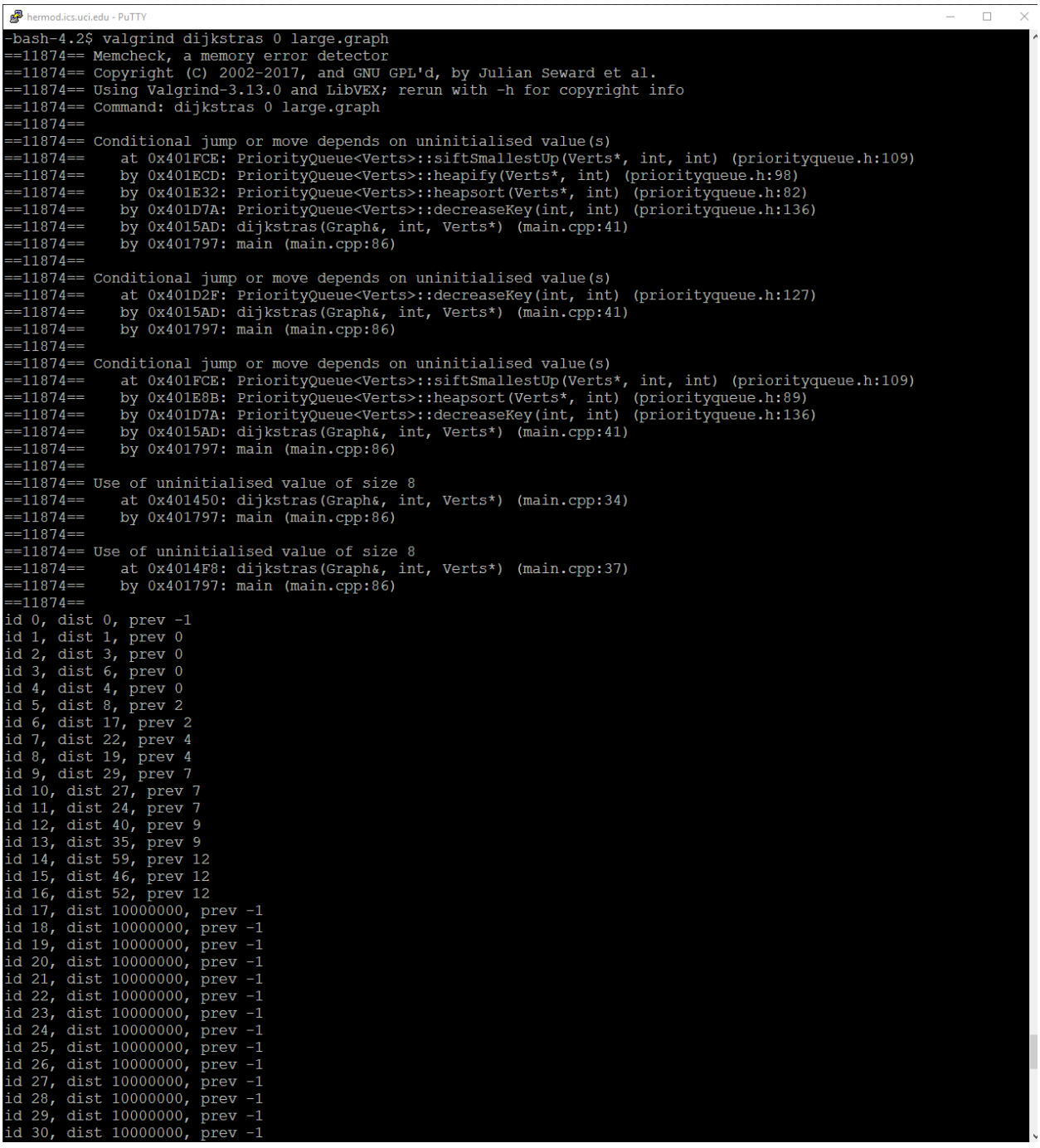
Homework 9

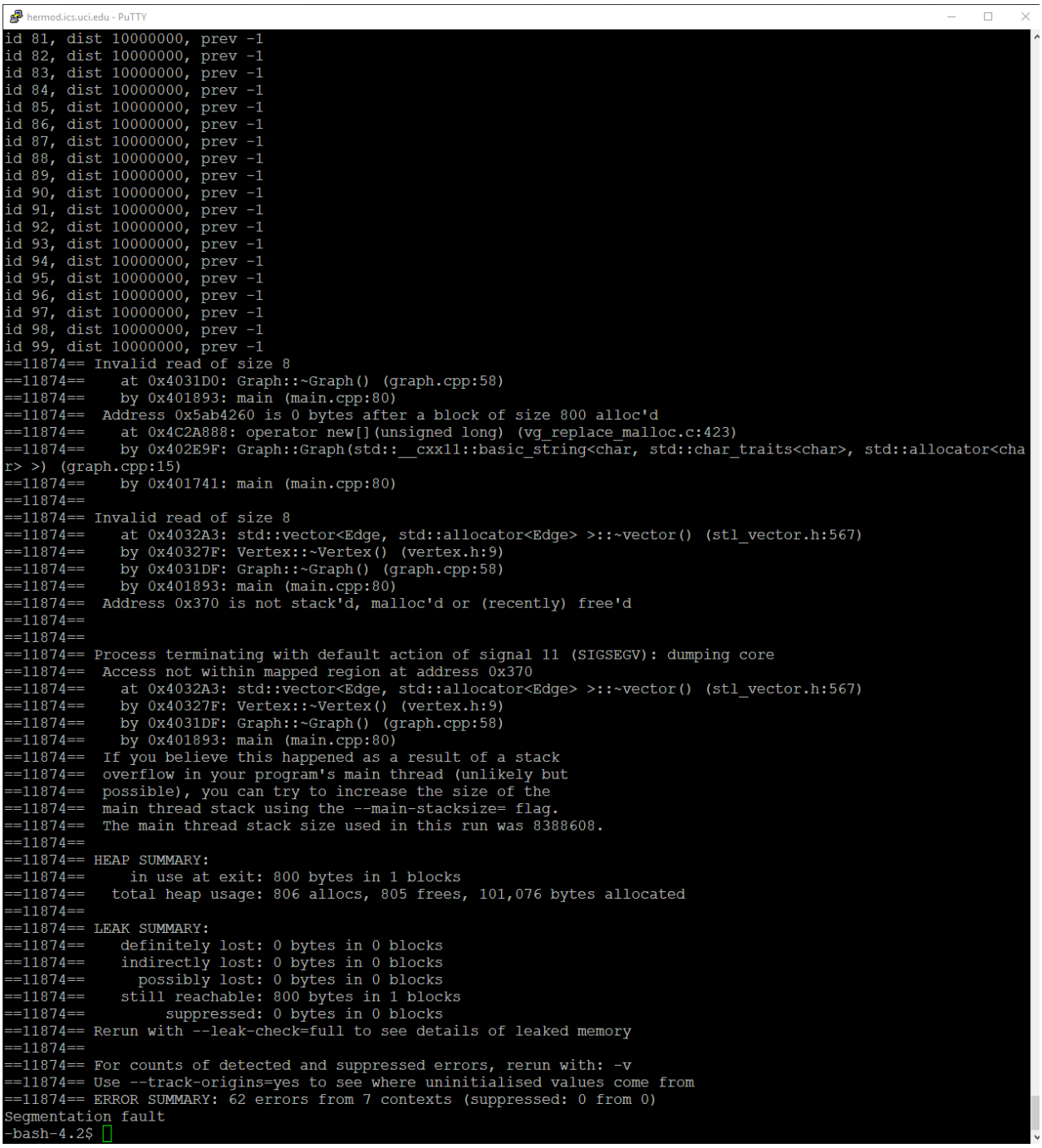
Start of Report

hermod.ics.uci.edu - PuTTY

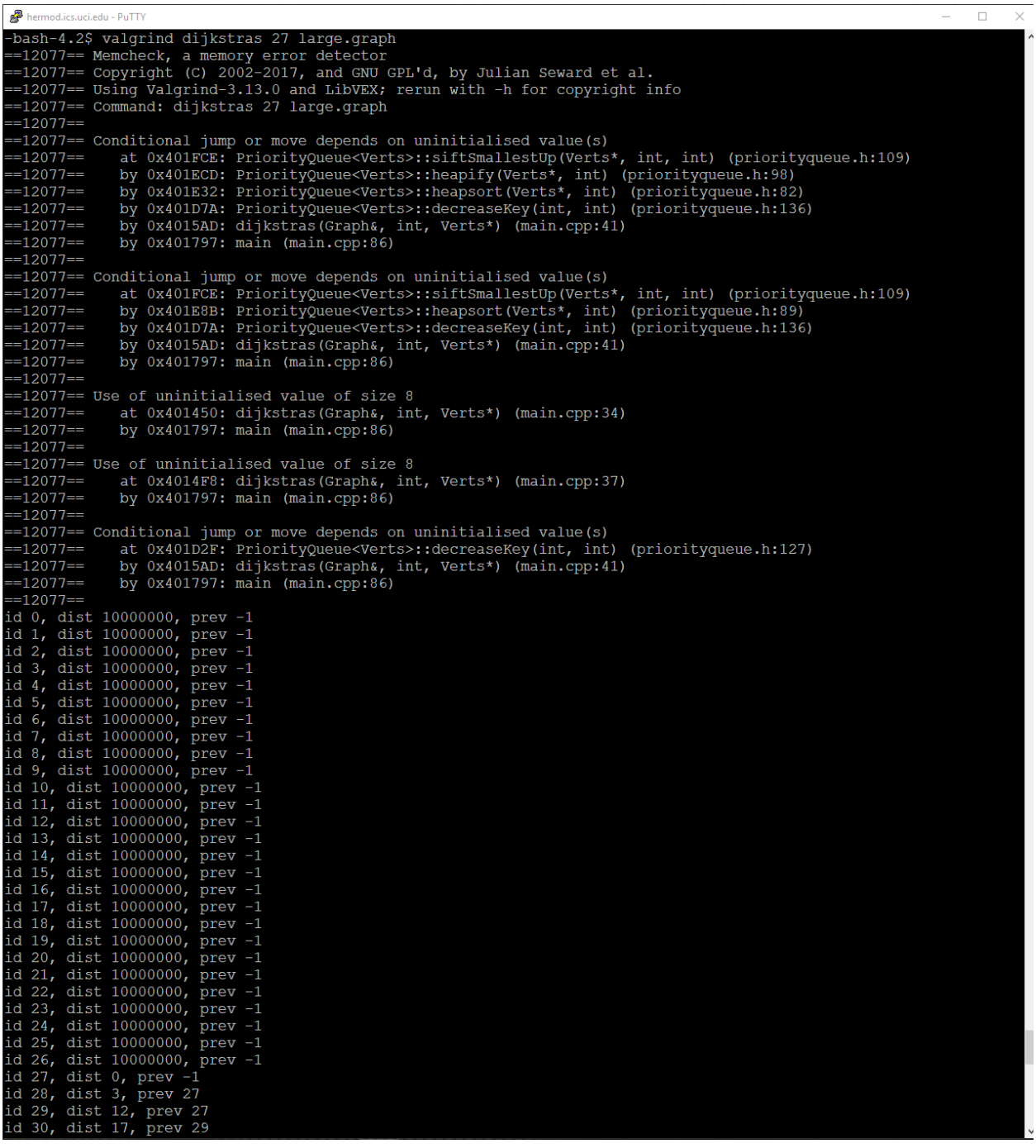
```
using namespace std;
#include <iostream>
#include <string>
#include <fstream>
#include <limits>
#include "graph.h"
#include "priorityqueue.h"


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 );
                        }
                }
        }
}
```

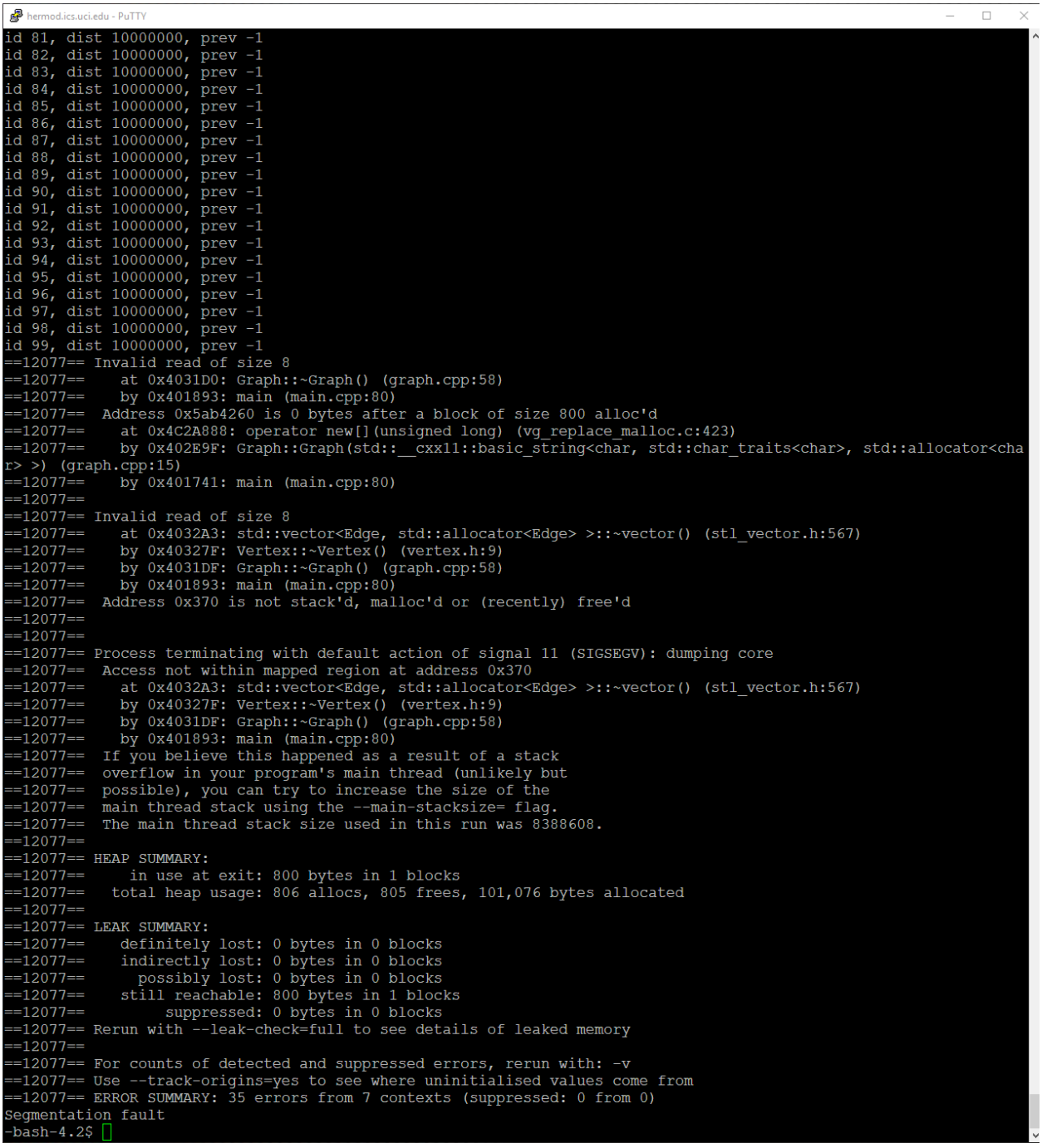
So here is my code for dijkstras with its time complexity. Now there is something wrong with this function because I am not getting the results that I want and I am getting memory leaks. I am not sure if it has to do with my priorityqueue class or my other graph class since I had to make changes to it.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

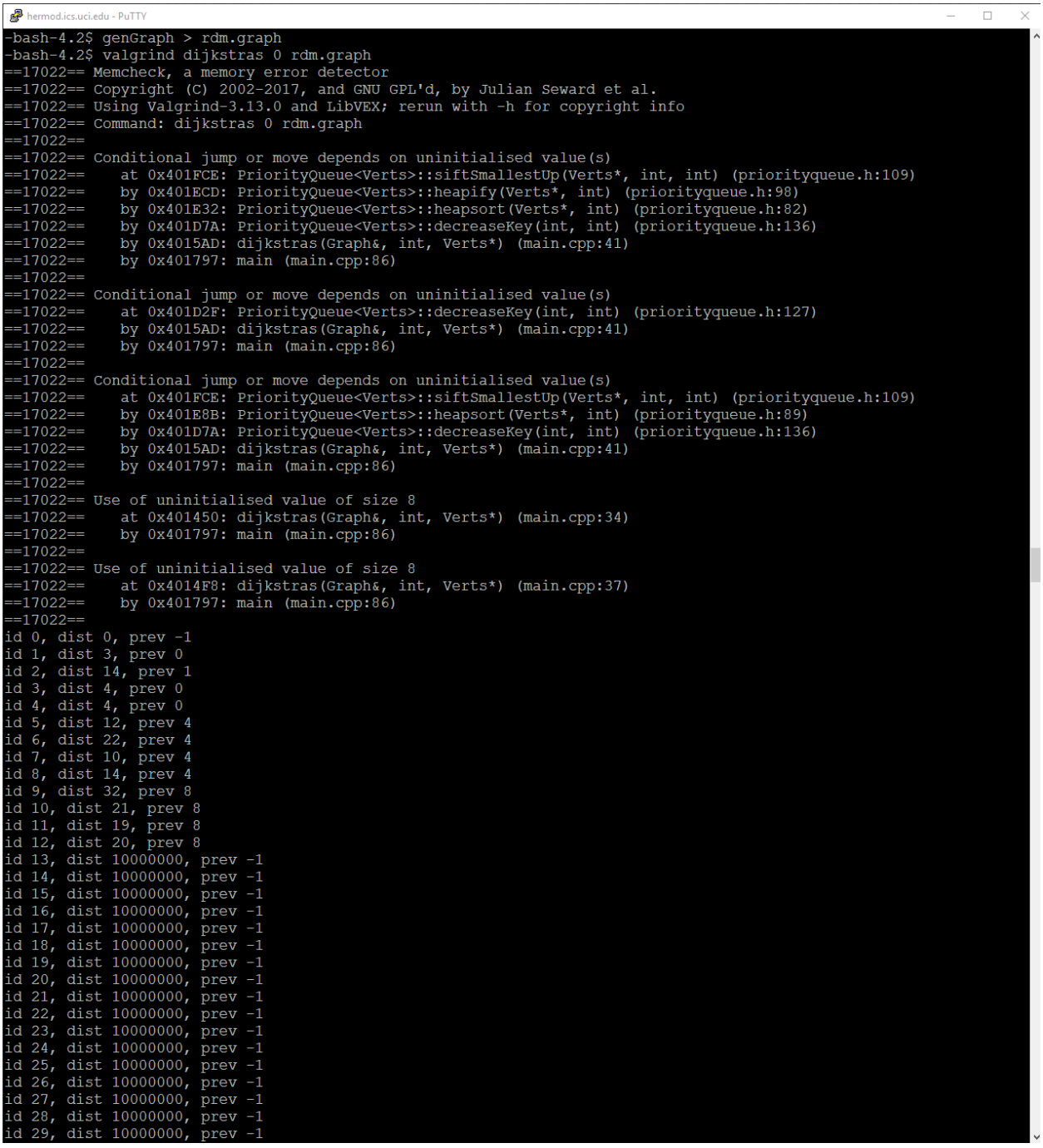


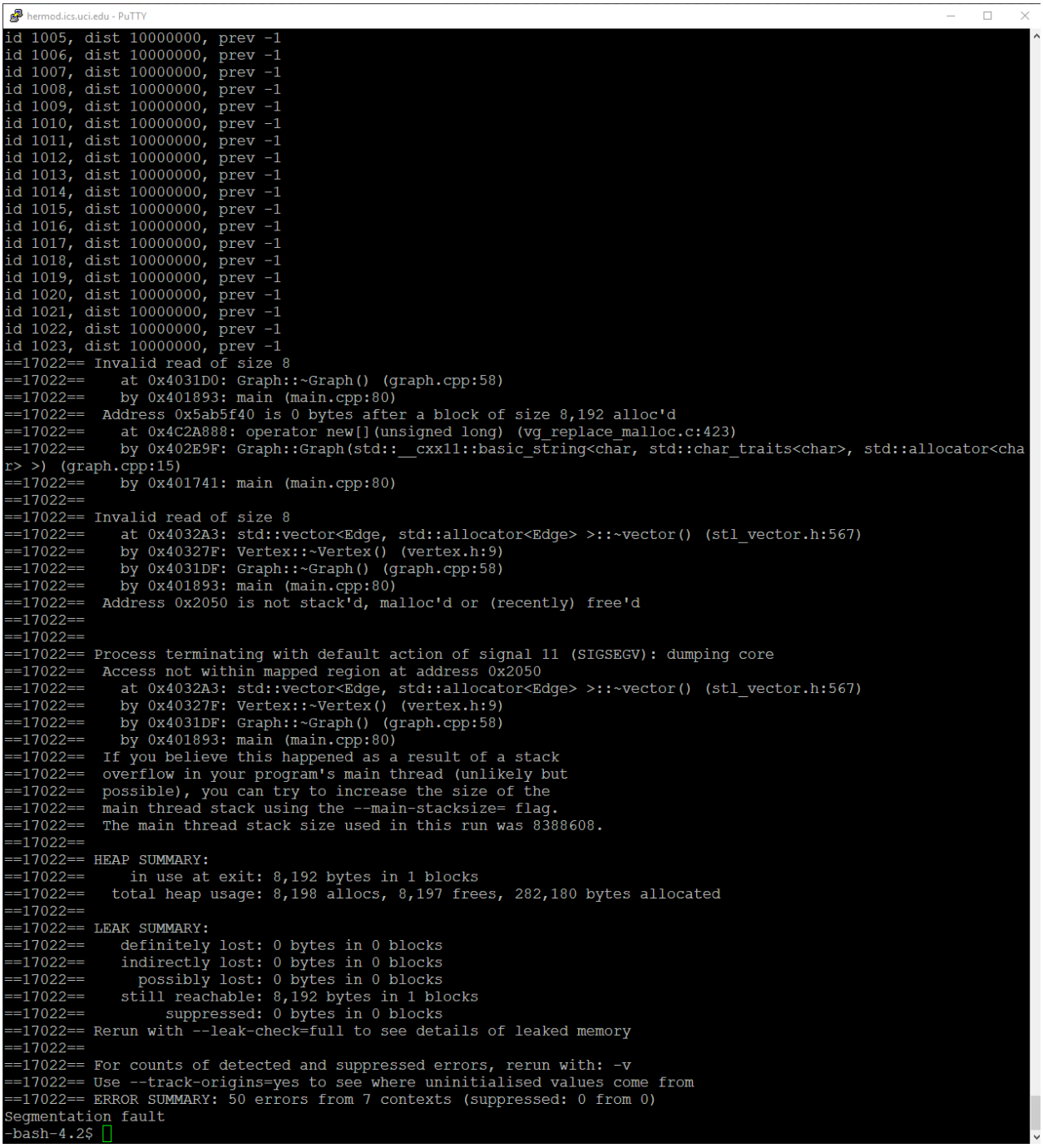
Here is my valgrind with source 0 ran with large.graph. I didn't know how to get the result I wanted and even then I was still confused on how I was supposed to produced the desired output if I didn't know how the information was obtained.


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



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



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.