Advanced Algorithms	Lecture 11 : May 3, 1995
Lecturer: Johan Håstad	*Scribe: Ann Bengtson*

Median finding

11.1 Introduction

In this lecture we will consider the problem of finding the median of n elements, i.e. the element with order number $n/2$. The most straightforward method is to sort the elements and then choose element number $n/2$. As sorting has complexity $O(n \log n)$ the time required for finding the median will also be $O(n \log n)$.

However, the information we want is just the element with order number $n/2$ and we are not really interested in the order among the larger or smaller elements. Algorithms for median finding needing only $O(n)$ comparisons are available and we will describe two such algorithms here.

11.2 Simple algorithm for median in $O(n)$

11.2.1 Algorithm description

In this algorithm we will assume that $n = 5m$.

Idea: Partition the n elements into m groups with 5 elements in each group. Find the median in each group. (Finding the median includes partitioning the elements into those smaller and those larger than the median.) Find the median of the m medians from the 5-groups. Call this last median x_0.

The result so far is illustrated by the rectangle of figure 11.1 where the elements are arranged so that the 5-element groups are placed columnwise in the rectangle. The largest elements are at the top and to the left. The elements of the upper left of the rectangle must all be larger than or equal to x_0. Similarly, the elements of the bottom right are all smaller than x_0. What is achieved here is the first step of a partitioning of the whole set around the element x_0. We will now partition the rest of

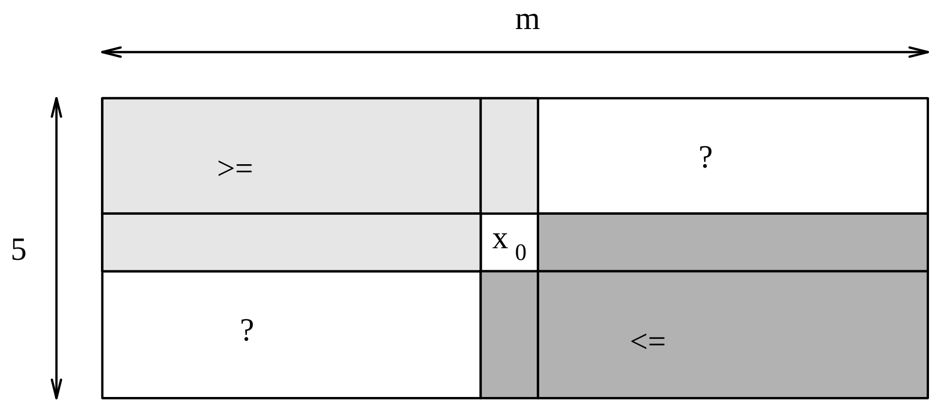

Figure 11.1: The two different shadings illustrate elements larger and smaller than x_0. We do not know how the non-shadowed elements are related to x_0.

the elements by comparing them with x_0. The set is now partitioned into two parts and the median must be in the largest of the two parts. We will continue the search in this part but we will no longer search for the median.

A more formal description for finding the median, or an element with order k, is the following.

Let $F(S,k)$ = find and return element of order k from set S. $|S| = n$. Assume $n = 5m$. If not, one group will be a little smaller.

1. Partition S into groups with 5 elements each.
2. Let $S_1 = \{\text{medians from these groups}\}$, $|S_1| = m$
3. $x_0 = F(S_1, m/2)$
4. $S_2 = \{x|x \in S, x < x_0\}$ $S_3 = \{x|x \in S, x > x_0\}$ If $|S_2| \geq k$ then $F(S_2, k)$ If $|S_3| \geq n - k + 1$ then $F(S_3, k + |S_3| - n)$. Otherwise, the answer is x_0.

When $|S|$ is small, $|S| \sim 10$, use any algorithm.

11.2.2 Algorithm analysis

$M(n)$ = the number of comparisons needed to find the kth element of n elements. We need to find the median of 5 elements. This can be done with 7 comparisons.

$M(n) \leq (n/5) \times$(number of comparisons for median of 5 elements) +
$M(n/5) + (4n/10) + M(max(|S_3|, |S_2))$

The second term comes from finding the median of medians (element x_0 above). The third term is the number of comparisons to complete the partitioning, the number of uncertain elements in the figure. The last term is the continued search among the remaining elements. The number of elements left cannot be larger than $(7n/10)$ because both S_2 and S_3 have at least $(3n/10)$ elements (see figure 11.1) so the largest of these must have at most $(7n/10)$ elements.

$$M(n) \leq 7n/5 + M(n/5) + 2n/5 + M(7n/10)$$

$$M(n) \leq 9n/5 + M(n/5) + M(7n/10)$$

The solution of a relation of this type where

$$M(n) \leq cn + M(an) + M(bn)$$

and $a + b < 1$ is

$$M(n) \leq \frac{cn}{1 - (a + b)}$$

With $c = 9/5$, $a = 1/5$ and $b = 7/10$ we get

$$M(n) \leq 18n$$

This algorithm was created in 1972 by Blum, Floyd, Pratt, Rivest, and Tarjan.

11.3 "Spider factory" median algorithm

11.3.1 Some preliminaries

We will now describe an algorithm which has complexity $5n + o(n)$. It was developed in 1976 by Schönhage, Pippenger and Paterson.

This algorithm uses the term "spider factory" which is a device that from a set of unordered elements will produce partial orders with $2k+1$ elements where k elements are larger and k elements are smaller than one element m. In the graphic image of

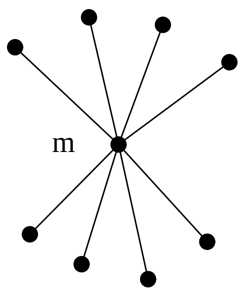

Figure 11.2: A spider with $2k+1$ elements

this partial order, m is pictured as the body of the spider and the other elements are connected to m with legs, each leg denoting that a relation between two elements has been established. Legs pointing upwards connect elements $> m$ and legs pointing downwards connect elements $< m$ to the body. (The desired result of any median finding algorithm is such a spider with $k = n/2$.)

The production cost of the spider factory is the following:

- Initial cost, before spider production can begin: $O(k^2)$
- Residual cost, when no more spiders can be created: $O(k^2)$
- Cost per output spider: $5k$

A suitable choice of k is $k \sim n^{1/4}$. Details of the spider factory will be given in section 11.3.4.

11.3.2 Algorithm

Run the spider factory until no more spiders are produced and arrange the spiders in a list sorted with respect to the middle elements of the spiders. Let n be the total number of elements and let r be the number of residual elements, i.e. the number elements that are left when all possible spiders are produced. The number of spiders in the spider list is then

$$s = \frac{n-r}{2k+1} = O(n/n^{1/4}) = O(n^{3/4})$$

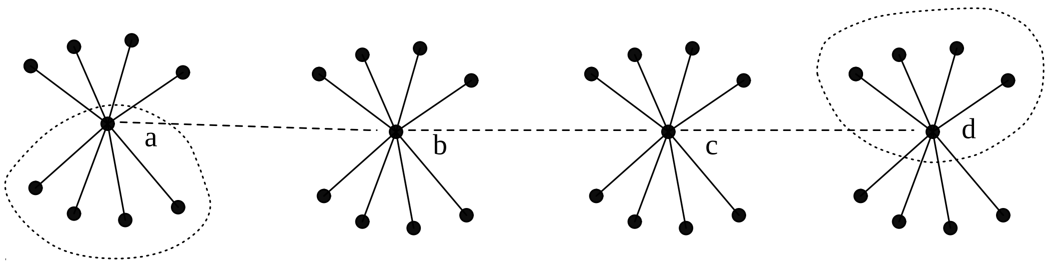

Figure 11.3: A list of 4 spiders sorted on the middle element, $a \le b \le c \le d$.

Now, consider the lower legs and the body of the leftmost spider, S_l. These $k+1$ elements are all smaller than p_1 elements where $p_1 = k + (s-1)(k+1)$ (upper legs of leftmost spider and upper legs and body of the other spiders). Each of the elements in S_l could be larger than at most p_2 elements where $p_2 = sk - 1 + r$. As long as $p_1 > p_2$ the median cannot be one of the elements in S_l and these elements can be discarded. In the same way the elements of the upper legs and body of the rightmost spider must all be larger than the median and can be discarded. The remaining parts of these two spiders will be returned to the factory and their elements used for building new spiders. Evaluating $p_1 > p_2$ gives the simple condition $s > r$. The procedure of

- chopping off partial spiders at ends of list
- returning remainders to the factory
- producing new spiders
- sorting spiders into list

will succesively reduce the number of remaining elements and the size of the list and is continued as long as $s > r$. When the procedure stops, run the simpler algorithm of section 11.2 on the remaining elements.

11.3.3 Analysis

Let t be the number of spiders produced. $t \sim n/k \sim n^{3/4}$ because to discard $k+1$ elements at each end of the list, two spiders are used. Therefore, each spider allows us to eliminate $k+1$ elements and as at most n elements will be eliminated the total number of spiders must be at most $n/(k+1) \sim n/k = n/n^{1/4} = n^{3/4}$.

The cost of this algorithm has 4 terms

- Initial cost: $O(k^2) = O(n^{1/2})$
- Residual cost: $O(k^3) = O(n^{3/4})$
- Cost for producing the t spiders: $5tk = 5n^{3/4}n^{1/4} = 5n$
- Cost for inserting spider elements in the sorted list: $t \log s = O(n^{3/4} \log n^{3/4})$

The initial cost is the cost to start the spider production and will be explained in the next section. The number of residual elements in the factory when spider production stops is $O(k^2)$, this will also be shown in the next section. The total residual is then $O(k^3)$ because at this stage, $s \sim k^2$ and the remaining number of elements is $s(2k+1) + k^2 = O(k^3)$. The cost of finding the median of these remaining elements is $O(k^3)$.

The dominating cost here is producing the spiders and so the overall cost for the algorithm for large values of n is $\sim 5n$.

11.3.4 The spider factory

When creating the spiders a crucial idea is to compare medium valued elements with each other and successively create partial orders which eventually can be used to construct the spiders that will leave the factory. By "medium valued elements" we mean elements which have been found to be smaller than some number of elements and larger than some number of elements, these numbers differing with 0 or 1. The following procedure is used. For illustration, see figure 11.4.

1. Begin with the unordered elements (instances of H_0).
2. Compare pairwise and mark the smallest element, the "loser" in each pair (instances of H_1 are created).
3. Compare the marked elements of the pairs and combine pairs to groups of 4 elements. This time we mark the largest element, the "winner" of each comparison (instances of H_2 are created).
4. The marked elements of the quadruples are compared, the losers are marked and groups of eight elements are formed (instances of H_3).

These steps are continued so that

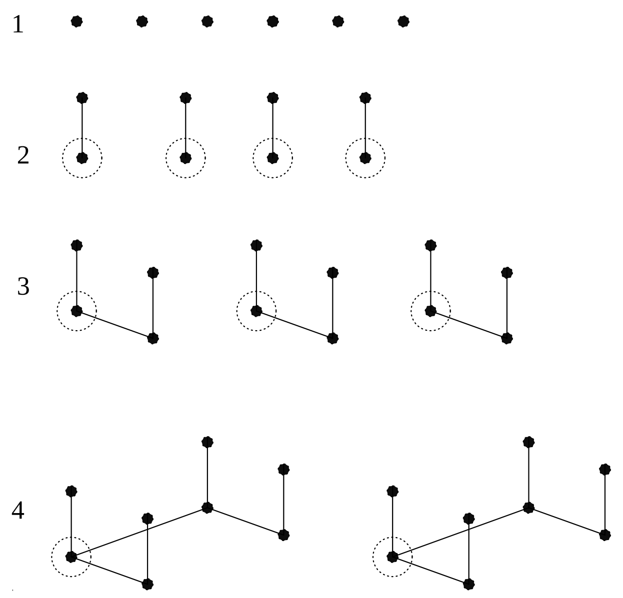

Figure 11.4: Illustration of steps $1, 2, 3$ and 4, creating partial orders of types H_0, H_1, H_2 and H_3.

- Each group H_{2i} is built from two groups H_{2i-1} and the winners in the comparisons are marked.
- Each group H_{2i+1} is built from two groups H_{2i} and the losers are marked.

Lemma 1 *H_j contains 2^j elements.*

Proof: Obvious from the construction of H_j ■

Lemma 2 *An instance of H_{2l} with center element v contains at least $2^l - 1$ elements which are $> v$ at least and $2^l - 1$ elements which are $< v$ and thus a spider with $k = 2^l - 1$ can be output from H_{2l}.*

Proof: The proof is by induction. Suppose the lemma is true for $l = p - 1$ and show that this implies that it is true for $l = p$. An instance of H_{2p} is built from 2 instances of H_{2p-1} and these are built from 4 instances of H_{2p-2}. v is the center point of one of these. Let the other 3 center points be u_1, u_2 and u_3. See figure 11.5. The number of elements $> v$ are

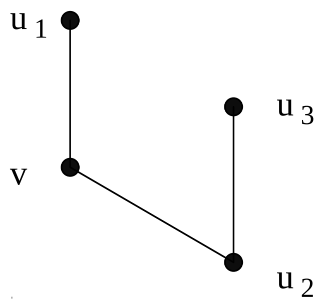

Figure 11.5: The center elements of 4 instances of H_{2p-2} used to build one instance of H_{2p} with center element v.

- u_1 itself, 1 element
- The elements $> v$ in v:s instance of $H_{2(p-1)}$, $2^{p-1}-1$ elements according to the assumption of the lemma.
- The elements $> u_1$ in its instance of $H_{2(p-1)}$, $2^{p-1}-1$ elements also according to the lemma.

Summing the three terms we get the total number of elements $> v$

$$1 + 2^{p-1} - 1 + 2^{p-1} - 1 = 2^p - 1.$$

In the same way, we can count the number of elements of H_{2p} that are $< v$ by using $<$ instead of $>$ and u_2 instead of u_1. We get the same result, $2^p - 1$. We have shown that if the lemma is true for $l = p-1$, it is also true for $l = p$. It is very easy to check that the lemma is true for some small numbers, $l = 0, 1, 2$ and therefore it is true for all values of l. ■

The spider factory will build instances of $H_0, H_1, \ldots, H_{2l-1}$ and whenever 2 instances of H_{2l-1} are available, an instance of H_{2l} is made and a spider with $k = 2^l - 1$ is output.

In order to estimate the complexity of the spider production we need to investigate what happens to the elements of H_{2l} that will not be part of the spider and in particular we must look at how many of the edges within an H_{2l} are broken when the spider is created.

Creating the spider from H_{2l}, its center v will be the center of the spider. We will now look at the 4 instances of H_{2l-2} with the centers v, u_1, u_2, and u_3.

- $H_{2l-2}(u_3)$ will just stay in the factory for further use, we cannot know how its elements are related to v.
- $H_{2l-2}(u_2)$: u_2 and all elements that we know are smaller go to the spider, the rest will stay in the factory.
- $H_{2l-2}(u_1)$: u_1 and larger elements go to the spider, the rest will stay in the factory.
- $H_{2l-2}(v)$: v will be the body of the spider, elements known to be larger or smaller will be in the spider.

The cost of the production is the number of comparisons which is equivalent to the number of edges between elements. We count the edges when they leave the factory as part of a spider and when they are destroyed in the spider production. Each spider leaving the factory has $2k$ edges. The edges lost are $F_{>}(i)$ for removing from H_{2i} the center element and elements larger than the center element and $F_{<}(i)$ for edges lost when removing smaller elements. $F_{>}(i)$ and $F_{<}(i)$ can be expressed recursively as

$$F_{>}(i) = 1 + 2F_{>}(i-1), \qquad F_{>}(1) = 1$$

and

$$F_{<}(i) = 2 + 2F_{<}(i-1), \qquad F_{<}(1) = 2$$

and the solutions are

$$F_{>}(i) = 2^i - 1$$

and

$$F_{<}(i) = 2 \times 2^i - 2$$

The cost of removing both larger and smaller elements is

$$F_{>,<}(i) = F_{>,<}(i-1) + F_{>}(i-1) + F_{<}(i-1) + 1, \qquad F_{>,<}(1) = 1$$

$$F_{>,<}(i) = F_{>,<}(i-1) + 3 \times 2^{i-1} - 2 =$$

$$F_{>,<}(1) + \sum_{j=1}^{i-1}(3 \times 2^j - 2) =$$

$$1 + 3(2^i - 1) - 2(i-1) \leq 3(2^i - 1)$$

The cost for the lost edges is $3(2^i - 1) = 3k$ per spider. To get the total cost for each produced spider we add the cost for the output edges, $2k$, and we get $5k$.

We can now specify the terms of the total cost of the algorithm:

- Cost per produced spider: $5k$.
- Initial cost, the production of the first $H_{2l} \sim 2^{2l} \sim k^2$.
- The maximum residual in the factory should be one instance of each of $H_0, H_1, \ldots, H_{2l-1}$ because whenever 2 instances of H_j esist, they will be combined into a H_{j+1}. As H_i has 2^i elements the residual will have at most $\sum_{i=0}^{2l-1} 2^i \leq 2^{2l} \sim O(k^2)$ elements. The cost for calculating the median of the residual is then $O(k^2)$.

Another algorithm from 1976 by the same authors has complexity $3n + o(n)$ and a very recent (1994) modification of this algorithm by Dor and Zwick has complexity $2.95n + o(n)$. The strongest known lower bound is $2n - o(n)$ and was obtained 1985 by Bent and John.