Running time of Kruskal's algorithm

Ask Question

The Kruskal's algorithm is the following:

```
MST-KRUSKAL(G,w)
1. A={}
2. for each vertex v∈ G.V
3.      MAKE-SET(v)
4. sort the edges of G.E into nondecreasing order by weight w
5. for each edge (u,v) ∈ G.E, taken in nondecreasing order by weight w
6.      if FIND-SET(u)!=FIND-SET(v)
7.         A=A U {(u,v)}
8.         Union(u,v)
9. return A
```

According to my textbook:

> Initializing the set A in line 1 takes O(1) time, and the time to sort the edges in line 4 is O(E lgE). The for loop of lines 5-8 performs O(E) FIND-SET and UNION operations on the disjoint-set forest. Along with the |V| MAKE-SET operations, these take a total of O((V+E)α(V)) time, where α is a very slowly growing function. Because we assume that G is connected, we have |E| <= |V|-1, and so the disjoint-set operations take O(E α(V)) time. Moreover, since α(V)=O(lgV)=O(lgE), the total running time of Kruskal's algorithm is O(E lgE). Observing that |E|<|V|^2, we have lg |E|=O(lgV), and so we can restate the running time of Kruskal's algorithm as O(E lgV).

Could you explain me why we deduce that the time to sort the edges in line 4 is O(E lgE)? Also how do we get that the total time complexity is O((V+E)α(V)) ?

In addition, suppose that all edge weights in a graph are integers from 1 to |V|. How fast can you make

range from 1 to W for some constant W?

How does the time complexity depend on the weight of the edges?

EDIT:

> In addition, suppose that all edge weights in a graph are integers from 1 to |V|. How fast can you make Kruskal's algorithm run?

I have thought the following:

In order the Kruskal's algorithm to run faster, we can sort the edges applying Counting Sort.

- The line 1 requires O(1) time.
- The lines 2-3 require O(v) time.
- The line 4 requires O(|V|+|E|) time.
- The lines 5-8 require O(|E|α(|V|)) time.
- The line 9 requires O(1) time.

So if we use Counting Sort in order to solve the edges, the time complexity of Kruskal will be

$$O(1) + O(|V|) + O(|E|) + O(|E|\alpha(|V|)) + O(1)$$
$$= O(|V|) + |E| + |E|\alpha(|V|))$$
$$\text{and since } \alpha(|V|) = O(\lg|V|), \text{ we have that :}$$

$$O(|V| + |E| + |E|\alpha(|V|))$$
$$= O(|V| + |E| + |E|\lg(|V|))$$
$$= O(|V| + (\lg(|V|) + 1)|E|)$$
$$= O(|V| + |E|\lg(|V|))$$

Could you tell me if my idea is right?

Also:

> What if the edges weights are integers in the range from 1 to W for some constant W?

We will again use Counting Sort. The algorithm will be the same. We find the time complexity as follows:

- The line 1 requires O(1) time.

|E|)=O(|E|) time.

- The lines 5-8 require O(|E|α(|V|)) time.
- The line 9 requires O(1) time.

So the time complexity will be:

$$O(1) + O(|V|) + O(W + |E|) + O(|E|\alpha(|V|)) + O(1)$$

$$= O(1) + O(|V|) + O(W) + O(|E|) + O(|E|\alpha(|V|)) + O(1)$$

$$= O(1) + O(|V|) + O(1) + O(|E|) + O(|E|\alpha(|V|)) + O(1)$$

$$= O(|V| + |E| + |E|\alpha(|V|))$$
and since $\alpha(|V|) = O(\lg|V|)$, we have that :

$$O(|V|+|E|+|E|\alpha(|V|)) = O(|V|+|E|+|E|\lg|V|) = O(|V|+(1+\lg|V|)|E|) = O(|V| + |E|\lg|V|)$$

performance time graph kruskals-algorithm

edited Nov 22 '17 at 20:28

Mooncrater
845 10 23

asked Apr 10 '15 at 12:14

Mary Star
141 2 19

1 Answer

> Could you explain me why we deduce that the time to sort the edges in line 4 is O(E*lgE)?

To sort a set of N items we use O(N*lg(N)) algorithm, which is quick sort, merge sort or heap sort. To sort E edges we therefore need O(E*lg(E)) time. This however is not necessary in some cases, as we could use sorting algorithm with better complexity (read further).

> Also how do we get that the total time complexity is O((V+E)α(V))?

I don't think total complexity is O((V+E)α(V)). That would be complexity of the 5-8 loop. O((V+E)α(V)) complexity comes from

data structure in some algorithmic book.

> How fast can you make Kruskal's algorithm run?

For first part, line 4, we have O(E*lg(E)) complexity and for second part, line 5-8, we have O((E+V)*α(V)) complexity. This two summed up yield O(E*lg(E)) complexity. If we use O(N*lg(N)) sort this can't be improved.

> What if the edges weights are integers in the range from 1 to W for some constant W?

If that is the case, than we could use counting sort for first part. Giving line 4 complexity of O(E+W) = O(E). In that case algorithm would have O((E+V)*α(V)) total complexity. Note that however O(E + W) in reality includes a constant that could be rather large and might be impractical for large W.

> How does the time complexity depend on the weight of the edges?

As said, if weight of the edges is small enough we can use counting sort and speed up the algorithm.

> EDIT:
>
> In addition, suppose that all edge weights in a graph are integers from 1 to |V|. How fast can you make Kruskal's algorithm run? I have thought the following:
>
> In order the Kruskal's algorithm to run faster, we can sort the edges applying Counting Sort.
>
> The line 1 requires O(1) time. The lines 2-3 require O(vα(|V|)) time. The line 4 requires O(|V|+|E|)

Your idea is correct, however you can make bounds smaller.

The lines 2-3 requires O(|V|) rather than O(|V|α(|V|)). We however simplified it to O(|V|α(|V|)) in previous calculations to make calculations easier.

With this you get the time of: O(1) + O(|V|) + O(|V| + |E|) + O(|E|α(|V|)) + O(1) = O(|V| + |E|) + O(|E|α(|V|))

You can simplify this to either O((|V| + |E|) * α(|V|) or to O(|V| + |E|*α(|V|).

So while you were correct, since O((|V| + |E|) * α(|V|) < O((|V| + |E|) * lg(|E|)

Calculations for the |W| are analogous.

edited Apr 14 '15 at 21:05

answered Apr 10 '15 at 12:45

Riko
1,452 1 12 29

1 Riko You said that "O((V+E)α(V)) would be the complexity of the 5-8 loop. O((V+E)α(V)) complexity comes from V MAKE-SET operations and E Union operations." But at this for-loop we don't have MAKE-SET operations. So do we also take into consideration the other for-loop? If so, why could we do so although the number of times that the two for-loops are executed isn't the same? Also you said that for the line 4 we have O(E*lgE) complexity and for line 5-8 we have O((E+V)α(V)) complexity. So don't we take into consideration the first for-loop? And how if we sum these two up, do we get O(E*lgE) ? – Mary Star Apr 10 '15 at 13:21

1 Yes, my bad, you also need to count first loop. Lines 2-3 and 5-8 together would than yield O((V+E)α(V)) complexity. If we sum O(ElgE) and O((E+V)α(V)) we get

1 N represents number of elements that we sort. It doesn't matter in what range they are, but that range should be reasonable small. To understand counting sort better read about it here: en.wikipedia.org/wiki/Counting_sort – Riko Apr 10 '15 at 13:58

1 Riko So is the time complexity of the lines 2-3 equal to O(Vα(V)) ? Also is it like that? O(ElgE)+O((E+V)α(V))=O(ElgE)+O((V+E)*lgE), since α(V)=O(lgE) and O(ElgE)+O((V+E)*lgE)=O(E*lgE+V*lgE+E*lgE)=O((V+E)*lgE). Then we know that |E|<=|V|-1, because we assume that G is connected. So isn't it as follows? (V+E)*lgE<=(2V-1)*lgE=O(V*lgE) or am I wrong? How do we deduce that O(ElgE)+O((E+V)α(V))=O(E*lgE) ? – Mary Star Apr 10 '15 at 14:05

1 Yes and yes, your calculations are correct to the point where you assume |E| <= |V|-1. If the graph is connected we know that number of edges is at least |V| - 1. Making that equation |E| >= |V|-1. If you change that you will get O(Elg(E)) – Riko Apr 10 '15 at 15:10