Домашнее задание 1.

Лабиринт

Общее описание задания

Задание направленно на проверку знаний основ языка C++ у студентов и знакомство с основными алгоритмами на графах. Задание состоит из следующих этапов:

1. написание программы на языке C++, решающую задачу поиска минимального пути в лабиринте (см. описание ниже);
2. проведение тестирования написанной программы с точки зрения корректности работы программы;
3. теоретический анализ времени работы используемого алгоритма и замеры производительности полученной реализации указанного алгоритма;
4. написание отчета, описывающего результаты всех указанных этапов.

Для выполнения задания предполагается использовать систему контроля версий Git. Каждый студент должен создать отдельный проект на сайте http://mks1.cmc.msu.ru/ и добавить меня в проект для осуществления выполнения задания. Проект должен иметь следующее название: «[SurnameNS] – Autumn 2014 – HW1», где [SurnameNS] – это ваша фамилия и инициалы на английском языке (например, IvanovSI). Задания, выполненные без использования системы Git и соответствующего сайта проверяться не будут. Любые вопросы (в том числе, если у Вас есть возражения, связанные с использованием Dropbox) по заданию присылать по электронной почте на следующий адрес: mikle.shupletsov@gmail.com. Тема письма имеет следующий формат: [318] [Фамилия Имя] [Вопрос].

Для ускорения проверки задания, Вы может в системе GitLab сделать соответствующий ticket на моем имя, таким образом я буду понимать, что Вы закончили выполнение задания и его можно проверять.

Срок выполнения задания: 1 ноября 2015 года (включительно).

Лабиринт

Лабиринт состоит из нескольких уровней. Соседние уровни соединены при помощи лестниц. Каждый уровень в лабиринте представляет собой квадрат из N на N клеток, где каждая клетка может быть одной из следующих типов:

- `'.'` – пустое пространство (по нему можно передвигаться);
- `'#'` – стены лабиринта (клетки по которым нельзя перемещаться);
- `'U'` – лестница на верхний уровень (из данной клетки можно перейти только на уровень выше);
- `'D'` – лестница на нижний уровень (из данной клетки можно перейти только на уровень ниже);
- `'P'` – портал (выход из лабиринта);
- `'S'` – начальная точка.

Передвигаться можно только вверх(`'n'`), вниз (`'s'`), влево (`'w'`) и вправо (`'e'`). Границы лабиринта можно считать непроходимыми, даже если они не окружены стенами (клетки `'#'`).

Формат входных данных.

Описание лабиринта считывается программой либо со стандартного потока ввода (cin), либо из файла. При этом программа должна корректно обрабатывать следующие два основных формата описания лабиринта:

1. «карта» - описание карты лабиринта по уровням (M);
2. «координаты» - список координат элементов лабиринта (L).

Каждая строка в указанных описаниях непосредственно относится к описанию формата или является комментарием в стиле языка С (например, “`//строка с комментарием`”

Формат «карта» (M)

Первая строка содержит символ «M», указывающий тип входного формата. На второй строке указывается одно целое число N – размер стороны любого из уровней лабиринта. На третьей строке указывается число K – число уровней в лабиринте (уровни лабиринта нумеруются с нулевого). Далее, следует описание всех клеток каждого из уровней лабиринта, начиная с нулевого.

Пример корректного входного описания лабиринта в формате «карта» (M):

```
M
4
2
//level 0
....
#...
.#..
#...
//level 1
.P..
....
...S
#.D#
```

Формат «координаты» (L)

Первая строка содержит символ «L», указывающий тип входного формата. На второй строке указывается одно целое число N – размер стороны любого из уровней лабиринта. На третьей строке указывается число K – число уровней в лабиринте (уровни лабиринта нумеруются с нулевого). Далее, следует описание лабиринта, в котором указываются координаты всех элементов (кроме, возможно, клеток `'.'`, описывающих пустое пространство лабиринта). При этом каждая строка описывает координаты одной клетки и имеет следующий формат: (‘строка’, ‘столбец’, ‘уровень’, ‘символ’). Клетки лабиринта описываются в произвольном порядке. Не исключается возможность, что описание содержит координаты пустого пространства.

Пример корректного входного описания лабиринта в формате «координаты» (L):

```
L
4
2
(1,0,0,#)
(2,1,0,#)
(3,0,0,#)
(0,1,1,P)
```

```
(2,3,1,S)
(3,0,1,#)
(3,2,1,D)
(3,3,1,#)
```

Формат выходных данных

Результат поиска минимального пути передается программой либо в стандартный поток вывода (cout), либо в файл. При этом программа должна корректно обрабатывать следующие два основных формата описания результата:

1. «карта» - минимальный путь на карте лабиринта (Map);
2. «координаты» - список координат минимального пути (List).

Формат «карта» (M)

Первая строка содержит символ «M», указывающий тип входного формата. На второй строке указывается одно целое число N – размер стороны любого из уровней лабиринта. На третьей строке указывается число K – число уровней в лабиринте (ровни лабиринта нумеруются с нулевого). Если путь не существует, то более никакой информации не выдается. В противном случае выдается описание всех клеток каждого из уровней лабиринта, начиная с нулевого с указанием минимального пути от клетки `'S'` до клетки `'P'`. Путь указывается следующим образом: символы всех клеток пустого пространства, которые находятся на минимальном пути, заменяются на символы `'n'`, `'e'`, `'s'`, `'w'`, `'d'` и `'u'`, которые указывают направление движения (изменения начинаются с клетки `'S'`, при этом клетка портала 'P' не должна меняться). Описание каждого уровня предваряется строкой с комментарием, указывающим на уровень лабиринта (например, "`//level 0`"). Других комментариев выходной формат содержать не должен.

Пример корректного выходного описания лабиринта в формате «карта» (M):

```
M
4
2
//level 0
....
#...
.#..
#...
//level 1
.Pww
...n
...n
#.d#
```

Формат «координаты» (L)

Первая строка содержит символ «L», указывающий тип входного формата. На второй строке указывается одно целое число N – размер стороны любого из уровней лабиринта. На третьей строке указывается число K – число уровней в лабиринте (уровни лабиринта нумеруются с нулевого). Если путь не существует, то более никакой информации не выдается. В противном случае на следующей строке выдается целое число, которое характеризует длину минимального пути от клетки `'S'` до клетки `'P'`(считается суммарное число клеток в найденном пути, за исключением клетки с порталом (клетка

`'P'`)) и с новой строки выдается описание минимального пути, в котором указываются координаты всех элементов задействованных в этом пути. При этом каждая строка описывает координаты одной клетки и имеет следующий формат: (строка, столбец, уровень, направление), где направление один из следующих символов: `'n'`, `'e'`, `'s'`, `'w'`, `'d'` и `'u'`, указывающий направление движения из клетки. Клетки лабиринта описываются в порядке прохождения пути, то есть от начальной клетки, в которой был символ `'S'` до последней клетки пути, которая указывает на клетку с порталом (клетка с символом `'P'`). Описание не должно содержать комментариев.

Пример корректного входного описания лабиринта в формате «координаты» (L):

```
L
4
2
4
(2,3,1,n)
(1,3,1,n)
(0,3,1,w)
(0,2,1,w)
```

Параметры командной строки.

Программа должна поддерживать следующие параметры командной строки:

1. `--help, -h` – при передаче этого параметра программа должна вывести краткую справку о работе с программой, которая включает краткое описание программы, а также все параметры командной строки и их назначение;
2. `--input (M|L), -i (M|L)` – параметр определяет тип входного формата (в скобках указаны возможные типы входного формата);
3. `--input_file "filename.txt", -I "filename.txt"` – если указан этот параметр, то программа считывает вход из файла с именем `"filename.txt"`, если параметр не указан, то считывание происходит со стандартного потока ввода (cin);
4. `--output (M|L), -o (M|L)` – параметр определяет тип выходного формата (в скобках указаны возможные типы выходного формата);
5. `--output_file "filename.txt", -O "filename.txt", -O` - если указан этот параметр, то программа записывает результат работы в файл с именем `"filename.txt"`, если параметр не указан, то запись происходит в стандартный поток вывода (cout);

Примеры корректного задания параметров командной строки

```
Maze.exe --help //вывод краткой справки о работе программы

Maze.exe –i M –o M /*программа считывает со стандартного потока
ввода карту в формате M выдает результат своей работы в
стандартный поток вывода в формате M.*/

Maze.exe --input M --output L -I test.txt –O result.txt
//*программа считывает карту в формате M из файла test.txt и
записывает результат своей работы в файл result.txt в формате
L.*/
```

Примеры некорректного задания параметров командной строки

```
Maze.exe –h –i M //не согласованные параметры коммандной строки

Maze.exe –i –o M //пропущен обязательный параметр при “-i”

Maze.exe --input M -I test.txt –O result.txt /*не задан выходной
формат представления лабиринта.*/

Maze.exe -o M -I test.txt /*не задан входной формат
представления лабиринта.*/
```

Тестирование программы и замеры производительности.

Программа должна быть протестирована на предмет корректного решения поставленной задачи, корректного считывания входного и генерации выходного форматов представления лабиринта и минимального пути в нем.

Должно быть проведено тестирование программы на случайно сгенерированных лабиринтах разного размера. Для генерации случайных графов предполагается написать соответствующую программу. Установить максимальные значения параметров лабиринта, при которых программа работает за приемлемое время (меньше одной минуты) и не превышает заданных ограничений по памяти (например, память Вашего компьютера) для разных форматов описания лабиринта. Сравнить полученные данные с теоретической оценкой времени работы реализованного Вами алгоритма поиска пути.

Полученные каждым студентом тесты, будут собраны в общую библиотеку тестов, которая будет использоваться для тестирования всех программ. Тесты, которые позволили найти ошибки в чужих программах, будут давать студентам дополнительные баллы.

Требования к отчету

Отчет по заданию должен содержать следующие основные разделы:

1. Введение.
2. Описание алгоритма.
3. Результаты тестирования и замеры производительности.
4. Список литературы.

Раздел «Введение» содержит постановку задачи и краткое описание полученных результатов.

Раздел «Описание алгоритма» содержит описание реализованного алгоритма поиска минимального пути и теоретическую оценку времени его работы.

Раздел «Результаты тестирования и замеры производительности» содержит описания того, как написанная программа тестировалась с точки зрения корректности решения задачи, и результаты сравнения скорости работы программы на различных входных данных с теоретической оценкой времени работы алгоритма. Для наглядности требуется привести не только таблицы, но и графики. Кроме того, в этом разделе должен быть приведен анализ полученных замеров производительности.

Раздел «Список литературы» содержит ссылки на статьи и электронные ресурсы, если таковые были упомянуты в тексте отчета.

Критерии оценки

Результаты выполнения домашнего задания оцениваются по следующим основным критериям:

1 корректность и качество реализации алгоритма поиска минимального пути;
2 правильность считывания входных форматов и генерации выходного формата представления лабиринта и найденного минимального пути в нем;
3 качество оформления кода (styleguide);
4 тестирование программы и анализ производительности;
5 структура и содержание отчета;
6 использование системы контроля версий.

Усложненный вариант задания

Для получения дополнительных баллов может быть выполнен усложненный вариант задания. В исходное описание задания вводятся следующие изменения. Вместо отметок начальной точки `'S'` и портала `'P'` в лабиринте присутствуют клетки, в которых указано целое число. При этом присутствуют все целые числа от `1` до `N` и при этом каждое число появляется ровно в двух различных клетках. По сути, каждая пара чисел задает начальную и конечную точку `i`-го маршрута в лабиринте.

Задача построить маршрут для каждой пары чисел, присутствующих в лабиринте. При этом считается, что все клетки на построенном маршруте превращаются в стены (в соответствующих клетках проставляется символ `'#'`), что может потенциально привести к невозможности построить оставшиеся маршруты.