

**DEVELOPMENT
OF
CROSSWORD PUZZLE GENERATOR**

Tugas Akhir

Disampaikan sebagai Bagian dari Persyaratan Kelulusan Diploma 3
Program Studi Teknik Informatika

Oleh:

Elga F. Silaban	11105022
Amson Ujung	11105081
Grace Isabella	11105083



**Politeknik Informatika Del
2008**

**Lembar Pengesahan Tugas Akhir
Politeknik Informatika Del**

**DEVELOPMENT
OF
CROSSWORD PUZZLE GENERATOR**

Oleh:

Elga F. Silaban	11105022
Amson Ujung	11105081
Grace Isabella	11105083

**Dinyatakan memenuhi syarat dan karenanya disetujui dan disahkan sebagai
Laporan Tugas Akhir Diploma 3
Program Studi Teknik Informasi**

Pembimbing 1,

Pembimbing 2,

Fazat N. Azizah, ST, MSc.
NIDN. 0415097402

Gandhi M.T. Manalu, S.Si
NIDN. 0118078201

Prakata

Puji syukur kepada Tuhan Yang Maha Esa atas rahmat dan karunia-Nya, Laporan Tugas Akhir (TA) dengan judul "*Development of Crossword Puzzle Generator*" dapat diselesaikan.

Penulisan laporan ini disusun sebagai dokumentasi laporan pelaksanaan Tugas Akhir mengenai pembangunan *Crossword Puzzle Generator* yang telah kami lakukan. Kami berharap semoga laporan ini dapat memberikan manfaat bagi pembaca terutama bagi pembaca yang ingin mengetahui dan mempelajari *Crossword Puzzle Generator*.

Kami mengucapkan terimakasih kepada Bapak Gandhi Maruli Tua Manalu sebagai pembimbing, yang telah memberikan petunjuk, bimbingan, dan saran perbaikan selama pelaksanaan Tugas Akhir.

Kami juga mengucapkan terimakasih kepada koordinator Tugas Akhir yang telah memberi motivasi dan bimbingan dalam persiapan dan pelaksanaan TA. Kami juga mengucapkan terimakasih kepada penguji yang telah memberikan kritik dan saran yang membangun bagi kesempurnaan dan kelengkapan dokumen laporan Tugas Akhir ini.

Kami juga mengucapkan terimakasih kepada direktur Politeknik Informatika Del yaitu Ibu Dr.Ir.Inggriani Liem yang telah memberikan masukan dan bimbingan dalam penyelesaian TA, dan kepada seluruh pihak yang telah memberikan dukungan dan bantuan selama proses penyelesaian Tugas Akhir.

Kami menyadari masih banyak kekurangan yang terdapat dalam laporan ini, untuk itu kritik dan saran yang membangun dari seluruh pembaca sangat kami harapkan demi terwujudnya kesempurnaan laporan Tugas Akhir ini.

Laguboti, 02 September 2008

Elga F. Silaban 11105022

Amson Ujung 11105081

Grace Isabella 11105083

Abstrak

Teka teki silang (*crossword puzzle*) merupakan permainan di mana pemain akan mengisi huruf-huruf yang merupakan jawaban dari pertanyaan ke dalam kotak-kotak yang disusun secara menurun atau mendatar, berdasarkan petunjuk yang diberikan. Pengisian juga memperhatikan kesesuaian huruf dari kata yang bersilangan dengan kata yang lain. Tujuan utama permainan ini adalah mengisi semua kotak kosong dengan kata yang merupakan jawaban dari pertanyaan. Proses pembuatan permainan ini membutuhkan waktu yang relatif lama apabila dilakukan secara manual. Oleh sebab itu, diperlukan suatu *generator* yang dapat menghasilkan *crossword puzzle*.

Pada Tugas Akhir ini, akan dihasilkan *generator* yang dapat menghasilkan *crossword puzzle*. Pendekatan yang dilakukan untuk menyelesaikan Tugas Akhir ini adalah dengan melakukan studi literatur mengenai *crossword puzzle*, algoritma *brute force* dan *backtracking*, kemudian mempelajari hasil yang diperoleh dari studi literatur, merancang program yang akan dibangun, dan mengimplementasikannya dengan menggunakan bahasa pemrograman **Java**, kemudian melakukan pengujian untuk mengetahui jumlah *crossword puzzle* yang *valid* yang dihasilkan oleh *Crossword Puzzle Generator*.

Crossword Puzzle Generator yang dibangun dapat menghasilkan *crossword puzzle*, tetapi belum menerapkan keseluruhan aturan untuk *crossword puzzle*. Aturan yang belum diterapkan yaitu aturan bahwa setiap kata tidak boleh saling tumpang tindih.

Hasil yang diperoleh dari pengerjaan Tugas Akhir ini masih belum sempurna karena masih terdapat *crossword puzzle* yang tidak *valid* dan program belum mampu menghasilkan semua *crossword puzzle* yang mungkin dihasilkan sesuai dengan yang diterapkan pada algoritma *backtracking*. Guna penyempurnaan dan pengembangan selanjutnya, sebaiknya dilakukan penghapusan terhadap solusi sebelumnya, sehingga tidak terdapat kata-kata yang tumpang tindih.

Kata kunci: *crossword puzzle*, *crossword puzzle generator*, *brute force*, *backtracking*.

DAFTAR ISI

Prakata.....	3
Abstrak.....	4
Bab 1. Pendahuluan	8
1.1 Latar Belakang	8
1.2 Tujuan	8
1.3 Lingkup	8
1.4 Pendekatan	9
1.5 Sistematika Penyajian	10
Bab 2. Tinjauan Pustaka	11
2.1 Algoritma	11
2.1.1 Algoritma <i>Brute Force</i>	13
2.1.2 Algoritma <i>Backtracking</i>	16
2.2 <i>Generator</i>	23
2.3 <i>Crossword Puzzle</i>	23
2.3.1 Jenis <i>Crossword Puzzle</i>	24
2.3.2 Aturan <i>Crossword Puzzle</i>	25
2.3.3 Prosedur Permainan <i>Crossword Puzzle</i>	25
2.4 Kesimpulan	27
Bab 3. Pelaksanaan	28
3.1 Studi Literatur	28
3.2 Perancangan Program	28
3.3 Implementasi Program.....	28
3.4 Pengujian terhadap Program <i>Crossword Puzzle Generator</i>	28
Bab 4. Hasil.....	30
4.1 Hasil Studi Literatur.....	30
4.2 Rancangan Program	31
4.2.1 <i>Input</i>	31
4.2.2 Proses	31
4.2.3 <i>Output</i>	43
4.3 Program <i>Crossword Puzzle Generator</i>	43
4.4 Hasil Pengujian <i>Crossword Puzzle Generator</i>	43
4.5 Kesimpulan	44
Bab 5. Pembahasan	45
5.1 Rancangan Program.....	45
5.2 Program <i>Crossword Puzzle Generator</i>	45
Bab 6. Kesimpulan dan Saran.....	46
6.1 Kesimpulan	46
6.2 Saran	46
Daftar Pustaka dan Rujukan.....	47
Daftar Pustaka	47
Rujukan	47
Lampiran A. Data Hasil Pengujian terhadap <i>Crossword Puzzle Generator</i>	48
Lampiran B. <i>Source Code</i> dari Program	57

Daftar Tabel

Tabel 1 Daftar pertanyaan.....	26
Tabel 2 Perbandingan antara Algoritma <i>Brute Force</i> dan <i>Backtracking</i>	30
Tabel 3 Spesifikasi komputer untuk pengujian.....	44
Tabel 4 Hasil pengujian terhadap <i>Crossword Puzzle Generator</i>	48

Daftar Gambar

Gambar 1 Contoh program rekursif	11
Gambar 2 Basis pada contoh program rekursif.....	12
Gambar 3 Rekurens pada contoh program rekursif	12
Gambar 4 Contoh <i>code</i> pada program iteratif.....	12
Gambar 5 Contoh looping pada program iteratif.....	12
Gambar 6 Rumus penghitungan kombinasi	14
Gambar 7 Rumus penghitungan faktorial	14
Gambar 8 Prosedur pada algoritma <i>brute force</i> 2	15
Gambar 9 Rumus penghitungan permutasi.....	15
Gambar 10 Prosedur pada algoritma <i>brute force</i> 3	16
Gambar 11 Contoh kerangka algoritma <i>backtracking</i> [13]	18
Gambar 12 Basis dari fungsi backtracking	20
Gambar 13 Rekurens dari fungsi backtracking	20
Gambar 14 Struktur data pohon/ <i>tree</i> pada sebuah penerapan algoritma <i>backtracking</i> [3]	20
Gambar 15 Subroutine <i>is_a_solution</i> [11]	21
Gambar 16 Subroutine <i>construct_candidates</i> [11]	21
Gambar 17 Subroutine <i>process_solution</i> [11]	22
Gambar 18 Solusi dari persoalan delapan ratu [1].....	23
Gambar 19 <i>Crossword puzzle</i> tipe <i>American style</i> [4].	24
Gambar 20 <i>Crossword puzzle</i> tipe <i>British style</i> [4]	24
Gambar 21 <i>Crossword puzzle</i> tipe <i>Japanese style</i> [4]	25
Gambar 22 <i>Crossword puzzle</i> sederhana yang masih kosong [5].....	26
Gambar 23 Jawaban <i>crossword puzzle</i> yang sudah selesai [5].....	27
Gambar 24 Kata pertama dimasukkan ke kotak <i>crossword puzzle</i>	32
Gambar 25 Ruang solusi untuk kata pertama	32
Gambar 26 Kata pertama dan kata kedua dimasukkan ke kotak	33
Gambar 27 Ruang solusi untuk kata kedua.....	34
Gambar 28 Ruang solusi untuk kata ketiga	35
Gambar 29 Kata pertama, kata kedua, dan kata ketiga dimasukkan ke dalam kotak	35
Gambar 30 <i>Crossword puzzle</i> yang sudah selesai	37
Gambar 31 <i>Activity diagram</i> untuk <i>Crossword Puzzle Generator</i>	38
Gambar 32 <i>Pseudocode</i> untuk <i>Crossword Puzzle Generator</i>	40
Gambar 33 Basis dari <i>pseudocode</i> untuk <i>Crossword Puzzle Generator</i>	40
Gambar 34 Rekurens pada <i>Crossword Puzzle Generator</i>	41
Gambar 35 <i>Class diagram</i>	42

Lembar Pengesahan Tugas Akhir
Politeknik Informatika Del

Development of Crossword Puzzle Generator

Oleh:

Elga F. Silaban 11105022

Amson Ujung 11105081

Grace Isabella 11105083

Dinyatakan memenuhi syarat dan karenanya disetujui dan disahkan sebagai

Laporan Tugas Akhir Diploma 3

Program Studi Teknik Informasi

Pembimbing 1,



Dr. Saiful Akbar, S.T., MT.
NIP. 132215078

Pembimbing 2,



Gandhi M.T. Manalu, S.Si.
NIDN. 0118078201

Prakata

Puji syukur kepada Tuhan Yang Maha Esa atas rahmat dan karunia-Nya, Laporan Tugas Akhir (TA) dengan judul "*Development of Crossword Puzzle Generator*" dapat diselesaikan.

Penulisan laporan ini disusun sebagai dokumentasi laporan pelaksanaan Tugas Akhir mengenai pembangunan *Crossword Puzzle Generator* yang telah kami lakukan. Kami berharap semoga laporan ini dapat memberikan manfaat bagi pembaca terutama bagi pembaca yang ingin mengetahui dan mempelajari *Crossword Puzzle Generator*.

Kami mengucapkan terimakasih kepada Bapak Dr. Saiful Akbar, S.T, MT. dan Bapak Gandhi Maruli Tua Manalu, S.Si sebagai pembimbing, yang telah memberikan petunjuk, bimbingan, dan saran perbaikan selama pelaksanaan Tugas Akhir.

Kami juga mengucapkan terimakasih kepada koordinator Tugas Akhir yang telah memberi motivasi dan bimbingan dalam persiapan dan pelaksanaan TA. Kami juga mengucapkan terimakasih kepada Ibu Fazat N. Azizah, S.T, MSc. dan Ibu Inte Christinawati Bu'ulolo, S.T sebagai penguji yang telah memberikan kritik dan saran yang membangun bagi kesempurnaan dan kelengkapan dokumen laporan Tugas Akhir ini.

Kami juga mengucapkan terimakasih kepada direktur Politeknik Informatika Del yaitu Ibu Dr.Ir.Inggriani yang telah memberikan masukan dan bimbingan dalam penyelesaian TA, dan kepada seluruh pihak yang telah memberikan dukungan dan bantuan selama proses penyelesaian Tugas Akhir.

Kami menyadari masih banyak kekurangan yang terdapat dalam laporan ini, untuk itu kritik dan saran yang membangun dari seluruh pembaca sangat kami harapkan demi terwujudnya kesempurnaan laporan Tugas Akhir ini.

Laguboti, 02 September 2008

Elga F. Silaban 11105022

Amson Ujung 11105081

Grace Isabella 11105083

DAFTAR ISI

Prakata.....	3
Abstrak.....	4
Bab 1. Pendahuluan	8
1.1 Latar Belakang	8
1.2 Tujuan	8
1.3 Lingkup	8
1.4 Pendekatan	9
1.5 Sistematika Penyajian	10
Bab 2. Tinjauan Pustaka	11
2.1 Algoritma	11
2.1.1 Algoritma <i>Brute Force</i>	13
2.1.2 Algoritma <i>Backtracking</i>	16
2.2 <i>Generator</i>	23
2.3 <i>Crossword Puzzle</i>	23
2.3.1 Jenis <i>Crossword Puzzle</i>	24
2.3.2 Aturan <i>Crossword Puzzle</i>	25
2.3.3 Prosedur Permainan <i>Crossword Puzzle</i>	25
2.4 Kesimpulan	27
Bab 3. Pelaksanaan	28
3.1 Studi Literatur	28
3.2 Perancangan Program	28
3.3 Implementasi Program	28
3.4 Pengujian terhadap Program <i>Crossword Puzzle Generator</i>	28
Bab 4. Hasil.....	30
4.1 Hasil Studi Literatur.....	30
4.2 Rancangan Program	31
4.2.1 <i>Input</i>	31
4.2.2 Proses	31
4.2.3 <i>Output</i>	43
4.3 Program <i>Crossword Puzzle Generator</i>	43
4.4 Hasil Pengujian <i>Crossword Puzzle Generator</i>	43
4.5 Kesimpulan	44
Bab 5. Pembahasan	45
5.1 Rancangan Program	45
5.2 Program <i>Crossword Puzzle Generator</i>	45
Bab 6. Kesimpulan dan Saran	46
6.1 Kesimpulan	46
6.2 Saran	46
Daftar Pustaka dan Rujukan.....	47
Daftar Pustaka.....	47
Rujukan	47
Lampiran A. Data Hasil Pengujian terhadap <i>Crossword Puzzle Generator</i>	48
Lampiran B. <i>Source Code</i> dari Program	57

Daftar Tabel

Tabel 1 Daftar pertanyaan.....	26
Tabel 2 Perbandingan antara Algoritma <i>Brute Force</i> dan <i>Backtracking</i>	30
Tabel 3 Spesifikasi komputer untuk pengujian.....	44
Tabel 4 Hasil pengujian terhadap <i>Crossword Puzzle Generator</i>	48

Lembar Pengesahan Tugas Akhir
Politeknik Informatika Del

Development of Crossword Puzzle Generator

Oleh:

Elga F. Silaban 11105022

Amson Ujung 11105081

Grace Isabella 11105083

Dinyatakan memenuhi syarat dan karenanya disetujui dan disahkan sebagai

Laporan Tugas Akhir Diploma 3

Program Studi Teknik Informasi

Pembimbing 1,



Dr. Saiful Akbar, S.T., MT.

NIP. 132215078

Pembimbing 2,



Gandhi M.T. Manalu, S.Si.

NIDN. 0118078201

Prakata

Puji syukur kepada Tuhan Yang Maha Esa atas rahmat dan karunia-Nya, Laporan Tugas Akhir (TA) dengan judul "*Development of Crossword Puzzle Generator*" dapat diselesaikan.

Penulisan laporan ini disusun sebagai dokumentasi laporan pelaksanaan Tugas Akhir mengenai pembangunan *Crossword Puzzle Generator* yang telah kami lakukan. Kami berharap semoga laporan ini dapat memberikan manfaat bagi pembaca terutama bagi pembaca yang ingin mengetahui dan mempelajari *Crossword Puzzle Generator*.

Kami mengucapkan terimakasih kepada Bapak Dr. Saiful Akbar, S.T, MT. dan Bapak Gandhi Maruli Tua Manalu, S.Si sebagai pembimbing, yang telah memberikan petunjuk, bimbingan, dan saran perbaikan selama pelaksanaan Tugas Akhir.

Kami juga mengucapkan terimakasih kepada koordinator Tugas Akhir yang telah memberi motivasi dan bimbingan dalam persiapan dan pelaksanaan TA. Kami juga mengucapkan terimakasih kepada Ibu Fazat N. Azizah, S.T, MSc. dan Ibu Inte Christinawati Bu'ulolo, S.T sebagai penguji yang telah memberikan kritik dan saran yang membangun bagi kesempurnaan dan kelengkapan dokumen laporan Tugas Akhir ini.

Kami juga mengucapkan terimakasih kepada direktur Politeknik Informatika Del yaitu Ibu Dr.Ir.Inggriani yang telah memberikan masukan dan bimbingan dalam penyelesaian TA, dan kepada seluruh pihak yang telah memberikan dukungan dan bantuan selama proses penyelesaian Tugas Akhir.

Kami menyadari masih banyak kekurangan yang terdapat dalam laporan ini, untuk itu kritik dan saran yang membangun dari seluruh pembaca sangat kami harapkan demi terwujudnya kesempurnaan laporan Tugas Akhir ini.

Laguboti, 02 September 2008

Elga F. Silaban 11105022

Amson Ujung 11105081

Grace Isabella 11105083

Abstrak

Teka teki silang (*crossword puzzle*) merupakan permainan di mana pemain akan mengisi huruf-huruf yang merupakan jawaban dari pertanyaan ke dalam kotak-kotak yang disusun secara menurun atau mendatar, berdasarkan petunjuk yang diberikan. Pengisian juga memperhatikan kesesuaian huruf dari kata yang bersilangan dengan kata yang lain. Tujuan utama permainan ini adalah mengisi semua kotak kosong dengan kata yang merupakan jawaban dari pertanyaan. Proses pembuatan permainan ini membutuhkan waktu yang relatif lama apabila dilakukan secara manual. Oleh sebab itu, diperlukan suatu *generator* yang dapat menghasilkan *crossword puzzle*.

Pada Tugas Akhir ini, akan dihasilkan *generator* yang dapat menghasilkan *crossword puzzle*. Pendekatan yang dilakukan untuk menyelesaikan Tugas Akhir ini adalah dengan melakukan studi literatur mengenai *crossword puzzle*, algoritma *brute force* dan *backtracking*, kemudian mempelajari hasil yang diperoleh dari studi literatur, merancang program yang akan dibangun, dan mengimplementasikannya dengan menggunakan bahasa pemrograman Java, kemudian melakukan pengujian untuk mengetahui jumlah *crossword puzzle* yang *valid* yang dihasilkan oleh *Crossword Puzzle Generator*.

Crossword Puzzle Generator yang dibangun dapat menghasilkan *crossword puzzle*, tetapi belum menerapkan keseluruhan aturan untuk *crossword puzzle*. Aturan yang belum diterapkan yaitu aturan bahwa setiap kata tidak boleh saling tumpang tindih.

Hasil yang diperoleh dari penggerjaan Tugas Akhir ini masih belum sempurna karena masih terdapat *crossword puzzle* yang tidak *valid* dan program belum mampu menghasilkan semua *crossword puzzle* yang mungkin dihasilkan sesuai dengan yang diterapkan pada algoritma *backtracking*. Guna penyempurnaan dan pengembangan selanjutnya, sebaiknya dilakukan penghapusan terhadap solusi sebelumnya, sehingga tidak terdapat kata-kata yang tumpang tindih.

Kata kunci: *crossword puzzle*, *crossword puzzle generator*, *brute force*, *backtracking*.

DAFTAR ISI

Prakata.....	3
Abstrak.....	4
Bab 1. Pendahuluan	8
1.1 Latar Belakang	8
1.2 Tujuan	8
1.3 Lingkup	8
1.4 Pendekatan	9
1.5 Sistematika Penyajian	10
Bab 2. Tinjauan Pustaka	11
2.1 Algoritma	11
2.1.1 Algoritma <i>Brute Force</i>	13
2.1.2 Algoritma <i>Backtracking</i>	16
2.2 <i>Generator</i>	23
2.3 <i>Crossword Puzzle</i>	23
2.3.1 Jenis <i>Crossword Puzzle</i>	24
2.3.2 Aturan <i>Crossword Puzzle</i>	25
2.3.3 Prosedur Permainan <i>Crossword Puzzle</i>	25
2.4 Kesimpulan	27
Bab 3. Pelaksanaan	28
3.1 Studi Literatur	28
3.2 Perancangan Program	28
3.3 Implementasi Program	28
3.4 Pengujian terhadap Program <i>Crossword Puzzle Generator</i>	28
Bab 4. Hasil.....	30
4.1 Hasil Studi Literatur.....	30
4.2 Rancangan Program	31
4.2.1 <i>Input</i>	31
4.2.2 Proses	31
4.2.3 <i>Output</i>	43
4.3 Program <i>Crossword Puzzle Generator</i>	43
4.4 Hasil Pengujian <i>Crossword Puzzle Generator</i>	43
4.5 Kesimpulan	44
Bab 5. Pembahasan	45
5.1 Rancangan Program	45
5.2 Program <i>Crossword Puzzle Generator</i>	45
Bab 6. Kesimpulan dan Saran	46
6.1 Kesimpulan	46
6.2 Saran	46
Daftar Pustaka dan Rujukan.....	47
Daftar Pustaka.....	47
Rujukan	47
Lampiran A. Data Hasil Pengujian terhadap <i>Crossword Puzzle Generator</i>	48
Lampiran B. <i>Source Code</i> dari Program	57

DAFTAR ISI

Prakata.....	3
Abstrak.....	4
Bab 1. Pendahuluan	8
1.1 Latar Belakang	8
1.2 Tujuan	8
1.3 Lingkup	8
1.4 Pendekatan	9
1.5 Sistematika Penyajian	10
Bab 2. Tinjauan Pustaka	11
2.1 Algoritma	11
2.1.1 Algoritma <i>Brute Force</i>	13
2.1.2 Algoritma <i>Backtracking</i>	16
2.2 <i>Generator</i>	23
2.3 <i>Crossword Puzzle</i>	23
2.3.1 Jenis <i>Crossword Puzzle</i>	24
2.3.2 Aturan <i>Crossword Puzzle</i>	25
2.3.3 Prosedur Permainan <i>Crossword Puzzle</i>	25
2.4 Kesimpulan	27
Bab 3. Pelaksanaan	28
3.1 Studi Literatur	28
3.2 Perancangan Program	28
3.3 Implementasi Program	28
3.4 Pengujian terhadap Program <i>Crossword Puzzle Generator</i>	28
Bab 4. Hasil.....	30
4.1 Hasil Studi Literatur.....	30
4.2 Rancangan Program	31
4.2.1 <i>Input</i>	31
4.2.2 Proses	31
4.2.3 <i>Output</i>	43
4.3 Program <i>Crossword Puzzle Generator</i>	43
4.4 Hasil Pengujian <i>Crossword Puzzle Generator</i>	43
4.5 Kesimpulan	44
Bab 5. Pembahasan	45
5.1 Rancangan Program	45
5.2 Program <i>Crossword Puzzle Generator</i>	45
Bab 6. Kesimpulan dan Saran	46
6.1 Kesimpulan	46
6.2 Saran	46
Daftar Pustaka dan Rujukan.....	47
Daftar Pustaka.....	47
Rujukan	47
Lampiran A. Data Hasil Pengujian terhadap <i>Crossword Puzzle Generator</i>	48
Lampiran B. <i>Source Code</i> dari Program	57

Daftar Tabel

Tabel 1 Daftar pertanyaan.....	26
Tabel 2 Perbandingan antara Algoritma <i>Brute Force</i> dan <i>Backtracking</i>	30
Tabel 3 Spesifikasi komputer untuk pengujian.....	44
Tabel 4 Hasil pengujian terhadap <i>Crossword Puzzle Generator</i>	48

Lembar Pengesahan Tugas Akhir
Politeknik Informatika Del

Development of Crossword Puzzle Generator

Oleh:

Elga F. Silaban 11105022

Amson Ujung 11105081

Grace Isabella 11105083

Dinyatakan memenuhi syarat dan karenanya disetujui dan disahkan sebagai

Laporan Tugas Akhir Diploma 3

Program Studi Teknik Informasi

Pembimbing 1,



Dr. Saiful Akbar, S.T., MT.

NIP. 132215078

Pembimbing 2,



Gandhi M.T. Manalu, S.Si.

NIDN. 0118078201

Prakata

Puji syukur kepada Tuhan Yang Maha Esa atas rahmat dan karunia-Nya, Laporan Tugas Akhir (TA) dengan judul "*Development of Crossword Puzzle Generator*" dapat diselesaikan.

Penulisan laporan ini disusun sebagai dokumentasi laporan pelaksanaan Tugas Akhir mengenai pembangunan *Crossword Puzzle Generator* yang telah kami lakukan. Kami berharap semoga laporan ini dapat memberikan manfaat bagi pembaca terutama bagi pembaca yang ingin mengetahui dan mempelajari *Crossword Puzzle Generator*.

Kami mengucapkan terimakasih kepada Bapak Dr. Saiful Akbar, S.T, MT. dan Bapak Gandhi Maruli Tua Manalu, S.Si sebagai pembimbing, yang telah memberikan petunjuk, bimbingan, dan saran perbaikan selama pelaksanaan Tugas Akhir.

Kami juga mengucapkan terimakasih kepada koordinator Tugas Akhir yang telah memberi motivasi dan bimbingan dalam persiapan dan pelaksanaan TA. Kami juga mengucapkan terimakasih kepada Ibu Fazat N. Azizah, S.T, MSc. dan Ibu Inte Christinawati Bu'ulolo, S.T sebagai penguji yang telah memberikan kritik dan saran yang membangun bagi kesempurnaan dan kelengkapan dokumen laporan Tugas Akhir ini.

Kami juga mengucapkan terimakasih kepada direktur Politeknik Informatika Del yaitu Ibu Dr.Ir.Inggriani yang telah memberikan masukan dan bimbingan dalam penyelesaian TA, dan kepada seluruh pihak yang telah memberikan dukungan dan bantuan selama proses penyelesaian Tugas Akhir.

Kami menyadari masih banyak kekurangan yang terdapat dalam laporan ini, untuk itu kritik dan saran yang membangun dari seluruh pembaca sangat kami harapkan demi terwujudnya kesempurnaan laporan Tugas Akhir ini.

Laguboti, 02 September 2008

Elga F. Silaban 11105022

Amson Ujung 11105081

Grace Isabella 11105083

Daftar Gambar

Gambar 1 Contoh program rekursif	11
Gambar 2 Basis pada contoh program rekursif.....	12
Gambar 3 Rekurens pada contoh program rekursif	12
Gambar 4 Contoh <i>code</i> pada program iteratif.....	12
Gambar 5 Contoh looping pada program iteratif.....	12
Gambar 6 Rumus penghitungan kombinasi	14
Gambar 7 Rumus penghitungan faktorial	14
Gambar 8 Prosedur pada algoritma <i>brute force</i> 2	15
Gambar 9 Rumus penghitungan permutasi.....	15
Gambar 10 Prosedur pada algoritma <i>brute force</i> 3	16
Gambar 11 Contoh kerangka algoritma <i>backtracking</i> [13]	18
Gambar 12 Basis dari fungsi backtracking	20
Gambar 13 Rekurens dari fungsi backtracking.....	20
Gambar 14 Struktur data pohon/ <i>tree</i> pada sebuah penerapan algoritma <i>backtracking</i> [3]	20
Gambar 15 Subroutine <i>is_a_solution</i> [11].....	21
Gambar 16 Subroutine <i>construct_candidates</i> [11]	21
Gambar 17 Subroutine <i>process_solution</i> [11]	22
Gambar 18 Solusi dari persoalan delapan ratu [1].....	23
Gambar 19 <i>Crossword puzzle</i> tipe <i>American style</i> [4].....	24
Gambar 20 <i>Crossword puzzle</i> tipe <i>British style</i> [4]	24
Gambar 21 <i>Crossword puzzle</i> tipe <i>Japanese style</i> [4]	25
Gambar 22 <i>Crossword puzzle</i> sederhana yang masih kosong [5].....	26
Gambar 23 Jawaban <i>crossword puzzle</i> yang sudah selesai [5].....	27
Gambar 24 Kata pertama dimasukkan ke kotak <i>crossword puzzle</i>	32
Gambar 25 Ruang solusi untuk kata pertama	32
Gambar 26 Kata pertama dan kata kedua dimasukkan ke kotak	33
Gambar 27 Ruang solusi untuk kata kedua.....	34
Gambar 28 Ruang solusi untuk kata ketiga	35
Gambar 29 Kata pertama, kata kedua, dan kata ketiga dimasukkan ke dalam kotak	35
Gambar 30 <i>Crossword puzzle</i> yang sudah selesai	37
Gambar 31 <i>Activity diagram</i> untuk <i>Crossword Puzzle Generator</i>	38
Gambar 32 <i>Pseudocode</i> untuk <i>Crossword Puzzle Generator</i>	40
Gambar 33 Basis dari <i>pseudocode</i> untuk <i>Crossword Puzzle Generator</i>	40
Gambar 34 Rekurens pada <i>Crossword Puzzle Generator</i>	41
Gambar 35 <i>Class diagram</i>	42

Bab 1. Pendahuluan

Dalam bab ini disampaikan uraian mengenai latar belakang yang menjadi dasar pemikiran dan pertimbangan dalam memilih topik, tujuan, lingkup dan pendekatan dari Tugas Akhir yang dilakukan, serta sistematika penyajian materi Tugas Akhir.

1.1 Latar Belakang

Crossword Puzzle atau yang dalam Bahasa Indonesia disebut juga dengan istilah Teka Teki Silang, merupakan permainan yang sering dimunculkan di media cetak. Permainan ini menarik karena jawaban dari setiap pertanyaan diisi ke dalam kotak yang panjangnya harus sesuai dengan jumlah huruf dari jawaban dan terdapat persilangan antara satu kata dengan kata yang lain. Selain menarik, permainan ini juga menantang karena mengharuskan pemain berpikir dan memiliki wawasan yang luas untuk menyelesaiannya.

Proses pembuatan permainan ini membutuhkan waktu yang relatif lama apabila dilakukan secara manual, mulai dari penyusunan pertanyaan dan jawaban, penyesuaian panjang kotak yang tersedia dengan panjang kata yang merupakan jawaban dari pertanyaan, sampai penentuan letak persilangan dari masing masing kata. Oleh sebab itu diperlukan suatu *generator* yang dapat menghasilkan *crossword puzzle*.

1.2 Tujuan

Tujuan dari penggerjaan Tugas Akhir ini adalah untuk membangun *generator* yang dapat menghasilkan *crossword puzzle*.

1.3 Lingkup

Pada Tugas Akhir ini dibahas mengenai *generator* yang dapat digunakan untuk menghasilkan *crossword puzzle* dengan menggunakan bahasa pemrograman Java. Yang dimaksud dengan *generator* dalam Tugas Akhir ini adalah rancangan program untuk menghasilkan *crossword puzzle* (dalam bentuk *pseudocode*) dan implementasi dari rancangan program yang dinamakan dengan program *Crossword Puzzle Generator*. *Crossword Puzzle* yang dihasilkan merupakan *crossword puzzle* jenis *British style*. Algoritma yang dikaji untuk menghasilkan *generator* ini adalah algoritma *brute force* dan

backtracking. Rancangan program yang dihasilkan bukan merupakan algoritma baru, tetapi hasil adaptasi dari algoritma yang sudah ada (*backtracking*).

1.4 Pendekatan

Untuk mencapai tujuan dari pengerojaan Tugas Akhir ini, dilakukan pendekatan sebagai berikut:

1. Studi literatur mengenai *crossword puzzle*, algoritma *brute force* dan *backtracking*. Studi literatur dilakukan dengan cara melakukan pencarian informasi di buku, *ebook*, dan *paper*.
2. Membangun *generator*. Proses dalam membangun *generator* ini dilakukan dengan:
 - a. Mempelajari *crossword puzzle*.
 - b. Mempelajari penggunaan algoritma *brute force* dan *backtracking* dalam penyelesaian masalah.
 - c. Merancang suatu program (dalam bentuk *pseudocode*) yang dapat digunakan untuk menghasilkan *crossword puzzle*.
 - d. Mengimplementasikan rancangan program tersebut dengan menggunakan bahasa pemrograman Java.
3. Melakukan pengujian terhadap program *Crossword Puzzle Generator*.

1.5 Sistematika Penyajian

Secara garis besar dokumen ini disusun dalam lima bagian termasuk bab pendahuluan yang di dalamnya diuraikan latar belakang, tujuan, lingkup, serta pendekatan yang dilakukan dalam menyelesaikan Tugas Akhir. Bab-bab berikutnya disampaikan dengan sistematika sebagai berikut:

Pada bab dua dijelaskan hasil tinjauan pustaka yang dihimpun dari pustaka yang relevan dengan Tugas Akhir yang dilakukan.

Pada bab tiga dijelaskan kegiatan yang dilaksanakan untuk mewujudkan tujuan-tujuan dari penggerjaan Tugas Akhir.

Pada bab empat dijelaskan hasil-hasil kegiatan yang diperoleh.

Pada bab lima disampaikan pembahasan terhadap hasil yang diperoleh.

Pada bab enam disampaikan kesimpulan dan saran yang diperoleh selama mengerjakan Tugas Akhir.

Bab 2. Tinjauan Pustaka

Dalam bab ini disampaikan uraian yang dihimpun dari sumber-sumber pustaka mengenai algoritma, algoritma *brute force* dan *backtracking*, *crossword puzzle*, dan kesimpulan tinjauan pustaka. Algoritma *brute force* dan *backtracking* merupakan contoh algoritma pencarian.

2.1 Algoritma

Algoritma merupakan prosedur yang digunakan untuk menyelesaikan suatu permasalahan tertentu dan membutuhkan masukan, kemudian mengubahnya menjadi *output* yang diinginkan [10]. Urutan langkah-langkah dalam algoritma dapat dinyatakan dalam bentuk *pseudocode* [12]. Algoritma dapat diimplementasikan secara rekursif atau iteratif. Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri dengan argumen yang cakupannya lebih sempit, atau dengan argumen lain yang berbeda dengan yang sebelumnya. Rekursif tidak akan berguna kalau argumennya tetap sama seperti sebelumnya. Rekursif akan selalu dilakukan sampai basisnya dicapai atau sampai sebuah *exit condition* terpenuhi [2]. Rekursif terdiri dari 2 komponen yaitu [8]:

1. Basis

Basis merupakan suatu keadaan yang dapat menyebabkan fungsi berhenti.

2. Rekurens

Rekurens merupakan bagian di mana terdapat pemanggilan terhadap fungsi tersebut dengan parameter yang nilainya mengecil menuju basis.

Contoh:

```
public int fac(int n) {  
    if (n == 0) {  
        return 1;  
    } else {  
        return n * fac(n-1);  
    }  
}
```

Gambar 1 Contoh program rekursif

Basis pada contoh di atas yaitu:

```
if (n == 0) {  
    return 1;  
}
```

Gambar 2 Basis pada contoh program rekursif

Rekurens pada contoh di atas yaitu:

```
else {  
    return n * fac(n-1);  
}
```

Gambar 3 Rekurens pada contoh program rekursif

Iteratif adalah fungsi yang diimplementasikan dengan *looping* (pengulangan). Iteratif membutuhkan variabel penampung sementara (*temporary variable*) yang digunakan sebagai *remainder* untuk mengontrol *looping* apakah berhenti atau dilanjutkan. Contoh *code* yang iteratif adalah sebagai berikut:

```
public int factorial(int n) {  
    int result = 1;  
    for (int i = 1; i <= n; i++) {  
        result = result*i;  
    }  
  
    return result;  
}
```

Gambar 4 Contoh *code* pada program iteratif

Pada contoh *code* di bawah yang dimaksud dengan variabel penampung sementara (*temporary variable*) adalah **i**, dan *looping* yang dimaksud adalah:

```
for (int i = 1; i <= n; i++) {  
    result = result*i;  
}
```

Gambar 5 Contoh looping pada program iteratif

2.1.1 Algoritma ***Brute Force***

Brute force adalah salah satu jenis algoritma yang digunakan untuk mencari semua solusi yang mungkin dari sebuah masalah. Algoritma *brute force* merupakan algoritma paling dasar yang digunakan untuk menyelesaikan masalah. Algoritma ini melakukan pencarian secara menyeluruh (*exhaustive search*) untuk mendapatkan semua solusi yang mungkin, tanpa memperhitungkan besar *resource* (memori komputer) yang digunakan untuk melakukan pencarian tersebut [2].

Pencarian secara menyeluruh dilakukan oleh *brute force*, dengan cara maju terus dan mencoba semua kemungkinan untuk mendapatkan solusi dari sebuah masalah. Algoritma ini hampir selalu menjadi cara yang pertama kali dipertimbangkan dalam menyelesaikan suatu masalah [2]. Algoritma *brute force* sering diimplementasikan secara rekursif.

Contoh masalah yang dapat diselesaikan dengan menggunakan algoritma *brute force* [9]:

1. *N-Queen Problem*, merupakan masalah dimana di dalam papan catur N x N akan diletakkan N buah ratu, dengan syarat pada saat peletakan tidak boleh ada posisi ratu yang mengancam (menempati) posisi ratu lainnya. Langkah ratu yang *valid* adalah *horizontal*, *vertical* dan *diagonal*. *N-Queen Problem* dapat diselesaikan dengan menggunakan algoritma *brute force* karena *brute force* akan mencoba seluruh kemungkinan peletakan ratu sehingga akan ditemukan solusi yang benar.
2. *Travelling Salesman Problem*, merupakan masalah dimana diberikan N buah kota serta diketahui jarak antara satu kota dengan yang lain. Temukan rute terpendek yang melalui setiap kota lainnya hanya sekali dan kembali lagi ke kota asal keberangkatan. *Travelling Salesman Problem* dapat diselesaikan dengan menggunakan algoritma *brute force* karena *brute force* akan mencoba semua jalan satu persatu dan menghitung jarak tempuhnya, sehingga ditemukan kombinasi untuk mendapatkan waktu yang paling singkat untuk mengunjungi semua kota.

Untuk lebih memahami proses dari algoritma *brute force*, berikut ini disajikan contoh penggunaannya dalam *N-Queen Problem*. Contoh solusi dari *N-Queen Problem* dengan delapan buah ratu adalah sebagai berikut [9]:

1. Brute Force 1

- a. Mencoba semua kemungkinan solusi penempatan delapan buah ratu pada petak-petak papan catur.
- b. Ada $C(64, 8) = 4.426.165.368$ kemungkinan solusi. Cara menghitung kombinasi solusi adalah sesuai rumus pada gambar 1.

$$C(x,y) = \frac{x!}{y!(x-y)!}$$

C = kombinasi

x = jumlah seluruh anggota (jumlah petak papan catur).

y = jumlah yang diambil (jumlah ratu).

Gambar 6 Rumus penghitungan kombinasi

Cara menghitung faktorial adalah sesuai rumus pada gambar 2.

$$n! = n(n-1)(n-2)\dots3.2.1$$

Gambar 7 Rumus penghitungan faktorial

2 . Brute Force 2

- a. Meletakkan masing-masing ratu hanya pada baris-baris yang berbeda. Untuk setiap baris, kita coba tempatkan ratu mulai dari kolom 1, 2, ..., 8.
- b. Jumlah kemungkinan solusi yang diperiksa berkurang menjadi $8^n = 16777216$ dimana $n = 8$.

Gambar 8 Prosedur pada algoritma *brute force* 2

3. Brute Force 3 (exhaustive search)

- a. Misalkan solusinya dinyatakan dalam vektor 8-tuple: $X = (x_1, x_2, \dots, x_8)$
 - b. Vektor solusi merupakan permutasi dari bilangan 1 sampai 8.
 - c. Jumlah permutasi bilangan 1 sampai 8 adalah $P(8, 1) = 8! = 40320$ buah. Cara menghitung permutasi adalah sesuai rumus pada gambar 3.

$$P(x,y) = \frac{x!}{(x-y)!}$$

P = permutasi

x = jumlah seluruh anggota (jumlah petak papan catur).

y = jumlah yang diambil (jumlah ratu).

Gambar 9 Rumus penghitungan permutasi

```

procedure Ratu2
{Mencari semua solusi penempatan delapan ratu pada petak-petak papan catur yang berukuran 8 x 8 }
Deklarasi
    X : vektor_solusi
    n, i : integer

Algoritma:
    n←40320 { Jumlah permutasi (1, 2, ..., 8) }
    i←1
    repeat
        X←Permutasi(8) { Bangkitan permutasi (1, 2, ..., 8) }

        { periksa apakah X merupakan solusi }
        if Solusi(X) then
            TulisSolusi(X)
        endif

        i←i+1 { ulangi untuk permutasi berikutnya }
    until i > n

```

Gambar 10 Prosedur pada algoritma *brute force* 3

2.1.2 Algoritma *Backtracking*

Algoritma *backtracking* adalah suatu cara yang sistematis untuk mencoba susunan yang mungkin dari sebuah ruang. Bentuk yang mungkin dari penyelesaiannya adalah hasil penyusunan (permutasi) dari beberapa objek, atau segala cara yang mungkin untuk membuat kumpulan (*subset*) dari objek-objek tersebut [11].

Algoritma *backtracking* adalah hasil penyempurnaan dari algoritma *brute force*. *Backtracking* tidak lagi melakukan *exhaustive search* seperti pada *brute force*, melainkan hanya mencoba susunan yang mungkin menjadi solusi yang benar, dan tidak mencari secara keseluruhan. Cara yang dipakai untuk mengimplementasikannya tetap sama yaitu secara rekursif.

Algoritma *backtracking* mencoba tiap kemungkinan sampai mendapatkan solusi yang benar. Selama pencarian, jika alternatif solusi tidak memberikan hasil yang benar, pencarian melakukan *backtrack*, kemudian mencoba alternatif selanjutnya. Jika alternatif yang dijalankan tidak memberikan hasil, pencarian akan kembali ke titik pilihan sebelumnya dan mencoba alternatif berikutnya yang ada di sana. Jika tidak ada titik pilihan lagi, maka pencarian gagal.

Contoh masalah yang dapat diselesaikan dengan menggunakan algoritma *backtracking* [9]:

1. *N-Queen Problem* (persoalannya sudah dijelaskan pada sub bab 2.1.1). *N-Queen Problem* dapat diselesaikan dengan menggunakan algoritma *backtracking* karena *backtracking* akan mencoba meletakkan ratu satu persatu ke dalam kotak, dan memeriksa apakah peletakan itu tepat (tidak mengancam ratu yang lain). Perbedaannya dengan *brute force*, algoritma ini akan mencoba meletakkan ratu satu persatu ke kotak dan jika langkah berikutnya tidak ditemukan akan berhenti, kembali ke langkah sebelumnya, dan mencoba langkah lain yang mungkin.
2. Mencari jalan keluar dari sebuah labirin/*maze*, merupakan suatu masalah dimana terdapat banyak jalur di dalam sebuah labirin dan cara untuk menyelesaikan masalahnya adalah dengan menemukan jalur yang benar untuk keluar dari labirin tersebut. Mencari jalan keluar dari sebuah labirin/*maze* dapat diselesaikan dengan menggunakan algoritma *backtracking* karena *backtracking* akan mencoba semua langkah yang mungkin untuk menemukan jalan keluar dari labirin. Sewaktu ditemukan jalan buntu maka akan dilakukan *backtrack* untuk kembali ke jalur sebelumnya dan mencoba jalur yang lain.

Berikut adalah contoh kerangka dari algoritma *backtracking* [13] dengan menggunakan bahasa C++:

```

bool finished = FALSE;           /* found all solutions yet? */

backtrack(int a[], int k, data input)
{
    int c[MAXCANDIDATES];        /* candidates for next position */
    int ncandidates;             /* next position candidate count */
    int i;                       /* counter */

    if (is_a_solution(a,k,input))
        process_solution(a,k,input);
    else {
        k = k+1;
        construct_candidates(a,k,input,c,&ncandidates);
        for (i=0; i<ncandidates; i++) {
            a[k] = c[i];
            backtrack(a,k,input);
            if (finished) return; /* terminate early */
        }
    }
}

```

Gambar 11 Contoh kerangka algoritma *backtracking* [13]

Keterangan mengenai peubah:

1. **bool finished:** Peubah yang bertipe *boolean* (benar atau salah) yang digunakan untuk mengetahui apakah semua solusi sudah ditemukan.
2. **int c[MAXCANDIDATES]:** Peubah yang bertipe *array of integer* (susunan angka) yang merupakan kumpulan dari semua solusi yang mungkin.
3. **int nCandidates:** Peubah yang bertipe *integer* (angka) untuk menyatakan jumlah dari kandidat solusi untuk posisi selanjutnya.
4. **int i:** peubah yang berguna sebagai *counter* dari *looping* yang dilakukan.

Fungsi *backtrack* mempunyai tiga parameter yaitu [13]:

1. *Integer* yang merupakan index dari *array*. *Array* tersebut merupakan kumpulan dari objek yang disusun untuk mendapatkan solusi lengkap (objek penyusun solusi).
2. *Integer* yang merepresentasikan urutan yang digunakan pada saat menyusun objek untuk membentuk solusi lengkap dan menyusun objek pada *array a*. Misalnya terdapat sepuluh objek yang akan disusun menjadi sebuah solusi, ketika proses terjadi dan penyusunan dilakukan, perlu diketahui sudah sampai pada urutan keberapakah sekarang.
3. Parameter untuk data tambahan yang mungkin akan digunakan.

Fungsi *backtrack* tersebut digunakan untuk mencari *subset* dengan masukan 1,2,3.

Output-nya yaitu [13]:

```
{ 1 2 3 }  
{ 1 2 }  
{ 1 3 }  
{ 1 }  
{ 2 3 }  
{ 2 }  
{ 3 }  
{ }
```

Fungsi *backtrack* melakukan pengecekan pada parameter *integer* yang merepresentasikan urutan yang digunakan pada saat menyusun objek untuk membentuk solusi lengkap dan menyusun objek pada *array a*, yang telah di-*input* untuk mencari tahu apakah sebuah solusi yang lengkap telah terpenuhi. Jika telah terpenuhi, maka akan dilakukan sebuah proses (menulis atau menampilkan pesan) yang akan memberi tanda bahwa sebuah solusi telah terbentuk. Jika belum terpenuhi, fungsi akan menentukan urutan (**k**) untuk objek penyusun solusi berikutnya. Untuk menentukan objek apa saja yang mungkin bisa digunakan sebagai objek penyusun solusi (*construct_candidates*), akan dicari objek penyusun yang cocok untuk urutan (**k**) yang sekarang. Ada kemungkinan akan ditemukan lebih dari satu objek penyusun, menggunakan satu persatu dari objek tersebut, dan mengeceknya. Demikian seterusnya dilakukan sampai semua kombinasi solusi yang lengkap diperoleh.

Sewaktu mencari objek-objek apa saja yang mungkin dapat digunakan sebagai objek penyusun solusi pada suatu urutan(k) tertentu, ada kemungkinan jika tidak ada lagi solusi yang diperoleh maka pencarian akan dilakukan pada urutan $k-1$ dan mencoba mencari kembali. Demikian seterusnya dilakukan sampai ditemukan objek penyusun solusi yang cocok. Tapi jika sampai $k=0$ tidak ditemukan juga objek penyusun yang cocok, maka fungsi akan berhenti melakukan pencarian.

Basis dari fungsi *backtrack* tersebut yaitu:

```
if (is_a_solution(a,k,input))
    Process_solution(a,k,input);
```

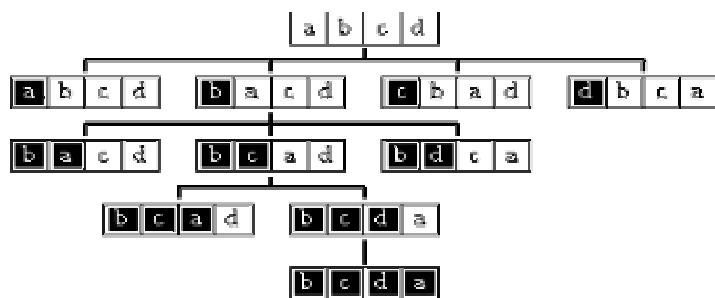
Gambar 12 Basis dari fungsi backtracking

Rekurens dari fungsi *backtrack* tersebut yaitu:

```
else {
    k = k+1;
    construct_candidates(a,k,input,c,&ncandidates);
    for (i=0; i<ncandidates; i++) {
        a[k] = c[i];
        backtrack(a,k,input);
        if (finished) return;
    }
}
```

Gambar 13 Rekurens dari fungsi backtracking

Berikut adalah contoh struktur data pohon/ *tree* pada sebuah penerapan algoritma *backtracking*:



Gambar 14 Struktur data pohon/ *tree* pada sebuah penerapan algoritma *backtracking* [3]

Algoritma *backtracking* mempunyai 3 buah *subroutine* [11]:

1. ***is_a_solution (a, k, input)*** adalah fungsi *boolean* yang berguna untuk menguji apakah elemen pertama yang ada sebanyak **k** dari *vector a* adalah solusi dari permasalahan yang diberikan. Argumen **input** berguna agar kita bisa memasukkan informasi tambahan ke dalam *routine*. Contohnya kita menggunakan argumen **input** untuk menetapkan jumlah dari solusi target.

```
is_a_solution(int a[], int k, int n)
{
    return (k == n);           /* is k == n? */
}
```

Gambar 15 Subroutine *is_a_solution* [11]

2. ***construct_candidates(a, k, input, c, ncandidates)*** adalah fungsi yang mengisi *array c* dengan himpunan lengkap dari kandidat yang mungkin untuk posisi ke-**k** dari **a**. Dan argumen **input** berguna agar kita bisa memasukkan informasi tambahan ke dalam *routine*.

```
construct_candidates(int a[], int k, int n, int c[], int *ncandidates)
{
    int i;                      /* counter */
    bool in_perm[NMAX];          /* who is in the permutation? */

    for (i=1; i<NMAX; i++) in_perm[i] = FALSE;
    for (i=0; i<k; i++) in_perm[ a[i] ] = TRUE;

    *ncandidates = 0;
    for (i=1; i<=n; i++)
        if (in_perm[i] == FALSE) {
            c[ *ncandidates ] = i;
            *ncandidates = *ncandidates + 1;
        }
}
```

Gambar 16 Subroutine *construct_candidates* [11]

3. `process_solution(a, k)` adalah fungsi yang mencetak, menghitung, atau melakukan proses lain ketika fungsi `is_a_solution` bernilai benar. Spesifikasi utama prosedur ini yaitu mencetak solusi ketika fungsi `is_a_solution` bernilai benar. *Initial state* dari *subroutine* ini yaitu belum mencetak solusi apapun. *Final state* dari *subroutine* ini yaitu mencetak solusi.

```

process_solution(int a[], int k)
{
    int i;                                /* counter */

    printf("{");
    for (i=1; i<=k; i++)
        if (a[i] == TRUE) printf(" %d",i);

    printf(" }\n");
}

```

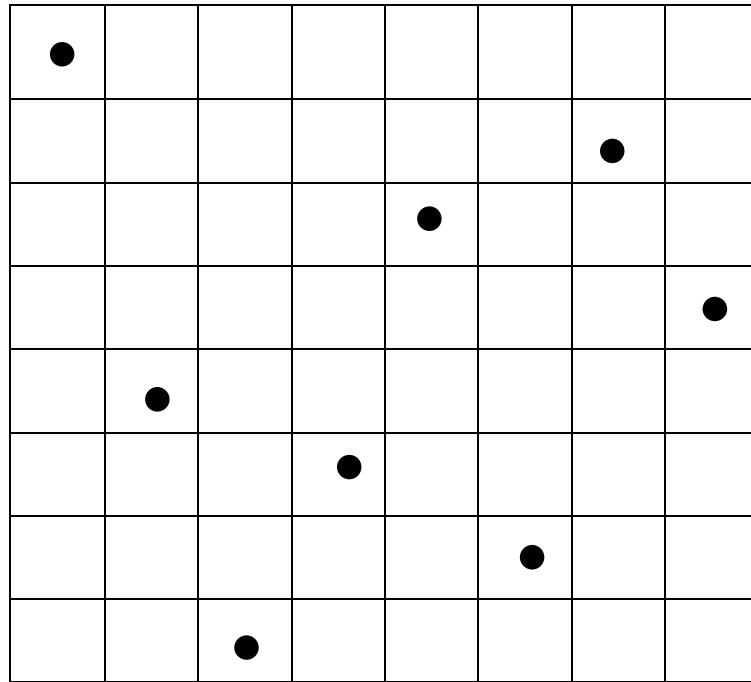
Gambar 17 Subroutine *process_solution* [11]

Untuk lebih memahami proses dari algoritma *backtracking*, berikut ini disajikan contoh penggunaannya dalam *N-Queen Problem*. Contoh solusi dari *N-Queen Problem* dengan delapan buah ratu adalah sebagai berikut [1]:

1. Mencari solusi dari penempatan delapan buah ratu pada petak-petak papan catur.
Langkah yang dilakukan yaitu:
 - a. Terlebih dahulu menempatkan satu ratu pada satu kolom.
 - b. Bila menemukan jalan yang tidak dapat dilalui, *backtrack*-lah ke kolom sebelumnya.
2. Ada $C(64, 8) = 4.426.165.368$ solusi dalam menempatkan delapan buah ratu pada enam puluh empat petak papan catur. Cara menghitung kombinasi solusi adalah sesuai rumus pada gambar 1.
3. Tidak ada ratu yang dapat berada pada baris atau kolom yang berisi ratu yang lain. Oleh sebab itu solusi penempatan ratu yang berada pada baris dan kolom yang sama dihapuskan.

4. Jumlah dari solusi yang mungkin dari penempatan delapan buah ratu pada petak-petak papan catur sekarang adalah $8! = 40,320$. Hal ini dihitung dengan rumus seperti pada gambar 2.

Salah satu solusi dari persoalan delapan ratu tersebut, tampak seperti pada gambar 18.



Gambar 18 Solusi dari persoalan delapan ratu [1]

2.2 Generator

Generator merupakan program yang memungkinkan seseorang untuk lebih mudah menghasilkan sesuatu dengan sedikit usaha dan pengetahuan. Dengan suatu program *generator*, pengguna hanya perlu menspesifikasikan langkah-langkah atau *rules* yang dibutuhkan program [7].

Salah satu contoh *generator* yaitu *Adventure Maker*. *Adventure Maker* merupakan *toolkit* yang dapat membuat permainan dan *software* multimedia secara otomatis dengan menggunakan beberapa masukan dari pengguna. Pengguna dapat membangun permainan tanpa perlu memprogram terlebih dahulu [6].

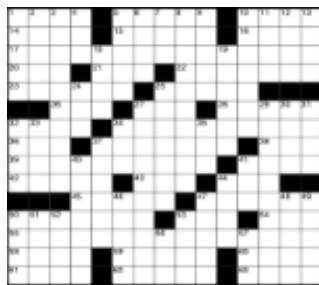
2.3 Crossword Puzzle

Pada sub bab ini dijelaskan mengenai jenis, aturan, dan prosedur dalam *crossword puzzle*.

2.3.1 Jenis Crossword Puzzle

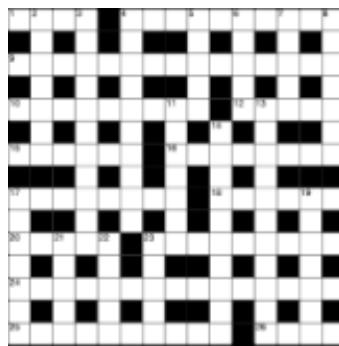
Crossword Puzzle merupakan permainan di mana pemain akan mengisi huruf-huruf yang merupakan jawaban dari pertanyaan ke dalam kotak-kotak yang disusun secara menurun atau mendatar, berdasarkan petunjuk yang diberikan. *Crossword puzzle* terdiri dari beberapa jenis sebagai berikut [4]:

1. *American style*, merupakan *crossword puzzle* yang mengijinkan kata-kata untuk saling tumpang tindih dan setiap kata harus memiliki setidaknya sebuah perpotongan dengan kata yang lain. Gambar dari contoh *crossword puzzle* tipe *American style* tampak seperti gambar 19.



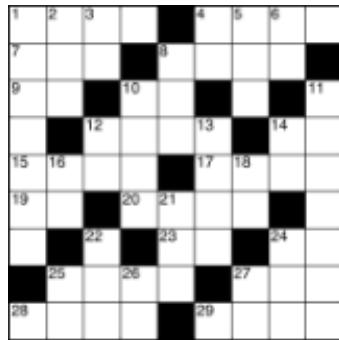
Gambar 19 *Crossword puzzle* tipe *American style* [4]

2. *British style*, merupakan *crossword puzzle* yang tidak mengijinkan kata-kata untuk saling tumpang tindih dan setiap kata harus memiliki setidaknya sebuah perpotongan dengan kata yang lain. Gambar dari contoh *crossword puzzle* tipe *British style* tampak seperti gambar 20.



Gambar 20 *Crossword puzzle* tipe *British style* [4]

3. *Japanese style*, merupakan *crossword puzzle* yang mengijinkan kata-kata untuk saling tumpang tindih, setiap kata harus memiliki setidaknya sebuah perpotongan dengan kata yang lain, dan sisi dari satu kotak hitam tidak boleh berhadapan dengan sisi dari kotak hitam lainnya. Gambar dari contoh *crossword puzzle* tipe *Japanese style* tampak seperti gambar 21.



Gambar 21 *Crossword puzzle* tipe *Japanese style* [4]

Crossword puzzle yang dibangun yaitu *crossword puzzle* jenis British *style*. Oleh sebab itu, *crossword puzzle* yang dibahas pada sub bab berikutnya adalah *crossword puzzle* jenis British *style*.

2.3.2 Aturan *Crossword Puzzle*

Crossword puzzle ada yang *valid* dan yang tidak *valid*. *Crossword puzzle* yang *valid* adalah *crossword puzzle* yang memenuhi aturan untuk *crossword puzzle* jenis British *style*, yaitu:

1. Setiap kata harus memiliki setidaknya sebuah perpotongan dengan kata yang lain.
2. Kata yang berpotongan diisi dengan huruf yang sama.
3. Tidak ada kata yang saling tumpang tindih.
4. Panjang kata tidak boleh melebihi panjang atau lebar kotak untuk *crossword puzzle*.

Crossword puzzle yang tidak *valid* adalah *crossword puzzle* yang tidak mengikuti keseluruhan aturan untuk *crossword puzzle*.

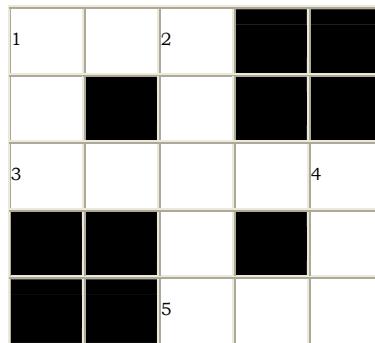
2.3.3 Prosedur Permainan *Crossword Puzzle*

Pada awal permainan tersedia sebuah *game board* berbentuk segi empat dengan dimensi tertentu atau bentuk lain yang tersusun dari kotak-kotak kecil. Jumlah kotak kecil yang terdapat pada *game board* segi empat lebih banyak daripada yang terdapat pada *game board* bentuk lainnya, oleh sebab itu *game board* berbentuk segi empat lebih sering digunakan. Dimensi merepresentasikan panjang dan lebar *board* yang dihitung berdasarkan jumlah kotak kecil yang terdapat pada baris dan kolom *board*. Di antara kotak kecil ini ada kotak yang dapat diisi oleh pemain, yaitu kotak kosong yang biasanya

direpresentasikan dengan kotak berwarna putih. Kotak putih yang harus diisi oleh pemain terdiri dari dua posisi, yaitu mendatar dan menurun. Kata yang sejajar dipisahkan paling sedikit oleh satu kotak dan setiap kata tidak boleh saling tumpang tindih. Pada kotak-kotak tertentu terjadi persilangan antara dua kata, di mana tempat yang bersilangan harus diisi dengan huruf yang sama. Pertanyaan dibagi dalam kategori mendatar dan menurun (tergantung posisi kotak-kotak yang harus diisi). Pemain harus menyesuaikan panjang kata dengan kotaknya dan memperhatikan huruf yang merupakan persilangan dari dua kata [9].

Pada umumnya dilakukan penomoran untuk jalur kotak menurun, mendatar dan pertanyaannya. Tujuan dari penomoran tersebut adalah agar pemain dapat mengetahui ke barisan kotak yang mana jawaban dari pertanyaan-pertanyaan akan dimasukkan, sehingga pemain dapat mengisi dan menyelesaikan permainan dengan benar.

Gambar 22 merupakan contoh dari *crossword puzzle* sederhana yang masih kosong [5].



Gambar 22 Crossword puzzle sederhana yang masih kosong [5]

Daftar pertanyaan untuk *crossword puzzle* tersebut dapat dilihat pada tabel 2 sebagai berikut:

Tabel 1 Daftar pertanyaan

Pertanyaan mendatar	Pertanyaan menurun
1. cukai	1. Alat transportasi umum.
3. tidak benar.	2. Memperbolehkan(bhs. Inggris).
5. Wanita Tuna Susila (singkat).	4. Jenis kertas.

Jawaban untuk *crossword puzzle* tersebut ditampilkan pada gambar 23.

¹ B	E	² A	.	.
U	.	L	.	.
³ S	A	L	A	⁴ H
.	.	O	.	V
.	.	⁵ W	T	S

Gambar 23 Jawaban *crossword puzzle* yang sudah selesai [5]

2.4 Kesimpulan

1. Algoritma *brute force* memecahkan masalah dengan sangat sederhana, karena algoritma ini mencari semua kemungkinan solusi yang dapat dibentuk, dan setelah semua solusi terbentuk, algoritma ini akan mengecek kelayakan dari setiap solusi yang dibentuk.
2. Dengan menggunakan algoritma *backtracking* kita tidak perlu memeriksa semua kemungkinan solusi, tetapi hanya perlu mengembangkan satu objek dan melakukan *backtrack* jika terjadi kesalahan. Hal ini dapat memberikan keuntungan, yaitu efisiensi waktu dalam penggerjaan.
3. Tujuan utama *crossword puzzle* adalah mengisi semua kotak kosong (berwarna putih) dengan kata yang merupakan jawaban sebuah pertanyaan. Pemain harus menyesuaikan panjang kata dengan kotaknya dan memperhatikan huruf yang merupakan persilangan dari dua kata.
4. *Crossword puzzle* yang *valid* merupakan *crossword puzzle* yang setiap katanya harus memiliki setidaknya sebuah perpotongan dengan kata yang lain, kata yang berpotongan diisi dengan huruf yang sama, tidak ada kata yang saling tumpang tindih, dan panjang kata tidak boleh melebihi panjang atau lebar kotak untuk *crossword puzzle*.

Bab 3. Pelaksanaan

Dalam bab ini disampaikan uraian mengenai pelaksanaan yang dilakukan untuk mencapai tujuan dari penggerjaan Tugas Akhir.

3.1 Studi Literatur

Proses yang dilakukan pertama kali untuk mencapai tujuan dari penggerjaan Tugas Akhir ini adalah mengumpulkan informasi mengenai *crossword puzzle*, algoritma *brute force* dan algoritma *backtracking*. Pengumpulan informasi dilakukan dengan melakukan pencarian informasi di buku, *ebook*, dan *paper*.

3.2 Perancangan Program

Informasi mengenai *crossword puzzle*, algoritma *brute force*, dan *backtracking* yang telah dikumpulkan kemudian dipelajari. Setelah algoritma tersebut dipelajari, cara kerja dari algoritma *backtracking* akan digunakan untuk menghasilkan rancangan program yang dapat digunakan untuk menghasilkan *crossword puzzle*, karena algoritma *backtracking* dianggap lebih baik daripada algoritma *brute force*. Rancangan program ini mengikuti aturan yang berlaku dalam *crossword puzzle*. Rancangan program yang dihasilkan adalah dalam bentuk *pseudocode*.

Perancangan diawali dengan membuat *activity diagram*, kemudian dilanjutkan dengan pembuatan struktur data untuk *generator*, yang pengimplementasianya dalam bentuk *class*.

3.3 Implementasi Program

Rancangan program yang dihasilkan akan diimplementasikan dalam bentuk program *Crossword Puzzle Generator*. Implementasi *generator* ini menggunakan bahasa pemrograman Java.

3.4 Pengujian terhadap Program *Crossword Puzzle Generator*

Pengujian dilakukan dengan tujuan untuk mengetahui berapa jumlah *crossword puzzle* yang *valid* yang dihasilkan oleh program. Pengujian dilakukan dengan memasukkan dimensi dari kotak dan jawaban dari pertanyaan pada *Crossword Puzzle Generator*.

Pengujian dilakukan beberapa kali dengan dimensi kotak yang berbeda, tetapi jawaban yang sama. Pengamatan hasil dilakukan dengan cara menghitung jumlah *crossword puzzle* yang *valid*, yang dihasilkan oleh *Crossword Puzzle Generator*. Dari hasil pengujian akan diketahui jumlah *crossword puzzle* yang dihasilkan oleh *Crossword Puzzle Generator*. Apabila dari hasil pengujian terdapat *crossword puzzle* yang tidak *valid*, berarti masih terdapat kekurangan pada program yang dibangun.

Bab 4. Hasil

Dalam bab ini disampaikan uraian mengenai hasil yang diperoleh dari pelaksanaan Tugas Akhir, yaitu berupa hasil studi literatur, rancangan program, program *Crossword Puzzle Generator*, hasil pengujian *Crossword Puzzle Generator*, disertai kesimpulan terhadap hasil yang telah diperoleh.

4.1 Hasil Studi Literatur

Dari studi literatur yang telah dilakukan, dapat disimpulkan perbandingan antara algoritma *brute force* dan *backtracking* adalah sebagai berikut:

Tabel 2 Perbandingan antara Algoritma *Brute Force* dan *Backtracking*

No	Algoritma <i>Brute Force</i>	Algoritma <i>Backtracking</i>
1	Memecahkan masalah dengan mencari semua kemungkinan solusi yang dapat dibentuk, dan setelah semua solusi terbentuk, algoritma ini akan mengecek kelayakan dari setiap solusi yang dibentuk.	Memecahkan masalah dengan mencoba tiap kemungkinan sampai mendapatkan solusi yang benar. Selama pencarian, jika alternatif solusi tidak memberikan hasil yang benar, pencarian melakukan <i>backtrack</i> , kemudian mencoba alternatif selanjutnya. Jika alternatif yang dijalankan tidak memberikan hasil, pencarian akan kembali ke titik pilihan sebelumnya dan mencoba alternatif berikutnya yang ada di sana.
2	Pemecahan masalah dapat dilakukan dengan lebih sederhana bila dibandingkan dengan menggunakan algoritma <i>backtracking</i> .	Pemecahan masalah lebih rumit bila dibandingkan dengan menggunakan algoritma <i>brute force</i> .
3	Memiliki jumlah <i>candidate</i> solusi yang lebih besar, bila dibandingkan dengan yang dimiliki oleh algoritma <i>brute force</i> .	Memiliki jumlah <i>candidate</i> solusi yang lebih kecil, bila dibandingkan dengan yang dimiliki oleh algoritma <i>backtracking</i> .
4	Membutuhkan waktu yang lebih lama di dalam memecahkan masalah, bila dibandingkan dengan menggunakan algoritma <i>backtracking</i> .	Membutuhkan waktu yang lebih singkat di dalam memecahkan masalah, bila dibandingkan dengan menggunakan algoritma <i>brute force</i> .

4.2 Rancangan Program

Rancangan program yang dihasilkan merupakan hasil adaptasi dari algoritma *backtracking*. Pada sub bab ini akan dibahas *input*, *output* dan proses yang terdapat pada program.

4.2.1 Input

Input yang digunakan untuk menyusun *crossword puzzle* ini adalah dimensi *board crossword puzzle* ($m*n$), dengan nilai m dapat sama atau berbeda dengan n , dan daftar kata-kata yang akan dimasukkan ke kotak.

4.2.2 Proses

Tahapan untuk menghasilkan *crossword puzzle* adalah sebagai berikut:

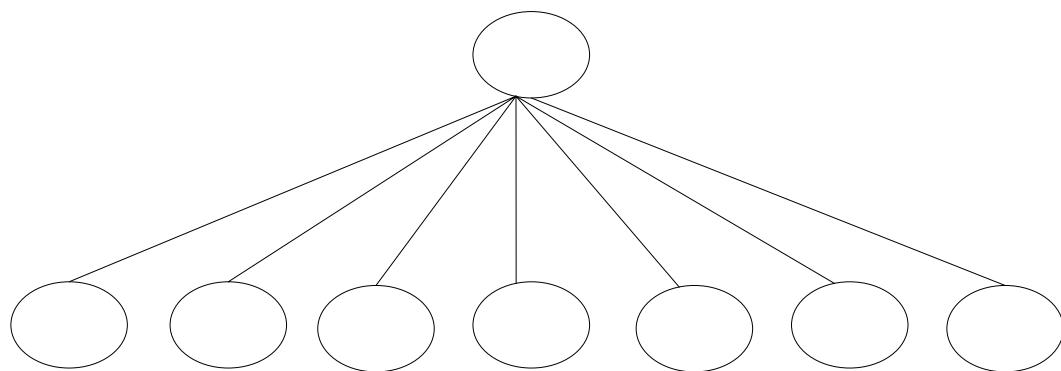
1. Menentukan dimensi kotak. Untuk penjelasan berikut ini dimensi yang digunakan adalah $10*10$.
2. Mendaftarkan kata-kata yang akan dimasukkan ke kotak. Contoh kumpulan kata yang akan dimasukkan ke dalam kotak *crossword puzzle* adalah sebagai berikut.

a. Komputer	f. Seperti
b. Orasi	g. Susi
c. Kado	h. Tahu
d. Tikar	i. Urap
e. Nasi	j. Rumah
3. Kata yang pertama kali ditemukan dimasukkan ke kotak secara *horizontal* atau *vertical*, dengan syarat jumlah kotak yang tersedia mencukupi jumlah huruf dari kata tersebut. Contohnya tampak seperti gambar 24.

K	O	M	P	U	T	E	R		

Gambar 24 Kata pertama dimasukkan ke kotak *crossword puzzle*

4. Mencari kata-kata yang dapat berpotongan dengan kata pertama. Kata-kata tersebut menjadi ruang solusi untuk kata pertama. Contohnya tampak seperti gambar 25.



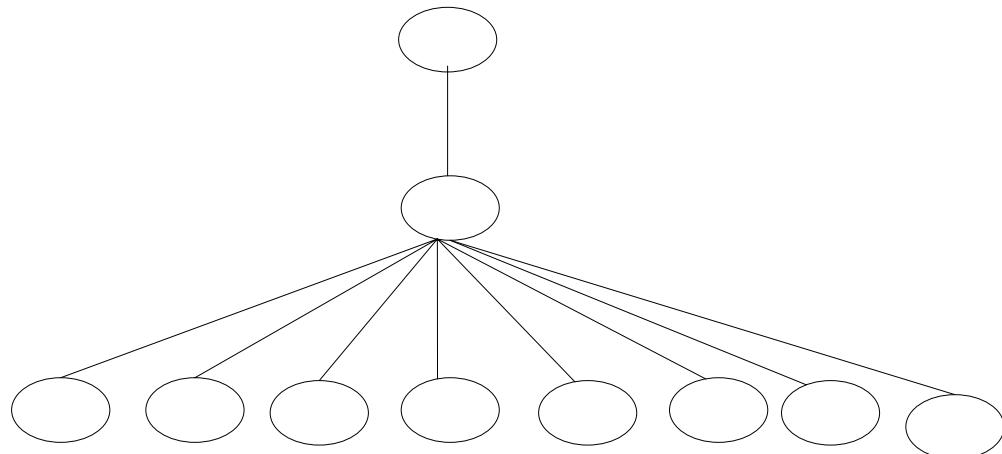
Gambar 25 Ruang solusi untuk kata pertama

5. Kata yang pertama kali ditemukan yang dapat bersilangan dengan kata pertama (disebut kata kedua) dijadikan *candidate* solusi, kemudian dimasukkan ke dalam kotak, dan tepat berpotongan dengan kata pertama, dengan syarat jumlah kotak yang tersedia mencukupi jumlah huruf dari kata tersebut. Bila jumlah kotak tidak mencukupi, maka kata kedua tersebut dihapus dari ruang solusi. Kemudian kita mencoba kata kedua berikutnya yang dapat bersilangan dengan kata pertama (kata kedua tersebut dijadikan sebagai *candidate* solusi), dan begitu seterusnya hingga kata kedua tersebut dapat dimasukkan ke dalam kotak. Contohnya tampak seperti gambar 26.

K	O	M	P	U	T	E	R		
R									
A									
S									
I									

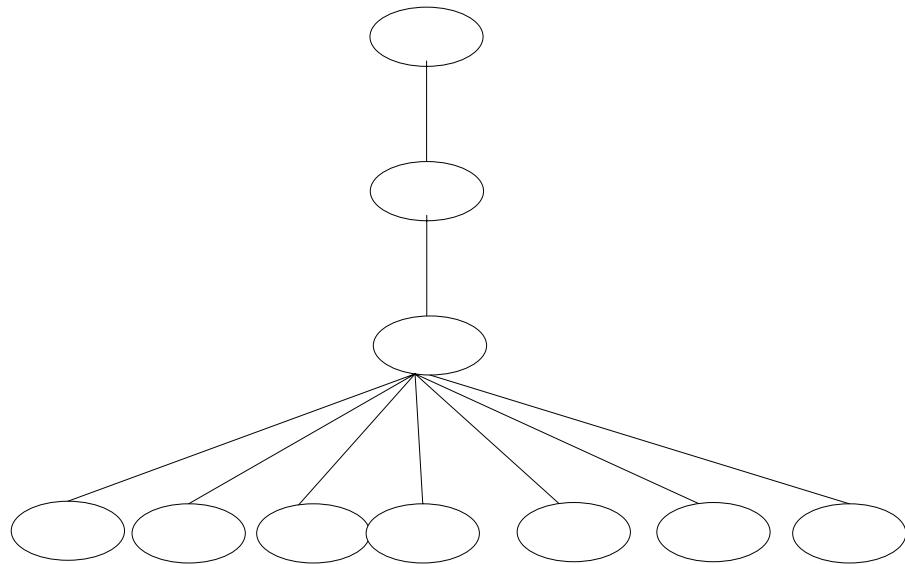
Gambar 26 Kata pertama dan kata kedua dimasukkan ke kotak

6. Mencari kata-kata yang dapat memotong kata kedua atau kata pertama, dan belum dimasukkan ke kotak. Kata-kata tersebut menjadi ruang solusi untuk kata kedua (contohnya tampak seperti gambar 27).



Gambar 27 Ruang solusi untuk kata kedua

7. Kata yang pertama kali ditemukan yang dapat bersilangan dengan kata kedua atau dengan kata pertama (disebut kata ketiga) dijadikan *candidate* solusi, kemudian dimasukkan ke dalam kotak, dan tepat berpotongan dengan kata kedua atau dengan kata pertama, dengan syarat jumlah kotak yang tersedia masih mencukupi untuk kata tersebut. Jika jumlah kotak yang tersedia tidak mencukupi, maka kata ketiga tersebut dihapus dari ruang solusi. Kemudian kita mencoba kata ketiga berikutnya yang dapat bersilangan dengan kata kedua atau dengan kata pertama, dan jumlah kotak yang tersedia mencukupi untuk kata tersebut. Demikian seterusnya sampai ditemukan kata ketiga yang dapat bersilangan dengan kata kedua atau dengan kata pertama, dan jumlah kotak yang tersedia mencukupi untuk penempatan dari kata ketiga tersebut. Apabila semua solusi untuk kata kedua tidak ada yang dapat dimasukkan ke kotak, maka kata kedua tersebut dihapus dari ruang solusi, dan diganti dengan kata berikutnya yang dapat memotong kata pertama, kemudian dimasukkan ke kotak. Apabila solusi untuk kata pertama tersebut (disebut kata kedua) dapat dimasukkan ke kotak, kita mencari kembali kata-kata yang dapat memotong kata kedua atau kata pertama (disebut kata ketiga). Contohnya tampak seperti gambar 28.



Gambar 28 Ruang solusi untuk kata ketiga

8. Kata ketiga yang dapat menyilang kata kedua dimasukkan ke dalam kotak, tepat berpotongan dengan kata kedua atau dengan kata pertama, dengan syarat jumlah kotak yang tersedia masih mencukupi untuk kata tersebut (contohnya tampak seperti gambar 29). Jika tidak, maka kita mencoba solusi untuk kata kedua berikutnya yang dapat dimasukkan ke dalam kotak.

K	O	M	P	U	T	E	R		
R									
K	A	D	O						
S									
I									

Gambar 29 Kata pertama, kata kedua, dan kata ketiga dimasukkan ke dalam kotak

9. Demikian seterusnya dilakukan seperti pada langkah nomor empat sampai delapan, hingga seluruh kata dapat dimasukkan ke dalam kotak.
10. Jika seluruh kata telah dimasukkan ke kotak, maka hasilnya dicetak, dan dilanjutkan dengan mencoba solusi berikutnya, hingga diperoleh *crossword puzzle* lainnya. Untuk memperoleh *crossword puzzle* lainnya, dilakukan percobaan terhadap setiap ruang solusi yang tersisa dari pencarian solusi pada *crossword puzzle* sebelumnya, hingga semua ruang solusi yang tersisa selesai dicoba. Percobaan terhadap ruang solusi ini dilakukan mulai dari ruang solusi terakhir yang dihasilkan untuk mencari kata terakhir yang memotong kata sebelumnya, hingga ruang solusi dari kata yang pertama kali dimasukkan ke kotak.

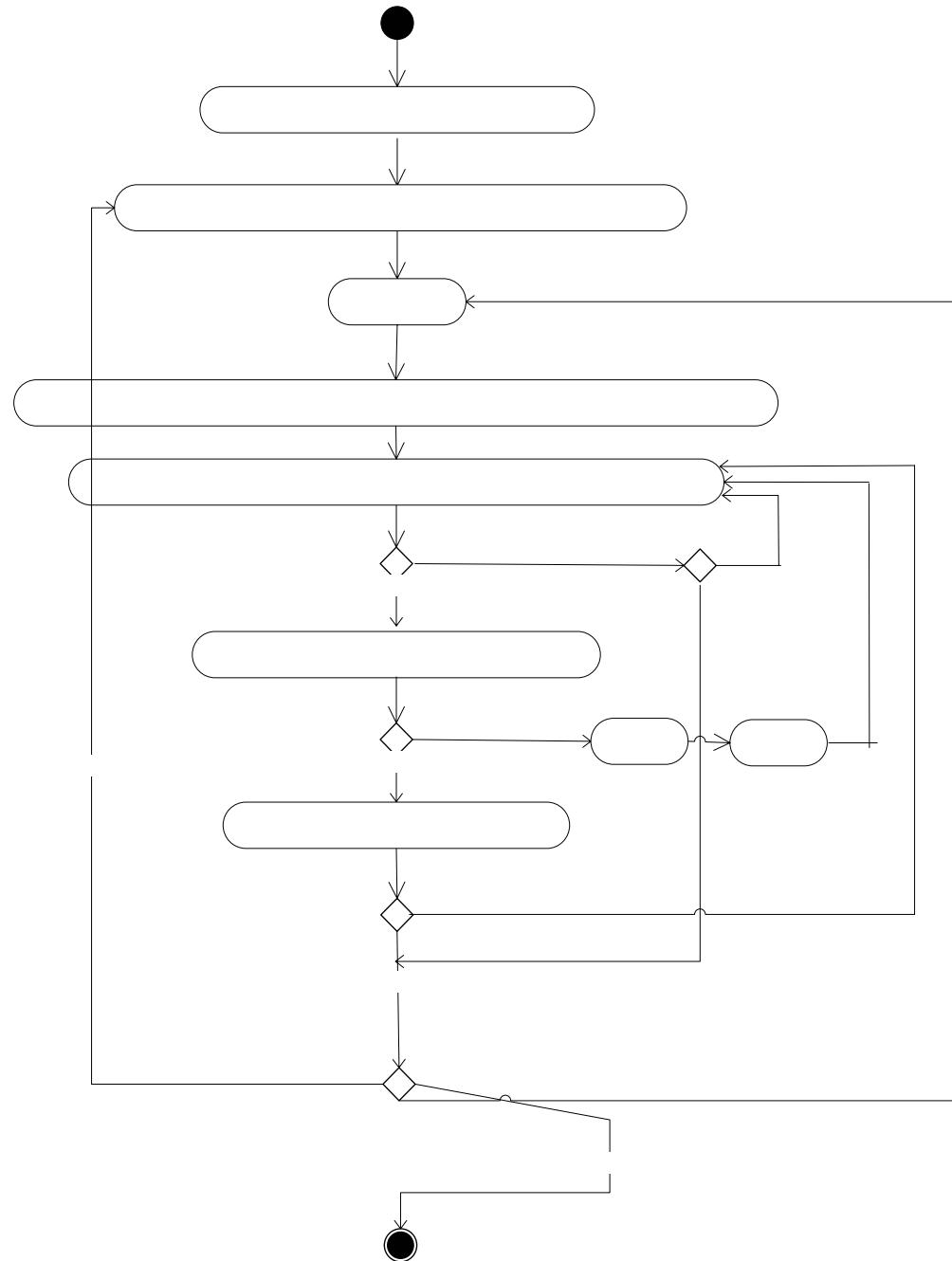
Pada contoh ini, percobaan dilakukan mulai dari ruang solusi untuk kata kesembilan yang belum dimasukkan ke kotak. Apabila ruang solusi dari kata kesembilan telah selesai dicoba, maka dilanjutkan dengan melakukan percobaan terhadap semua ruang solusi dari kata kedelapan yang belum dimasukkan ke kotak. Demikian selanjutnya sampai semua ruang solusi untuk kata kesembilan hingga kata pertama yang belum dimasukkan ke kotak selesai dicoba. Apabila semua ruang solusi dari kata pertama tersebut telah selesai dicoba, maka kata pertama yang menempati posisi sebagai kata yang pertama kali dimasukkan ke kotak, diganti posisinya menjadi *horizontal* atau *vertical* (apabila pada pencarian solusi sebelumnya kata pertama diletakkan pada posisi *horizontal*, maka posisinya diganti menjadi *vertical*), dan dilakukan kembali langkah-langkah dari nomor empat sampai sepuluh. Apabila pencarian solusi dari kata pertama yang pertama kali dimasukkan ke kotak telah selesai, maka kata pertama tersebut diganti dengan kata berikutnya yang terdapat pada kumpulan kata yang akan dimasukkan ke kotak, kemudian dilakukan kembali langkah-langkah seperti pada nomor tiga sampai sepuluh. Demikian dilakukan seterusnya hingga semua kata menempati posisi pertama pada kotak *crossword puzzle*.

Apabila *Crossword Puzzle Generator* berhasil meng-generate *crossword puzzle*, akan diperoleh minimal satu *crossword puzzle* dari semua percobaan yang dilakukan. Salah satu contoh *crossword puzzle* yang sudah selesai tampak seperti gambar 30.

K	O	M	P	U	T	E	R		
					I				
		R			K	A	D	O	
		U			A			R	
		M			R			A	
		A						S	
T	A	H	U		N	A	S	I	
			R				U		
			A				S		
	S	E	P	E	R	T	I		

Gambar 30 *Crossword puzzle* yang sudah selesai

Activity diagram dari program *Crossword Puzzle Generator* tersebut ditampilkan pada gambar 31. *Activity diagram* tersebut digunakan sebagai rancangan yang menunjukkan alur program dalam menghasilkan *crossword puzzle*.



ambil s

Gambar 31 *Activity diagram untuk Crossword Puzzle Generator*

kumpulkan semua kata

Setelah perancangan program dilakukan, diperoleh hasil perancangan dalam bentuk *pseudocode*. *Pseudocode* yang dihasilkan menggunakan tipe data buatan sebagai berikut:

1. *Word* (tipe data untuk merepresentasikan kata yang dimasukkan oleh pengguna, memiliki panjang dan nilai (*value*)). Bentuk fisik dari struktur data *Word* adalah String.
2. *Position* (tipe data yang digunakan untuk merepresentasikan posisi dari *Word* yang akan diletakkan ke kotak (*box*), apakah posisinya *horizontal* atau *vertical*). *Position* hanya memiliki 2 nilai yaitu *horizontal* atau *vertical*. Bentuk fisik dari struktur data *Position* adalah array of String.
3. *Box* (tipe data yang digunakan untuk merepresentasikan tempat dari huruf yang akan diletakkan pada papan (*Board*)). *Box* memiliki nilai x dan y untuk menentukan tempat *Box* tersebut dalam *Board*, *used* digunakan untuk mengetahui apakah *Box* tersebut telah terisi atau tidak. Bentuk fisik dari struktur data *Box* adalah integer.
4. *Solution* adalah tipe data untuk merepresentasikan sebuah solusi. Solusi memiliki *Word* nilai dari solusi, *Position* pada posisi apa *Solution* tersebut akan diletakkan (*horizontal* atau *vertical*), dan *Box* untuk meletakkan satu persatu karakter dari *Word* (*array of box*).
5. *Board* adalah tipe data untuk merepresentasikan tempat di mana kotak-kotak akan disusun berdasarkan dimensi. *Board* memiliki panjang dan lebar (dalam satuan *box*). Bentuk fisik dari struktur data *Board* adalah array 2 dimensi dari box.

Pseudocode yang dihasilkan tampak sebagai berikut:

```

procedure backtrack(input listOfWord:vector<Word>)
{meletakkan semua solusi ke kotak
Deklarasi

Algoritma:
    if listOfWord sudah seluruhnya dimasukkan ke kotak
    then
        Cetakcrossword()
    else
        for (tiap Word yang ada didalam listOfWord){
            for (tiap Position yang mungkin) { //vertical,
                // horizontal
                    for (tiap Box yang ada dalam Board) {
                        Solution = new Solution (currentWord,
currentPosition, currentBox)
                        if (Solution dapat diletakkan) {
                            putWord ()
                            listOfWord = listOfWord - currentWord
                            backtrack (listOfWord)
                            // untuk mengembalikan seperti keadaan
                            // sebelumnya agar dapat melakukan
                            // bactrack
                            undoPut();
                        }
                    }
                }
            }
        }
    endif
    if Solution tidak ada
    then
        print "tidak ada solusi"
    endif
}

```

Gambar 32 Pseudocode untuk Crossword Puzzle Generator

Basis dari pseudocode tersebut yaitu:

```

if listOfWord sudah seluruhnya dimasukkan ke kotak
    then
        Cetakcrossword()

```

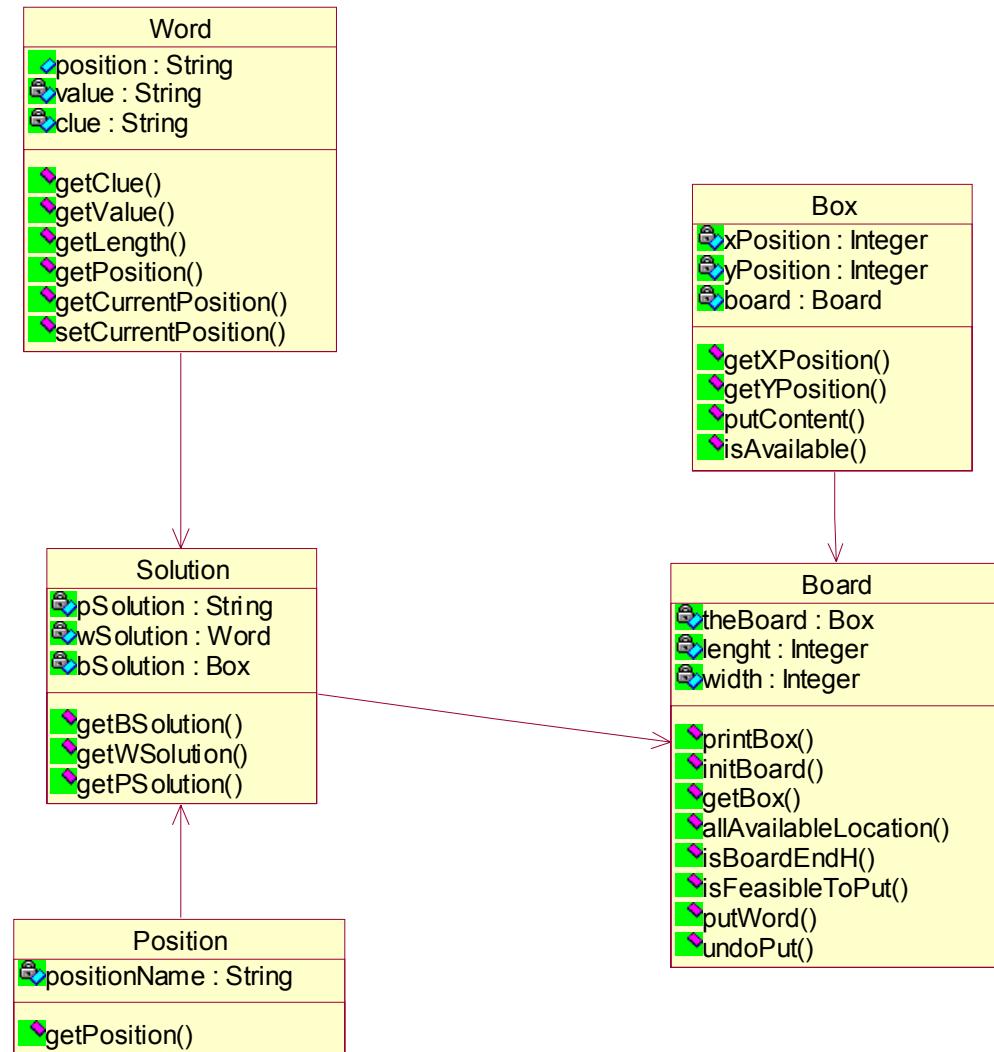
Gambar 33 Basis dari pseudocode untuk Crossword Puzzle Generator

Rekurens dari *pseudocode* tersebut yaitu:

```
else
    for (tiap Word yang ada didalam listOfWord){
        for (tiap Position yang mungkin) { //vertical,
        // horizontal
            for (tiap Box yang ada dalam Board) {
                Solution = new Solution (currentWord,
currentPosition, currentBox)
                if (Solution dapat diletakkan) {
                    putWord ()
                    listOfWord = listOfWord - currentWord
                    backtrack (listOfWord)
                    // untuk mengembalikan seperti keadaan
                    // sebelumnya agar dapat melakukan
                    // bactrack
                    undoPut ();
                }
            }
        }
    }
endif
if Solution tidak ada
then
    print "tidak ada solusi"
endif
```

Gambar 34 Rekurens pada Crossword Puzzle Generator

Class diagram yang telah dihasilkan tampak seperti pada gambar 35.



Gambar 35 *Class diagram*

Word merupakan *class* yang akan merepresentasikan kata yang dimasukkan oleh pengguna, *Position* merupakan *class* yang digunakan untuk merepresentasikan posisi dari *Word* yang akan diletakkan ke dalam *Board*, *Solution* merupakan *class* yang digunakan untuk merepresentasikan solusi yang memiliki nilai *Word* dari solusi, *Board* merupakan *class* yang digunakan untuk merepresentasikan tempat di mana kotak-kotak akan diletakkan berdasarkan dimensi kotak yang merupakan masukan dari pengguna, *Box* merupakan *class* yang merepresentasikan tempat dimana huruf akan diletakkan pada *Board*.

4.2.3 Output

Output yang akan dihasilkan adalah *crossword puzzle* yang di dalamnya telah tersusun kata-kata yang merupakan masukan dari pengguna. *Output* yang dihasilkan terdiri dari satu atau lebih *crossword puzzle* yang memiliki susunan kata-kata yang berbeda di setiap *crossword puzzle*.

4.3 Program Crossword Puzzle Generator

Dengan menggunakan *pseudocode* yang dapat dilihat pada gambar 32, dibangun program *Crossword Puzzle Generator* dengan menggunakan bahasa pemrograman Java. Program yang dibangun dapat menghasilkan *crossword puzzle* sesuai dengan besar dimensi kotak, dan jawaban dari pertanyaan. Asumsi dalam pembangunan *Crossword Puzzle Generator* ini adalah data yang dimasukkan pengguna merupakan data yang *valid*, yaitu data yang tipe dan rentang nilainya sesuai dengan tipe dan rentang nilai masukan yang dibutuhkan program, dan panjang kata tidak melebihi dimensi kotak.

4.4 Hasil Pengujian Crossword Puzzle Generator

Pengujian dilakukan dengan tujuan untuk mengetahui berapa jumlah *crossword puzzle* yang *valid* yang dapat dihasilkan oleh program. Pengujian dilakukan dengan memasukkan dimensi dari kotak dan jawaban dari pertanyaan pada *Crossword Puzzle Generator*. Apabila *Crossword Puzzle Generator* dapat menghasilkan *crossword puzzle* yang *valid* dan sesuai dengan masukan, maka program telah sesuai dengan yang diharapkan.

Dari tiga kali pengujian yang telah dilakukan dengan spesifikasi komputer yang terdapat pada tabel 1, tanpa adanya aplikasi lain yang sedang berjalan, diperoleh data mengenai

dimensi kotak dan jumlah *crossword puzzle* yang dihasilkan, hasilnya tampak seperti pada Lampiran A.

Tabel 3 Spesifikasi komputer untuk pengujian

Item	Specification/ Version
RAM	768 MB.
Processor	Intel Pentium IV 1.8 GHz.
Sistem operasi	Windows XP Professional.

Berdasarkan data hasil pengujian terhadap *Crossword Puzzle Generator* yang tampak pada Lampiran A Data hasil pengujian terhadap *Crossword Puzzle Generator*, dapat dilihat bahwa program *Crossword Puzzle Generator* yang dibangun dapat menghasilkan *crossword puzzle*, walaupun *crossword puzzle* yang dihasilkan ada yang *valid* dan yang tidak *valid*. *Crossword puzzle* yang dihasilkan tidak *valid* karena tidak memenuhi aturan untuk *crossword puzzle* yang *valid* sesuai dengan yang diuraikan pada sub bab 2.3.2.

4.5 Kesimpulan

Kesimpulan yang diperoleh dari bab hasil adalah sebagai berikut:

1. Rancangan program untuk menghasilkan *crossword puzzle* telah diselesaikan. Hasilnya ditampilkan dalam bentuk *pseudocode*.
2. Program *Crossword Puzzle Generator* yang telah dibangun dapat menghasilkan *crossword puzzle*.
3. Hasil pengujian menunjukkan bahwa program masih belum sempurna, karena selain menghasilkan *crossword puzzle* yang *valid*, juga dihasilkan *crossword puzzle* yang tidak *valid*.

Bab 5. Pembahasan

Dalam bab ini disampaikan pembahasan mengenai hasil yang diperoleh di dalam Tugas Akhir.

5.1 Rancangan Program

Pseudocode yang dihasilkan hanya mengadaptasi algoritma *backtracking* dan menggunakan rekursif untuk pengimplementasianya. Oleh sebab itu terdapat beberapa perubahan, contohnya terdapat fungsi yang digunakan untuk menghapus solusi sebelumnya.

5.2 Program *Crossword Puzzle Generator*

Program yang dihasilkan dapat menghasilkan *crossword puzzle*, tetapi tidak keseluruhan *crossword puzzle* yang dihasilkan merupakan *crossword puzzle* yang *valid*. Hal ini disebabkan karena adanya kata yang saling tumpang tindih.

Bab 6. Kesimpulan dan Saran

Dalam bab ini dijelaskan kesimpulan dan saran mengenai materi yang dibahas di dalam Tugas Akhir. Kesimpulan menyampaikan penjelasan ringkas mengenai pencapaian hasil yang diperoleh selama pengerjaan Tugas Akhir. Saran merupakan masukan untuk perbaikan dan penyempurnaan hasil pengerjaan Tugas Akhir.

6.1 Kesimpulan

Dari pelaksanaan Tugas Akhir dapat disimpulkan pokok-pokok kesimpulan berikut:

1. Rancangan program yang telah dihasilkan dapat diimplementasikan untuk menghasilkan *crossword puzzle*.
2. Program yang dihasilkan dapat menghasilkan *crossword puzzle*, tetapi seluruh aturan *crossword puzzle* yang terdapat pada bab Tinjauan Pustaka belum diterapkan seluruhnya. Aturan yang belum diterapkan yaitu aturan bahwa setiap kata tidak boleh saling tumpang tindih.
3. Proses penghapusan yang tidak sempurna dari solusi sebelumnya mengakibatkan masih tertinggal huruf yang menyebabkan *crossword puzzle* menjadi tidak *valid*.

6.2 Saran

Guna penyempurnaan dan pengembangan selanjutnya sebaiknya dilakukan penghapusan terhadap solusi sebelumnya, sehingga tidak terdapat kata-kata yang tumpang tindih dan memperbesar jumlah *crossword puzzle* yang *valid*.

Lampiran A. Data Hasil Pengujian terhadap Crossword Puzzle Generator

Tabel 4 Hasil pengujian terhadap Crossword Puzzle Generator

Dimensi Kotak	Kata	Jumlah crossword puzzle yang dihasilkan	Jumlah crossword puzzle yang valid
8 x 8	Komputer, pantai, ikan, rasa	<p>7 crossword puzzle</p> <pre> komputer ***a***** ***n*r** ***t*a** ***a*s** ***ikan* ***** */ ***** */ </pre> <pre> komputer ***a***** ***n*i** ***t*k** **rasa** ***i*n** ***** */ ***** */ </pre> <pre> komputer ***a***** *i*n**** *k*t**** rasa**** *n*i**** ***** */ ***** */ </pre> <pre> k***** o***** m***** pantai** u****k** t***rasa e****n** r***** */ </pre> <pre> k***** o***** m***** pantai** u****k** t*rasa** e****n** r***** */ </pre>	6 crossword puzzle

Dimensi Kotak	Kata	Jumlah crossword puzzle yang dihasilkan	Jumlah crossword puzzle yang valid
		k*i***** o*k***** mrasa*** pantai** u***** t***** e***** r***** k***r*** o*ikan** m***s*** pantai** u***** t***** e***** r***** 	
9 x 9	Komputer, pantai, ikan, rasa	18 crossword puzzle KOMPUTER* ***A***** ***N***** ***T***** ***A*R*** ***IKAN** *****S*** *****A*** ***** KOMPUTER* ***A***** ***N*R*** ***T*A*** ***A*S*** ***IKAN** ***** ***** ***** KOMPUTER* ***A***** ***N*I*** ***T*K*** **RASA*** ***I*N*** ***** ***** ***** 	11 crossword puzzle

Dimensi Kotak	Kata	Jumlah crossword puzzle yang dihasilkan	Jumlah crossword puzzle yang valid
		KOMPUTER* ***A***** *I N **** *K T **** RASA **** *N I **** ***** **** ***** **** ***** **** KOMPUTER* ***PANTAI ***** SK ***** AA ***** NN ***** **** ***** **** ***** **** ***** **** KOMPUTER* ***** A* ***** S* ***PANTAI ***** K ***** AA ***** NN ***** **** ***** **** KOMPUIER* ***** K A* ***** A S* ***PANTAI ***** **** ***** **** ***** **** ***** **** ***** **** KOMPUIER* ***** AP ***** SA ***** IKAN ***** T ***** A ***** I ***** **** ***** ****	

Dimensi Kotak	Kata	Jumlah crossword puzzle yang dihasilkan	Jumlah crossword puzzle yang valid
		K***** O***** M***** PANTAI*** U****K*** T***RASA* E****N*** R***** ***** K***** O***** M***** PANTAI*** U****K*** T*RASA*** E****N*** R***** ***** K*I***** O*K***** MRASA*** PANTAI*** U***** T***** E***** R***** ***** K***** O***** M**R*** PANTAI*** U***S*** T*IKAN*** E***** R***** ***** K***R*** O*IKAN*** M***S*** PANTAI*** U***** T***** E***** R***** *****	

Dimensi Kotak	Kata	Jumlah crossword puzzle yang dihasilkan	Jumlah crossword puzzle yang valid
		K***** O***** M***** PP***** UA***** TN***** ET***** RASA***** *IKAN**** K***** O***** M***** P**P***** U**A***** T**N***** E**T***** RASA***** ***IKAN** K***** O***** M***** P**P***** UIKAN*** T**N***** E**T***** RASA***** ***I***** K***** O***** M***** P**P***** U**A***** IKAN***** E**T***** RASA***** ***I***** K***** O***** M***** P***** U***** I**I***** E**K***** RASA***** *PANTAI**	
10 x 10	Komputer, pantai, ikan, rasa	18 crossword puzzle	10 crossword puzzle

Dimensi Kotak	Kata	Jumlah crossword puzzle yang dihasilkan	Jumlah crossword puzzle yang valid
		KOMPUTER** ****A***** ***N***** ***T***** ***A*R**** ***IKAN*** *****S**** *****A**** *****A***** *****A***** *****A***** KOMPUTER** ****A***** ***N*R**** ***T*A**** ***A*S**** ***IKAN*** *****A**** *****A**** *****A**** *****A**** *****A**** KOMPUTER** ****A***** ***N*I**** ***T*K**** **RASA**** ***I*N**** *****A**** *****A**** *****A**** *****A**** *****A**** KOMPUTER** ****A***** *I*N***** *K*T***** RASA***** *N*I***** *****A**** *****A**** *****A**** *****A**** KOMPUTER** ***PANTAI* *****SK* *****AA* *****N* *****A**** *****A**** *****A**** *****A**** *****A**** *****A**** 	

Dimensi Kotak	Kata	Jumlah crossword puzzle yang dihasilkan	Jumlah crossword puzzle yang valid
		KOMPUTER** ***** A** ***** S** ***PANTAI* ***** K* ***** A* ***** N* ***** ***** ***** ***** KOMPUIER** ***** K*A** ***** A*S** ***PANTAI* ***** ***** ***** ***** ***** KOMPUIER** ***** AP* ***** SA* ***** IKAN* ***** T* ***** A* ***** I* ***** ***** ***** ***** K***** O***** M***** PANTAI**** U****K**** T****RASA** E****N**** R***** ***** ***** K***** O***** M***** PANTAI**** U****K**** T*RASA**** E****N**** R***** ***** *****	

Dimensi Kotak	Kata	Jumlah crossword puzzle yang dihasilkan	Jumlah crossword puzzle yang valid
		K*I***** O*K***** MRASA***** PANTAI**** U***** T***** E***** R***** ***** ***** K***** O***** M***R**** PANTAI**** U***S**** T*IKAN**** E***** R***** ***** ***** K***R**** O*IKAN**** M***S**** PANTAI**** U***** T***** E***** R***** ***** ***** K***** O***** M***** PP***** UA***** TN***** ET***** RASA***** *IKAN**** ***** K***** O***** M***** P**P***** U**A***** T**N***** E**T***** RASA***** ***IKAN*** *****	

Dimensi Kotak	Kata	Jumlah crossword puzzle yang dihasilkan	Jumlah crossword puzzle yang valid
		K***** O***** M***** P**P***** UIKAN***** T**N***** E**T***** RASA***** ***I***** ***** K***** O***** M***** P**P***** U**A***** IKAN***** E**T***** RASA***** ***I***** ***** K***** O***** M***** P***** U***** I**I***** E**K***** RASA***** *PANTAI*** *****	

Lampiran B. Source Code dari Program

1. Code untuk melakukan backtrack

```
private void backtrack(Vector<Word> listOfWord) {
    Vector<Box> avaibleLocations = gameBoard.allAvaibleLocation();

    if (listOfWord.isEmpty()) {
        // Base case
        gameBoard.printBox();
        counter++;
        System.out.println();
    } else {
        // Recurrence case
        for (Word currentWord : listOfWord) {
            Vector<Word> listOfWord2 = new Vector<Word>(listOfWord);
            for (String currentPosition: currentWord.getPosition()) {
                for (Box currentBox : avaibleLocations) {

                    Solution solution = new Solution(currentWord,
currentPosition, currentBox);

                    if (gameBoard.isFeasibleToPut(solution)) {

                        gameBoard.putWord(solution);

                        listOfWord2.removeElement(currentWord);

                        backtrack(listOfWord2);

                        // clean the previous solution box.
                        gameBoard.undoPut();
                    }
                }
            }
        }
    }
}
```

2. Code untuk melakukan pengecekan apakah kata dapat diletakkan ke kotak atau tidak.

```
public boolean isFeasibleToPut(Solution solution) {  
    boolean state = false;  
    Word word = solution.getWSolution();  
    Box newBox;  
    currentBoxToPut.removeAllElements();  
  
    // to test all the box for the length of word  
    if (solution.getPSolution().equals("horizontal")) {  
        if (solution.getBSolution().getYPosition() +  
            word.getLength() <= length) {  
            for (int i = 0; i < word.getLength(); i++) {  
                newBox =  
                    theBoard[solution.getBSolution().getXPosition()][solution.getBSolution()  
                        .getYPosition() + i];  
  
                if (newBox.isAvailable()) {  
                    state = true;  
                    currentBoxToPut.addElement(newBox);  
                } else {  
                    state = false;  
                    break;  
                }  
            }  
        }  
    } else if (solution.getPSolution().equals("vertical")) {  
        if (solution.getBSolution().getXPosition() +  
            word.getLength() <= width) {  
            for (int i = 0; i < word.getLength(); i++) {  
                newBox =  
                    theBoard[solution.getBSolution().getXPosition() + i][solution.getBSolution()  
                        .getYPosition()];  
                if (newBox.isAvailable()) {  
                    state = true;  
                    currentBoxToPut.addElement(newBox);  
                } else {  
                    state = false;  
                    break;  
                }  
            }  
        }  
    }  
    return state;  
}
```

3. *Code* untuk melakukan *undo*.

```
public void undoPut() {
    Solution prev = prevSolution.pop();

    int length = prev.getWSolution().getLength();

    int x = prev.getBSolution().getXPosition();
    int y = prev.getBSolution().getYPosition();

    Box test;

    for (int i = 0; i < length; i++) {

        test = theBoard[x][y];
        test.deleteContent();

        if (prev.getPSolution().equals("horizontal")) {
            y++;
        } else {
            x++;
        }
    }
}
```

4. *Code* untuk meletakkan kata ke kotak.

```
public void putWord(Solution solution) {

    for (int i = 0; i < solution.getWSolution().getLength() ; i++) {
        char content = solution.getWSolution().split()[i];
        currentBoxToPut.elementAt(i).putContent(content);
    }
    prevSolution.push(solution);
}
```

Daftar Pustaka dan Rujukan

Daftar Pustaka

- [1] Bruno R. Preiss, 1998, "Data Structures and Algorithms with Object Oriented Design Patterns in Java", Department of Electrical and Computer Engineering, University of Waterloo, Canada.
- [2] George Eliot, <http://www.crosswordtournament.com/more/wynne.html>, diakses 15 Agustus 2008.
- [3] Ingriani Liem and Sri Purwanti, "Diktat Kuliah IF221 Dasar Pemrograman Pemrograman Fungsional", Institut teknologi Bandung, 2000.
- [4] Munir, Rinaldi, "Trans bahan kuliah ke2", Institut Teknologi Bandung, 2004.

Rujukan

- [1] Addison-Wesley, 2006, "Chapter6 Recursion as A Problem Solving Technique", Pearson Addison Wesley.
- [2] Ahmed Shamsul Arefin, 2006, "Art of Programming Contest", Gyankosh Prokashoni, Bangladesh.
- [3] Gurari,
<http://www.cse.ohio-state.edu/~gurari/course/cis680/cis680Ch19.html>, diakses 28 Juli 2008.
- [4] <http://en.wikipedia.org/wiki/Crossword>, diakses 27 Agustus 2008.
- [5] http://id.wikipedia.org/wiki/Teka-teki_silang/, diakses 15 Agustus 2008.
- [6] <http://www.adventuremaker.com/overview.htm>, diakses 31 Agustus 2008.
- [7] <http://www.computerhope.com/jargon/p/proggene.htm>, diakses 27 Agustus 2008.
- [8] Ingriani Liem, "Analisis Rekurens", Institut teknologi Bandung, 2007.
- [9] Munir, Rinaldi, "Strategi Algoritmik", Institut Teknologi Bandung, 2007.
- [10] Steven S. Skiena, 1997, "The Algorithm Design Manual", Department of Computer Science, New York.
- [11] Steven S. Skiena, "Backtracking",
http://www.cs.sunysb.edu/_skiena/lecture15.pdf, diakses 12 Maret 2008.
- [12] Steven S. Skiena, "Introduction to Algorithms",
http://www.cs.sunysb.edu/_skiena/lecture1.pdf, diakses 12 Maret 2008.
- [13] Steven S. Skiena & Miguel A. Revilla, 2003, "Programming Challenges", Springer, New York.