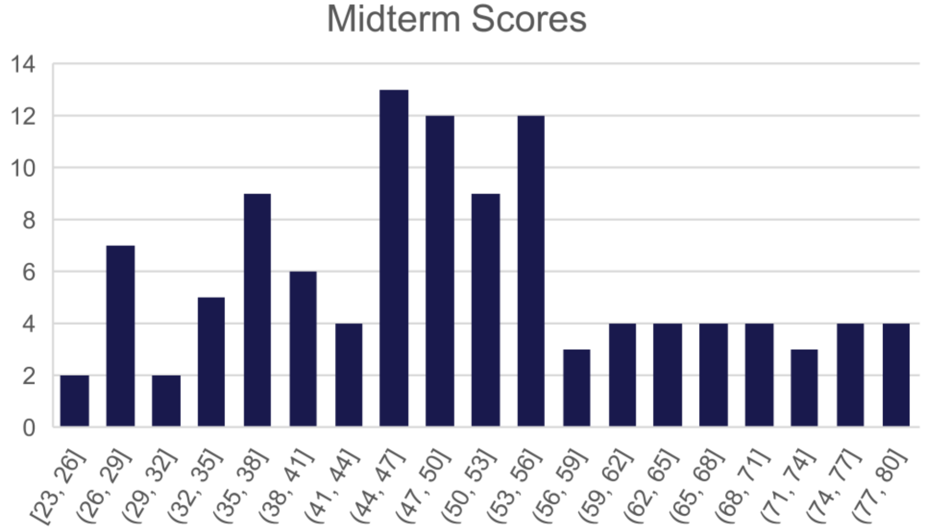
# CSE 421
# Algorithms

Lecture 15, Winter 2019

Closest Pair, Multiplication

# Announcements

- Midterm returned with solution set
- Next week,  Dynamic Programming
  – Chapter 6




Midterm Average: 50.5

# Divide and Conquer Algorithms

- Mergesort, Quicksort
- Strassen's Algorithm
- Inversion counting
- Median
- Closest Pair Algorithm (2d)
- Integer Multiplication (Karatsuba's Algorithm)
- FFT
  - Polynomial Multiplication
  - Convolution

# Median: BFPRT Algorithm

Select(A, k){

    x = BFPRT(A)

    $S_1$ = {y in A | y < x}; $S_2$ = {y in A | y > x}; $S_3$ = {y in A | y = x}

    if ($|S_2|$ >= k)

        return Select($S_2$, k)

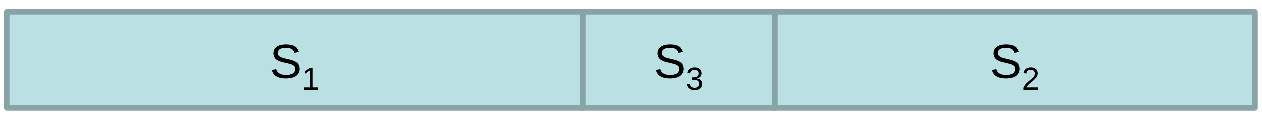
    else if ($|S_2| + |S_3|$ >= k)

        return x

    else

        return Select($S_1$, k - $|S_2|$ - $|S_3|$)

}




BFPRT(A){

    n = |A|

    Split A into n/5 sets of size 5

    M be the set of medians of these sets

    x = Select(M, n/10)    /* x is the median of M */

    return x

}

# BFPRT Recurrence

$$T(n) <= T(3n/4) + T(n/5) + c\,n$$

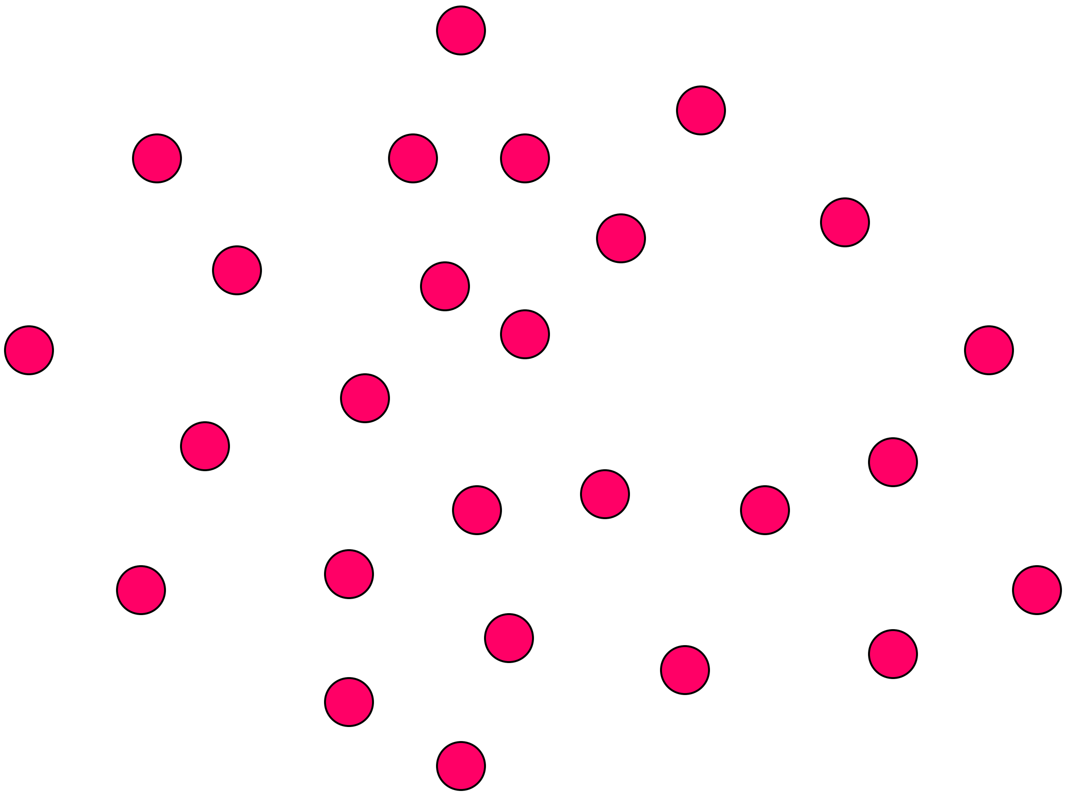
Prove that $T(n) <= 20\,c\,n$

# Median

- In practice,  select the pivot by choosing an element at random
- Heuristics such as median-of-three gives improved performance
- BFPRT is NOT a practical algorithm
- Why groups of five?
  - Odd number
  - Three does not allow linear bound to be proven
  - Seven gives a worse constant factor

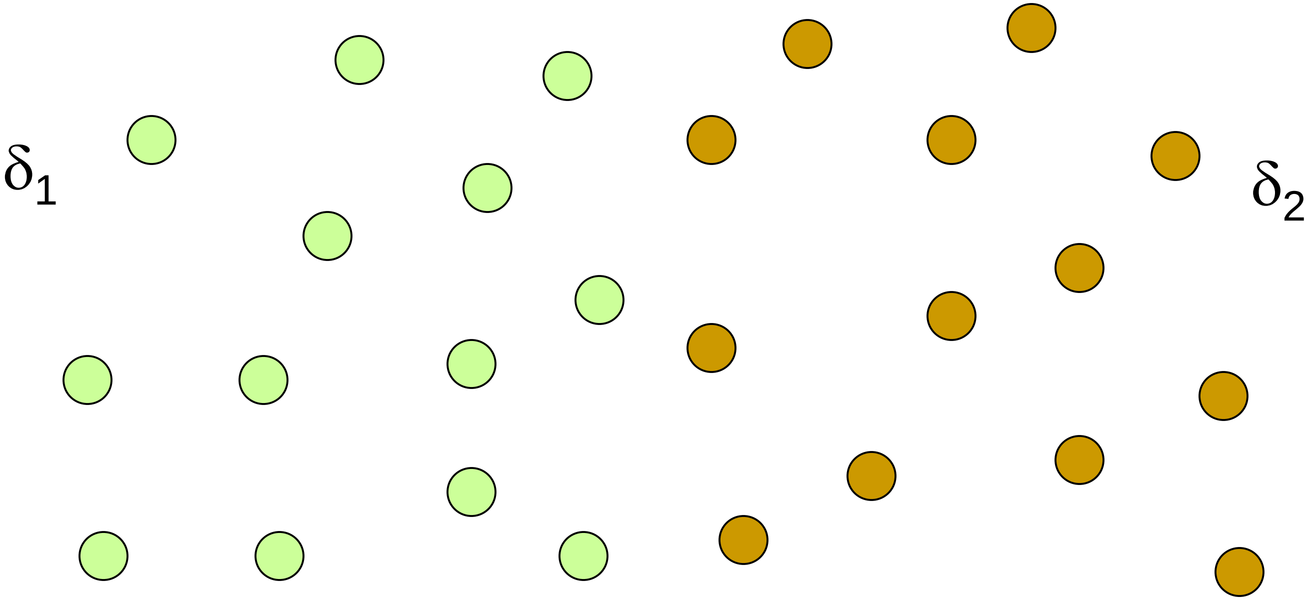
# Closest Pair Problem (2D)

- Given a set of points find the pair of points p, q that minimizes dist(p, q)



# Divide and conquer

- If we solve the problem on two subsets, does it help? (Separate by median x coordinate)



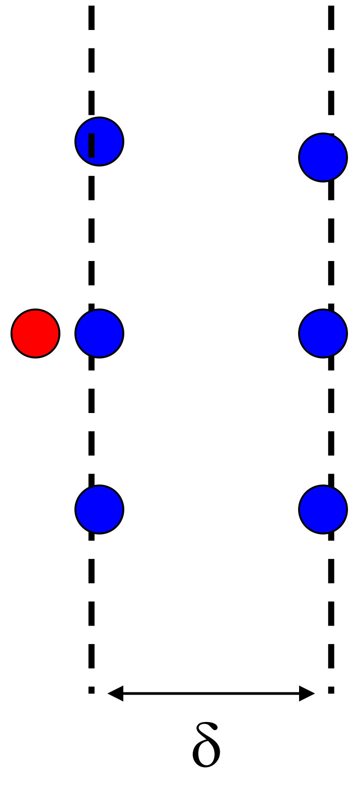
# Packing Lemma

Suppose that the minimum distance between points is at least $\delta$, what is the maximum number of points that can be packed in a ball of radius $\delta$?

# Combining Solutions

- Suppose the minimum separation from the sub problems is $\delta$
- In looking for cross set closest pairs, we only need to consider points with $\delta$ of the boundary
- How many cross border interactions do we need to test?

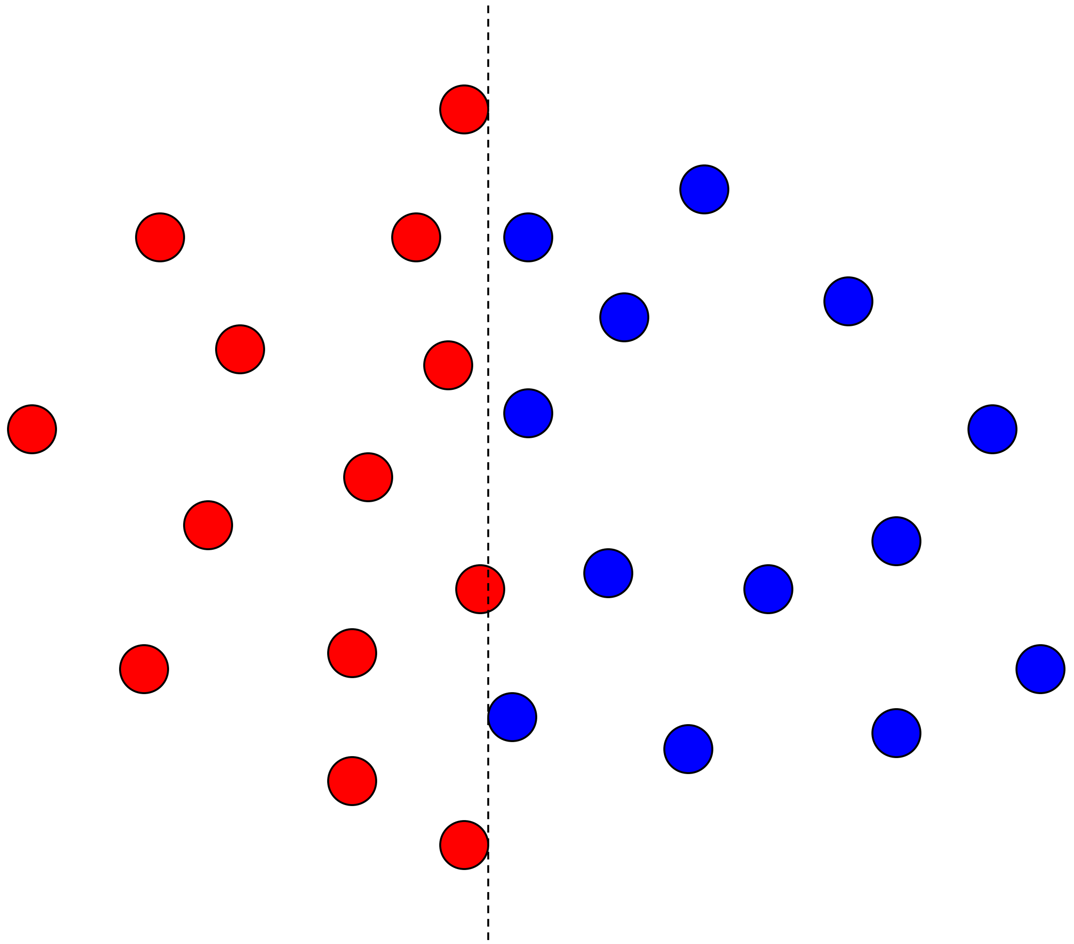
# A packing lemma bounds the number of distances to check



# Details

- Preprocessing: sort points by y
- Merge step
  - Select points in boundary zone
  - For each point in the boundary
    - Find highest point on the other side that is at most $\delta$ above
    - Find lowest point on the other side that is at most $\delta$ below
    - Compare with the points in this interval (there are at most 6)

# Identify the pairs of points that are compared in the merge step following the recursive calls



# Algorithm run time

- After preprocessing:
  - $T(n) = cn + 2\,T(n/2)$

# Integer Arithmetic

```
    9715480283945084383094856701043643845790217965702956767
+   1242431098234099057329075097179898430928779579277597977
-----------------------------------------------------------
```

Runtime for standard algorithm to add two n digit numbers:

```
    2095067093034680994318596846868779409766717133476767930
X   5920175091777634709677679342929097012308956679993010921
-----------------------------------------------------------
```

Runtime for standard algorithm to multiply two n digit numbers:

# Recursive Multiplication Algorithm (First attempt)

$x = x_1 2^{n/2} + x_0$

$y = y_1 2^{n/2} + y_0$

$xy = (x_1 2^{n/2} + x_0)(y_1 2^{n/2} + y_0)$

$\quad = x_1 y_1 2^n + (x_1 y_0 + x_0 y_1) 2^{n/2} + x_0 y_0$

Recurrence:

Run time:

# Simple algebra

$x = x_1 2^{n/2} + x_0$

$y = y_1 2^{n/2} + y_0$

$xy = x_1 y_1 2^n + (x_1 y_0 + x_0 y_1) 2^{n/2} + x_0 y_0$

$p = (x_1 + x_0)(y_1 + y_0) = x_1 y_1 + x_1 y_0 + x_0 y_1 + x_0 y_0$

# Karatsuba's Algorithm

Multiply n-digit integers x and y

Let $x = x_1 2^{n/2} + x_0$ and $y = y_1 2^{n/2} + y_0$

Recursively compute

$a = x_1 y_1$

$b = x_0 y_0$

$p = (x_1 + x_0)(y_1 + y_0)$

Return $a2^n + (p - a - b)2^{n/2} + b$

Recurrence: $T(n) = 3T(n/2) + cn$

$\log_2 3 = 1.58496250073\ldots$

# Next week

- Dynamic Programming!