Parma University, Math. Department

Algorithms and Data Structures
Order statistics

Marco Pellegrini

Summary of lecture 7

- Summary of this lecture: (CLR 10).
  Finding the element of rank k in an (unsorted) array.
  The smallest, largest, median elements.

Solution 1: Use sorting.

- Problem: Given an array A with n numbers, and an integer k with $1 \le k \le n$, find the k-th number in (increasing) sorted order.
- Easy solution. Sort(A) and look in position A[k].
- We can do sorting in time O(n log n). The game now is to solve the problem faster than O(n log n)

Easy case: minimum

- Minimum(A: array of integers)
  min := A[1]
  FOR i = 2 TO length(A) DO
  IF min > A[i] THEN
  min := A[i]
  END IF
  END FOR
  RETURN min
- Exercise: write a code to find the second smallest element in A.

Ramdom-select method

- We see now a method based on ideas we used for quick-sort. The main idea is that after the RANDOMIZED-PARTITION call on A we can tell quickly on which of the two sub-arrays we have to search and forget the other part.
- ```
  RANDOM-SELECT(A,p,r,i)          %always i≤r-p+1.
    IF p=r THEN RETURN A[p] ENDIF
    q := RANDOMIZED-PARTITION(A,p,r)
    k := length(A[p..q]) % q-p+1
    IF i ≤ k
      THEN RETURN RANDOM-SELECT(A,p,q,i)
      ELSE RETURN RANDOM-SELECT(A,q+1,r, i-k)
    ENDIF
  ```
- Initial call RANDOM-SELECT(A,1,length(A),i)

Analysis of RANDOM-SELECT.

- Worst case. In the worst case the split is 1 to n-1 and we always recur on the long subarray.
- We get $T(n) < T(n-1) + O(n)$. Solving $T(n) = O(n^2)$.
- Best case. The array is always split in half.
- We get $T(n) < cn + T(n/2)$.
- Proof by induction that $T(n) < dn$ for a constant d.
- $T(n) < cn + d(n/2) = n(d/2 + c)$. Now $(d/2+c) < d$ when we choose $d > 2c$.
  ```

What happens if for g>2 the split is always (1/g) to (1-1/g) and we recur in the long array?

Let try to prove T(n)<dn by induction.

The recursion is T(n)<cn+T((1-1/g)n).

So T(n) < cn + d(1-1/g)n = n(c+d(1-1/g)) <dn when c+d(1-1/g) < d, that is when d> gc.

HW2.1 (CLR 10-2.1) Write an iterative version of RANDOM-SELECT.

Average time of Random-select

Randomized-select returns index 1 with probability 2/n and any index 2,3,..,n-1 with probability 1/n.

Assume T(n) is an increasing function.

$$T(n) \le \frac{2}{n} T(\max(1, n-1)) + \frac{1}{n}\sum_{k=2}^{n-1} T(\max(k, n-k) + O(n) =$$

$$= \frac{1}{n} T(\max(1, n-1)) + \frac{1}{n}\sum_{k=1}^{n-1} T(\max(k, n-k) + O(n) \le$$

$$\le \frac{1}{n} \sum_{k=\lceil n/2 \rceil}^{n-1} 2T(k) + O(n)$$

(1/n)T(n-1)<(1/n)O(n^2)=O(n)

Now we prove by induction T(n)≤cn.

$$T(n) \le \frac{1}{n} \sum_{k=\lceil n/2 \rceil}^{n-1} 2T(k) + O(n) \le \frac{1}{n} \sum_{k=\lceil n/2 \rceil}^{n-1} 2ck + O(n) \le$$

$$\le \frac{2c}{n} \sum_{k=\lceil n/2 \rceil}^{n-1} k + O(n) = \frac{2c}{n}\left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lceil n/2 \rceil - 1} k\right) + O(n) =$$

$$= \frac{2c}{n}\left(\tfrac{1}{2} n(n-1) - \tfrac{1}{2}\lceil n/2 \rceil(\lceil n/2 \rceil - 1)\right) + O(n) \le$$

$$= \tfrac{3}{4} cn - \tfrac{1}{2} c + dn \le cn$$

When c>4d.

Selection in worst-case linear time.

RANDOM-SELECT is easy to program and it is linear on average. Can we get worst case time O(n)?

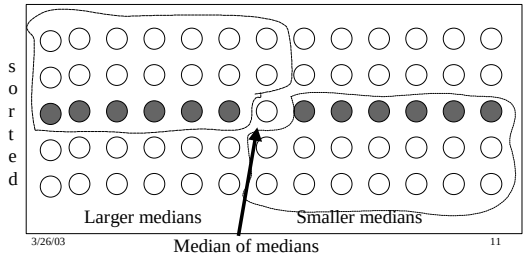
Yes if we can guarantee that we always split the array in a ratio bounded by a constant, so we need a smarter PARTITION procedure.

Equivalently we need to choose our pivot more carefully so that neither of the two subarrays is too small.

Idea: use recursively the SELECT procedure to help finding such a pivot.

Idea for choosing the pivot

- (1) Split the array A in groups of 5.
  (2) Find the median element in each group.
  (3) find the median of the medians.
  (4) this is a good pivot.



Deterministic selection in linear time

```
DETRIMINISTIC-SELECT(A,p,r,i)    % always i≤r-p+1.
  IF r-p<d THEN RETURN RANDOM-SELECT(A,p,r,i) ENDIF
  B=MEDIANS-BY-GROUPS-OF5(A,p,r)

  q := DETERMINISIC-SELECT(B,1,lenght(B),⌈length(B)/2⌉)
  k := length(A[p..q])  % q-p+1
  IF i ≤ k
    THEN RETURN DETERMINISTIC-SELECT(A, p, q, i)
    ELSE  RETURN DETERMINISTIC-SELECT(A, q+1, r, i-k)
  ENDIF
```

```
MEDIANS-BY-GROUPS-OF5(A:array,p,r): array
i=1, b=p
REPEAT
   f= min(b+4,r)
   m=⌈(f+b)/2⌉
   INSERTION-SORT(A,b,f) % modified to work on portions of A
   B[i]=A[m], i++
   b=f+1
UNTIL (b>r)
RETURN B
```

This code on an array of n elements runs in time O(n).

Worst case time of determinsitic-select

Consider the two lines:

B=MEDIANS-BY-GROUPS-OF5(A,p,r)

q := DETERMINISIC-SELECT(B,1,lenght(B), ⌈length(B)/2⌉)

Call n the number of elements of A between indices p and r.

B has ⌈n/5⌉ elements. q has rank ⌈ (1/2) ⌈n/5⌉ ⌉.

So there are ⌈ (1/2) ⌈n/5⌉ ⌉ -2 columns in A giving at least 3 elements smaller than q.

$$|A_{<q}| = |\{x \in A, x < q\}| \geq 3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5}\right\rceil\right\rceil - 2\right) \geq \tfrac{3}{10}n - 6$$

So there ara *at most* (7/10)n+6 elements in A larger than q.

Doing a symmetric argument we find that there are *at most* (7/10)n+6 elements in A smaller than q.

Moreover (7/10)n+6<n for n>20. So the cosntant d is at least 20.

$$T(n) \leq \begin{cases} O(1) & n < d \\ O(n) + T(\lceil n/5 \rceil) + T(\lceil 7n/10 \rceil + 6) & n > d \end{cases}$$

First recursive call on B

Second recursive call

Now we prove by induction that T(n)<cn.

$$T(n) \leq c\lceil n/5 \rceil + c\lceil 7n/10 + 6 \rceil + en \leq c(n/5+1) + c(7n/10+7) + en =$$

$$= \tfrac{9}{10}cn + 8c + en \leq cn$$

True for c>10(e+1), and n>8c.

Exercise.

- CLR 10.3-3. Can we make Quick-sort run in time O(n log n) in the worst case?

Conclusions

- We have see how we can find the element of rank k in a set of numbers in linear time.
- We have a randomized version in expected linear time and a deterministic version in worst case linear time.
- Using the same method Quicksort can be made worst case O(n log n).
- Reading assignment x next lecture: CLR 9.