1.1

The following is the pseudo code for the algorithm of interest:

ALGORITHM getNumInversionsBruteForce ($A[0...n-1]$)

```
for i <- 0 to n-2 do
    for j <-        n-1 do
        if A[ ]  A[j]
            numInversions = numInversions +1
```

Given this, we see that the last line is the basic operation. Let's find the efficiency class:

$$C(n) = \sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} (n-1)-(i+1)+1 = \sum_{i=0}^{n-2} n-1-i$$

$$= (n-1)\sum_{i=0}^{n-2} 1 - \sum_{i=0}^{n-2} i = (n-1)(n-1) + \frac{1}{2}(n-2)(n-1)$$

$$= (n-1)[(n-\cancel{1}) + \frac{1}{2}(n-\cancel{2})] = (n-1)(\frac{1}{2}n)$$

$$= \frac{1}{2}n(n-1) = \Theta(n^2)$$

1.2

The following is the recurrence for the number of excecutions for the basic operation in the algorithm's best case:

$$C(n) = 2C(\tfrac{n}{2}) + \tfrac{n}{2} \quad , \quad C(1) = 0$$

$n = 2^k$

$$C(n) = 2C(n/2) + n/2 = 2(2C(\tfrac{n}{4}) + n/4) + n/2$$

$$= 2^2 C(n/2^2) + n/2 + n/2 = 2^2 C(n/2^2) + 2(n/2)$$

$$= 2^2(2C(n/8) + n/8) + 2(n/2)$$

$$= 2^3 C(n/2^3) + 3(n/2)$$

$$C(n) = 2^i C(\tfrac{n}{2^i}) + i(n/2)$$

Let $i = k$

$$C(n) = 2^i C(\tfrac{n}{2^i}) + i(n/2) = 2^k C(n/2^k) + k(n/2)$$

$$= 2^k C(1) + k(n/2) = 2^k \times 0 + k(n/2)$$

$$= k(n/2)$$

$n = 2^k \qquad k = \log_2 n$

$$C(n) = k(n/2) = \tfrac{1}{2} n \log_2 n \in \Theta(n \log n)$$

Let's consider the master theorem:

$C(n) = 2C(n/2) + n/2$

$T(n) = aT(n/b) + f(n)$

$f(n) = n/2$

$\rightarrow a = 2, b = 2, d = 1$

$b^d = 2^1 = 2$ $\boxed{\therefore a = b^d}$

Hence, $C(n) \in \Theta(n \log n)$ (consistent w/ above)

1.3 As was calculated before, the efficiency class of the first algorithm is $\Theta(n^2)$ while that of the second algorithm is $\Theta(n \log n)$.
This is reflected in the programs excecution time, as the first algorithm takes 8.89 seconds while the second one only takes 0.0156 seconds.

2.1 This algorithm considers every combination of 2 points. ~~and compares this to~~ For each combination, the remaining $n-2$ points must be compared to this line. Hence, the efficiency class may be found as follows:

$$C(n) = \sum_{k=0}^{n-3} \underbrace{\sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1}$$

From 1.1, we found that $\sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \frac{1}{2}n(n-1)$

Hence:

$$C(n) = \sum_{k=0}^{n-3} \frac{1}{2}n(n-1)$$

$$= \frac{1}{2}n(n-1) \sum_{k=0}^{n-3} 1$$

$$= \frac{1}{2}n(n-1)\left[(n-3)-(0)+1\right]$$

$$C(n) = \frac{1}{2}n(n-1)(n-2) \in \Theta(n^3)$$

2.2 The following is the recurence for quick hull's best case.

$$C(n) = 2(n/2) + \frac{1}{2}n$$

~~where x is some constant~~

Similar to 1.2, we arrive at

$$C(n) = \frac{1}{2}n\log_2 n \in \Theta(n\log n)$$

Using the master theorem, we see that

$$b^d = 2^1 = 2 = \boxed{a = b^d}$$

Hence, $C(n) \in \Theta(n\log n)$