平成 19 年度第 2 次募集試験

東京大学情報理工学系研究科創造情報学専攻

プログラミング

注意事項

1. 試験の合図まで，この問題冊子を開いてはいけない．
2. この表紙の下部にある受験番号欄に受験番号を記入しなさい．
3. 答案用紙と下書き用紙が 1 枚ずつ配られる．それぞれに受験番号を記入しなさい．途中で余白が足りなくなったら，試験監督に申し出なさい．最初の 3 問は, PC を用いずに答案用紙に答えを書く問題である．
4. 各受験者に配られた USB メモリに ASCII コードで書かれたファイルが 4 個入っている (`q4.txt`, `q5.txt`, `q6.txt`, `q7.txt`)．改行はすべて CR と LF の対で書かれている (CR は carriage return，LF は line feed である)．

   試験開始前に，USB メモリから上記の 4 つのファイルを自分の PC にコピーしなさい．各ファイルの中身を見て, ASCII 文字列でそれらしいデータが見えることを確認しなさい．確認できたら，PC から手を離しなさい．ファイルにアクセスできない，あるいは中身が ASCII 文字列として読めないなどの場合は試験監督に申し出なさい．言うまでもなく，USB メモリの中身は全受験者に共通である．
5. プログラミング言語は各自の得意なものを使用しなさい．
6. プログラミング言語のマニュアルは 1 冊に限り試験中に参照してもよい．
7. 試験終了時までに，自分の PC 上に受験番号名のディレクトリ/フォルダを作成し，作成したプログラムおよび関連ファイルをその下にコピーしなさい．作成したディレクトリ/フォルダを各受験者に渡された USB メモリにコピーしなさい．
8. 試験終了時に，USB メモリ, 答案用紙, 下書き用紙を回収する．
9. 回収後，試験監督が周回し，各受験者の結果をごく簡単に確認するので，そのまま座席で待機しなさい．全員の確認が終わるまで部屋を出てはいけない．
10. 午後のプログラミングの口頭試問中にプログラムの動作をより精密に確認する．各自の PC 上でプログラムがなるべくすぐに実行できるようにしておきなさい．
11. 全員の確認が終了した後，**各自の PC とこの問題冊子を残し**，部屋から退出しなさい．

受験番号＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

四方を壁で囲まれた $n \times n$ [m^2] ($n \leq 60$) の正方形の敷地がある. 敷地の内部は, 1[m] 間隔の格子点の間に 1[m] 幅のパネルを立て, 壁をつくることができる. ただし, 同じ場所にパネルを重ねて立てることはできない. パネルをたくさん立てて迷路のようなものを作成することができる (図 1, 図 2). 単位正方形をマス目と呼ぶ. マス目は上下左右に壁がなければ, 中にいる人はそれぞれの方向のマス目に 1 ステップで移動できる. 以下, [m] の単位は省略する.

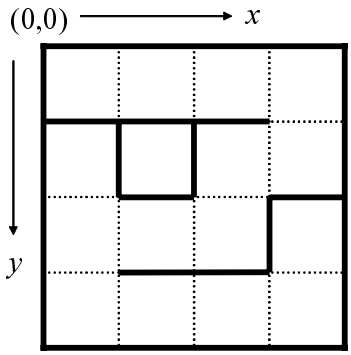


図 1　迷路 1

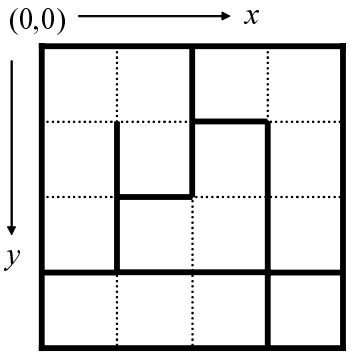


図 2　迷路 2

問 1　3×3 の敷地には, 敷地を複数の領域に区切らないように最大何枚までパネルを立てることができるか? 4×4 の敷地の場合はどうか? 答えは答案用紙に書きなさい.

問 2　$n \times n$ の敷地には, 敷地を複数の領域に区切らないように最大何枚までパネルを立てることができるか? n の式として表しなさい. また, その式が成り立つ理由を説明しなさい. 答えは答案用紙に書きなさい. [ヒント: マス目を頂点とする網の目のグラフの連結性を考えるとよい. なお, これは以下の問題には関係しない.]

問 3　敷地にどのようにパネルを立てるかの設計図は, 図 1 や図 2 のような図面が人にとってわかりやすいが, コンピュータに入力するには設計図を適当な文字列で記号化するのが便利である. いろいろな記号化が考えられるが, ここでは以下のような記法を採用する.

- 一番最初に敷地の大きさを 1 行に書く. 4×4 の場合は 4 が最初の行に書かれる.
- 図 1, 図 2 のように, 左上の頂点を座標 $(0, 0)$ とし, 右方向の座標 x と, 下方向の座標 y を対にした (x, y) で格子点を表す (敷地の境界を含む).
- 格子点 (x, y) と格子点 $(x + 1, y)$ を結ぶパネルは `(x,y)-` と表す. 格子点 (x, y) と格子点 $(x, y + 1)$ を結ぶパネルは `(x,y)|` と書く. `-` は横方向, `|` は縦方向というつもりである.
- 設計図の 2 行目から, 上記のパネルの記法を空白, タブ, 改行などで区切って書き並べる. パネルの順序は問わない.

(1) 図 1 のパネルの立て方を上記の記法にしたがって答案用紙に書きなさい.
(2) 図 2 のパネルの立て方を上記の記法にしたがって答案用紙に書きなさい.

問 4　問 3 の記法に従って書かれた設計図を図示するプログラムを書きなさい. 任意の描画プログラムを使ってよいが, 壁でないところと壁が明確に判別できることが必要である (外周を壁として表示することを忘れないこと). このプログラムを使って, ファイル `q4.txt` にある設計図を図示しなさい. なお, ASCII 文字だけを使って図示するときは, `+`, `-`, `|` の記号と空白を使い, 例えば図 1 の場合, 右のように出力すること.

```
+-+-+-+-+
|         |
+-+-+-+-+ +
| | |     |
+ +-+ +-+-+
|       | |
+ +-+-+ + +
|       |
+-+-+-+-+-+
```

問 5　ファイル `q5.txt` に敷地の設計図がある. この敷地の中にできた, お互いに行き来のできない閉じた領域の面積を降順に出力するプログラムを書きなさい. つまり, 一番広い領域のマス目の数から順に, 一番狭い領域のマス目の数まで空白で区切って出力しなさい. 図 1 の場合は `15 1`, 図 2 の場合は `5 4 3 3 1` という出力になる.

問 6　ファイル `q6.txt` に敷地の設計図がある. 左上隅のマス目から出発して, 右下隅のマス目まで, 最短ステップで行ける経路の一つをマス目の座標の列として出力しなさい. マス目の座標は左上の格子点の座標をそのまま使う. すなわちマス目を (x,y) という座標で表現し, 座標と座標の間は空白もしくは改行とする (見やすくなるように配慮しなさい). 図 1 の場合は, 例えば以下のような出力になる.

```
(0,0) (1,0) (2,0) (3,0) (3,1) (2,1) (2,2) (1,2) (0,2)
(0,3) (1,3) (2,3) (3,3)
```

問 7　「旅の扉」という不思議な仕掛けをマス目の上に置けることになった. 旅の扉は 2 つが対となって異なるマス目上に置くことができる. 迷路を歩いていて, 旅の扉のあるマス目に入った場合, そこで旅の扉に入る決心をすると, 対となる旅の扉があるマス目に一瞬のうちにワープができる. もちろん入らないで通過してもよい. なお, 旅の扉はどちら側からでももう一方の旅の扉のマス目にワープできる. また, 1 つのマス目には高々1 個の旅の扉しか置けない. 旅の扉は設計図では以下のように記述される.

(x_1,y_1)*(x_2,y_2)

これは座標 (x_1, y_1) と (x_2, y_2) に対となる旅の扉が置かれていることを意味する. ファイル `q7.txt` に旅の扉がいくつか配置された設計図がある. 左上隅から右下隅に移動する最短経路の一つを問 6 と同じような記法で出力しなさい. ただし, (x_1, y_1) から (x_2, y_2) へ旅の扉を使った場合

(x_1,y_1)*(x_2,y_2)

と出力しなさい. これが 1 ステップと計算されることに注意.

2007 Winter Examination

Department of Creative Informatics
Graduate School of Information Science and Technology
The University of Tokyo

Programming

INSTRUCTIONS

1. Do not open this problem brochure until the signal to begin is given.
2. Write your examinee ID below on this cover.
3. An answer sheet and a draft sheet are delivered. Write down your examinee ID on both sheets. If you need another sheet in the examination, raise you hand and request it to the test proctor. The first three questions should be answered on the answer sheet, without using your PC.
4. The USB memory delivered beforehand to each examinee contains four ASCII text files: `q4.txt`, `q5.txt`, `q6.txt` and `q7.txt`. Newline is represented by a pair of carriage return (CR) and linefeed (LF) in these files.

   Before examination starts, copy these files to your PC and browse the files to make sure you can see some meaningfully-looking data written in ASCII texts. If you can be sure you can read ASCII characters, keep your hands away from your PC. If you cannot read the files or any ASCII text, consult the test proctor. Needless to say, the contents of the USB memory is common to all examinees.
5. You may choose your favorite programming language.
6. You can consult only one printed manual of the programming language in the examination.
7. By the examination end, make a directory/folder on your PC, whose name is the same as your examinee ID, and put your program files and relevant files under the directory/folder. Copy the directory/folder into the USB memory.
8. At the examination end, the USB memory, the answer sheet and draft sheet are collected.
9. After the collection, stay at your seat, until all examinee results have been checked briefly by the test proctor.
10. After the brief check, try to retain your program execution environment on the PC so as to be able to resume it as soon as possible at the oral examination.
11. **Leave your PC and this brochure together in the room** for the oral examination and leave the room until you are called.

Examinee ID _____

Consider an $n \times n[\mathrm{m}^2]$ ($n \leq 60$) square ground surrounded by walls. You can make a maze-like field by standing 1[m] wide wall panels inside the ground, fitting both ends of each panel at 1[m] interval lattice points as shown in Figure 1 and 2. You cannot stand more than one wall panel on the same place. We will call $1 \times 1[\mathrm{m}^2]$ square on the ground a cell. A person on the ground can walk from a cell to its right, left, top, or bottom adjacent cell in one step if there is not a wall panel between them. We will omit the unit [m] hereafter.

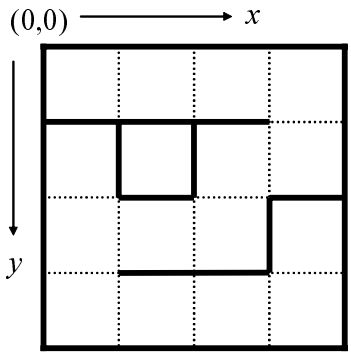


Fig. 1 maze 1

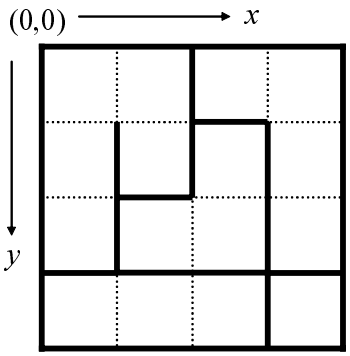


Fig. 2 maze 2

Q1 How many wall panels at most can you stand inside a 3×3 ground so as not to make more than one separate areas? How about for a 4×4 ground? Write your answers on the answer sheet.

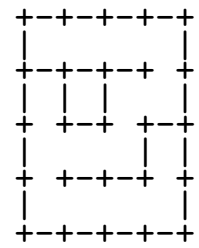
Q2 How many wall panels at most can you stand inside an $n \times n$ ground so as not to make more than one separate areas? Write your answer as a function of n and its reason on the answer sheet. [Hints: Consider a graph consisting of the cells as nodes. The edges of the graph represent whether you can walk between the adjacent cells in one step. This question is not relevant to the following questions.]

Q3 Human can easily understand the design of wall panel setting if it is illustrated as Figure 1 and 2, but programmers are glad if it is represented by an ASCII text. Among various conceivable ways of ASCII text representation, we will use the following notation.

- In the first line, write the size of the side of the ground as a single integer. For example, write `4` for the ground shown in Figure 1.
- A lattice point is represented by its coordinate (x, y) whose origin $(0, 0)$ is the upper-left corner of the ground as shown Figure 1 and 2.
- Write a wall panel standing between lattice points (x, y) and $(x + 1, y)$ as `(`x`,`y`)-`, and write a wall panel standing between lattice points (x, y) and $(x, y + 1)$ as `(`x`,`y`)|`. You can easily understand the meanings of the sign `-` and `|`.
- From the second line, write the set of wall panels standing on the ground as a sequence of panel notations described above, delimited by whitespace (blank, tab, or newline). The order of panels may be arbitrary.

(1) Write the design of Figure 1 in the notation described above on the answer sheet.
(2) Write the design of Figure 2 in the notation described above on the answer sheet.

Q4 Make a program which shows the wall panels of a ground graphically, given a design described in the notation specified in Question 3. You can use your favorite drawing software. Wall panels should be clearly distinguished from other line segments where no wall panel stands. And do not forget to draw the outermost walls of the ground. Execute your program for the design in the file `q4.txt`. If you want to make a program which outputs only ASCII characters, use the following convention as illustrated in the right figure for Figure 1: `+` for lattice point, `-` or `|` for wall panel, and blank for others.



Q5 There is a design in the file `q5.txt`. Write a program which outputs the sequence of integers in descending order, each of which shows the number of cells of separate areas. In other words, your program should output the number of cells included in the largest area, then the number of cells included in the second largest area, and so on. For example, the output for Figure 1 will be `15 1`, and the output for Figure 2 will be `5 4 3 3 1`.

Q6 There is a design in the file `q6.txt`. Write a program which outputs a shortest path between the upper-left cell (start) and the lower-right cell (goal). The path is a sequence of cell coordinates, where each cell coordinate `(`x`,`y`)` is represented by the coordinate of its upper-left lattice point. In the path, cell coordinates should be delimited by whitespace for legibility. For example, the output for the maze 1 in Figure 1 will be

```
(0,0) (1,0) (2,0) (3,0) (3,1) (2,1) (2,2) (1,2) (0,2)
(0,3) (1,3) (2,3) (3,3)
```

Q7 Now, the designer can put a pair of miraculous *warp doors* on a pair of cells. If you encounters a warp door on a cell while you are wandering in a maze, you can simply ignore the door, or you can decide to enter the door. If you enter the door, you can jump in one step onto the cell where the counterpart warp door is put. The jump is bidirectional between the pair of warp doors, and only one warp door can be put in a cell. The warp door is described in the design as follows:

`(`x_1`,`y_1`)*(`x_2`,`y_2`)`

which means that a pair of warp doors is put on cells (x_1, y_1) and (x_2, y_2).

There is a design in the file `q7.txt` which may contain some numbers of warp door pairs. Write a program which outputs a shortest path between the upper-left cell (start) and the lower-right cell (goal) as in the Question 6. If the path contains jump using warp doors, jump from (x_1, y_1) to (x_2, y_2) should be output as

`(`x_1`,`y_1`)*(`x_2`,`y_2`)`

Note that a jump across a pair of warp doors is counted as one step.