Algorithm Design and Analysis

วิชาบังคับก่อน: 204251 และ 206281

ผู้สอน: ตอน 1 ผศ. เบญจมาศ ปัญญางาม
ตอน 2 ผศ. ดร. จักริน ชวชาติ

บทที่ 5

อัลกอริทึมแบ่งแยกและเอาชนะ
(Divide and Conquer algorithms Part3)

Selection problem

Find the i-th order statistic in set of n (distinct) elements

- input: A set A = $\{a_1, a_2,...,a_n\}$ and i such that $1 \leq i \leq n$
- output: An element x ∈ A that is larger than exactly i –1 other elements of A

- Minimum is the first element (when i = 1)
- Maximum is the last element (when i = N)
- Median is a special case for
  - If n is odd, then $((n+1)/2)^{th}$ element.
  - If n is even then,
    - $(\lfloor \frac{n+1}{2} \rfloor^{th})$ element, lower median
    - $(\lceil \frac{n+1}{2} \rceil^{th})$ element, upper median

Selection Problem : Brute-force

Solving by Sorting e.g. Merge sort

- Idea : Just sort the numbers in ascending order and return the i^{th} element of the array
- Step 1: Sort the elements in ascending order with any algorithm of complexity O(n log n).
- Step 2: Return the i^{th} element of the sorted array

- Any selection can be done in O(1)
- So total time : O(n logn)

- Selection problem can be solved in O(n log n) by sorting

- Can we do it better?
- How many comparisons are needed to ger the max and min of n elements?

- Can we find the k^{th} smallest element faster?
  - There is a deterministic O(n) algorithm, but it is very complicated and not very practical.
  - However, there is a simple randomized algorithm, whose expected running time is O(n).

Selection Problem: Probabilistic Selection

- Solving by Probabilistic Selection : QuickSelect
- A probabilistic algorithm that uses random number and provide a measure for the quality of the approximated result.
- Sometimes better to chose randomly than to spend a lot of time to compute optimal choice.
- A practical randomized algorithm with O(n) expected running time
- Returns (a close approximation of) the k^{th} smallest element

Selection Problem: Probabilistic Selection

Idea of QuickSelect

- A divide-and-conquer algorithm to select i^{th} smallest element of $S=\{a_1, a_2, ..., a_n\}$
- Similar to quicksort, partition the input array recursively, but
  - QuickSort: works on both sides of the partition,
  - QuickSelect: works on one side
- Called prune-and-search, prune one side, just search the other side).

Selection Problem: Probabilistic Selection

Quick-select is a randomized selection algorithm based on the prune-and-search paradigm:

- Prune: pick a random element x (called pivot) and partition S into
- L - elements less than x
- E - elements equal x
- G -elements greater than x
- Search: depending on i, either answer is in E, or we need to recur on either L or G

Selection Problem: Probabilistic Selection

Idea: prune-and-search technique

1. Partition S into three subsets
2. $S_1=\{a_j|a_j<x\}$, $S_2=\{a_j \mid a_j=x\}$, $S_3=\{a_j|a_j>x\}$, $x \in S$
3. If $|S_1|>k$, search k^{th} smallest element in S_1 recursively, (prune S_2 and S_3 away)
4. Else If $|S_1|+|S_2|>k$, then return x (the k^{th} smallest element)
5. Else search $(k-(|S_1|+|S_2|))^{th}$ in S_3 recursively, (prune S_1 and S_2 away)

How to select x such that S_1 and S_3 are nearly equal?

Selection Problem: Probabilistic Selection

QuickSelect(A,p,r,k)

1. if p=r then return A[p]
2. /*find A[p,q-1] ≤ A[q] ≤ A[q+1,r]*/
3. q=RandomizedPartition(A,p,r);
4. m = q-p+1
5. if k=m then return A[q]
6. else if k<m then
7. return QuickSelect(A,p,q-1,k)
8. else
9. return QuickSelect(A,q+1,r,k-m)

The remaining sequence in each recursive call

k=5, A={7 4 9 3 2 6 5 1 8}

k=2, A={7 4 9 6 5 8 }

k=2, A={7 4 6 5 }

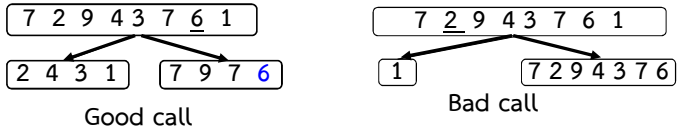
k=1, A={7 6 5 }

5

Selection Problem: Probabilistic Selection

Expected Running Time

- A recursive call of quick-select on a sequence of size s
- Good call: the sizes of L and G are each less than 3s/4
- Bad call: one of L and G has size greater than 3s/4



- A call is good with probability 1/2

Selection Problem: Probabilistic Selection

Analysis of QuickSelect

- Worst-case running time $O(n^2)$, why?
  - Unlucky and always partition : (minimum or maximum)
  - An empty side and a side with remaining elements.
  - $T(n)=\Theta(n)+ \Theta(n-1)+...+ \Theta(2)= \Theta(n(n+1)-1)= \Theta(n^2)$.
- But in average, it is good.
- Using probabilistic analysis/random variable
  - The expected running time is O(n).
  - Moreover, no particular input elicits the worst-case behavior, because of "randomness"
- Can we do better, such that O(n) in worst case?

Selection in worst-case linear time O(n)

Goal: Select the i^{th} smallest element of $S = \{a_1, a_2,...,a_n\}$

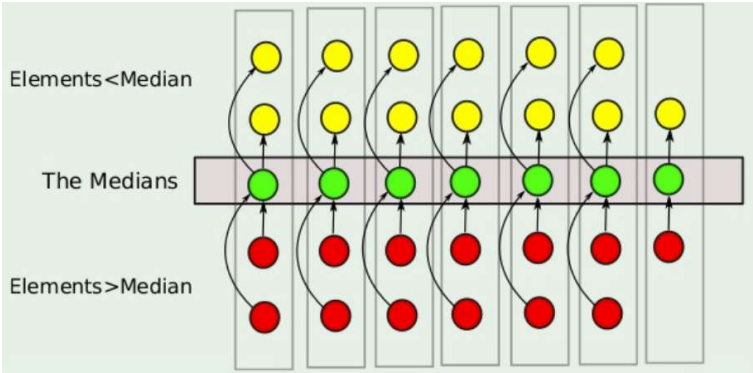
Solution: Use the so called prune-and-search technique

1. Let $x \in S$, and partition S into three subsets
2. $S_1=\{a_j|a_j<x\}$, $S_2=\{a_j \mid a_j=x\}$, $S_3=\{a_j|a_j>x\}$
3. If $|S_1|>i$, search i^{th} smallest element in S_1 recursively, (prune S_2 and S_3 away)
4. Else If $|S_1|+|S_2|>i$, then return x (the i^{th} smallest element)
5. Else search $(k-(|S_1|+|S_2|))^{th}$ in S_3 recursively, (prune S_1 and S_2 away)

- How to select x such that S_1 and S_3 are nearly equal in cardinality? Force an even search!!

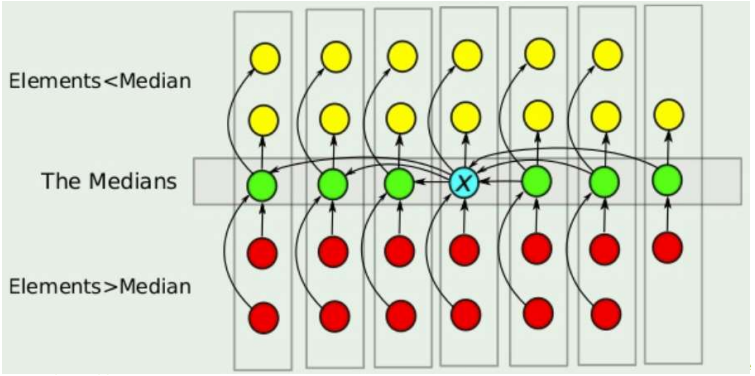
The way to select x

- Divide n elements of S into $\left\lceil \frac{n}{5} \right\rceil$ groups of 5 elements each
- Arrows go from less to greater!!



The way to select x

- Find the median of median
- Again the arrows indicate the order from less to greater



Selection Problem: Deterministic Selection

- S={12,34,0,3,22,4,17,32,3,28,43,82,25,27,34,2,19,12,5,18,20,33,16,33,21,30,3,47}

12	4	43	2	20	30
34	17	82	19	33	3
0	32	25	12	16	47
3	3	27	5	33	
22	28	34	18	21	

Selection Problem: Deterministic Selection

Find the median of each group (M)

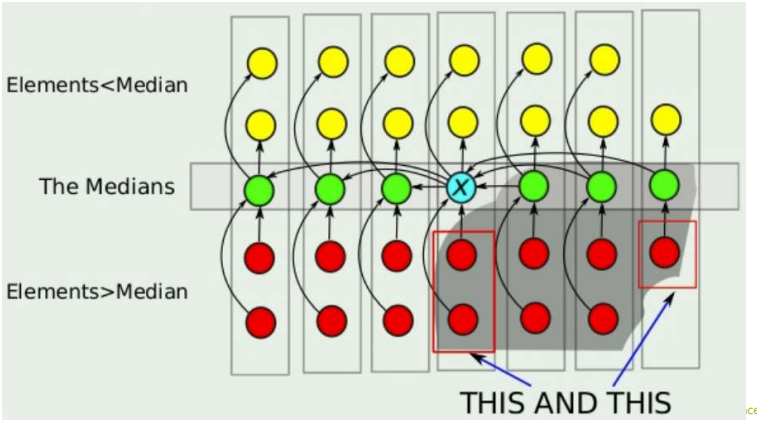
0	3	25	2	16	3
3	4	27	5	20	30
12	17	34	12	21	47
22	28	43	18	33	
34	32	82	19	33	

12 12 17 21 30 34

0	2	3	16	3	25
3	5	4	20	30	27
12	12	17	21	47	34
22	18	28	33		43
34	19	32	33		82

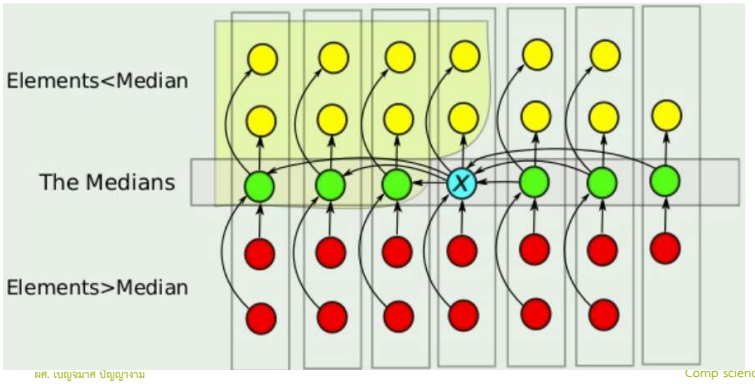
The way to select x

- At least $\frac{1}{2}\left\lceil\frac{n}{5}\right\rceil - 2$possible groups with 3 elements greater than x.



The way to select x

- At least $\frac{1}{2}\left\lceil\frac{n}{5}\right\rceil - 2$possible groups with 3 elements less than x.



- Thus, we have
- First

$$3\left(\frac{1}{2}\left\lceil\frac{n}{5}\right\rceil - 2\right) = \frac{3n}{10} - 6 \text{ elements} < x$$

- Second

$$3\left(\frac{1}{2}\left\lceil\frac{n}{5}\right\rceil - 2\right) = \frac{3n}{10} - 6 \text{ elements} > x$$

SELECT the i^{th} element in n elements

1. Divide n elements into $\left\lceil\frac{n}{5}\right\rceil$groups of 5 elements O(n)
2. Find the median of each group O(n)
3. Use SELECT recursively to find the median of x of the above $\left\lceil\frac{n}{5}\right\rceil$ medians $T(\left\lceil\frac{n}{5}\right\rceil)$
4. Partition n elements around x into S_1,S_2,and S_3 O(n)
5. If $|S_1| > i$ then search i^{th} smallest element in S_1 recursively

   Else if $|S_1|+|S_2|>i$ then return x
   Else search $(i-(|S_1|+|S_2|))^{th}$ in S_3 recursively

Step 5

- At least half of the medians in step 2 are greater or equal than x, thus at least $\frac{1}{2}\left\lceil\frac{n}{5}\right\rceil - 2$ groups contribute 3 elements which are greater or equal than x. i.e., $3\left(\frac{1}{2}\left\lceil\frac{n}{5}\right\rceil - 2\right) \geq \frac{3n}{10} - 6$
- Similarly, the number of elements less or equal than x is also at least $\frac{3n}{10} - 6$
- This, $|S_1|$ is at most $\frac{7n}{10} + 6$, similarly for $|S_3|$
- Thus, SELECT in step 5 is called recursively on at most $\frac{7n}{10} + 6$.

- Recurrence is:
- $$T(n) = \begin{cases} O(1) & if\ n < some\ value (i.e. 140) \\ T\left(\left\lceil\frac{n}{5}\right\rceil\right) + T\left(\frac{7n}{10} + 6\right) + O(n) & if\ n \geq some\ value (i.e. 140) \end{cases}$$

T(Median-of-medians)

T(recursive call)

T(Partition)

Solve recurrence by substitution

- Suppose T(n) $\leq$ cn, for some c.
- T(n) $\leq$ c $\left\lceil\frac{n}{5}\right\rceil$ + c(7n/10+6) + an
  $\leq$ cn/5+ c + 7/10 cn+6c + an
  $\leq$ 9/10 cn+7c+an
  $\leq$ cn+(-cn/10+an+7c)
- Which is at most cn if -cn/10+an+7c < 0.
  - i.e., c $\geq$10a(n/(n-70)) when n>70.
- So select n=140, and then c $\geq$20a.
- Note: n may not be 140, any integer>70 is OK.