AN $O(|E| \log\log |V|)$ ALGORITHM FOR FINDING MINIMUM SPANNING TREES *

Andrew Chi-chih YAO
Department of Computer Science, University of Illinois, Urbana, Illinois 61801, USA

Received 30 December 1975, revised version received 9 June 1975

Minimum spanning tree, linear median finding algorithm

1. Introduction

Given a connected, undirected graph $G = (V, E)$ and a function c which assigns a cost $c(e)$ to every edge $e \in E$, it is desired to find a spanning tree T for G such that $\Sigma_{e \in T}\, c(e)$ is minimal. In this note we describe an algorithm which finds a minimum spanning tree (MST) in $O(|E| \log\log |V|)$ time. Previously the best MST algorithms known have running time $O(|E| \times \log |V|)$ for sparse graphs [1], and more recently Tarjan [2] has an algorithm that requires $O(|E| \times \sqrt{\log |V|})$ time.*

Our algorithm is a modification of an algorithm by Sollin [3]. His method works by successively enlarging components of the MST. In the first stage the minimum-cost edge incident upon each node of G is found. These edges are part of the MST sought. The groups of vertices that are connected by these edges are then identified. By shrinking each such group of vertices to a single node, we obtain a new graph with at most $\frac{1}{2}|V|$ nodes. This process is repeated for a number of times, at each stage for a new graph, until finally a single contracted node remains. Clearly each stage of this procedure involves $O(|E|)$ operations, and $\log |V|$ stages are necessary in the worst case. Thus this algorithm requires a total of $O(|E| \log |V|)$ operations.

In our algorithm, we first partition the set of edges incident with each node v into k levels $E_v^{(1)}, E_v^{(2)}, \ldots, E_v^{(k)}$ so that $c(e) \leqslant c(e')$ if $e \in E_v^{(i)}$, $e' \in E_v^{(j)}$ and $i < j$. This can be done in $O(|E| \log k)$ time by repeatedly applying the linear median-finding algorithm [4]. Having accomplished this, we follow basically Sollin's algorithm as outlined above. Note that the number of operations needed in this phase is now reduced to

$$O\left(\frac{|E|}{k} \log |V|\right)$$

since only approximately $|E|/k$ edges have to be examined at each stage to find the minimum-cost edges incident with all the nodes. Therefore, the total number of operations required by our algorithm is

$$O\left(|E| \log k + \frac{|E|}{k} \log |V|\right),$$

which is $O(|E| \log\log |V|)$ if we choose k to be $\log |V|$.

2. Algorithm

For the moment, assume $|E| \geqslant |V| \log |V|$. If $|E| < |V| \log |V|$, the algorithm needs a slight modification as will be discussed later.

The algorithm uses three sets T, VS and ES. T is used to collect edges of the final spanning tree. The set VS contains the vertex sets corresponding to the connected components of the spanning tree found so far. And ES contains, for each vertex set W in VS, an edge set $E(W)$. Initially we have $VS = \{\{v\} | v \in V\}$ and $ES = \{\{\text{all the edges incident upon } v\} | v \in V\}$. The algorithm also uses an integer parameter k, a *level function* $l: V \to \{1, 2, \ldots, k, k+1\}$, and a function low: $V \to$ real numbers.

* Research is partially supported by NSF GJ-41538.

```
Procedure MST;
  begin
    T ← φ; VS ← φ; ES ← φ;
    for each vertex v ∈ V do
      begin add the singleton set {v} to VS;
            add the set E({v}) = {all the edges incident with v} to ES;
            divide E({v}) into k levels of equal size according to cost, i.e. obtain E_v^(1), E_v^(2), ..., E_v^(k) with the
              property that ∪_{j=1}^{k} E_v^(j) = E({v}) and c(e) ≤ c(e') if e ∈ E_v^(i), e' ∈ E_v^(j), and i < j;
            set l(v) ← 1;
      end;
    while |VS| > 1 do
      begin
B.          take a vertex set W from VS;
            for each vertex v ∈ W do
              begin low (v) ← ∞;
                while low(v) = ∞ and l(v) ≤ k do
                  begin for each edge e = (v, v') in E_v^(l(v)) do
                        begin
C.                        if v' ∈ W then delete e from E_v^(l(v))
D.                        else low(v) ← min {low(v), c(e)};
                        end;
                      if low(v) = ∞ then l(v) ← l(v) + 1;
                  end
              end;
F.          find the edge e = (v, v') in E(W) whose cost is equal to min {low(v)|v ∈ W};
H.          in VS, replace W and the vertex set W' containing v' by W ∪ W';
I.          in ES, replace E(W) and E(W') by E(W) ∪ E(W');
            add e to T;
      end;
    output T;
  end MST
```

3. Remarks

(1) In the above procedure, the set VS is implemented with a circular queue. Step B corresponds to removing W from the front of the queue; step H corresponds to deleting W' and adding the new W to the tail of the queue. A full cycle of the queue, in which every vertex set of VS is merged with some others, corresponds to one "stage" of the algorithm as discussed before.

(2) Step A is done by repeatedly applying the median-finding algorithm [4, 5], for $i = 0, 1, \ldots, \log k-1$, to 2^i sets of size $|E(\{v\})|/2^i$. Since $\Sigma_v |E(\{v\})| = 2|E|$, this takes a total of $O(c|E|\log k)$ operations when a cn median-finding algorithm is used. The number of comparisons used is then $c \cdot 2|E|\log k$ + (lower order terms).

(3) Step C is executed at most $2|E|$ times, since each edge (v, v') of G can be thrown out at most twice – once as (v, v'), once as (v', v).

(4) Steps D and F amount to approximately

$$\log|V| \sum_{v\in V} \left\lceil \frac{|E(\{v\})|}{k} \right\rceil \leq \log|V| \left(\frac{2|E|}{k} + |V| \right)$$

min operations.

(5) The set union operation in step I can be implemented in a straightforward manner; for example, as in [6]. The total number of operations incurred is $O(|V|\log|V|)$.

It follows that the total cost of this algorithm is of the order

$$|E|\log k + \log|V| \frac{|E|}{k} + |V|\log|V| . \quad (1)$$

Taking $k = \log V$ and noting $|E| \geqslant |V|\log|V|$, the above expression by constant $\times$ ($|E|\log\log|V|$).

If $|E| < |V|\log|V|$ for a graph G, we will first let $k = 1$ and execute procedure MST until each vertex set in VS is of size at least $\log|V|$. This process takes at most $\log\log|V|$ "stages" since the size of the smallest vertex set in VS at least doubles after each stage. Hence the number of operations involved in this process is $O(|E|\log\log|V|)$ (including $2|E|\log\log|V|$ comparisons). The result can be regarded as a new graph $G' = (V', E')$ where $|V'| \leqslant |V|/\log|V|$ and $|E'| \leqslant |E|$. We now apply Procedure MST to G' with $k = \log|V|$. The number of operations required is given by (1),

$$|E'|\log k + \log|V'| \frac{|E'|}{k} + |V'|\log|V'|$$
$$\leqslant |E|\log k + \log|V| \frac{|E|}{k} + |V| ,$$

which is again $O(|E|\log\log|V|)$ for $k = \log|V|$.

(6) We have exhibited an algorithm with $O(|E|\log\log|V|)$ running time. The number of comparisons used is asymptotically $c'|E|\log\log|V|$ + (lower order terms) with $c' = 6$ when $|E| \geqslant |V| \log |V|$ and $c' = 8$ when $|E| < |V|\log|V|$, if a $3n$ median-finding algorithm [5] is employed.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Reading, Mass., 1974).

[2] R.E. Tarjan, unpublished.

[3] C. Berge and A. Chouila-Houri, *Programming, Games and Transportation Networks* (Wiley, 1965) p. 179.

[4] M. Blum, R.W. Floyd, V.R. Pratt, R.L. Rivest and R.E. Tarjan, "Time Bounds for Selection", JCSS 7 (1973) 448–461.

[5] A. Schönhage, M. Paterson and N. Pippenger, "Finding the Median", Theory of Computation Report No. 6, The University of Warwick.

[6] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Reading, Mass., 1974), section 4.6.