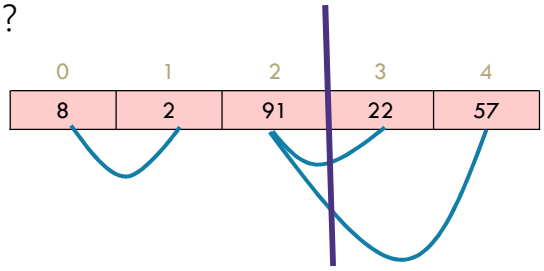
Inversions across the middle

Fix some i on the left side. How many j on the right side form inversions?



So how do we find all the "crossing inversions"

$\frac{n}{2}$ elements, each checking $\frac{n}{2}$ others, so that's time…$\Theta(n^2)$ to merge

Master Theorem

Given a recurrence of the following form, where a, b, c, and d are constants:

$$T(n) = \begin{cases} d & \text{if } n \text{ is at most some constant} \\ aT\left(\frac{n}{b}\right) + f(n) & \text{otherwise} \end{cases}$$

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + \Theta(n^2) \text{ if } n \geq 2 \\ \Theta(1) \quad \text{othwerwise} \end{cases}$$

Where $f(n)$ is $\Theta(n^c)$

If $\log_b a < c$ then $T(n) \in \Theta(n^c)$

If $\log_b a = c$ then $T(n) \in \Theta(n^c \log n)$

If $\log_b a > c$ then $T(n) \in \Theta\left(n^{\log_b a}\right)$

Pause

Lets get some intuition

$O(n \log n)$ is closest to which of these:

$O(n)$
$O(n^{1.1})$
$O(n\sqrt{n})$
$O(n^2)$

Almost there…

```
int CountInversions(A, int start, int stop)
  inversions = 0
  if(start >= stop)
    return 0
  int midpoint = (stop-start)/2 + start
  inversions += CountInversions(A, start, midpoint)
  inversions += CountInversions(A, midpoint+1, end)
  sort(A, midpoint+1, end)
  for(int i=start; i <= midpoint; i++)
    int k = binarySearch(A, midpoint+1, end, i)
    inversions += k-(midpoint+1)+1
  return inversions
```