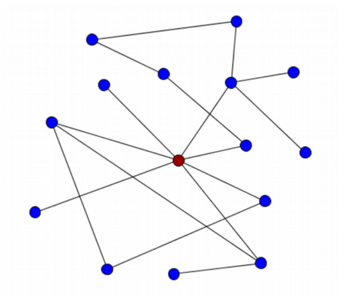
CS 302 – Assignment #09

Purpose: Learn concepts regarding disjoint sets data structure.
Due: Tuesday (11/07) → Must be submitted on-line before class.
Points: Part A → 75 pts, Part B → 25 pts

Assignment:

Part A:

A sensor network[1] is comprised of spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to other locations. This will only work well if there is a high degree of connectivity between the various sensor nodes. However, this is not always possible. We will create a program to read node connection information and determine connectivity in a sensor network.



Design and implement a C++ class, ***disjointSets***, to implement a disjoint sets[2] data structure. The disjoint sets class will use arrays for the parent links and sizes. A test program is provided to allow independent testing of the disjoint set class.

WHEN A USER TAKES A PHOTO, THE APP SHOULD CHECK WHETHER THEY'RE IN A NATIONAL PARK...
SURE, EASY GIS LOOKUP. GIMME A FEW HOURS.
...AND CHECK WHETHER THE PHOTO IS OF A BIRD.
I'LL NEED A RESEARCH TEAM AND FIVE YEARS.
IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

Once the disjoint sets class is working, create a new class, ***sensorNetwork***, to read the data and determine connectivity and provide some summary information.

A main will be provided. Refer to the UML descriptions for implementation details.

Part B:

Create and submit a brief write-up, not to exceed ~500 words, including the following:

- Name, Assignment, Section.
- Summary of data structures used.
- Note some other applications for the disjoint sets data structure.
- Big-O for disjoint sets functions (*setUnion* and *setFind*).
- Provide an explanation of why path compression is useful for this specific application.

Submission:

- Submit a compressed zip file of the program source files, header files, and makefile via the on-line submission by 23:50.
- Submit a copy of the write-up (PDF format).

All necessary files must be included in the ZIP file. The grader will download, uncompress, and type **make**. You must have a valid, working *makefile*.

1 For more information, refer to: https://en.wikipedia.org/wiki/Wireless_sensor_network
2 For more information, refer to: http://en.wikipedia.org/wiki/Disjoint-set_data_structure

Class Descriptions

- Disjoint Sets Class
  The disjoint sets class will implement functions specified below.

disjointSets
-totalSize: int
-setsCount: int
-links: *int
-sizes: *int
-MIN_SIZE=10: static constexpr int
-MAX_SIZE=3500000: static constexpr int
+disjointSets()
+~disjointSets()
+createSets(int): bool
+getTotalSets() const: int
+getSetCount() const: int
+getSetSize(const int) const: int
+printSets() const: void
+setUnion(int, int): int
+setFind(int): int

Function Descriptions

- The *disjointSets()* constructor should initialize the class variables to an empty state as appropriate (0, NULL, etc.).
- The *~disjointSets()* destructor should delete the dynamically allocated memory.
- The *createSets()* function should create the new sets of the passed size. The parameter must be checked to ensure it is $\geq$ MIN_SIZE and $\leq$ to MAX_SIZE. If invalid, nothing should be changed. If the sets are successfully created, the function should return TRUE and FALSE otherwise.
- The *getTotalSets()* function should return total sets class variable which represents the number of total possible sets.
- The *getSetCount()* function should return the current number of unique sets (which will always be $\leq$ to the number of possible sets).
- The *getSetSize()* function should return the current set size of the passed set.
- The *printSet()* function should print the current disjoint set status; index, links, and sizes. Refer to the example output for formatting. This is primarily used for debugging.
- The *setUnion()* function should perform a *union-by-size* operation between the two passed sets, update the sizes, and return the parent. If the set sizes are equal, the parent should be the 1st argument.
- The *setFind()* functions should search for and return the parent of the passed set. *Note*, this may be the set itself. *Note*, the function ***must*** perform path compression.

- Sensor Network Class
  The sensor network class will implement functions specified below.

sensorNetwork
-maxSensors: int
+sensorNetwork()
+~sensorNetwork()
+readFile(const string): bool
+findLimits(int &, int &, int &, int &, int &, int &): void
+showStats() const: void
+createReport(const string): bool

Function Descriptions

- The *sensorNetwork()* constructor should initialize the class variables as appropriate (0, NULL, etc.).
- The *~sensorNetwork()* destructor should delete any dynamically allocated memory. Additionally, reset all class variables as appropriate.
- The *readFile()* function should read the sensor connectivity data file. When done, the file should be closed. Refer to the File Format section for details of the file contents. If the file operations are successful, the function should return true or if there are any I/O errors return false.
- The *findLimits()* function should find the count of nodes and parent node for each of the 1st, 2nd largest and smallest subgroups (in that order).
- The *showStats()* function should display a summary of the connectivity information including the total count of sensors and number of connected subgroups. Additionally, for the 1st, 2nd largest and smallest subgroups the count of nodes, Parent Node, and connectivity percentage should be displayed. Refer to the example output for formatting.
- The *createReport()* function should write a report to the passed file name containing a connectivity status for every unique connected group. The report should contain columns for group number (0 – count of groups), parent node, group size, and connectivity percentage. The field sizes for the report are 8, 12, 12, and 12 (in that order). Refer to the example output for formatting examples.

Refer to the example executions for output formatting. Make sure your program includes the appropriate documentation. See Program Evaluation Criteria for CS 302 for additional information. ***Note, points will be deducted for especially poor style or inefficient coding.***

File Format

The first line in the file is the title, the second line contains the count of unique nodes, and the third line is the count of connections in the file. Every line thereafter includes the from node, the to node, and a signal strength (which is not used in the program). For example:

```
Sensor Network 0 for simple testing
11
55
0 1 29.0
0 2 20.0
0 3 21.0
...
```

Note, this example was truncated.

Example Execution:

This is example output for the disjoint sets test program.

```
ed-vm% ./testDS
*****************************************************************
CS 302 - Assignment #9
Disjoint Sets

----------------------------------------
Test Set 0

Initial State:
  index:  0  1  2  3  4  5  6  7  8  9
  links: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
  sizes:  1  1  1  1  1  1  1  1  1  1

----------------------------------------
  union(0,1) -> 0
  union(2,3) -> 2
  union(4,5) -> 4

New State 1:
Size: 10
  index:  0  1  2  3  4  5  6  7  8  9
  links: -1  0 -1  2 -1  4 -1 -1 -1 -1
  sizes:  2  0  2  0  2  0  1  1  1  1

----------------------------------------
  union(0,2) -> 0
  union(4,6) -> 4
  union(4,7) -> 4
  union(4,8) -> 4

New State 2:
  index:  0  1  2  3  4  5  6  7  8  9
  links: -1  0  0  2 -1  4  4  4  4 -1
  sizes:  4  0  0  0  5  0  0  0  0  1

----------------------------------------
  setFind(1): 0
  setFind(2): 0
  setFind(4): 4
  setFind(7): 4

New State 3:
  index:  0  1  2  3  4  5  6  7  8  9
  links: -1  0  0  2 -1  4  4  4  4 -1
  sizes:  4  0  0  0  5  0  0  0  0  1

----------------------------------------
m0123 = 0
m45678 = 4
  union(m0123,m45678) -> 4

New State 4:
  index:  0  1  2  3  4  5  6  7  8  9
  links:  4  0  0  2 -1  4  4  4  4 -1
  sizes:  0  0  0  0  9  0  0  0  0  1

----------------------------------------
  setFind(3): 4
  setFind(5): 4
  setFind(7): 4

New State 5:
  index:  0  1  2  3  4  5  6  7  8  9
  links:  4  0  4  4 -1  4  4  4  4 -1
  sizes:  0  0  0  0  9  0  0  0  0  1

----------------------------------------
  setFind(0): 4

New State 6:
  index:  0  1  2  3  4  5  6  7  8  9
  links:  4  0  4  4 -1  4  4  4  4 -1
  sizes:  0  0  0  0  9  0  0  0  0  1

----------------------------------------
  setFind(4): 4
  setFind(6): 4
  setFind(8): 4

Final State:
  index:  0  1  2  3  4  5  6  7  8  9
  links:  4  0  4  4 -1  4  4  4  4 -1
  sizes:  0  0  0  0  9  0  0  0  0  1

----------------------------------------
Test Set 1


Initial State:
  index:  0  1  2  3  4  5  6  7  8  9
  links: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
  sizes:  1  1  1  1  1  1  1  1  1  1

Set Size: 9

Final State:
  index:  0  1  2  3  4  5  6  7  8  9
  links: -1 -1  1  1  1  1  1  1  1  1
  sizes:  1  9  0  0  0  0  0  0  0  0

testDS: Error 0
----------------------------------------
Quiz Test Set


Initial State:
  index:  0  1  2  3  4  5  6  7  8  9
  links: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
  sizes:  1  1  1  1  1  1  1  1  1  1


Intermediate State 1:
  index:  0  1  2  3  4  5  6  7  8  9
  links: -1  0 -1  2 -1  4 -1 -1 -1 -1
  sizes:  2  0  2  0  2  0  1  1  1  1


Intermediate State 2:
  index:  0  1  2  3  4  5  6  7  8  9
  links: -1  0  0  2 -1  4  4  4  4 -1
  sizes:  4  0  0  0  5  0  0  0  0  1


Final State:
  index:  0  1  2  3  4  5  6  7  8  9
  links:  4  0  0  0 -1  4  4  4  4 -1
  sizes:  0  0  0  0  9  0  0  0  0  1

find(0) = 4
find(4) = 4
find(7) = 4
find(6) = 4
find(8) = 4
find(9) = 9

----------------------------------------
Large Set Test

Large Set Test, Sets Count - OK
Large Set Test, Sets - OK

*****************************************************************
Game Over, thank you for playing.
ed-vm%
```

Example Execution:

Below are a couple of example executions. Given the following execution;

```
ed-vm%
ed-vm%
ed-vm% ./connect -df bad0.txt -rpt tmp.txt
*****************************************************************
CS 302 - Assignment #9
Sensor Network Connectivity Program

readFile: Error, invalid data file range.
Error reading input file.

*****************************************************************
Game Over, thank you for playing.
ed-vm%
ed-vm%
ed-vm%
ed-vm% ./connect -df test/sn2.txt -rpt rpt2.txt
*****************************************************************
CS 302 - Assignment #9
Sensor Network Connectivity Program

  Total Sensors in Network:      63732
  Total Connected Sub-Groups:    145

  Largest Connected Group:       63392
  Parent Node (largest group):   1
  Connectivity (largest group):  99.47

  2nd Largest Connected Group:   11
  Parent Node (2nd group):       51423
  Connectivity (2nd group):      0.02

  Smallest Connected Group:      0
  Parent Node (smallest group):  2
  Connectivity (smallest group): 0.00


*****************************************************************
Game Over, thank you for playing.
ed-vm%
ed-vm%
ed-vm% cat rpt2.txt
Sensor Network Connectivity Report
---------------------------------

Total Unique Groups: 145

   Group      Parent       Group     Connect
  Number        Node        Size     Percent
  ------    --------    --------    --------
       0           0           1        0.00
       1           1       63392       99.47
       2       15606           2        0.00
       3       37787           2        0.00
       4       39977           2        0.00
       5       40939           2        0.00
       6       41381           2        0.00
       7       41688           2        0.00
       8       42328           2        0.00
       9       42885           2        0.00
      10       43721           2        0.00
      11       44728           2        0.00
      12       45578           2        0.00
      13       47050           3        0.00
      14       47053           2        0.00
      15       47621           3        0.00
      16       47685           3        0.00
      17       48368           5        0.01
      18       48662           2        0.00
      19       48790           2        0.00
      20       48868           2        0.00
      21       48885           2        0.00
      22       48981           3        0.00
```

23	49146	2	0.00
24	49212	2	0.00
25	49268	3	0.00
26	49587	2	0.00
27	49938	2	0.00
28	49960	2	0.00
29	50908	2	0.00
30	51266	2	0.00
31	51382	2	0.00
32	51423	11	0.02
33	51524	4	0.01
34	52290	2	0.00
35	52323	2	0.00
36	52555	2	0.00
37	52649	3	0.00
38	52667	3	0.00
39	52880	2	0.00
40	52892	2	0.00
41	53299	2	0.00
42	53498	2	0.00
43	53793	2	0.00
44	53809	2	0.00
45	53949	3	0.00
46	54138	3	0.00
47	54185	2	0.00
48	54654	2	0.00
49	54882	3	0.00
50	54960	3	0.00
51	55178	2	0.00
52	55297	2	0.00
53	55357	3	0.00
54	55407	3	0.00
55	55461	2	0.00
56	55499	2	0.00
57	55520	3	0.00
58	55550	2	0.00
59	55652	2	0.00
60	55687	3	0.00
61	55912	2	0.00
62	56089	2	0.00
63	56246	3	0.00
64	56286	2	0.00
65	56418	2	0.00
66	56489	2	0.00
67	56531	2	0.00
68	56611	5	0.01
69	56714	2	0.00
70	56787	2	0.00
71	57053	2	0.00
72	57102	3	0.00
73	57168	2	0.00
74	57286	2	0.00
75	57517	4	0.01
76	57839	2	0.00
77	57934	3	0.00
78	58351	2	0.00
79	58361	2	0.00
80	58433	2	0.00
81	58517	2	0.00
82	58888	2	0.00
83	58900	2	0.00
84	59141	4	0.01
85	59176	2	0.00
86	59181	2	0.00
87	59190	2	0.00
88	59203	2	0.00
89	59259	2	0.00
90	59295	3	0.00
91	59447	2	0.00
92	59466	3	0.00
93	59638	2	0.00
94	59642	2	0.00
95	59838	2	0.00
96	59883	3	0.00
97	59925	3	0.00

```
      98        60083            3         0.00
      99        60107            5         0.01
     100        60179            2         0.00
     101        60357            2         0.00
     102        60394            2         0.00
     103        60525            2         0.00
     104        60815            2         0.00
     105        60945            3         0.00
     106        60974            2         0.00
     107        61043            2         0.00
     108        61214            4         0.01
     109        61243            2         0.00
     110        61277            2         0.00
     111        61445            3         0.00
     112        61852            2         0.00
     113        61908            2         0.00
     114        61935            2         0.00
     115        61954            2         0.00
     116        61963            2         0.00
     117        62191            2         0.00
     118        62220            2         0.00
     119        62267            2         0.00
     120        62297            3         0.00
     121        62301            2         0.00
     122        62311            2         0.00
     123        62354            3         0.00
     124        62472            2         0.00
     125        62509            2         0.00
     126        62568            2         0.00
     127        62636            2         0.00
     128        62831            2         0.00
     129        62840            2         0.00
     130        62876            2         0.00
     131        62921            2         0.00
     132        62926            2         0.00
     133        63124            2         0.00
     134        63212            2         0.00
     135        63214            2         0.00
     136        63334            2         0.00
     137        63541            2         0.00
     138        63554            2         0.00
     139        63575            2         0.00
     140        63619            2         0.00
     141        63632            2         0.00
     142        63646            2         0.00
     143        63682            2         0.00
     144        63697            2         0.00

ed-vm%
ed-vm%
```