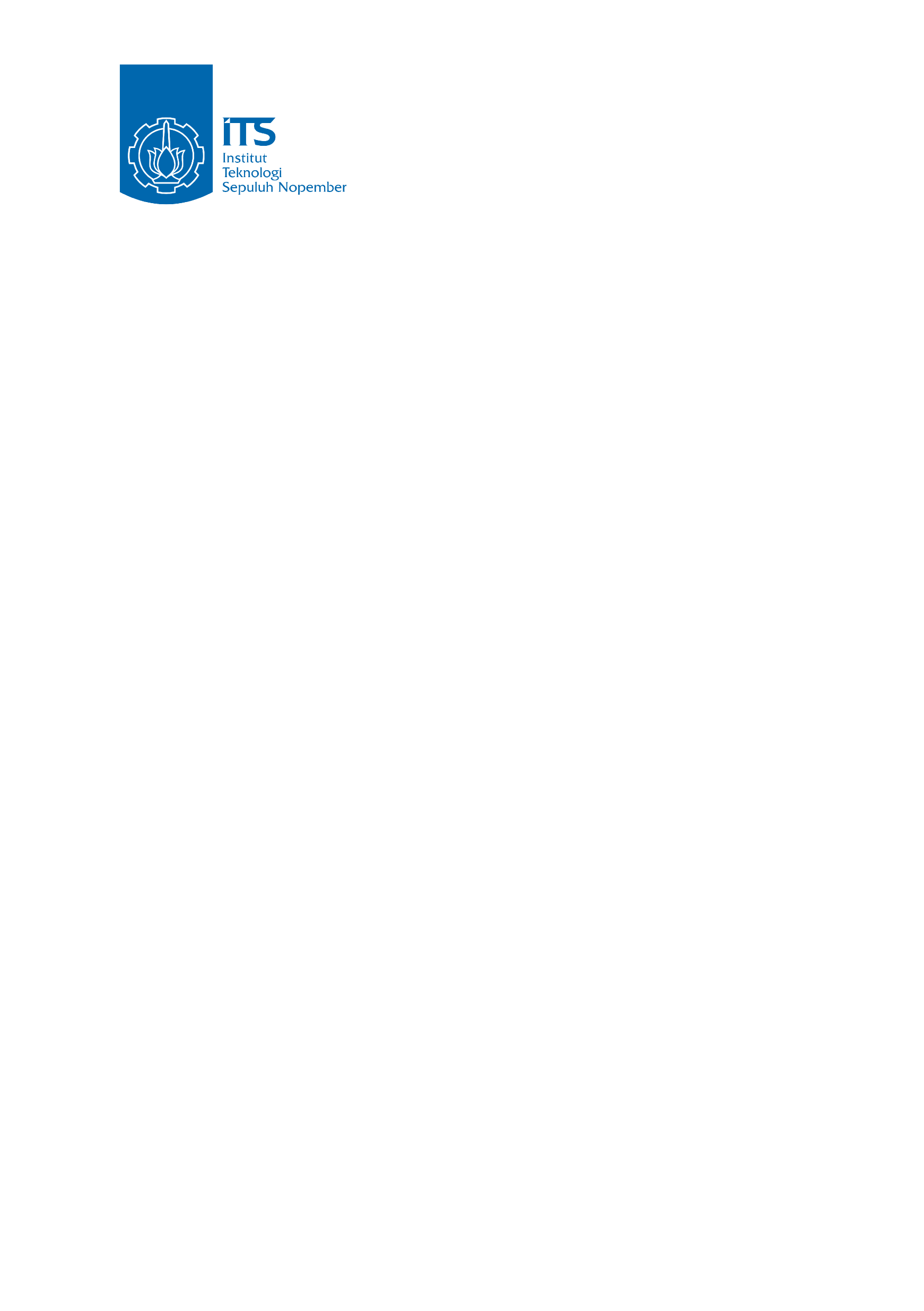
Ir. F.X. Arunanto, M.Sc.

JURUSAN TEKNIK INFORMATIKA

Fakultas Teknologi Informasi

Institut Teknologi Sepuluh Nopember Surabaya, 2017

*Halaman ini sengaja dikosongkan*



TUGAS AKHIR - KI141502

**DESAIN, ANALISIS DAN IMPLEMENTASI ALGORITMA KOM- PUTASI STRING DENGAN METODE DYNAMIC PROGRAM- MING 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

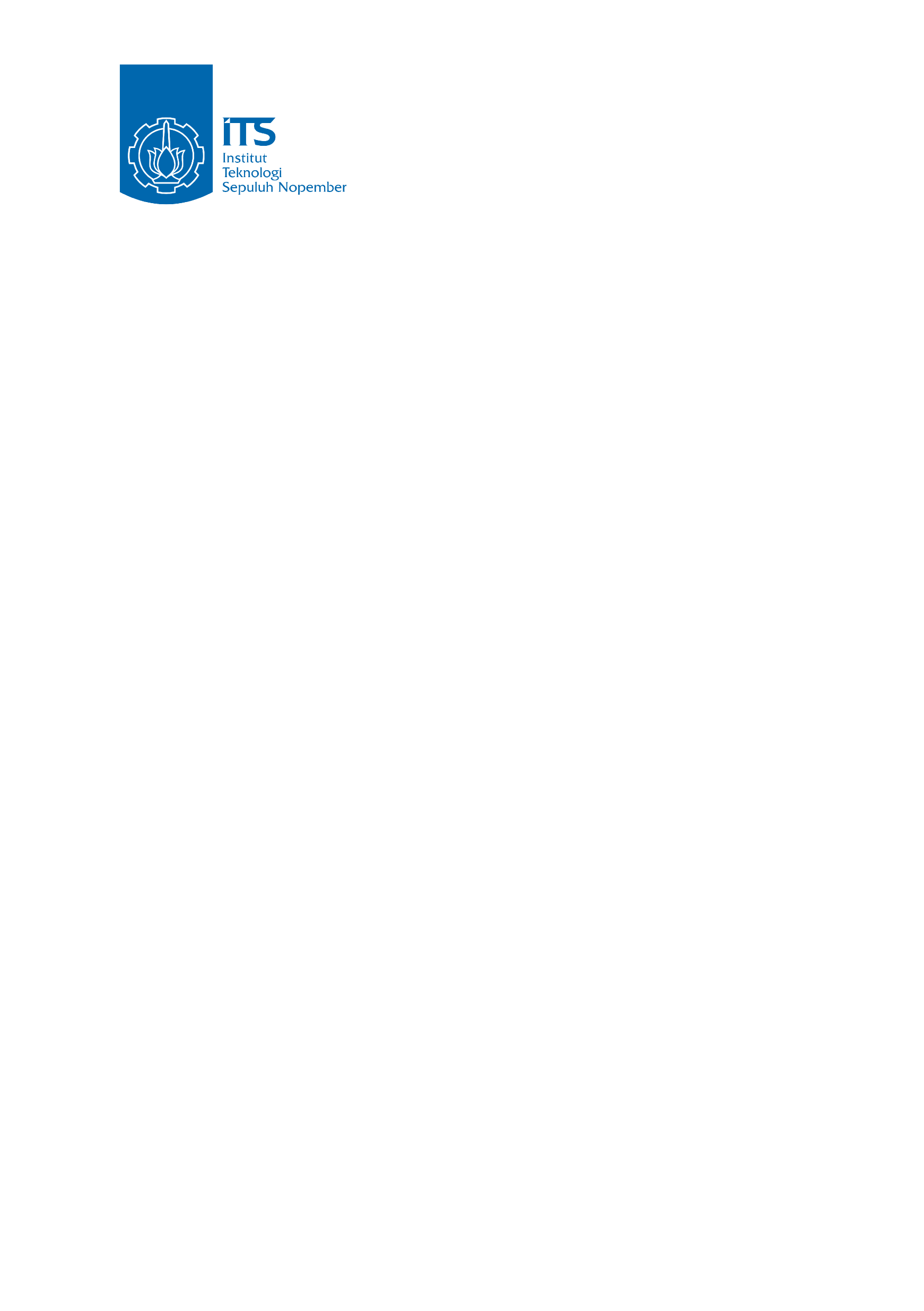
Ir. F.X. Arunanto, M.Sc.

JURUSAN 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 PROGRAM- MING 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*

**WORDS**

**TUGAS AKHIR**

Diajukan Guna Memenuhi Salah Satu Syarat Memperoleh Gelar Sarjana Komputer

pada

Bidang Studi Algoritma Pemrograman Program Studi S-1 Jurusan 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**

vii

*Halaman ini sengaja dikosongkan*

**WORDS**

Nama : DEWANGGA WINASFORCEPTA WINARDI NRP : 5113 100 098

Jurusan : 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 orig*1 *dan orig*2*. Diberikan tiga tahapan proses enkripsi untuk menghasilkan string ad*1 *dan ad*2*. Tahap pertama adalah string orig*1 *diacak sususan karakter- karakternya. Tahap kedua adalah string orig*2 *diacak susunan karakter-karakternya. Tahap terakhir adalah salah satu karakter dari string orig*1 *atau orig*2 *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 orig*1 *dan string ad*1 *dijumlahkan dengan jarak dari string orig*2 *dan string ad*2*. Tentukan jumlah kemungkinan kombinasi string orig*1 *dan orig*2 *jika diberikan string ad*1*, ad*2 *dan nilai X.*

*Meet in the middle adalah sebuah teknik pencarian dengan para- digma divide and conquer yang membagi permasalahan menjadi dua, lalu menyelesaikannya masing-masing, lalu menggabungkan- nya kembali.*

*Dynamic programming adalah sebuah paradigma untuk menda- patkan nilai optimal dari beberapa kemungkinan jawaban, dima- na permasalahan tersebut memiliki submasalah tumpang tindih dan*

ix

*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*

(2*|S| M AX\_DIST* 2 *T* )*, dimana S adalah panjang string yang diberikan dan M AX\_DIST adalah jarak antar string maksimal.*

O ∗ ∗ | |

**Kata Kunci: string, divide and conquer, meet in the mi- ddle, dynamic programming**

**ALGORITHM DESIGN, ANALYSIS AND IMPLEMENTA- TION FOR STRING COMPUTATION USING DYNAMIC PROGRAMMING AND MEET IN THE MIDDLE TECHNI- QUE 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 orig*1 *and orig*2*. Given three steps to encrypt orig*1 *and orig*2 *to become ad*1 *and ad*2*. First step is shuffle the letters position of string orig*1*. The Second step is shuffle the letters position of string orig*2*. The last step is replace one letter from string orig*1 *or orig*2 *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*(*orig*1*, ad*1) + *distance*(*orig*2*, ad*2)*. Find the number of possible string orig*1 *and orig*2 *from given string ad*1*, ad*2 *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 ans- wer.*

*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*

xi

*the middle and dynamic programming.*

*The solution will run in* (2*|S| M AX\_DIST* 2 *T* ) *time comple- xity where S is the given string ad*1 *and ad*2 *and M AX\_DIST is the maximum distance between given strings.*

| |

O ∗ ∗

**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 selesainya Tugas Akhir ini diharapkan apa yang telah diker- jakan 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 te- lah memberikan dukungan baik secara langsung maupun tidak lang- sung selama penulis mengerjakan Tugas Akhir maupun selama me- nempuh masa studi antara lain:

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

xiii

* Bapak Rully Soelaiman, S.Kom., M.Kom. selaku Dosen Pembimbing yang telah banyak meluangkan waktu untuk memberikan ilmu, nasihat, motivasi, pandangan dan bim- bingan kepada penulis baik selama penulis menempuh masa kuliah maupun selama pengerjaan 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 La- boratorium Pemrograman yang selalu menjadi teman untuk berbagi ilmu.
* Seluruh tenaga pengajar dan karyawan Jurusan Teknik Infor- matika 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 pe- nulis 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](#_bookmark0) ………………………………………………...……….. i [LEMBAR PENGESAHAN](#_bookmark0) vii

[ABSTRAK](#_bookmark0) ix

[ABSTRACT](#_bookmark0) xi

[KATA PENGANTAR](#_bookmark1) xiii

[DAFTAR ISI](#_bookmark2) xv

[DAFTAR TABEL](#_bookmark3) xix

[DAFTAR GAMBAR](#_bookmark4) xxxiii

[DAFTAR KODE SUMBER](#_bookmark5) xxxv

[BAB I PENDAHULUAN](#_bookmark6) 1

* 1. [Latar Belakang](#_bookmark7) 1
  2. [Rumusan Masalah](#_bookmark8) 2
  3. [Batasan Masalah](#_bookmark9) 2
  4. [Tujuan](#_bookmark10) 3
  5. [Manfaat](#_bookmark11) 3
  6. [Metodologi](#_bookmark12) 3
  7. [Sistematika Penulisan](#_bookmark13) 4

[BAB II DASAR TEORI](#_bookmark14) 7

* 1. [Deskripsi Permasalahan](#_bookmark15) 7
  2. [Deskripsi Umum](#_bookmark17) 8
     1. [String](#_bookmark18) 8
     2. [Rekurens](#_bookmark19) 10
     3. [Divide and Conquer](#_bookmark21) 10
     4. [Meet In The Middle](#_bookmark22) 10
     5. [Dynamic Programming](#_bookmark23) 10
     6. [State](#_bookmark24) 11
     7. [Bitmask](#_bookmark25) 11
  3. [Analisa Submasalah Optimal](#_bookmark26) 11

xv

* + 1. [Membagi Permasalah Menjadi Submasalah](#_bookmark27)

[yang *Independent*](#_bookmark27)11

* + 1. [Submasalah Optimal untuk Menghitung Jumlah Kombinasi String *Orig* dari String](#_bookmark36)

[*Ad* Tanpa Operasi *Replace* dengan Jarak *D*](#_bookmark36)17

* + 1. [Submasalah Optimal untuk Menghitung Jumlah Kombinasi String *Orig* dari String](#_bookmark38)

[*Ad* dengan Operasi *Replace* dengan Jarak *D*](#_bookmark38)19

* 1. [Pemodelan Relasi Rekurens](#_bookmark42) 23
     1. [Pemodelan Relasi Rekurens Submasalah Optimal untuk Menghitung Jumlah Ke- mungkinan String Awal Tanpa Operasi *Re-*](#_bookmark48)

[*place* dengan Jarak *X dist*](#_bookmark48)28

[−](#_bookmark48)

* + 1. [Pemodelan Relasi Rekurens Submasalah Optimal untuk Menghitung Jumlah Ke- mungkinan String Awal dengan Sekali Operasi *Replace* dengan Jarak *X dist*](#_bookmark53) . . 36

[−](#_bookmark53)

[BAB III DESAIN](#_bookmark62) 49

* 1. [Desain Umum Sistem](#_bookmark63) 49
  2. [Desain Fungsi Preprocess](#_bookmark64) 49
  3. [Desain Fungsi Init](#_bookmark67) 50
  4. [Desain Fungsi Solve](#_bookmark70) 53
     1. [Desain Fungsi F](#_bookmark72) 54
     2. [Desain Fungsi G](#_bookmark77) 57

[BAB IV IMPLEMENTASI](#_bookmark90) 69

* 1. [Lingkungan Implementasi](#_bookmark91) 69
  2. [Rancangan Data](#_bookmark92) 69
     1. [Data Masukan](#_bookmark93) 70
     2. [Data Keluaran](#_bookmark94) 70
  3. [Implementasi Algoritma](#_bookmark95) 70
     1. [Header-Header yang Diperlukan](#_bookmark96) 70
     2. [Variabel Global](#_bookmark98) 71
     3. [Implementasi Fungsi Main](#_bookmark100) 72

xvii

* + 1. [Implementasi Fungsi Preprocess](#_bookmark102) 72
    2. [Implementasi Fungsi ReadInput](#_bookmark104) 73
    3. [Implementasi Fungsi Init](#_bookmark106) 73
    4. [Implementasi Fungsi Solve](#_bookmark108) 74
    5. [Implementasi Fungsi F](#_bookmark110) 75
    6. [Implementasi Fungsi F1](#_bookmark113) 76
    7. [Implementasi Fungsi G](#_bookmark115) 77
    8. [Implementasi Fungsi G1](#_bookmark117) 77
    9. [Implementasi Fungsi G2](#_bookmark120) 78
    10. [Implementasi Fungsi G3](#_bookmark122) 79
    11. [Implementasi Fungsi Duplicate Rule 1](#_bookmark124) …………79
    12. [Implementasi Fungsi Duplicate Rule 2](#_bookmark127) ………....80
    13. [Implementasi Fungsi Duplicate Rule 3](#_bookmark129) …………81

[**BAB V UJI COBA DAN EVALUASI**](#_bookmark131) **83**

* 1. [Lingkungan Uji Coba](#_bookmark132) 83
  2. [Uji Coba Kebenaran](#_bookmark133) 83
  3. [Analisa Kompleksitas Waktu](#_bookmark135) 88

[**BAB VI KESIMPULAN**](#_bookmark136) **91**

* 1. [Kesimpulan](#_bookmark137) 91
  2. [Saran](#_bookmark138) 91

[**DAFTAR PUSTAKA**](#_bookmark139) **93**

[**LAMPIRAN A Tabel Himpunan** *setBit* **Setelah Fungsi Preprocess Dijalankan**](#_bookmark144) **95**

[**LAMPIRAN B Hasil Uji Coba Kebenaran pada Situs SPOJ**](#_bookmark180) **…………………………………………………………………….131**

[**LAMPIRAN C Tabel Simulasi Perhitungan Jumlah**](#_bookmark186)[**Kemungkinan String orig1 dan orig2 pada kasus string ad1=c, string ad2=n dan X=1**](#_bookmark186) **....………………………………..135**

[**Lampiran D Tabel Simulasi Perhitungan Jumlah**](#_bookmark195)[**Kemungkinan String orig1 dan orig2 pada kasus string ad1=kbenh, string ad2=kbenh dan X=5**](#_bookmark195)**…………………..........................................139**

[**BIODATA PENULIS**](#_bookmark259) **199**

*Halaman ini sengaja dikosongkan*

# DAFTAR TABEL

[Tabel 2.1 Kombinasi string *orig*1 dan *orig*2 dengan](#_bookmark16)

[*ad*1 = *c*, string *ad*2 = *n* dan *X* = 1](#_bookmark16) 8

[Tabel 2.2 Kombinasi string *orig*1 dan *orig*2 dengan](#_bookmark20)

[*ad*1 = *bd*, string *ad*2 = *gj* dan *X* = 5](#_bookmark20) 9

[Tabel 2.3 Kombinasi string *orig*1 dengan nilai string](#_bookmark30)

*[ad](#_bookmark30)*[1 =](#_bookmark30) *[bd](#_bookmark30)* [tanpa operasi](#_bookmark30) *[replace](#_bookmark30)* 15

[Tabel 2.4 Kombinasi string *orig*2 dengan nilai string](#_bookmark31)

[*ad*2 = *gj* dengan operasi *replace*](#_bookmark31)15

[Tabel 2.5 Kombinasi string *orig*1 dengan nilai string](#_bookmark32) [*ad*1 = *bd* tanpa operasi *replace* dan string](#_bookmark32) [*orig*2 dengan nilai string *ad*2 = *gj* dengan](#_bookmark32)

[operasi *replace* dengan *X* = 5](#_bookmark32) 16

[Tabel 2.6 Kombinasi string *orig*1 dengan nilai string](#_bookmark33)

[*ad*1 = *bd* dengan operasi *replace*](#_bookmark33)16

[Tabel 2.7 Kombinasi string *orig*2 dengan nilai string](#_bookmark34)

[*ad*2 = *gj* tanpa operasi *replace*](#_bookmark34)16

[Tabel 2.8 Kombinasi string *orig*1 dengan nilai string](#_bookmark35) [*ad*1 = *bd* dengan operasi *replace* dan string](#_bookmark35) [*orig*2 dengan nilai string *ad*2 = *gj* tanpa ope-](#_bookmark35)

[rasi *replace* dengan *X* = 5](#_bookmark35) 17

[Tabel 2.9 Daftar notasi persamaan 2.4.1](#_bookmark47) 28

[Tabel 2.10 Daftar notasi persamaan 2.4.2, 2.4.3 dan 2.4.4](#_bookmark50) 29

[Tabel 2.11 Daftar notasi persamaan 2.4.5, 2.4.6, 2.4.7,](#_bookmark54)

[2.4.8, 2.4.9 dan 2.4.10 (1)](#_bookmark54) 35

[Tabel 2.12 Daftar notasi persamaan 2.4.5, 2.4.6, 2.4.7,](#_bookmark55)

[2.4.8, 2.4.9 dan 2.4.10 (2)](#_bookmark55) 36

xix

[Tabel 3.1 Hasil *charF irstP os*(*S,C*) dan](#_bookmark68) [*charLastP os*(*S,C*) dengan string](#_bookmark68) [*S* = ”*inicontoh*” setelah fungsi init di-](#_bookmark68)

[jalankan](#_bookmark68) 52

[Tabel 3.2 Simulasi fungsi *F* dengan *S* = ”*kbenh*”, *X* =](#_bookmark73)

[5 dan *dist* = 5](#_bookmark73) 55

[Tabel 3.3 Simulasi fungsi *G* dengan *S* = ”*kbenh*”, *X* =](#_bookmark78)

[5 dan *dist* = 0 (1)](#_bookmark78) 59

[Tabel 3.4 Simulasi fungsi *G* dengan *S* = ”*kbenh*”, *X* =](#_bookmark79)

[5 dan *dist* = 0 (2)](#_bookmark79) 60

[Tabel 3.5 Simulasi fungsi *G* dengan *S* = ”*kbenh*”, *X* =](#_bookmark80)

[5 dan *dist* = 0 (3)](#_bookmark80) 61

[Tabel 3.6 Simulasi fungsi *G* dengan *S* = ”*kbenh*”, *X* =](#_bookmark81)

[5 dan *dist* = 0 (4)](#_bookmark81) 62

[Tabel 3.7 Simulasi fungsi *G* dengan *S* = ”*kbenh*”, *X* =](#_bookmark82)

[5 dan *dist* = 0 (5)](#_bookmark82) 63

[Tabel 3.8 Simulasi fungsi *G* dengan *S* = ”*kbenh*”, *X* =](#_bookmark83)

[5 dan *dist* = 0 (6)](#_bookmark83) 64

[Tabel 5.1 Kecepatan maksimal, minimal dan rata-rata](#_bookmark134) [dari hasil uji coba pengumpulan 30 kali pada](#_bookmark134)

[situs pengujian daring SPOJ](#_bookmark134) 84

[Tabel A.1 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark145)

[cess dijalankan (1)](#_bookmark145) 95

[Tabel A.2 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark146)

[cess dijalankan (2)](#_bookmark146) 96

[Tabel A.3 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark147)

[cess dijalankan (3)](#_bookmark147) 97

[Tabel A.4 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark148)

[cess dijalankan (4)](#_bookmark148) 98

[Tabel A.5 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark149)

[cess dijalankan (5)](#_bookmark149) 99

[Tabel A.6 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark150)

[cess dijalankan (6)](#_bookmark150) 100

[Tabel A.7 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark151)

[cess dijalankan (7)](#_bookmark151) 101

[Tabel A.8 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark152)

[cess dijalankan (8)](#_bookmark152) 102

[Tabel A.9 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark153)

[cess dijalankan (9)](#_bookmark153) 103

[Tabel A.10 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark154)

[cess dijalankan (10)](#_bookmark154) 104

[Tabel A.11 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark155)

[cess dijalankan (11)](#_bookmark155) 105

[Tabel A.12 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark156)

[cess dijalankan (12)](#_bookmark156) 106

[Tabel A.13 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark157)

[cess dijalankan (13)](#_bookmark157) 107

[Tabel A.14 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark158)

[cess dijalankan (14)](#_bookmark158) 108

[Tabel A.15 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark159)

[cess dijalankan (15)](#_bookmark159) 109

[Tabel A.16 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark160)

[cess dijalankan (16)](#_bookmark160) 110

[Tabel A.17 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark161)

[cess dijalankan (17)](#_bookmark161) 111

[Tabel A.18 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark162)

[cess dijalankan (18)](#_bookmark162) 112

[Tabel A.19 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark163)

[cess dijalankan (19)](#_bookmark163) 113

[Tabel A.20 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark164)

[cess dijalankan (20)](#_bookmark164) 114

[Tabel A.21 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark165)

[cess dijalankan (21)](#_bookmark165) 115

[Tabel A.22 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark166)

[cess dijalankan (22)](#_bookmark166) 116

[Tabel A.23 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark167)

[cess dijalankan (23)](#_bookmark167) 117

[Tabel A.24 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark168)

[cess dijalankan (24)](#_bookmark168) 118

[Tabel A.25 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark169)

[cess dijalankan (25)](#_bookmark169) 119

[Tabel A.26 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark170)

[cess dijalankan (26)](#_bookmark170) 120

[Tabel A.27 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark171)

[cess dijalankan (27)](#_bookmark171) 121

[Tabel A.28 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark172)

[cess dijalankan (28)](#_bookmark172) 122

[Tabel A.29 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark173)

[cess dijalankan (29)](#_bookmark173) 123

[Tabel A.30 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark174)

[cess dijalankan (30)](#_bookmark174) 124

[Tabel A.31 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark175)

[cess dijalankan (31)](#_bookmark175) 125

[Tabel A.32 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark176)

[cess dijalankan (32)](#_bookmark176) 126

[Tabel A.33 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark177)

[cess dijalankan (33)](#_bookmark177) 127

[Tabel A.34 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark178)

[cess dijalankan (34)](#_bookmark178) 128

[Tabel A.35 Tabel himpunan setBit setelah fungsi prepro-](#_bookmark179)

[cess dijalankan (35)](#_bookmark179) 129

[Tabel C.1 Simulasi perhitungan jumlah kombinasi string](#_bookmark187) [*orig*1 tanpa operasi *replace* dengan *dist* = 0](#_bookmark187) [pada kasus string *ad*1 = *c*, string *ad*2 = *n* dan](#_bookmark187)

[*X* = 1](#_bookmark187) 135

|  |  |  |
| --- | --- | --- |
| [Tabel C.2](#_bookmark188) | [Simulasi perhitungan jumlah kombinasi string](#_bookmark188)  [*orig*2 dengan operasi *replace* dengan *dist* = 0](#_bookmark188)  [pada kasus string *ad*1 = *c*, string *ad*2 = *n* dan](#_bookmark188) |  |
| [Tabel C.3](#_bookmark189) | [*X* = 1](#_bookmark188) . . . . . . . . . . . . . . . . . . . . .  [Simulasi perhitungan jumlah kombinasi string](#_bookmark189) | 135 |
|  | [*orig*1 dengan operasi *replace* dengan *dist* = 0](#_bookmark189)  [pada kasus string *ad*1 = *c*, string *ad*2 = *n* dan](#_bookmark189)  [*X* = 1](#_bookmark189) . . . . . . . . . . . . . . . . . . . . . | 136 |
| [Tabel C.4](#_bookmark190) | [Simulasi perhitungan jumlah kombinasi string](#_bookmark190) |  |
|  | [*orig*2 tanpa operasi *replace* dengan *dist* = 0](#_bookmark190)  [pada kasus string *ad*1 = *c*, string *ad*2 = *n* dan](#_bookmark190)  [*X* = 1](#_bookmark190) . . . . . . . . . . . . . . . . . . . . . | 136 |
| [Tabel C.5](#_bookmark191) | [Simulasi perhitungan jumlah kombinasi string](#_bookmark191) |  |
|  | [*orig*1 tanpa operasi *replace* dengan *dist* = 1](#_bookmark191)  [pada kasus string *ad*1 = *c*, string *ad*2 = *n* dan](#_bookmark191)  [*X* = 1](#_bookmark191) . . . . . . . . . . . . . . . . . . . . . | 136 |
| [Tabel C.6](#_bookmark192) | [Simulasi perhitungan jumlah kombinasi string](#_bookmark192) |  |
|  | [*orig*2 dengan operasi *replace* dengan *dist* = 1](#_bookmark192)  [pada kasus string *ad*1 = *c*, string *ad*2 = *n* dan](#_bookmark192)  [*X* = 1](#_bookmark192) . . . . . . . . . . . . . . . . . . . . . | 137 |
| [Tabel C.7](#_bookmark193) | [Simulasi perhitungan jumlah kombinasi string](#_bookmark193) |  |
|  | [*orig*1 dengan operasi *replace* dengan *dist* = 1](#_bookmark193)  [pada kasus string *ad*1 = *c*, string *ad*2 = *n* dan](#_bookmark193)  [*X* = 1](#_bookmark193) . . . . . . . . . . . . . . . . . . . . . | 137 |
| [Tabel C.8](#_bookmark194) | [Simulasi perhitungan jumlah kombinasi string](#_bookmark194) |  |
|  | [*orig*2 tanpa operasi *replace* dengan *dist* = 1](#_bookmark194)  [pada kasus string *ad*1 = *c*, string *ad*2 = *n* dan](#_bookmark194)  [*X* = 1](#_bookmark194) . . . . . . . . . . . . . . . . . . . . . | 137 |
| [Tabel D.1](#_bookmark196) | [Simulasi perhitungan jumlah kombinasi string](#_bookmark196) |  |
|  | [*orig*1 tanpa operasi *replace* dengan *dist* = 0](#_bookmark196)  [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark196)  [*kbenh* dan *X* = 5](#_bookmark196) . . . . . . . . . . . . . . . | 139 |

|  |  |  |
| --- | --- | --- |
| [Tabel D.2](#_bookmark197) | [Simulasi perhitungan jumlah kombinasi string](#_bookmark197)  [*orig*1 tanpa operasi *replace* dengan *dist* = 0](#_bookmark197)  [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark197) |  |
| [Tabel D.3](#_bookmark198) | [*kbenh* dan *X* = 5](#_bookmark197) . . . . . . . . . . . . . . .  [Simulasi perhitungan jumlah kombinasi string](#_bookmark198) | 140 |
|  | [*orig*2 dengan operasi *replace* dengan *dist* = 0](#_bookmark198)  [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark198)  [*kbenh* dan *X* = 5 (1)](#_bookmark198) . . . . . . . . . . . . . | 141 |
| [Tabel D.4](#_bookmark199) | [Simulasi perhitungan jumlah kombinasi string](#_bookmark199) |  |
|  | [*orig*2 dengan operasi *replace* dengan *dist* = 0](#_bookmark199)  [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark199)  [*kbenh* dan *X* = 5 (2)](#_bookmark199) . . . . . . . . . . . . . | 142 |
| [Tabel D.5](#_bookmark200) | [Simulasi perhitungan jumlah kombinasi string](#_bookmark200) |  |
|  | [*orig*2 dengan operasi *replace* dengan *dist* = 0](#_bookmark200)  [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark200)  [*kbenh* dan *X* = 5 (3)](#_bookmark200) . . . . . . . . . . . . . | 143 |
| [Tabel D.6](#_bookmark201) | [Simulasi perhitungan jumlah kombinasi string](#_bookmark201) |  |
|  | [*orig*2 dengan operasi *replace* dengan *dist* = 0](#_bookmark201)  [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark201)  [*kbenh* dan *X* = 5 (4)](#_bookmark201) . . . . . . . . . . . . . | 144 |
| [Tabel D.7](#_bookmark202) | [Simulasi perhitungan jumlah kombinasi string](#_bookmark202) |  |
|  | [*orig*2 dengan operasi *replace* dengan *dist* = 0](#_bookmark202)  [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark202)  [*kbenh* dan *X* = 5 (5)](#_bookmark202) . . . . . . . . . . . . . | 145 |
| [Tabel D.8](#_bookmark203) | [Simulasi perhitungan jumlah kombinasi string](#_bookmark203) |  |
|  | [*orig*2 dengan operasi *replace* dengan *dist* = 0](#_bookmark203)  [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark203)  [*kbenh* dan *X* = 5 (6)](#_bookmark203) . . . . . . . . . . . . . | 146 |
| [Tabel D.9](#_bookmark204) | [Simulasi perhitungan jumlah kombinasi string](#_bookmark204) |  |
|  | [*orig*1 dengan operasi *replace* dengan *dist* = 0](#_bookmark204)  [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark204)  [*kbenh* dan *X* = 5 (1)](#_bookmark204) . . . . . . . . . . . . . | 147 |

[Tabel D.10 Simulasi perhitungan jumlah kombinasi string](#_bookmark205) [*orig*1 dengan operasi *replace* dengan *dist* = 0](#_bookmark205) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark205)

[*kbenh* dan *X* = 5 (2)](#_bookmark205) 148

[Tabel D.11 Simulasi perhitungan jumlah kombinasi string](#_bookmark206) [*orig*1 dengan operasi *replace* dengan *dist* = 0](#_bookmark206) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark206)

[*kbenh* dan *X* = 5 (3)](#_bookmark206) 149

[Tabel D.12 Simulasi perhitungan jumlah kombinasi string](#_bookmark207) [*orig*2 tanpa operasi *replace* dengan *dist* = 0](#_bookmark207) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark207)

[*kbenh* dan *X* = 5 (1)](#_bookmark207) 150

[Tabel D.13 Simulasi perhitungan jumlah kombinasi string](#_bookmark208) [*orig*2 tanpa operasi *replace* dengan *dist* = 0](#_bookmark208) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark208)

[*kbenh* dan *X* = 5 (2)](#_bookmark208) 151

[Tabel D.14 Simulasi perhitungan jumlah kombinasi string](#_bookmark209) [*orig*1 tanpa operasi *replace* dengan *dist* = 1](#_bookmark209) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark209)

[*kbenh* dan *X* = 5](#_bookmark209) 152

[Tabel D.15 Simulasi perhitungan jumlah kombinasi string](#_bookmark210) [*orig*2 dengan operasi *replace* dengan *dist* = 1](#_bookmark210) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark210)

[*kbenh* dan *X* = 5 (1)](#_bookmark210) 153

[Tabel D.16 Simulasi perhitungan jumlah kombinasi string](#_bookmark211) [*orig*2 dengan operasi *replace* dengan *dist* = 1](#_bookmark211) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark211)

[*kbenh* dan *X* = 5 (2)](#_bookmark211) 154

[Tabel D.17 Simulasi perhitungan jumlah kombinasi string](#_bookmark212) [*orig*2 dengan operasi *replace* dengan *dist* = 1](#_bookmark212) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark212)

[*kbenh* dan *X* = 5 (3)](#_bookmark212) 155

[Tabel D.18 Simulasi perhitungan jumlah kombinasi string](#_bookmark213) [*orig*2 dengan operasi *replace* dengan *dist* = 1](#_bookmark213) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark213)

[*kbenh* dan *X* = 5 (4)](#_bookmark213) 156

[Tabel D.19 Simulasi perhitungan jumlah kombinasi string](#_bookmark214) [*orig*2 dengan operasi *replace* dengan *dist* = 1](#_bookmark214) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark214)

[*kbenh* dan *X* = 5 (5)](#_bookmark214) 157

[Tabel D.20 Simulasi perhitungan jumlah kombinasi string](#_bookmark215) [*orig*1 dengan operasi *replace* dengan *dist* = 1](#_bookmark215) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark215)

[*kbenh* dan *X* = 5 (1)](#_bookmark215) 158

[Tabel D.21 Simulasi perhitungan jumlah kombinasi string](#_bookmark216) [*orig*1 dengan operasi *replace* dengan *dist* = 1](#_bookmark216) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark216)

[*kbenh* dan *X* = 5 (2)](#_bookmark216) 159

[Tabel D.22 Simulasi perhitungan jumlah kombinasi string](#_bookmark217) [*orig*1 dengan operasi *replace* dengan *dist* = 1](#_bookmark217) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark217)

[*kbenh* dan *X* = 5 (3)](#_bookmark217) 160

[Tabel D.23 Simulasi perhitungan jumlah kombinasi string](#_bookmark218) [*orig*2 tanpa operasi *replace* dengan *dist* = 1](#_bookmark218) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark218)

[*kbenh* dan *X* = 5](#_bookmark218) 160

[Tabel D.24 Simulasi perhitungan jumlah kombinasi string](#_bookmark219) [*orig*1 tanpa operasi *replace* dengan *dist* = 2](#_bookmark219) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark219)

[*kbenh* dan *X* = 5](#_bookmark219) 161

[Tabel D.25 Simulasi perhitungan jumlah kombinasi string](#_bookmark220) [*orig*2 dengan operasi *replace* dengan *dist* = 2](#_bookmark220) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark220)

[*kbenh* dan *X* = 5 (1)](#_bookmark220) 162

[Tabel D.26 Simulasi perhitungan jumlah kombinasi string](#_bookmark221) [*orig*2 dengan operasi *replace* dengan *dist* = 2](#_bookmark221) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark221)

[*kbenh* dan *X* = 5 (2)](#_bookmark221) 163

[Tabel D.27 Simulasi perhitungan jumlah kombinasi string](#_bookmark222) [*orig*2 dengan operasi *replace* dengan *dist* = 2](#_bookmark222) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark222)

[*kbenh* dan *X* = 5 (3)](#_bookmark222) 164

[Tabel D.28 Simulasi perhitungan jumlah kombinasi string](#_bookmark223) [*orig*2 dengan operasi *replace* dengan *dist* = 2](#_bookmark223) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark223)

[*kbenh* dan *X* = 5 (4)](#_bookmark223) 165

[Tabel D.29 Simulasi perhitungan jumlah kombinasi string](#_bookmark224) [*orig*2 dengan operasi *replace* dengan *dist* = 2](#_bookmark224) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark224)

[*kbenh* dan *X* = 5 (5)](#_bookmark224) 166

[Tabel D.30 Simulasi perhitungan jumlah kombinasi string](#_bookmark225) [*orig*1 dengan operasi *replace* dengan *dist* = 2](#_bookmark225) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark225)

[*kbenh* dan *X* = 5 (1)](#_bookmark225) 167

[Tabel D.31 Simulasi perhitungan jumlah kombinasi string](#_bookmark226) [*orig*1 dengan operasi *replace* dengan *dist* = 2](#_bookmark226) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark226)

[*kbenh* dan *X* = 5 (2)](#_bookmark226) 168

[Tabel D.32 Simulasi perhitungan jumlah kombinasi string](#_bookmark227) [*orig*1 dengan operasi *replace* dengan *dist* = 2](#_bookmark227) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark227)

[*kbenh* dan *X* = 5 (3)](#_bookmark227) 169

[Tabel D.33 Simulasi perhitungan jumlah kombinasi string](#_bookmark228) [*orig*2 tanpa operasi *replace* dengan *dist* = 2](#_bookmark228) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark228)

[*kbenh* dan *X* = 5](#_bookmark228) 170

[Tabel D.34 Simulasi perhitungan jumlah kombinasi string](#_bookmark229) [*orig*1 tanpa operasi *replace* dengan *dist* = 3](#_bookmark229) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark229)

[*kbenh* dan *X* = 5](#_bookmark229) 171

[Tabel D.35 Simulasi perhitungan jumlah kombinasi string](#_bookmark230) [*orig*2 dengan operasi *replace* dengan *dist* = 3](#_bookmark230) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark230)

[*kbenh* dan *X* = 5 (1)](#_bookmark230) 172

[Tabel D.36 Simulasi perhitungan jumlah kombinasi string](#_bookmark231) [*orig*2 dengan operasi *replace* dengan *dist* = 3](#_bookmark231) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark231)

[*kbenh* dan *X* = 5 (2)](#_bookmark231) 173

[Tabel D.37 Simulasi perhitungan jumlah kombinasi string](#_bookmark232) [*orig*2 dengan operasi *replace* dengan *dist* = 3](#_bookmark232) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark232)

[*kbenh* dan *X* = 5 (3)](#_bookmark232) 174

[Tabel D.38 Simulasi perhitungan jumlah kombinasi string](#_bookmark233) [*orig*1 dengan operasi *replace* dengan *dist* = 3](#_bookmark233) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark233)

[*kbenh* dan *X* = 5 (1)](#_bookmark233) 175

[Tabel D.39 Simulasi perhitungan jumlah kombinasi string](#_bookmark234) [*orig*1 dengan operasi *replace* dengan *dist* = 3](#_bookmark234) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark234)

[*kbenh* dan *X* = 5 (2)](#_bookmark234) 176

[Tabel D.40 Simulasi perhitungan jumlah kombinasi string](#_bookmark235) [*orig*1 dengan operasi *replace* dengan *dist* = 3](#_bookmark235) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark235)

[*kbenh* dan *X* = 5 (3)](#_bookmark235) 177

[Tabel D.41 Simulasi perhitungan jumlah kombinasi string](#_bookmark236) [*orig*1 dengan operasi *replace* dengan *dist* = 3](#_bookmark236) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark236)

[*kbenh* dan *X* = 5 (4)](#_bookmark236) 178

[Tabel D.42 Simulasi perhitungan jumlah kombinasi string](#_bookmark237) [*orig*1 dengan operasi *replace* dengan *dist* = 3](#_bookmark237) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark237)

[*kbenh* dan *X* = 5 (5)](#_bookmark237) 179

[Tabel D.43 Simulasi perhitungan jumlah kombinasi string](#_bookmark238) [*orig*2 tanpa operasi *replace* dengan *dist* = 3](#_bookmark238) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark238)

[*kbenh* dan *X* = 5](#_bookmark238) 179

[Tabel D.44 Simulasi perhitungan jumlah kombinasi string](#_bookmark239) [*orig*1 tanpa operasi *replace* dengan *dist* = 4](#_bookmark239) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark239)

[*kbenh* dan *X* = 5](#_bookmark239) 180

[Tabel D.45 Simulasi perhitungan jumlah kombinasi string](#_bookmark240) [*orig*2 dengan operasi *replace* dengan *dist* = 4](#_bookmark240) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark240)

[*kbenh* dan *X* = 5 (1)](#_bookmark240) 181

[Tabel D.46 Simulasi perhitungan jumlah kombinasi string](#_bookmark241) [*orig*2 dengan operasi *replace* dengan *dist* = 4](#_bookmark241) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark241)

[*kbenh* dan *X* = 5 (2)](#_bookmark241) 182

[Tabel D.47 Simulasi perhitungan jumlah kombinasi string](#_bookmark242) [*orig*2 dengan operasi *replace* dengan *dist* = 4](#_bookmark242) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark242)

[*kbenh* dan *X* = 5 (3)](#_bookmark242) 183

[Tabel D.48 Simulasi perhitungan jumlah kombinasi string](#_bookmark243) [*orig*1 dengan operasi *replace* dengan *dist* = 4](#_bookmark243) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark243)

[*kbenh* dan *X* = 5 (1)](#_bookmark243) 184

[Tabel D.49 Simulasi perhitungan jumlah kombinasi string](#_bookmark244) [*orig*1 dengan operasi *replace* dengan *dist* = 4](#_bookmark244) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark244)

[*kbenh* dan *X* = 5 (2)](#_bookmark244) 185

[Tabel D.50 Simulasi perhitungan jumlah kombinasi string](#_bookmark245) [*orig*1 dengan operasi *replace* dengan *dist* = 4](#_bookmark245) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark245)

[*kbenh* dan *X* = 5 (3)](#_bookmark245) 186

[Tabel D.51 Simulasi perhitungan jumlah kombinasi string](#_bookmark246) [*orig*1 dengan operasi *replace* dengan *dist* = 4](#_bookmark246) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark246)

[*kbenh* dan *X* = 5 (4)](#_bookmark246) 187

[Tabel D.52 Simulasi perhitungan jumlah kombinasi string](#_bookmark247) [*orig*1 dengan operasi *replace* dengan *dist* = 4](#_bookmark247) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark247)

[*kbenh* dan *X* = 5 (5)](#_bookmark247) 188

[Tabel D.53 Simulasi perhitungan jumlah kombinasi string](#_bookmark248) [*orig*2 tanpa operasi *replace* dengan *dist* = 4](#_bookmark248) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark248)

[*kbenh* dan *X* = 5](#_bookmark248) 188

[Tabel D.54 Simulasi perhitungan jumlah kombinasi string](#_bookmark249) [*orig*1 tanpa operasi *replace* dengan *dist* = 5](#_bookmark249) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark249)

[*kbenh* dan *X* = 5](#_bookmark249) 189

[Tabel D.55 Simulasi perhitungan jumlah kombinasi string](#_bookmark250) [*orig*2 dengan operasi *replace* dengan *dist* = 5](#_bookmark250) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark250)

[*kbenh* dan *X* = 5 (1)](#_bookmark250) 190

[Tabel D.56 Simulasi perhitungan jumlah kombinasi string](#_bookmark251) [*orig*2 dengan operasi *replace* dengan *dist* = 5](#_bookmark251) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark251)

[*kbenh* dan *X* = 5 (2)](#_bookmark251) 191

[Tabel D.57 Simulasi perhitungan jumlah kombinasi string](#_bookmark252) [*orig*2 dengan operasi *replace* dengan *dist* = 5](#_bookmark252) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark252)

[*kbenh* dan *X* = 5 (3)](#_bookmark252) 192

[Tabel D.58 Simulasi perhitungan jumlah kombinasi string](#_bookmark253) [*orig*1 dengan operasi *replace* dengan *dist* = 5](#_bookmark253) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark253)

[*kbenh* dan *X* = 5 (1)](#_bookmark253) 193

[Tabel D.59 Simulasi perhitungan jumlah kombinasi string](#_bookmark254) [*orig*1 dengan operasi *replace* dengan *dist* = 5](#_bookmark254) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark254)

[*kbenh* dan *X* = 5 (2)](#_bookmark254) 194

[Tabel D.60 Simulasi perhitungan jumlah kombinasi string](#_bookmark255) [*orig*1 dengan operasi *replace* dengan *dist* = 5](#_bookmark255) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark255)

[*kbenh* dan *X* = 5 (3)](#_bookmark255) 195

[Tabel D.61 Simulasi perhitungan jumlah kombinasi string](#_bookmark256) [*orig*1 dengan operasi *replace* dengan *dist* = 5](#_bookmark256) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark256)

[*kbenh* dan *X* = 5 (4)](#_bookmark256) 196

[Tabel D.62 Simulasi perhitungan jumlah kombinasi string](#_bookmark257) [*orig*1 dengan operasi *replace* dengan *dist* = 5](#_bookmark257) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark257)

[*kbenh* dan *X* = 5 (5)](#_bookmark257) 197

[Tabel D.63 Simulasi perhitungan jumlah kombinasi string](#_bookmark258) [*orig*2 tanpa operasi *replace* dengan *dist* = 5](#_bookmark258) [pada kasus string *ad*1 = *kbenh*, string *ad*2 =](#_bookmark258)

[*kbenh* dan *X* = 5](#_bookmark258) 198

*Halaman ini sengaja dikosongkan*

[Gambar 2.1 Ilustrasi umum penyelesaian permasalahan](#_bookmark28) [dengan metode *meet in the middle* tanpa ope-](#_bookmark28)

[rasi *replace*](#_bookmark28)12

[Gambar 2.2 Ilustrasi umum penyelesaian permasalahan dengan metode *meet in the middle* dengan operasi *replace*](#_bookmark29)14

[Gambar 2.3 Ilustrasi umum penyelesaian permasalahan](#_bookmark37) [dengan metode *meet in the middle* dengan](#_bookmark37) [operasi *replace* tanpa mempedulikan kombi-](#_bookmark37)

[nasi string yang dihasilkan](#_bookmark37) 18

[Gambar 2.4 Ilustrasi perhitungan jumlah kombinasi](#_bookmark39) [string *orig* dari string *ad* tanpa operasi](#_bookmark39) [*replace* dengan nilai string *ad* = *bcd* dan](#_bookmark39)

[*D* = 4](#_bookmark39) 20

[Gambar 2.5 Contoh kasus tumpang tindih pada perhi-](#_bookmark40) [tungan kombinasi string *orig* dari string *ad*](#_bookmark40)[tanpa operasi *replace* dengan nilai string](#_bookmark40)

[*ad* = *bcd* dan *D* = 4](#_bookmark40) 21

[Gambar 2.6 Ilustrasi perhitungan jumlah kombinasi](#_bookmark41) [string *orig* dari string *ad* tanpa operasi](#_bookmark41) [*replace* dengan nilai string *ad* = *bcd* dan](#_bookmark41) [*D* = 4 tanpa menghitung kasus yang](#_bookmark41)

[tumpang tindih](#_bookmark41) 22

[Gambar 2.7 Ilustrasi perhitungan jumlah kombinasi](#_bookmark44) [string *orig* dari string *ad* dengan operasi](#_bookmark44)

[*replace* dengan nilai string *ad* = *be* dan *D* = 1](#_bookmark44) 24

xxxiii

[Gambar 2.8 Contoh kasus tumpang tindih pada perhi-](#_bookmark45) [tungan kombinasi string *orig* dari string *ad*](#_bookmark45)[dengan operasi *replace* dengan nilai string](#_bookmark45)

[*ad* = *be* dan *D* = 1](#_bookmark45) 25

[Gambar 2.9 Submasalah perhitungan jumlah kombinasi](#_bookmark46) [string *orig* terhadap string *ad* tanpa operasi](#_bookmark46) [*replace* pada submasalah perhitungan jum-](#_bookmark46) [lah kombinasi string *orig* terhadap string *ad*](#_bookmark46)

[dengan operasi *replace*](#_bookmark46)26

[Gambar 3.1 Pseudocode Fungsi Main](#_bookmark65) 50

[Gambar 3.2 Pseudocode Fungsi Preprocess](#_bookmark66) 50

[Gambar 3.3 Pseudocode Fungsi Init](#_bookmark69) 53

[Gambar 3.4 Pseudocode Fungsi Solve](#_bookmark71) 54

[Gambar 3.5 Pseudocode Fungsi F](#_bookmark74) 56

[Gambar 3.6 Pseudocode Fungsi F1](#_bookmark75) 57

[Gambar 3.7 Pseudocode Fungsi duplicate\_rule1](#_bookmark76) 57

[Gambar 3.8 Pseudocode Fungsi G](#_bookmark84) 65

[Gambar 3.9 Pseudocode Fungsi G1](#_bookmark85) 65

[Gambar 3.10 Pseudocode Fungsi G2](#_bookmark86) 66

[Gambar 3.11 Pseudocode Fungsi G3](#_bookmark87) 66

[Gambar 3.12 Pseudocode Fungsi duplicate\_rule2](#_bookmark88) 66

[Gambar 3.13 Pseudocode Fungsi duplicate\_rule3](#_bookmark89) 67

[Gambar B.1 Hasil uji coba pada situs penilaian SPOJ](#_bookmark181) . . 131 [Gambar B.2 Hasil pengujian sebanyak 30 kali pada situs](#_bookmark182)

[penilaian daring SPOJ (1)](#_bookmark182) 131

[Gambar B.3 Hasil pengujian sebanyak 30 kali pada situs](#_bookmark183) [penilaian daring SPOJ (2)](#_bookmark183) 132

[Gambar B.4 Hasil pengujian sebanyak 30 kali pada situs](#_bookmark184) [penilaian daring SPOJ (3)](#_bookmark184) 132

[Gambar B.5 Grafik hasil uji coba pada situs SPOJ seba-](#_bookmark185)

[nyak 30 kali](#_bookmark185) 133

[Kode Sumber 4.3.1 Header yang diperlukan](#_bookmark97) 71

[Kode Sumber 4.3.2 Variabel global](#_bookmark99) 72

[Kode Sumber 4.3.3 Fungsi main](#_bookmark101) 72

[Kode Sumber 4.3.4 Fungsi preprocess](#_bookmark103) 73

[Kode Sumber 4.3.5 Fungsi readInput](#_bookmark105) 73

[Kode Sumber 4.3.6 Fungsi init](#_bookmark107) 74

[Kode Sumber 4.3.7 Fungsi solve](#_bookmark109) 75

[Kode Sumber 4.3.8 Fungsi F (1)](#_bookmark111) 75

[Kode Sumber 4.3.9 Fungsi F (2)](#_bookmark112) 76

[Kode Sumber 4.3.10 Fungsi F1](#_bookmark114) 76

[Kode Sumber 4.3.11 Fungsi G](#_bookmark116) 77

[Kode Sumber 4.3.12 Fungsi G1 (1)](#_bookmark118) 78

[Kode Sumber 4.3.13 Fungsi G1 (2)](#_bookmark119) 78

[Kode Sumber 4.3.14 Fungsi G2](#_bookmark121) 78

[Kode Sumber 4.3.15 Fungsi G3](#_bookmark123) 79

[Kode Sumber 4.3.16 Fungsi duplicate\_rule1 (1)](#_bookmark125) 80

[Kode Sumber 4.3.17 Fungsi duplicate\_rule1 (2)](#_bookmark126) 80

[Kode Sumber 4.3.18 Fungsi duplicate\_rule2](#_bookmark128) 80

[Kode Sumber 4.3.19 Fungsi duplicate\_rule3](#_bookmark130) 81

xxxv

*Halaman ini sengaja dikosongkan*

# BAB I PENDAHULUAN

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

# Latar Belakang

Tugas Akhir ini mengacu pada permasalahan klasik SPOJ 9967 *Pla- ying With Words*. Diberikan dua buah string *orig*1 dan *orig*2. Di- berikan tiga tahapan proses enkripsi untuk menghasilkan string *ad*1 dan *ad*2 sebagai berikut:

* + 1. String *orig*1 diacak urutan karakter-karakternya.
    2. String *orig*2 diacak urutan karakter-karakternya.
    3. Salah satu karakter dari string *orig*1 atau *orig*2 diganti de- ngan karakter sebelum atau sesudahnya dalam alfabet.

Jarak dua buah string didefinisikan sebagai jumlah dari selisih mut- lak dari karakter-karakter pada posisi yang sama. Diberikan sebuah bilangan bulat *X* yang merupakan jarak dari string *orig*1 dan string *ad*1 dijumlahkan dengan jarak dari string *orig*2 dan string *ad*2. Ber- apakah jumlah kemungkinan kombinasi string *orig*1 dan *orig*2 jika diberikan string *ad*1, *ad*2 dan nilai *X*.

Untuk menyelesaikan permasalahan di atas, penulis akan menggu- nakan pendekatan solusi dengan teknik *dynamic programming* dan *meet in the middle*. Selain dapat menjawab pertanyaan dengan be- nar, 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 1

mengenai performa algoritma dengan teknik *dynamic programming*

dan *meet in the middle*.

# Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah se- bagai 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 me- nyelesaikan permasalahan klasik SPOJ 9967 *Playing With* *Words*.

# Batasan Masalah

Permasalahan yang dibahas pada Tugas Akhir ini memiliki bebera- pa 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 *ad*1 dan *ad*2 maksimal bernilai 10.
    4. Karakter pada string masukan *ad*1 dan *ad*2 berada dalam ren- tang ‘*b*‘ *ad*1*i, ad*2*i* ‘*y*‘.

≤ ≤

* + 1. Nilai masukan *X* tidak melebihi 100000.
    2. Batas waktu eksekusi program adalah 6*.*469 detik.

# Tujuan

Tujuan dari Tugas Akhir ini adalah sebagai berikut:

* + 1. Menyelesaikan permasalahan klasik SPOJ 9967 *Playing With Words* dengan algoritma yang telah dirancang dan diimple- mentasikan menggunakan teknik *dynamic programming* dan *meet in the middle*.
    2. Melakukan uji kebenaran terhadap algoritma yang telah di- rancang dan diimplementasikan.
    3. Menganalisa performa dari algoritma yang dibangun.

# Manfaat

Manfaat dari Tugas Akhir ini adalah sebagai berikut:

* + 1. Mengetahui performa dari algoritma yang dirancang dengan teknik *meet in the middle* dan pendekatan *dynamic program- ming* untuk menyelesaikan permasalahan klasik SPOJ 9967 *Playing With Words*.

# Metodologi

Metodologi yang digunakan dalam pengerjaan Tugas Akhir ini ada- lah sebagai berikut:

* + 1. Penyusunan proposal Tugas Akhir

Pada tahap ini dilakukan penyusunan proposal Tugas Akhir yang berisi permasalahan dan gagasan solusi yang akan diteli- ti pada permasalahan klasik SPOJ 9967 *Playing With Words*.

* + 1. Studi literatur

Pada tahap ini dilakukan pencarian informasi dan studi litera- tur mengenai pengetahuan atau metode yang dapat digunak- an dalam penyelesaian masalah. Informasi didapatkan dari materi-materi yang berhubungan dengan algoritma yang di- gunakan untuk penyelesaian permasalahan ini, materi-materi

tersebut didapatkan dari buku, jurnal, maupun internet.

* + 1. Desain

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

* + 1. Implementasi perangkat lunak

Pada tahap ini dilakukan implementasi atau realiasi dari ran- cangan desain algoritma yang telah dibangun pada tahap de- sain ke dalam bentuk program.

* + 1. 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 apa- kah luaran dari program telah sesuai.

* + 1. Penyusunan buku Tugas Akhir

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

# Sistematika Penulisan

Berikut adalah sistematika penulisan buku Tugas Akhir ini:

* + 1. BABI: PENDAHULUAN

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

* + 1. BAB II: DASAR TEORI

Bab ini berisi dasar teori mengenai permasalahan dan algori- tma penyelesaian yang digunakan dalam Tugas Akhir

* + 1. BAB III: DESAIN

Bab ini berisi desain algoritma dan struktur data yang digu- nakan dalam penyelesaian permasalahan.

* + 1. BAB IV: IMPLEMENTASI

Bab ini berisi implementasi berdasarkan desain algortima

yang telah dilakukan pada tahap desain.

* + 1. BAB V: UJI COBA DAN EVALUASI

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

* + 1. 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 da- sar pengerjaan Tugas Akhir ini.

# Deskripsi Permasalahan

Amr M. sangat curiga dengan sebuah iklan yang terdiri dari 2 buah string *ad*1 dan *ad*2. Ia menduga kedua string tersebut menyimp- an pesan tersembunyi. Setelah menghubungi sumber yang terper- caya, ia menemukan proses untuk menyembunyikan pesan terse- but. Pesan asli yang dibawa selalu berupa dua buah string *orig*1 dan *orig*2. Berikut adalah langkah-langkah transformasi pesan asli menjadi pesan pada iklan:

* + 1. Karakter-karakter pada string *orig*1 diacak urutannya.
    2. Karakter-karakter pada string *orig*2 diacak urutannya.
    3. Salah satu karakter dari string *orig*1 atau *orig*2 diganti de- ngan karakter sebelum atau sesudahnya dalam alfabet.

Langkah-langkah di atas akan menghasilkan string *ad*1 dan *ad*2 dari *orig*1 dan *orig*2 secara berurutan. Contohnya untuk string *orig*1 = *bcd* dan *orig*2 = *wcy* dapat menghasilkan string *ad*1 = *dcb* dan *ad*2 = *cxy* di mana *cxy* berasal dari *wcy* yang diacak menjadi *cwy* dan karakter *w* digantikan dengan karakter *x*.

Setelah melakukan riset, Amr menemukan sebuah jarak *X*, di ma- na *X* adalah *jarak*(*orig*1 + *ad*1) ditambah dengan *jarak*(*orig*2 + *ad*2). Jarak antara dua string didefinisikan sebagai jumlah dari se- lisih absolut dari karakter-karakter pada posisi yang sama. Contoh- nya *jarak*(*ab, cd*) = |*a′′* −*′ c′*| + |*′b′* −*′ d′*| = 4. Diberikan string

7

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

Tabel 2.1 Kombinasi string *orig*1 dan *orig*2 dengan *ad*1 = *c*, string *ad*2 =

*n* dan *X* = 1

*ad*1, *ad*2 dan sebuah bilangan bulat *X*. Hitung jumlah kemungkin- an string *orig*1 dan *orig*2 yang mungkin.

Sebagai contoh, Tabel [2.1](#_bookmark16) adalah kombinasi dari string *orig*1 dan *orig*2 dengan string *ad*1 = *c*, string *ad*2 = *n* dan *X* = 1. Jawaban dari kasus tersebut adalah 4 karena terdapat 4 kombinasi dengan *X* = 1. Contoh berikutnya adalah kasus ketika *ad*1 = *bd*, string *ad*2 = *gj* dan *X* = 5. Berdasarkan daftar kombinasi string *orig*1 dan *orig*2 pada Tabel [2.2](#_bookmark20), terdapat 8 kombinasi string *orig*1 dan string *orig*2 yang memenuhi kriteria *X* = 5 sehingga jawaban akhir dari kasus tersebut adalah 8.

# Deskripsi Umum

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

# String

Pada dunia ilmu komputer, string didefinisikan sebagai sebuah rangkaian karakter. String pada umumnya dipahami sebagai sebu- ah struktur data dan diimplementasi menggunakan struktur data *ar- ray*[[1](#_bookmark140)].

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

Tabel 2.2 Kombinasi string *orig*1 dan *orig*2 dengan *ad*1 = *bd*, string

*ad*2 = *gj* dan *X* = 5

# Rekurens

Ketika sebuah algoritma mengandung sebuah persamaan rekursif yang memanggil dirinya sendiri, waktu prosesnya dapat dikatakan sebagai rekurens. Rekurens adalah sebuah persamaan atau pertidak- samaan yang mendeskripsikan sebuah fungsi dalam hal nilai pada masukan yang lebih kecil[[2](#_bookmark141)].

## Divide and Conquer

Dalam ilmu komputer, *divide and conquer* (D&C) adalah pa- radigma perancangan algoritma yang bekerja dengan memecah permasalahan menjadi dua atau lebih submasalah dengan karakte- ristik yang sama atau berkaitan hingga cukup sederhana untuk di- selesaikan 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 ha- nya satu submasalah[[3](#_bookmark142)].

## Meet In The Middle

Dalam dunia pemrograman komputer, *meet in the middle* adalah se- buah teknik pencarian dua arah dengan membagi dua permasalahan, lalu menyelesaikannya secara terpisah, lalu menggabungkan kedu- anya untuk mendapatkan hasil yang diinginkan[[3](#_bookmark142)].

## Dynamic Programming

Dalam dunia ilmu komputer, *dynamic programming* adalah sebuah metode penyelesaian masalah yang memecah sebuah permasalahan yang rumit menjadi submasalah-submasalah yang lebih sederha- na. *dynamic programming* bersifat efektif ketika submasalah dari permasalahan yang diberikan mungkin berasal dari lebih dari satu pilihan. Teknik kunci dari *dynamic programming* adalah menyimp-

an solusi untuk setiap submasalah untuk digunakan jika submasalah tersebut muncul kembali[[2](#_bookmark141)].

## State

*State* atau *state variable* adalah himpunan variabel parameter dari sebuah submasalah dari permasalahan yang [diberikan[4](#_bookmark143)].

## Bitmask

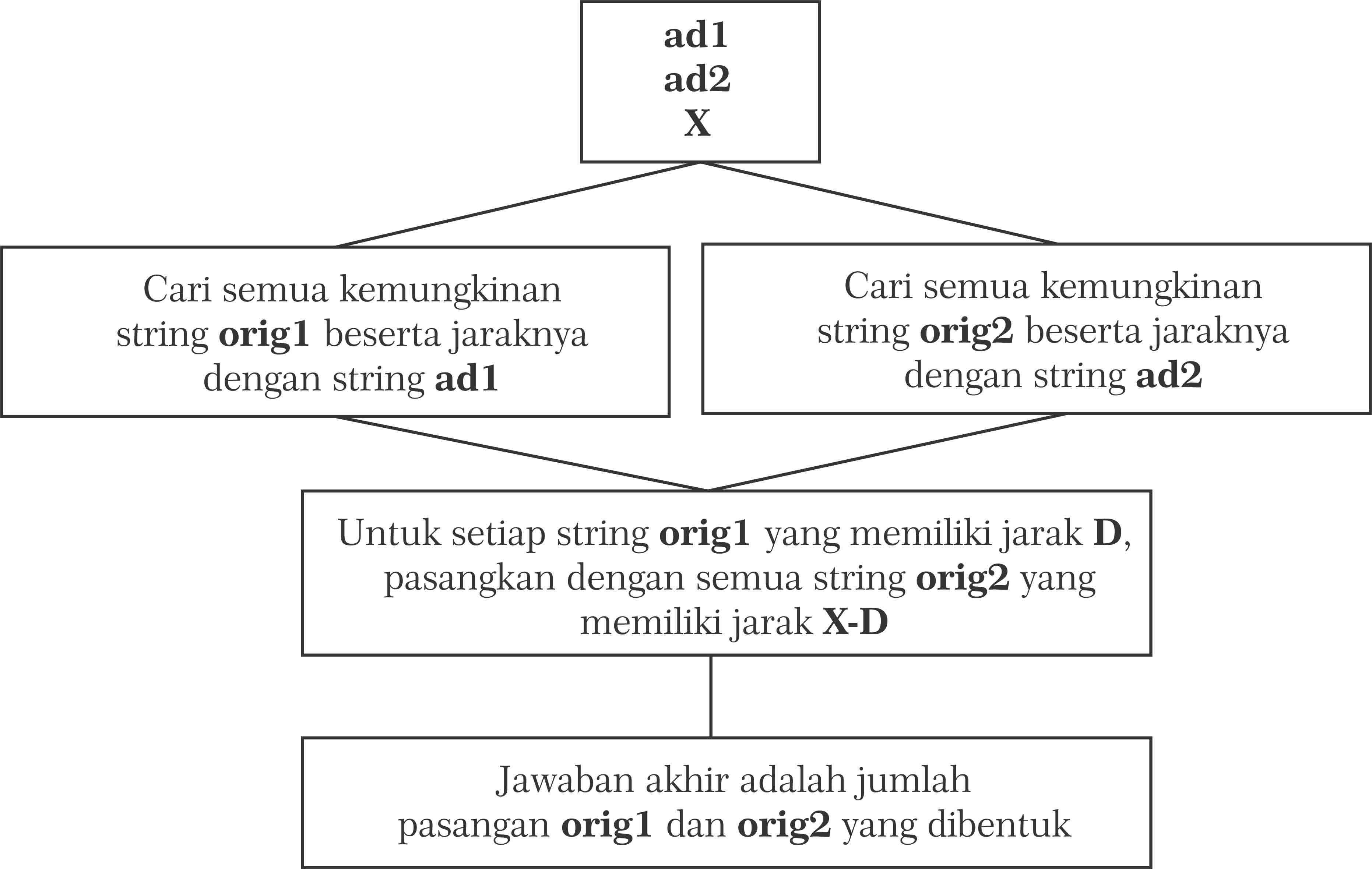
Bitmask adalah sebuah bilangan bulat yang disimpan dan direpre- sentasikan sebagai himpunan dari nilai *boolean*. Salah satu contoh pemanfaatan teknik *bitmasking* adalah penggunaan *bitmask* sebagai salah satu index pada tabel memo pada teknik *dynamic program- ming*[[3](#_bookmark142)].

# 2.3 Analisa Submasalah Optimal

Pada subbab ini akan dijelaskan mengenai submasalah-submasalah yang jawabannya dapat membangun jawaban akhir. Tujuan uta- ma dari permasalahan yang diberikan adalah untuk mencari jumlah kemungkinan string *orig*1 dan *orig*2 dari string *ad*1 dan *ad*2 yang memiliki jarak *dist*(*orig*1*, ad*1)+ *dist*(*orig*2*, ad*2) = *X* yang ber- ikutnya disebut dengan jawaban akhir.

**2.3.1 Membagi Permasalah Menjadi Submasalah yang *Inde- pendent***

Pada permasalahan klasik SPOJ 9967 *Playing With Words*, jawab- an akhir merupakan banyak kombinasi string *orig*1 dan string *orig*2 yang mungkin. Dapat dilihat bahwa perhitungan jumlah kombina- si string *orig*1 dari string *ad*1 dan perhitungan jumlah kombinasi string *orig*2 dari string *ad*2 tidak memiliki keterkaitan satu sama lain. Artinya adalah dapat dihitung jumlah kombinasi string *orig*1 dari string *ad*1 tanpa mempedulikan kondisi string *orig*2. Begitu



Gambar 2.1 Ilustrasi umum penyelesaian permasalahan dengan metode

*meet in the middle* tanpa operasi *replace*

juga sebaliknya pada perhitungan jumlah kombinasi string *orig*2. Dengan kata lain perhitungan jumlah kombinasi string *orig*1 dan *orig*2 dapat dilakukan secara terpisah. Teknik tersebut disebut de- ngan *meet in the middle*.

Apabila operasi *replace* pada permasalahan klasik SPOJ 9967 *Pla- ying With Words* diabaikan, maka jawaban akhir dari dapat dihitung dengan alur secara umum pada Gambar [2.1](#_bookmark28). Karena operasi *repla- ce* diabaikan, maka jawaban akhir dapat dibentuk dengan mencari seluruh kemungkinan pasangan string *orig*1 yang memiliki jarak *D* terhadap string *ad*1 dengan string *orig*2 yang memiliki jarak *X D* terhadap string *ad*2.

−

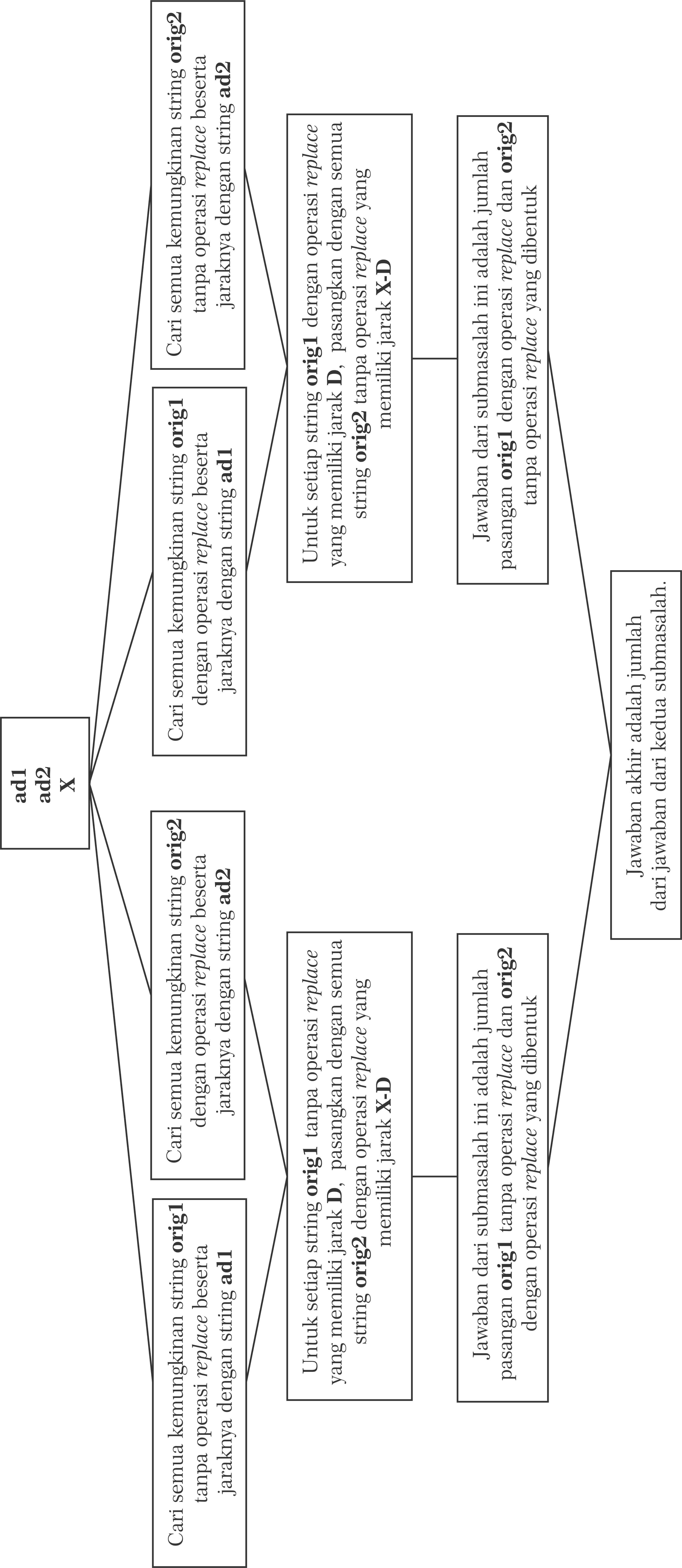
Ketika terdapat operasi *replace*, di mana operasi *replace* adalah ope- rasi di mana salah satu karakter pada string *orig*1 atau *orig*2 diganti dengan karakter sebelumnya atau sesudahnya secara alfabetis, ma- ka perlu dilakukan penyesuaian pada proses perhitungan jawaban. Selain perhitungan kombinasi string *orig*1 dan string *orig*2, per- lu juga dilakukan perhitungan untuk kombinasi string *orig*1 dan string *orig*2 setelah dilakukan satu kali operasi *replace*. Sehingga jawaban akhir dari permasalahan klasik SPOJ 9967 *Playing With Words* adalah total dari jumlah kemungkinan pasangan string *orig*1 tanpa operasi *replace* yang memiliki jarak *D* terhadap string *ad*1 dengan string *orig*2 dengan operasi *replace* yang memiliki jarak *X D* terhadap string *ad*2 dan jumlah kemungkinan pasangan string *orig*1 dengan operasi *replace* yang memiliki jarak *D* terhadap string *ad*1 dengan string *orig*2 tanpa operasi *replace* yang memili- ki jarak *X D* terhadap string *ad*2. Gambar [2.2](#_bookmark29) adalah ilustrasi umum penyelesaian permasalahan klasik SPOJ 9967 *Playing With Words* dengan metode *meet in the middle*.

−

−

Sebagai contoh kasus ketika string *orig*1 = *bd*, string *orig*2 = *gj* dan *X* = 5. Langkah pertama untuk menyelesaikan kasus ini adalah dengan mencari semua kemungkinan kombinasi string *orig*1 tanpa operasi *replace* dan semua kemungkinan string *orig*2 dengan opera- si *replace*. Tabel [2.3](#_bookmark30) menunjukkan seluruh kombinasi string *orig*1 tanpa operasi *replace* beserta jarak masing-masing dengan string *ad*1 dan Tabel [2.4](#_bookmark31) menunjukkan seluruh kombinasi string *orig*2 dengan operasi *replace* beserta jarak masing-masing dengan string *ad*2. Berikutnya adalah mencari seluruh pasangan kombinasi string *orig*1 tanpa operasi *replace* dengan jarak terhadap string *ad*1 sebe- sar *D* dan kombinasi string *orig*2 dengan operasi *replace* dengan jarak terhadap string *ad*2 sebesar *X D* yang hasilnya dapat dili- hat pada Tabel [2.5](#_bookmark32). Berikutnya hal yang sama juga dilakukan un- tuk kombinasi string *orig*1 dengan operasi *replace* dan string *orig*2 tanpa operasi *replace*. Tabel [2.6](#_bookmark33) menunjukkan seluruh kombinasi string *orig*1 dengan operasi *replace* beserta jarak masing-masing

−



Gambar 2.2 Ilustrasi umum penyelesaian permasalahan dengan metode

*meet in the middle* dengan operasi *replace*

|  |  |
| --- | --- |
| *orig*1 | Jarak dengan *ad*1 |
| *bd* | 0 |
| *db* | 4 |

Tabel 2.3 Kombinasi string *orig*1 dengan nilai string *ad*1 = *bd* tanpa operasi *replace*

|  |  |
| --- | --- |
| *orig*2 | Jarak dengan *ad*2 |
| *f j* | 1 |
| *hj* | 1 |
| *gi* | 1 |
| *gk* | 1 |
| *ig* | 5 |
| *kg* | 7 |
| *jf* | 7 |
| *jh* | 5 |

Tabel 2.4 Kombinasi string *orig*2 dengan nilai string *ad*2 = *gj* dengan operasi *replace*

dengan string *ad*1 dan Tabel [2.7](#_bookmark34) menunjukkan seluruh kombina- si string *orig*2 tanpa operasi *replace* beserta jarak masing-masing dengan string *ad*2. Berikutnya adalah mencari seluruh pasangan kombinasi string *orig*1 dengan operasi *replace* dengan jarak terha- dap string *ad*1 sebesar *D* dan kombinasi string *orig*2 dengan operasi *replace* tanpa jarak terhadap string *ad*2 sebesar *X D* yang hasil- nya dapat dilihat pada Tabel [2.8](#_bookmark35). Jawaban akhir dari kasus ketika string *orig*1 = *bd*, string *orig*2 = *gj* dan *X* = 5 adalah jumlah dari kombinasi pasangan string *orig*1 dan string *orig*2 pada Tabel

−

[2.5](#_bookmark32) dan Tabel [2.8](#_bookmark35).

Pada deskripsi permasalahan klasik SPOJ 9967 *Playing With Words* jawaban akhir adalah banyak kemungkinan string *orig*1 dan *orig*2 tanpa perlu menyertakan daftar string *orig*1 dan *orig*2 yang mung-

|  |  |  |  |
| --- | --- | --- | --- |
| *orig*1 | Jarak dengan *ad*1 | *orig*2 | Jarak dengan *ad*2 |
| *bd* | 0 | *ig* | 5 |
| *bd* | 0 | *jh* | 5 |
| *db* | 4 | *f j* | 1 |
| *db* | 4 | *hj* | 1 |
| *db* | 4 | *gi* | 1 |
| *db* | 4 | *gk* | 1 |

Tabel 2.5 Kombinasi string *orig*1 dengan nilai string *ad*1 = *bd* tanpa operasi *replace* dan string *orig*2 dengan nilai string *ad*2 = *gj* dengan operasi *replace* dengan *X* = 5

|  |  |
| --- | --- |
| *orig*1 | Jarak dengan *ad*1 |
| *ad* | 1 |
| *cd* | 1 |
| *bc* | 1 |
| *be* | 1 |
| *cb* | 3 |
| *eb* | 5 |
| *da* | 5 |
| *dc* | 3 |

Tabel 2.6 Kombinasi string *orig*1 dengan nilai string *ad*1 = *bd* dengan operasi *replace*

|  |  |
| --- | --- |
| *orig*2 | Jarak dengan *ad*2 |
| *gj* | 0 |
| *jg* | 6 |

Tabel 2.7 Kombinasi string *orig*2 dengan nilai string *ad*2 = *gj* tanpa operasi *replace*

|  |  |  |  |
| --- | --- | --- | --- |
| *orig*1 | Jarak dengan *ad*1 | *orig*2 | Jarak dengan *ad*2 |
| *eb* | 5 | *gj* | 0 |
| *da* | 5 | *gj* | 0 |

Tabel 2.8 Kombinasi string *orig*1 dengan nilai string *ad*1 = *bd* dengan operasi *replace* dan string *orig*2 dengan nilai string *ad*2 = *gj* tanpa ope- rasi *replace* dengan *X* = 5

kin. Maka dari itu, proses perhitungan jawaban akhir dapat diseder- hanakan agar algoritma yang dibangun lebih optimal. Seperti yang terlihat pada ilustrasi umum penyelesaian pada Gambar [2.3](#_bookmark37), proses pencarian kombinasi string *orig*1 dan *orig*2 dapat diganti dengan hanya menghitung jumlah kemungkinan kombinasi string *orig*1 de- ngan atau tanpa operasi *replace* dengan jarak terhadap string *ad*1 se- besar *D* dan jumlah kemungkinan kombinasi string *orig*2 dengan atau tanpa operasi *replace* dengan jarak terhadap string *ad*2 sebesar *D* dengan 0 ≤ *D* ≤ *X*.

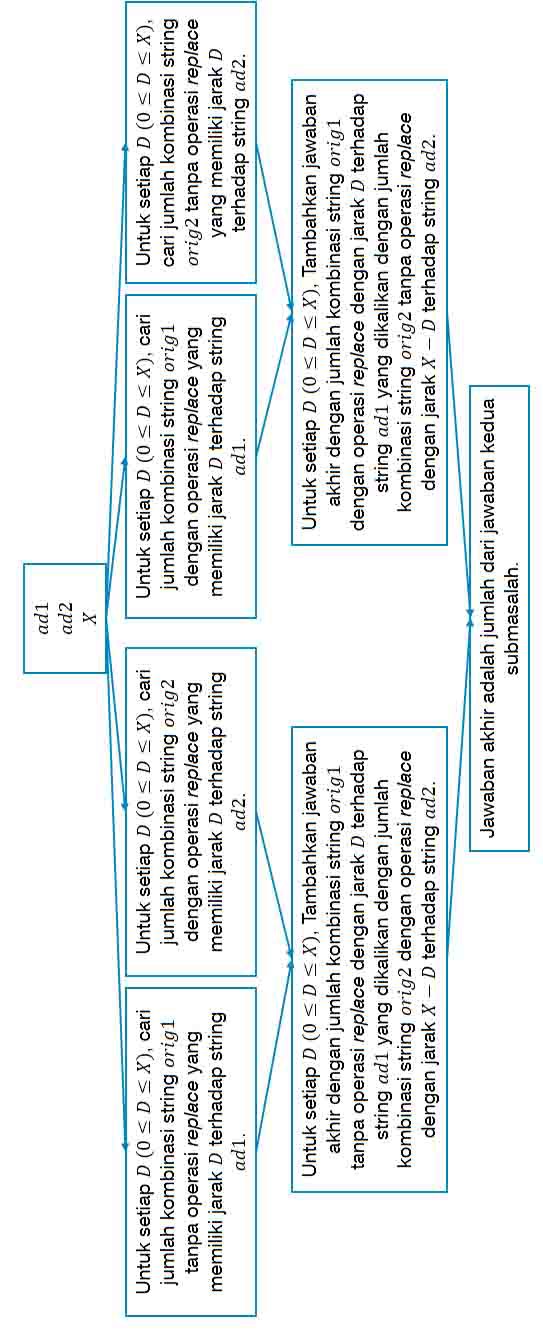
# 2.3.2 Submasalah Optimal untuk Menghitung Jumlah Kom- binasi String *Orig* dari String *Ad* Tanpa Operasi *Repla- ce* dengan Jarak *D*

Ilustrasi pada Gambar [2.3](#_bookmark37) 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 *orig*1 dan string *orig*2 tanpa operasi *replace* dengan jarak *D*.

Perhitungan jumlah kombinasi string *orig* tanpa operasi *replace* da- pat dilakukan dengan memanfaatkan teknik *bitmasking*. Gambar

[2.4](#_bookmark39) adalah ilustrasi perhitungan kombinasi string *orig* dari string

*ad* dengan nilai string *ad* = *bcd* dan *D* = 4 tanpa operasi *repla-*



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

*ce*. 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.

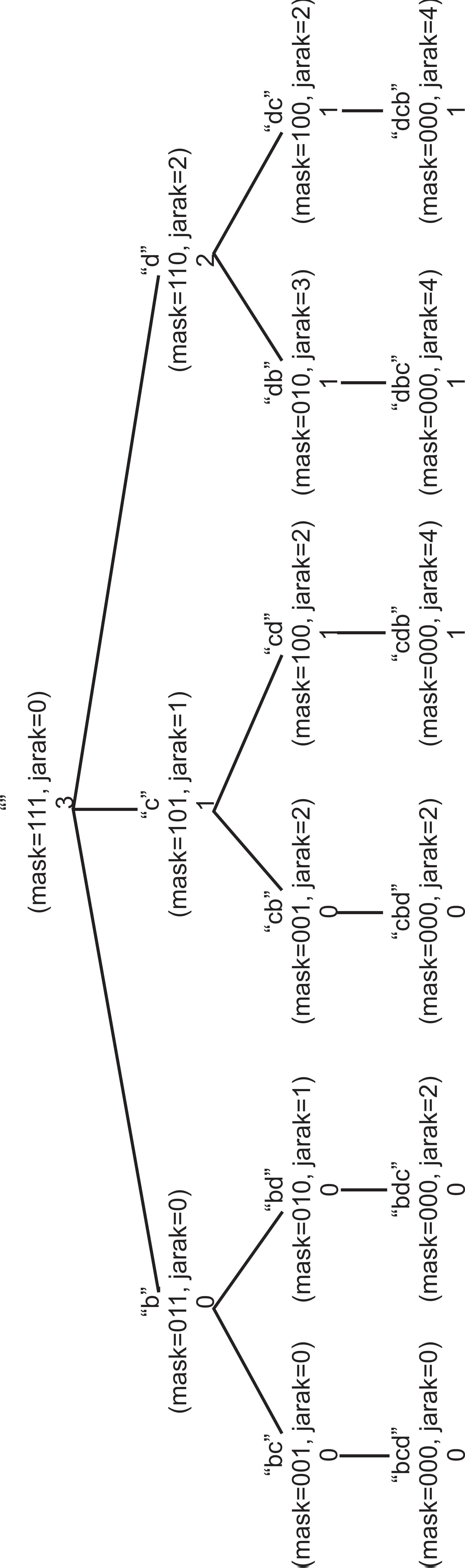
Namun, seperti yang dapat dilihat pada Gambar [2.5](#_bookmark40), terdapat bebe- rapa kasus di mana pada suatu *state*, *state* dengan nilai tersebut ber- sifat tumpang tindih dengan suatu *state* lain. Pada Gambar [2.5](#_bookmark40), *state* yang saling tumpang tindih ditandai dengan tulisan berwarna merah. Dengan adanya kasus *state* yang tumpang tindih tersebut dapat di- manfaatkan untuk melakukan optimasi pada algoritma yang diran- cang dengan tidak melakukan perhitungan ulang pada *state* yang su- dah pernah muncul sebelumnya. Sehingga proses perhitungan yang sebelumnya seperti dengan ilustrasi pada Gambar [2.4](#_bookmark39) dapat diseder- hanakan menjadi seperti yang terdapat pada Gambar [2.6](#_bookmark41). Teknik tersebut dikenal dengan teknik *dynamic programming*.

# 2.3.3 Submasalah Optimal untuk Menghitung Jumlah Kom- binasi String *Orig* dari String *Ad* dengan Operasi *Re- place* dengan Jarak *D*

Ilustrasi pada Gambar [2.3](#_bookmark37) 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 *orig*1 dan string *orig*2 tanpa operasi *replace* dengan jarak *D*.

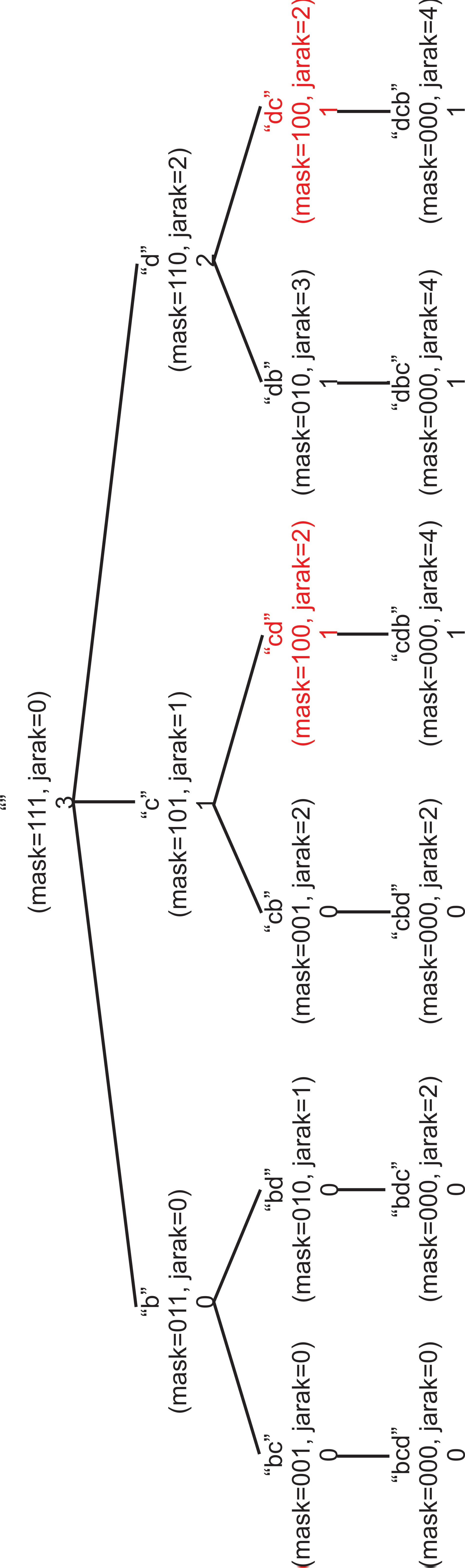
Cara untuk mendapatkan jawaban dari submasalah ini hampir sama dengan cara mencari jawaban pada submasalah perhitungan jumlah kombinasi string *orig* dari string *ad* tanpa operasi *replace*. Gambar

[2.7](#_bookmark44) merupakan ilustrasi dari penyelesaian submasalah perhitungan jumlah kombinasi string *orig* dari string *ad* dengan operasi *replace* dengan nilai *ad* = *be* dan *D* = 1. Sedikit berbeda dengan penye-

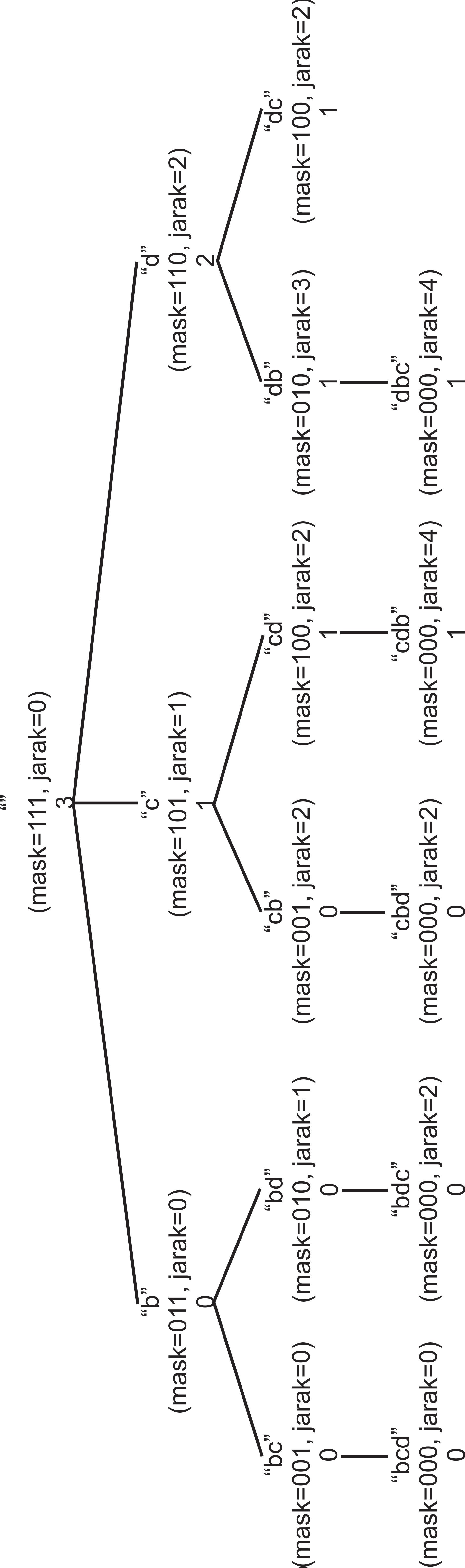


Gambar 2.4 Ilustrasi perhitungan jumlah kombinasi string *orig* dari string

*ad* tanpa operasi *replace* dengan nilai string *ad* = *bcd* dan *D* = 4



Gambar 2.5 Contoh kasus tumpang tindih pada perhitungan kombina- si 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

lesaian 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](#_bookmark45) sehingga dapat dila- kukan optimasi menggunakan teknik *dynamic programming* untuk meningkatkan efisiensi algoritma yang dibangun. Pada Gambar [2.9](#_bookmark46) dapat dilihat bahwa terdapat beberapa *state* yang ternyata memi- liki kondisi yang mampu diselesaikan dengan metode yang sama dengan metode penyelesaian submasalah perhitungan jumlah kom- binasi 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 kom- binasi string *orig* dari string *ad* tanpa operasi *replace*.

# Pemodelan Relasi Rekurens

*answer* = ∑*dist*=*min*(*X,*250) ((*F* ∗

*dist*=0

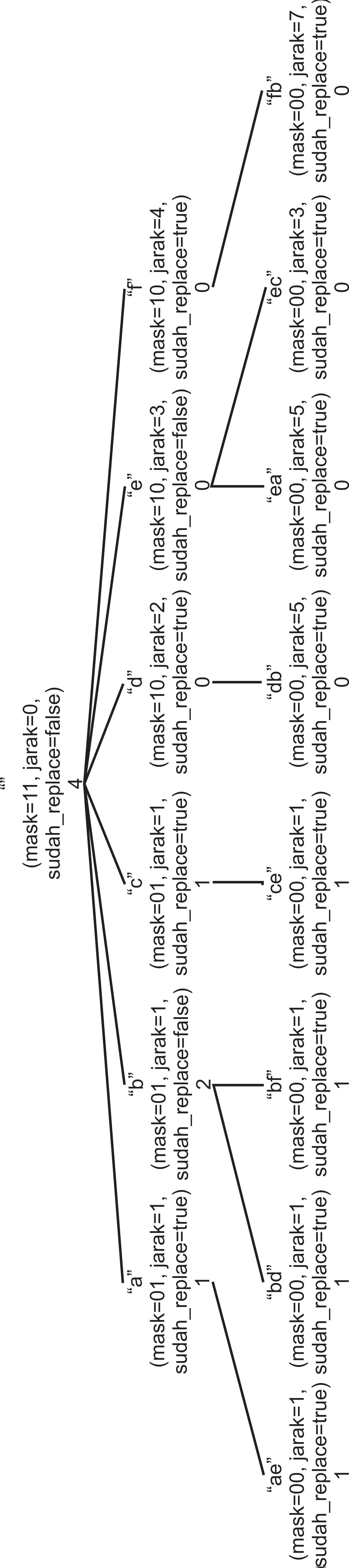
(*S*0*,*2*|S*0*|,bound−dist*)

*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*)))

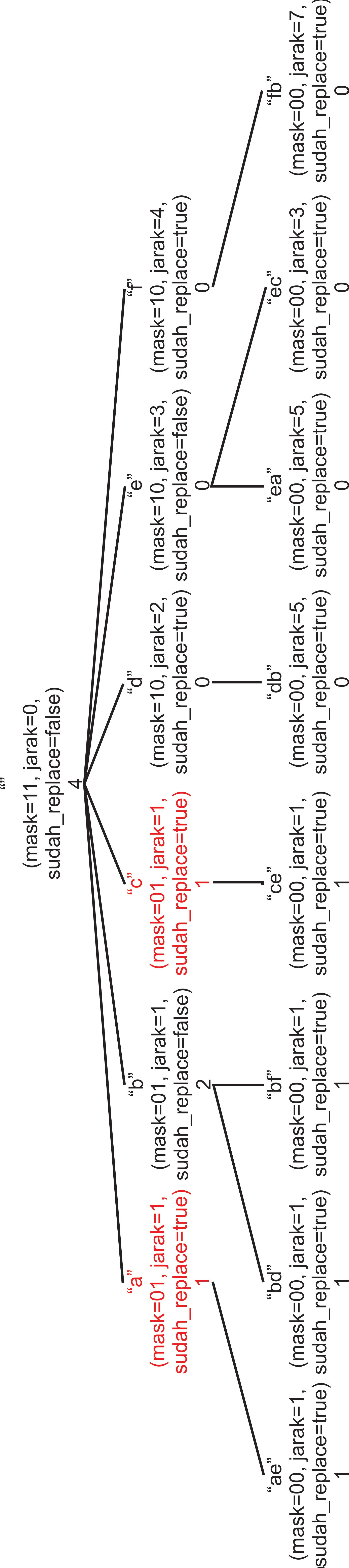
(2.4.1)

Pada subbab ini akan dijelaskan tentang relasi rekurens berdasark- an analisis pada subbab [2.3](#_bookmark26). Pada subbagian [2.3.1](#_bookmark27), dijelaskan bah- wa permasalahan dapat dipecah menjadi dua submasalah yang da- pat diselesaikan tanpa bergantung satu sama lain dengan memecah permasalahan berdasarkan masing-masing string *ad*. Karena terda-

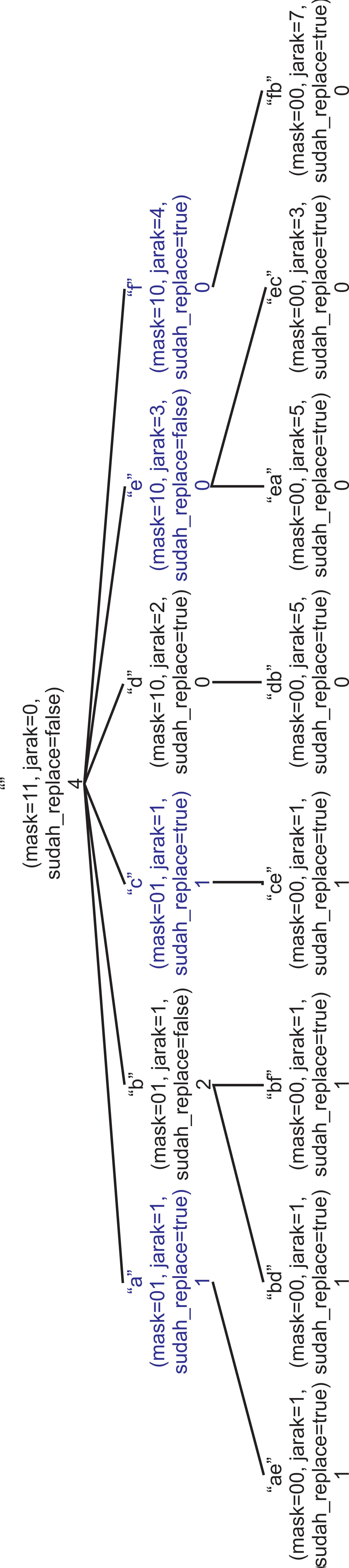


Gambar 2.7 Ilustrasi perhitungan jumlah kombinasi string *orig* dari string

*ad* dengan operasi *replace* dengan nilai string *ad* = *be* dan *D* = 1



Gambar 2.8 Contoh kasus tumpang tindih pada perhitungan kombina- si 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* terha- dap string *ad* tanpa operasi *replace* pada submasalah perhitungan jumlah kombinasi string *orig* terhadap string *ad* dengan operasi *replace*

pat sebuah operasi *replace* yang dilakukan, maka untuk menyele- saikan masing-masing submasalah harus dilakukan dua jenis perhi- tungan, yaitu operasi perhitungan jumlah kemungkinan string *orig* tanpa operasi *replace* dan operasi perhitungan jumlah kemungkinan string *orig* dengan operasi *replace*. Kedua operasi tersebut didefi- nisikan dalam bentuk fungsi sebagai berikut:

* + 1. *F*(*S,mask,dist*), yaitu fungsi untuk menghitung jumlah ke- mungkinan 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 ke- mungkinan string awal dari string *S* dengan sekali operasi *re- place* 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* da- pat dihitung dengan memanfaatkan kedua fungsi di atas. Persamaan

[2.4.1](#_bookmark43) merupakan persamaan untuk menghitung jawaban utama da- ri permasalahan klasik SPOJ 9967 *Playing With Words* di mana *S*0 adalah string *ad*1, *S*1 adalah string *ad*2, *X* adalah jumlah jarak(*ad*1, *orig*1) dengan jarak(*ad*2, *orig*2) dan *bound* = *min*(250*, X*) de- ngan Tabel [2.9](#_bookmark47) adalah daftar notasi yang digunakan pada persama- an 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 mung- kin melebihi angka 250. Sehingga apabila nilai masukan *X >* 250 dapat diasumsikan bahwa hal tersebut tidak mungkin.

|  |  |
| --- | --- |
| Notasi | Deskripsi |
| *dist* | Nilai jarak yang diiterasi dari 0  hingga *min*(*X,* 250) |
| *X* | Masukan yang merepresentasik-  an jumlah jarak string *orig*1 de- ngan string *ad*1 dan jarak string *orig*2 dengan string *ad*2 |
| *S*0 | String masukan yang merepre-  sentasikan *ad*1 |
| *S*1 | String masukan yang merepre-  sentasikan *ad*2 |
| |*S*0| | Panjang string *ad*1 |
| |*S*1| | Panjang string *ad*2 |
| *bound* | Nilai batas jarak maksimal yang  bernilai *min*(*X,* 250) |

Tabel 2.9 Daftar notasi persamaan [2.4.1](#_bookmark43)

# Pemodelan Relasi Rekurens Submasalah Optimal untuk Menghitung Jumlah Kemungkinan String Awal Tanpa Operasi *Replace* dengan Jarak *X*− *dist*

0*,* if *dist > bound,*



*F*(*S,mask,dist*) =

or (*mask* = 0 and

 *dist*

1*,* if *mask* = 0 and*,*

*F*(*S,mask,dist*) =

*bound*)

∑*i*=*N SB*(*mask*)

*dist* = *bound*



*i*=0

*F* 1(*S,mask,set*\_*bit*(*mask*)*i,dist*)*,* otherwise

(2.4.2)

|  |  |
| --- | --- |
| Notasi | Deskripsi |
| *S* | String yang akan dicari kemung-  kinan string awalnya. |
| *mask* | Sebuah bilangan bulat yang ber-  tugas sebagai *bitmask* yang me- representasikan kondisi karakter mana saja yang sudah diambil pada kondisi (*state*) tersebut. |
| *dist* | Jarak string yang sudah ter-  bentuk 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 digunak- an. |
| *N SB*(*mask*) | Mengembalikan jumlah angka 1  pada *mask* apabila direpresenta- sikan dalam basis biner |
| *set*\_*bit*(*mask*) | Himpunan index bilangan berni-  lai satu dari *mask* apabila direp- resentasikan dalam basis biner. |
| *is*\_*on*(*mask,idx*) | Mengembalikan nilai *true* apa-  bila bilangan pada index *idx* pa- da *mask* bernilai 1 apabila di- representasikan dalam basis bi- ner. |

Tabel 2.10 Daftar notasi persamaan [2.4.2](#_bookmark49), [2.4.3](#_bookmark51) dan [2.4.4](#_bookmark52)

*F* 1(*S,mask,idx,dist*) =

*F*(*S,mask* 2*idx,dist*+

*|Sidx−ScurIdx|*)*,* idx = |S| - 1 or

 *−*





*duplicate*\_*rule*1(*S,mask,idx*) =



*T rue*

0*,* otherwise

(2.4.3)

*T rue,* if *idx <* |*S*| − 1 and

*duplicate*\_*rule*1(*S,mask,idx*) =







(*Sidx* ̸= *Sidx*+1

or ((*Sidx* =

*Sidx*+1) and (*is*\_*on*(*mask,idx*+1) =

))



{

*F alse*

*F alse,* otherwise

(2.4.4)

*is*\_*on*(*mask,idx*) =

*T rue,* (*mask* & 2*idx*) = 1

*F alse,* otherwise

Pada bagian ini akan dijelaskan beberapa persamaan reku- rens dengan daftar notasi seperti yang terdapat pada Tabel

* 1. Pada persamaan [2.4.1](#_bookmark43) terdapat fungsi *F*(*S,mask,dist*) yang merupakan fungsi untuk menghitung jumlah kemungkin- an string *orig* dari string *S* tanpa operasi *replace* dengan ja- rak *X dist*. Nilai dari fungsi *F*(*S,mask,dist*) adalah ha- sil penjumlahan seluruh *state* yang berhubungan, yaitu *state*

−

*F*(*S,mask* 2*idx,dist*+ *Sidx ScurIdx* ) 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*+*|Sidx−ScurIdx|*) 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*). Per- samaan [2.4.3](#_bookmark51) adalah persamaan rekurens untuk menentukan apakah *state F*(*S,mask−*2*idx,dist*+*|Sidx−ScurIdx|*) merupakan *state* yang valid dari sebuah *state F*(*S,mask,dist*). Persamaan [2.4.2](#_bookmark49) adalah relasi reku- rens dari submasalah perhitungan jumlah kemungkinan string *orig*

dari string *S* tanpa operasi *replace* dengan jarak *X* − *dist*.

Fungsi *duplicate*\_*rule*1(*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* ak- an melakukan perhitungan seluruh kombinasi string *S* yang mung- kin 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*0*S*1*S*2
    2. *S*0*S*2*S*1
    3. *S*1*S*0*S*2
    4. *S*1*S*2*S*0
    5. *S*2*S*0*S*1
    6. *S*2*S*1*S*0

Sehingga apabila string *S* = *bcc*, maka kombinasi string yang ter- bentuk adalah sebagai berikut:

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

Terdapat beberapa string yang bersifat duplikat sehingga tidak da- pat dikatakan sebagai string yang berbeda. Sehingga banyak kom- binasi string berbeda dari *S* = *bcc* adalah 3 dengan rincian sebagai

berikut:

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

Konsep dasar dari fungsi *duplicate*\_*rule*1(*S,mask,idx*) adalah de- ngan menerapkan aturan hanya boleh memilih karakter *Sidx* apabila *Sidx* = *Sidx*+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 disi- mulasikan 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 ma- sukan yang akan dicari jumlah kombinasi string awalnya tan- pa operasi *replace* adalah *S* = *bcc*.
2. Parameter kedua yang bernilai 7 merepresentasikan bahwa pada *state* tersebut nilai *mask* = 7 atau apabila direpresenta- sikan 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 men- capai kondisi valid sebuah kombinasi string awal di mana *bound* = 5.

−

Karena himpunan *set*\_*bit*(7) = {0*,* 1*,* 2}, maka hasil dari pe- manggilan fungsi *F*(*bcc,*7*,*3) adalah hasil dari penjumlahan hasil fungsi-fungsi yang akan dipanggil pada fungsi tersebut. Beri- kut adalah fungsi-fungsi yang dipanggil pada pemanggilan fungsi *F*(*bcc,*7*,*3):

* 1. *F* 1(*bcc,*7*,*0*,*3) yang akan memanggil fungsi *F*(*bcc,*6*,*3) karena memenuhi syarat *duplicate*\_*rule*1(*bcc,*7*,*0) = *T rue*.
  2. *F* 1(*bcc,*7*,*1*,*3) yang akan mengembalikan nilai 0 karena nilai *idx* = *S* 1 dan nilai *duplicate*\_*rule*1(*bcc,*7*,*1) = *T rue* bukan merupakan kondisi yang memenuhi syarat terpanggil- nya fungsi *F*(*bcc,*5*,*4).

̸ | | − ̸

* 1. *F* 1(*bcc,*7*,*2*,*3) yang akan memanggil fungsi *F*(*bcc,*3*,*4) karena memenuhi syarat *idx* = |*S*| − 1 di mana |*S*| = 3 sehing- ga |*S*| − 1 = 2 dan *idx* = 2.

Proses dilanjutkan secara rekursif, yaitu dengan pemanggilan fung- si *F*(*bcc,*6*,*3). Karena himpunan *set*\_*bit*(6) = {1*,* 2}, maka hasil da- ri pemanggilan fungsi *F*(*bcc,*6*,*3) adalah hasil dari penjumlahan ha- sil fungsi-fungsi yang akan dipanggil pada fungsi tersebut. Beri- kut adalah fungsi-fungsi yang dipanggil pada pemanggilan fungsi *F*(*bcc,*6*,*3):

1. *F* 1(*bcc,*6*,*1*,*3) yang akan mengembalikan nilai 0 karena nilai *idx* = *S* 1 dan nilai *duplicate*\_*rule*1(*bcc,*6*,*0) = *T rue* bukan merupakan kondisi yang memenuhi syarat terpanggil- nya fungsi *F*(*bcc,*4*,*3).

̸ | | − ̸

1. *F* 1(*bcc,*6*,*2*,*3) yang akan memanggil fungsi *F*(*bcc,*2*,*3) karena memenuhi syarat *idx* = |*S*| − 1 di mana |*S*| = 3 sehing- ga |*S*| − 1 = 2 dan *idx* = 2.

Proses berikutnya adalah pemanggilan fungsi *F*(*bcc,*2*,*3). Kare- na himpunan *set*\_*bit*(2) = 1 , maka nilai dari fungsi *F*(*bcc,*2*,*3) sama dengan nilai dari satu-satunya fungsi yang dipanggil pa- da fungsi tersebut, yaitu *F* 1(*bcc,*2*,*1*,*3). Fungsi *F* 1(*bcc,*2*,*1*,*3) ak- an memanggil fungsi *F*(*bcc,*0*,*3) karena memenuhi kondisi nilai *duplicate*\_*rule*1(*bcc,*2*,*1) = *T rue*. Fungsi *F*(*bcc,*0*,*3) sendiri akan mengembalikan nilai 0 karena kondisi ketika *mask* = 0 adalah kondisi dasar (*base case*) dan *dist* ̸= *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 dipang- gil pada fungsi tersebut. Berikut adalah fungsi-fungsi yang dipang- gil pada pemanggilan fungsi *F*(*bcc,*3*,*4):

{ }

1. *F* 1(*bcc,*3*,*0*,*4) yang akan memanggil fungsi *F*(*bcc,*2*,*5) karena memenuhi syarat *duplicate*\_*rule*1(*bcc,*3*,*0) = *T rue*.
2. *F* 1(*bcc,*3*,*1*,*4) yang akan memanggil fungsi *F*(*bcc,*1*,*4) karena memenuhi syarat *duplicate*\_*rule*1(*bcc,*3*,*1) = *T rue*.

Proses berikutnya adalah pemanggilan fungsi *F*(*bcc,*2*,*5). Kare- na himpunan *set*\_*bit*(2) = 1 , maka nilai dari fungsi *F*(*bcc,*2*,*5) sama dengan nilai dari satu-satunya fungsi yang dipanggil pa- da fungsi tersebut, yaitu *F* 1(*bcc,*2*,*1*,*5). Fungsi *F* 1(*bcc,*2*,*1*,*5) ak- an memanggil fungsi *F*(*bcc,*0*,*5) karena memenuhi kondisi nilai *duplicate*\_*rule*1(*bcc,*2*,*1) = *T rue*. 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 *F* 1(*bcc,*1*,*0*,*4). Fungsi *F* 1(*bcc,*1*,*0*,*4) akan memanggil fungsi *F*(*bcc,*0*,*5) karena memenuhi kondisi nilai *duplicate*\_*rule*1(*bcc,*1*,*0) = *T rue*. Fungsi *F*(*bcc,*0*,*5) sendiri akan mengembalikan nilai 1 karena kondi- si ketika *mask* = 0 adalah kondisi dasar (*base case*) dan *dist* = *bound* di mana *dist* = 5 dan *bound* = 5. Sehingga nilai dari fung- si *F*(*bcc,*1*,*4) = 1, fungsi *F*(*bcc,*3*,*4) = 2 dan nilai akhir dari fungsi *F*(*bcc,*7*,*3) = 2.

{ }

|  |  |
| --- | --- |
| Notasi | Deskripsi |
| *S* | String yang akan dicari kemung-  kinan string awalnya. |
| *mask* | Sebuah bilangan bulat yang ber-  tugas sebagai *bitmask* yang me- representasikan kondisi karakter mana saja yang sudah diambil pada kondisi (*state*) tersebut. |
| *dist* | Jarak string yang sudah ter-  bentuk 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 digunak- an. |
| *N SB*(*mask*) | Mengembalikan jumlah angka 1  pada *mask* apabila direpresenta- sikan dalam basis biner |
| *set*\_*bit*(*mask*) | Himpunan index bilangan berni-  lai satu dari *mask* apabila direp- resentasikan dalam basis biner. |
| *is*\_*on*(*mask,idx*) | Mengembalikan nilai *true* apa-  bila bilangan pada index *idx* pa- da *mask* dalam basis biner ber- nilai 1. |

Tabel 2.11 Daftar notasi persamaan [2.4.5](#_bookmark56), [2.4.6](#_bookmark57), [2.4.7](#_bookmark58), [2.4.8](#_bookmark59), [2.4.9](#_bookmark60) dan

[2.4.10](#_bookmark61) (1)

|  |  |
| --- | --- |
| Notasi | Deskripsi |
| *charLastP os*(*S,C*) | Index terbesar dari karakter *C*  pada string *S*. |
| *charF irstP os*(*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*. |

Tabel 2.12 Daftar notasi persamaan [2.4.5](#_bookmark56), [2.4.6](#_bookmark57), [2.4.7](#_bookmark58), [2.4.8](#_bookmark59), [2.4.9](#_bookmark60) dan

[2.4.10](#_bookmark61) (2)

# Pemodelan Relasi Rekurens Submasalah Optimal untuk Menghitung Jumlah Kemungkinan String Awal dengan Sekali Operasi *Replace* dengan Jarak *X*− *dist*

0*, dist > bound* or

∑*i*=*N SB*(*mask*)

*mask* = *bound*





*i*=0

*G*(*S,mask,dist*) =

(*G*1(*S,mask,*

*set*\_*bit*(*mask*)*i,dist*)

+*G*2(*S,mask,*

*set*\_*bit*(*mask*)*i,dist*)



+*G*3(*S,mask, set*\_*bit*(*mask*)*i,dist*))*,* otherwise



(2.4.5)

*G*(*S,mask*





−2*idx,dist*



*G*1 = +*|Sidx*

(*S,mask,idx,dist*)



*−S*



*curIdx*

*|*)*, idx* = |*S*| − 1 or

*duplicate*\_*rule*1(*S,mask,idx*) =

*T rue*

0*,* otherwise

*F*(*S,mask*

*−*2*idx,dist*

+*|Sidx*+1

= *−ScurIdx|*)*, idx* = |*S*| − 1 or

(*duplicate*\_*rule*1(*S,mask,idx*) =

*G*2

(*S,mask,idx,dist*)

(2.4.6)





*T rue* and

*duplicate*\_*rule*2(*S,mask,idx*) =

*T rue*)

0*,* otherwise

(2.4.7)

*G*3(*S,mask,idx,dist*)

*F*(*S,mask*

*−ScurIdx|*)*,* (*idx* = |*S*| − 1 or

+*|Sidx−*1

*−*2*idx,dist*

= 

*duplicate*\_*rule*1(*S,mask,idx*) =





*T rue*) and (*idx* = 0 or *duplicate*\_*rule*3(*S,mask,idx*) = *T rue*)

0*,* otherwise

(2.4.8)

*T rue,* if *idx <* |*S*| − 1 and

*duplicate*\_*rule*2(*S,mask,idx*) =



(*charF irstP os*(*S,Sidx*+1) =

−1 or

(*charF irstP os*(*S,Sidx*+1) ̸=



−1 and *is*\_*on*(*mask*,



*charF irstP os*(*S,Sidx*+1)) =

*F alse F alse,* otherwise

(2.4.9)

Pada bagian ini akan dijelaskan beberapa persamaan rekurens de- ngan daftar notasi seperti yang terdapat pada Tabel [2.11](#_bookmark54) dan Ta- bel [2.12](#_bookmark55). Pada persamaan [2.4.1](#_bookmark43) 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 fung- si *G*(*S,mask,dist*) adalah hasil penjumlahan dari seluruh *state* yang berhubungan dan valid. Terdapat tiga kasus *state* yang mungkin,

yaitu:

*T rue,* if *idx >* 0 − 1 and

*duplicate*\_*rule*3(*S, mask, idx*) = 



(*charLastP os*(*S, Sidx*−

or





1) = −1

(*charLastP os*(*S, Sidx*− 1) ̸= −1 and *is*\_*on*(*mask*, *charLastP os*(*S,Sidx−*1)) =

*T rue*

*F alse,* otherwise

(2.4.10)

* + - 1. *State G*(*S,mask−*2*idx,dist*+*|Sidx−ScurIdx|*) dengan kasus ketika memilih *Sidx* sebagai *origcurIdx* tanpa melakukan *replace*.
      2. *State F*(*S,mask−*2*idx,dist*+*|Sidx*+1*−ScurIdx|*) dengan kasus keti- ka memilih *Sidx*+1, dengan kata lain mengganti karakter *Sidx* dengan karakter setelahnya dalam alfabet sebagai *origcurIdx*.
      3. *State F*(*S,mask−*2*idx,dist*+*|Sidx−*1*−ScurIdx|*) dengan kasus ke- tika memilih *Sidx* − 1, dengan kata lain mengganti karak- ter *Sidx* dengan karakter sebelumnya dalam alfabet, sebagai *origcurIdx*.

Masing-masing jenis *state* yang berhubungan langsung dengan *state G*(*S,mask,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,mask,dist*):

1. Persamaan [2.4.6](#_bookmark57) adalah persamaan yang menentukan apa- kah *state G*(*S,mask−*2*idx,dist*+*|Sidx−ScurIdx|*) merupakan sebu- ah *state* yang valid dari *state G*(*S,mask,dist*).
2. Persamaan [2.4.7](#_bookmark58) adalah persamaan yang menentukan apakah *state F*(*S,mask−*2*idx,dist*+*|Sidx*+1*−ScurIdx|*) merupakan sebuah *state* yang valid dari *state G*(*S,mask,dist*).
3. Persamaan [2.4.8](#_bookmark59) adalah persamaan yang menentukan apakah *state F*(*S,mask−*2*idx,dist*+*|Sidx−*1*−ScurIdx|*) merupakan sebuah *state* yang valid dari *state G*(*S,mask,dist*).

Fungsi *duplicate*\_*rule*2(*S,mask,idx*) dan fungsi *duplicate*\_*rule*3(*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 meru- pakan kombinasi dari string *S* yang memiliki panjang 3 karakter dengan sekali operasi *replace*:

1. *S*0 + 1*S*1*S*2

2. *S*0*S*1 + 1*S*2

3. *S*0*S*1*S*2 + 1

4. *S*0 − 1*S*1*S*2

5. *S*0*S*1 − 1*S*2

6. *S*0*S*1*S*2 − 1

7. *S*0 + 1*S*2*S*1

8. *S*0*S*2 + 1*S*1

9. *S*0*S*2*S*1 + 1

10. *S*0 − 1*S*2*S*1

11. *S*0*S*2 − 1*S*1

12. *S*0*S*2*S*1 − 1

13. *S*1 + 1*S*0*S*2

14. *S*1*S*0 + 1*S*2

15. *S*1*S*0*S*2 + 1

16. *S*1 − 1*S*0*S*2

17. *S*1*S*0 − 1*S*2

18. *S*1*S*0*S*2 − 1

19. *S*1 + 1*S*2*S*0

20. *S*1*S*2 + 1*S*0

21. *S*1*S*2*S*0 + 1

22. *S*1 − 1*S*2*S*0

23. *S*1*S*2 − 1*S*0

24. *S*1*S*2*S*0 − 1

25. *S*2 + 1*S*0*S*1

26. *S*2*S*0 + 1*S*1

27. *S*2*S*0*S*1 + 1

28. *S*2 − 1*S*0*S*1

29. *S*2*S*0 − 1*S*1

30. *S*2*S*0*S*1 − 1

31. *S*2 + 1*S*1*S*0

32. *S*2*S*1 + 1*S*0

33. *S*2*S*1*S*0 + 1

34. *S*2 − 1*S*1*S*0

35. *S*2*S*1 − 1*S*0

36. *S*2*S*1*S*0 − 1

Sehingga apabila string *S* = *bcc*, maka kombinasi string yang ter- bentuk 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*\_*rule*2(*S,mask,idx*) mirip dengan konsep dasar dari fungsi *duplicate*\_*rule*1(*S,mask,idx*). Hanya sa- ja karena pada fungsi *G*(*S,mask,dist*) terdapat kondisi di mana ka- rakter *Si* yang dipilih diganti dengan karakter *Si* + 1 yang meru- pakan karakter berikutnya dalam alfabet, aturan yang diterapkan berbeda dengan fungsi *duplicate*\_*rule*1(*S,mask,idx*). Pada fungsi *duplicate*\_*rule*1(*S,mask,idx*) diterapkan aturan hanya boleh memi- lih sebuah karakter *Sidx* apabila tidak ada karakter *Sidx* + 1 yang muncul pada string *S* atau karakter *Sidx* + 1 pertama yang muncul atau dengan kata lain karakter *Sidx* + 1 dengan index terkecil pada string *S* telah dipilih sebelumnya.

Fungsi *duplicate*\_*rule*3(*S,mask,idx*) memiliki konsep dasar yang mirip dengan fungsi *duplicate*\_*rule*2(*S,mask,idx*). Hanya saja pada fungsi *duplicate*\_*rule*3(*S,mask,idx*) bertujuan untuk mencegah du- plikasi pada kondisi fungsi *G*(*S,mask,dist*) memilih sebuah karakter *Sidx* yang berikutnya digantikan dengan karakter *Sidx* 1 yang me- rupakan karakter sebelumnya dalam alfabet. Sehingga aturan yang diterapkan adalah hanya boleh memilih karakter *Sidx* apabila karak- ter *Sidx* −1 yang merupakan karakter sebelumnya pada alfabet tidak muncul pada string *S* atau karakter *Sidx* − 1 yang terakhir muncul atau dengan kata lain karakter *Sidx* 1 dengan index terbesar pada string *S* belum dipilih sebelumnya.

−

−

Untuk penjelasan fungsi *G*(*S,mask,dist*) yang lebih jelas, akan disi- mulasikan 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 ma- sukan yang akan dicari jumlah kombinasi string awalnya tan- pa operasi *replace* adalah *S* = *bcc*.
2. Parameter kedua yang bernilai 7 merepresentasikan bahwa pada *state* tersebut nilai *mask* = 7 atau apabila direpresenta- sikan 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 men- capai kondisi valid sebuah kombinasi string awal di mana *bound* = 3.

−

Karena himpunan *set*\_*bit*(7) = {0*,* 1*,* 2}, maka hasil dari pe- manggilan fungsi *G*(*bcc,*7*,*2) adalah hasil dari penjumlahan hasil fungsi-fungsi yang akan dipanggil pada fungsi tersebut. Beri- kut adalah fungsi-fungsi yang dipanggil pada pemanggilan fungsi *G*(*bcc,*7*,*2):

1. Fungsi *G*1(*bcc,*7*,*0*,*2) yang akan memanggil fungsi *G*(*bcc,*6*,*2)

karena memenuhi kondisi *duplicate*\_*rule*1(*bcc,*7*,*0) = *T rue*.

1. Fungsi *G*2(*bcc,*7*,*0*,*2) yang mana karena *idx* ̸= |*S*| − 1 dan *duplicate*\_*rule*2(*bcc,*7*,*0) = *F alse* tidak akan memanggil fungsi *F*(*bcc,*6*,*3) dan akan mengembalikan nilai 0.
2. Fungsi *G*3(*bcc,*7*,*0*,*2) yang akan memanggil fungsi *F*(*bcc,*6*,*3) karena memenuhi kondisi *duplicate*\_*rule*1(*bcc,*7*,*0) = *T rue* dan *duplicate*\_*rule*3(*bcc,*7*,*0) = *T rue*. Fungsi *F*(*bcc,*6*,*3) akan mengembalikan nilai 1.
3. Fungsi *G*1(*bcc,*7*,*1*,*2) yang mana karena *idx* ̸= |*S*| − 1 dan *duplicate*\_*rule*1(*bcc,*7*,*1) = *F alse* tidak akan memanggil fungsi *G*(*bcc,*5*,*3) dan akan mengembalikan nilai 0.
4. Fungsi *G*2(*bcc,*7*,*1*,*2) yang mana karena *idx* |*S*| − 1 dan

*duplicate*\_*rule*1(*bcc,*7*,*1) = *F alse* tidak akan memanggil fungsi *F*(*bcc,*5*,*4) dan akan mengembalikan nilai 0.

1. Fungsi *G*3(*bcc,*7*,*1*,*2) yang mana karena *idx* ̸= |*S*| − 1 dan *duplicate*\_*rule*1(*bcc,*7*,*1) = *F alse* tidak akan memanggil fungsi *F*(*bcc,*5*,*2) dan akan mengembalikan nilai 0.
2. Fungsi *G*1(*bcc,*7*,*2*,*2) yang akan memanggil fungsi *G*(*bcc,*3*,*3)

karena memenuhi kondisi *idx* = |*S*| − 1.

1. Fungsi *G*2(*bcc,*7*,*2*,*2) yang akan memanggil fungsi *F*(*bcc,*3*,*4) ka- rena memenuhi kondisi *idx* = *S* 1. Fungsi *F*(*bcc,*3*,*4) akan mengembalikan nilai 0.

| | −

1. Fungsi *G*3(*bcc,*7*,*2*,*2) yang akan memanggil fungsi *F*(*bcc,*3*,*2) ka- rena 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 dipang- gil pada fungsi tersebut. Berikut adalah fungsi-fungsi yang dipang- gil pada pemanggilan fungsi *G*(*bcc,*6*,*2):

{ }

1. Fungsi *G*1(*bcc,*6*,*1*,*2) yang mana karena *idx* ̸= |*S*| − 1 dan *duplicate*\_*rule*1(*bcc,*6*,*1) = *F alse* tidak akan memanggil fungsi *G*(*bcc,*4*,*2) dan akan mengembalikan nilai 0.
2. Fungsi *G*2(*bcc,*6*,*1*,*2) yang mana karena *idx* |*S*| − 1 dan

*duplicate*\_*rule*1(*bcc,*6*,*1) = *F alse* tidak akan memanggil fungsi *F*(*bcc,*4*,*3) dan akan mengembalikan nilai 0.

1. Fungsi *G*3(*bcc,*6*,*1*,*2) yang mana karena *idx* ̸= |*S*| − 1 dan *duplicate*\_*rule*1(*bcc,*6*,*1) = *F alse* tidak akan memanggil fungsi *F*(*bcc,*4*,*3) dan akan mengembalikan nilai 0.
2. Fungsi *G*1(*bcc,*6*,*2*,*2) yang akan memanggil fungsi *G*(*bcc,*2*,*2)

karena memenuhi kondisi *idx* = |*S*| − 1.

1. Fungsi *G*2(*bcc,*6*,*2*,*2) yang akan memanggil fungsi *F*(*bcc,*2*,*3) ka- rena memenuhi kondisi *idx* = |*S*| − 1. Fungsi *F*(*bcc,*2*,*3) akan

mengembalikan nilai 1.

1. Fungsi *G*3(*bcc,*6*,*2*,*2) yang mana karena *idx* ̸= 0 dan *duplicate*\_*rule*3(*bcc,*6*,*1) = *F alse* tidak akan memanggil fungsi *F*(*bcc,*2*,*3) dan akan mengembalikan nilai 0.

Berikutnya adalah pemanggilan fungsi *G*(*bcc,*2*,*2). Karena himpun- an *set*\_*bit*(2) = 1 , maka hasil dari pemanggilan fungsi *G*(*bcc,*2*,*2) adalah hasil dari penjumlahan hasil fungsi-fungsi yang akan dipang- gil pada fungsi tersebut. Berikut adalah fungsi-fungsi yang dipang- gil pada pemanggilan fungsi *G*(*bcc,*2*,*2):

{ }

1. Fungsi *G*1(*bcc,*2*,*1*,*2) yang akan memanggil fungsi *G*(*bcc,*0*,*2) karena memenuhi kondisi *duplicate*\_*rule*1(*bcc,*2*,*1) = *T rue*. Fungsi *G*(*bcc,*0*,*2) akan mengembalikan nilai 0 karena pada ka- sus dasar *mask* = 0, fungsi *G*(*S,mask,dist*) akan selalu meng- embalikan nilai 0.
2. Fungsi *G*2(*bcc,*2*,*1*,*2) yang akan memanggil fungsi *F*(*bcc,*0*,*3) karena memenuhi kondisi *duplicate*\_*rule*1(*bcc,*2*,*1) = *T rue* dan *duplicate*\_*rule*2(*bcc,*2*,*1) = *T rue*. Fungsi *F*(*bcc,*0*,*3) akan mengembalikan nilai 1.
3. Fungsi *G*3(*bcc,*2*,*1*,*2) yang mana karena *idx* |*S*| − 1 dan

*duplicate*\_*rule*3(*bcc,*2*,*1) = *F alse* 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 dipang- gil pada fungsi tersebut. Berikut adalah fungsi-fungsi yang dipang- gil pada pemanggilan fungsi *G*(*bcc,*3*,*3):

{ }

1. Fungsi *G*1(*bcc,*3*,*0*,*3) yang akan memanggil fungsi *G*(*bcc,*2*,*4) karena memenuhi kondisi *duplicate*\_*rule*1(*bcc,*3*,*0) = *T rue*. Nilai dari fungsi *G*(*bcc,*2*,*4) adalah 0 karena kasus dasar dari fungsi *G*(*S,mask,dist*) ketika *dist > bound* akan mengemba-

likan nilai 0.

1. Fungsi *G*2(*bcc,*3*,*0*,*3) yang mana karena *idx* ̸= |*S*| − 1 dan *duplicate*\_*rule*2(*bcc,*3*,*0) = *F alse* tidak akan memanggil fungsi *F*(*bcc,*2*,*3) dan akan mengembalikan nilai 0.
2. Fungsi *G*3(*bcc,*3*,*0*,*3) yang akan memanggil fungsi *F*(*bcc,*2*,*5) karena memenuhi kondisi *duplicate*\_*rule*1(*bcc,*3*,*0) = *T rue* dan *idx* = 0. Nilai dari *F*(*bcc,*2*,*5) adalah 0.
3. Fungsi *G*1(*bcc,*3*,*1*,*3) yang akan memanggil fungsi *G*(*bcc,*1*,*3)

karena memenuhi kondisi *duplicate*\_*rule*1(*bcc,*3*,*0) = *T rue*.

1. Fungsi *G*2(*bcc,*3*,*1*,*3) yang akan memanggil fungsi *F*(*bcc,*1*,*4) karena memenuhi kondisi *duplicate*\_*rule*1(*bcc,*3*,*1) = *T rue* dan *duplicate*\_*rule*2(*bcc,*3*,*1) = *T rue*. Nilai dari *F*(*bcc,*1*,*4) adalah 0.
2. Fungsi *G*3(*bcc,*3*,*1*,*3) yang akan memanggil fungsi *F*(*bcc,*1*,*4) karena memenuhi kondisi *duplicate*\_*rule*1(*bcc,*3*,*1) = *T rue* dan *duplicate*\_*rule*3(*bcc,*3*,*1) = *T rue*. Nilai dari *F*(*bcc,*1*,*4) adalah 0.

Berikutnya adalah pemanggilan fungsi *G*(*bcc,*1*,*3). Karena himpun- an *set*\_*bit*(1) = 0 , maka hasil dari pemanggilan fungsi *G*(*bcc,*1*,*3) adalah hasil dari penjumlahan hasil fungsi-fungsi yang akan dipang- gil pada fungsi tersebut. Berikut adalah fungsi-fungsi yang dipang- gil pada pemanggilan fungsi *G*(*bcc,*1*,*3):

{ }

1. Fungsi *G*1(*bcc,*1*,*0*,*3) yang akan memanggil fungsi *G*(*bcc,*0*,*4) karena memenuhi kondisi *duplicate*\_*rule*1(*bcc,*1*,*0) = *T rue*. 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 *G*2(*bcc,*1*,*0*,*3) yang akan memanggil fungsi *F*(*bcc,*0*,*3) karena memenuhi kondisi *duplicate*\_*rule*1(*bcc,*1*,*0) = *T rue* dan *duplicate*\_*rule*2(*bcc,*1*,*0) = *T rue*. Nilai dari fungsi *F*(*bcc,*0*,*3) adalah 1.
3. Fungsi *G*3(*bcc,*1*,*0*,*3) yang akan memanggil fungsi *F*(*bcc,*0*,*5)

karena memenuhi kondisi *duplicate*\_*rule*1(*bcc,*1*,*0) = *T rue*

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 menyele- saikan permasalahan klasik SPOJ 9967 *Playing With Words*.

# Desain Umum Sistem

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

Program akan diawali dengan melakukan *preprocess* lalu dilanjutk- an dengan menerima masukan berupa banyak data uji. Untuk seti- ap data uji berupa sebuah baris yang terdiri dari tiga data masukan yang dipisahkan oleh sebuah spasi, yaitu string *ad*1, string *ad*2 dan bilangan bulat *X*. String *ad*1 dan *ad*2 adalah string hasil enkripsi sesuai dengan deskripsi permasalahan klasik SPOJ 9967 *Playing With Words* dan *X* = *dist*(*ad*1*, orig*1) + *dist*(*ad*2*, orig*2) di mana *dist*(*st*1*, st*2) adalah total jarak absolut masing-masing karakter *st*1 dan *st*2 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](#_bookmark65).

# 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 bi- langan yang telah ditentukan. Konstanta rentang bilangan yang di- gunakan adalah 0 hingga 210 di mana 10 merupakan panjang mak-

49

Main()

1

2

3

4

5

6

7

preprocess()

*TC* = Input()

**for** *T* = 0 **to** *T C* − 1

readInput() init() solveProblem() writeOutput()

Gambar 3.1 Pseudocode Fungsi Main

preprocess()

2

3

4

1 **for** *num* = 0 **to** 210 − 1

**for** *bitP os* = 0 **to** 10 − 1

**if** *isBitOn*(*num, bitP os*) *setBit*(*powerN um*).push(*bitP os*)

Gambar 3.2 Pseudocode Fungsi Preprocess

simal string *ad*1 dan *ad*2 yang mungkin. Tabel [A.1](#_bookmark145) sampai dengan Tabel [A.35](#_bookmark179) adalah nilai dari himpunan *setBit* setelah fungsi prepro- cess dijalankan. Gambar [3.2](#_bookmark66) adalah *pseudocode* untuk fungsi *prep-* *rocess*.

# Desain Fungsi Init

Fungsi init merupakan fungsi yang bertujuan untuk melakukan ini- sialisasi nilai awal dan perhitungan data-data yang diperlukan untuk menyelesaikan permasalahan klasik SPOJ 9967 *Playing With Wo- rds* untuk setiap kasus uji.

Karena algoritma yang dibangun menggunakan pendekatan para- digma *dynamic programming* yang menggunakan teknik memoi- sasi, maka algoritma yang dibangun harus melakukan inisialisa- si nilai untuk setiap memo yang digunakan. Terdapat dua vari- abel memo yang digunakan pada algoritma yang dibangun, yai- tu *memoF*(*idx,mask,dist*) untuk mencatat hasil perhitungan fungsi *F*(*S,mask,dist*) dan *memoG*(*idx,mask,dist*) untuk mencatat hasil per- hitungan fungsi *G*(*S,mask,d*).

Untuk mempermudah menyelesaikan permasalahan klasik SPOJ 9967 *Playing With Words*, algoritma yang dibangun membutuhk- an string masukan *ad*1 dan *ad*2 dalam keadaan yang sudah terurut *ascending* secara alfabetis.

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

* + 1. *maxM ask*(*S*) yaitu nilai maksimal *mask* untuk string *S*. Ni- lai maksimal *mask* dari string *S* adalah 2*|S|* − 1.
    2. *charF irstP os*(*S,C*) yaitu posisi pertama karakter *C* pada string *S*.
    3. *charLastP os*(*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](#_bookmark68) adalah nilai dari *charF irstP os*(*S,C*) dan *charLastP os*(*S,C*). Nilai dari *maxM ask*(*S*) adalah 511. Gambar

* 1. adalah *pseudocode* dari fungsi init.

|  |  |  |
| --- | --- | --- |
| *C* | *charF irstP os*(*S,C*) | *charLastP os*(*S,C*) |
| *a* | ∅ | ∅ |
| *b* | ∅ | ∅ |
| *c* | 0 | 0 |
| *d* | ∅ | ∅ |
| *e* | ∅ | ∅ |
| *f* | ∅ | ∅ |
| *g* | ∅ | ∅ |
| *h* | 1 | 1 |
| *i* | 2 | 3 |
| *j* | ∅ | ∅ |
| *k* | ∅ | ∅ |
| *l* | ∅ | ∅ |
| *m* | ∅ | ∅ |
| *n* | 4 | 5 |
| *o* | 6 | 7 |
| *p* | ∅ | ∅ |
| *q* | ∅ | ∅ |
| *r* | ∅ | ∅ |
| *s* | ∅ | ∅ |
| *t* | 8 | 8 |
| *u* | ∅ | ∅ |
| *v* | ∅ | ∅ |
| *w* | ∅ | ∅ |
| *x* | ∅ | ∅ |
| *y* | ∅ | ∅ |
| *z* | ∅ | ∅ |

Tabel 3.1 Hasil *charF irstP os*(*S,C*) dan *charLastP os*(*S,C*) dengan string *S* = ”*inicontoh*” setelah fungsi init dijalankan

init() 1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

*memoF* = ∅

*memoG* = ∅

*charF irstP os* = ∅

*charLastP os* = ∅

sort(*ad*1)

*maxM ask*(*ad*1) = 2*|ad*1*|* − 1

**for** *i* = 0 **to** |*ad*1| − 1

*charLastP os*(*ad*1*,ad*1*i*) = *i*

**if** *charF irstP os*(*ad*1*,ad*1*i*) = ∅

*charF irstP os*(*ad*1*,ad*1*i*) = *i*

sort(*ad*2)

*maxM ask*(*ad*2) = 2*|ad*2*|* − 1

**for** *i* = 0 **to** |*ad*2| − 1

*charLastP os*(*ad*2*,ad*2*i*) = *i*

**if** *charF irstP os*(*ad*2*,ad*2*i*) = ∅

*charF irstP os*(*ad*2*,ad*2*i*) = *i*

Gambar 3.3 Pseudocode Fungsi Init

# Desain Fungsi Solve

Fungsi solve adalah fungsi yang bertujuan untuk menyelesaikan permasalahan sesuai dengan deskripsi permasalahan untuk setiap input yang diberikan. Setelah melalui proses *preprocessing*, mem- baca masukan dan inisialisasi, masukan akan diolah untuk mengha- silkan 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](#_bookmark42).

Fungsi solve merupakan fungsi yang mengimplementasi persamaan

[2.4.1](#_bookmark43). Fungsi solve sendiri akan membutuhkan beberapa fungsi-

solve(*ad*1*, ad*2*, X*)

1

2

3

4

5

6

7

8

9

*ret* = 0

*bound* = *min*(*X,* 250)

**for** *dist* = 0 **to** *min*(250*, X*)

*rem* = *X* − *dist*

10

**if** *rem >* 250

continue

**if** *rem <* 0

break

*ret* = *ret* + *F*(*ad*1*,maxM askad*1*,bound−dist*)

+*G*(*ad*2*,maxM askad*2*,bound−rem*)

*ret* = *ret* + *G*(*ad*1*,maxM askad*1*,bound−dist*)

11

**return** *ret*

+*F*(*ad*2*,maxM askad*2*,bound−rem*)

Gambar 3.4 Pseudocode Fungsi Solve

fungsi lain untuk membantu. Fungsi-fungsi tersebut antara lain fungsi *F*(*S,mask,dist*) dan fungsi *G*(*S,mask,dist*). Gambar [3.4](#_bookmark71) adalah *pseudocode* dari fungsi solve.

# 3.4.1 Desain Fungsi F

Pada *pseudocode* pada Gambar [3.4](#_bookmark71), terdapat perhitungan dengan menggunakan fungsi *F*(*S,mask,dist*) pada baris 9 dan 10. Seperti yang telah dijelaskan pada bagian [2.4.1](#_bookmark48), fungsi *F*(*S,mask,dist*) ada- lah fungsi untuk menghitung jumlah kemungkinan string *orig* da- ri 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*+ *Sidx ScurIdx* ) di mana

*− | − |*

*curIdx* adalah jumlah bit tidak menyala pada *mask* dan *idx* adalah

*set*\_*bit*(*mask*)*i* untuk setiap i di mana 0 ≤ *i* ≤ *N SBmask* dengan

|  |  |  |
| --- | --- | --- |
| 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) | *base case* | 0 |
| *F*(*behkn,*23*,*14) | *base case* | 0 |
| *F*(*behkn,*27*,*11) | *base case* | 0 |
| *F*(*behkn,*29*,*8) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*22*,*11) | *base case* | 0 |
| *F*(*behkn,*26*,*8) | *base case* | 0 |
| *F*(*behkn,*28*,*5) | *F*(*behkn,*24*,*5) + *F*(*behkn,*20*,*8) +  *F*(*behkn,*12*,*11) | 1 |
| *F*(*behkn,*12*,*11) | *base case* | 0 |
| *F*(*behkn,*20*,*8) | *base case* | 0 |
| *F*(*behkn,*24*,*5) | *F*(*behkn,*16*,*5) + *F*(*behkn,*8*,*8) | 1 |
| *F*(*behkn,*8*,*8) | *base case* | 0 |
| *F*(*behkn,*16*,*5) | *F*(*behkn,*0*,*5) | 1 |
| *F*(*behkn,*0*,*5) | *base case* | 1 |

Tabel 3.2 Simulasi fungsi *F* dengan *S* = ”*kbenh*”, *X* = 5 dan *dist* = 5

*set*\_*bit*(*mask*) adalah Himpunan index bit menyala pada *mask* dan *N SBmask* adalah jumlah bit menyala pada *mask*. Algoritma pada fungsi *F*(*S,mask,dist*) akan didasari oleh persamaan [2.4.2](#_bookmark49). Gambar

[3.5](#_bookmark74) adalah *pseudocode* dari fungsi *F*(*S,mask,dist*).

F(*S, mask, dist*)

1

2

3

4

5

6

7

8

9

10

11

12

13

**if** *dist > bound* ∨ (*mask* = 0 ∧ *dist* ̸= *bound*)

**return** 0

**if** *mask* = 0 ∧ *dist* = *bound*

**return** 1

**if** *memoF*(*S,mask,dist*) ̸= ∅

**return** *memoF*(*S,mask,dist*)

*numberOf SetBit* = size(*setBit*(*mask*))

*retV al* = 0

**for** *i* = 0 **to** *numberOf SetBit* − 1

break

*ret* = *ret* + *F* 1(*S,mask,setBit*(*mask*)*i,dist*)

**return** *memoF*(*S,mask,dist*) = *ret*

**if** *setBit*(*mask*)*i* ≥ *length*(*S*)

Gambar 3.5 Pseudocode Fungsi F

Karena tidak semua *state* yang terhubung dengan *sta- te F*(*S,mask,dist*) valid, maka diperlukan sebuah fungsi *F* 1(*S,mask,idx,dist*) untuk menentukan valid atau tidaknya sebuah *state F*(*S,mask−*2*idx,dist*+*|Sidx−ScurIdx|*) yang terbentuk dari *state F*(*S,mask,dist*). Perancangan algoritma fungsi *F* 1(*S,mask,idx,dist*)

akan didasari oleh persamaan [2.4.3](#_bookmark51). Gambar [3.6](#_bookmark75) adalah *pseu- docode* dari fungsi *F* 1(*S,mask,idx,dist*) dan Gambar [3.7](#_bookmark76) adalah *pseudocode* dari fungsi *duplicate*\_*rule*1(*S, mask, idx*). Tabel

[3.2](#_bookmark73) adalah simulasi fungsi *F* dengan *S* = ”*kbenh*”, *X* = 5 dan

*dist* = 5.

3

2 **if** *idx* = length(*S*) − 1 ∨ duplicateRule1(*S,mask,idx*)

F1(*S, mask, idx, dist*)

1 *curIdx* = length(*S*) − size(*setBit*(*mask*))

**return** *F*(*S,mask−*2*idx,dist*+*|Sidx−ScurIdx|*)

Gambar 3.6 Pseudocode Fungsi F1

duplicate\_rule1(*S, mask, idx*)

1 **return** *idx < length*(*S*) − 1 ∧ (*Sidx* ̸=*idx*+1 ∨

(*Sidx* = *Sidx*+1 ∧ ¬*isBitOn*(*mask,idx*+1)))

Gambar 3.7 Pseudocode Fungsi duplicate\_rule1

# Desain Fungsi G

Pada *pseudocode* pada Gambar [3.4](#_bookmark71), terdapat perhitungan dengan menggunakan fungsi *G*(*S,mask,dist*) pada baris 9 dan 10. Seper- ti yang telah dijelaskan pada bagian [2.4.1](#_bookmark48), 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 penjum- lahan dari seluruh *state* yang berhubungan dengan dengan *state G*(*S,mask,dist*) yang valid. Gambar [3.8](#_bookmark84) adalah *pseudocode* dari fung- si *G*(*S,mask,dist*). Terdapat tiga kasus *state* yang mungkin, yai- tu:

−

* + - 1. *State G*(*S,mask* 2*idx,dist*+ *Sidx ScurIdx* ) dengan kasus ketika mengambil karakter posisi *idx* pada string *S* sebagai karak- ter posisi *curIdx* pada string *orig* tanpa melakukan *replace*.

*− | − |*

Fungsi *G*1(*S,mask,idx,dist*) adalah fungsi yang melakukan va-

lidasi terhadap *state* jenis pertama. Gambar [3.9](#_bookmark85) adalah *pseu- docode* dari fungsi *G*1(*S,mask,idx,dist*).

* + - 1. *State F*(*S,mask* 2*idx,dist*+ *Sidx*+1 *ScurIdx* ) dengan kasus ke- tika mengambil karakter posisi *idx* pada string *S* sebagai

*− | − |*

karakter posisi *curIdx* pada string *orig* dengan melakukan *replace* dengan karakter setelahnya secara alfabetis. Fung- si *G*2(*S,mask,idx,dist*) adalah fungsi yang melakukan validasi terhadap *state* jenis kedua. Gambar [3.10](#_bookmark86) adalah *pseudocode* dari fungsi *G*2(*S,mask,idx,dist*) dan Gambar [3.12](#_bookmark88) adalah *pseu- docode* dari fungsi *duplicate*\_*rule*2(*S, mask, idx*).

* + - 1. *State F*(*S,mask* 2*idx,dist*+ *Sidx* 1 *ScurIdx* ) dengan kasus keti- ka mengambil karakter posisi *idx* pada string *S* sebagai ka- rakter posisi *curIdx* pada string *orig* dengan melakukan *re-*

*− | − − |*

*place* dengan karakter sebelumnya secara alfabetis. Fung- si *G*3(*S,mask,idx,dist*) adalah fungsi yang melakukan validasi terhadap *state* jenis ketiga. Gambar [3.11](#_bookmark87) adalah *pseudocode* dari fungsi *G*3(*S,mask,idx,dist*) dan Gambar [3.13](#_bookmark89) adalah *pseu- docode* dari fungsi *duplicate*\_*rule*3(*S, mask, idx*).

Tabel [3.3](#_bookmark78) sampai dengan [3.8](#_bookmark83) adalah simulasi fungsi *G* dengan *S* =

”*kbenh*”, *X* = 5 dan *dist* = 0.

|  |  |  |
| --- | --- | --- |
| 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 3.3 Simulasi fungsi *G* dengan *S* = ”*kbenh*”, *X* = 5 dan *dist* = 0

(1)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*13*,*13) | *base case* | 0 |
| *F*(*behkn,*21*,*10) | *base case* | 0 |
| *F*(*behkn,*25*,*7) | *base case* | 0 |
| *F*(*behkn,*28*,*7) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*13*,*13) | *base case* | 0 |
| *G*(*behkn,*13*,*12) | *base case* | 0 |
| *F*(*behkn,*21*,*8) | *base case* | 0 |
| *F*(*behkn,*21*,*10) | *base case* | 0 |
| *G*(*behkn,*21*,*9) | *base case* | 0 |
| *F*(*behkn,*25*,*5) | *F*(*behkn,*24*,*11) + *F*(*behkn,*17*,*8) +  *F*(*behkn,*9*,*11) | 0 |
| *F*(*behkn,*9*,*11) | *base case* | 0 |
| *F*(*behkn,*17*,*8) | *base case* | 0 |
| *F*(*behkn,*24*,*11) | *base case* | 0 |
| *F*(*behkn,*25*,*7) | *base case* | 0 |
| *G*(*behkn,*25*,*6) | *base case* | 0 |
| *F*(*behkn,*28*,*7) | *base case* | 0 |
| *F*(*behkn,*28*,*5) | *F*(*behkn,*24*,*5) + *F*(*behkn,*20*,*8) +  *F*(*behkn,*12*,*11) | 1 |
| *F*(*behkn,*12*,*11) | *base case* | 0 |
| *F*(*behkn,*20*,*8) | *base case* | 0 |

Tabel 3.4 Simulasi fungsi *G* dengan *S* = ”*kbenh*”, *X* = 5 dan *dist* = 0

(2)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*24*,*5) | *memoF*(*behkn,*24*,*5) | 1 |
| *G*(*behkn,*28*,*6) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*22*,*7) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*14*,*10) | *base case* | 0 |
| *G*(*behkn,*14*,*9) | *base case* | 0 |
| *F*(*behkn,*22*,*5) | *F*(*behkn,*20*,*8) + *F*(*behkn,*18*,*5) +  *F*(*behkn,*6*,*11) | 0 |
| *F*(*behkn,*6*,*11) | *base case* | 0 |
| *F*(*behkn,*18*,*5) | *memoF*(*behkn,*18*,*5) | 0 |
| *F*(*behkn,*20*,*8) | *base case* | 0 |
| *F*(*behkn,*22*,*7) | *base case* | 0 |
| *G*(*behkn,*22*,*6) | *base case* | 0 |
| *F*(*behkn,*26*,*2) | *F*(*behkn,*24*,*5) + *F*(*behkn,*18*,*5) +  *F*(*behkn,*10*,*8) | 1 |
| *F*(*behkn,*10*,*8) | *base case* | 0 |
| *F*(*behkn,*18*,*5) | *memoF*(*behkn,*18*,*5) | 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 |
| *F*(*behkn,*26*,*4) | *F*(*behkn,*24*,*7) + *F*(*behkn,*18*,*7) +  *F*(*behkn,*10*,*10) | 0 |
| *F*(*behkn,*10*,*10) | *base case* | 0 |
| *F*(*behkn,*18*,*7) | *base case* | 0 |
| *F*(*behkn,*24*,*7) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*10*,*10) | *base case* | 0 |
| *G*(*behkn,*10*,*9) | *base case* | 0 |
| *F*(*behkn,*18*,*5) | *F*(*behkn,*16*,*11) + *F*(*behkn,*2*,*8) | 0 |
| *F*(*behkn,*2*,*8) | *base case* | 0 |
| *F*(*behkn,*16*,*11) | *base case* | 0 |
| *F*(*behkn,*18*,*7) | *base case* | 0 |
| *G*(*behkn,*18*,*6) | *base case* | 0 |
| *F*(*behkn,*24*,*7) | *base case* | 0 |
| *F*(*behkn,*24*,*5) | *F*(*behkn,*16*,*5) + *F*(*behkn,*8*,*8) | 1 |
| *F*(*behkn,*8*,*8) | *base case* | 0 |
| *F*(*behkn,*16*,*5) | *memoF*(*behkn,*16*,*5) | 1 |
| *G*(*behkn,*24*,*6) | *base case* | 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 3.6 Simulasi fungsi *G* dengan *S* = ”*kbenh*”, *X* = 5 dan *dist* = 0

(4)

|  |  |  |
| --- | --- | --- |
| 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 3.7 Simulasi fungsi *G* dengan *S* = ”*kbenh*”, *X* = 5 dan *dist* = 0

(5)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fungsi | Perhitungan Nilai | | | Nilai |
| *F*(*behkn,*0*,*5) | *base case* | | | 1 |
| *G*(*behkn,*16*,*6) | *base case* | | | 0 |
| *F*(*behkn,*24*,*1) | *memoF*(*behkn,*24*,*1) | | | 0 |
| *F*(*behkn,*24*,*1) | *F*(*behkn,*16*,*1) + *F*(*behkn,*8*,*4) | | | 0 |
| *F*(*behkn,*8*,*4) | *memoF*(*behkn,*8*,*4) | | | 0 |
| *F*(*behkn,*16*,*1) | *memoF*(*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) | *base case* | | | 1 |
| *F*(*behkn,*8*,*4) | *F*(*behkn,*0*,*7) | | | 0 |
| *F*(*behkn,*0*,*7) | *base case* | | | 0 |
| *G*(*behkn,*8*,*3) | *G*(*behkn,*0*,*6)  *F*(*behkn,*0*,*7) | + *F*(*behkn,*0*,*5) | + | 1 |
| *F*(*behkn,*0*,*7) | *base case* | | | 0 |
| *F*(*behkn,*0*,*5) | *base case* | | | 1 |
| *G*(*behkn,*0*,*6) | *base case* | | | 0 |
| *F*(*behkn,*16*,*1) | *memoF*(*behkn,*16*,*1) | | | 0 |
| *F*(*behkn,*16*,*1) | *F*(*behkn,*0*,*1) | | | 0 |
| *F*(*behkn,*0*,*1) | *base case* | | | 0 |
| *G*(*behkn,*16*,*0) | *G*(*behkn,*0*,*0)  *F*(*behkn,*0*,*1) | + *F*(*behkn,*0*,*1) | + | 0 |
| *F*(*behkn,*0*,*1) | *base case* | | | 0 |
| *F*(*behkn,*0*,*1) | *base case* | | | 0 |
| *G*(*behkn,*0*,*0) | *base case* | | | 0 |

Tabel 3.8 Simulasi fungsi *G* dengan *S* = ”*kbenh*”, *X* = 5 dan *dist* = 0

(6)

G(*S, mask, dist*)

1

2

3

4

5

6

7

8

9

10

11

12

13

**if** *dist > bound* ∨ *mask* = 0

**return** 0

**if** *memoG*(*S,mask,dist*)

**return** *memoG*(*S,mask,dist*)

≠ ∅

*numberOf SetBit* = size(*setBit*(*mask*))

*retV al* = 0

**for** *i* = 0 **to** *numberOf SetBit* − 1

break

*ret* = *ret* + *G*1(*S,mask,setBit*(*mask*)*i,dist*) *ret* = *ret* + *G*2(*S,mask,setBit*(*mask*)*i,dist*) *ret* = *ret* + *G*3(*S,mask,setBit*(*mask*)*i,dist*)

**return** *memoF*(*S,mask,dist*) = *ret*

**if** *setBit*(*mask*)*i* ≥ *length*(*S*)

Gambar 3.8 Pseudocode Fungsi G

3

2 **if** *idx* = length(*S*) − 1 ∨ duplicateRule1(*S,mask,idx*)

G1(*S, mask, idx, dist*)

1 *curIdx* = length(*S*) − size(*setBit*(*mask*))

**return** *G*(*S,mask−*2*idx,dist*+*|Sidx−ScurIdx|*)

Gambar 3.9 Pseudocode Fungsi G1

2 **if** *idx* = length(*S*) − 1 ∨ (duplicateRule2(*S,mask,idx*)∧

G2(*S, mask, idx, dist*)

1 *curIdx* = length(*S*) − size(*setBit*(*mask*))

3

duplicateRule1(*S,mask,idx*))

**return** *F*(*S,mask−*2*idx,dist*+*|*(*Sidx*+1)*−ScurIdx|*)

Gambar 3.10 Pseudocode Fungsi G2

G3(*S, mask, idx, dist*)

1. *curIdx* = length(*S*) − size(*setBit*(*mask*))
2. **if** (*idx* = length(*S*) − 1 ∨ duplicateRule1(*S,mask,idx*))∧

(*idx* = 0 ∨ duplicateRule3(*S,mask,idx*))

3

**return** *F*(*S,mask−*2*idx,dist*+*|*(*Sidx−*1)*−ScurIdx|*)

Gambar 3.11 Pseudocode Fungsi G3

duplicate\_rule2(*S, mask, idx*)

1 **return** *idx < length*(*S*) − 1 ∧ (*charF irstP os*(*S,Sidx*+1)

= ∅ ∨ (*charF irstP os*(*S,Sidx*+1) ̸= ∅

∧¬*isBitOn*(*mask,charF irstP os*(*S,Sidx*+1))))

Gambar 3.12 Pseudocode Fungsi duplicate\_rule2

duplicate\_rule3(*S, mask, idx*)

1 **return** *idx >* 0 ∧ (*charLastP os*(*S,Sidx−*1)

= ∅ ∨ (*charLastP os*(*S,Sidx−*1) ̸= ∅

∧*isBitOn*(*mask,charLastP os*(*S,Sidx−*1))))

Gambar 3.13 Pseudocode Fungsi duplicate\_rule3

*Halaman ini sengaja dikosongkan*

# BAB IV IMPLEMENTASI

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

# Lingkungan Implementasi

Lingkungan implementasi dalam pembuatan Tugas Akhir ini meli- puti perangkat keras dan perangkat lunak yang digunakan untuk me- lakukan 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.

# Rancangan Data

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

69

# 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 ma- sukan, baris pertama berupa sebuah bilangan bulat yang merepre- sentasikan jumlah kasus uji yang ada pada berkas tersebut. Un- tuk setiap kasus uji, masukan berupa sebuah baris masukan yang terdiri dari dua buah string *ad*1 dan *ad*2, yang merepresentasikan string hasil transformasi dari string *orig*1 dan *orig*2 secara berturut- turut, diikuti oleh sebuah bilangan bulat *X* yang merepresentasikan *dist*(*ad*1*, orig*1) + *dist*(*ad*2*, orig*2).

# Data Keluaran

Data keluaran yang dihasilkan oleh program hanya berupa satu nilai, yaitu jumlah kemungkinan string *orig*1 dan *orig*2 yang mungkin membentuk string *ad*1 dan *ad*2.

# Implementasi Algoritma

Pada subbab ini akan dijelaskan tentang implementasi proses algo- ritma secara keseluruhan berdasarkan desain yang telah dijelaskan pada bab [III](#_bookmark62).

# Header-Header yang Diperlukan

Implementasi algoritma dengan teknik *meet in the middle* dan *dyna- mic programming* untuk menyelesaikan permasalahan klasik SPOJ 9967 *Playing With Words* membutuhkan lima buah header yaitu cs- tdio, algorithm, vector, string dan cstring, seperti yang terlihat pada

Kode Sumber [4.3.1](#_bookmark97).

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 membe- rikan keluaran. Header vector berisi struktur data yang digunak- an untuk menyimpan data himpunan index bit menyala dari sebu- ah bilangan bulat. Header algorithm berisi modul yang memiliki fungsi-fungsi yang sangat berguna dalam membantu mengimple- mentasi 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*.

# Variabel Global

Variabel global digunakan untuk memudahkan dalam mengakses data yang digunakan lintas fungsi. Kode sumber implementasi va- riabel global dapat dilihat pada Kode Sumber [4.3.2](#_bookmark99).

1

2

3

4

5

6

7

8

9

10

**using namespace** std;

vector<**int**> set\_bit[(1 << 11)]; string S[2];

**int** charLastPos[2][50]; **int** charFirstPos[2][50]; **int** X, bound;

**int** memoF[2][(1 << 10) + 2][250 + 2];

**int** memoG[2][(1 << 10) + 2][250 + 2];

**int** maxMask[2];

Kode Sumber 4.3.2 Variabel global

# Implementasi Fungsi Main

Fungsi main adalah implementasi algoritma yang dirancang pada Gambar [3.1](#_bookmark65). Implementasi fungsi main dapat dilihat pada Kode Sumber [4.3.3](#_bookmark101).

1. **int** main() {
2. preprocess();
3. **int** t;

4 scanf("%d", &t);

1. **for** (**int** tc=1; tc<=t; tc++) {
2. readInput();
3. init();
4. **long long** ans = solveProblem();
5. writeOutput(tc, ans);

10 }

11 **return** 0;

12 }

Kode Sumber 4.3.3 Fungsi main

# Implementasi Fungsi Preprocess

Fungsi preprocess adalah implementasi dari hasil perancangan pada

*pseudocode* pada Gambar [3.2](#_bookmark66). Implementasi dari fungsi preprocess

dapat dilihat pada Kode Sumber [4.3.4](#_bookmark103).

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))

*<→* set\_bit[i].

5

6

7

}

*<→* push\_back(j);

}

}

Kode Sumber 4.3.4 Fungsi preprocess

# Implementasi Fungsi ReadInput

Fungsi readInput akan membaca masukan dari berkas uji untuk se- tiap kasus ujinya. Pada awalnya, fungsi akan membaca masukan string *ad*1, lalu dilanjutkan dengan membaca string *ad*2 dan diakhi- ri dengan membaca sebuah bilangan bulat *X*. Implementasi fungsi readInput dapat dilihat pada Kode Sumber [4.3.5](#_bookmark105).

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

# Implementasi Fungsi Init

Implementasi dari fungsi init dapat dilihat pada Kode Sumber [4.3.6](#_bookmark107).

1. **void** init() {
2. memset(charLastPos, -1, **sizeof**

3

memset(charFirstPos, -1, **sizeof**

*<→* charLastPos);

4

5

6

7

memset(memoF, -1, **sizeof** memoF); memset(memoG, -1, **sizeof** memoG); **for** (**int** idx = 0; idx < 2; idx++) {

sort(S[idx].begin(), S[idx].end()

*<→* charFirstPos);

8

maxMask[idx] = (1 << S[idx].

*<→* );

9

**for** (**int** i = 0; i < S[idx].length

*<→* length()) - 1;

10

*<→* (); i++) {

11

charLastPos[idx][S[idx][i

*<→* ] - 'a'] = i;

**if** (charFirstPos[idx][S[

*<→* idx][i] - 'a'] ==

12

*<→* -1) {

charFirstPos[idx

*<→* ][S[idx][i]

*<→* - 'a'] = i

13

14

15

16

17

}

*<→* ;

}

}

bound = min(X, 250);

}

Kode Sumber 4.3.6 Fungsi init

# Implementasi Fungsi Solve

Fungsi solve adalah implementasi dari desain algoritma pada Gambar [3.4](#_bookmark71) dimana algoritma tersebut adalah hasil perancangan berdasarkan persamaan [2.4.1](#_bookmark43). Implementasi dari fungsi solve dapat dilihat pada Kode Sumber [4.3.7](#_bookmark109).

1. **long long** solveProblem() {
2. **long long** ret = 0;
3. **for** (**int** dist = 0; dist <= min(250, X);

4

5

6

7

*<→* dist++) {

**int** rem = X - dist;

**if** (rem > 250) **continue**; **if** (rem < 0) **break**;

ret += F(0, maxMask[0], bound -

*<→* dist) \* G(1, maxMask[1],

8

ret += G(0, maxMask[0], bound -

*<→* bound - rem);

*<→* dist) \* F(1, maxMask[1],

9

10

11

}

**return** ret;

*<→* bound - rem);

}

Kode Sumber 4.3.7 Fungsi solve

# Implementasi Fungsi F

terdapat fungsi F yang telah dirancang algoritmanya pada subbab

[3.4.1](#_bookmark72). Implementasi dari fungsi F dapat dilihat pada Kode Sumber

[4.3.8](#_bookmark111) dan [4.3.9](#_bookmark112).

1

2

**for** (**int** i=0; i<NSB; i++) {

**if** (set\_bit[mask][i] >= S[idx].

3

ret += F1(idx, mask, set\_bit[mask

*<→* length()) **break**;

4

5

6

}

*<→* ][i], dist);

**return** memoF[idx][mask][dist] = ret;

}

Kode Sumber 4.3.8 Fungsi F (1)

Kode Sumber 4.3.9 Fungsi F (2)

1

**if** (set\_bit[mask][i] >= S[idx].

2

ret += F1(idx, mask, set\_bit[mask

*<→* length()) **break**;

3

4

5

}

*<→* ][i], dist);

**return** memoF[idx][mask][dist] = ret;

}

# Implementasi Fungsi F1

Fungsi F1 adalah implementasi dari perancangan pada *pseudo- code* pada Gambar [3.6](#_bookmark75) yang dirancang berdasarkan persamaan

[2.4.3](#_bookmark51). Implementasi dari fungsi F1 dapat dilihat pada Kode Sumber [4.3.10](#_bookmark114).

1 **long long** F1(**int** idx, **int** mask, **int** charIdx, **int**

2

*<→* dist) {

3

**int** curIdx = S[idx].length() -

*<→*  builtin\_popcount(mask);

**if** (charIdx == S[idx].length() - 1 ||

*<→* duplicate\_rule1(idx, mask, charIdx)

4

*<→* ) {

**return** F(idx, mask - (1 <<

*<→* charIdx), dist + abs(S[idx

*<→* ][charIdx] - S[idx][curIdx

5

6

7

}

**return** 0;

*<→* ]));

}

Kode Sumber 4.3.10 Fungsi F1

# Implementasi Fungsi G

Implementasi dari fungsi G dapat dilihat pada Kode Sumber [4.3.11](#_bookmark116). Algoritma pada fungsi F menggunakan pendekatan *dynamic pro- gramming*.

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

5

6

7

**int** NSB = set\_bit[mask].size();

**long long** ret = 0;

**for** (**int** i=0; i<NSB; i++) {

**if** (set\_bit[mask][i] >= S[idx].

*<→* memoG[idx][mask][dist];

8

ret += G1(idx, mask, set\_bit[mask

*<→* length()) **break**;

9

ret += G2(idx, mask, set\_bit[mask

*<→* ][i], dist);

10

ret += G3(idx, mask, set\_bit[mask

*<→* ][i], dist);

11

12

13

}

*<→* ][i], dist);

**return** memoG[idx][mask][dist] = ret;

}

Kode Sumber 4.3.11 Fungsi G

# Implementasi Fungsi G1

Fungsi G1 adalah implementasi dari perancangan pada *pseudocode* pada Gambar [3.9](#_bookmark85) yang dirancang berdasarkan persamaan [2.4.6](#_bookmark57). Im- plementasi dari fungsi G1 dapat dilihat pada Kode Sumber [4.3.12](#_bookmark118) dan [4.3.13](#_bookmark119).

Kode Sumber 4.3.12 Fungsi G1 (1)

1 **long long** G1(**int** idx, **int** mask, **int** charIdx, **int**

2

*<→* dist) {

3

**int** curIdx = S[idx].length() -

*<→*  builtin\_popcount(mask);

**if** (charIdx == S[idx].length() - 1 ||

*<→* duplicate\_rule1(idx, mask, charIdx)

*<→* ) {

1

**return** G(idx, mask - (1 <<

*<→* charIdx), dist + abs(S[idx

*<→* ][charIdx] - S[idx][curIdx

2

3

4

}

**return** 0;

*<→* ]));

}

Kode Sumber 4.3.13 Fungsi G1 (2)

# Implementasi Fungsi G2

Fungsi G2 adalah implementasi dari perancangan pada *pseudo- code* pada Gambar [3.10](#_bookmark86) yang dirancang berdasarkan persamaan

* + 1. Implementasi dari fungsi G2 dapat dilihat pada Kode Sumber [4.3.14](#_bookmark121).

1 **long long** G2(**int** idx, **int** mask, **int** charIdx, **int**

2

*<→* dist) {

3

**int** curIdx = S[idx].length() -

*<→*  builtin\_popcount(mask);

**if** (charIdx == S[idx].length() - 1 || (

*<→* duplicate\_rule2(idx, mask, charIdx)

*<→* && duplicate\_rule1(idx, mask,

4

*<→* charIdx))) {

**return** F(idx, mask - (1 <<

*<→* charIdx), dist + abs((S[idx

5 }

6 **return** 0;

7 }

*<→* ][charIdx] + 1) - S[idx][

*<→* curIdx]));

Kode Sumber 4.3.14 Fungsi G2

# Implementasi Fungsi G3

Fungsi G3 adalah implementasi dari perancangan pada *pseudo- code* pada Gambar [3.11](#_bookmark87) yang dirancang berdasarkan persamaan

* + 1. Implementasi dari fungsi G3 dapat dilihat pada Kode Sumber [4.3.15](#_bookmark123).

1 **long long** G3(**int** idx, **int** mask, **int** charIdx, **int**

2

*<→* dist) {

3

**int** curIdx = S[idx].length() -

*<→*  builtin\_popcount(mask);

**if** ((charIdx == S[idx].length() - 1 ||

*<→* duplicate\_rule1(idx, mask, charIdx)

*<→* ) && (charIdx == 0 ||

*<→* duplicate\_rule3(idx, mask, charIdx)

4

*<→* ) ) {

**return** F(idx, mask - (1 <<

*<→* charIdx), dist + abs((S[idx

*<→* ][charIdx] - 1) - S[idx][

5

6

7

}

**return** 0;

*<→* curIdx]));

}

Kode Sumber 4.3.15 Fungsi G3

# Implementasi Fungsi Duplicate Rule 1

Fungsi duplicate\_rule1 adalah implementasi dari perancangan pada

*pseudocode* pada Gambar [3.7](#_bookmark76) yang dirancang berdasarkan persa-

maan [2.4.4](#_bookmark52). Implementasi dari fungsi duplicate\_rule1 dapat dilihat pada Kode Sumber [4.3.16](#_bookmark125) and [4.3.17](#_bookmark126).

1 **bool** duplicate\_rule1(**int** idx, **int** mask, **int**

2

3

*<→* charIdx) {

**return** (charIdx < S[idx].length() - 1 && (S[idx][charIdx] != S[idx][

*<→* charIdx + 1]

Kode Sumber 4.3.16 Fungsi duplicate\_rule1 (1)

1

|| (S[idx][charIdx] == S[idx][

2

&& !isBitOn(mask, charIdx + 1))))

*<→* charIdx + 1]

3

}

*<→* ;

Kode Sumber 4.3.17 Fungsi duplicate\_rule1 (2)

# Implementasi Fungsi Duplicate Rule 2

Fungsi duplicate\_rule2 adalah implementasi dari perancangan pada *pseudocode* pada Gambar [3.12](#_bookmark88) yang dirancang berdasarkan persa- maan [2.4.9](#_bookmark60). Implementasi dari fungsi duplicate\_rule2 dapat dilihat pada Kode Sumber [4.3.18](#_bookmark128).

1 **bool** duplicate\_rule2(**int** idx, **int** mask, **int**

2

3

*<→* charIdx) {

**return** (charIdx < S[idx].length() - 1 && (charFirstPos[idx][(S[idx][

4

|| (charFirstPos[idx][(S[idx][

*<→* charIdx] + 1) - 'a'] == -1

5

&& !isBitOn(mask, charFirstPos[

*<→* charIdx]) + 1 - 'a'] != -1

*<→* idx][S[idx][charIdx] + 1 -

6

}

*<→* 'a']))));

Kode Sumber 4.3.18 Fungsi duplicate\_rule2

# Implementasi Fungsi Duplicate Rule 3

Fungsi duplicate\_rule3 adalah implementasi dari perancangan pada *pseudocode* pada Gambar [3.13](#_bookmark89) yang dirancang berdasarkan persamaan [2.4.10](#_bookmark61). Implementasi dari fungsi duplicate\_rule1 dapat dilihat pada Kode Sumber [4.3.19](#_bookmark130).

1 **bool** duplicate\_rule3(**int** idx, **int** mask, **int**

2

3

*<→* charIdx) {

**return** (charIdx > 0

&& (charLastPos[idx][(S[idx][

4

|| (charLastPos[idx][(S[idx][

*<→* charIdx] - 1) - 'a'] == -1

5

&& isBitOn(mask, charLastPos[idx

*<→* charIdx] - 1) - 'a'] != -1

*<→* ][(S[idx][charIdx] - 1) - '

6

}

*<→* a']))));

Kode Sumber 4.3.19 Fungsi duplicate\_rule3

*Halaman ini sengaja dikosongkan*

# BAB V

**UJI COBA DAN EVALUASI**

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

# 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 gcc versi 6.3.

# Uji Coba Kebenaran

Uji coba kebenaran dilakukan dengan mengirimkan kode sum- ber program ke dalam situs penilaian daring SPOJ dan melaku- kan hasil uji coba kasus sederhana dengan langkah-langkah sesu- ai 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 pa- da situs SPOJ ditunjukkan pada Gambar [B.1](#_bookmark181).

Selain itu, dilakukan pengujian sebanyak 30 kali pada situs peni- laian daring SPOJ untuk melihat variasi waktu dan memori yang dibutuhkan program. Hasil uji coba sebanyak 30 kali dapat dilihat pada Gambar [B.2](#_bookmark182), [B.3](#_bookmark183), [B.4](#_bookmark184) dan [B.5](#_bookmark185).

83

|  |  |
| --- | --- |
| Waktu Maksimal | 1*.*08 detik |
| Waktu Minimal | 0*.*98 detik |
| Waktu Rata-Rata | 1*.*01 detik |
| Memori Maksimal | 19 MB |
| Memori Minimal | 19 MB |
| Memori Rata-Rata | 19 MB |

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

Dari hasil uji coba pada Gambar [B.2](#_bookmark182), [B.3](#_bookmark183), [B.4](#_bookmark184) dan [B.5](#_bookmark185) dapat kita tarik beberapa informasi seperti yang tertera pada Tabel [5.1](#_bookmark134).

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. Ka- sus pertama yaitu kasus di mana nilai dari string *ad*1 = *c*, string *ad*2 = *n* dan *X* = 1.

Sesuai dengan algoritma yang telah dirancang, berdasarkan persa- maan [2.4.1](#_bookmark43) yang sudah ditransformasikan ke dalam bentuk *pseudo- code* pada Gambar [3.1](#_bookmark65), algoritma akan melakukan iterasi variabel *dist* dari 0 hingga *X*.

Berikutnya, pada iterasi *dist* = 1, nilai jawaban akhir akan ditam- bahkan dengan *F*(*c,*1*,*0), yang merupakan jumlah kombinasi string *orig*1 tanpa operasi *replace* dengan jarak 1 terhadap string *ad*1, yang berdasarkan ilustrasi pada Tabel [C.5](#_bookmark191), bernilai 0 yang dikalik- an dengan *G*(*n,*1*,*1), yang merupakan jumlah kombinasi string *orig*2 dengan operasi *replace* yang memiliki jarak 0 terhadap string *ad*2, yang berdasarkan ilustrasi pada Tabel [C.6](#_bookmark192), bernilai 0. Lalu nilai jawaban akhir akan ditambahkan dengan *G*(*c,*1*,*0), yang merupakan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan jarak 1 terhadap string *ad*1, yang berdasarkan ilustrasi pada Tabel [C.7](#_bookmark193),

bernilai 2 yang dikalikan dengan *F*(*n,*1*,*1), yang merupakan jumlah kombinasi string *orig*2 tanpa operasi *replace* yang memiliki jarak 0 terhadap string *ad*2, yang berdasarkan ilustrasi pada Tabel [C.8](#_bookmark194), bernilai 1. Sehingga nilai jawaban dari iterasi 1 adalah 2 dan nilai jawaban akhir pad kasus di mana string *ad*1 = *c*, string *ad*2 = *n* dan *X* = 1 adalah 4.

Kasus berikutnya adalah kasus di mana string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5. Proses yang dilakukan sama seperti pada kasus sebelumnya, yaitu dimulai dengan mengiterasi variable *dist* dari 0 hingga *X*.

Pada iterasi *dist* = 0, nilai jawaban akhir akan ditambahkan de- ngan *F*(*behkn,*31*,*5), yang merupakan jumlah kombinasi string *orig*1 tanpa operasi *replace* dengan jarak 0 terhadap string *ad*1, yang ber- dasarkan ilustrasi pada Tabel [D.1](#_bookmark196), bernilai 1 yang dikalikan dengan *G*(*behkn,*31*,*0), yang merupakan jumlah kombinasi string *orig*2 de- ngan operasi *replace* yang memiliki jarak 5 terhadap string *ad*2, yang berdasarkan ilustrasi pada Tabel [D.3](#_bookmark198) sampai dengan Tabel [D.8](#_bookmark203), bernilai 8. Lalu nilai jawaban akhir akan ditambahkan dengan *G*(*behkn,*31*,*5), yang merupakan jumlah kombinasi string *orig*1 de- ngan operasi *replace* dengan jarak 0 terhadap string *ad*1, yang ber- dasarkan ilustrasi pada Tabel [D.9](#_bookmark204) sampai dengan Tabel [D.11](#_bookmark206), ber- nilai 0 yang dikalikan dengan *F*(*behkn,*31*,*0), yang merupakan jumlah kombinasi string *orig*2 tanpa operasi *replace* yang memiliki jarak 5 terhadap string *ad*2, yang berdasarkan ilustrasi pada Tabel [D.12](#_bookmark207) sampai dengan Tabel [D.13](#_bookmark208), bernilai 0. Sehingga nilai jawaban da- ri iterasi 0 adalah 8 dan nilai jawaban akhir hingga pada iterasi 0 adalah 8.

Berikutnya, pada iterasi *dist* = 1, nilai jawaban akhir akan Beri- kutnya, pada iterasi *dist* = 1, nilai jawaban akhir akan ditambahk- an dengan *F*(*behkn,*31*,*4), yang merupakan jumlah kombinasi string *orig*1 tanpa operasi *replace* dengan jarak 1 terhadap string *ad*1, yang berdasarkan ilustrasi pada Tabel [D.14](#_bookmark209), bernilai 0 yang dikalik-

an dengan *G*(*behkn,*31*,*1), yang merupakan jumlah kombinasi string *orig*2 dengan operasi *replace* yang memiliki jarak 4 terhadap string *ad*2, yang berdasarkan ilustrasi pada Tabel [D.15](#_bookmark210) sampai dengan Ta- bel [D.19](#_bookmark214), bernilai 0. Lalu nilai jawaban akhir akan ditambahkan de- ngan *G*(*behkn,*31*,*4), yang merupakan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan jarak 1 terhadap string *ad*1, yang berdasarkan ilustrasi pada Tabel [D.20](#_bookmark215) sampai dengan Tabel [D.22](#_bookmark217), bernilai 10 yang dikalikan dengan *F*(*behkn,*31*,*1), yang merupakan jumlah kombinasi string *orig*2 tanpa operasi *replace* yang memili- ki jarak 4 terhadap string *ad*2, yang berdasarkan ilustrasi pada Tabel

[D.23](#_bookmark218) sampai dengan Tabel [D.23](#_bookmark218), 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 ditam- bahkan dengan *F*(*behkn,*31*,*3), yang merupakan jumlah kombinasi string *orig*1 tanpa operasi *replace* dengan jarak 2 terhadap string *ad*1, yang berdasarkan ilustrasi pada Tabel [D.24](#_bookmark219), bernilai 0 yang dikalikan dengan *G*(*behkn,*31*,*2), yang merupakan jumlah kombinasi string *orig*2 dengan operasi *replace* yang memiliki jarak 3 terha- dap string *ad*2, yang berdasarkan ilustrasi pada Tabel [D.25](#_bookmark220) sam- pai dengan Tabel [D.29](#_bookmark224), bernilai 0. Lalu nilai jawaban akhir akan ditambahkan dengan *G*(*behkn,*31*,*3), yang merupakan jumlah kom- binasi string *orig*1 dengan operasi *replace* dengan jarak 2 terha- dap string *ad*1, yang berdasarkan ilustrasi pada Tabel [D.30](#_bookmark225) sampai dengan Tabel [D.32](#_bookmark227), bernilai 0 yang dikalikan dengan *F*(*behkn,*31*,*2), yang merupakan jumlah kombinasi string *orig*2 tanpa operasi *re- place* yang memiliki jarak 3 terhadap string *ad*2, yang berdasarkan ilustrasi pada Tabel [D.33](#_bookmark228) sampai dengan Tabel [D.33](#_bookmark228), bernilai 0. Se- hingga 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 ditam- bahkan dengan *F*(*behkn,*31*,*2), yang merupakan jumlah kombinasi

string *orig*1 tanpa operasi *replace* dengan jarak 3 terhadap string *ad*1, yang berdasarkan ilustrasi pada Tabel [D.34](#_bookmark229), bernilai 0 yang dikalikan dengan *G*(*behkn,*31*,*3), yang merupakan jumlah kombinasi string *orig*2 dengan operasi *replace* yang memiliki jarak 2 terha- dap string *ad*2, yang berdasarkan ilustrasi pada Tabel [D.35](#_bookmark230) sam- pai dengan Tabel [D.37](#_bookmark232), bernilai 0. Lalu nilai jawaban akhir akan ditambahkan dengan *G*(*behkn,*31*,*2), yang merupakan jumlah kom- binasi string *orig*1 dengan operasi *replace* dengan jarak 3 terha- dap string *ad*1, yang berdasarkan ilustrasi pada Tabel [D.38](#_bookmark233) sampai dengan Tabel [D.42](#_bookmark237), bernilai 0 yang dikalikan dengan *F*(*behkn,*31*,*3), yang merupakan jumlah kombinasi string *orig*2 tanpa operasi *re- place* yang memiliki jarak 2 terhadap string *ad*2, yang berdasarkan ilustrasi pada Tabel [D.43](#_bookmark238) sampai dengan Tabel [D.43](#_bookmark238), bernilai 0. Se- hingga 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 ditam- bahkan dengan *F*(*behkn,*31*,*1), yang merupakan jumlah kombinasi string *orig*1 tanpa operasi *replace* dengan jarak 4 terhadap string *ad*1, yang berdasarkan ilustrasi pada Tabel [D.44](#_bookmark239), bernilai 0 yang dikalikan dengan *G*(*behkn,*31*,*4), yang merupakan jumlah kombinasi string *orig*2 dengan operasi *replace* yang memiliki jarak 1 terha- dap string *ad*2, yang berdasarkan ilustrasi pada Tabel [D.45](#_bookmark240) sam- pai dengan Tabel [D.47](#_bookmark242), bernilai 10. Lalu nilai jawaban akhir akan ditambahkan dengan *G*(*behkn,*31*,*1), yang merupakan jumlah kom- binasi string *orig*1 dengan operasi *replace* dengan jarak 4 terha- dap string *ad*1, yang berdasarkan ilustrasi pada Tabel [D.48](#_bookmark243) sampai dengan Tabel [D.52](#_bookmark247), bernilai 0 yang dikalikan dengan *F*(*behkn,*31*,*4), yang merupakan jumlah kombinasi string *orig*2 tanpa operasi *re- place* yang memiliki jarak 1 terhadap string *ad*2, yang berdasarkan ilustrasi pada Tabel [D.53](#_bookmark248) sampai dengan Tabel [D.53](#_bookmark248), bernilai 0. Se- hingga 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 ditam- bahkan dengan *F*(*behkn,*31*,*0), yang merupakan jumlah kombinasi string *orig*1 tanpa operasi *replace* dengan jarak 5 terhadap string *ad*1, yang berdasarkan ilustrasi pada Tabel [D.54](#_bookmark249), bernilai 0 yang dikalikan dengan *G*(*behkn,*31*,*5), yang merupakan jumlah kombinasi string *orig*2 dengan operasi *replace* yang memiliki jarak 0 terha- dap string *ad*2, yang berdasarkan ilustrasi pada Tabel [D.55](#_bookmark250) sam- pai dengan Tabel [D.57](#_bookmark252), bernilai 0. Lalu nilai jawaban akhir akan ditambahkan dengan *G*(*behkn,*31*,*0), yang merupakan jumlah kom- binasi string *orig*1 dengan operasi *replace* dengan jarak 5 terha- dap string *ad*1, yang berdasarkan ilustrasi pada Tabel [D.58](#_bookmark253) sampai dengan Tabel [D.62](#_bookmark257), bernilai 8 yang dikalikan dengan *F*(*behkn,*31*,*5), yang merupakan jumlah kombinasi string *orig*2 tanpa operasi *re- place* yang memiliki jarak 0 terhadap string *ad*2, yang berdasarkan ilustrasi pada Tabel [D.63](#_bookmark258) sampai dengan Tabel [D.63](#_bookmark258), bernilai 1. Se- hingga nilai jawaban dari iterasi 5 adalah 8 dan nilai jawaban akhir untuk kasus di mana string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 adalah 16.

# 5.3 Analisa Kompleksitas Waktu

Pada *pseudocode* pada Gambar [3.1](#_bookmark65), terdapat fungsi preprocess. Kompleksitas waktu dari fungsi preprocess adalah (2*|S| S* ) di mana *S* adalah string masukan.

O ∗ | |

Berikutnya untuk setiap kasus uji terdapat empat fungsi uta- ma. Dengan menggunakan *master theorem*, fungsi readInput memiliki kompleksitas (1). Berdasarkan *pseudocode* fung- si init() pada Gambar [3.3](#_bookmark69), fungsi init dapat dipecah menja- di dua bagian utama, yaitu inisialisasi *memoF* dan *memoG* dan inisialisasi *charF irstP os* dan *charLastP os*. Inisialisa- si *memoF* dan *memoG* masing-masing memiliki kompleksitas waktu (2 2*|S| M AX*\_*DIST* ) di mana *M AX*\_*DIST* ada- lah jarak maksimum antar string yang mungkin. Sedangkan ini-

O

O ∗ ∗

sialisasi *charF irstP os* dan *charLastP os* memiliki kompleksi- tas waktu O(|*S*|*log*|*S*| + |*S*|) atau dapat disederhanakan menjadi O(|*S*|*log*|*S*|). Sehingga fungsi ini memiliki kompleksitas O(2 ∗ 2*|S|* ∗ *M AX*\_*DIST* + |*S*|*log*|*S*|) dan dapat disederhanakan men- jadi O(2*|S|* ∗ *M AX*\_*DIST* ).

Fungsi berikutnya adalah fungsi solve. Pada fungsi solve ter- dapat sebuah iterasi sebanyak *min*(*M AX*\_*DIST, X*) di ma- na pada kasus terburuk, *min*(*M AX*\_*DIST, X*) bisa mencapai *M AX*\_*DIST* . Di dalam iterasi tersebut, fungsi solve memang- gil fungsi F dan fungsi G masing-masing sebanyak dua kali yang memiliki kompleksitas waktu (2 2*|S| M AX*\_*DIST* ) atau da- pat disederhanakan menjadi (2*|S| M AX*\_*DIST* ). Sehingga kompleksitas dari fungsi solve secara keseluruhan adalah (2*|S| M AX*\_*DIST* 2).

O ∗

O ∗

O ∗ ∗

Terakhir adalah fungsi writeOutput. Kompleksitas waktu dari fung- si writeOutput adalah (1). Sehingga secara keseluruhan, kom- pleksitas waktu dari algoritma yang telah dirancang pada Tugas Akhir ini adalah O(2*|S|* ∗ *M AX*\_*DIST* 2 ∗ *T* ).

O

Pada umumnya, eksekusi program pada situs penilaian daring SPOJ adalah 1 detik untuk setiap 100*.*000*.*000 proses. Pada kasus terbu- ruk, yaitu ketika *S* = 10, *M AX*\_*DIST* = 250 dan *T* = 10, ekse- kusi program dengan kompleksitas waktu (2*|S| M AX*\_*DIST* 2  *T* ) akan melakukan 640*.*000*.*000 proses dimana jika dengan meng- gunakan standar berupa 100*.*000*.*000 proses per detik, program akan membutuhkan waktu eksekusi sebesar 6*,* 4 detik. Sehingga Algori- tma dengan kompleksitas waktu (2*|S| M AX*\_*DIST* 2 *T* ) cukup untuk menyelesaikan permasalahan klasik SPOJ 9967 *Playing With Words*.

O ∗ ∗

| |

O ∗ ∗

*Halaman ini sengaja dikosongkan*

# BAB VI KESIMPULAN

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

# 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* da- pat menyelesaikan permasalahan permasalahan klasik SPOJ 9967 *Playing With Words* dengan benar.
    2. Kompleksitas waktu sebesar (2*|S| M AX*\_*DIST* 2 *T* )

O ∗ ∗

cukup untuk menyelesaikan permasalahan klasik SPOJ 9967

*Playing With Words*.

* + 1. Waktu yang dibutuhkan program untuk menyelesaikan permasalahan klasik SPOJ 9967 *Playing With Words* mini- mum 0*.*98 detik, maksimum 1*.*08 detik dan rata-rata 1*.*01 de- tik. Memori yang dibutuhkan adalah sebesar 19 MB.

# Saran

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

* + 1. Paradigma *dynamic programming* adalah pendekatan yang

91

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

* + 1. 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/> fineber- g/mfc158/week10lecture.htm. [Accessed 24-May-2017].
2. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, **In- troduction To Algorithm,**Second., Cambridge, Massachusetts London, England: The MIT Press, 2001.
3. S. Halim and F. Halim, **Competitive Programming 3**.Singa- pore, 2013.
4. E. Elmaghiraby, **Journal of Mathematical Analysis and Ap- plications,** vol. 29, no. 3, pp. 523–557, Mar. 1970.

93

*Halaman ini sengaja dikosongkan*

# LAMPIRAN A

**TABEL HIMPUNAN** *SET BIT* **SETELAH FUNGSI PREPROCESS DIJALANKAN**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.1 Tabel himpunan setBit setelah fungsi preprocess dijalankan (1)

95

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.2 Tabel himpunan setBit setelah fungsi preprocess dijalankan (2)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.3 Tabel himpunan setBit setelah fungsi preprocess dijalankan (3)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.4 Tabel himpunan setBit setelah fungsi preprocess dijalankan (4)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.5 Tabel himpunan setBit setelah fungsi preprocess dijalankan (5)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.6 Tabel himpunan setBit setelah fungsi preprocess dijalankan (6)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.7 Tabel himpunan setBit setelah fungsi preprocess dijalankan (7)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.8 Tabel himpunan setBit setelah fungsi preprocess dijalankan (8)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 233 | 0 | 3 | 5 | 6 | 7 | - | - | - | - | - |
| 234 | 1 | 3 | 5 | 6 | 7 | - | - | - | - | - |
| 235 | 0 | 1 | 3 | 5 | 6 | 7 | - | - | - | - |
| 236 | 2 | 3 | 5 | 6 | 7 | - | - | - | - | - |
| 237 | 0 | 2 | 3 | 5 | 6 | 7 | - | - | - | - |
| 238 | 1 | 2 | 3 | 5 | 6 | 7 | - | - | - | - |
| 239 | 0 | 1 | 2 | 3 | 5 | 6 | 7 | - | - | - |
| 240 | 4 | 5 | 6 | 7 | - | - | - | - | - | - |
| 241 | 0 | 4 | 5 | 6 | 7 | - | - | - | - | - |
| 242 | 1 | 4 | 5 | 6 | 7 | - | - | - | - | - |
| 243 | 0 | 1 | 4 | 5 | 6 | 7 | - | - | - | - |
| 244 | 2 | 4 | 5 | 6 | 7 | - | - | - | - | - |
| 245 | 0 | 2 | 4 | 5 | 6 | 7 | - | - | - | - |
| 246 | 1 | 2 | 4 | 5 | 6 | 7 | - | - | - | - |
| 247 | 0 | 1 | 2 | 4 | 5 | 6 | 7 | - | - | - |
| 248 | 3 | 4 | 5 | 6 | 7 | - | - | - | - | - |
| 249 | 0 | 3 | 4 | 5 | 6 | 7 | - | - | - | - |
| 250 | 1 | 3 | 4 | 5 | 6 | 7 | - | - | - | - |
| 251 | 0 | 1 | 3 | 4 | 5 | 6 | 7 | - | - | - |
| 252 | 2 | 3 | 4 | 5 | 6 | 7 | - | - | - | - |
| 253 | 0 | 2 | 3 | 4 | 5 | 6 | 7 | - | - | - |
| 254 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | - | - | - |
| 255 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | - | - |
| 256 | 8 | - | - | - | - | - | - | - | - | - |
| 257 | 0 | 8 | - | - | - | - | - | - | - | - |
| 258 | 1 | 8 | - | - | - | - | - | - | - | - |
| 259 | 0 | 1 | 8 | - | - | - | - | - | - | - |
| 260 | 2 | 8 | - | - | - | - | - | - | - | - |
| 261 | 0 | 2 | 8 | - | - | - | - | - | - | - |
| 262 | 1 | 2 | 8 | - | - | - | - | - | - | - |

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

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.10 Tabel himpunan setBit setelah fungsi preprocess dijalankan (10)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.11 Tabel himpunan setBit setelah fungsi preprocess dijalankan (11)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.12 Tabel himpunan setBit setelah fungsi preprocess dijalankan (12)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.13 Tabel himpunan setBit setelah fungsi preprocess dijalankan (13)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.14 Tabel himpunan setBit setelah fungsi preprocess dijalankan (14)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.15 Tabel himpunan setBit setelah fungsi preprocess dijalankan (15)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.16 Tabel himpunan setBit setelah fungsi preprocess dijalankan (16)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.17 Tabel himpunan setBit setelah fungsi preprocess dijalankan (17)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.18 Tabel himpunan setBit setelah fungsi preprocess dijalankan (18)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.19 Tabel himpunan setBit setelah fungsi preprocess dijalankan (19)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.20 Tabel himpunan setBit setelah fungsi preprocess dijalankan (20)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.21 Tabel himpunan setBit setelah fungsi preprocess dijalankan (21)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.22 Tabel himpunan setBit setelah fungsi preprocess dijalankan (22)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.23 Tabel himpunan setBit setelah fungsi preprocess dijalankan (23)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.24 Tabel himpunan setBit setelah fungsi preprocess dijalankan (24)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.25 Tabel himpunan setBit setelah fungsi preprocess dijalankan (25)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.26 Tabel himpunan setBit setelah fungsi preprocess dijalankan (26)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.27 Tabel himpunan setBit setelah fungsi preprocess dijalankan (27)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.28 Tabel himpunan setBit setelah fungsi preprocess dijalankan (28)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.29 Tabel himpunan setBit setelah fungsi preprocess dijalankan (29)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.30 Tabel himpunan setBit setelah fungsi preprocess dijalankan (30)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.31 Tabel himpunan setBit setelah fungsi preprocess dijalankan (31)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.32 Tabel himpunan setBit setelah fungsi preprocess dijalankan (32)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.33 Tabel himpunan setBit setelah fungsi preprocess dijalankan (33)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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.34 Tabel himpunan setBit setelah fungsi preprocess dijalankan (34)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index*  *N um* | 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 |

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

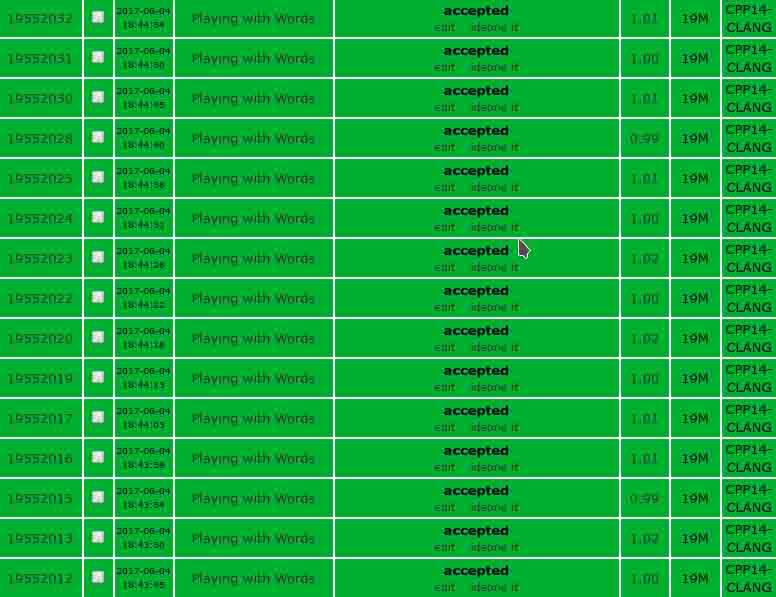
*Halaman ini sengaja dikosongkan*

# LAMPIRAN B

**HASIL UJI COBA KEBENARAN PADA SITUS SPOJ**

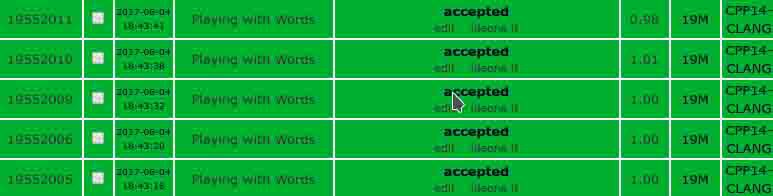


Gambar B.1 Hasil uji coba pada situs penilaian SPOJ

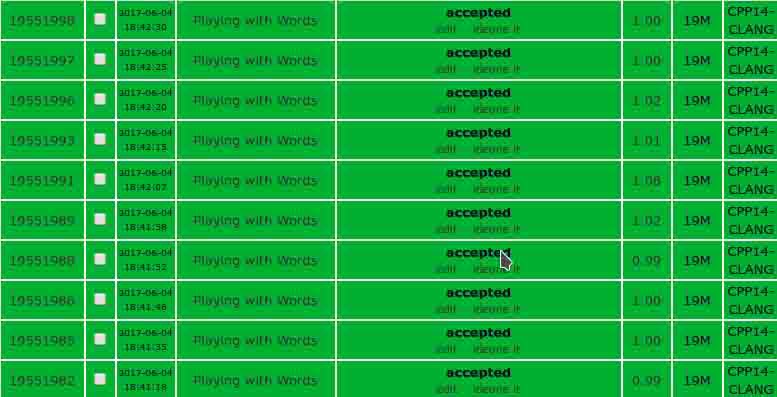


Gambar B.2 Hasil pengujian sebanyak 30 kali pada situs penilaian daring SPOJ (1)

131

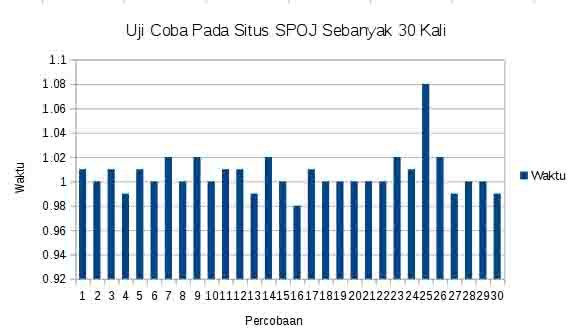


Gambar B.3 Hasil pengujian sebanyak 30 kali pada situs penilaian daring SPOJ (2)



Gambar B.4 Hasil pengujian sebanyak 30 kali pada situs penilaian daring SPOJ (3)

133



Gambar B.5 Grafik hasil uji coba pada situs SPOJ sebanyak 30 kali

*Halaman ini sengaja dikosongkan*

# LAMPIRAN C

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

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*c,*0*,*1) | *base case* | 1 |
| *F*(*c,*1*,*1) | *F*(*c,*0*,*1) | 1 |

Tabel C.1 Simulasi perhitungan jumlah kombinasi string *orig*1 tanpa ope- rasi *replace* dengan *dist* = 0 pada kasus string *ad*1 = *c*, string *ad*2 = *n* dan *X* = 1

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *G*(*n,*0*,*0) | *base case* | 0 |
| *F*(*n,*0*,*1) | *base case* | 1 |
| *F*(*n,*0*,*1) | *base case* | 1 |
| *G*(*n,*1*,*0) | *G*(*n,*0*,*0) + *F*(*n,*0*,*1) + *F*(*n,*0*,*1) | 2 |

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

135

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *G*(*c,*0*,*1) | *base case* | 0 |
| *F*(*c,*0*,*2) | *base case* | 0 |
| *F*(*c,*0*,*2) | *base case* | 0 |
| *G*(*c,*1*,*1) | *G*(*c,*0*,*1) + *F*(*c,*0*,*2) + *F*(*c,*0*,*2) | 0 |

Tabel C.3 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 0 pada kasus string *ad*1 = *c*, string *ad*2 = *n* dan *X* = 1

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*n,*0*,*0) | *base case* | 0 |
| *F*(*n,*1*,*0) | *F*(*n,*0*,*0) | 0 |

Tabel C.4 Simulasi perhitungan jumlah kombinasi string *orig*2 tanpa ope- rasi *replace* dengan *dist* = 0 pada kasus string *ad*1 = *c*, string *ad*2 = *n* dan *X* = 1

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*c,*0*,*0) | *base case* | 0 |
| *F*(*c,*1*,*0) | *F*(*c,*0*,*0) | 0 |

Tabel C.5 Simulasi perhitungan jumlah kombinasi string *orig*1 tanpa ope- rasi *replace* dengan *dist* = 1 pada kasus string *ad*1 = *c*, string *ad*2 = *n* dan *X* = 1

137

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *G*(*n,*0*,*1) | *base case* | 0 |
| *F*(*n,*0*,*2) | *base case* | 0 |
| *F*(*n,*0*,*2) | *base case* | 0 |
| *G*(*n,*1*,*1) | *G*(*n,*0*,*1) + *F*(*n,*0*,*2) + *F*(*n,*0*,*2) | 0 |

Tabel C.6 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 1 pada kasus string *ad*1 = *c*, string *ad*2 = *n* dan *X* = 1

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *G*(*c,*0*,*0) | *base case* | 0 |
| *F*(*c,*0*,*1) | *base case* | 1 |
| *F*(*c,*0*,*1) | *base case* | 1 |
| *G*(*c,*1*,*0) | *G*(*c,*0*,*0) + *F*(*c,*0*,*1) + *F*(*c,*0*,*1) | 2 |

Tabel C.7 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 1 pada kasus string *ad*1 = *c*, string *ad*2 = *n* dan *X* = 1

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*n,*0*,*1) | *base case* | 1 |
| *F*(*n,*1*,*1) | *F*(*n,*0*,*1) | 1 |

Tabel C.8 Simulasi perhitungan jumlah kombinasi string *orig*2 tanpa ope- rasi *replace* dengan *dist* = 1 pada kasus string *ad*1 = *c*, string *ad*2 = *n* dan *X* = 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**

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*0*,*5) | *base case* | 1 |
| *F*(*behkn,*16*,*5) | *F*(*behkn,*0*,*5) | 1 |
| *F*(*behkn,*8*,*8) | *base case* | 0 |
| *F*(*behkn,*24*,*5) | *F*(*behkn,*16*,*5) + *F*(*behkn,*8*,*8) | 1 |
| *F*(*behkn,*20*,*8) | *base case* | 0 |
| *F*(*behkn,*12*,*11) | *base case* | 0 |
| *F*(*behkn,*28*,*5) | *F*(*behkn,*24*,*5) + *F*(*behkn,*20*,*8) +  *F*(*behkn,*12*,*11) | 1 |
| *F*(*behkn,*26*,*8) | *base case* | 0 |
| *F*(*behkn,*22*,*11) | *base case* | 0 |
| *F*(*behkn,*14*,*14) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*27*,*11) | *base case* | 0 |
| *F*(*behkn,*23*,*14) | *base case* | 0 |

Tabel D.1 Simulasi perhitungan jumlah kombinasi string *orig*1 tanpa ope- rasi *replace* dengan *dist* = 0 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5

139

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*15*,*17) | *base case* | 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.2 Simulasi perhitungan jumlah kombinasi string *orig*1 tanpa ope- rasi *replace* dengan *dist* = 0 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fungsi | Perhitungan Nilai | | | Nilai |
| *G*(*behkn,*0*,*0) | *base case* | | | 0 |
| *F*(*behkn,*0*,*1) | *base case* | | | 0 |
| *F*(*behkn,*0*,*1) | *base case* | | | 0 |
| *G*(*behkn,*16*,*0) | *G*(*behkn,*0*,*0)  *F*(*behkn,*0*,*1) | + *F*(*behkn,*0*,*1) | + | 0 |
| *F*(*behkn,*0*,*1) | *base case* | | | 0 |
| *F*(*behkn,*16*,*1) | *F*(*behkn,*0*,*1) | | | 0 |
| *F*(*behkn,*16*,*1) | *memoF*(*behkn,*16*,*1) | | | 0 |
| *G*(*behkn,*0*,*6) | *base case* | | | 0 |
| *F*(*behkn,*0*,*5) | *base case* | | | 1 |
| *F*(*behkn,*0*,*7) | *base case* | | | 0 |
| *G*(*behkn,*8*,*3) | *G*(*behkn,*0*,*6)  *F*(*behkn,*0*,*7) | + *F*(*behkn,*0*,*5) | + | 1 |
| *F*(*behkn,*0*,*7) | *base case* | | | 0 |
| *F*(*behkn,*8*,*4) | *F*(*behkn,*0*,*7) | | | 0 |
| *F*(*behkn,*0*,*5) | *base case* | | | 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) | *memoF*(*behkn,*16*,*1) | | | 0 |
| *F*(*behkn,*8*,*4) | *memoF*(*behkn,*8*,*4) | | | 0 |
| *F*(*behkn,*24*,*1) | *F*(*behkn,*16*,*1) + *F*(*behkn,*8*,*4) | | | 0 |
| *F*(*behkn,*24*,*1) | *memoF*(*behkn,*24*,*1) | | | 0 |
| *G*(*behkn,*16*,*6) | *base case* | | | 0 |
| *F*(*behkn,*0*,*5) | *base case* | | | 1 |
| *F*(*behkn,*16*,*5) | *F*(*behkn,*0*,*5) | | | 1 |
| *F*(*behkn,*16*,*7) | *base case* | | | 0 |

Tabel D.3 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 0 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (1)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *G*(*behkn,*4*,*6) | *base case* | 0 |
| *F*(*behkn,*4*,*7) | *base case* | 0 |
| *F*(*behkn,*0*,*11) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*4*,*7) | *base case* | 0 |
| *F*(*behkn,*20*,*4) | *F*(*behkn,*16*,*7) + *F*(*behkn,*4*,*7) | 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) | *base case* | 0 |
| *F*(*behkn,*12*,*7) | *base case* | 0 |
| *F*(*behkn,*8*,*8) | *base case* | 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,*24*,*1) | *memoF*(*behkn,*24*,*1) | 0 |
| *F*(*behkn,*20*,*4) | *memoF*(*behkn,*20*,*4) | 0 |
| *F*(*behkn,*12*,*7) | *base case* | 0 |
| *F*(*behkn,*28*,*1) | *F*(*behkn,*24*,*1) + *F*(*behkn,*20*,*4) +  *F*(*behkn,*12*,*7) | 0 |

Tabel D.4 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 0 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (2)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*28*,*1) | *memoF*(*behkn,*28*,*1) | 0 |
| *G*(*behkn,*24*,*6) | *base case* | 0 |
| *F*(*behkn,*16*,*5) | *memoF*(*behkn,*16*,*5) | 1 |
| *F*(*behkn,*8*,*8) | *base case* | 0 |
| *F*(*behkn,*24*,*5) | *F*(*behkn,*16*,*5) + *F*(*behkn,*8*,*8) | 1 |
| *F*(*behkn,*24*,*7) | *base case* | 0 |
| *G*(*behkn,*18*,*6) | *base case* | 0 |
| *F*(*behkn,*18*,*7) | *base case* | 0 |
| *F*(*behkn,*16*,*11) | *base case* | 0 |
| *F*(*behkn,*2*,*8) | *base case* | 0 |
| *F*(*behkn,*18*,*5) | *F*(*behkn,*16*,*11) + *F*(*behkn,*2*,*8) | 0 |
| *G*(*behkn,*10*,*9) | *base case* | 0 |
| *F*(*behkn,*10*,*10) | *base case* | 0 |
| *F*(*behkn,*10*,*8) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*18*,*7) | *base case* | 0 |
| *F*(*behkn,*10*,*10) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*26*,*2) | *F*(*behkn,*24*,*5) + *F*(*behkn,*18*,*5) +  *F*(*behkn,*10*,*8) | 1 |

Tabel D.5 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 0 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (3)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *G*(*behkn,*22*,*6) | *base case* | 0 |
| *F*(*behkn,*22*,*7) | *base case* | 0 |
| *F*(*behkn,*20*,*8) | *base case* | 0 |
| *F*(*behkn,*18*,*5) | *memoF*(*behkn,*18*,*5) | 0 |
| *F*(*behkn,*6*,*11) | *base case* | 0 |
| *F*(*behkn,*22*,*5) | *F*(*behkn,*20*,*8) + *F*(*behkn,*18*,*5) +  *F*(*behkn,*6*,*11) | 0 |
| *G*(*behkn,*14*,*9) | *base case* | 0 |
| *F*(*behkn,*14*,*10) | *base case* | 0 |
| *F*(*behkn,*14*,*8) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*14*,*10) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*24*,*5) | *memoF*(*behkn,*24*,*5) | 1 |
| *F*(*behkn,*20*,*8) | *base case* | 0 |
| *F*(*behkn,*12*,*11) | *base case* | 0 |
| *F*(*behkn,*28*,*5) | *F*(*behkn,*24*,*5) + *F*(*behkn,*20*,*8) +  *F*(*behkn,*12*,*11) | 1 |

Tabel D.6 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 0 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (4)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*28*,*7) | *base case* | 0 |
| *G*(*behkn,*25*,*6) | *base case* | 0 |
| *F*(*behkn,*25*,*7) | *base case* | 0 |
| *F*(*behkn,*24*,*11) | *base case* | 0 |
| *F*(*behkn,*17*,*8) | *base case* | 0 |
| *F*(*behkn,*9*,*11) | *base case* | 0 |
| *F*(*behkn,*25*,*5) | *F*(*behkn,*24*,*11) + *F*(*behkn,*17*,*8) +  *F*(*behkn,*9*,*11) | 0 |
| *G*(*behkn,*21*,*9) | *base case* | 0 |
| *F*(*behkn,*21*,*10) | *base case* | 0 |
| *F*(*behkn,*21*,*8) | *base case* | 0 |
| *G*(*behkn,*13*,*12) | *base case* | 0 |
| *F*(*behkn,*13*,*13) | *base case* | 0 |
| *F*(*behkn,*13*,*11) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*25*,*7) | *base case* | 0 |
| *F*(*behkn,*21*,*10) | *base case* | 0 |
| *F*(*behkn,*13*,*13) | *base case* | 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) | *memoF*(*behkn,*28*,*5) | 1 |
| *F*(*behkn,*25*,*5) | *memoF*(*behkn,*25*,*5) | 0 |
| *F*(*behkn,*21*,*8) | *base case* | 0 |
| *F*(*behkn,*13*,*11) | *base case* | 0 |

Tabel D.7 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 0 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (5)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*27*,*7) | *base case* | 0 |
| *F*(*behkn,*26*,*8) | *base case* | 0 |
| *F*(*behkn,*25*,*5) | *memoF*(*behkn,*25*,*5) | 0 |
| *F*(*behkn,*19*,*11) | *base case* | 0 |
| *F*(*behkn,*11*,*14) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*23*,*10) | *base case* | 0 |
| *F*(*behkn,*23*,*8) | *base case* | 0 |
| *G*(*behkn,*15*,*12) | *base case* | 0 |
| *F*(*behkn,*15*,*13) | *base case* | 0 |
| *F*(*behkn,*15*,*11) | *base case* | 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.8 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 0 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (6)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *G*(*behkn,*0*,*5) | *base case* | 0 |
| *F*(*behkn,*0*,*6) | *base case* | 0 |
| *F*(*behkn,*0*,*6) | *base case* | 0 |
| *G*(*behkn,*16*,*5) | *G*(*behkn,*0*,*5) + *F*(*behkn,*0*,*6) +  *F*(*behkn,*0*,*6) | 0 |
| *F*(*behkn,*16*,*6) | *base case* | 0 |
| *F*(*behkn,*16*,*6) | *base case* | 0 |
| *G*(*behkn,*8*,*8) | *base case* | 0 |
| *F*(*behkn,*8*,*9) | *base case* | 0 |
| *F*(*behkn,*8*,*7) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*24*,*6) | *base case* | 0 |
| *G*(*behkn,*20*,*8) | *base case* | 0 |
| *F*(*behkn,*20*,*9) | *base case* | 0 |
| *F*(*behkn,*20*,*7) | *base case* | 0 |
| *G*(*behkn,*12*,*11) | *base case* | 0 |
| *F*(*behkn,*12*,*12) | *base case* | 0 |
| *F*(*behkn,*12*,*10) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*28*,*6) | *base case* | 0 |

Tabel D.9 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 0 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (1)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *G*(*behkn,*26*,*8) | *base case* | 0 |
| *F*(*behkn,*26*,*9) | *base case* | 0 |
| *F*(*behkn,*26*,*7) | *base case* | 0 |
| *G*(*behkn,*22*,*11) | *base case* | 0 |
| *F*(*behkn,*22*,*12) | *base case* | 0 |
| *F*(*behkn,*22*,*10) | *base case* | 0 |
| *G*(*behkn,*14*,*14) | *base case* | 0 |
| *F*(*behkn,*14*,*15) | *base case* | 0 |
| *F*(*behkn,*14*,*13) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*30*,*6) | *base case* | 0 |
| *G*(*behkn,*29*,*8) | *base case* | 0 |
| *F*(*behkn,*29*,*9) | *base case* | 0 |
| *F*(*behkn,*29*,*7) | *base case* | 0 |
| *G*(*behkn,*27*,*11) | *base case* | 0 |
| *F*(*behkn,*27*,*12) | *base case* | 0 |
| *F*(*behkn,*27*,*10) | *base case* | 0 |
| *G*(*behkn,*23*,*14) | *base case* | 0 |
| *F*(*behkn,*23*,*15) | *base case* | 0 |
| *F*(*behkn,*23*,*13) | *base case* | 0 |
| *G*(*behkn,*15*,*17) | *base case* | 0 |

Tabel D.10 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 0 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (2)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*15*,*18) | *base case* | 0 |
| *F*(*behkn,*15*,*16) | *base case* | 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.11 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 0 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (3)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*0*,*0) | *base case* | 0 |
| *F*(*behkn,*16*,*0) | *F*(*behkn,*0*,*0) | 0 |
| *F*(*behkn,*0*,*6) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*4*,*6) | *base case* | 0 |
| *F*(*behkn,*20*,*3) | *F*(*behkn,*16*,*6) + *F*(*behkn,*4*,*6) | 0 |
| *F*(*behkn,*12*,*6) | *base case* | 0 |
| *F*(*behkn,*28*,*0) | *F*(*behkn,*24*,*0) + *F*(*behkn,*20*,*3) +  *F*(*behkn,*12*,*6) | 0 |
| *F*(*behkn,*24*,*6) | *base case* | 0 |
| *F*(*behkn,*18*,*6) | *base case* | 0 |
| *F*(*behkn,*10*,*9) | *base case* | 0 |
| *F*(*behkn,*26*,*3) | *F*(*behkn,*24*,*6) + *F*(*behkn,*18*,*6) +  *F*(*behkn,*10*,*9) | 0 |
| *F*(*behkn,*22*,*6) | *base case* | 0 |
| *F*(*behkn,*14*,*9) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*25*,*6) | *base case* | 0 |
| *F*(*behkn,*21*,*9) | *base case* | 0 |
| *F*(*behkn,*13*,*12) | *base case* | 0 |
| *F*(*behkn,*29*,*3) | *F*(*behkn,*28*,*6) + *F*(*behkn,*25*,*6) +  *F*(*behkn,*21*,*9) + *F*(*behkn,*13*,*12) | 0 |

Tabel D.12 Simulasi perhitungan jumlah kombinasi string *orig*2 tanpa operasi *replace* dengan *dist* = 0 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (1)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*27*,*6) | *base case* | 0 |
| *F*(*behkn,*23*,*9) | *base case* | 0 |
| *F*(*behkn,*15*,*12) | *base case* | 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.13 Simulasi perhitungan jumlah kombinasi string *orig*2 tanpa operasi *replace* dengan *dist* = 0 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (2)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*0*,*4) | *base case* | 0 |
| *F*(*behkn,*16*,*4) | *F*(*behkn,*0*,*4) | 0 |
| *F*(*behkn,*8*,*7) | *base case* | 0 |
| *F*(*behkn,*24*,*4) | *F*(*behkn,*16*,*4) + *F*(*behkn,*8*,*7) | 0 |
| *F*(*behkn,*20*,*7) | *base case* | 0 |
| *F*(*behkn,*12*,*10) | *base case* | 0 |
| *F*(*behkn,*28*,*4) | *F*(*behkn,*24*,*4) + *F*(*behkn,*20*,*7) +  *F*(*behkn,*12*,*10) | 0 |
| *F*(*behkn,*26*,*7) | *base case* | 0 |
| *F*(*behkn,*22*,*10) | *base case* | 0 |
| *F*(*behkn,*14*,*13) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*27*,*10) | *base case* | 0 |
| *F*(*behkn,*23*,*13) | *base case* | 0 |
| *F*(*behkn,*15*,*16) | *base case* | 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.14 Simulasi perhitungan jumlah kombinasi string *orig*1 tanpa operasi *replace* dengan *dist* = 1 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fungsi | Perhitungan Nilai | | | Nilai |
| *G*(*behkn,*0*,*1) | *base case* | | | 0 |
| *F*(*behkn,*0*,*2) | *base case* | | | 0 |
| *F*(*behkn,*0*,*2) | *base case* | | | 0 |
| *G*(*behkn,*16*,*1) | *G*(*behkn,*0*,*1)  *F*(*behkn,*0*,*2) | + *F*(*behkn,*0*,*2) | + | 0 |
| *F*(*behkn,*0*,*2) | *base case* | | | 0 |
| *F*(*behkn,*16*,*2) | *F*(*behkn,*0*,*2) | | | 0 |
| *F*(*behkn,*16*,*2) | *memoF*(*behkn,*16*,*2) | | | 0 |
| *G*(*behkn,*0*,*7) | *base case* | | | 0 |
| *F*(*behkn,*0*,*6) | *base case* | | | 0 |
| *F*(*behkn,*0*,*8) | *base case* | | | 0 |
| *G*(*behkn,*8*,*4) | *G*(*behkn,*0*,*7)  *F*(*behkn,*0*,*8) | + *F*(*behkn,*0*,*6) | + | 0 |
| *F*(*behkn,*0*,*8) | *base case* | | | 0 |
| *F*(*behkn,*8*,*5) | *F*(*behkn,*0*,*8) | | | 0 |
| *F*(*behkn,*8*,*3) | *memoF*(*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) | *memoF*(*behkn,*16*,*2) | | | 0 |
| *F*(*behkn,*8*,*5) | *memoF*(*behkn,*8*,*5) | | | 0 |
| *F*(*behkn,*24*,*2) | *F*(*behkn,*16*,*2) + *F*(*behkn,*8*,*5) | | | 0 |
| *F*(*behkn,*24*,*2) | *memoF*(*behkn,*24*,*2) | | | 0 |
| *G*(*behkn,*16*,*7) | *base case* | | | 0 |
| *F*(*behkn,*16*,*6) | *base case* | | | 0 |
| *F*(*behkn,*16*,*8) | *base case* | | | 0 |

Tabel D.15 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 1 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (1)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *G*(*behkn,*4*,*7) | *base case* | 0 |
| *F*(*behkn,*4*,*8) | *base case* | 0 |
| *F*(*behkn,*4*,*6) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*4*,*8) | *base case* | 0 |
| *F*(*behkn,*20*,*5) | *F*(*behkn,*16*,*8) + *F*(*behkn,*4*,*8) | 0 |
| *F*(*behkn,*20*,*3) | *memoF*(*behkn,*20*,*3) | 0 |
| *G*(*behkn,*12*,*7) | *base case* | 0 |
| *F*(*behkn,*12*,*8) | *base case* | 0 |
| *F*(*behkn,*12*,*6) | *base case* | 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) | *memoF*(*behkn,*24*,*2) | 0 |
| *F*(*behkn,*20*,*5) | *memoF*(*behkn,*20*,*5) | 0 |
| *F*(*behkn,*12*,*8) | *base case* | 0 |
| *F*(*behkn,*28*,*2) | *F*(*behkn,*24*,*2) + *F*(*behkn,*20*,*5) +  *F*(*behkn,*12*,*8) | 0 |
| *F*(*behkn,*28*,*2) | *memoF*(*behkn,*28*,*2) | 0 |
| *G*(*behkn,*24*,*7) | *base case* | 0 |
| *F*(*behkn,*24*,*6) | *base case* | 0 |
| *F*(*behkn,*24*,*8) | *base case* | 0 |
| *G*(*behkn,*18*,*7) | *base case* | 0 |

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

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*18*,*8) | *base case* | 0 |
| *F*(*behkn,*18*,*6) | *base case* | 0 |
| *G*(*behkn,*10*,*10) | *base case* | 0 |
| *F*(*behkn,*10*,*11) | *base case* | 0 |
| *F*(*behkn,*10*,*9) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*18*,*8) | *base case* | 0 |
| *F*(*behkn,*10*,*11) | *base case* | 0 |
| *F*(*behkn,*26*,*5) | *F*(*behkn,*24*,*8) + *F*(*behkn,*18*,*8) +  *F*(*behkn,*10*,*11) | 0 |
| *F*(*behkn,*26*,*3) | *memoF*(*behkn,*26*,*3) | 0 |
| *G*(*behkn,*22*,*7) | *base case* | 0 |
| *F*(*behkn,*22*,*8) | *base case* | 0 |
| *F*(*behkn,*22*,*6) | *base case* | 0 |
| *G*(*behkn,*14*,*10) | *base case* | 0 |
| *F*(*behkn,*14*,*11) | *base case* | 0 |
| *F*(*behkn,*14*,*9) | *base case* | 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.17 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 1 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (3)

|  |  |  |
| --- | --- | --- |
| 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) | *base case* | 0 |
| *F*(*behkn,*14*,*11) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*28*,*6) | *base case* | 0 |
| *F*(*behkn,*28*,*8) | *base case* | 0 |
| *G*(*behkn,*25*,*7) | *base case* | 0 |
| *F*(*behkn,*25*,*8) | *base case* | 0 |
| *F*(*behkn,*25*,*6) | *base case* | 0 |
| *G*(*behkn,*21*,*10) | *base case* | 0 |
| *F*(*behkn,*21*,*11) | *base case* | 0 |
| *F*(*behkn,*21*,*9) | *base case* | 0 |
| *G*(*behkn,*13*,*13) | *base case* | 0 |
| *F*(*behkn,*13*,*14) | *base case* | 0 |
| *F*(*behkn,*13*,*12) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*25*,*8) | *base case* | 0 |
| *F*(*behkn,*21*,*11) | *base case* | 0 |

Tabel D.18 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 1 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (4)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*13*,*14) | *base case* | 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) | *memoF*(*behkn,*29*,*3) | 0 |
| *G*(*behkn,*27*,*7) | *base case* | 0 |
| *F*(*behkn,*27*,*8) | *base case* | 0 |
| *F*(*behkn,*27*,*6) | *base case* | 0 |
| *G*(*behkn,*23*,*10) | *base case* | 0 |
| *F*(*behkn,*23*,*11) | *base case* | 0 |
| *F*(*behkn,*23*,*9) | *base case* | 0 |
| *G*(*behkn,*15*,*13) | *base case* | 0 |
| *F*(*behkn,*15*,*14) | *base case* | 0 |
| *F*(*behkn,*15*,*12) | *base case* | 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.19 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 1 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (5)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *G*(*behkn,*0*,*4) | *base case* | 0 |
| *F*(*behkn,*0*,*5) | *base case* | 1 |
| *F*(*behkn,*0*,*5) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*8*,*8) | *base case* | 0 |
| *F*(*behkn,*8*,*6) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*20*,*8) | *base case* | 0 |
| *F*(*behkn,*20*,*6) | *base case* | 0 |
| *G*(*behkn,*12*,*10) | *base case* | 0 |
| *F*(*behkn,*12*,*11) | *base case* | 0 |
| *F*(*behkn,*12*,*9) | *base case* | 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.20 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 1 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (1)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *G*(*behkn,*26*,*7) | *base case* | 0 |
| *F*(*behkn,*26*,*8) | *base case* | 0 |
| *F*(*behkn,*26*,*6) | *base case* | 0 |
| *G*(*behkn,*22*,*10) | *base case* | 0 |
| *F*(*behkn,*22*,*11) | *base case* | 0 |
| *F*(*behkn,*22*,*9) | *base case* | 0 |
| *G*(*behkn,*14*,*13) | *base case* | 0 |
| *F*(*behkn,*14*,*14) | *base case* | 0 |
| *F*(*behkn,*14*,*12) | *base case* | 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) | *memoF*(*behkn,*30*,*5) | 1 |
| *F*(*behkn,*30*,*5) | *memoF*(*behkn,*30*,*5) | 1 |
| *G*(*behkn,*29*,*7) | *base case* | 0 |
| *F*(*behkn,*29*,*8) | *base case* | 0 |
| *F*(*behkn,*29*,*6) | *base case* | 0 |
| *G*(*behkn,*27*,*10) | *base case* | 0 |
| *F*(*behkn,*27*,*11) | *base case* | 0 |
| *F*(*behkn,*27*,*9) | *base case* | 0 |
| *G*(*behkn,*23*,*13) | *base case* | 0 |
| *F*(*behkn,*23*,*14) | *base case* | 0 |
| *F*(*behkn,*23*,*12) | *base case* | 0 |
| *G*(*behkn,*15*,*16) | *base case* | 0 |
| *F*(*behkn,*15*,*17) | *base case* | 0 |

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

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*15*,*15) | *base case* | 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.22 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 1 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (3)

|  |  |  |
| --- | --- | --- |
| 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) | *base case* | 0 |
| *F*(*behkn,*23*,*10) | *base case* | 0 |
| *F*(*behkn,*15*,*13) | *base case* | 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.23 Simulasi perhitungan jumlah kombinasi string *orig*2 tanpa operasi *replace* dengan *dist* = 1 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*0*,*3) | *base case* | 0 |
| *F*(*behkn,*16*,*3) | *F*(*behkn,*0*,*3) | 0 |
| *F*(*behkn,*8*,*6) | *base case* | 0 |
| *F*(*behkn,*24*,*3) | *F*(*behkn,*16*,*3) + *F*(*behkn,*8*,*6) | 0 |
| *F*(*behkn,*20*,*6) | *base case* | 0 |
| *F*(*behkn,*12*,*9) | *base case* | 0 |
| *F*(*behkn,*28*,*3) | *F*(*behkn,*24*,*3) + *F*(*behkn,*20*,*6) +  *F*(*behkn,*12*,*9) | 0 |
| *F*(*behkn,*26*,*6) | *base case* | 0 |
| *F*(*behkn,*22*,*9) | *base case* | 0 |
| *F*(*behkn,*14*,*12) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*27*,*9) | *base case* | 0 |
| *F*(*behkn,*23*,*12) | *base case* | 0 |
| *F*(*behkn,*15*,*15) | *base case* | 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.24 Simulasi perhitungan jumlah kombinasi string *orig*1 tanpa operasi *replace* dengan *dist* = 2 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fungsi | Perhitungan Nilai | | | Nilai |
| *G*(*behkn,*0*,*2) | *base case* | | | 0 |
| *F*(*behkn,*0*,*3) | *base case* | | | 0 |
| *F*(*behkn,*0*,*3) | *base case* | | | 0 |
| *G*(*behkn,*16*,*2) | *G*(*behkn,*0*,*2)  *F*(*behkn,*0*,*3) | + *F*(*behkn,*0*,*3) | + | 0 |
| *F*(*behkn,*0*,*3) | *base case* | | | 0 |
| *F*(*behkn,*16*,*3) | *F*(*behkn,*0*,*3) | | | 0 |
| *F*(*behkn,*16*,*3) | *memoF*(*behkn,*16*,*3) | | | 0 |
| *G*(*behkn,*0*,*8) | *base case* | | | 0 |
| *F*(*behkn,*0*,*7) | *base case* | | | 0 |
| *F*(*behkn,*0*,*9) | *base case* | | | 0 |
| *G*(*behkn,*8*,*5) | *G*(*behkn,*0*,*8)  *F*(*behkn,*0*,*9) | + *F*(*behkn,*0*,*7) | + | 0 |
| *F*(*behkn,*8*,*6) | *base case* | | | 0 |
| *F*(*behkn,*8*,*4) | *memoF*(*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) | *memoF*(*behkn,*16*,*3) | | | 0 |
| *F*(*behkn,*8*,*6) | *base case* | | | 0 |
| *F*(*behkn,*24*,*3) | *F*(*behkn,*16*,*3) + *F*(*behkn,*8*,*6) | | | 0 |
| *F*(*behkn,*24*,*3) | *memoF*(*behkn,*24*,*3) | | | 0 |
| *G*(*behkn,*16*,*8) | *base case* | | | 0 |
| *F*(*behkn,*16*,*7) | *base case* | | | 0 |
| *F*(*behkn,*16*,*9) | *base case* | | | 0 |
| *G*(*behkn,*4*,*8) | *base case* | | | 0 |

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

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*4*,*9) | *base case* | 0 |
| *F*(*behkn,*4*,*7) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*20*,*4) | *memoF*(*behkn,*20*,*4) | 0 |
| *G*(*behkn,*12*,*8) | *base case* | 0 |
| *F*(*behkn,*12*,*9) | *base case* | 0 |
| *F*(*behkn,*12*,*7) | *base case* | 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) | *memoF*(*behkn,*24*,*3) | 0 |
| *F*(*behkn,*20*,*6) | *base case* | 0 |
| *F*(*behkn,*12*,*9) | *base case* | 0 |
| *F*(*behkn,*28*,*3) | *F*(*behkn,*24*,*3) + *F*(*behkn,*20*,*6) +  *F*(*behkn,*12*,*9) | 0 |
| *F*(*behkn,*28*,*3) | *memoF*(*behkn,*28*,*3) | 0 |
| *G*(*behkn,*24*,*8) | *base case* | 0 |
| *F*(*behkn,*24*,*7) | *base case* | 0 |
| *F*(*behkn,*24*,*9) | *base case* | 0 |
| *G*(*behkn,*18*,*8) | *base case* | 0 |
| *F*(*behkn,*18*,*9) | *base case* | 0 |
| *F*(*behkn,*18*,*7) | *base case* | 0 |

Tabel D.26 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 2 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (2)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *G*(*behkn,*10*,*11) | *base case* | 0 |
| *F*(*behkn,*10*,*12) | *base case* | 0 |
| *F*(*behkn,*10*,*10) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*26*,*4) | *memoF*(*behkn,*26*,*4) | 0 |
| *G*(*behkn,*22*,*8) | *base case* | 0 |
| *F*(*behkn,*22*,*9) | *base case* | 0 |
| *F*(*behkn,*22*,*7) | *base case* | 0 |
| *G*(*behkn,*14*,*11) | *base case* | 0 |
| *F*(*behkn,*14*,*12) | *base case* | 0 |
| *F*(*behkn,*14*,*10) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*22*,*9) | *base case* | 0 |
| *F*(*behkn,*14*,*12) | *base case* | 0 |
| *F*(*behkn,*30*,*3) | *F*(*behkn,*28*,*3) + *F*(*behkn,*26*,*6) +  *F*(*behkn,*22*,*9) + *F*(*behkn,*14*,*12) | 0 |

Tabel D.27 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 2 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (3)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*30*,*3) | *memoF*(*behkn,*30*,*3) | 0 |
| *G*(*behkn,*28*,*8) | *base case* | 0 |
| *F*(*behkn,*28*,*7) | *base case* | 0 |
| *F*(*behkn,*28*,*9) | *base case* | 0 |
| *G*(*behkn,*25*,*8) | *base case* | 0 |
| *F*(*behkn,*25*,*9) | *base case* | 0 |
| *F*(*behkn,*25*,*7) | *base case* | 0 |
| *G*(*behkn,*21*,*11) | *base case* | 0 |
| *F*(*behkn,*21*,*12) | *base case* | 0 |
| *F*(*behkn,*21*,*10) | *base case* | 0 |
| *G*(*behkn,*13*,*14) | *base case* | 0 |
| *F*(*behkn,*13*,*15) | *base case* | 0 |
| *F*(*behkn,*13*,*13) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*29*,*4) | *memoF*(*behkn,*29*,*4) | 0 |
| *G*(*behkn,*27*,*8) | *base case* | 0 |
| *F*(*behkn,*27*,*9) | *base case* | 0 |
| *F*(*behkn,*27*,*7) | *base case* | 0 |
| *G*(*behkn,*23*,*11) | *base case* | 0 |
| *F*(*behkn,*23*,*12) | *base case* | 0 |
| *F*(*behkn,*23*,*10) | *base case* | 0 |
| *G*(*behkn,*15*,*14) | *base case* | 0 |

Tabel D.28 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 2 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (4)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*15*,*15) | *base case* | 0 |
| *F*(*behkn,*15*,*13) | *base case* | 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.29 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 2 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (5)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *G*(*behkn,*0*,*3) | *base case* | 0 |
| *F*(*behkn,*0*,*4) | *base case* | 0 |
| *F*(*behkn,*0*,*4) | *base case* | 0 |
| *G*(*behkn,*16*,*3) | *G*(*behkn,*0*,*3) + *F*(*behkn,*0*,*4) +  *F*(*behkn,*0*,*4) | 0 |
| *F*(*behkn,*16*,*4) | *memoF*(*behkn,*16*,*4) | 0 |
| *F*(*behkn,*16*,*4) | *memoF*(*behkn,*16*,*4) | 0 |
| *G*(*behkn,*8*,*6) | *base case* | 0 |
| *F*(*behkn,*8*,*7) | *base case* | 0 |
| *F*(*behkn,*0*,*8) | *base case* | 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) | *memoF*(*behkn,*24*,*4) | 0 |
| *F*(*behkn,*24*,*4) | *memoF*(*behkn,*24*,*4) | 0 |
| *G*(*behkn,*20*,*6) | *base case* | 0 |
| *F*(*behkn,*20*,*7) | *base case* | 0 |
| *F*(*behkn,*16*,*8) | *base case* | 0 |
| *F*(*behkn,*4*,*8) | *base case* | 0 |
| *F*(*behkn,*20*,*5) | *F*(*behkn,*16*,*8) + *F*(*behkn,*4*,*8) | 0 |
| *G*(*behkn,*12*,*9) | *base case* | 0 |
| *F*(*behkn,*12*,*10) | *base case* | 0 |
| *F*(*behkn,*12*,*8) | *base case* | 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.30 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 2 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (1)

|  |  |  |
| --- | --- | --- |
| 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) | *base case* | 0 |
| *F*(*behkn,*26*,*7) | *base case* | 0 |
| *F*(*behkn,*24*,*8) | *base case* | 0 |
| *F*(*behkn,*18*,*8) | *base case* | 0 |
| *F*(*behkn,*10*,*11) | *base case* | 0 |
| *F*(*behkn,*26*,*5) | *F*(*behkn,*24*,*8) + *F*(*behkn,*18*,*8) +  *F*(*behkn,*10*,*11) | 0 |
| *G*(*behkn,*22*,*9) | *base case* | 0 |
| *F*(*behkn,*22*,*10) | *base case* | 0 |
| *F*(*behkn,*22*,*8) | *base case* | 0 |
| *G*(*behkn,*14*,*12) | *base case* | 0 |
| *F*(*behkn,*14*,*13) | *base case* | 0 |
| *F*(*behkn,*14*,*11) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*29*,*7) | *base case* | 0 |
| *F*(*behkn,*28*,*8) | *base case* | 0 |
| *F*(*behkn,*25*,*8) | *base case* | 0 |

Tabel D.31 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 2 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (2)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*21*,*11) | *base case* | 0 |
| *F*(*behkn,*13*,*14) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*27*,*10) | *base case* | 0 |
| *F*(*behkn,*27*,*8) | *base case* | 0 |
| *G*(*behkn,*23*,*12) | *base case* | 0 |
| *F*(*behkn,*23*,*13) | *base case* | 0 |
| *F*(*behkn,*23*,*11) | *base case* | 0 |
| *G*(*behkn,*15*,*15) | *base case* | 0 |
| *F*(*behkn,*15*,*16) | *base case* | 0 |
| *F*(*behkn,*15*,*14) | *base case* | 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.32 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 2 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (3)

|  |  |  |
| --- | --- | --- |
| 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) | *base case* | 0 |
| *F*(*behkn,*23*,*11) | *base case* | 0 |
| *F*(*behkn,*15*,*14) | *base case* | 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.33 Simulasi perhitungan jumlah kombinasi string *orig*2 tanpa operasi *replace* dengan *dist* = 2 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*0*,*2) | *base case* | 0 |
| *F*(*behkn,*16*,*2) | *F*(*behkn,*0*,*2) | 0 |
| *F*(*behkn,*8*,*5) | *memoF*(*behkn,*8*,*5) | 0 |
| *F*(*behkn,*24*,*2) | *F*(*behkn,*16*,*2) + *F*(*behkn,*8*,*5) | 0 |
| *F*(*behkn,*20*,*5) | *memoF*(*behkn,*20*,*5) | 0 |
| *F*(*behkn,*12*,*8) | *base case* | 0 |
| *F*(*behkn,*28*,*2) | *F*(*behkn,*24*,*2) + *F*(*behkn,*20*,*5) +  *F*(*behkn,*12*,*8) | 0 |
| *F*(*behkn,*26*,*5) | *memoF*(*behkn,*26*,*5) | 0 |
| *F*(*behkn,*22*,*8) | *base case* | 0 |
| *F*(*behkn,*14*,*11) | *base case* | 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) | *memoF*(*behkn,*29*,*5) | 0 |
| *F*(*behkn,*27*,*8) | *base case* | 0 |
| *F*(*behkn,*23*,*11) | *base case* | 0 |
| *F*(*behkn,*15*,*14) | *base case* | 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 *orig*1 tanpa operasi *replace* dengan *dist* = 3 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *G*(*behkn,*0*,*3) | *base case* | 0 |
| *F*(*behkn,*0*,*4) | *base case* | 0 |
| *F*(*behkn,*0*,*4) | *base case* | 0 |
| *G*(*behkn,*16*,*3) | *G*(*behkn,*0*,*3) + *F*(*behkn,*0*,*4) +  *F*(*behkn,*0*,*4) | 0 |
| *F*(*behkn,*0*,*4) | *base case* | 0 |
| *F*(*behkn,*16*,*4) | *F*(*behkn,*0*,*4) | 0 |
| *F*(*behkn,*16*,*4) | *memoF*(*behkn,*16*,*4) | 0 |
| *G*(*behkn,*8*,*6) | *base case* | 0 |
| *F*(*behkn,*8*,*7) | *base case* | 0 |
| *F*(*behkn,*8*,*5) | *memoF*(*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) | *memoF*(*behkn,*16*,*4) | 0 |
| *F*(*behkn,*8*,*7) | *base case* | 0 |
| *F*(*behkn,*24*,*4) | *F*(*behkn,*16*,*4) + *F*(*behkn,*8*,*7) | 0 |
| *F*(*behkn,*24*,*4) | *memoF*(*behkn,*24*,*4) | 0 |
| *G*(*behkn,*20*,*6) | *base case* | 0 |
| *F*(*behkn,*20*,*7) | *base case* | 0 |
| *F*(*behkn,*20*,*5) | *memoF*(*behkn,*20*,*5) | 0 |
| *G*(*behkn,*12*,*9) | *base case* | 0 |
| *F*(*behkn,*12*,*10) | *base case* | 0 |
| *F*(*behkn,*12*,*8) | *base case* | 0 |

Tabel D.35 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 3 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (1)

|  |  |  |
| --- | --- | --- |
| 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) | *memoF*(*behkn,*24*,*4) | 0 |
| *F*(*behkn,*20*,*7) | *base case* | 0 |
| *F*(*behkn,*12*,*10) | *base case* | 0 |
| *F*(*behkn,*28*,*4) | *F*(*behkn,*24*,*4) + *F*(*behkn,*20*,*7) +  *F*(*behkn,*12*,*10) | 0 |
| *F*(*behkn,*28*,*4) | *memoF*(*behkn,*28*,*4) | 0 |
| *G*(*behkn,*26*,*6) | *base case* | 0 |
| *F*(*behkn,*26*,*7) | *base case* | 0 |
| *F*(*behkn,*26*,*5) | *memoF*(*behkn,*26*,*5) | 0 |
| *G*(*behkn,*22*,*9) | *base case* | 0 |
| *F*(*behkn,*22*,*10) | *base case* | 0 |
| *F*(*behkn,*22*,*8) | *base case* | 0 |
| *G*(*behkn,*14*,*12) | *base case* | 0 |
| *F*(*behkn,*14*,*13) | *base case* | 0 |
| *F*(*behkn,*14*,*11) | *base case* | 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) | *memoF*(*behkn,*28*,*4) | 0 |
| *F*(*behkn,*26*,*7) | *base case* | 0 |

Tabel D.36 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 3 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (2)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*22*,*10) | *base case* | 0 |
| *F*(*behkn,*14*,*13) | *base case* | 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) | *memoF*(*behkn,*30*,*4) | 0 |
| *G*(*behkn,*29*,*6) | *base case* | 0 |
| *F*(*behkn,*29*,*7) | *base case* | 0 |
| *F*(*behkn,*29*,*5) | *memoF*(*behkn,*29*,*5) | 0 |
| *G*(*behkn,*27*,*9) | *base case* | 0 |
| *F*(*behkn,*27*,*10) | *base case* | 0 |
| *F*(*behkn,*27*,*8) | *base case* | 0 |
| *G*(*behkn,*23*,*12) | *base case* | 0 |
| *F*(*behkn,*23*,*13) | *base case* | 0 |
| *F*(*behkn,*23*,*11) | *base case* | 0 |
| *G*(*behkn,*15*,*15) | *base case* | 0 |
| *F*(*behkn,*15*,*16) | *base case* | 0 |
| *F*(*behkn,*15*,*14) | *base case* | 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.37 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 3 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fungsi | Perhitungan Nilai | | | Nilai |
| *G*(*behkn,*0*,*2) | *base case* | | | 0 |
| *F*(*behkn,*0*,*3) | *base case* | | | 0 |
| *F*(*behkn,*0*,*3) | *base case* | | | 0 |
| *G*(*behkn,*16*,*2) | *G*(*behkn,*0*,*2)  *F*(*behkn,*0*,*3) | + *F*(*behkn,*0*,*3) | + | 0 |
| *F*(*behkn,*16*,*3) | *memoF*(*behkn,*16*,*3) | | | 0 |
| *F*(*behkn,*16*,*3) | *memoF*(*behkn,*16*,*3) | | | 0 |
| *G*(*behkn,*0*,*8) | *base case* | | | 0 |
| *F*(*behkn,*0*,*7) | *base case* | | | 0 |
| *F*(*behkn,*0*,*9) | *base case* | | | 0 |
| *G*(*behkn,*8*,*5) | *G*(*behkn,*0*,*8)  *F*(*behkn,*0*,*9) | + *F*(*behkn,*0*,*7) | + | 0 |
| *F*(*behkn,*8*,*6) | *base case* | | | 0 |
| *F*(*behkn,*0*,*7) | *base case* | | | 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) | *memoF*(*behkn,*24*,*3) | | | 0 |
| *F*(*behkn,*24*,*3) | *memoF*(*behkn,*24*,*3) | | | 0 |
| *G*(*behkn,*16*,*8) | *base case* | | | 0 |
| *F*(*behkn,*16*,*7) | *base case* | | | 0 |
| *F*(*behkn,*16*,*9) | *base case* | | | 0 |
| *G*(*behkn,*4*,*8) | *base case* | | | 0 |
| *F*(*behkn,*4*,*9) | *base case* | | | 0 |
| *F*(*behkn,*4*,*7) | *base case* | | | 0 |

Tabel D.38 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 3 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (1)

|  |  |  |
| --- | --- | --- |
| 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) | *base case* | 0 |
| *F*(*behkn,*16*,*7) | *base case* | 0 |
| *F*(*behkn,*4*,*7) | *base case* | 0 |
| *F*(*behkn,*20*,*4) | *F*(*behkn,*16*,*7) + *F*(*behkn,*4*,*7) | 0 |
| *G*(*behkn,*12*,*8) | *base case* | 0 |
| *F*(*behkn,*12*,*9) | *base case* | 0 |
| *F*(*behkn,*12*,*7) | *base case* | 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) | *memoF*(*behkn,*28*,*3) | 0 |
| *F*(*behkn,*28*,*3) | *memoF*(*behkn,*28*,*3) | 0 |
| *G*(*behkn,*24*,*8) | *base case* | 0 |
| *F*(*behkn,*24*,*7) | *base case* | 0 |
| *F*(*behkn,*24*,*9) | *base case* | 0 |
| *G*(*behkn,*18*,*8) | *base case* | 0 |
| *F*(*behkn,*18*,*9) | *base case* | 0 |
| *F*(*behkn,*18*,*7) | *base case* | 0 |
| *G*(*behkn,*10*,*11) | *base case* | 0 |
| *F*(*behkn,*10*,*12) | *base case* | 0 |
| *F*(*behkn,*10*,*10) | *base case* | 0 |

Tabel D.39 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 3 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (2)

|  |  |  |
| --- | --- | --- |
| 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) | *base case* | 0 |
| *F*(*behkn,*24*,*7) | *base case* | 0 |
| *F*(*behkn,*18*,*7) | *base case* | 0 |
| *F*(*behkn,*10*,*10) | *base case* | 0 |
| *F*(*behkn,*26*,*4) | *F*(*behkn,*24*,*7) + *F*(*behkn,*18*,*7) +  *F*(*behkn,*10*,*10) | 0 |
| *G*(*behkn,*22*,*8) | *base case* | 0 |
| *F*(*behkn,*22*,*9) | *base case* | 0 |
| *F*(*behkn,*22*,*7) | *base case* | 0 |
| *G*(*behkn,*14*,*11) | *base case* | 0 |
| *F*(*behkn,*14*,*12) | *base case* | 0 |
| *F*(*behkn,*14*,*10) | *base case* | 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) | *memoF*(*behkn,*30*,*3) | 0 |
| *F*(*behkn,*30*,*3) | *memoF*(*behkn,*30*,*3) | 0 |
| *G*(*behkn,*28*,*8) | *base case* | 0 |
| *F*(*behkn,*28*,*7) | *base case* | 0 |
| *F*(*behkn,*28*,*9) | *base case* | 0 |

Tabel D.40 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 3 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (3)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *G*(*behkn,*25*,*8) | *base case* | 0 |
| *F*(*behkn,*25*,*9) | *base case* | 0 |
| *F*(*behkn,*25*,*7) | *base case* | 0 |
| *G*(*behkn,*21*,*11) | *base case* | 0 |
| *F*(*behkn,*21*,*12) | *base case* | 0 |
| *F*(*behkn,*21*,*10) | *base case* | 0 |
| *G*(*behkn,*13*,*14) | *base case* | 0 |
| *F*(*behkn,*13*,*15) | *base case* | 0 |
| *F*(*behkn,*13*,*13) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*28*,*7) | *base case* | 0 |
| *F*(*behkn,*25*,*7) | *base case* | 0 |
| *F*(*behkn,*21*,*10) | *base case* | 0 |
| *F*(*behkn,*13*,*13) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*27*,*9) | *base case* | 0 |
| *F*(*behkn,*27*,*7) | *base case* | 0 |
| *G*(*behkn,*23*,*11) | *base case* | 0 |
| *F*(*behkn,*23*,*12) | *base case* | 0 |

Tabel D.41 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 3 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (4)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*23*,*10) | *base case* | 0 |
| *G*(*behkn,*15*,*14) | *base case* | 0 |
| *F*(*behkn,*15*,*15) | *base case* | 0 |
| *F*(*behkn,*15*,*13) | *base case* | 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.42 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 3 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (5)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*30*,*3) | *memoF*(*behkn,*30*,*3) | 0 |
| *F*(*behkn,*29*,*6) | *base case* | 0 |
| *F*(*behkn,*27*,*9) | *base case* | 0 |
| *F*(*behkn,*23*,*12) | *base case* | 0 |
| *F*(*behkn,*15*,*15) | *base case* | 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.43 Simulasi perhitungan jumlah kombinasi string *orig*2 tanpa operasi *replace* dengan *dist* = 3 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*0*,*1) | *base case* | 0 |
| *F*(*behkn,*16*,*1) | *F*(*behkn,*0*,*1) | 0 |
| *F*(*behkn,*8*,*4) | *memoF*(*behkn,*8*,*4) | 0 |
| *F*(*behkn,*24*,*1) | *F*(*behkn,*16*,*1) + *F*(*behkn,*8*,*4) | 0 |
| *F*(*behkn,*20*,*4) | *memoF*(*behkn,*20*,*4) | 0 |
| *F*(*behkn,*12*,*7) | *base case* | 0 |
| *F*(*behkn,*28*,*1) | *F*(*behkn,*24*,*1) + *F*(*behkn,*20*,*4) +  *F*(*behkn,*12*,*7) | 0 |
| *F*(*behkn,*26*,*4) | *memoF*(*behkn,*26*,*4) | 0 |
| *F*(*behkn,*22*,*7) | *base case* | 0 |
| *F*(*behkn,*14*,*10) | *base case* | 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) | *memoF*(*behkn,*29*,*4) | 0 |
| *F*(*behkn,*27*,*7) | *base case* | 0 |
| *F*(*behkn,*23*,*10) | *base case* | 0 |
| *F*(*behkn,*15*,*13) | *base case* | 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.44 Simulasi perhitungan jumlah kombinasi string *orig*1 tanpa operasi *replace* dengan *dist* = 4 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *G*(*behkn,*0*,*4) | *base case* | 0 |
| *F*(*behkn,*0*,*5) | *base case* | 1 |
| *F*(*behkn,*0*,*5) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*8*,*8) | *base case* | 0 |
| *F*(*behkn,*8*,*6) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*20*,*8) | *base case* | 0 |
| *F*(*behkn,*20*,*6) | *base case* | 0 |
| *G*(*behkn,*12*,*10) | *base case* | 0 |
| *F*(*behkn,*12*,*11) | *base case* | 0 |
| *F*(*behkn,*12*,*9) | *base case* | 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.45 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 4 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (1)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *G*(*behkn,*26*,*7) | *base case* | 0 |
| *F*(*behkn,*26*,*8) | *base case* | 0 |
| *F*(*behkn,*26*,*6) | *base case* | 0 |
| *G*(*behkn,*22*,*10) | *base case* | 0 |
| *F*(*behkn,*22*,*11) | *base case* | 0 |
| *F*(*behkn,*22*,*9) | *base case* | 0 |
| *G*(*behkn,*14*,*13) | *base case* | 0 |
| *F*(*behkn,*14*,*14) | *base case* | 0 |
| *F*(*behkn,*14*,*12) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*22*,*11) | *base case* | 0 |
| *F*(*behkn,*14*,*14) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*29*,*8) | *base case* | 0 |

Tabel D.46 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 4 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (2)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*29*,*6) | *base case* | 0 |
| *G*(*behkn,*27*,*10) | *base case* | 0 |
| *F*(*behkn,*27*,*11) | *base case* | 0 |
| *F*(*behkn,*27*,*9) | *base case* | 0 |
| *G*(*behkn,*23*,*13) | *base case* | 0 |
| *F*(*behkn,*23*,*14) | *base case* | 0 |
| *F*(*behkn,*23*,*12) | *base case* | 0 |
| *G*(*behkn,*15*,*16) | *base case* | 0 |
| *F*(*behkn,*15*,*17) | *base case* | 0 |
| *F*(*behkn,*15*,*15) | *base case* | 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.47 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 4 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fungsi | Perhitungan Nilai | | | Nilai |
| *G*(*behkn,*0*,*1) | *base case* | | | 0 |
| *F*(*behkn,*0*,*2) | *base case* | | | 0 |
| *F*(*behkn,*0*,*2) | *base case* | | | 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) | *base case* | | | 0 |
| *F*(*behkn,*0*,*6) | *base case* | | | 0 |
| *F*(*behkn,*0*,*8) | *base case* | | | 0 |
| *G*(*behkn,*8*,*4) | *G*(*behkn,*0*,*7)  *F*(*behkn,*0*,*8) | + *F*(*behkn,*0*,*6) | + | 0 |
| *F*(*behkn,*8*,*5) | *memoF*(*behkn,*8*,*5) | | | 0 |
| *F*(*behkn,*0*,*6) | *base case* | | | 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) | *base case* | | | 0 |
| *F*(*behkn,*16*,*6) | *base case* | | | 0 |
| *F*(*behkn,*16*,*8) | *base case* | | | 0 |
| *G*(*behkn,*4*,*7) | *base case* | | | 0 |
| *F*(*behkn,*4*,*8) | *base case* | | | 0 |
| *F*(*behkn,*4*,*6) | *base case* | | | 0 |

Tabel D.48 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 4 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (1)

|  |  |  |
| --- | --- | --- |
| 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) | *base case* | 0 |
| *F*(*behkn,*4*,*6) | *base case* | 0 |
| *F*(*behkn,*20*,*3) | *F*(*behkn,*16*,*6) + *F*(*behkn,*4*,*6) | 0 |
| *G*(*behkn,*12*,*7) | *base case* | 0 |
| *F*(*behkn,*12*,*8) | *base case* | 0 |
| *F*(*behkn,*12*,*6) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*24*,*6) | *base case* | 0 |
| *F*(*behkn,*24*,*8) | *base case* | 0 |
| *G*(*behkn,*18*,*7) | *base case* | 0 |
| *F*(*behkn,*18*,*8) | *base case* | 0 |
| *F*(*behkn,*18*,*6) | *base case* | 0 |
| *G*(*behkn,*10*,*10) | *base case* | 0 |
| *F*(*behkn,*10*,*11) | *base case* | 0 |
| *F*(*behkn,*10*,*9) | *base case* | 0 |

Tabel D.49 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 4 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (2)

|  |  |  |
| --- | --- | --- |
| 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) | *memoF*(*behkn,*26*,*5) | 0 |
| *F*(*behkn,*24*,*6) | *base case* | 0 |
| *F*(*behkn,*18*,*6) | *base case* | 0 |
| *F*(*behkn,*10*,*9) | *base case* | 0 |
| *F*(*behkn,*26*,*3) | *F*(*behkn,*24*,*6) + *F*(*behkn,*18*,*6) +  *F*(*behkn,*10*,*9) | 0 |
| *G*(*behkn,*22*,*7) | *base case* | 0 |
| *F*(*behkn,*22*,*8) | *base case* | 0 |
| *F*(*behkn,*22*,*6) | *base case* | 0 |
| *G*(*behkn,*14*,*10) | *base case* | 0 |
| *F*(*behkn,*14*,*11) | *base case* | 0 |
| *F*(*behkn,*14*,*9) | *base case* | 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) | *memoF*(*behkn,*30*,*2) | 0 |
| *F*(*behkn,*30*,*2) | *memoF*(*behkn,*30*,*2) | 0 |
| *G*(*behkn,*28*,*7) | *base case* | 0 |
| *F*(*behkn,*28*,*6) | *base case* | 0 |
| *F*(*behkn,*28*,*8) | *base case* | 0 |
| *G*(*behkn,*25*,*7) | *base case* | 0 |

Tabel D.50 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 4 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (3)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*25*,*8) | *base case* | 0 |
| *F*(*behkn,*25*,*6) | *base case* | 0 |
| *G*(*behkn,*21*,*10) | *base case* | 0 |
| *F*(*behkn,*21*,*11) | *base case* | 0 |
| *F*(*behkn,*21*,*9) | *base case* | 0 |
| *G*(*behkn,*13*,*13) | *base case* | 0 |
| *F*(*behkn,*13*,*14) | *base case* | 0 |
| *F*(*behkn,*13*,*12) | *base case* | 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) | *memoF*(*behkn,*29*,*5) | 0 |
| *F*(*behkn,*28*,*6) | *base case* | 0 |
| *F*(*behkn,*25*,*6) | *base case* | 0 |
| *F*(*behkn,*21*,*9) | *base case* | 0 |
| *F*(*behkn,*13*,*12) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*27*,*8) | *base case* | 0 |
| *F*(*behkn,*27*,*6) | *base case* | 0 |
| *G*(*behkn,*23*,*10) | *base case* | 0 |
| *F*(*behkn,*23*,*11) | *base case* | 0 |
| *F*(*behkn,*23*,*9) | *base case* | 0 |

Tabel D.51 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 4 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (4)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *G*(*behkn,*15*,*13) | *base case* | 0 |
| *F*(*behkn,*15*,*14) | *base case* | 0 |
| *F*(*behkn,*15*,*12) | *base case* | 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.52 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 4 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (5)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*30*,*4) | *memoF*(*behkn,*30*,*4) | 0 |
| *F*(*behkn,*29*,*7) | *base case* | 0 |
| *F*(*behkn,*27*,*10) | *base case* | 0 |
| *F*(*behkn,*23*,*13) | *base case* | 0 |
| *F*(*behkn,*15*,*16) | *base case* | 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.53 Simulasi perhitungan jumlah kombinasi string *orig*2 tanpa operasi *replace* dengan *dist* = 4 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*0*,*0) | *base case* | 0 |
| *F*(*behkn,*16*,*0) | *F*(*behkn,*0*,*0) | 0 |
| *F*(*behkn,*8*,*3) | *memoF*(*behkn,*8*,*3) | 0 |
| *F*(*behkn,*24*,*0) | *F*(*behkn,*16*,*0) + *F*(*behkn,*8*,*3) | 0 |
| *F*(*behkn,*20*,*3) | *memoF*(*behkn,*20*,*3) | 0 |
| *F*(*behkn,*12*,*6) | *base case* | 0 |
| *F*(*behkn,*28*,*0) | *F*(*behkn,*24*,*0) + *F*(*behkn,*20*,*3) +  *F*(*behkn,*12*,*6) | 0 |
| *F*(*behkn,*26*,*3) | *memoF*(*behkn,*26*,*3) | 0 |
| *F*(*behkn,*22*,*6) | *base case* | 0 |
| *F*(*behkn,*14*,*9) | *base case* | 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) | *memoF*(*behkn,*29*,*3) | 0 |
| *F*(*behkn,*27*,*6) | *base case* | 0 |
| *F*(*behkn,*23*,*9) | *base case* | 0 |
| *F*(*behkn,*15*,*12) | *base case* | 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.54 Simulasi perhitungan jumlah kombinasi string *orig*1 tanpa operasi *replace* dengan *dist* = 5 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Fungsi | Perhitungan Nilai | | | | Nilai |
| *G*(*behkn,*0*,*5) | *base case* | | | | 0 |
| *F*(*behkn,*0*,*6) | *base case* | | | | 0 |
| *F*(*behkn,*0*,*6) | *base case* | | | | 0 |
| *G*(*behkn,*16*,*5) | *G*(*behkn,*0*,*5)  *F*(*behkn,*0*,*6) | + | *F*(*behkn,*0*,*6) | + | 0 |
| *F*(*behkn,*16*,*6) | *base case* | | | | 0 |
| *F*(*behkn,*16*,*6) | *base case* | | | | 0 |
| *G*(*behkn,*8*,*8) | *base case* | | | | 0 |
| *F*(*behkn,*8*,*9) | *base case* | | | | 0 |
| *F*(*behkn,*8*,*7) | *base case* | | | | 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) | *base case* | | | | 0 |
| *F*(*behkn,*24*,*6) | *base case* | | | | 0 |
| *G*(*behkn,*20*,*8) | *base case* | | | | 0 |
| *F*(*behkn,*20*,*9) | *base case* | | | | 0 |
| *F*(*behkn,*20*,*7) | *base case* | | | | 0 |
| *G*(*behkn,*12*,*11) | *base case* | | | | 0 |
| *F*(*behkn,*12*,*12) | *base case* | | | | 0 |
| *F*(*behkn,*12*,*10) | *base case* | | | | 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) | *base case* | | | | 0 |
| *F*(*behkn,*28*,*6) | *base case* | | | | 0 |

Tabel D.55 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 5 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (1)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *G*(*behkn,*26*,*8) | *base case* | 0 |
| *F*(*behkn,*26*,*9) | *base case* | 0 |
| *F*(*behkn,*26*,*7) | *base case* | 0 |
| *G*(*behkn,*22*,*11) | *base case* | 0 |
| *F*(*behkn,*22*,*12) | *base case* | 0 |
| *F*(*behkn,*22*,*10) | *base case* | 0 |
| *G*(*behkn,*14*,*14) | *base case* | 0 |
| *F*(*behkn,*14*,*15) | *base case* | 0 |
| *F*(*behkn,*14*,*13) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*30*,*6) | *base case* | 0 |
| *G*(*behkn,*29*,*8) | *base case* | 0 |
| *F*(*behkn,*29*,*9) | *base case* | 0 |
| *F*(*behkn,*29*,*7) | *base case* | 0 |
| *G*(*behkn,*27*,*11) | *base case* | 0 |
| *F*(*behkn,*27*,*12) | *base case* | 0 |
| *F*(*behkn,*27*,*10) | *base case* | 0 |
| *G*(*behkn,*23*,*14) | *base case* | 0 |
| *F*(*behkn,*23*,*15) | *base case* | 0 |
| *F*(*behkn,*23*,*13) | *base case* | 0 |
| *G*(*behkn,*15*,*17) | *base case* | 0 |

Tabel D.56 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 5 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (2)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*15*,*18) | *base case* | 0 |
| *F*(*behkn,*15*,*16) | *base case* | 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.57 Simulasi perhitungan jumlah kombinasi string *orig*2 dengan operasi *replace* dengan *dist* = 5 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fungsi | Perhitungan Nilai | | | Nilai |
| *G*(*behkn,*0*,*0) | *base case* | | | 0 |
| *F*(*behkn,*0*,*1) | *base case* | | | 0 |
| *F*(*behkn,*0*,*1) | *base case* | | | 0 |
| *G*(*behkn,*16*,*0) | *G*(*behkn,*0*,*0)  *F*(*behkn,*0*,*1) | + *F*(*behkn,*0*,*1) | + | 0 |
| *F*(*behkn,*16*,*1) | *memoF*(*behkn,*16*,*1) | | | 0 |
| *F*(*behkn,*16*,*1) | *memoF*(*behkn,*16*,*1) | | | 0 |
| *G*(*behkn,*0*,*6) | *base case* | | | 0 |
| *F*(*behkn,*0*,*5) | *base case* | | | 1 |
| *F*(*behkn,*0*,*7) | *base case* | | | 0 |
| *G*(*behkn,*8*,*3) | *G*(*behkn,*0*,*6)  *F*(*behkn,*0*,*7) | + *F*(*behkn,*0*,*5) | + | 1 |
| *F*(*behkn,*8*,*4) | *memoF*(*behkn,*8*,*4) | | | 0 |
| *F*(*behkn,*0*,*5) | *base case* | | | 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) | *memoF*(*behkn,*24*,*1) | | | 0 |
| *F*(*behkn,*24*,*1) | *memoF*(*behkn,*24*,*1) | | | 0 |
| *G*(*behkn,*16*,*6) | *base case* | | | 0 |
| *F*(*behkn,*16*,*5) | *memoF*(*behkn,*16*,*5) | | | 1 |
| *F*(*behkn,*16*,*7) | *base case* | | | 0 |
| *G*(*behkn,*4*,*6) | *base case* | | | 0 |
| *F*(*behkn,*4*,*7) | *base case* | | | 0 |
| *F*(*behkn,*0*,*11) | *base case* | | | 0 |
| *F*(*behkn,*4*,*5) | *F*(*behkn,*0*,*11) | | | 0 |

Tabel D.58 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 5 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (1)

|  |  |  |
| --- | --- | --- |
| 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) | *base case* | 0 |
| *F*(*behkn,*12*,*7) | *base case* | 0 |
| *F*(*behkn,*8*,*8) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*24*,*5) | *memoF*(*behkn,*24*,*5) | 1 |
| *F*(*behkn,*24*,*7) | *base case* | 0 |
| *G*(*behkn,*18*,*6) | *base case* | 0 |
| *F*(*behkn,*18*,*7) | *base case* | 0 |
| *F*(*behkn,*16*,*11) | *base case* | 0 |
| *F*(*behkn,*2*,*8) | *base case* | 0 |
| *F*(*behkn,*18*,*5) | *F*(*behkn,*16*,*11) + *F*(*behkn,*2*,*8) | 0 |
| *G*(*behkn,*10*,*9) | *base case* | 0 |
| *F*(*behkn,*10*,*10) | *base case* | 0 |

Tabel D.59 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 5 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (2)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*10*,*8) | *base case* | 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) | *memoF*(*behkn,*26*,*4) | 0 |
| *F*(*behkn,*24*,*5) | *memoF*(*behkn,*24*,*5) | 1 |
| *F*(*behkn,*18*,*5) | *memoF*(*behkn,*18*,*5) | 0 |
| *F*(*behkn,*10*,*8) | *base case* | 0 |
| *F*(*behkn,*26*,*2) | *F*(*behkn,*24*,*5) + *F*(*behkn,*18*,*5) +  *F*(*behkn,*10*,*8) | 1 |
| *G*(*behkn,*22*,*6) | *base case* | 0 |
| *F*(*behkn,*22*,*7) | *base case* | 0 |
| *F*(*behkn,*20*,*8) | *base case* | 0 |
| *F*(*behkn,*18*,*5) | *memoF*(*behkn,*18*,*5) | 0 |
| *F*(*behkn,*6*,*11) | *base case* | 0 |
| *F*(*behkn,*22*,*5) | *F*(*behkn,*20*,*8) + *F*(*behkn,*18*,*5) +  *F*(*behkn,*6*,*11) | 0 |
| *G*(*behkn,*14*,*9) | *base case* | 0 |
| *F*(*behkn,*14*,*10) | *base case* | 0 |
| *F*(*behkn,*14*,*8) | *base case* | 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) | *memoF*(*behkn,*30*,*1) | 0 |

Tabel D.60 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 5 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (3)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*30*,*1) | *memoF*(*behkn,*30*,*1) | 0 |
| *G*(*behkn,*28*,*6) | *base case* | 0 |
| *F*(*behkn,*28*,*5) | *memoF*(*behkn,*28*,*5) | 1 |
| *F*(*behkn,*28*,*7) | *base case* | 0 |
| *G*(*behkn,*25*,*6) | *base case* | 0 |
| *F*(*behkn,*25*,*7) | *base case* | 0 |
| *F*(*behkn,*24*,*11) | *base case* | 0 |
| *F*(*behkn,*17*,*8) | *base case* | 0 |
| *F*(*behkn,*9*,*11) | *base case* | 0 |
| *F*(*behkn,*25*,*5) | *F*(*behkn,*24*,*11) + *F*(*behkn,*17*,*8) +  *F*(*behkn,*9*,*11) | 0 |
| *G*(*behkn,*21*,*9) | *base case* | 0 |
| *F*(*behkn,*21*,*10) | *base case* | 0 |
| *F*(*behkn,*21*,*8) | *base case* | 0 |
| *G*(*behkn,*13*,*12) | *base case* | 0 |
| *F*(*behkn,*13*,*13) | *base case* | 0 |
| *F*(*behkn,*13*,*11) | *base case* | 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.61 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 5 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (4)

|  |  |  |
| --- | --- | --- |
| 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) | *base case* | 0 |
| *F*(*behkn,*13*,*11) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*27*,*7) | *base case* | 0 |
| *F*(*behkn,*26*,*8) | *base case* | 0 |
| *F*(*behkn,*25*,*5) | *memoF*(*behkn,*25*,*5) | 0 |
| *F*(*behkn,*19*,*11) | *base case* | 0 |
| *F*(*behkn,*11*,*14) | *base case* | 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) | *base case* | 0 |
| *F*(*behkn,*23*,*10) | *base case* | 0 |
| *F*(*behkn,*23*,*8) | *base case* | 0 |
| *G*(*behkn,*15*,*12) | *base case* | 0 |
| *F*(*behkn,*15*,*13) | *base case* | 0 |
| *F*(*behkn,*15*,*11) | *base case* | 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.62 Simulasi perhitungan jumlah kombinasi string *orig*1 dengan operasi *replace* dengan *dist* = 5 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5 (5)

|  |  |  |
| --- | --- | --- |
| Fungsi | Perhitungan Nilai | Nilai |
| *F*(*behkn,*30*,*5) | *memoF*(*behkn,*30*,*5) | 1 |
| *F*(*behkn,*29*,*8) | *base case* | 0 |
| *F*(*behkn,*27*,*11) | *base case* | 0 |
| *F*(*behkn,*23*,*14) | *base case* | 0 |
| *F*(*behkn,*15*,*17) | *base case* | 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.63 Simulasi perhitungan jumlah kombinasi string *orig*2 tanpa operasi *replace* dengan *dist* = 5 pada kasus string *ad*1 = *kbenh*, string *ad*2 = *kbenh* dan *X* = 5

# BIODATA PENULIS

**Dewangga Winasforcepta Winardi**, lahir di Surabaya tanggal 18 Mei 1995. Penu- lis 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 Tek- nik Informatika ITS.

Selama kuliah di Teknik Informatika ITS, penulis mengambil bidang minat Algoritma

Pemrograman (AP). Penulis pernah menjadi asisten dosen dan prak- tikum untuk mata kuliah Dasar Pemrograman (2015 dan 2016), Struktur data (2015 dan 2016). Selama menempuh perkuliahan pe- nulis juga aktif mengikut kompetisi pemrograman tingkat nasio- nal dan menjadi Juara 2 kategori pemrograman pada lomba COM- PFEST Universitas Indonesia 2014. Selain itu penulis juga aktif di kegiatan organisasi dan kepanitiaan diantaranya menjadi staff De- partemen Riset dan Teknologi HMTC ITS, wakil ketua National Programming Contest Schematics 2014, ketua National Program- ming Contest 2014, panitia Pemusatan Latihan Nasional 2 TOKI 2014, 2015, 2016 dan 2017 di ITS dan technical comitee Olimpiade Sains Nasional 2015 di Jogjakarta. Penulis dapat dihubungi melalui surel di

[dewangga.winardi@gmail.com.](mailto:dewangga.winardi@gmail.com)

199