KIT205 Data Structures & Algorithms

Assignment 2

Introduction

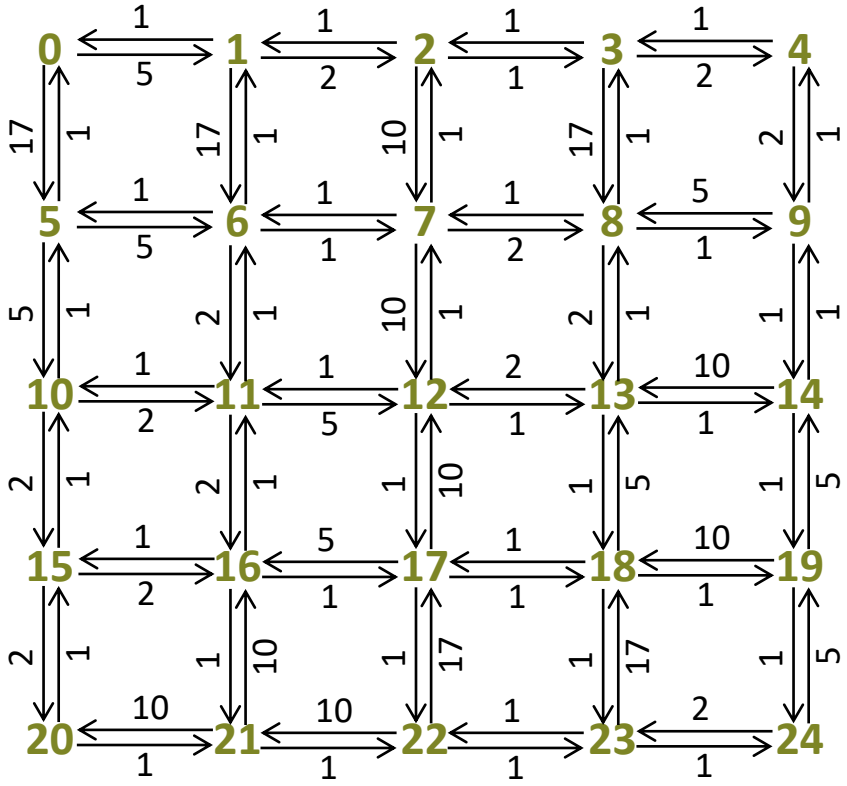
- *You are a member of a team developing a mars rover. Your job is to develop the path-finding system and it is crucially important that you minimise energy consumption. Given a digital elevation map of the terrain, you must identify the path between two points of least energy cost.*

First Step – Build the Graph

DEM

12	14	15	15	16
16	18	18	19	17
18	19	21	20	17
19	20	18	18	15
20	17	14	14	13

Graph



(using cost_funcA)

Code Base – Map Generator

```
int** make_dem(int size, int roughness);
```

- Uses a variation on the Diamond-Square algorithm for generating fractal terrain
- Returns a square grid of elevations in the range 0-99
  - size
    - Must be a power of 2 plus 1
      - e.g. 3, 5, 9, 17, 33, 65,…
  - roughness
    - 0 will give a smooth sloping terrain
    - Higher values give rougher terrain

Code Base – Cost Functions

- `int cost_funcA(int diff);`
- `int cost_funcB(int diff);`
- Returns the cost of moving between adjacent squares and increasing altitude by `diff`

Code Base – Print Functions

- `void print_2D(int** array2D, int size);`
- `void print_2D_ascii(int** array2D, int size);`

- These functions print a 2D int array
- Values in the range 0-99 will be printed correctly
  - Either as ints or as asci-art
- Values that are less than 0 are printed as a marker ( )
  - You should use this to mark the path that you have found with negative numbers

Steps

- I suggest you approach the exam the following way:

1. Work through the week 8 tutorial and make sure you understand the adjacency list data structures
2. Convert your graph construction code from week 8 to build a graph based on the DEM (more on that later)
3. Use you in_degrees code to check that you have built your graph properly
   - If you use the same variable names you should not need to change this code from week 8
4. In a separate project, modify your week 8 code so that it builds a simple graph such as the ones we used as examples in lectures
5. Using this simple graph, work on the solutions for Dijkstra and Floyd
   - I actually think Floyd is a bit easier
6. Finally test your algorithms using your DEM code
   - Solutions are available on MyLO for comparison

Building the Graph

- The basic idea is:
  - Loop through all of the required vertices
    - There will be size*size of them since it's a square DEM
  - You can do this by using either:
    - Nested loops that go through the x and y coordinates and then convert them to graph indices using v = x*size+y
    - A single loop that goes through the vertices and then calculate the corresponding x and y coordinates using x = v/size, y = v%size
  - For each vertex you then (conditionally) add 4 edges using the cost functions to calculate the weight
    - One left (if you're not on the left edge)
    - One right (if you're not on the right edge)
    - One up (if you're not on the top edge)
    - One down (if you're not on the bottom edge)

Shortest Path Functions

- Each shortest path function returns:
  - A distance array (2D array for Floyd)
  - A path array (1D prev array for Dijkstra, 2D next array for Floyd)
- In C a function can only return one thing, so I suggest returning nothing!
- Instead pass (pointers to) the variables that you want to modify (just like the scanf function)
- i.e.

```
void floyd(Graph *self, int** dist, int**next);
void dijkstra(Graph *self, int source, int* dist, int*prev);
```

MyLO

- If you have general questions about the assignment, make sure you ask them on the MyLO discussion forums
- Please make sure you check the discussion forum and announcements on MyLO for more tips and clarifications