Journal of Computer and Communications, 2017, 5, 63-69
http://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Research and Improvement of Kruskal Algorithm

Haiming Li, Qiyang Xia, Yong Wang

School of Computer and Information Engineering, Shanghai University of Electric Power, Shanghai, China
Email: zjlhm@189.cn

How to cite this paper: Li, H.M., Xia, Q.Y. and Wang, Y. (2017) Research and Improvement of Kruskal Algorithm. *Journal of Computer and Communications*, 5, 63-69.
https://doi.org/10.4236/jcc.2017.512007

Received: October 10, 2017
Accepted: October 27, 2017
Published: October 30, 2017

Abstract

It's a very popular issue regarding the minimum cost spanning tree which is of great practical and economical significance to solve it in a concise and accelerated way. In this paper, the basic ideas of Kruskal algorithm were discussed and then presented a new improved algorithm—two branch Kruskal algorithm, which is improved to choose a middle value. Finally, because the time complexity is reduced, and the process is more convenient, it is concluded that the improved Kruskal algorithm is more effective in most cases compared with the Kruskal algorithm.

Keywords

Minimum Spanning Tree, Classical Kruskal Algorithm, Two Branch Kruskal Algorithm, Time Complexity

1. Introduction

How to select the best path in many ways that cost the least is the problem what people often encounter. For example, one wants to open a communication network or a pipeline, and how to design it to take it at least? The problem like this can be attributed to the problem of minimum cost spanning tree. It was an important application of graphs in data structure. To obtain a spanning tree was to choose n − 1 edges from a cost undirected graph and this graph was still connected, and at the same time considered the minimum cost of the tree. Prim algorithm and Kruskal algorithm were classics in algorithms of minimum cost spanning tree [1] [2] [3] [4]. The time complexity of every algorithm was different. Is there a simpler way to find the minimum cost spanning tree? That's the main content that we're going to talk about.

Kruskal's algorithm is a minimum-spanning-tree algorithm which finds an edge of the least possible weight that connects any two trees in the forest. It is a greedy algorithm in graph theory as it finds a minimum spanning tree for a connected weighted graph adding increasing cost arcs at each step. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. If the graph is not connected, then it finds a minimum spanning forest (a minimum spanning tree for each connected component).

This algorithm first appeared in Proceedings of the American Mathematical Society, pp. 48-50 in 1956, and was written by Joseph Kruskal. Its basic idea was to choose n − 1 edges one time, and then used the greed technique—choosing one edge with minimum cost that cannot form cycle to join the set which has been selected [1]. If the selected edge can be cycled, then it will not form a spanning tree [5] [6]. This algorithm is divided into x steps, where x is the total number of edges of the tree, judging only an edge according to the cost increasing order of these × edges at a time. When people considered an edge and added it into the selected edges, if the cycle appeared, then abandon it. Otherwise, it would be selected for collection.

Other algorithms for this problem include Prim's algorithm, Reverse-delete algorithm, and Borůvka's algorithm.

2. Deep Discussion of the Kruskal Algorithm

2.1. The Basic Ideas of the Kruskal Algorithm

Suppose that a connected graph with n vertexes G = {V, E}, non-connected graph without edges T = {V, φ}. Each vertex in graph G is a connected component, increase the cost of all the edges in graph G, and remove the edge with the smallest cost. If the two vertexes of the edge fall on different two connected components, one can select it into T; otherwise, give up it, and choose a minimum cost edge from the remaining edges again. Repeat in this case, the connected components are merged step by step until all vertexes are on the same connected component. M-1 edges were chosen for T, so it is the minimum cost spanning tree with M-1 edges of G. [4] [7].

Some functions of the Kruskal algorithm can be implemented by using some efficient algorithms, such as the minimum heap and parallel query algorithm.

2.2. The Procedure of the Kruskal Algorithm

```
//Kruskal algorithm for constructing minimum cost spanning tree
//Input: weighted connected graph G=<V, E>
//Output: Et, edge collection from minimum cost spanning tree composed of G
Ranking E in the order of non decreasing order of edge cost W (E)
Et←Ø; encounter←0
//Initialize the vertex set of the tree and the size of the collection
k←0
```

```
//Initialize the number of initialized edges
while encounter<|V|-1 do
    k←k+1
    if Et∪{ei}Non loop
        Et←Et{ei}; encounter←encounter+1
Return Et [6].
```

3. Improved Algorithm

3.1. The Definition of the Two Branch Kruskal Algorithm

It is necessary to analyze the classic Kruskal algorithm's time complexity in the process of solving the minimum cost spanning tree. In the process of solving, it needed to detect the adjacency matrix when establishing the minimum heap, and the calculated time efficiency was $O(n^2)$. At the same time did heap insert operation for x times, and each insert operation needed call stack algorithm, therefore, heap insert operation time efficiency is O (|E|log2|E|). It needed to call stack the output heap operation for x in the process of construct minimum cost spanning tree, so the classic Kruskal algorithm's time complexity for solving the minimum cost spanning tree is O(|E|lg|E|). A new algorithm called two branch Kruskal algorithm has been proposed after the Kruskal algorithm was studied and analyzed carefully [2] [7].

3.2. The Basic Ideas of the Two Branch Kruskal Algorithm

After the two branch Kruskal algorithm was studied and analyzed carefully, the basic ideas of this algorithm are as follows:

1) Supposed a connected graph G = {V, E} with n vertexes and a non connected graph without vertex T = {V, φ}. The cost of all the edges in graph G is an array A at first , and select a cost K as an intermediate value, then array A is divided into two stature, A1 is composed of all the costs less than the middle value k, and A2 is composed of all the costs that are greater than the middle k. Then obtain the edge set E1 corresponding to A1 and the edge set E2 corresponding to A2 according to the principle of cost corresponding to edge respectively.

2) The vertex set is divided into n equivalent classes, (the total number of the equivalent classes n = E1 + E2), each equivalent class contains a vertex.

3) In graph G1 = {V, E1}, one can deal with each edge by ascending order of cost, if one edge connects two different equivalent classes, then this edge will be added to T, and the two equivalent classes will be merged into the same class immediately.

4) Performing the previous operation repeatedly until each edge of the E1 has been considered; At this point, the total number of the equivalent classes is 1 + E2 for graph G= {V, E}, and then deal with all edges E2 in G2 = {V, E2}again, if one edge connects to the vertexes of two different equivalent classes in graph G, the edge will be added to T, and the two equivalent classes a remerged into the same equivalent class at the same time immediately.

5) Performing the previous operation repeatedly until each edge of the E2 has been considered. Finally the total number of the equivalent classes is 1 for the graph G = {V, E} at this time, so that you obtained minimum cost spanning tree of graph G.

3.3. The Procedure of the Two Branch Kruskal Algorithm (G)

```
//Two branch Kruskal algorithm for constructing minimum spanning tree
//Input: weighted connected graph G=<V, E>, intermediate value K
//Output: Et, edge collection of minimum spanning tree that composed G
Select an intermediate value K from the set E
The E is divided into E1 composed by the edge which weight is less than K and E2 composed by the edge which weight is more than or equal to K, Then sorted the collection E1 according to the non-decreasing order of the weight w (e1) of the edge.
Et←Ø; encounter←0
//Initialize the vertex set of the tree and the size of the collection
k←0
//Initialize the number of initialized edges
while encounter<|V|-1 do//Handle edges in E1
    k←k+1
if Et∪{ei} No loop
Et←Et {i}; encounter←encounter+1
while encounter<|V|-1 do//Handle edges in E2
    k←k+1
if Et∪{ei} No loop
    Et←Et {ei}; encounter←encounter+1
Return Et
```

3.4. Examples of Two Branch Kruskal Algorithm

Question: Using the two branch Kruskal algorithm to find the minimum spanning tree for Figure 1.

Solution:

1) Obtain the array A = {3, 1, 4, 4, 6, 5, 5, 6, 2, 8} by the weight of all the edges from the set of E.

2) Select 5 as the middle value of K from the array A, then A will be divided into A1 = {1, 3, 4, 4, 2} and A2 = {6, 5, 5, 6, 8}

3) Corresponding edge set E1 = {(b, c) (a, b) (b, f) (c, f) (e, f)}, and E2 = {(a, e) (a, f) (d, f) (c, d) (d, e)}

4) Handle E1:

a) Handle edge (b, c) for Figure 2.

b) Handle edge (e, f) for Figure 3.

c) Handle edge (a, b) for Figure 4.

d) Handle edge (b, f) for Figure 5.

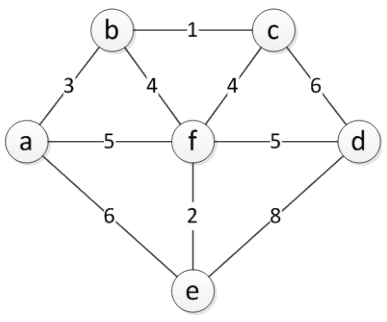

Figure 1. Example tree.

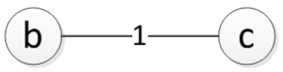

Figure 2. Edge (b, c).

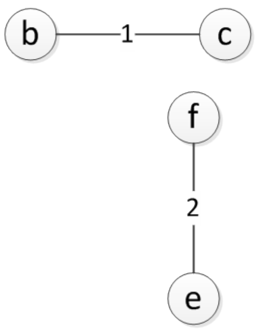

Figure 3. Edge (e, f).

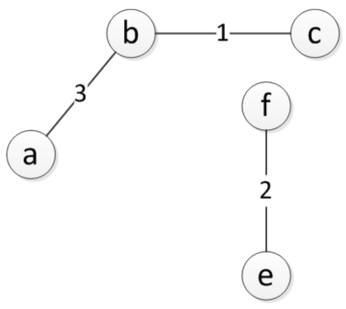

Figure 4. Edge (a, c) and (f, e).

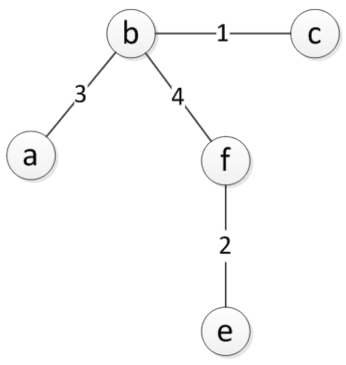

Figure 5. Edge (b, f).

5) Handle E2:

A. Handle edge (d, f) for Figure 6.

Figure 6 is the minimum spanning tree.

4. Comparison between the Classic and the Improved Kruskal Algorithm

The classic Kruskal algorithm's time complexity for solving the minimum cost spanning tree is O(|E|lg|E|). Then we consider the time complexity of this improved Kruskal algorithm: in the normal case (the value chose in step 1 of the algorithm is not the maximum or minimum value of the costs), |E1| = e1, |E2| = e2, so E1 + E2 = e. Its sum of the maximum time cost of the initial construction of the heap is (2 e1 − loge1 − 1) + (2 e2 − loge2 − 1). The maximum time to rebuild the heap is e1 loge1 + e2loge2. In addition, the time of the two sub sequence overhead in the first step is e. So the maximum total time cost of the improved Kruskal algorithm is BT1 = e + e + 2(e1 − 1)loge1 + loge2 − 2(e2 − 1) under normal circumstances.

Because the time cost is influenced by selection of the value and the arrangement method of edge costs, the average time cost of the two branch Kruskal algorithm is difficult to estimate. So it is no theoretical comparison about the average time cost between the two algorithms.

5. Conclusion

The two branch Kruskal algorithm is an improvement of the classical Kruskal algorithm, compared with the classical Kruskal algorithm. It has a middle value and uses classical Kruskal algorithm for two times. In the extreme case, the two branch Kruskal algorithm has a meaningless step of division according to the intermediate value compared with the classical Kruskal algorithm which causes that the time complexity of the two branch Kruskal algorithm is greater than the classical Kruskal algorithm; but the two branch Kruskal algorithm's time complexity is less than the classical Kruskal algorithm under normal circumstances.

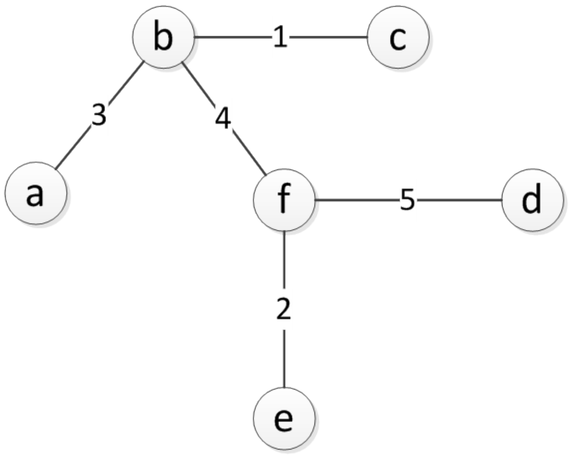

Figure 6. Edge (d, f).

Therefore, one can try to select the appropriate intermediate values to solve the minimum cost spanning tree in real life. It can reduce the cost to a certain extent.

Acknowledgements

This work is supported by Shanghai University of electric power smart grid collaborative innovation project (A-0009-17-002-05), Shanghai Pujiang Program (16PJ1433100) and Shanghai Municipal Natural Science Foundation (16ZR1436300).

References

[1] Huang, R.M. (2011) An Improved Kruskal Algorithm—Two Branch Kruskal Algorithm. Chinese Scientific Papers Online, 1-13.

[2] Levitin, A. (2015) Algorithm Design and Analysis Basics. 3rd Edition, Tsinghua University Press, Beijing.

[3] Wang, W. and Meng, S.Y. (2010) Research and Improvement of Algorithm. *Journal of Chongqing University of Arts and Science* (*Natural Science Edition*), **29**, 25-27.

[4] Hu, Z.Q. (2008) The Realization and Analysis of Kruskal Algorithm. *Computer Knowledge and Technology*, **11**, 311-312.

[5] Yang, X.Y. and Qian, N. (2009) Improvement of Kruskal Algorithm and Realization of VB. *Journal of Chuzhou Vocational & Technical College*, 8, No. 1.

[6] Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C. (2007) Introduction to Algorithms. 2nd Edition, Mechanical Industry Press, Beijing, 344-352.

[7] Liang, X.C. (2008) Application of Minimum Spanning Tree in Network Design. *Journal of Suzhou Education Institute*, No. 2, 150-153.