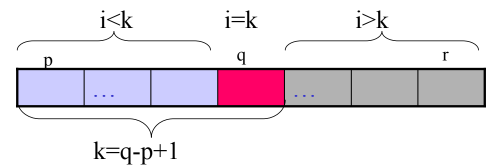
Medians and Order Statistics

- Order statistic
  - The i_{th} order statistic of a set of n elements is the i_{th} smallest element
- Selection problem
  - Find the i_{th} smallest element
    - The i_{th} element if the array is sorted on non-decreasing order
- The median
  - n is odd
    - The median is the $(n+1)/2_{th}$ element
  - n is even
    - Upper median: n/2 + 1
    - Lower median: n/2
  - For simplicity
    - We refer the median as the lower median: the $\lfloor (n+1)/2 \rfloor_{th}$ smallest element

Divide & Conquer for Selection

```
select(A[p..r], i) // return the i-th smallest element of A[p..r]
{
    if (p==r)   // i must be 1 here
      return A[p];

    q = partition(A[p..r]); // same partition as used in quick sort
    k = q-p+1;  // the k-th element now is the k-th smallest element

    if (i==k) return A[q];
    if (i<k)   return  select(A[p..q-1], i);
    if (i>k)   return  select(A[q+1..r], i-k);
}
```



Cost

- Worst case
  - $\Theta(n^2)$
  - Can we make it $\Theta(n)$?
- Average cost

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

$$\max(k-1, n-k) = \begin{cases} k-1 & if \quad k > \left\lfloor \frac{n}{2} \right\rfloor \\ n-k & if \quad k \le \left\lfloor \frac{n}{2} \right\rfloor \end{cases}$$

Average cost

- Guess that $T(n) \le cn$ for some constant c

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

$$\le \frac{2}{n}\sum_{k=\left\lfloor \frac{n}{2} \right\rfloor}^{n-1} T(k) + O(n) \le \frac{2c}{n}\sum_{k=\left\lfloor \frac{n}{2} \right\rfloor}^{n-1} k + O(n)$$

$$= \frac{2c}{n}\left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\left\lfloor \frac{n}{2} \right\rfloor - 1} k\right) + O(n)$$

$$= \frac{2c}{n}\left(\frac{1}{2}n(n-1) - \frac{1}{2}\left\lfloor \frac{n}{2} \right\rfloor\left(\left\lfloor \frac{n}{2} \right\rfloor - 1\right)\right) + O(n)$$

$$\le c(n-1) - \frac{c}{n}\frac{n-1}{2}\frac{n-3}{2} + O(n)$$

$$= \frac{3}{4}cn - \frac{3c}{4n} + O(n) \le \frac{3}{4}cn + O(n) \le cn$$

Relation between selection and median

- Any selection can be used to find the median
  - Select $\lfloor (n+1)/2 \rfloor_{th}$ smallest element
- Any median algorithm can be used for selection
  - Choose median as pivot
  - Partition the array A[p..r] into three sections:
    - T[p..q-1] all elements less than the pivot
    - T[q]: the element equal to the pivot
    - T[q+1, r] all elements greater than the pivot
  - We are either done or continue selection on one section

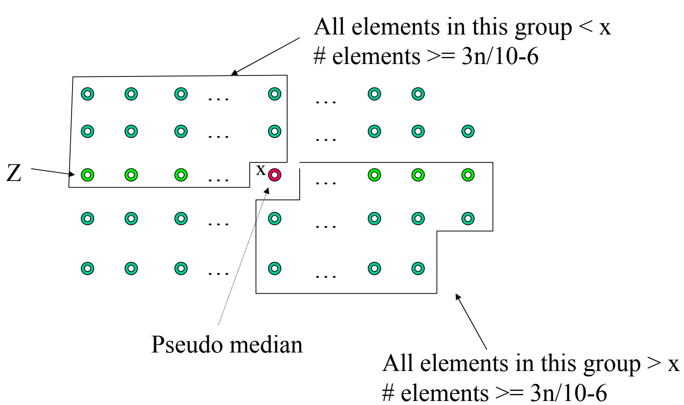
Selection using pseudomedian

```
select(A[p..r], i) {
    if (p==r) return A[p];

    x = pseudomedian(A[p..r]);
    q = partition'(A[p..r], x);
    k = q-p+1;

    if (i==k)
        return A[q];
    if (i<k)
        return  select(A[p..q-1], i);
    if (i>k)
        return  select(A[q+1..r], i-k);
}
```

```
pseudomedian(T[1..n])
{
   if (n <= 5)
      return adhocmedian(A);
   z = ⌈n/5⌉;
   for (i=1; i<=z; i++)
      Z[i] = adhocmedian(A[5i-4..min(5i,n)]);
   return select (Z[1..z], ⌊(z+1)/2⌋);
}
```

We assume the elements are distinct.

How does pseudomedian()?



At most 7n/10+6 elements are larger (smaller) than p

Theorem

- The selection algorithm used with pseudomedian finds the i-th smallest among n elements in Θ(n) in the worst case. In particular, the median can be found in linear time in the worst case.
  - There exists a const *a* such that

$$T(n) \le an + T\left(\left\lceil \frac{n}{5} \right\rceil\right) + \max\{T(m) \mid m \le \frac{7n}{10} + 6\}$$

  - We like to show $T(n) \in O(n)$

Constructive induction

- We show that *t(n)<=cn* for a constant c and all *n>=1*
  - Induction basis?
  - Induction hypothesis?
  - Induction step

$$T(n) \le T(\left\lceil \frac{n}{5} \right\rceil) + \max\{T(m) \mid m \le \frac{7n}{10} + 6\} + an$$

$$\le (\frac{n}{5} + 1)c + (\frac{7n}{10} + 6)c + an$$

$$= \frac{9c}{10}n + 7c + an$$

$$= cn + (-\frac{cn}{10} + 7c + an)$$

Strassen's algorithm: matrix multiplication

- Classic matrix multiplication takes time $\Theta(n^3)$

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

- Strassen's algorithm makes it $\Theta(n^{\lg 7}) \approx \Theta(n^{2.81})$

How does it work?

- Given $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ *and* $B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$
- Define

$$m_1 = (a_{21} + a_{22} - a_{11})(b_{22} - a_{12} + b_{11})$$
$$m_2 = a_{11} b_{11}$$
$$m_3 = a_{12} b_{21}$$
$$m_4 = (a_{11} - a_{21})(b_{22} - b_{12})$$
$$m_5 = (a_{21} + a_{22})(b_{12} - b_{11})$$
$$m_6 = (a_{12} - a_{21} + a_{11} - a_{22}) b_{22}$$
$$m_7 = a_{22}(b_{11} + b_{22} - b_{12} - b_{21})$$

- Then $C = \begin{pmatrix} m_2 + m_3 & m_1 + m_2 + m_5 + m_6 \\ m_1 + m_2 + m_4 - m_7 & m_1 + m_2 + m_4 + m_5 \end{pmatrix}$

Cost analysis

- Let *t(n)* be the time needed to multiply two *n x n* matrices.
  - $t(n) = 7t(n/2) + g(n)$ where $g(n) \in \Theta(n^2)$
  - Applying the Master theorem, we get $t(n) \in \Theta(n^{\lg 7})$