Simple Divide and Conquer Algorithms

1 Counting Inversions

Inversion: a pair indices `i, j` such that `i < j` and `A[i] > A[j]`. Possible to solve problem in $O(n \log n)$ time by using the Mergesort algorithm:

```
count_and_split(A):
   if n == 1: return A, 0
   else:
      (B, x) = count_and_split(left)
      (C, y) = count_and_split(right)
      (D, z) = merge_and_count(B, C)

      return D, x+y+z

merge_and_count(B, C):
   i = j = inversions = 0
   D = []
   while:
      if B[i] < C[j]:
         D += B[i++]
      else:
         D += C[j++]
         inversions += (len(B) - i)
   return D, inversions
```

2 Strassen's Subcubic Matrix Multiplication

Traditional matrix multiplication is done as such:

$$X = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, Y = \begin{pmatrix} e & f \\ g & h \end{pmatrix}; X \cdot Y = Z = \begin{pmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{pmatrix}$$

The naïve implementation of this algorithm requires 8 recursive calls and runs in $O(n^3)$ time. Strassen's algorithm computes the product of two square matrices with 7 recursive calls and accomplishes a subcubic running time that is still polynomial.

The 7 products $p_1 \ldots p_7$ are:

$$\begin{aligned}
p_1 &= a(f-h) \\
p_2 &= (a+b)h \\
p_3 &= (c+d)e \\
p_4 &= d(g-e) \\
p_5 &= (a+d)(e+h) \\
p_6 &= (b-d)(g+h) \\
p_7 &= (a-c)(e+f)
\end{aligned}$$

And the final product is:

$$Z = \begin{pmatrix} p_5 + p_4 - p_2 + p_6 & p_1 + p_2 \\ p_3 + p_4 & p_3 - p_7 \end{pmatrix}$$