

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
        {
            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.

```
hermod.ics.uci.edu - PuTTY
-bash-4.2$ make
echo -----compiling main.cpp to create executable program kruskals-----
-----compiling main.cpp to create executable program kruskals-----
g++ -ggdb -std=c++11 -Wpedantic -Wall -Wextra -Werror -Wzero-as-null-pointer-constant -ggdb main.cpp vertex.h vert
ex.cpp edge.h edge.cpp graph.h graph.cpp -o kruskals
-bash-4.2$ valgrind kruskals large.graph
==10572== Memcheck, a memory error detector
==10572== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==10572== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==10572== Command: kruskals large.graph
==10572==
Total Weight: 315
[0-1] (1)
[13-17] (1)
[14-16] (1)
[29-33] (1)
[31-32] (1)
[62-65] (1)
[24-28] (1)
[26-27] (1)
[55-56] (1)
[47-51] (1)
[2-4] (1)
[5-6] (1)
[93-97] (1)
[95-98] (1)
[39-43] (1)
[84-85] (1)
[8-12] (1)
[71-75] (1)
[77-80] (1)
[74-78] (1)
[63-67] (1)
[66-68] (1)
[3-5] (2)
[58-62] (2)
[26-29] (2)
[53-54] (2)
[96-99] (2)
[11-14] (2)
[92-94] (2)
[45-46] (2)
[83-87] (2)
[20-21] (2)
[41-42] (2)
[38-41] (2)
[3-7] (2)
[37-40] (2)
[75-78] (2)
[37-38] (2)
[73-75] (2)
[72-74] (2)
[34-35] (2)
[69-73] (2)
[7-11] (2)
[66-70] (2)
[64-67] (2)
[30-34] (3)
[58-60] (3)
[56-58] (3)
[56-59] (3)
[27-28] (3)
[52-54] (3)
[50-52] (3)
[10-14] (3)
```

```
[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)
```

```
hermod.ics.uci.edu - PuTTY
[79-81] (7)
[36-40] (7)
[204-205] (7)
[281-285] (7)
[278-279] (7)
[530-534] (7)
[527-529] (7)
[64-66] (7)
[1022-0] (8)
[501-505] (8)
[120-122] (8)
[121-123] (8)
[465-466] (8)
[915-916] (8)
[113-117] (8)
[435-437] (8)
[861-864] (8)
[201-202] (8)
[796-798] (8)
[785-786] (8)
[782-783] (8)
[769-770] (8)
[768-771] (8)
[95-98] (8)
[749-753] (8)
[694-695] (8)
[172-174] (8)
[158-161] (8)
[313-316] (8)
[153-156] (8)
[295-298] (8)
[35-39] (8)
[276-280] (8)
[538-542] (8)
[265-269] (8)
[122-126] (9)
[449-453] (9)
[894-896] (9)
[862-865] (9)
[426-428] (9)
[830-834] (9)
[47-49] (9)
[702-705] (9)
[38-42] (9)
[125-128] (10)
[854-856] (10)
[211-215] (10)
[363-366] (10)
[174-178] (10)
[856-858] (11)
[362-365] (11)
[967-970] (11)
[126-129] (12)
[960-963] (12)
[91-95] (12)
==10836==
==10836== HEAP SUMMARY:
==10836==      in use at exit: 0 bytes in 0 blocks
==10836==    total heap usage: 4,138 allocs, 4,138 frees, 482,844 bytes allocated
==10836==
==10836== All heap blocks were freed -- no leaks are possible
==10836==
==10836== For counts of detected and suppressed errors, rerun with: -v
==10836== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
-bash-4.2$
```

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

End of Report.