**Problem 1.** The optimal strategy for P2 is to use binary search to determine which numbers to ask for. That is, first ask about the median $m_1$ of all numbers that are candidates for $i$. After learning whether $i < m_1$ or $i > m_1$, continue the search by asking about the median $m_2$ of all candidates smaller or greater than $m_1$, respectively. Continue this until $i$ is found. With every question, the number of remaining candidates for $i$ is cut in half so this approach finds $i$ in $O(\log n)$ steps, where $n$ is the initial number of candidates, *which can be smaller than $k$* – the range of values is practically irrelevant, only the number of candidates in that range matters. If $n = 2^p - 1$ for $p \in \mathbb{Z}$, then $i$ is found after at most $p$ guesses. Technically it is just $p - 1$ since the last guess is unnecessary: if only one candidate remains, P2 knows that $i$ has to be that number and does not need to ask about it. Example: $n = 7 = 2^3 - 1$. After the first guess, 3 candidates remain, after the second guess 1 candidate remains and the third guess (the unnecessary one) confirms that $i$ is in fact equal to that number.

1. $n = 1023 = 2^{10} - 1$, so 10 guesses are needed.
2. There are $n = \lfloor 1023/2 \rfloor = 511 = 2^9 - 1$ even numbers between 1 and 1023, so 9 guesses are needed.
3. At first there are 1023 candidates. The first guess reduces this to 511 candidates. The hint reduces this again to 255 candidates (number of even number between 1 and 511). Since $255 = 2^8 - 1$, 8 more guesses are needed to find and confirm $i$. So 9 guesses are needed in total.
4. $n = \lfloor 2047/8 \rfloor = 255 = 2^8 - 1$ since there are 255 numbers between 1 and 2047 that are divisible by 8. So 8 guesses are needed.
5. $n = \lfloor \sqrt{255} \rfloor = 15 = 2^4 - 1$ since there are 15 square numbers between 1 and 255: 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, and 225. So 4 guesses are needed.
6. $n = \lfloor \log_2 64 \rfloor + 1 = 7 = 2^3 - 1$ since there are 7 powers of 2 between 1 and 64: 1, 2, 4, 8, 16, 32, and 64. So 3 guesses are needed.

**Problem 2.** Inversions can easily be found while merging a sorted "left" subarray $B$ ($A[p..q]$) and a sorted "right" subarray $C$ ($A[q+1..r]$). Recall that the way merge works is to maintain indices $i$ and $j$ for $B$ and $C$, respectively, starting at the beginning. It compares $B[i]$ with $C[j]$, copies the smaller element to the output and increments the index that pointed to that element. This is repeated until one of the indices reaches the end of its array. Every time we find that $C[j] < B[i]$ for the current $i, j$ we know that there are inversions (all elements in $B$ have lower indices in $A$ than the elements in $C$). The number of inversions in this case is equal to the number of elements remaining in $B$, that is, $B$.length $-i + 1$, since each of those is larger than $C[j]$. You can find a detailed discussion of this here `https://www.youtube.com/watch?v=7_AJfusC6UQ` and here `https://www.youtube.com/watch?v=I6ygiW8xN7Y`. (This lecture series is a good resource in general.)

**Problem 3.**

1. Claim: $T(n) = 7 * (n - 1) + 5$

   Proof (by induction):

   Predicate $P(n)$: $T(n) = 7 * (n - 1) + 5$

   Base case ($n = 1$): $T(1) = 5 = 7 * 0 + 5 = 7 * (1 - 1) + 5$ ✓

   Induction Hypothesis: Assume for purposes of induction that $P(n)$ holds for some $n \in \mathbb{N}$

   Inductive step ($P(n) \implies P(n + 1)$):

   $T(n + 1) = T(n) + 7 =_{IH} 7 * (n - 1) + 5 + 7 = 7n + 5 = 7 * ((n + 1) - 1) + 5$ ✓

2. Claim: $T(n) = 3 * 2^{n-1}$

   Proof (by induction):

   Predicate $P(n)$: $T(n) = 3 * 2^{n-1}$

   Base case ($n = 1$): $T(1) = 3 = 3 * 1 = 3 * 2^0 = 3 * 2^{1-1}$ ✓

   Induction Hypothesis: Assume for purposes of induction that $P(n)$ holds for some $n \in \mathbb{N}$

   Inductive step ($P(n) \implies P(n + 1)$):

   $T(n + 1) = 2 * T(n) =_{IH} 2 * 3 * 2^{n-1} = 3 * 2^n = 3 * 2^{(n+1)-1}$ ✓

**Problem 4.**

1. $a = 8, b = 2, \log_b a = 3, f(n) = n^3$
   Since $f(n) \in \Theta(n^{\log_b a})$, it is $T(n) \in \Theta(n^3 \log n)$ according to case 2 of the Master Theorem.
2. $a = 2, b = 3/2, \log_b a \approx 1.71, f(n) = n^2$
   $f(n) \in \Omega(n^{\log_b a+\epsilon})$ for $0 < \epsilon \leq 0.28$ and $af(n/b) = 2(2n/3)^2 = 8n^2/9 \leq cn^2$ for $8/9 \leq c < 1$ $\forall n \geq 1$ $\rightarrow T(n) \in \Theta(n^2)$ according to case 3 of the Master Theorem.

**Problem 5.** Note that for $p = 1, r = n$ we have $q = \lfloor n/3 \rfloor$. Two recursive calls to `Rec` are made, each for a distance of the two parameters of about $q = n/3$. After the recursive calls return, the `while` loop runs. It goes through $\lfloor \sqrt{r-p+1} \rfloor$ iterations, which for $p = 1, r = n$ means $\Theta(\sqrt{n})$. These observations give us the following recurrence for the runtime of `Rec` and the value of `c` as a function of $n$: $c(n) = 2c(n/3) + \sqrt{n}$ (the Master Theorem allows us to ignore the floor function around $n/3$).

Analyze the recurrence according to the Master Theorem. $a = 2, b = 3, \log_b a \approx 0.63, f(n) = \sqrt{n} = n^{0.5}$. Since $f(n) \in O(n^{\log_b a-\epsilon})$ for $0 < \epsilon \leq 0.12$ it is $c(n) \in \Theta(n^{\log_3 2})$ according to case 1 of the Master Theorem.

**Problem 6.** We can pad both numbers with leading zeroes to ensure that their length is the same power of 2. This clearly does not change the result of the multiplication. Example: $123 * 45 = 0123 * 0045$.

**Problem 7.** All we have to do is change the condition of the `if` statement inside the `for` loop of the `Partition` function from `if A[j]` $\leq$ `x` to `if A[j]` $\geq$ `x`. This will put elements greater than `x` in the first section (`A[p]` to `A[i]`) and those less than or equal to `x` in the second section (`A[i+1]` to `A[j-1]`). The proof is completely analogous to the one for ascending order.