Raul Cervantes

77825705

HW8: Start of Report

```
vector<Edge> Kruskals(const Graph& g)
//Big-Oh = O(N)
{
        Set s(g.num_edge);
        for(int i = 0; i < g.num_vertex; ++i)
        {
                s.MakeSet(g.vertices[i].id);
        }
        unsigned int N = g.num_vertex;
        priority_queue <Edge, vector<Edge>, MyCompare> Q;
        for(auto e: g.vertices->edges)
        {
                Q.push(e);
        }
        vector<Edge> T;
        while(T.size() < N-1)
        {
                Edge e = Q.top();
                Q.pop();
                if(s.Find(e.src) != s.Find(e.dst))
                {
                        T.push_back(e);
                        s.Union(e.src, e.dst);
                }
        }
        return T;
}
```

```
class MyCompare
{
public:
        template<typename T>
        bool operator()(T a, T b)
        //Big-Oh = O(1)
        {
                return a.weight >  b.weight;
        }
};
```

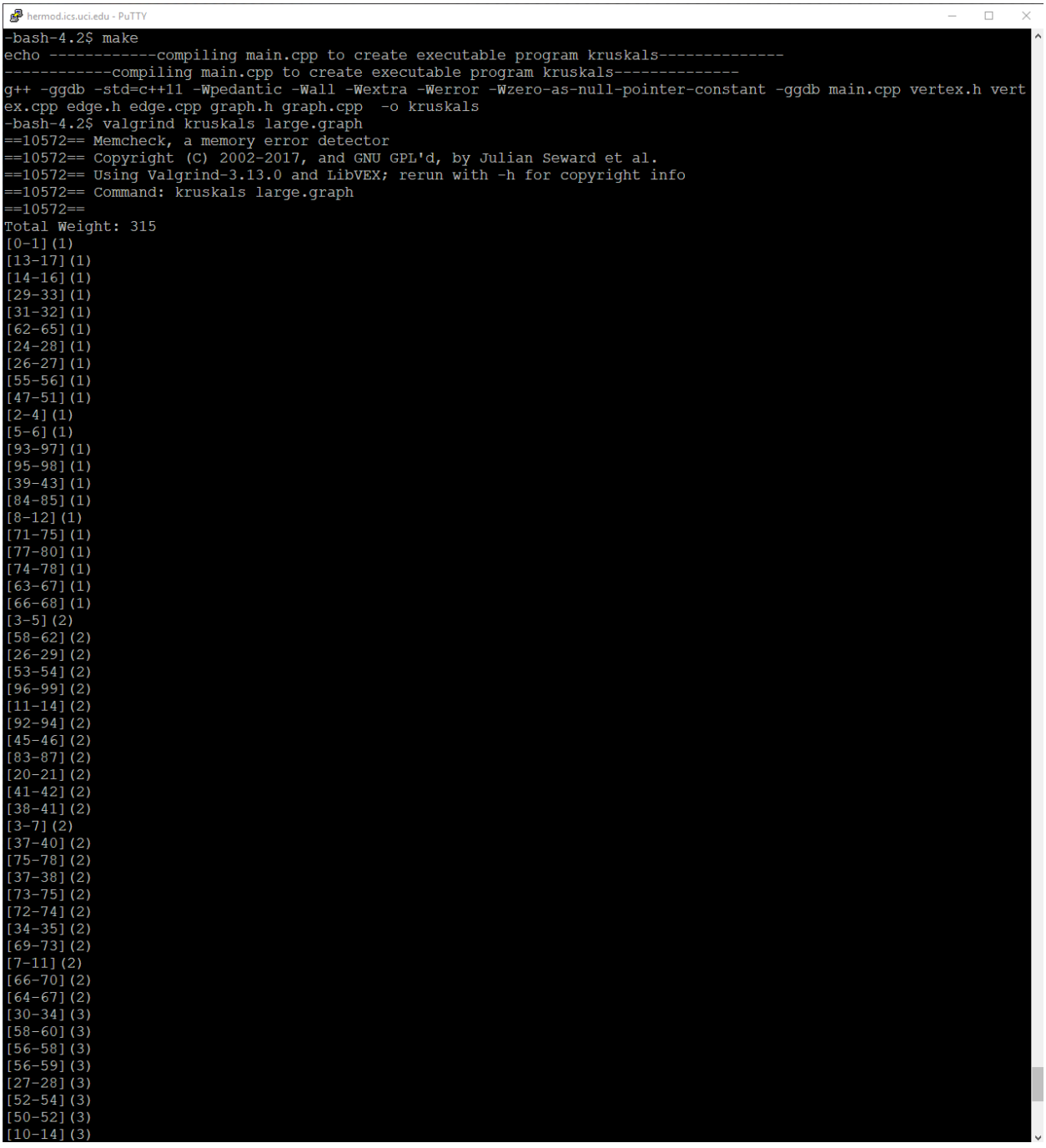
hermod.ics.uci.edu - PuTTY

```
struct Set
{
        int count;
        SetNode* elements;

        Set(int N): count(N), elements(new SetNode[N])
        {
        }

        void error(const char* msg)
        //Big-Oh = O(1)
        {
                cerr<< "ERROR: " << msg <<endl;
                exit(-1);
        }

        void verify_set_id(int x)
        //Big-Oh = O(1)
        {
                if(x<0 || x>=count)
                {
                        error("ID out of bounds");
                }
        }

        void MakeSet(int x)
        //Big-Oh = O(1)
        {
                verify_set_id(x);
                SetNode& my_node = elements[x];

                my_node.parent = &my_node;
                my_node.rank = 0;
                my_node.size = 1;
        }

        SetNode* Find(SetNode* x)
        //Big-Oh = O(1)
        {
                if(x->parent != x)
                {
                        x->parent = Find(x->parent);
                }
                return x->parent;
        }

        SetNode* Find(int x)
        //Big-Oh = O(1)
        {
                verify_set_id(x);
                SetNode& my_node = elements[x];
                return Find(&my_node);
        }

        void Union(int x, int y)
        //Big-Oh = O(1)
        {
                SetNode* xRoot = Find(x);
                SetNode* yRoot = Find(y);
                if(xRoot == yRoot)
                {
                        return;
                }
       if(xRoot->size < yRoot->size)
                                                                    80,1-8        10%
```

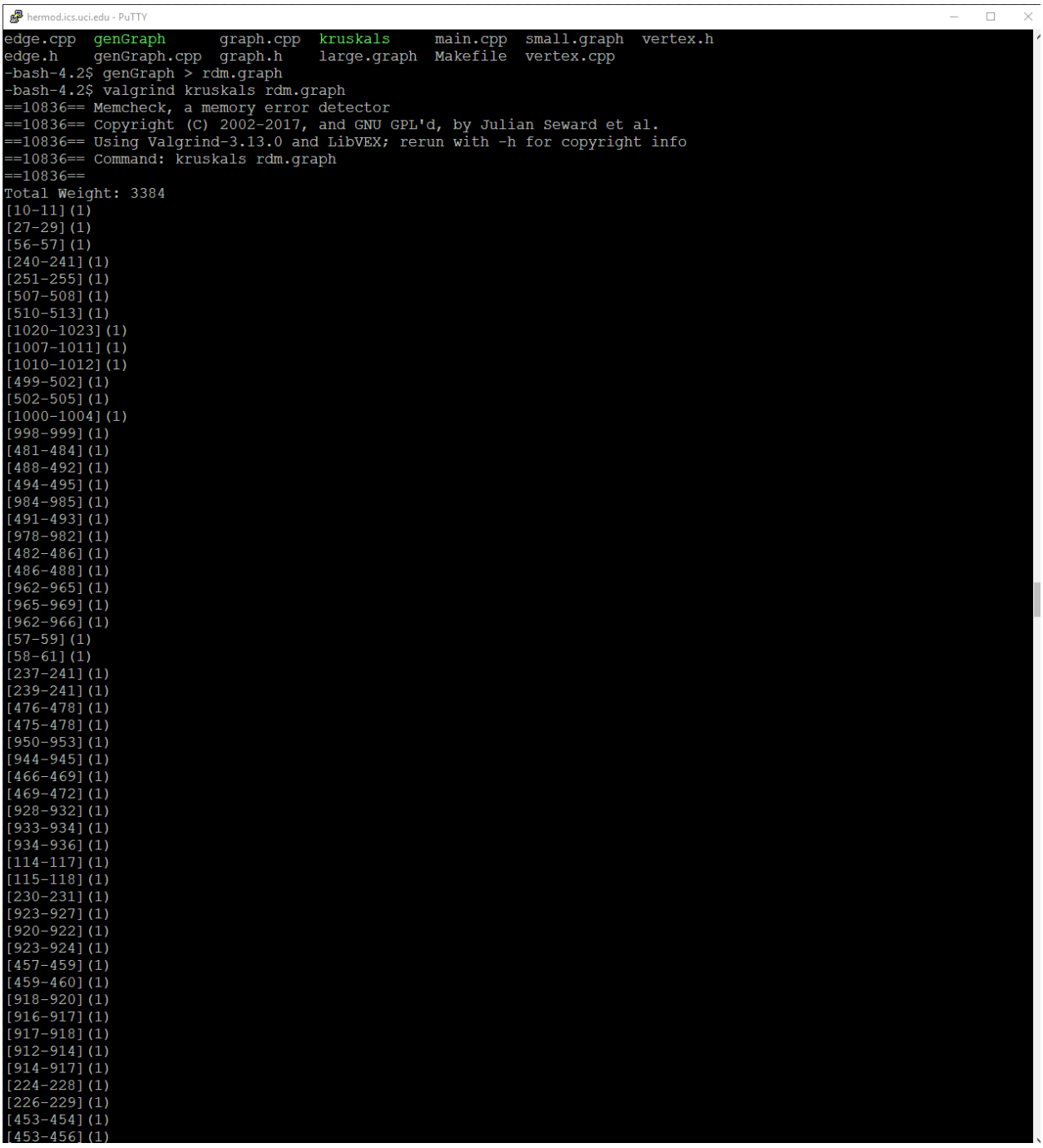
hermod.ics.uci.edu - PuTTY

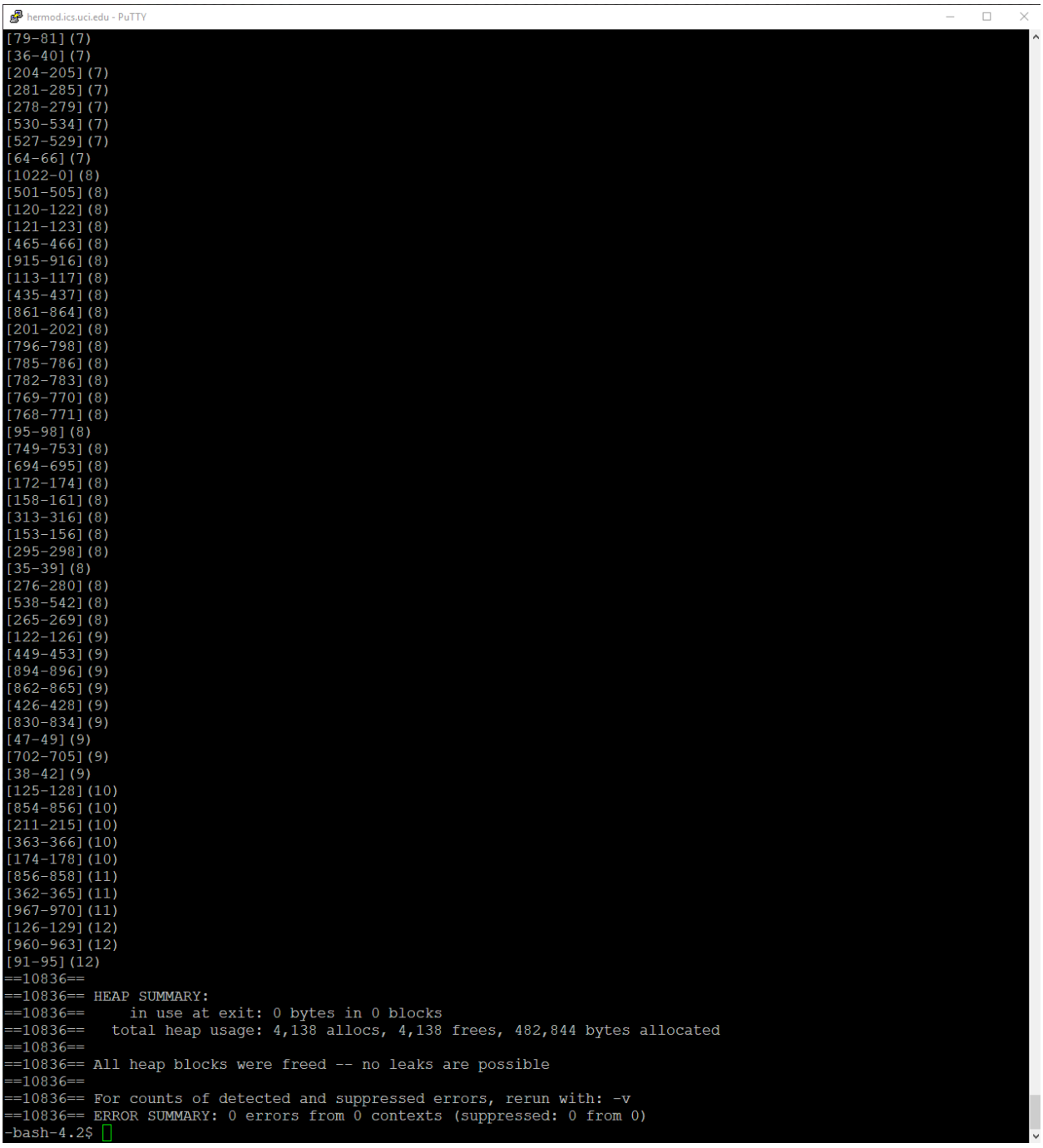
```
        if(xRoot->size < yRoot->size)
                {
                        swap(xRoot, yRoot);
                }
                yRoot->parent = xRoot;
                xRoot->size = xRoot->size + yRoot->size;
        }

        ~Set()
        {
                delete[] elements;
        }
};
```

Above is my kruskals function and the set struct with its Big-Oh Notation time complexities.



```
[90-91](3)
[90-93](3)
[0-2](3)
[19-20](3)
[37-39](3)
[67-69](3)
[59-63](4)
[49-53](4)
[94-96](4)
[91-95](4)
[44-45](4)
[42-46](4)
[84-87](4)
[76-78](4)
[2-3](4)
[67-70](4)
[15-17](5)
[28-32](5)
[29-30](5)
[54-58](5)
[54-57](5)
[51-55](5)
[12-13](5)
[23-26](5)
[93-96](5)
[46-49](5)
[44-48](5)
[88-91](5)
[87-89](5)
[81-85](5)
[80-84](5)
[80-82](5)
[78-80](5)
[18-19](5)
[35-36](5)
[16-17](5)
[97-0](6)
[88-89](6)
[86-88](6)
[9-13](6)
[75-79](6)
[17-21](6)
[61-63](7)
[21-25](7)
[34-37](7)
[22-26](8)
==10572==
==10572== HEAP SUMMARY:
==10572==     in use at exit: 0 bytes in 0 blocks
==10572==   total heap usage: 433 allocs, 433 frees, 126,684 bytes allocated
==10572==
==10572== All heap blocks were freed -- no leaks are possible
==10572==
==10572== For counts of detected and suppressed errors, rerun with: -v
==10572== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
-bash-4.2$
```


hermod.ics.uci.edu - PuTTY
edge.cpp genGraph graph.cpp kruskals main.cpp small.graph vertex.h
edge.h genGraph.cpp graph.h large.graph Makefile vertex.cpp
-bash-4.2$ genGraph > rdm.graph
-bash-4.2$ valgrind kruskals rdm.graph
==10836== Memcheck, a memory error detector
==10836== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==10836== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==10836== Command: kruskals rdm.graph
==10836==
Total Weight: 3384
[10-11](1)
[27-29](1)
[56-57](1)
[240-241](1)
[251-255](1)
[507-508](1)
[510-513](1)
[1020-1023](1)
[1007-1011](1)
[1010-1012](1)
[499-502](1)
[502-505](1)
[1000-1004](1)
[998-999](1)
[481-484](1)
[488-492](1)
[494-495](1)
[984-985](1)
[491-493](1)
[978-982](1)
[482-486](1)
[486-488](1)
[962-965](1)
[965-969](1)
[962-966](1)
[57-59](1)
[58-61](1)
[237-241](1)
[239-241](1)
[476-478](1)
[475-478](1)
[950-953](1)
[944-945](1)
[466-469](1)
[469-472](1)
[928-932](1)
[933-934](1)
[934-936](1)
[114-117](1)
[115-118](1)
[230-231](1)
[923-927](1)
[920-922](1)
[923-924](1)
[457-459](1)
[459-460](1)
[918-920](1)
[916-917](1)
[917-918](1)
[912-914](1)
[914-917](1)
[224-228](1)
[226-229](1)
[453-454](1)
[453-456](1)



Above is my valgrind for my large.graph and rdm.graph along with the execution and making of the rdm.graph.

End of Report.