Fakultas Ilmu Komputer -- Universitas Indonesia
CSCM603130 - Sistem Cerdas
Semester Ganjil - 2016/2017

Tugas 1
Batas pengumpulan: Jumat, 21 Oktober 2016 pukul 23.55 WIB (waktu SCeLE)

Ketentuan Pengerjaan Tugas

- Dalam tugas ini, Anda tidak perlu mengimplementasikan sendiri algoritma-algoritma search yang diperlukan, melainkan menggunakan library (source code) yang bisa diperoleh dari website buku acuan "AI: A Modern Approach" (http://aima.cs.berkeley.edu/code.html). Secara khusus, dalam tugas ini Anda diasumsikan menggunakan library **aima-java** dengan implementasi dalam bahasa Java (https://github.com/aimacode/aima-java**)**.
- Pada library **aima-java** terdapat implementasi berbagai macam pseudocode yang terdapat di dalam buku acuan. Sub-project yang diperlukan hanyalah **aima-core**. Package yang paling penting untuk diperhatikan adalah **aima.core.search** dan **aima.core.logic.propositional** yang memiliki beberapa class dan interface penting untuk membantu menyelesaikan kedua problem pada tugas ini.
- Plagiarisme adalah pelanggaran yang sangat serius. Anda **TIDAK** diperkenankan menyalin program lain yang berasal dari sumber manapun. Anda dapat mendiskusikan tugas ini dengan peserta kuliah lainnya, asisten dosen, dosen, dsb., tetapi program yang Anda buat harus merupakan hasil pekerjaan Anda sendiri. **Peserta yang terbukti melanggar akan diberikan nilai akhir E.**

Ketentuan Pengumpulan Tugas

Yang perlu dikumpulkan adalah:

- File source code Tugas1A.java, Tugas1B.java dan semua file source code *.java lainnya yang dibutuhkan, diletakkan dalam subdirectory yang sesuai dengan spesifikasi package-nya. Berikan dokumentasi pada semua file source code yang Anda buat dan bagian source code yang Anda modifikasi.
- Sebuah file bahasan.pdf yang berisi pembahasan anda (lihat penjelasan pembahasan yang diharapkan di setiap soal).
- Gabungkan semua file di atas ke dalam sebuah arsip ZIP dengan nama file mengikuti format ASDOS_NPM_TUGAS1_SC, misalnya: MADE_123456789_TUGAS1_SC.zip. Bagian ASDOS disesuaikan dengan Asdos yang di-*assign* kepada Anda dan mengikuti format nama berikut: VINA, SHIRLEY, SYUKRI, HARYO, MADE, ARUNI, ILHAM.
- Pengumpulan dilakukan pada submission space di SCeLE. Deadline pengumpulan adalah Jumat, 21 Oktober 2016, pukul 23:55. Submission harus dilakukan sebelum batas akhir, atau peserta dianggap tidak mengumpulkan tugas sama sekali.

A. Jarvis, help me collect my stuffs!

Deskripsi Umum

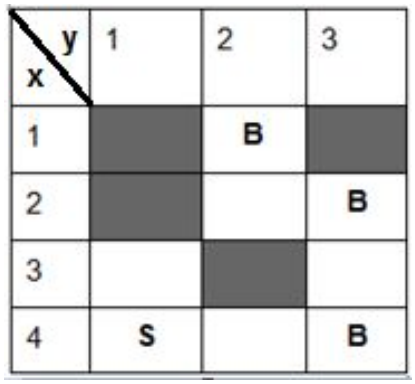
Sesudah kejadian hiking yang dialami Tony (lihat Tutorial 1), Tony berusaha untuk memperbaiki baju besinya sesampainya di rumah agar baju besi tersebut lebih efisien dalam penggunaan oli. Namun, dalam proses upgrade baju besi, Tony mengalami kecelakaan ketika menggunakan baju besi tersebut untuk ujicoba penerbangan yang mengakibatkan patah tulang kaki. Alhasil, Tony terpaksa menggunakan kursi roda selama pemulihan.

Suatu hari, Tony ingin membereskan ruang lab-nya dibantu oleh Jarvis. Tony perlu mengumpulkan semua barang-barangnya yang terpencar di lab sedemikian sehingga ia dapat menempatkannya di tempat awal barang tersebut disimpan. Karena Tony masih merasa kesulitan menggunakan kursi roda untuk bergerak dari satu tempat ke tempat lain, ia perlu bantuan Jarvis untuk memikirkan cara yang paling cepat dan ringkas untuk mengumpulkan semua barang.

Environment lab dapat diperinci sebagai berikut:

- Environment didefinisikan sebagai sebuah matriks berukuran *m* baris x *n* kolom. Sel pojok kiri atas beralamat (1,1) dan pojok kanan bawah beralamat (m,n). Pada environment terdapat beberapa penghalang yaitu perabotan di lab yang tidak dapat dilewati oleh Tony.
- Terdapat tepat satu sel yang ditentukan sebagai sel start (S). Pada initial state, Tony berada di sel tersebut.
- Terdapat beberapa sel yang ditentukan sebagai posisi barang-barang Tony berada (B).
- Terdapat beberapa sel yang ditentukan sebagai penghalang yaitu posisi di mana perabotan berada (ditandai dengan sel berwarna abu-abu) yang menyebabkan Tony tidak dapat melewatinya.

Sebagai contoh, perhatikan gambar di bawah. Sel start ada di alamat (4, 1),barang-barang Tony berada di alamat (1, 2), (2, 3), dan (4, 3), sedangkan penghalang berada di alamat (1,1), (1,3), (2,1), (3,2).



Tony hanya dapat bergerak lurus, yang meliputi action **ATAS, BAWAH, KIRI,** dan **KANAN**, serta melakukan action **AMBIL**. Jika Tony memasuki sel yang berisi barang, maka diharuskan melakukan action **AMBIL**. Kemudian, terdapat *cost* untuk action **ATAS, BAWAH, KIRI,** atau **KANAN** yang dilakukan Tony, yaitu 1. Sedangkan *cost* untuk action **AMBIL** adalah 0.

Tujuan Tony adalah mengambil seluruh barang yang ada di lab dengan cost termurah berdasarkan analisis strategi yang dilakukan oleh Jarvis. Anda diminta untuk mengimplementasikan sebuah program dengan menggunakan bahasa pemograman Java untuk memecahkan permasalahan di atas.

Format Masukan

Program Anda harus bisa membaca sebuah file yang mewakili suatu konfigurasi ruangan lab. File tersebut didefinisikan dengan extension `.lab`, yaitu sebuah file teks biasa yang isinya mengikuti aturan sebagai berikut:

- Baris pertama berisi dua buah bilangan bulat positif `m` dan `n` yang dipisahkan oleh sebuah koma: `m` menyatakan jumlah baris dan `n` menyatakan jumlah kolom.
- Baris kedua berisi dua buah bilangan bulat positif yang dipisahkan oleh sebuah koma, menyatakan baris dan kolom posisi awal Tony (S).
- Baris ketiga berisikan posisi-posisi barang-barang di lab yang dipisahkan dengan sebuah spasi, di mana setiap posisi barang direpresentasikan dengan format "`(x,y)`" tanpa tanda kutip. Dalam hal ini, `x` adalah posisi valid baris dan `y` adalah posisi valid kolom pada lab. Dipastikan bahwa nilai dari `x` dan `y` berbeda dengan posisi sel start dan nilai posisi tiap barang berbeda.
- Baris keempat berisikan koordinat-koordinat yang tidak bisa dilalui (penghalang) yang direpresentasikan dengan format "`(x,y)`" tanpa tanda kutip.

Sebagai contoh, perhatikan gambar di bawah yang menunjukkan isi sebuah file bernama `test.lab`, yang menyatakan konfigurasi awal lab pada gambar di atas.

```
4,3
4,1
(1,2) (2,3) (4,3)
(1,1) (1,3) (2,1) (3,2)
```

Format Keluaran

Program Anda harus bisa menulis sebuah file yang mewakili suatu solusi terhadap sebuah konfigurasi lab. File tersebut didefinisikan dengan extension `.solution`, yaitu sebuah file yang terdiri dari beberapa baris. Baris pertama berisi sebuah bilangan bulat tidak negatif yang merupakan total cost dari strategi yang dihasilkan, baris kedua berisi sebuah bilangan bulat tidak negatif yang merupakan total node yang diekspansi ketika search dilakukan, kemudian baris-baris berikutnya berisi tepat sebuah kata yang dapat dipilih dari himpunan berikut {ATAS, BAWAH, KIRI, KANAN AMBIL}.

Contoh isi `test.solution`:

```
6
74
KANAN
KANAN
AMBIL
ATAS
ATAS
AMBIL
KIRI
ATAS
AMBIL
```

File ini merupakan salah satu solusi dari permasalahan yang dinyatakan dalam file `test.lab` di atas.

Spesifikasi program

Soal ini tidak membutuhkan implementasi GUI apapun. Yang dibutuhkan adalah fasilitas membaca argumen-argumen yang diberikan pada *command-line*. Fasilitas utama yang harus disediakan program Anda adalah mencari solusi berdasarkan konfigurasi ruangan lab (dan posisi awal Tony) yang didefinisikan dalam sebuah file **`.lab`**. Solusi yang ditemukan harus optimal, dan ditulis ke dalam sebuah file **`.solution`** beserta informasi lainnya yang dibutuhkan (lihat Format Keluaran).

Spesifikasi *command-line*:

`java Tugas1A <strategi> <file input> <file output>`

- **`<strategi>`** adalah spesifikasi strategi search yang digunakan. Strategi search yang digunakan adalah sebagai berikut:
  1. *Iterative Deepening Search*. (kode *command-line parameter*: **`ids`**).
  2. *A* search* dengan *heuristic function* sesuai definisi *heuristic function* yang Anda berikan. (kode *command-line parameter*: **`a*`**).
- **`<file input>`** adalah nama file **`.lab`** yang berisi konfigurasi ruangan lab (dan posisi awal Tony).
- **`<file output>`** adalah nama file `.solution` yang berisi solusi optimal berupa rangkaian langkah-langkah yang harus diambil Tony. Sebagai contoh, jika program Anda dipanggil dengan parameter sebagai berikut:

  `java Tugas1A a* test.lab test.solution`

  maka search staregy yang digunakan adalah *A**, file input adalah **`test.lab`**, dan file output adalah **`test.solution`**.

Ujicoba Empiris

Di course page SCeLE akan tersedia sekumpulan file **`.lab`** yang mendefinisikan berbagai jenis kondisi *environment* yang harus diselesaikan oleh program Anda. Jalankan program Anda untuk

mencari solusi semua konfigurasi lab tersebut dengan menggunakan strategi IDS dan A*, lalu buatlah rangkuman statistik jumlah node yang di-expand sebagai berikut:

Nama file	IDS	A*
test1.lab	... nodes	... nodes
test2.lab	... nodes	... nodes
test3.lab	... nodes	... nodes
test4.lab	... nodes	... nodes
...	...	...

Laporan Pembahasan

Laporkan dalam file bahasan.pdf perumusan masalah di atas sebagai sebuah state space search:

- Jelaskan bagaimana cara Anda merepresentasikan sebuah *state*.
- Jelaskan juga definisi *result function* dan *goal test*.
- Berdasarkan representasi state yang Anda buat di atas, jelaskan *heuristic function* yang dapat membantu mempercepat proses *search* mencari solusi. Jelaskan dasar pemikiran Anda dalam merancang *heuristic* tersebut. Jelaskan juga apakah fungsi tersebut bersifat *admissible*. Jika Anda memiliki teknik atau pendekatan yang menurut Anda dapat memperbaiki kinerja program Anda, jelaskan dasar pemikiran dan asumsi yang Anda ambil!

Selain itu, bahaslah hasil ujicoba empiris yang Anda peroleh.

B. How to (logically!) solve Sudoku?

Deskripsi Umum

Sudoku atau *Number Place* adalah puzzle yang terdiri dari n^2 x n^2 *grid* yang terbentuk dari nxn *sub-grid* yang disebut sebagai *block*. Beberapa kotak (*cell*) dari puzzle telah diisi oleh salah satu angka dari 1 hingga n^2.

Contoh sebuah sudoku berukuran 9 x 9:

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Tujuan puzzle ini adalah meletakkan angka 1 sampai 9 pada kotak-kotak yang kosong di atas, sehingga tiap baris, kolom dan *block* terisi oleh 9 angka berbeda.

Berikut adalah contoh solusi dari sudoku berukuran 9x9 di atas:

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Anda diminta untuk mengimplementasikan program yang dapat menyelesaikan sudoku puzzle dengan cara menerjemahkan sudoku puzzle ke dalam **SAT problem**. Untuk soal ini, Anda bisa membaca dan menggunakan referensi yang disediakan di SCeLE ("Sudoku as a SAT problem" oleh I. Lynce & J. Ouaknine) yang menjelaskan representasi Sudoku sebagai SAT problem.

Format Masukan

Program Anda harus bisa membaca sebuah file yang mewakili suatu konfigurasi sebuah Sudoku. File tersebut didefinisikan dengan extension `.sudoku`, yaitu sebuah file yang isinya mengikuti aturan sebagai berikut:

- Baris pertama berisi sebuah bilangan bulat positif n^2, di mana n^2 menyatakan ukuran sudoku puzzle n^2 x n^2.
- Baris kedua sampai baris ke-(n^2+1) adalah konfigurasi Sudoku puzzle n^2xn^2, di mana elemen *cell* yang masih kosong direpresentasikan dengan 0, dan dua cell berturutan dalam satu baris dipisahkan dengan spasi.

Sebagai contoh, gambar di bawah menunjukkan isi sebuah file bernama `test.sudoku` yang menyatakan konfigurasi awal sudoku di atas.

```
9
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9
```

Format Keluaran

Program yang Anda kerjakan harus menampilkan solusi terhadap masalah Sudoku yang diberikan. Bila diberikan Sudoku puzzle berukuran n^2xn^2, maka program harus menghasilkan file keluaran dengan extension `.solution` berisi Sudoku dimana kotak-kotaknya telah terisi penuh dengan angka 1 hingga n^2. Baris pertama sampai baris ke-n^2 adalah konfigurasi Sudoku puzzle n^2xn^2, di mana elemen *cell* yang sebelumnya kosong diisi dengan sebuah angka dari 1 hingga n^2, dan dua cell berturutan dalam satu baris dipisahkan dengan spasi. Sebagai contoh, gambar di bawah menunjukkan isi file bernama `sudoku.solution` yang menyatakan solusi dari konfigurasi awal sudoku di atas.

Contoh isi `sudoku.solution`:

```
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
```

Spesifikasi Program

Tugas ini tidak membutuhkan implementasi GUI apapun. Yang dibutuhkan adalah fasilitas membaca argumen-argumen yang diberikan pada *command-line*. Fasilitas utama yang harus disediakan program anda adalah untuk mencari solusi untuk sebuah sudoku yang didefinisikan dalam sebuah file **`.sudoku`**. Solusi yang ditemukan ditulis ke dalam sebuah file **`.solution`**.

Spesifikasi *command-line*:

```
java Tugas1B <strategi> <file input> <file output>
```

- **`<strategi>`** adalah spesifikasi strategy search yang digunakan. Strategi search yang digunakan adalah sebagai berikut:
  1. DPLL, algoritma backtracking sistematik untuk menyelesaikan problem *SAT* (kode *command-line parameter*: dpll).
  2. *WalkSAT*, algoritma local search untuk menyelesaikan problem *SAT* (kode *command-line parameter*: walksat).
- **`<file input>`** adalah nama file `.sudoku` yang berisi konfigurasi awal sudoku.
- **`<file output>`** adalah nama file `.solution` yang berisi solusi Sudoku berupa konfigurasi sudoku yang telah terisi dengan angka 1 hingga n^2 dengan memenuhi constraints yang ada.

Sebagai contoh, jika program anda dipanggil dengan parameter sebagai berikut:

```
java Tugas1B walksat test.sudoku test.solution
```

maka program Anda mencari solusi dengan strategi WalkSAT sedemikian sehingga Sudoku dengan input file **`test.sudoku`** (lihat format masukan) mempunyai solusi yang dapat dilihat di **`test.solution`** (lihat format keluaran).

Ujicoba Empiris

Di course page SCeLE akan tersedia sekumpulan file **`.sudoku`** yang dapat digunakan sebagai input program Anda. Lakukan eksekusi program Anda pada input tersebut untuk setiap strategi search, yaitu DPLL dan WalkSAT.

Catatan

Berdasarkan pseudocode algoritma DPLL-Satisfiable di buku dan implementasinya di library aima-java, DPLL-Satisfiable hanya memeriksa apakah suatu kalimat logika proposisi satisfiable atau tidak (yaitu, mengembalikan *true* atau *false*). Oleh karena itu, Anda perlu memodifikasi beberapa bagian kode pada file **DPLLSatisfiable.java** agar Anda dapat mengakses assignment variabel-variabel proposisi dari kalimat logika proposisi yang menjadi input bagi algoritma DPLL-Satisfiable.

Laporan Pembahasan

Laporkan dalam file bahasan.pdf perumusan masalah di atas sebagai SAT problem. Secara khusus, Anda diharapkan memberikan uraian berikut:

- Penjelasan mengenai representasi sudoku sebagai SAT problem.

- Penjelasan mengenai hasil ujicoba metode DPLL dan WalkSAT pada semua sudoku yang diberikan.
- Penjelasan mengenai perubahan yang dilakukan pada file DPLLSatisfiable.java untuk mendapatkan solusi sudoku.