CISC 320 Introduction to Algorithms
Fall 2005

Lecture 7
Selection and Order statistics

- Problem: Given an array E containing n elements with keys from some linearly ordered set, find an element with the k-th smallest key.
- Why this is interesting?
  - max, min, median, mean, …
- If the array is ordered (with O(n log n) time), then E[k] is the answer.
- Can we do better?


```
findMax(E, n)
max = E[0];
for (I = 1; I < n; i++)
   if(max < E[i])
      max = E[i];
return max;
```

It takes n-1 comparisons to find the largest key, that is better than O(n log n).

- Is this the best we can do for finding the largest key by comparisons?

Yes.

- For n distinct keys, only one is the largest => n-1 losers.
- Each comparison generate only one loser => n-1 comparisons needed.
- If there are two or more nonlosers left when the algorithm terminates, it can not be sure it has identified the max.

- 2ndLargest

  apply findMax once and remove the max, then apply findMax again.

  (n-1) + (n-2) = 2n – 3.

- i-th key

  (n -1) + (n-2) + … (n-i)

- Median i= n/2

  $\sum_{i=1 \text{ to } n/2} (n-i) = (3/8)\, n^2 - n/4 \in O(n^2)$

Note:
- This is even worse than sorting the array first.
- Finding the median seems to the hardest selection problem.

- Divide and conquer?
  - Simple minded one: partitioning S → S1 and S2, then the median is in the larger set, say S2, and we gain by ignoring the smaller set.

    Then we do this recursively on S2!

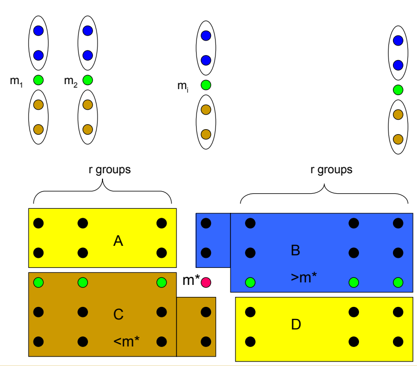
    Wait, the median of S2 is not the median of S.

Selection in worst-case linear time

Algorithm select(S, k)

1. Divide the n elements into n/5 groups of 5 elements each and at most one group made up of the remaining n mod 5 elements.
2. Find the median of each n/5 groups
3. Use select recursively to find the median m* of n/5 medians found in step 2
4. Partition using m* as pivot:
   Compare each key in the sections A and D to m*.
   Let S1 = C ∪{keys from A ∪ D that are smaller than m*}
   Let S2 = B ∪{keys from A ∪ D that are larger than m*}
5. Divide and conquer:

```
if (k = |S1| + 1)
    return m*; // because m* is the k-th smallest key
else if (k ≤ |S1|)
    return select(S1, k); // the k-th smallest key of S is in S1, and is the k-th smallest key in S1.
else
    return select(S2, k-|S1|-1); // the k-th smallest key of S is in S2, and
                                 // it is the k-|S1|-1 smallest key in S2.
```

Select: complexity analysis

For simplicity, let n = 5(2r+1) and integer r > 0.

1. Find medians of 5 keys: 6 comparisons
2. There are n/5 sets: 6(n/5)
3. Recursively find the median m* of the medians: T(n/5)
4. Compare all keys in section A and D to m*: 4r comparisons.
5. Recursively subset S1 or S2: T(7r +2)
   B and C section each has 3r+2 elements, plus
   4r elements from A and D. r ≈ n/10.

$$T(n) = T(7n/10) + T(n/5) + (6/5)n + (4/10)n$$
$$= T(7n/10) + T(n/5) + (8/5)n$$

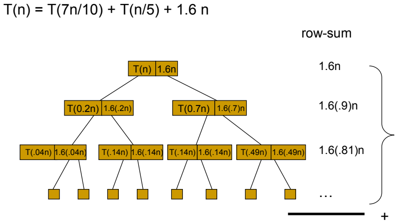
- Important observation:

  $(n/5) + (7n/10) = (9/10)n < n$

  Parts add up to less than the whole.

  This implies a decreasing geometric series when untangle the recurrence equation.

$T(n) = T(7n/10) + T(n/5) + 1.6\,n$



$$T(n) = 1.6\,n\,(1+ 9/10 + (9/10)^2 + \ldots) \le 1.6n \sum_{i=0}^{\infty}(9/10)^i$$
$$= 1.6n\,(1/(1-9/10))$$
$$= 16\,n$$

More generally, for recurrence equation

$T(n) = cn + T(a\ n) + T(b\ n)$,

if $a+b < 1$, then

$T(n) \le c\ [\ 1/\ (1-a-b)]\ n$

Question: will algorithm *select* still be linear if we divide the keys into sets of 3, or 7?

Selection algorithms:

For median selection,

Blum, Floyd, Pratt, Rivest & Tarjan
5.34n

Dor and Zwick (1995) 2.95n

Dor and Zwick (1996) $(2+\varepsilon)n$
$\varepsilon \approx 2^{-80}$ used in proof of lower bound.