Laporan Tugas Besar II

IF 3051 Strategi Algoritma

FireFighter

Disusun oleh:

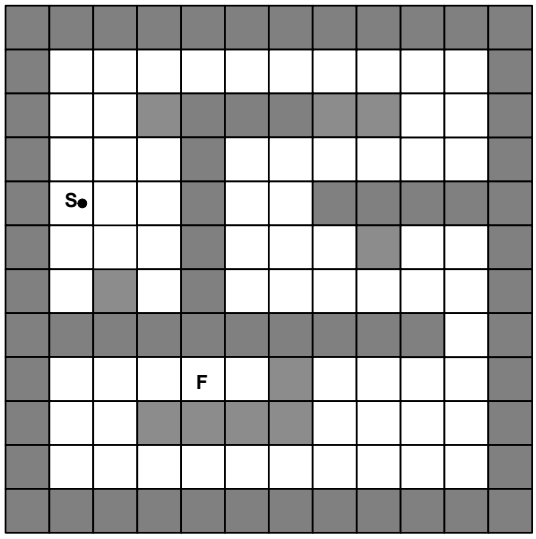
Sanrio Hernanto	/ 13507019
Shauma Hayyu	/ 13507025
Jonathan Marcel	/ 13507072

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung
Jl. Ganesha 10, Bandung 40132

Bab 1

Deskripsi Masalah

Sebuah robot diberi tugas memadamkan api. Robot mempunyai sensor yang dapat mengetahui ada sumber api di dalam sebuah ruangan. Ruangan itu berbentuk lorong-lorong yang berliku-liku laksana sebuah labirin (*maze*). Misalkan posisi robot adalah pada titik S (*start*), robot berjalan menuju sumber api di titik *F* (*fire*) seperti Gambar 1 (kotak-kotak yang diarsir menyatakan tembok, sedangkan kotak-kotak putih menyatakan jalan yang bisa dilalui. Robot hanya bisa berjalan secara vertikal dan horizontal saja (tidak bisa diagonal). Robot melewati setiap kotak (*grid*) putih setiap kali ia berjalan. Robot berhenti di titik F dan memadamkan apin di titik itu. Jika robot tidak berhasil mencapai titik F (misalnya berada di dalam ruang tertutup), maka pesan "Sumber api tidak bisa dicapai" ditampilkan.



Tugas anda adalah membantu robot menentukan lintasan dari titik S ke titik F dengan menggunakan algoritma DFS (*Backtracking*) dan solusi pembandingnya dengan algoritma BFS. Rancanglah algoritma DFS dan BFS untuk persoalan ini. Tuliskan deskripsi algoritma DFS dan BFS tersebut di dalam laporan anda (lihat struktur laporan). Setelah algoritma *Backtracking* dan BFS anda rancang, implementasikan menjadi sebuah program yang mengandung kreativitas seperti yang dijelaskan di dalam spesifikasi di bawah ini.

1. Bentuk lorong yang dilalui robot tidak harus selalu seperti Gambar 1. Program memiliki sebuah editor sederhana yang dapat digunakan oleh pengguna untuk mendefinisikan area jelajah robot (ingat editor labirin yang pernah didemokan di dalam kelas). Area yang dijelajahi robot dapat dinyatakan sebagai matriks biner (0 dan 1). Elemen bernilai 1 menyatakan *grid* yang bisa dilalui, sedangkan elemen bernilai 0 menyatakan tembok. Matriks tersebut dapat disimpan ke dalam arsip teks untuk sewaktu-waktu dibuka lagi. Jadi, editor memiliki menu *Baru* (membuat area baru), *Buka* (membuka area yang disimpan di dalam arsip) *Simpan* (menyimpan area yang sudah dibuat), dan *Tutup* arsip.
2. Robot berjalan *step by step*, yaitu satu *grid* demi satu *grid*. Program harus dapat menampilkan animasi gerakan robot tersebut, termasuk gerakan *backtraking*-nya (jika pilihan algoritmanya DFS). Jika algoritma BFS yang dipilih, maka tidak ada gerakan *backtrack*-nya. Gerakan robot dapat ditampilkan dengan simbol tertentu..
3. Titik S dan F dapat ditentukan oleh dispesifikasikan oleh pengguna dengan mengklik grid yang dimaksudkan menggunakan tetikus.
4. Jejak atau lintasan yang dilalui robot dapat dinyatakan dengan anak panah berwarna atau cara lain yang dapat dimengerti.
5. Program dapat menampilkan lintasan yang dilalui robot.
6. Jika tidak solusi, maka program menampilkan pesan “Tidak ada solusi”.
7. Kecepatan gerakan robot dapat diatur (lambat, sedang, cepat).
8. Program dilengkapi dengan menu *Bantuan* yang berisi *Perihal* (*about*) pembuat program dan Cara penggunaan program.
9. Beri nama aplikasi ini dengan nama yang unik dan menarik, misalnya *The FireFinder*.
10. Bonus: bonus nilai jika dapat mengoptimasi lintasan dengan algoritma *backtracking* (dihasilkan lintasan terpendek dengan algoritma *backtracking*).

Bab 2
Dasar Teori

Algoritma Traversal di dalam Graf

Algoritma traversal di dalam graf adalah mengunjungi simpul-simpul dengan cara yang sistematik. Terdapat dua algoritma traversal untuk graf, yaitu pencarian melebar (*Breadth First Search* atau BFS) dan pencarian mendalam (*Depth First Search* atau DFS).

Definisi Algoritma BFS

Misalkan kita mempunya graf G dengan n buah simpul. Traversal dimulai dari simpul v. Algoritma BFS ialah sebagai berikut, mula-mula kunjungi simpul v, kemudian kunjungi semua simpul yang bertetangga dengan simpul v. Selanjutnya simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul tadi dikunjungi, dan seterusnya. Jika graf berbentuk pohon berakar, maka semua simpul pada aras d dikunjungi lebih dahulu sebelum simpul-simpul pada aras d + 1.

Definisi Algoritma DFS

Misalkan kita mempunya graf G dengan n buah simpul. Traversal dimulai dari simpul v. Algoritma DFS ialah sebagai berikut, mula-mula kunjungi simpul v, kemudian kunjungi simpul w yang bertetangga dengan simpul v, lalu dilakukan pencarian mendalam lagi dimulai secara rekursif dari simpul w. Ketika mencapai simpul u sedemikian sehingga semua simpul yang bertetangga dengannya telah dikunjungi, pencarian akan di-backtracking ke simpul terakhir yang dikunjungi sebelumnya dan mempunyai simpul w yang belum dikunjungi. Pencarian mendalam dilakukan lagi dari simpul w.

Definisi Algoritma Backtracking

Algoritma backtracking adalah algoritma yang berbasis pada DFS untuk mencari solusi persoalan secara lebih mangkus. Secara sistematis algoritma ini akan mencari solusi persoalan di antara semua kemungkinan solusi yang ada. Sehingga, kita tidak perlu memeriksa semua

kemungkinan solusi yang ada, dengan kata lain hanya akan mempertimbangkan pencarian yang mengarah ke solusi saja.

Properti Umum Algoritma Backtracking

Properti berikut didefinisikan untuk menerapkan metode backtracking.

1. Solusi persoalan

Solusi dinyatakan sebagai vektor dengan n-tuple.

2. Fungsi pembangkit nilai x_k

Dinyatakan sebagai: T(k)

T(k) membangkitkan nilai untuk x_k, yang merupakan komponen vektor solusi.

3. Fungsi seleksi – dinyatakan dengan predikat SELEKSI

Dinyatakan sebagai: $B(x_1, x_2 \ldots x_k)$

Fungsi pembatas menentukan apakah $(x_1, x_2 \ldots x_k)$ mengarah ke solusi. Jika ya, maka pembangkitan nilai untuk x_{k+1} dilanjutkan, tetapi jika tidak, maka $(x_1, x_2 \ldots x_k)$ dibuang dan tidak dipertimbangkan lagi dalam pencarian solusi.

Prinsip Pencarian Solusi dengan Metode Backtracking

Langkah-langkah pencarian solusi adalah sebagai berikut.

1. Solusi dicari dengan membentuk lintasan dari akar ke daun. Aturan pembentukan yang dipakai ialah metode DFS. Simpul yang sudah dilahirkan disebut simpul hidup. Simpul hidup yang sedang diperluas disebut simpul-E. Penomoran simpul dari atas ke bawah sesuai urutan kelahirannya (prinsip DFS).
2. Setiap simpul-E diperluas, lintasan yang dibangun bertambah panjang. Lintasan yang tidak mengarah ke solusi akan dibunuh menjadi simpul mati. Fungsi untuk membunuh simpul ini menerapkan fungsi pembatas.
3. Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian diteruskan dengan membangkitkan simpul anak lainnya. Bila tidak ada lagi simpul anak yang bisa dibangkitkan, pencarian solusi dilanjutkan dengan melakukan *backtracking* ke

simpul hidup terdekat. Simpul ini akan menjadi simpul-E yang baru. Lintasan akan kembali dibangun sampai membentuk solusi.

4. Pencarian dihentikan bila kita telah menemukan solusi atau tidak ada lagi simpul hidup untuk *backtracking*.

Bab 3
Analisis Pemecahan Masalah

Algoritma Backtracking

Pada algoritma backtracking pertama-tama akan dilakukan pembangkitan simpul. Prioritas pembangkitan simpul ialah sebagai berikut: simpul bergerak ke kanan merupakan prioritas pertama, disusul dengan simpul bergerak ke kiri, kemudian simpul bergerak ke bawah, dan terakhir ialah simpul bergerak ke atas. Apabila telah memilih salah satu simpul, simpul hidup ini akan membangkitkan lintasan juga berdasarkan prioritas tersebut.

Kemudian setelah itu akan dilakukan pengecekan apabila pemadam menemui dinding maka pemadam tidak akan melewati dinding tersebut, sebaliknya pemadam akan melakukan backtrack sesuai simpul yang telah dibangkitkan. Di dalam kasus ini berarti, apabila pemadam bergerak ke kanan, maka dia akan melakukan backtrack ke kiri apabila tidak menemui solusi yang dimaksud. Hal ini juga berlaku untuk ketiga arah lainnya. Di dalam kasus ini solusi berarti pemadam menemukan api.

Setelah itu simpul akan terus dibangkitkan sesuai prinsip DFS, yaitu semakin ke dalam sampai menemukan solusi. Apabila solusi tidak ditemukan, dilakukan *backtrack* menuju simpul hidup yang terdekat. Begitupun seterusnya simpul hidup ini akan dibangkitkan lagi hingga menuju solusi. Algoritma ini akan berhenti apabila solusi telah ditemukan atau apabila simpul yang ada semuanya telah dibangkitkan dan tidak menemukan solusi.

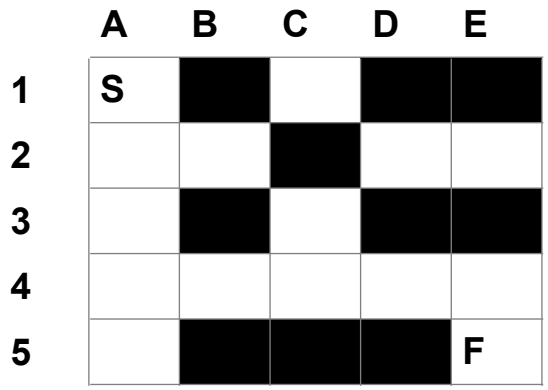
Algoritma BFS

Pada algoritma BFS pertama-tama akan dibangkitkan semua simpul kemungkinan pergerakan pemadam api. Setelah itu setiap simpul tersebut akan ditelusuri berurutan, dimulai dari gerak ke kanan, gerak ke kiri, gerak ke bawah, dan yang terakhir gerak ke atas. Prioritas pergerakan pemadam ini sama dengan yang di algoritma backtracking.

Perbedaan antara algoritma *backtracking* dengan BFS ialah, pada algoritma backtracking setelah pemadam memilih salah satu simpul, simpul tersebut (disebut simpul hidup) akan membangkitkan lintasan terus menerus hingga mencapai solusi. Jika tidak akan dilakukan backtracking. Sedangkan pada algoritma BFS, setiap simpul akan ditelusuri satu per satu untuk setiap kedalaman, hingga mencapai solusi atau tidak mencapai solusi.

Pada BFS digunakan struktur data antrian. Simpul yang pertamakali dibangkitkan akan dimasukkan ke dalam antrian. Setelah itu elemen pertama pada antrian akan diambil lagi dan semua simpul tetangganya yang mungkin dimasukkan ke dalam antrian. Hal ini dilakukan terus menerus hingga simpul yang diambil dari antrian adalah solusi atau antriannya kosong, yang berarti tidak ada solusi ditemukan.

Untuk lebih jelasnya mengenai perbedaan antara BFS dan backtracking dapat dilihat pada kasus sebagai berikut:



Lintasan yang akan dilalui pemadam pada algoritma backtracking ialah:

A1, A2, B2, A2(backtrack), A3, A4, B4, C4, D4, E4, E5

Sedangkan lintasan yang akan dilalui pemadam pada algoritma BFS ialah:

A1, A2, B2, A3, A4, B4, A5, C4, D4, C3, E4, E5

Bab 4
Implementasi dan pengujian

Spesifikasi Teknis Program

Struktur Data

Senarai (array)

Pada program ini digunakan sebuah senarai, yaitu senarai field[ ] bertipe integer digunakan untuk menyimpan bentuk dari maze.

Antrian(queue)

Pada program ini digunakan sebuah antrian, yaitu antrian q yang menyimpan data bertipe point yang digunakan pada algoritma BFS.

List

Terdapat beberapa list yang digunakan dalam program ini, antara lain:

1. List generik left, menyimpan sekuens gambar untuk gerakan karakter ke kiri.
2. List generik right, menyimpan sekuens gambar untuk gerakan karakter ke kanan.
3. List generik up, menyimpan sekuens gambar untuk gerakan karakter ke atas.
4. List generik down, menyimpan sekuens gambar untuk gerakan karakter ke bawah.
5. List generik area, menyimpan label yang membentuk gambar dinding dan jalan.

Fungsi dan Prosedur

Terdapat beberapa fungsi maupun prosedur yang digunakan di dalam program ini antara lain:

```
public void newfield(int x, int y)
//prosedur untuk membuat map baru

public void fileread(String file)
//prosedur untuk membaca file eksternal map

public void filesave(Stream myStream)
//prosedur untuk men-save map yang dibuat menjadi file eksternal

public void editsf(Label X)
//prosedur yang mengatur edit start dan finish

public bool BFS(Point v)
//algoritma BFS

public bool SolveMaze(Point P)
//algoritma DFS dengan backtracking

public void newfield(int x, int y)
//prosedur untuk membuat field baru

private void targetimage()
//prosedur untuk membuat start dan finish kembali ke posisi awal

public void initlabel()
//prosedur untuk membuat inisialisasi label
```

Antarmuka

Pembuatan antarmuka program ini menggunakan kakas Microsoft Visual C# 2008 Express Edition. Bentuk program ini merupakan windows form application.

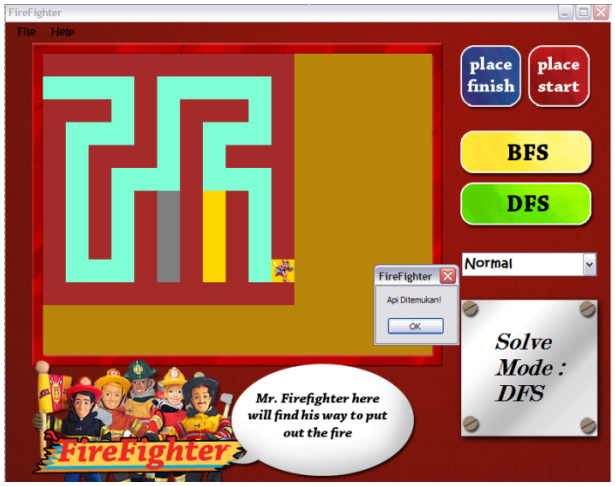
Screenshot Program

Tampilan ini merupakan tampilan pertama kali ketika program dijalankan. User bisa memilih antara start, load, maupun exit.

Apabila memilih new, maka user dapat menemui form seperti itu. Silahkan masukkan ukuran map yang diinginkan. Setelah itu user dapat membentuk maze yang diinginkan pada map tersebut.

Gambar disamping menunjukkan map yang telah selesai dibuat. User dapat meletakkan tempat start dan tempat finish dengan mengklik button yang berada di kanan atas. Untuk lebih jelasnya terdapat panel pada kanan bawah yang menunjukkan mode yang sedang terjadi.

Untuk memulai simulasi silahkan mengklik tombol BFS ataupun DFS.



Gambar disamping menunjukkan simulasi telah berhasil dijalankan dengan DFS. Perhatikan bahwa kecepatan pemadam bergerak dapat ditentukan dengan memilih 5 macam speed (dalam gambar ini speednya ialah normal) yang terdapat pada list box di bagian kanan.

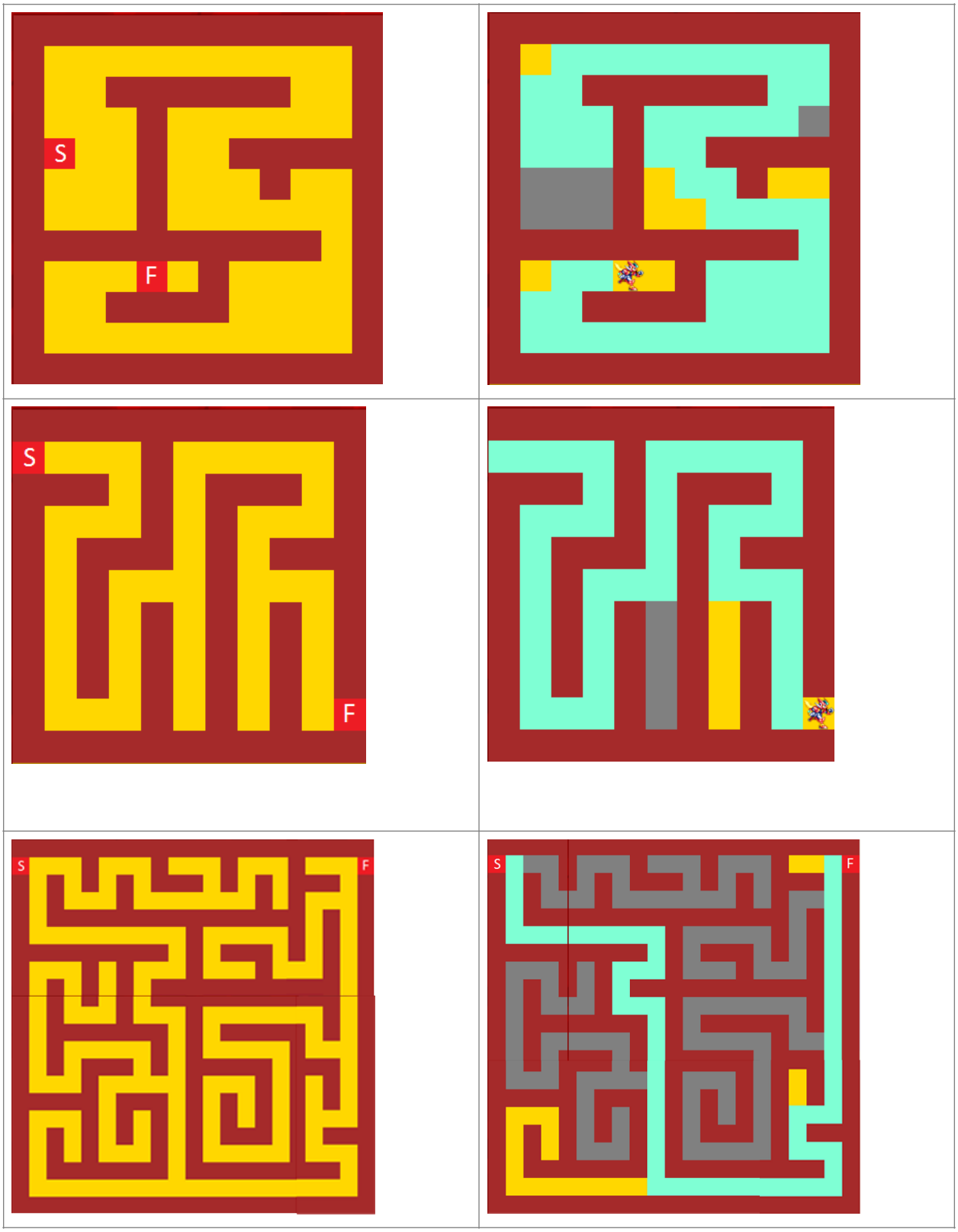
Selain itu terdapat menustrip, yang memiliki dua pilihan yaitu file dan help. Pada menu file user dapat memilih untuk new, open, maupun save map. Sedangkan pada menu help user dapat melihat help program dan aboutbox.

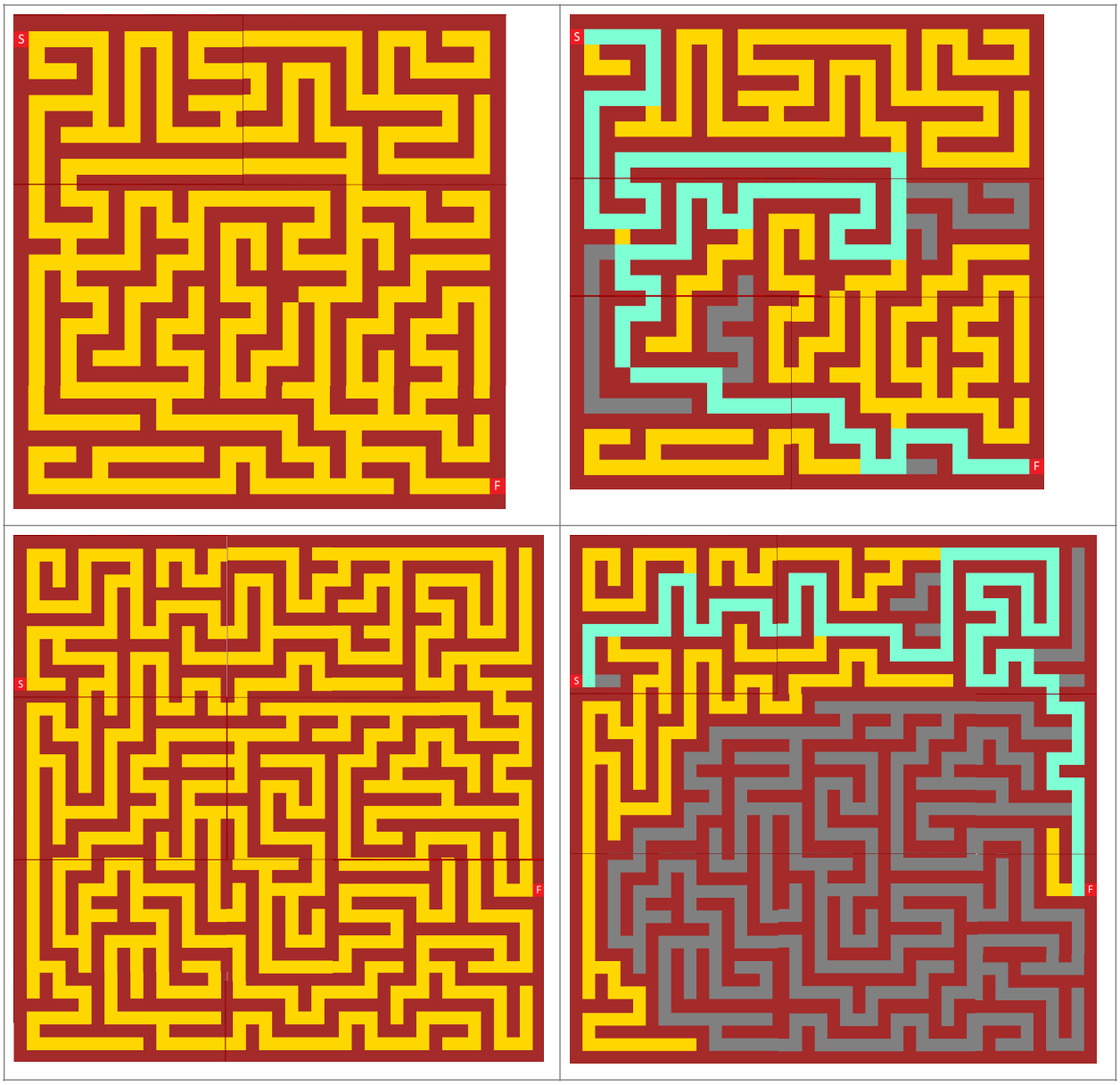
Eksperimen/pengujian

DFS

Jalur yang berwarna biru muda ialah jalur yang benar mencapai finish. Jalur yang berwarna abu-abu ialah jalur ketika pemadam melakukan *backtrack***.** Jalur yang berwarna kuning ialah jalur yang tidak dilalui pemadam.

Before	After


S
F
S
F
S
F
S
F



Analisis Hasil Pengujian

Diantara DFS atau BFS tidak ada algoritma yang lebih unggul dalam kecepatannya mencari *finish* (api), semuanya tergantung pada bentuk *maze*-nya. Terkadang untuk *maze* tertentu DFS lebih unggul, sedang untuk *maze* tertentu BFS lebih cepat dalam mencapai *finish*. Selain itu kecepatan pencarian solusi juga bergantung pada prioritas gerakan robot, apakah robot tersebut bergerak ke kiri dulu atau ke kanan dulu, ke atas dulu atau ke bawah dulu.

Bab 5
Kesimpulan dan Saran

Kesimpulan

Kesimpulan dari tugas ini ialah:

Program berhasil mensimulasikan program pemadam mencari api dalam *maze*. Algoritma yang digunakan dalam simulasi ini ada dua, yaitu algoritma BFS dan DFS dengan *backtracking*. Masing-masing algoritma memiliki cara penyelesaian yang berbeda-beda dalam menyelesaikan masalah pencarian api dalam *maze*, dan masing-masing algoritma tidak ada yang unggul dari satu sama lainnya, semua bergantung dari bentuk *maze* dan prioritas gerakan pemadam.

Saran

Beberapa saran yang dapat diberikan untuk perbaikan program ini antara lain:

Jumlah api dalam labirin bisa lebih dari satu.

Tampilan program bisa dibuat menjadi 3-D.

Penerapan algoritma *branch and bound* untuk BFS.

Referensi

Munir, Rinaldi. 2009. Diktat Kuliah IF3051 Strategi Algoritma.