CS580 Algorithm Design, Analysis, and Implementation

Lecture notes, Jan 20, 2022
Wufei Ma

1 Selection

Problem: Find the k-th smallest key in a set of keys.

Selection of the minimum.

```
1     Function FindMin
2         min := 1
3         for i := 2 to n do
4             if A[min] > A[i]:
5                 min := i
6             endif
7         endfor
```

Randomized selection. Find the i-th smallest element. Similar to quick sort.

```
1     Function Random-Select(p, r, i):
2         if p == r:
3             return p
4         else:
5             q := Random-Partition(p, r)
6             k := q - p + 1
7             if i <= k:
8                 return Random-Select(p, q, i)
9             else:
10                return Random-Select(q+1, r, i-k)
```

In the worst case scenario, every time we remove 1 item from the array and the time complexity is

$$T(n) = T(1) + T(n-1) + O(n) = O(n^2)$$

Now we consider the average case analysis of `Random-Select`. Recall that with probability $\frac{2}{n}$ the array is split 1 and $n-1$, and for $2 \leq k \leq n-1$, with probability $\frac{1}{n}$ it is

split into k and $n-k$. Therefore,

$$\begin{aligned} T(n) &\leq \frac{1}{n}\left(T(\max\{1,n-1\}) + \sum_{k=1}^{n-1} T(\max k, n-k)\right) + \Theta(n) \\ &\leq \frac{1}{n}\left(T(n-1) + 2\sum_{k=\lceil k/2\rceil}^{n-1} T(k) + \Theta(n)\right) \\ &= \frac{2}{n}\sum_{k=\lceil k/2\rceil}^{n-1} T(k) + O(n) \end{aligned}$$

Assume $T(n) \leq cn$ for some constant c. It follows that

$$\begin{aligned} T(n) &= \frac{2}{n}\left(\sum_{k=1}^{n-1} T(k) - \sum_{k=1}^{\lceil k/2\rceil - 1} T(k)\right) + O(n) \\ &\leq \frac{2c}{n}\left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lceil k/2\rceil - 1} k\right) + O(n) \\ &\leq c(n-1) - \frac{c}{2}\left(\frac{n}{2} - 1\right) + O(n) \\ &\leq \frac{3c}{4}n + O(n) \leq cn \end{aligned}$$

Insertion sort with jumps. We sort the elements `p, p+k, p+2k, \dots`.

```
1     Procedure Insertion-Sort(p, k, n):
2         last := p + k
3         while last <= n do:
4             pos := last
5             while pos > p and A[pos] > A[pos-k] do:
6                 swap(A[pos], A[pos-k])
7                 pos := pos - k
8             endwhile
9             last := last + k
10        endwhile
```

Deterministic selection.

1. We start by dividing the keys in groups of 5. So for $k = \lceil n/5 \rceil$, `A[j], A[j+k], ..., A[j+4k]` is a group of 5.
2. Now we find the median of each group.
3. Find the median of the medians by calling this algorithm recursively.
4. Partition the array with the median-of-medians as the pivot.

5. If $i < q$, then we find the key on the left part; otherwise, we find the key on the right part.

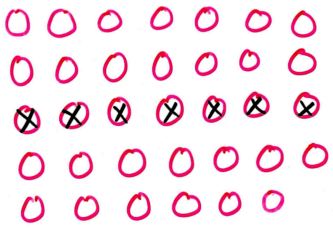

Figure 1: Divide the keys into groups of 5.

We implement this algorithm with insertion sort.

```
1     Function Select(p, r, i):
2         if r - p + 1 <= 50:
3             Insertion-Sort(p, 1, r)
4             return A[p+i-1]
5         else:
6             k = ((r - p + 1) + 4) // 5
7             for j = 0 to k-1 do:
8                 Insertion-Sort(p+j, k, r)
9             endfor
10            medmed = Select(p+2k, p+3k-1, k // 2)
11            swap(A[p], A[medmed])
12            q = Partition(p, r)
13            if i <= q - p + 1 then:
14                return Select(p, q, i)
15            else:
16                return Select(q+1, r, i - (q-p+1))
17            endif
18        endif
```

To analyze this algorithm, we ignore the ceiling and floor functions.

1. The number of median less than or greater than the `medmed` is $\frac{n}{10}$.
2. The number of elements less than or greater than these medians is $\frac{2n}{10}$.
3. We throw away (prune) at least $\frac{3n}{10}$ elements.

Now we estimate the time complexity.

$$T(n) \leq T(n/5) + T(7n/10) + O(n)$$

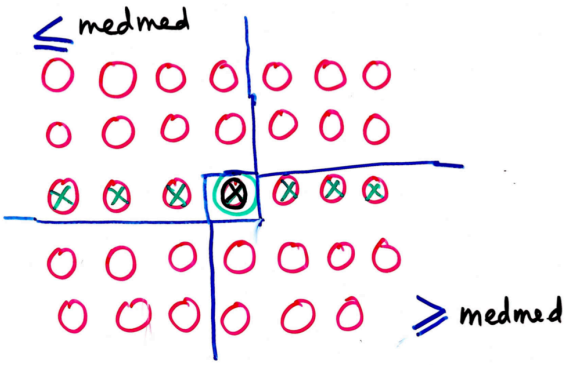

Figure 2: Divide the keys into groups of 5.

We prove $T(n) = O(n)$ by assuming $T(n) \leq cn$. It follows that

$$\begin{aligned} T(n) &\leq c\frac{n}{5} + c\frac{7n}{10} + O(n) \\ &\leq c\frac{9n}{10} + O(n) \\ &\leq cn \end{aligned}$$

We choose c such that $O(n) = c'n \leq c\frac{n}{10}$, which is $c \geq 10c'$.