EFFICIENT ALGORITHMS FOR FINDING MINIMUM SPANNING TREES IN UNDIRECTED AND DIRECTED GRAPHS

H. N. GABOW*, Z. GALIL**, T. SPENCER*** and R. E. TARJAN

Received 23 January 1985
Revised 1 December 1985

Recently, Fredman and Tarjan invented a new, especially efficient form of heap (priority queue). Their data structure, the Fibonacci heap (or F-heap) supports arbitrary deletion in $O(\log n)$ amortized time and other heap operations in $O(1)$ amortized time. In this paper we use F-heaps to obtain fast algorithms for finding minimum spanning trees in undirected and directed graphs. For an undirected graph containing $n$ vertices and $m$ edges, our minimum spanning tree algorithm runs in $O(m \log \beta(m, n))$ time, improved from $O(m \log \log m)$ time, where $\beta(m, n) = \min \{i \log^{10} n / m \leq m/n \}$. Our minimum spanning tree algorithm for directed graphs runs in $O(n \log n + m \log \log \log (m/n) + n)$ time, improved from $O(n \log n + m \log \log \log (m/n + 2) n)$. Both algorithms can be extended to allow a degree constraint at one vertex.

1. Introduction

A heap (sometimes called a priority queue) is an abstract data structure consisting of a collection of items, each with a real-valued key, on which at least the following operations are possible:

make heap: Return a new, empty heap.

insert $(x, h)$: Insert item $x$, with predefined key, into heap $h$, assuming that it is not already in $h$.

find min $(h)$: Return an item of minimum key in heap $h$, without changing $h$.

delete min $(h)$: Delete from heap $h$ an item of minimum key and return it. If $h$ is empty, return a special null item.

A variety of other operations on heaps are sometimes useful. These include the following:

meld $(h_1, h_2)$: Return the heap formed by taking the union of item-disjoint heaps $h_1$ and $h_2$. This operation destroys $h_1$ and $h_2$.  

---

* Research supported in part by National Science Foundation Grant MCS—8302648.
** Research supported in part by National Science Foundation Grant MCS—8303139.
*** Research supported in part by National Science Foundation Grant MCS—8300984 and a United States Army Research Office Program Fellowship, DAAG29—83-G0020.

AMS subject classification (1980): 68 B 15, 68 C 05
decrease key \((A, x, h)\): Decrease the key of item \(x\) in heap \(h\) by subtracting the non-negative real number \(A\). This operation assumes that the location of \(x\) in \(h\) is known.

delete \((x, h)\): Delete item \(x\) from heap \(h\), assuming that its location in \(h\) is known.

Fredman and Tarjan [7] recently invented a heap implementation called the Fibonacci heap (abbreviated F-heap) that supports delete min and delete on an \(n\)-item heap in \(O(\log n)\) amortized time and all the other heap operations listed above in \(O(1)\) amortized time. (By amortized time we mean the time of an operation averaged over a worst-case sequence of operations. For a discussion of this concept see Tarjan’s survey paper [19].) The importance of this result is in its reduction of the time for decrease key to \(O(1)\) from the \(O(\log n)\) of previous heap implementations. Since decrease key is a central operation in many network optimization algorithms, F-heaps lead to improved running times for such algorithms. In particular, Fredman and Tarjan reduced the running time of Dijkstra’s shortest path algorithm from \(O(m \log (m/n+2)n)\) to \(O(n \log n+m)\) and showed how to find minimum spanning trees in undirected graphs in \(O(m \beta (m, n))\) time, improved from \(O(m \log \log (m/n+2)n)\). Here \(n\) and \(m\) are the numbers of vertices and edges in the problem graph, respectively, and \(\beta (m, n) = \min \{\log^{(i)} n \leq m/n\}\), where \(\log^{(i)} n\) is defined by \(\log^{(0)} (x) = x, \log^{(i+1)} x = \log \log^{(i)} x\). (Throughout this paper we use base-two logarithms.)

Our purpose in this paper is to explore the use of F-heaps in computing minimum spanning trees in undirected and directed graphs. Our two main results, discussed in Sections 2 and 3, respectively, are as follows:

(i) By adding the idea of packets [8, 9] to the Fredman—Tarjan undirected minimum spanning tree algorithm, we reduce its running time to \(O(m \log \beta (m, n))\) from \(O(m \beta (m, n))\).

(ii) By observing that in certain situations items can be moved among F-heaps in \(O(1)\) amortized time per item moved, we obtain an implementation of Edmonds’ minimum directed spanning tree algorithm [16] with a running time of \(O(n \log n+m)\), improved from the \(O(n \log n+m \log \log (m/n+2)n)\) bound of Gabow, Galil, and Spencer [8]. Our algorithm is also substantially simpler than the previous one.

Both our algorithms extend to allow a degree constraint at one vertex as we discuss in Section 4. Section 5 contains a few concluding remarks. The undirected minimum spanning tree algorithm described here originally appeared in a symposium paper [8] by the first three authors.

2. Minimum spanning trees in undirected graphs

Let \(G=(V, E)\) be a connected, undirected graph with vertex set \(V\) of size \(n\) and edge set \(E\) of size \(m\), such that each edge \(\{v, w\}\) has a real-valued cost \(c(v, w)\). Connectivity implies \(m \geq n-1\); we shall assume \(m= n-1\) implies \(G\) itself is a tree. A minimum spanning tree of \(G\) is a spanning tree whose total edge cost is minimum. The problem of computing minimum spanning trees has a long history [11, 17]; the first algorithm was proposed by Boruvka [2] in 1926. Before the invention of F-heaps, the best known time bound was \(O(m \log \log (m/n+2)n)\) [4], a slight improvement over Yao’s \(O(m \log \log n)\) bound [20]. Fredman and Tarjan [7] used F-heaps to obtain an \(O(m \beta (m, n))\) bound. We shall modify their algorithm so that it runs in \(O(m \log \beta (m, n))\) time.