1. a) we can write the distribution of random variable X_i. since $p(X_i=1)=\frac{1}{n}$.

X_i	1	0
P	$\frac{1}{n}$	0

$E(X_i)=\frac{1}{n}$

b) Since each element in this array is equally possible to be selected, and only one element could be selected as pivot. the recursion equation will be when the first element is selected.

$$T(n)=X_i(T(0)+T(n)+\Theta(n))$$

if the i-th element is selected:

$$T(n)=X_i(T(i-1)+T(n-i)+\Theta(n)).$$

We can write these equations into one:

$$T(n)=\sum_{q=1}^{n}X_q(T(q-1)+T(n-q)+\Theta(n))\}$$

since only one element satisfies $X_q=1$. thus

$$E[T(n)]=E[\sum_{q=1}^{n}X_q(T(q-1)+T(n-q)+\Theta(n))]$$

c) Firstly X_q and $T(q-1)+T(n-q)$ is independent (the i-th value chosen as the pivot is independent from the running time of the subproblem). also $\sum_{q=1}^{n}[T(q-1)+T(n-q)]$

$=2\sum_{q=2}^{n-1}T(q)+\Theta(1)$.. then we get:

$$E[T(n)]=\frac{2}{n}\sum_{q=2}^{n-1}E[T(q)]+\Theta(1)+\Theta(n)$$

$$=\frac{2}{n}\sum_{q=2}^{n-1}E[T(q)]+\Theta(n).$$

d) $\sum_{k=2}^{n-1} k\lg k \le \sum_{k=1}^{\lceil \frac{n}{2}\rceil -1} k\lg k + \underline{\sum_{\lceil \frac{n}{2}\rceil}^{n} k\lg k} \le \sum_{\lceil \frac{n}{2}\rceil}^{n} k \cdot \lg n.$

$$\le \frac{(\frac{n}{2})^2}{2} \log\frac{n}{2} + \frac{(\frac{n}{2}+n)\frac{n}{2}}{2} \log n.$$

$$= \frac{n^2}{8}\log n - \frac{n^2}{8} + \frac{3}{8}n^2 \log n$$

$$= \frac{1}{2}n^2 \log n - \frac{n^2}{8}$$

e) assume $E(T(m)) = O(m\lg m)$ for all m that satisfies $m<n$ there exist constant a such that $E[T(n)] \le an\lg n$ when n is large enough. when $m=n$.

$$E[T(n)] \le \frac{2}{n}\sum_{q=2}^{n-1} aq\lg q + \Theta(n)$$

$$\le \frac{2a}{n}\left(\frac{1}{2}n^2\lg n - \frac{1}{8}n^2\right) + \Theta(n)$$

$$= an\lg n - \frac{1}{4}an + \Theta(n) \le an\lg n$$

when $-\frac{1}{4}an + \Theta(n) \le 0$. when $\frac{1}{4}a$ is larger than the constant in $\Theta(n)$, the inequality is correct. Thus $E[T(n)] = O(n\lg n)$. Also we know that quick-sort best case running time is $\Omega(n\lg n)$, so the average case is $\Theta(n\lg n)$

(Actually we can also proof $E[T(n)] = \Omega(n\lg n)$ in rigorous mathematics deduction, but for this question we just have one inequality, so we just use our known conclusion to proof the lower bound)

2. a) classical propbabillity problem.

$$P_i = \frac{(i-1)(n-i)}{C_n^3} = \frac{6(i-1)(n-i)}{n(n-1)(n-2)}$$

b) $i = \lfloor \frac{n+1}{2} \rfloor$, substitute into P_i above:

$$\frac{6 \times \frac{\lfloor n-1 \rfloor}{2} \times \frac{\lceil n-1 \rceil}{2}}{n(n-1)(n-2)} = \frac{3}{2} \times \frac{n-1}{n(n-2)}$$

$$\lim_{n \to \infty} \frac{3}{2} \times \frac{n-1}{n(n-2)} \Big/ \frac{1}{n} = \frac{3}{2}$$

c) $$\lim_{n \to \infty} \sum_{i=\frac{n}{3}}^{\frac{2}{3}n} \frac{6(i-1)(n-i)}{n(n-1)(n-2)} = \lim_{n \to \infty} \frac{6}{n(n-1)(n-2)} \sum_{i=\frac{n}{3}}^{\frac{2}{3}n} (i-1)(n-i)$$

$$= \lim_{n \to \infty} \frac{6}{n(n-1)(n-2)} \sum_{i=\frac{n}{3}}^{\frac{2}{3}n} (in - i^2 - n + i) = \frac{13}{27}$$

d). Since we know even in the best case the array is equally divided into two parts, the running time is still $\Theta(n\lg n)$, and we still need to do patition for the median-of-3 method, which will add some constant time to patition to find the median. Thus it affects only for the constant factor.

3.

A

1	2	3	4	5	6	7	8	9	10
8	2	5	3	6	1	2	5	7	2

C

0	1	2	3	4	5	6	7	8
0	1	3	1	0	2	1	1	1

⇒ C

0	1	2	3	4	5	6	7	8
0	1	4	5	5	7	8	9	10

B

1	2	3	4	5	6	7	8	9	10
		~~2~~	2						

C

0	1	3	5	5	7	8	9	10

1	2	3	4	5	6	7	8	9	10
			2					7	

0	1	3	5	5	7	8	8	10

1	2	3	4	5	6	7	8	9	10
			2			5		7	

0	1	3	5	5	6	8	8	10

1	2	3	4	5	6	7	8	9	10
		2	2			5		7	

0	1	2	5	5	6	8	8	10

1	2	3	4	5	6	7	8	9	10
1		2	2			5		7	

0	0	2	5	5	6	8	8	10

1	2	3	4	5	6	7	8	9	10
1		2	2			5	6	7	

0	0	2	5	5	6	7	8	10

1	2	3	4	5	6	7	8	9	10
1		2	2	3		5	6	7	

0	0	2	4	5	6	7	8	10

1	2	3	4	5	6	7	8	9	10
1		2	2	3	5	5	6	7	

0	0	2	4	5	5	7	8	10

1	2	3	4	5	6	7	8	9	10
1	2	2	2	3	5	5	6	7	

0	0	1	4	5	5	7	8	10

1	2	3	4	5	6	7	8	9	10
1	2	2	2	3	5	5	6	7	8

0	0	1	4	5	5	7	8	9

4. We just take the part of COUNTING-SORT, and add some codes into it: ~~We name it as count SORT.~~

```
COUNTING-SORT(A, B, k, a, b)
    let C[0..k] be a new array
    for i = 0 to k
        C[i] = 0    // Clear to 0
    for j = 1 to A.length
        C[A[j]] = C[A[j]] + 1
    for i = 1 to k
        C[i] = C[i] + C[i-1]
    // C[i] now contains the number of elements
     less or equal to i
    m = C[b] - C[a-1]    // m is exactly what we want
    for j = A.length downto 1
        B[C[A[j]]] = A[j]
        C[A[j]] = C[A[j]] - 1
```

5. We can use Radix sort with base 10 and length 3, thus the running time will be $\Theta(3(n+n)) = \Theta(n)$ in range $(0, n^3-1)$

6. a) sort the numbers using mergesort or heapsort, which will take $\Theta(n\lg n)$ worst-case time. Put the i largest elements into the output array will take $\Theta(i)$ time.

Total worst case time is: $\Theta(n\lg n + i)$

b) We use the input array to build a max-heap. For each time we need to call max-heapify, ~~and~~ which will cost ~~Θ~~ $\Theta(h)$ time in the worst senario and Extract-MAX, which will cost $\Theta(1)$ time. And we need to repeat i times. ~~Thus the total worst case time is $\Theta(i\lg n + i)$.~~ Also we need to build the max-heap, which will take $\Theta(n)$ time in worst case. ~~Thus~~

Suppose half of the i extractions are from a heap with $\geq \frac{n}{2}$ elements, so those $\frac{i}{2}$ extractions will take $\frac{i}{2}\Omega(\lg\frac{n}{2}) = \Omega(i\lg n)$ time

Thus worst-case $\Theta(n + i\lg n)$.

c) First we use Random-select to select the i-th largest value, which will cost $\Theta(n)$ time. Then we use mergesort or heapsort to sort the partitioned array, which will cost $\Theta(i\lg i)$ time. Thus the total worst case running time is $\Theta(i\lg i + n)$.

7. a) From n elements to select k elements. C_n^k.

P(k elements hash to one slot) $= (\frac{1}{n})^k (1-\frac{1}{n})^{n-k}$

thus $Q_k = (\frac{1}{n})^k (1-\frac{1}{n})^{n-k} C_n^k$.

b) If $M=k$. then k is the largest length among all slots

$$P_k = Pr\{M=k\} = Pr\{\max(X_i) = k\} \le \sum_{i=1}^{n} Pr\{X_i = k\}$$

$= nQ_k$. ~~which is the possibility~~

c) $$Q_k = (\frac{1}{n})^k (1-\frac{1}{n})^{n-k} C_n^k$$

$$= \frac{(n-1)^{n-k}}{n^n} \cdot \frac{n\cdot(n-1)\cdots(n-k+1)}{k!}$$

$$\le \frac{(n-1)^{n-k}}{n^n} \cdot \frac{n^k}{\sqrt{2\pi k}(\frac{k}{e})^k}$$

$$\le \frac{\cancel{(n-1)}\, n^{n-k} \cdot n^k}{n^n} \cdot \frac{e^k}{\sqrt{2\pi k}\, k^k} \le \frac{e^k}{k^k}$$

d) $$\lg Q_{k_0} = \frac{c\lg n(\lg e - \lg c)}{\lg\lg n} + c\lg n\left(\frac{\lg\lg\lg n}{\lg\lg n} - 1\right)$$

The max of $\frac{\lg\lg\lg n}{\lg\lg n}$ is $\frac{1}{e\log 2} \approx$ ~~0.530~~ 0.5, and converge to 0 when $n \to \infty$. For a large n. if $c>3$. then

$$\lg Q_{k_0} < -3\lg n = \lg\frac{1}{n^3} \quad \text{Thus } Q_{k_0} < \frac{1}{n^3}$$

e) $$E(M) = \sum_{i=1}^{\frac{c\lg n}{\lg\lg n}} P(M=i)\cdot i + \sum_{i=\frac{c\lg n}{\lg\lg n}+1}^{n} P(M=i)\cdot i$$

$$\leq \sum_{i=1}^{\frac{c\lg n}{\lg\lg n}} P(M=i)\cdot \frac{c\lg n}{\lg\lg n} + \sum_{i=\frac{c\lg n}{\lg\lg n}+1}^{n} P(M=i)\cdot n$$

$$= P\left(M \leq \frac{c\lg n}{\lg\lg n}\right)\cdot \frac{c\lg n}{\lg\lg n} + P\left(M > \frac{c\lg n}{\lg\lg n}\right)\cdot n$$

$$< \frac{c\lg n}{\lg\lg n} + \frac{1}{n^2}\cdot n = O\left(\frac{\lg n}{\lg\lg n}\right)$$

since $\frac{1}{n} = O\left(\frac{c\lg n}{\lg\lg n}\right)$