① [a] 4 marks HW 3

Let X be the outcome of a single roll of a fair 6 face die.

so, $X \in \{1, 2, 3, 4, 5, 6\}$

and $P(X=1) = P(X=2) = P(X=3) = P(x=4) = P(x=5)$
$= P(x=6) = \frac{1}{6}$

Now $E[X] = \sum_{x_i} x_i \, P(x_i)$

$= 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + 3 \cdot \frac{1}{6} + 4 \cdot \frac{1}{6} + 5 \cdot \frac{1}{6} + 6 \cdot \frac{1}{6}$

$= \frac{7}{2}$

[b] 4 marks

Let X_i be the indicator random variable counting when face i is up, where $i = 1$ to 6

and let X be the random variable for sum of n die rolls.

Now $E[X] = \sum_{i=1}^{6} E[X_i] \cdot i$

$= \sum_{i=1}^{6} \frac{n}{6} \cdot i \quad \left[E[X_i] = \frac{n}{6}\right]$

$= \frac{n}{6} \cdot \sum_{i=1}^{6} i$

$= \frac{n}{6} \cdot \frac{6(6+1)}{2}$

$= \frac{7}{2} n$

(2) 5 marks

let X be the random variable which denotes our gain

so $X(11) = 10$

$X(\alpha 1) = X(1\alpha) = 1$ [where α is any value between 2-6]

$X(\alpha\alpha) = -0.5$

Again $P(X=10) = \frac{1}{6} \times \frac{1}{6} = \frac{1}{36}$

$P(X=1) = \frac{1}{6} \cdot \frac{5}{6} + \frac{1}{6} \cdot \frac{5}{6} = \frac{10}{36}$

$P(X=-0.5) = 1 - \left(\frac{1}{36} + \frac{10}{36}\right) = \frac{25}{36}$

Expected value, $E[X] = P(X=10) \times 10 + P(X=1) \times 1 + P(X=-0.5) \times -0.5$

$= \frac{1}{36} \times 10 + \frac{10}{36} \times 1 + \frac{25}{36}(-0.5)$

$= \frac{5}{24}$

(3) $\boxed{a}$ 2 marks

In best case pivot will be the middle element and we have two equal partition. Random number generator will be called once for each subproblem and number of calls can be expressed as

$C(n) = 2C(n/2) + 1$

$\Rightarrow C(n) = \Theta(n)$

[b] 2 marks

In worst case alway partition into one subproblem with one less element

so, $C(n) = C(n-1) + 1$

$\Rightarrow C(n) = \Theta(n)$

[c] 2 marks

From best case and worst case complexity of random number generator calls we can conclude that average case should be $\Theta(n)$

(4) [a] 3.5 marks

With n equal keys each time the partition routine partition (A, P, q) is called, it will return q because the loop condition $A[j] \le x$ will always be true. And a subproblem with n-1 element will be generated if there were n element in the original array.

so, we can write $T(n) = T(n-1) + n$

$\Rightarrow T(n) = \Theta(n^2)$

In the same arrangment randomized quicksort will select pivot randomly but the partition routine still returns a n-1 and 0 partition. So the complexity remains same, $\Theta(n^2)$.

$\boxed{b}$ 3.5 marks

In the previous case $\boxed{a}$ the partition routine returns left partition with size $n-1$ and right partition with size 0, if the original subproblem has size n.

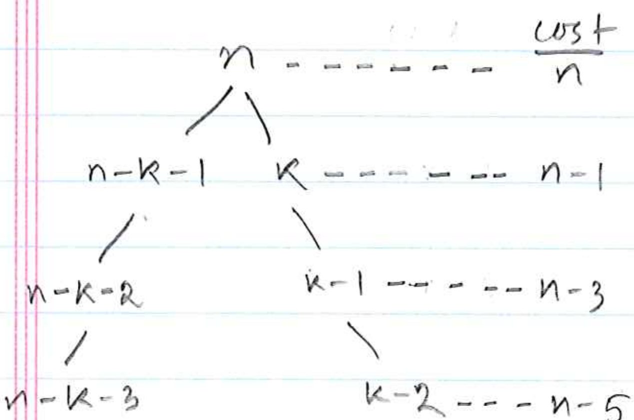
If we change $A[j] \le x$ to $A[j] < x$ in the pseudo code for partition, then left partition with size 0 and right partition with size $n-1$ will be created.

ie $T(n) = T(n-1) + 1$

$\Rightarrow T(n) = \theta(n^2)$

$\boxed{c}$ 3.5 marks

In deterministic quicksort with two distinct keys the smaller number can be selected as pivot or the larger one. But in either case there will be two subproblem with same keys after first partition. So the recursion tree will have the height of $\max(n-k-1, k)$ where k = number of element in one subarray



so, runtime

$$= \sum_{k=0}^{\max(n-k-1,k)} n-(2k+1)$$

$$\le \sum_{k=0}^{n} n-(2k+1)$$

$$= O(n^2)$$

d) 3.5 marks

```
threeWayPartition (A, p, q) {
    x = A[p]
    i = p ;
    k = p ;
    for j = p+1 to q {
        if (A[j] < x)
            i++;
            swap (A[i], A[j]);
        else if (A[j] == x)
            swap (A[j], A[k]);
            k--;
    }
    mid = min (k-j , q-k+1);
    swap (A[j... j+mid-1], A[q-mid+1 ... q]);
    return [i, k];
}
```

e) 3.5 marks

In the worst case, only one element of a distinct key is present in the array and we choose the other distinct key as pivot. The recursion tree will be look like the following:

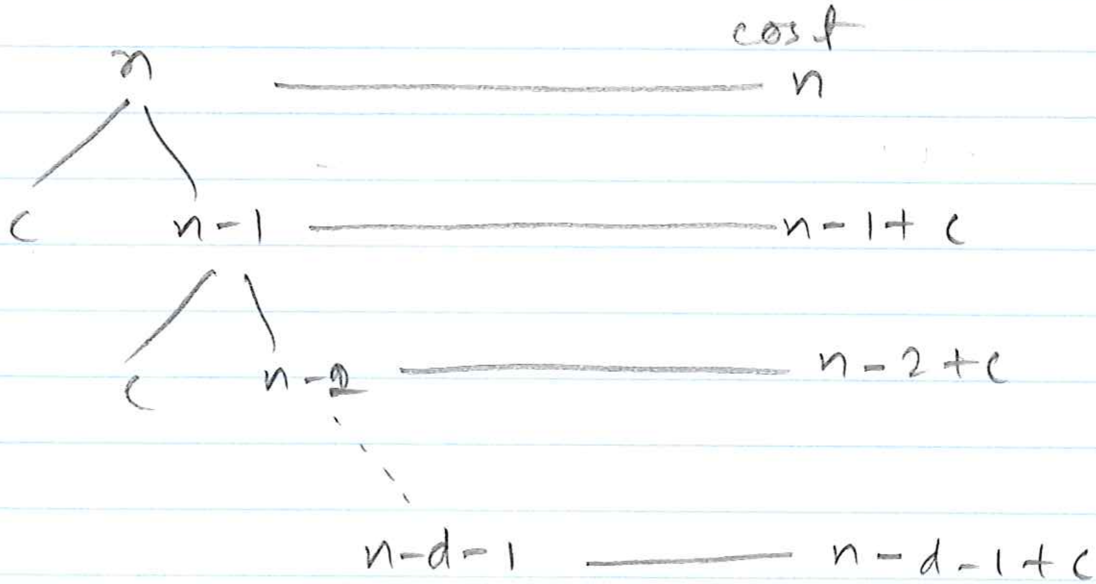
so the runtime

$= n + n - 1 + c$

$= O(n)$

$\boxed{7}$ <u>3.5 marks</u>

In the worst case we have 'd' height of the tree. Each time the partition routine is called all the element with some particular key (which is selected as pivot) will no longer exist in the subproblem. Worst case will happen when we have only one element for $d-1$ keys and $(n-d+1)$ elements have the same key, furthermore we select keys with single element as pivot each time. In such case the recursion tree will look like the following one:



$$\text{Total runtime} = n+(n-1+c)+(n-2+c)+(n-d-1+c)$$

$$= n+(n-1)+(n-2)+ \ldots +(n-d) +(d-1)c$$

$$\leq n+n+n+ \ldots +n+(d-1)c$$

$$= dn+(d-1)c$$

$$= O(dn)$$