

Homework 9

Start of Report

```
hermodics.uci.edu - PuTTY
using namespace std;
#include <iostream>
#include <string>
#include <fstream>
#include <limits>
#include "graph.h"
#include "priorityqueue.h"

struct Verts
{
    int id;
    int dist;
    int prev;
};

void dijkstras(Graph& g, int src, Verts* items)
//Big-Oh = O(N^2)
{
    PriorityQueue<Verts> Q(g.num_vertex);
    for(int i = 0; i < g.num_vertex; ++i)
    {
        items[i].id = g.vertices[i]->id;
        items[i].dist = 10000000;
        items[i].prev = -1;
        Q.enqueue(items[i]);
    }
    items[src].dist = 0;
    Q.setFrontZero(src);
    while(!Q.isEmpty())
    {
        Verts thing = Q.dequeue();
        int id = thing.id;
        for(auto some: g.vertices[id]->edges)
        {
            int v = some.dst;
            if(items[v].dist > items[id].dist + some.weight)
            {
                items[v].dist = items[id].dist + some.weight;
                items[v].prev = id;
                Q.decreaseKey(v, items[v].dist );
            }
        }
    }
}
```

So here is my code for dijkstras with its time complexity. Now there is something wrong with this function because I am not getting the results that I want and I am getting memory leaks. I am not sure if it has to do with my priorityqueue class or my other graph class since I had to make changes to it. Eitherway, I am out of time and this was the best that I can produce.

```
hermod.ics.uci.edu - PuTTY
-bash-4.2$ valgrind dijkstras 0 large.graph
==11874== Memcheck, a memory error detector
==11874== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==11874== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==11874== Command: dijkstras 0 large.graph
==11874==
==11874== Conditional jump or move depends on uninitialised value(s)
==11874==   at 0x401FCE: PriorityQueue<Verts>::siftSmallestUp(Verts*, int, int) (priorityqueue.h:109)
==11874==   by 0x401ECD: PriorityQueue<Verts>::heapify(Verts*, int) (priorityqueue.h:98)
==11874==   by 0x401E32: PriorityQueue<Verts>::heapsort(Verts*, int) (priorityqueue.h:82)
==11874==   by 0x401D7A: PriorityQueue<Verts>::decreaseKey(int, int) (priorityqueue.h:136)
==11874==   by 0x4015AD: dijkstras(Graph&, int, Verts*) (main.cpp:41)
==11874==   by 0x401797: main (main.cpp:86)
==11874==
==11874== Conditional jump or move depends on uninitialised value(s)
==11874==   at 0x401D2F: PriorityQueue<Verts>::decreaseKey(int, int) (priorityqueue.h:127)
==11874==   by 0x4015AD: dijkstras(Graph&, int, Verts*) (main.cpp:41)
==11874==   by 0x401797: main (main.cpp:86)
==11874==
==11874== Conditional jump or move depends on uninitialised value(s)
==11874==   at 0x401FCE: PriorityQueue<Verts>::siftSmallestUp(Verts*, int, int) (priorityqueue.h:109)
==11874==   by 0x401E8B: PriorityQueue<Verts>::heapsort(Verts*, int) (priorityqueue.h:89)
==11874==   by 0x401D7A: PriorityQueue<Verts>::decreaseKey(int, int) (priorityqueue.h:136)
==11874==   by 0x4015AD: dijkstras(Graph&, int, Verts*) (main.cpp:41)
==11874==   by 0x401797: main (main.cpp:86)
==11874==
==11874== Use of uninitialised value of size 8
==11874==   at 0x401450: dijkstras(Graph&, int, Verts*) (main.cpp:34)
==11874==   by 0x401797: main (main.cpp:86)
==11874==
==11874== Use of uninitialised value of size 8
==11874==   at 0x4014F8: dijkstras(Graph&, int, Verts*) (main.cpp:37)
==11874==   by 0x401797: main (main.cpp:86)
==11874==
id 0, dist 0, prev -1
id 1, dist 1, prev 0
id 2, dist 3, prev 0
id 3, dist 6, prev 0
id 4, dist 4, prev 0
id 5, dist 8, prev 2
id 6, dist 17, prev 2
id 7, dist 22, prev 4
id 8, dist 19, prev 4
id 9, dist 29, prev 7
id 10, dist 27, prev 7
id 11, dist 24, prev 7
id 12, dist 40, prev 9
id 13, dist 35, prev 9
id 14, dist 59, prev 12
id 15, dist 46, prev 12
id 16, dist 52, prev 12
id 17, dist 10000000, prev -1
id 18, dist 10000000, prev -1
id 19, dist 10000000, prev -1
id 20, dist 10000000, prev -1
id 21, dist 10000000, prev -1
id 22, dist 10000000, prev -1
id 23, dist 10000000, prev -1
id 24, dist 10000000, prev -1
id 25, dist 10000000, prev -1
id 26, dist 10000000, prev -1
id 27, dist 10000000, prev -1
id 28, dist 10000000, prev -1
id 29, dist 10000000, prev -1
id 30, dist 10000000, prev -1
```

```

hermod.ics.uci.edu - PuTTY
id 81, dist 10000000, prev -1
id 82, dist 10000000, prev -1
id 83, dist 10000000, prev -1
id 84, dist 10000000, prev -1
id 85, dist 10000000, prev -1
id 86, dist 10000000, prev -1
id 87, dist 10000000, prev -1
id 88, dist 10000000, prev -1
id 89, dist 10000000, prev -1
id 90, dist 10000000, prev -1
id 91, dist 10000000, prev -1
id 92, dist 10000000, prev -1
id 93, dist 10000000, prev -1
id 94, dist 10000000, prev -1
id 95, dist 10000000, prev -1
id 96, dist 10000000, prev -1
id 97, dist 10000000, prev -1
id 98, dist 10000000, prev -1
id 99, dist 10000000, prev -1
==11874== Invalid read of size 8
==11874==    at 0x4031D0: Graph::~~Graph() (graph.cpp:58)
==11874==    by 0x401893: main (main.cpp:80)
==11874== Address 0x5ab4260 is 0 bytes after a block of size 800 alloc'd
==11874==    at 0x4C2A888: operator new[](unsigned long) (vg_replace_malloc.c:423)
==11874==    by 0x402E9F: Graph::Graph(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >) (graph.cpp:15)
==11874==    by 0x401741: main (main.cpp:80)
==11874==
==11874== Invalid read of size 8
==11874==    at 0x4032A3: std::vector<Edge, std::allocator<Edge> >::~~vector() (stl_vector.h:567)
==11874==    by 0x40327F: Vertex::~Vertex() (vertex.h:9)
==11874==    by 0x4031DF: Graph::~~Graph() (graph.cpp:58)
==11874==    by 0x401893: main (main.cpp:80)
==11874== Address 0x370 is not stack'd, malloc'd or (recently) free'd
==11874==
==11874== Process terminating with default action of signal 11 (SIGSEGV): dumping core
==11874== Access not within mapped region at address 0x370
==11874==    at 0x4032A3: std::vector<Edge, std::allocator<Edge> >::~~vector() (stl_vector.h:567)
==11874==    by 0x40327F: Vertex::~Vertex() (vertex.h:9)
==11874==    by 0x4031DF: Graph::~~Graph() (graph.cpp:58)
==11874==    by 0x401893: main (main.cpp:80)
==11874== If you believe this happened as a result of a stack
==11874== overflow in your program's main thread (unlikely but
==11874== possible), you can try to increase the size of the
==11874== main thread stack using the --main-stacksize= flag.
==11874== The main thread stack size used in this run was 8388608.
==11874==
==11874== HEAP SUMMARY:
==11874==    in use at exit: 800 bytes in 1 blocks
==11874== total heap usage: 806 allocs, 805 frees, 101,076 bytes allocated
==11874==
==11874== LEAK SUMMARY:
==11874==    definitely lost: 0 bytes in 0 blocks
==11874==    indirectly lost: 0 bytes in 0 blocks
==11874==    possibly lost: 0 bytes in 0 blocks
==11874==    still reachable: 800 bytes in 1 blocks
==11874==    suppressed: 0 bytes in 0 blocks
==11874== Rerun with --leak-check=full to see details of leaked memory
==11874==
==11874== For counts of detected and suppressed errors, rerun with: -v
==11874== Use --track-origins=yes to see where uninitialised values come from
==11874== ERROR SUMMARY: 62 errors from 7 contexts (suppressed: 0 from 0)
Segmentation fault
-bash-4.2$

```

Here is my valgrind with source 0 ran with large.graph. I didn't know how to get the result I wanted and even then I was still confused on how I was supposed to produced the desired output if I didn't know how the information was obtained.

```
hermod.ics.uci.edu - PuTTY
-bash-4.2$ valgrind dijkstras 27 large.graph
==12077== Memcheck, a memory error detector
==12077== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==12077== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==12077== Command: dijkstras 27 large.graph
==12077==
==12077== Conditional jump or move depends on uninitialised value(s)
==12077==   at 0x401FCE: PriorityQueue<Verts>::siftSmallestUp(Verts*, int, int) (priorityqueue.h:109)
==12077==   by 0x401ECD: PriorityQueue<Verts>::heapify(Verts*, int) (priorityqueue.h:98)
==12077==   by 0x401E32: PriorityQueue<Verts>::heapsort(Verts*, int) (priorityqueue.h:82)
==12077==   by 0x401D7A: PriorityQueue<Verts>::decreaseKey(int, int) (priorityqueue.h:136)
==12077==   by 0x4015AD: dijkstras(Graph&, int, Verts*) (main.cpp:41)
==12077==   by 0x401797: main (main.cpp:86)
==12077==
==12077== Conditional jump or move depends on uninitialised value(s)
==12077==   at 0x401FCE: PriorityQueue<Verts>::siftSmallestUp(Verts*, int, int) (priorityqueue.h:109)
==12077==   by 0x401E8B: PriorityQueue<Verts>::heapsort(Verts*, int) (priorityqueue.h:89)
==12077==   by 0x401D7A: PriorityQueue<Verts>::decreaseKey(int, int) (priorityqueue.h:136)
==12077==   by 0x4015AD: dijkstras(Graph&, int, Verts*) (main.cpp:41)
==12077==   by 0x401797: main (main.cpp:86)
==12077==
==12077== Use of uninitialised value of size 8
==12077==   at 0x401450: dijkstras(Graph&, int, Verts*) (main.cpp:34)
==12077==   by 0x401797: main (main.cpp:86)
==12077==
==12077== Use of uninitialised value of size 8
==12077==   at 0x4014F8: dijkstras(Graph&, int, Verts*) (main.cpp:37)
==12077==   by 0x401797: main (main.cpp:86)
==12077==
==12077== Conditional jump or move depends on uninitialised value(s)
==12077==   at 0x401D2F: PriorityQueue<Verts>::decreaseKey(int, int) (priorityqueue.h:127)
==12077==   by 0x4015AD: dijkstras(Graph&, int, Verts*) (main.cpp:41)
==12077==   by 0x401797: main (main.cpp:86)
==12077==
id 0, dist 10000000, prev -1
id 1, dist 10000000, prev -1
id 2, dist 10000000, prev -1
id 3, dist 10000000, prev -1
id 4, dist 10000000, prev -1
id 5, dist 10000000, prev -1
id 6, dist 10000000, prev -1
id 7, dist 10000000, prev -1
id 8, dist 10000000, prev -1
id 9, dist 10000000, prev -1
id 10, dist 10000000, prev -1
id 11, dist 10000000, prev -1
id 12, dist 10000000, prev -1
id 13, dist 10000000, prev -1
id 14, dist 10000000, prev -1
id 15, dist 10000000, prev -1
id 16, dist 10000000, prev -1
id 17, dist 10000000, prev -1
id 18, dist 10000000, prev -1
id 19, dist 10000000, prev -1
id 20, dist 10000000, prev -1
id 21, dist 10000000, prev -1
id 22, dist 10000000, prev -1
id 23, dist 10000000, prev -1
id 24, dist 10000000, prev -1
id 25, dist 10000000, prev -1
id 26, dist 10000000, prev -1
id 27, dist 0, prev -1
id 28, dist 3, prev 27
id 29, dist 12, prev 27
id 30, dist 17, prev 29
```

```

hermod.ics.uci.edu - PuTTY
id 81, dist 10000000, prev -1
id 82, dist 10000000, prev -1
id 83, dist 10000000, prev -1
id 84, dist 10000000, prev -1
id 85, dist 10000000, prev -1
id 86, dist 10000000, prev -1
id 87, dist 10000000, prev -1
id 88, dist 10000000, prev -1
id 89, dist 10000000, prev -1
id 90, dist 10000000, prev -1
id 91, dist 10000000, prev -1
id 92, dist 10000000, prev -1
id 93, dist 10000000, prev -1
id 94, dist 10000000, prev -1
id 95, dist 10000000, prev -1
id 96, dist 10000000, prev -1
id 97, dist 10000000, prev -1
id 98, dist 10000000, prev -1
id 99, dist 10000000, prev -1
==12077== Invalid read of size 8
==12077==    at 0x4031D0: Graph::~~Graph() (graph.cpp:58)
==12077==    by 0x401893: main (main.cpp:80)
==12077== Address 0x5ab4260 is 0 bytes after a block of size 800 alloc'd
==12077==    at 0x4C2A888: operator new[](unsigned long) (vg_replace_malloc.c:423)
==12077==    by 0x402E9F: Graph::Graph(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >) (graph.cpp:15)
==12077==    by 0x401741: main (main.cpp:80)
==12077==
==12077== Invalid read of size 8
==12077==    at 0x4032A3: std::vector<Edge, std::allocator<Edge> >::~~vector() (stl_vector.h:567)
==12077==    by 0x40327F: Vertex::~Vertex() (vertex.h:9)
==12077==    by 0x4031DF: Graph::~~Graph() (graph.cpp:58)
==12077==    by 0x401893: main (main.cpp:80)
==12077== Address 0x370 is not stack'd, malloc'd or (recently) free'd
==12077==
==12077== Process terminating with default action of signal 11 (SIGSEGV): dumping core
==12077== Access not within mapped region at address 0x370
==12077==    at 0x4032A3: std::vector<Edge, std::allocator<Edge> >::~~vector() (stl_vector.h:567)
==12077==    by 0x40327F: Vertex::~Vertex() (vertex.h:9)
==12077==    by 0x4031DF: Graph::~~Graph() (graph.cpp:58)
==12077==    by 0x401893: main (main.cpp:80)
==12077== If you believe this happened as a result of a stack
==12077== overflow in your program's main thread (unlikely but
==12077== possible), you can try to increase the size of the
==12077== main thread stack using the --main-stacksize= flag.
==12077== The main thread stack size used in this run was 8388608.
==12077==
==12077== HEAP SUMMARY:
==12077==    in use at exit: 800 bytes in 1 blocks
==12077== total heap usage: 806 allocs, 805 frees, 101,076 bytes allocated
==12077==
==12077== LEAK SUMMARY:
==12077==    definitely lost: 0 bytes in 0 blocks
==12077==    indirectly lost: 0 bytes in 0 blocks
==12077==    possibly lost: 0 bytes in 0 blocks
==12077==    still reachable: 800 bytes in 1 blocks
==12077==    suppressed: 0 bytes in 0 blocks
==12077== Rerun with --leak-check=full to see details of leaked memory
==12077==
==12077== For counts of detected and suppressed errors, rerun with: -v
==12077== Use --track-origins=yes to see where uninitialised values come from
==12077== ERROR SUMMARY: 35 errors from 7 contexts (suppressed: 0 from 0)
Segmentation fault
-bash-4.2$

```

Here is my valgrind run of node 27 with large.graph. Again, you can see like before that I had the same problem as before.

```
hermod.ics.uci.edu - PuTTY
-bash-4.2$ genGraph > rdm.graph
-bash-4.2$ valgrind dijkstras 0 rdm.graph
==17022== Memcheck, a memory error detector
==17022== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==17022== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==17022== Command: dijkstras 0 rdm.graph
==17022==
==17022== Conditional jump or move depends on uninitialised value(s)
==17022==   at 0x401FCE: PriorityQueue<Verts>::siftSmallestUp(Verts*, int, int) (priorityqueue.h:109)
==17022==   by 0x401ECD: PriorityQueue<Verts>::heapify(Verts*, int) (priorityqueue.h:98)
==17022==   by 0x401E32: PriorityQueue<Verts>::heapsort(Verts*, int) (priorityqueue.h:82)
==17022==   by 0x401D7A: PriorityQueue<Verts>::decreaseKey(int, int) (priorityqueue.h:136)
==17022==   by 0x4015AD: dijkstras(Graph&, int, Verts*) (main.cpp:41)
==17022==   by 0x401797: main (main.cpp:86)
==17022==
==17022== Conditional jump or move depends on uninitialised value(s)
==17022==   at 0x401D2F: PriorityQueue<Verts>::decreaseKey(int, int) (priorityqueue.h:127)
==17022==   by 0x4015AD: dijkstras(Graph&, int, Verts*) (main.cpp:41)
==17022==   by 0x401797: main (main.cpp:86)
==17022==
==17022== Conditional jump or move depends on uninitialised value(s)
==17022==   at 0x401FCE: PriorityQueue<Verts>::siftSmallestUp(Verts*, int, int) (priorityqueue.h:109)
==17022==   by 0x401E8B: PriorityQueue<Verts>::heapsort(Verts*, int) (priorityqueue.h:89)
==17022==   by 0x401D7A: PriorityQueue<Verts>::decreaseKey(int, int) (priorityqueue.h:136)
==17022==   by 0x4015AD: dijkstras(Graph&, int, Verts*) (main.cpp:41)
==17022==   by 0x401797: main (main.cpp:86)
==17022==
==17022== Use of uninitialised value of size 8
==17022==   at 0x401450: dijkstras(Graph&, int, Verts*) (main.cpp:34)
==17022==   by 0x401797: main (main.cpp:86)
==17022==
==17022== Use of uninitialised value of size 8
==17022==   at 0x4014F8: dijkstras(Graph&, int, Verts*) (main.cpp:37)
==17022==   by 0x401797: main (main.cpp:86)
==17022==
id 0, dist 0, prev -1
id 1, dist 3, prev 0
id 2, dist 14, prev 1
id 3, dist 4, prev 0
id 4, dist 4, prev 0
id 5, dist 12, prev 4
id 6, dist 22, prev 4
id 7, dist 10, prev 4
id 8, dist 14, prev 4
id 9, dist 32, prev 8
id 10, dist 21, prev 8
id 11, dist 19, prev 8
id 12, dist 20, prev 8
id 13, dist 10000000, prev -1
id 14, dist 10000000, prev -1
id 15, dist 10000000, prev -1
id 16, dist 10000000, prev -1
id 17, dist 10000000, prev -1
id 18, dist 10000000, prev -1
id 19, dist 10000000, prev -1
id 20, dist 10000000, prev -1
id 21, dist 10000000, prev -1
id 22, dist 10000000, prev -1
id 23, dist 10000000, prev -1
id 24, dist 10000000, prev -1
id 25, dist 10000000, prev -1
id 26, dist 10000000, prev -1
id 27, dist 10000000, prev -1
id 28, dist 10000000, prev -1
id 29, dist 10000000, prev -1
```

```

hermod.ics.uci.edu - PuTTY
id 1005, dist 10000000, prev -1
id 1006, dist 10000000, prev -1
id 1007, dist 10000000, prev -1
id 1008, dist 10000000, prev -1
id 1009, dist 10000000, prev -1
id 1010, dist 10000000, prev -1
id 1011, dist 10000000, prev -1
id 1012, dist 10000000, prev -1
id 1013, dist 10000000, prev -1
id 1014, dist 10000000, prev -1
id 1015, dist 10000000, prev -1
id 1016, dist 10000000, prev -1
id 1017, dist 10000000, prev -1
id 1018, dist 10000000, prev -1
id 1019, dist 10000000, prev -1
id 1020, dist 10000000, prev -1
id 1021, dist 10000000, prev -1
id 1022, dist 10000000, prev -1
id 1023, dist 10000000, prev -1
==17022== Invalid read of size 8
==17022==    at 0x4031D0: Graph::~~Graph() (graph.cpp:58)
==17022==    by 0x401893: main (main.cpp:80)
==17022== Address 0x5ab5f40 is 0 bytes after a block of size 8,192 alloc'd
==17022==    at 0x4C2A888: operator new[](unsigned long) (vg_replace_malloc.c:423)
==17022==    by 0x402E9F: Graph::Graph(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>) (graph.cpp:15)
==17022==    by 0x401741: main (main.cpp:80)
==17022== Invalid read of size 8
==17022==    at 0x4032A3: std::vector<Edge, std::allocator<Edge> >::~~vector() (stl_vector.h:567)
==17022==    by 0x40327F: Vertex::~Vertex() (vertex.h:9)
==17022==    by 0x4031DF: Graph::~~Graph() (graph.cpp:58)
==17022==    by 0x401893: main (main.cpp:80)
==17022== Address 0x2050 is not stack'd, malloc'd or (recently) free'd
==17022==
==17022== Process terminating with default action of signal 11 (SIGSEGV): dumping core
==17022== Access not within mapped region at address 0x2050
==17022==    at 0x4032A3: std::vector<Edge, std::allocator<Edge> >::~~vector() (stl_vector.h:567)
==17022==    by 0x40327F: Vertex::~Vertex() (vertex.h:9)
==17022==    by 0x4031DF: Graph::~~Graph() (graph.cpp:58)
==17022==    by 0x401893: main (main.cpp:80)
==17022== If you believe this happened as a result of a stack
==17022== overflow in your program's main thread (unlikely but
==17022== possible), you can try to increase the size of the
==17022== main thread stack using the --main-stacksize= flag.
==17022== The main thread stack size used in this run was 8388608.
==17022==
==17022== HEAP SUMMARY:
==17022==    in use at exit: 8,192 bytes in 1 blocks
==17022== total heap usage: 8,198 allocs, 8,197 frees, 282,180 bytes allocated
==17022==
==17022== LEAK SUMMARY:
==17022==    definitely lost: 0 bytes in 0 blocks
==17022==    indirectly lost: 0 bytes in 0 blocks
==17022==    possibly lost: 0 bytes in 0 blocks
==17022==    still reachable: 8,192 bytes in 1 blocks
==17022==    suppressed: 0 bytes in 0 blocks
==17022== Rerun with --leak-check=full to see details of leaked memory
==17022==
==17022== For counts of detected and suppressed errors, rerun with: -v
==17022== Use --track-origins=yes to see where uninitialised values come from
==17022== ERROR SUMMARY: 50 errors from 7 contexts (suppressed: 0 from 0)
Segmentation fault
-bash-4.2$

```

Here is my valgrind with node 0 using rdm.graph. Same Problem.

If I had more time and some assistance with this assignment I would keep working at it. But I have to study for this class's final tomorrow.

End of Report.