Design and Analysis of Algorithms

ผศ. ดร. สมชาย ประสิทธิ์จูตระกูล
ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย
2542

คำเตือน

เนื้อหาอันรวมถึงข้อความ ตัวเลข สัญลักษณ์ รูปภาพ และคำบรรยาย อาจมีข้อผิดพลาดแฝงอยู่ ผู้จัดทำจะไม่รับผิดชอบต่อความเสียหายทั้งทางด้านผลการเรียน สุขภาพกาย และสุขภาพจิตใดๆ อันเนื่องมาจากการใช้สื่อการเรียนนี้

Algorithm Design Techniques

Divide-and-Conquer

Outline

- General idea
- Examples
  - Mergesort, Quicksort
  - Selection
  - Exponentiation
  - Matrix Multiplication
  - Closest point
- Conclusion

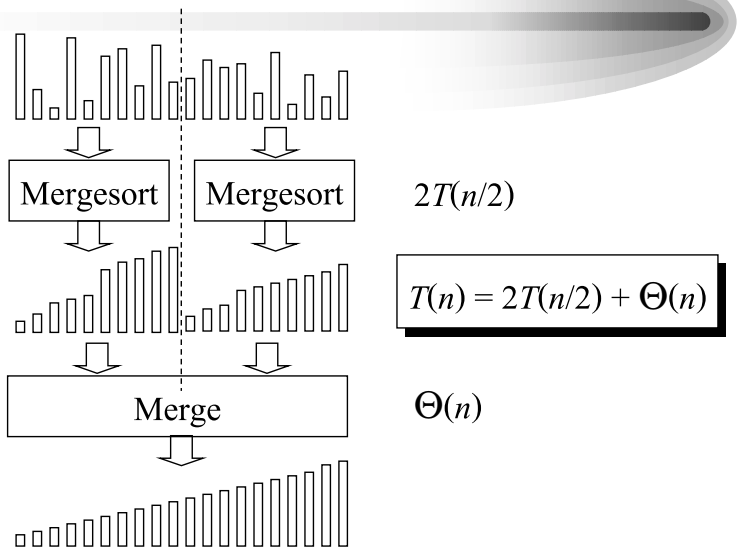
General Idea

- Divide problem instance into subinstances of the same kind
- For each subinstance (conquer)
  - solve it directly if it is trivial
  - solve it recursively otherwise
- Combine subinstance solutions to give the solution for the bigger problem instance

General Idea

```
D&Q( P )
{
  if ( P is trivial ) return Solve( P )

  Divide P into P1, P2, …, Pk

  for ( i = 1 to k )
    Si = D&Q( Pi )

  S = Combine( S1, S2, …, Sk )

  return S
}
```

Mergesort



$2T(n/2)$

$T(n) = 2T(n/2) + \Theta(n)$

$\Theta(n)$

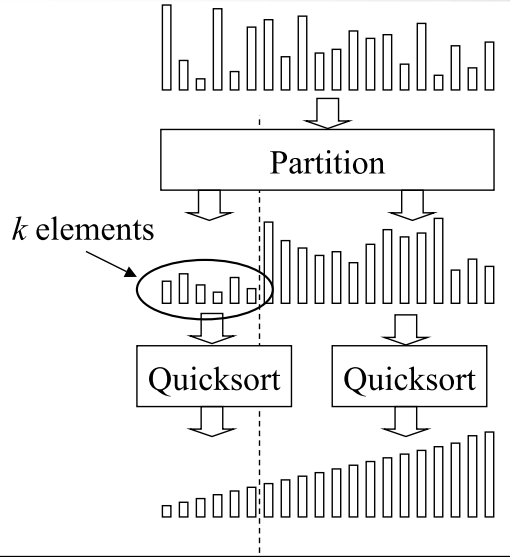
Mergesort : Analysis

- $T(n) = 2T(n/2) + \Theta(n)$
- Master method : $a = 2,\ b = 2,\ f(n) = \Theta(n)$
- $c = \log_b a = \log_2 2 = 1$
- $n^c = n^1,\ \ f(n) = \Theta(n^c) = \Theta(n)$
- $T(n) = \Theta(\ n^c \log n\ ) = \Theta(\ n \log n\ )$

Mergesort : Quiz

- What if we do not divide the array in half ?
  - 40-60
  - 20-80
  - 10-90
  - 1-99
  - random

Quicksort



Quicksort : Worst-Case Analysis

- $T(n) = T(k) + T(n\text{-}k) + \Theta(n)$
- Worst-case when $k = 1$ in every step

$$\begin{aligned} T(n) &= T(1) + T(n-1) + \Theta(n) \\ &= T(n-1) + \Theta(n) \\ &= \sum_{i=1}^{n} \Theta(i) \\ &= \Theta\left(\sum_{i=1}^{n} i \right) \\ &= \Theta\left(n^2\right) \end{aligned}$$

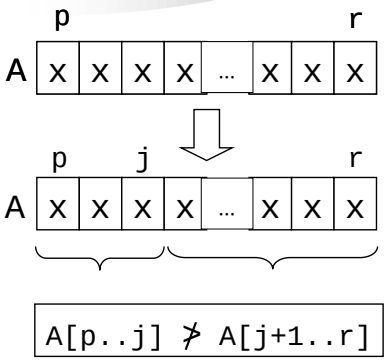
Quicksort : Best-Case Analysis

- $T(n) = T(k) + T(n\text{-}k) + \Theta(n)$
- Base-case when $k = n/2$ in every step

$$\begin{aligned} T(n) &= T(n/2) + T(n/2) + \Theta(n) \\ &= 2T(n/2) + \Theta(n) \\ &= \Theta(n \log n) \end{aligned}$$

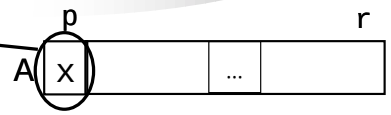
Quicksort : Partition

```
Partition( A, p, r )
{



  return j
}
```



Quicksort : Partition

```
Partition( A, p, r )
{
  c = A[p]



  return j
}
```



Quicksort : Partition

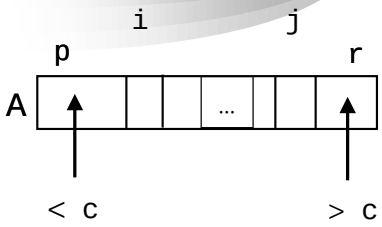
```
Partition( A, p, r )
{
  c = A[p]
  i = p-1;  j = r+1




  return j
}
```

Quicksort : Partition

```
Partition( A, p, r )
{
  c = A[p]
  i = p-1;  j = r+1
  while ( i < j ) {









  }
  return j
}
```

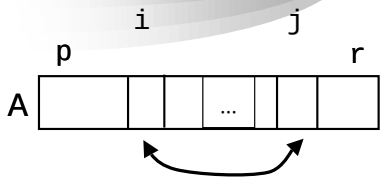
Quicksort : Partition

```
Partition( A, p, r )
{
  c = A[p]
  i = p-1;  j = r+1
  while ( i < j ) {
    repeat j = j-1
    until( A[j] ≤ c )

    repeat i = i+1
    until( A[i] ≥ c )


  }
  return j
}
```



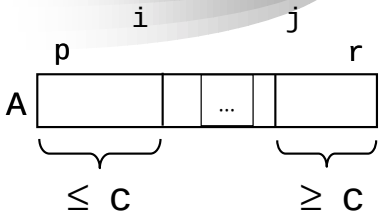
Quicksort : Partition

```
Partition( A, p, r )
{
  c = A[p]
  i = p-1;  j = r+1
  while ( i < j ) {
    repeat j = j-1
    until( A[j] ≤ c )

    repeat i = i+1
    until( A[i] ≥ c )

    if i < j
       exchange A[i], A[j]
  }
  return j
}
```



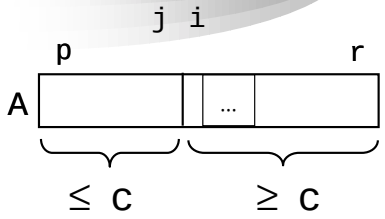
Quicksort : Partition

```
Partition( A, p, r )
{
  c = A[p]
  i = p-1;  j = r+1
  while ( i < j ) {
    repeat j = j-1
    until( A[j] ≤ c )

    repeat i = i+1
    until( A[i] ≥ c )

    if i < j
       exchange A[i], A[j]
  }
  return j
}
```

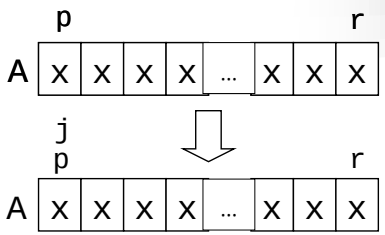


Quicksort : Partition

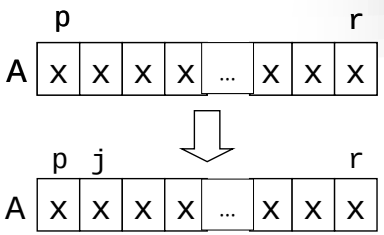
```
Partition( A, p, r )
{
  c = A[p]
  i = p-1;  j = r+1
  while ( i < j ) {
    repeat j = j-1
    until( A[j] ≤ c )

    repeat i = i+1
    until( A[i] ≥ c )

    if i < j
       exchange A[i], A[j]
  }
  return j
}
```



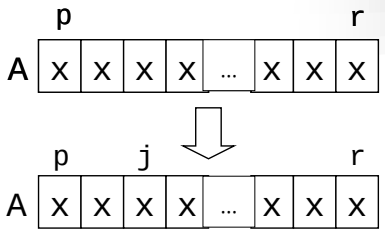
Quicksort : Partition



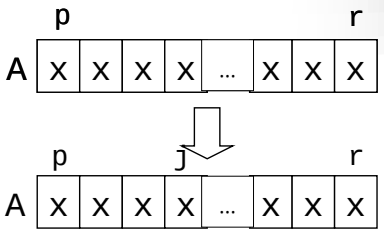
Quicksort : Partition



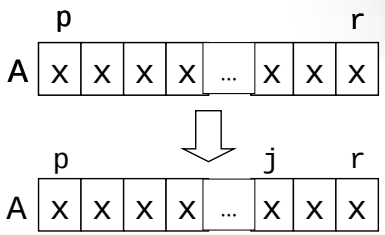
Quicksort : Partition



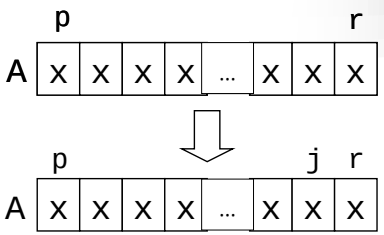
Quicksort : Partition



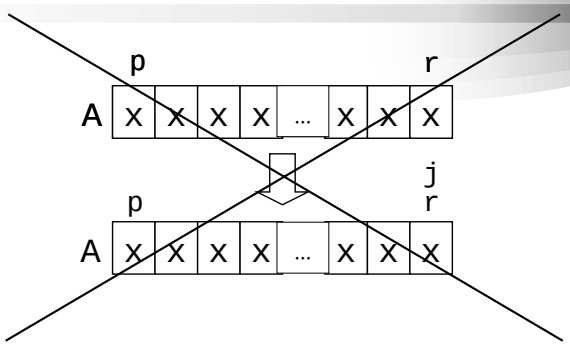
Quicksort : Partition



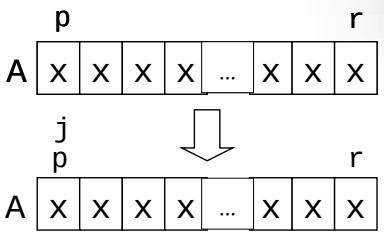
Quicksort : Partition



Quicksort : Partition



Quicksort : Partition



Randomized Partition

```
Randomized-Partition( A, p, r )
{
  i = random( p, r )
  exchange A[p], A[i]
  partition( A, p, r )
}
```

Randomized Quicksort : Analysis

- $T(n) = T(k) + T(n\text{-}k) + \Theta(n)$

$$T(n) = \frac{1}{n}\left(T(1) + T(n-1) + \sum_{j=1}^{n-1} \left(T(j) + T(n-j) \right) \right) + \Theta(n)$$

$$= \frac{1}{n} \sum_{j=1}^{n-1} \left(T(j) + T(n-j) \right) + \Theta(n)$$

$$= \frac{1}{n}\left(\sum_{j=1}^{n-1} T(j) + \sum_{j=1}^{n-1} T(n-j) \right) + \Theta(n)$$

$$= \frac{2}{n} \sum_{j=1}^{n-1} T(j) + \Theta(n)$$

Randomized Quicksort : Analysis

$$T(n) = \frac{2}{n} \sum_{j=1}^{n-1} T(j) + \Theta(n)$$

$$T(n) = \frac{2}{n} \sum_{j=1}^{n-1} T(j) + cn$$

$$nT(n) = 2\sum_{j=1}^{n-1} T(j) + cn^2$$

$$(n-1)T(n-1) = 2\sum_{j=1}^{n-2} T(j) + c(n-1)^2$$

$$nT(n) - (n-1)T(n-1) = 2T(n-1) + c'n$$

$$nT(n) = (n+1)T(n-1) + c'n$$

Randomized Quicksort : Analysis

$$nT(n) = (n+1)T(n-1) + c'n$$

$$\times \frac{1}{n(n+1)}$$

$$\frac{T(n)}{n+1} = \frac{T(n-1)}{n} + \frac{c'}{n+1}$$

telescope

$$\frac{T(n)}{n+1} = \frac{T(1)}{2} + \sum_{i=2}^{n} \frac{c'}{i+1}$$

$$\sum_{i=1}^{n} \frac{1}{n} = O(\log n)$$

$$\frac{T(n)}{n+1} = O(\log n)$$

$$T(n) = O(n \log n)$$

Quicksort

- Quicksort
  - With random input data, quicksort runs in expected O(n log n) time
- Randomized Quicksort
  - With high probability, randomized quicksort runs in O(n log n) time

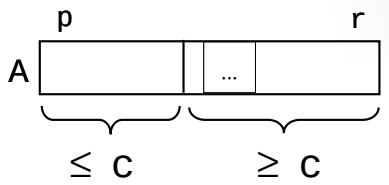
Randomization

- A general tool to improve algorithms with bad worst-case but good average-case performance
- "Robin Hood Effect"
- The worst case is still there but we almost certainly won't see it
- Stay tuned.....

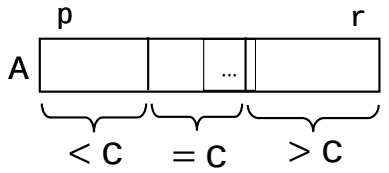
Quicksort : Quiz #1

- If all the elements are the same,
  - how does Partition divide the array ?
  - what is the running time of Quicksort ?

Quicksort : Quiz #2



$\Theta(n)$



$\Theta(n)$

Selection

- Input : A set A of n (distinct) numbers and a number i, with $1 \le i \le n$
- Output : The element x in A that is larger than exactly i-1 other elements of A

Selection Algorithms

❶ Sort the A and then pick the i^{th} element

$O(n \log n)$

❷ Build a (max) heap from the n elements of A

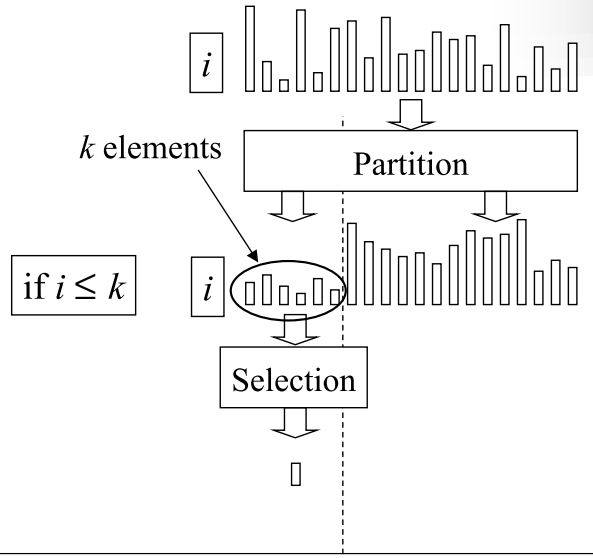
Extract-Max i times

$O(n + i \log n) = O(n \log n)$

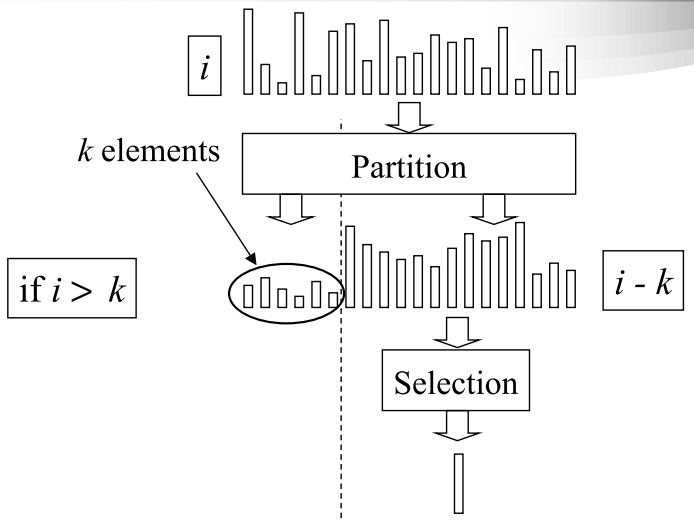
Selection

- Minimum : select the 1^{st} smallest element : $\Theta(n)$
- Maximum : select the n^{th} smallest element : $\Theta(n)$
- Select the i^{th} smallest element : $\Theta(n)$?

Selection in Expected Linear Time



Selection in Expected Linear Time



Selection in Expected Linear Time

```
Randomized-Selection( A, p, r, i )
{
  if p = r then return A[p]
  j = Randomized-Partition( A, p, r )
  k = j - p + 1
  if i < k
    return Randomized-Selection( A, p, j, i )
  else
    return Randomized-Selection( A, j+1, r, i-k )
}
```

Randomized Selection : Analysis

- $T(n) \le T(\max(k, n-k)) + \Theta(n)$

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

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

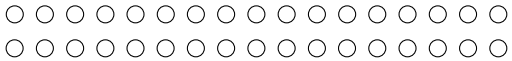
$$= \frac{2}{n}\sum_{k=\lceil n/2\rceil}^{n-1} T(k) + \Theta(n)$$

$$= O(n) \quad \text{[CLR p.188-189]}$$

Selection in Worst-Case Linear Time

- Choose Partition that guarantees a good split
- Median-of-median-of-five partitioning
  - guarantee 30%-70% split in worst case
  - selection achieves a linear time in worst case

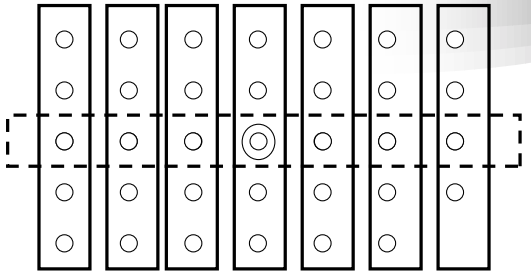
Median-of-Median-of-Five



$n = 34$

divide into $n/5$ groups of 5 elements

Median-of-Median-of-Five

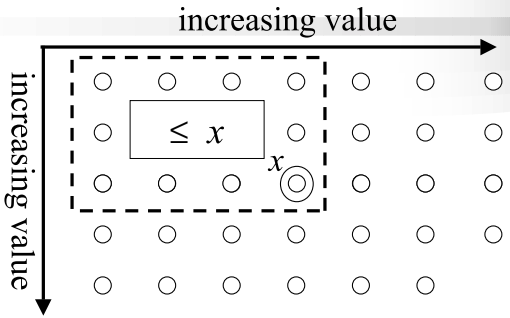


$n = 34$

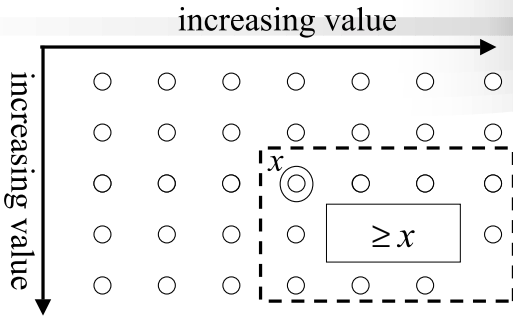
$T(n/5)+\Theta(n)$

divide into $n/5$ groups of 5 elements	$\Theta(n)$
Find the median of each of the $n/5$ groups	$\Theta(n)$
Find the median of the $n/5$ medians just found	$T(n/5)$

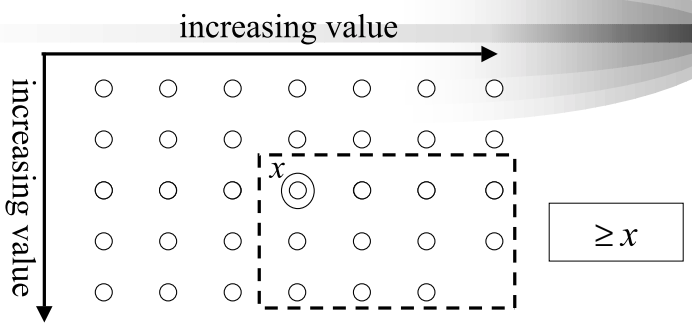
Median-of-Median-of-Five



Median-of-Median-of-Five



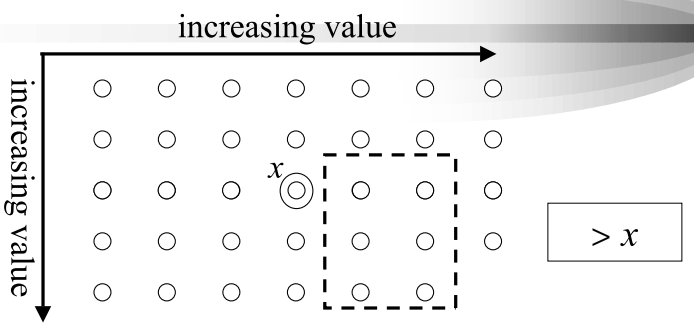
Median-of-Median-of-Five



$\geq x$

$\left\lceil \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil \right\rceil$ groups

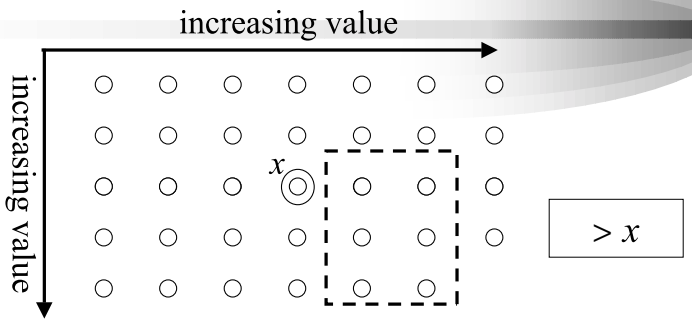
Median-of-Median-of-Five



$> x$

$\left\lceil \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil \right\rceil - 2$ groups

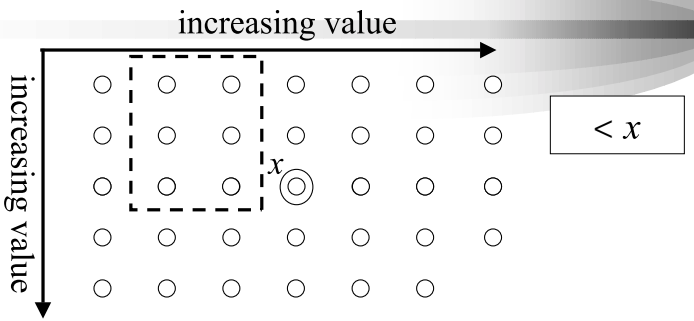
Median-of-Median-of-Five



$> x$

There are at least $3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right)$ elements that $> x$

Median-of-Median-of-Five

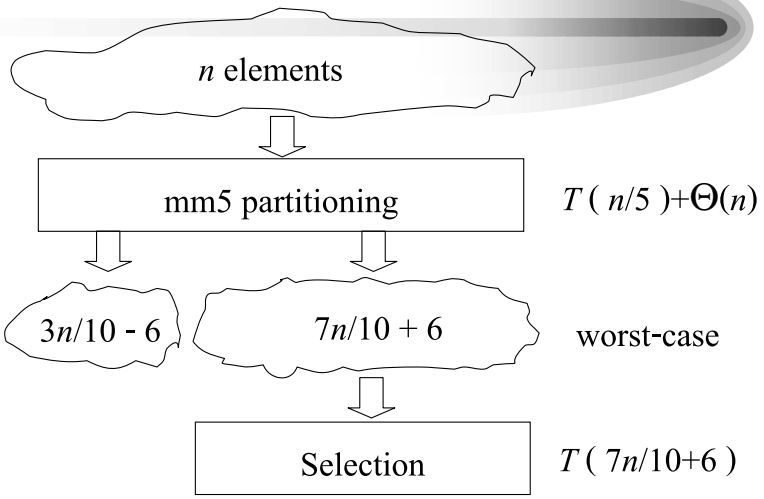


$< x$

There are at least $3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right)$ elements that $< x$

$$3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right) \geq \frac{3n}{10} - 6$$

Selection using MM5 Partitioning



$T(n) = T(7n/10+6) + T(n/5) + \Theta(n)$

Selection using MM5 Partitioning

- $T(n) = T(7n/10+6) + T(n/5) + \Theta(n)$
- $7n/10+6 + n/5 < n$ when $n > 60$
- $T(n) = \Theta(n)$

Selection : Quiz

- Can we use mm7 partitioning ?
- Can we use mm9 partitioning ?
- Can we use mm3 partitioning ?
- Should we use mm5 partitioning in Quicksort ?

Exponentiation

- Input : Three integers a, n, and k $n, k > 0$
- Output : $x = a^n \bmod k$

Very Slow Version

$$a^n \bmod k = \left(a \cdot \left(a^{n-1} \bmod k\right)\right) \bmod k$$

```
ExpoSlow( a, n, k )
{
  x = a mod k
  for ( i = 1 to n-1 )
    x = (x * a) mod k
  return x
}
```

$\Theta(\ n\ )$

$\Theta(\ 2^m\ )$, m is #bits representing the value n

Imagine if *a, n,* and *k* are 200-digit long

Divide and Conquer : $2^{92} \bmod 10$

- $2^{92} = 2^{46} \times 2^{46} = 4 \times 4 \bmod 10 = 6$
- $2^{46} = 2^{23} \times 2^{23} = 8 \times 8 \bmod 10 = 4$
- $2^{23} = 2^{11} \times 2^{11} \times 2 = 8 \times 8 \times 2 \bmod 10 = 8$
- $2^{11} = 2^{5} \times 2^{5} \times 2 = 2 \times 2 \times 2 \bmod 10 = 8$
- $2^{5} = 2^{2} \times 2^{2} \times 2 = 4 \times 4 \times 2 \bmod 10 = 2$
- $2^{2} = 2^{1} \times 2^{1} = 2 \times 2 \bmod 10 = 4$

Exponentiation : D&C

```
ExpoDC( a, n, k )
{
  if n = 1 then return a mod k
  x = ExpoDC( a, n/2, k )
  if n is even then
     return( (x*x) mod k )
  else
     return( (a*x*x) mod k )
}
```

$$T(n) = T(n/2) + \Theta(1) = \Theta(\log n)$$

$$a^n = \begin{cases} a & \text{if } n = 1 \\ \left(a^{n/2}\right)^2 & \text{if } n \text{ is even} \\ a \cdot \left(a^{n/2}\right)^2 & \text{otherwise} \end{cases}$$

$\Theta(\ m\ )$, m is #bits representing the value n

Fibonacci Number Revisited

$$F_n = F_{n-1} + F_{n-2} \quad n > 1,\ F_0 = 0, F_1 = 1$$

0 1 1 2 3 5 8 13 ...

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^1 \qquad \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^4 = \begin{bmatrix} 2 & 3 \\ 3 & 5 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^2 = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^5 = \begin{bmatrix} 3 & 5 \\ 5 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^3 = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^6 = \begin{bmatrix} 5 & 8 \\ 8 & 13 \end{bmatrix}$$

Fibonacci Number Revisited

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n = \begin{bmatrix} F_{n-1} & F_n \\ F_n & F_{n+1} \end{bmatrix}$$

$\Theta(\log n)$

Exponentiation : Quiz

- $a^{15} = a\,(a(a\,a^2)^2)^2$ requires six multiplications.
- Can you compute a^{15} using only five multiplications ?

Matrix Multiplication

- Input : two n x n matrices, A and B
- Output : matrix muiltiplication of A and B

Simple Matrix Multiplication

```
Matrix-Mult( A, B, C, n )
{
  for ( i = 1 to n ) {
    for ( j = 1 to n ) {
      C[i][j] = 0
      for ( k = 1 to n ) {
        C[i][j] += A[i][k] * B[k][j]
      }
    }
  }
}
```

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} \cdot b_{k,j}$$

The number of scalar multiplications = n^3

$\Theta(n^3)$

Winograd's Algorithm

$$X_i = \sum_{k=1}^{n/2} a_{i,2k-1} \cdot a_{i,2k}$$

$$Y_j = \sum_{k=1}^{n/2} b_{2k-1,j} \cdot b_{2k,j}$$

$$c_{i,j} = \sum_{k=1}^{n/2} (a_{i,2k-1} + b_{2k,j}) \cdot (a_{i,2k} + b_{2k-1,j}) - X_i - Y_j$$

Number of scalar multiplications = $0.5n^3 + n^2$

$$\Theta(n^3)$$

Divide-and-Conquer

$$A = \begin{bmatrix} a_{1\,1} \cdots a_{1\,1} & \cdots a_{1\,n} \\ \vdots\ A_{11}\ \vdots & A_{12}\ \vdots \\ a_{n/2\,1} \cdots a_{n/2\,n/2} & \cdots a_{n/2\,n} \\ \hline \vdots\ A_{21}\ \vdots & A_{22}\ \vdots \\ a_{n\,1} \cdots a_{1\,1} & \cdots a_{n\,n} \end{bmatrix}$$

Divide-and-Conquer

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$
$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$
$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$
$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

$$T(n) = 8T(n/2) + \Theta(n^2)$$

$$\log_2 8 = 3,\ n^2 = O(n^{3-e})$$

$$T(n) = \Theta(n^3)$$

Strassen's Algorithm

$$M_1 = (A_{12} - A_{22})(B_{21} + B_{22})$$
$$M_2 = (A_{11} + A_{22})(B_{11} + B_{22})$$
$$M_3 = (A_{11} - A_{21})(B_{11} + B_{12})$$
$$M_4 = (A_{11} + A_{12})B_{22}$$
$$M_5 = A_{11}(B_{12} - B_{22})$$
$$M_6 = A_{22}(B_{21} - B_{11})$$
$$M_7 = (A_{21} + A_{22})B_{11}$$

$$T(n) = 7T(n/2) + \Theta(n^2)$$
$$= O(n^{2.81})$$

$$C_{11} = M_1 + M_2 - M_4 + M_6$$
$$C_{12} = M_4 + M_5$$
$$C_{21} = M_8 + M_7$$
$$C_{22} = M_2 - M_3 + M_5 - M_7$$

Decimal War

- 1969 : theoretical milestone : O($n^{2.81}$)
- 1978 : O($n^{2.796}$)
- 1979 : O($n^{2.521813}$)
- 1980 : O($n^{2.521801}$)
- 1986 : O($n^{2.376}$)
- None of these algorithms is of much practical use.

Closest Points

- Input : n points on a 2D plane.
- Output : the distance of the closest pair of points

$$d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Closest Points : Brute Force

- Try all possible pairs of points
- There are $n(n\text{-}1)/2$ pairs
- $\Theta(n^2)$

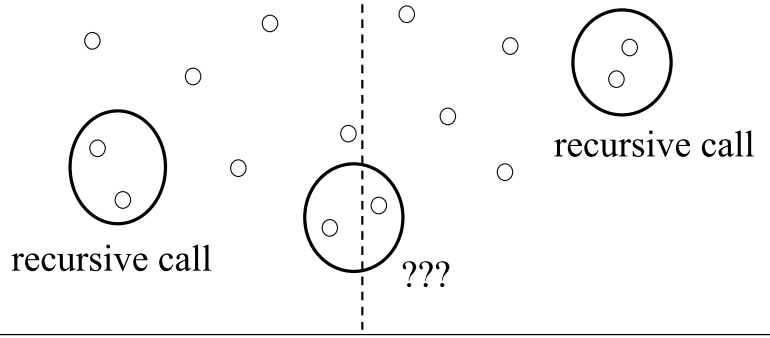
Closest Points : Divide and Conquer

$T(n) = 2T(n/2) + \Theta(\ ?\ )$

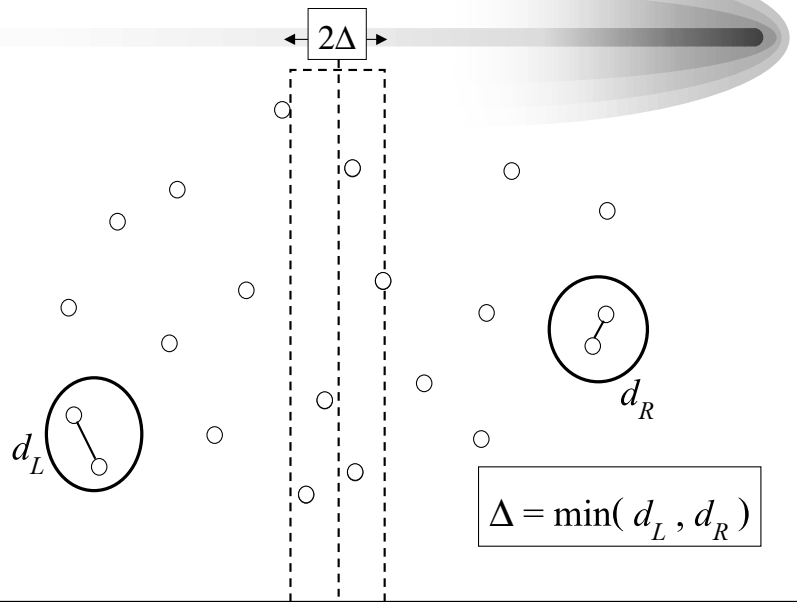
$T(n) = 2T(n/2) + \Theta(n^2)$ $= \Theta(n^2)$

$T(n) = 2T(n/2) + \Theta(n \log n)$ $= \Theta(n \log^2 n)$

$T(n) = 2T(n/2) + \Theta(n)$ $= \Theta(n \log n)$

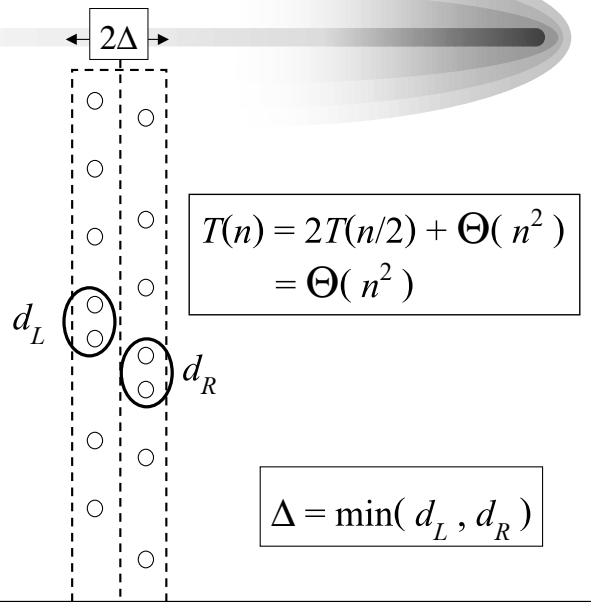


Closest Points : Divide and Conquer



$\Delta = \min(d_L , d_R)$

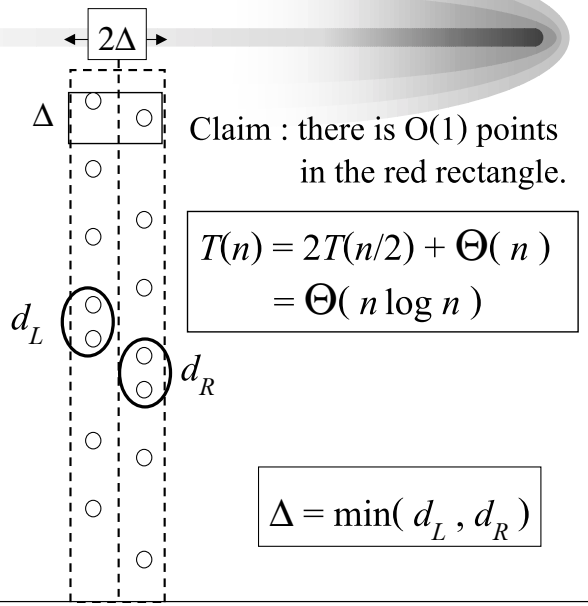
Closest Points : Divide and Conquer



$T(n) = 2T(n/2) + \Theta(n^2)$
$= \Theta(n^2)$

$\Delta = \min(d_L , d_R)$

Closest Points : Divide and Conquer

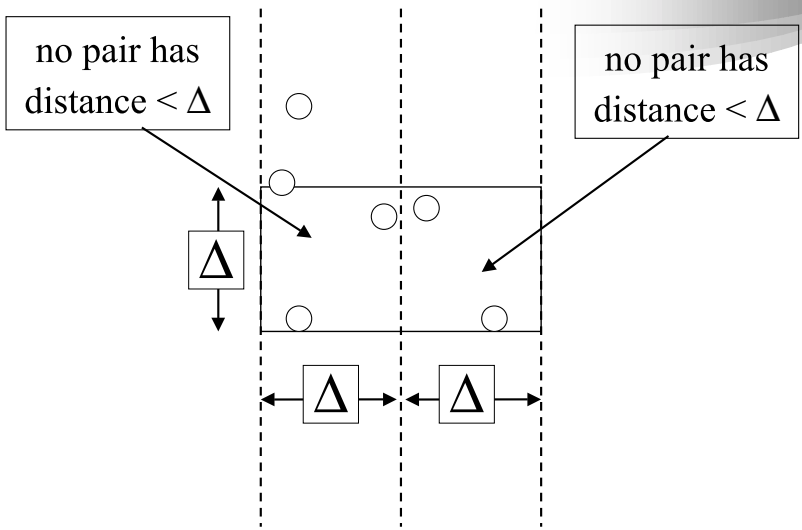


Claim : there is O(1) points in the red rectangle.

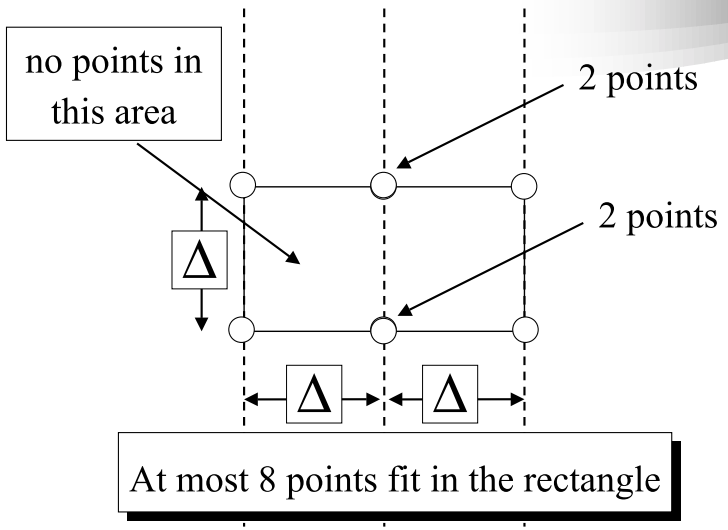
$T(n) = 2T(n/2) + \Theta(n)$
$= \Theta(n \log n)$

$\Delta = \min(d_L , d_R)$

Points in the Peep Hole



Points in the Peep Hole



Closest Points : Quiz

- How do we divide the set of points into 2 groups ?
- Which points are in the 2Δ strip ?
- How do we step down the peep hole ?

Hint: pre-processing : sorting

Convex Hull

DC-Convex-Hull

Animation

Conclusion

- Divide-Conquer-Combine
- Determine which part dominates the work from the recurrence $a\,T(n/b) + f(n)$
- Usually divide instance size in halve
- Avoid overlapped subinstances