22C:153 Homework 1
Solutions

(1) In solving this problem we make two assumptions:

(a) The prize is placed behind each curtain with equal probability.

(b) If the emcee has a choice of revealing one of two curtains, then he is equally likely to choose either.

Let A denote the event that contestant chooses curtain 1, let B denote the event that emcee reveals curtain 2, and let P_i denote the event that prize is behind i^{th} curtain (for $i = 1, 2, 3$).

Since P_1 and P_3 are disjoint and since P_2 is impossible, we have

$$Pr[B] = Pr[B \cap P_1] + Pr[B \cap P_3] = Pr[P_1] * Pr[B|P_1] + Pr[P_3] * Pr[B|P_3].$$

$= \frac{1}{3} * \frac{1}{2} + \frac{1}{3} * 1 = \frac{1}{2}$

$Pr[P_1|B] = \frac{Pr[P_1]*Pr[B|P_1]}{Pr[B]}$

$= \frac{\frac{1}{3}*\frac{1}{2}}{\frac{1}{2}} = \frac{1}{3}$

$Pr[P_3|B] = \frac{Pr[P_3]*Pr[B|P_3]}{Pr[B]}$

$= \frac{\frac{1}{3}*1}{\frac{1}{2}} = \frac{2}{3}$

So, it is a good idea to switch the choice once the emcee reveals a curtain. This is true for any initial choice of the contestant and any choice (if he has any) of the emcee.

(2) $t * Pr[X \geq t] = t * \sum_{i \geq t} Pr[X = i] = \sum_{i \geq t} t * Pr[X = i]$

$\leq \sum_{i \geq t} i * Pr[X = i]$

$\leq \sum_{i \geq 0} i * Pr[X = i]$

$\leq E[X].$

(3) Let X_{ij} be an indicator variable for the event that the pair (i, j) is an inversion. By definition, if (i, j) is an inversion we have $i < j$ and $A[i] > A[j]$. Given that $A[i] = k$, the probability that $A[j] < A[i]$ is $(k-1)/(n-1)$. Since $A[i] = k$ with probability $1/n$, the probability that (i, j) is an inversion is:

$$Pr[X_{ij} = 1] = 1/n \sum_{k=1}^{n} (k-1)/(n-1) = 1/2.$$

Let X denote the total number of inversions.

$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$

$E[X]$ denotes the expected number of inversions which is given as:

$E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}]$ (by linearity of expectation)

$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2}$

$= \frac{1}{2}\binom{n}{2}$

(4) $Pr[\text{offset} = i] = \frac{1}{n}$, for all $1 \le i \le n$. Each element of $A[i]$ can be in any of the position depending on the value of offset. So, with a probability of $\frac{1}{n}$, each element can wind up in any position in B.
Now, using this algorithm, only the circular permutations of A will be obtained, which form n of the $n!$ permutations and others cannot be obtained from this. Hence it is not uniform.

(5-a)

$$X_i = \begin{cases} 1 & \text{if } i^{th} \text{ smallest element is picked as a pivot} \\ 0 & \text{otherwise} \end{cases}$$

$E[X_i] = \frac{1}{n}$ as each element is equally likely to be picked as a pivot.

(5-b) If the q^{th} element is picked as a pivot, the running time of the algorithm is $T(n) = T(q-1) + T(n-q) + \Theta(n)$.
This can be rewritten as:
$T(n) = \sum_{q=1}^{n} X_q(T(q-1) + T(n-q) + \Theta(n))$ from (5-a). Thus,
$E[T(n)] = E\left(\sum_{q=1}^{n} X_q(T(q-1) + T(n-q) + \Theta(n))\right)$.

(5-c) By linearity of expectation we have:

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

Now note that $E[X_q(T(q-1)+T(n-q)+\Theta(n))] = 1 * \frac{1}{n} * E[T(q-1)+T(n-q)+\Theta(n)] + 0 = 1/n(E[T(q-1)] + E[T(n-q)] + \Theta(n))$.
By expanding the equation in (5-c) and using the above equation we get:

$$E[T(n)] = \frac{1}{n} * [(T(0) + T(n-1)) + (T(1) + T(n-2)) +$$
$$... + (T(n-2) + T(1)) + (T(n-1) + T(0)) + \Theta(n)]$$
$$= \tfrac{2}{n} * \sum_{q=0}^{n-1} E[T(q)] + \Theta(n).$$

(5-d) $\sum_{k=1}^{n-1} k\lg(k) \leq \sum_{k=1}^{n/2-1} k\lg(k) + \sum_{k=n/2}^{n-1} k\lg(k)$

$$\leq \sum_{k=1}^{n/2-1} k\lg(n/2) + \sum_{k=n/2}^{n-1} k\lg n$$

$$\leq (\lg n - 1)\sum_{k=1}^{n/2-1} k + \lg n \sum_{k=n/2}^{n-1} k$$

$$\leq \lg n \sum_{k=1}^{n-1} k - \sum_{k=1}^{n/2-1} k$$

$\leq \frac{1}{2}n^2\lg n - \frac{1}{8}n^2$ because $\sum_{k=1}^{n-1} k = n(n-1)/2 \leq n^2/2$ and similarly $\sum_{k=1}^{n/2-1} k \leq 1/2 * (n/2) * (n/2-1) \leq n^2/8$.

(5-e) Substituting $q\lg q$ in place of $E[T(q)]$ int (5-c) we get:

$E[T(n)] = \frac{2}{n}\sum_{q=0}^{n-1} q\lg q + \Theta(n)$

$\leq \frac{2}{n} * c\sum_{q=0}^{n-1} q\lg q + c'n$ for some $c' > 0$

$\leq \frac{2}{n} * c * (\frac{1}{2}n^2\lg n - \frac{1}{8}n^2) + c'n$

$\leq cn\lg n - \frac{c}{4}n + c'n$

$\leq cn\lg n$

only if $\frac{c}{4}n \geq c'n => c \geq 4c'$

So, if we choose c such that $c \geq 4c'$ we have $E[T(n)] = \Theta(n\lg n)$.

(6-a) The statement after the call for *Quicksort'*, does nothing but the second call for *Quicksort* in the actual algorithm.

(6-b) When the input array is in the increasing order then the worst case happens.

(6-c) The change we should make in the code is to send the smaller of both the partitions to the recursive call of *Quicksort'*. The modified algorithm is:

```
$QUICKSORT'(A, p, r)$
    while $p < r$
        $q \leftarrow PARTITION(A, p, r)$
        if $(r - q \geq q - p)$ then
            $QUICKSORT'(A, p, q-1)$
            $p \leftarrow q + 1$
        else
```

$QUICKSORT'(A, q+1, r)$

$r \leftarrow q-1$

(7) $$\sum_{i=1}^{k-2}\sum_{j=i+1}^{k-1}\frac{2}{(k-i+1)} = \sum_{i=1}^{k-2} 2 * \frac{(k-i-1)}{(k-i+1)}$$
$$\leq \sum_{i=1}^{k-2} 2$$
$$= O(n).$$

$$\sum_{i=k+1}^{n-1}\sum_{j=i+1}^{n}\frac{2}{(j-k+1)} = \sum_{j=n}^{k+2}\sum_{i=j-1}^{k+1}\frac{2}{(j-k+1)}$$
$$= \sum_{j=n}^{k+2} 2 * \frac{(j-k-1)}{(j-k+1)}$$
$$\leq \sum_{j=n}^{k+2} 2$$
$$= O(n).$$

$j-i+1$ is nothing but the size of the block from i to j, inclusive. So, the third double summation can be written as:

$$\sum_{i=1}^{k}\sum_{j=k}^{n}\frac{2}{(j-i+1)} \leq \sum_{s=1}^{n-1}\frac{2}{s} * (\text{Number of blocks of size } s)$$

When $i = k$ and $j = k$, we have a block of size 1 and there is only one block of size 1. When $i = k$ and $j = k+1$ we have a block of size 2. When $i = k-1$ and $j = k$ we have a block of size 2. So, we have 2 blocks of size 2. Similarly, number of blocks of size $s \leq s$. This is an equality if the size we are looking for is $\leq k$, otherwise it is an inequality. So,

$$\sum_{i=1}^{k}\sum_{j=k}^{n}\frac{2}{(j-i+1)} \leq \sum_{s=1}^{n-1} 2$$
$$= O(n).$$

Hence, the give equation is of order $O(n)$.

(8) In the case of graphs whose mincut is only one edge and the vertices are divided into two groups of around $n/2$ vertices, (each group can be of the form $K_{n/2}$), then the new algorithm gives exponentially bad result.

Let the two groups be A and B. At any step, a good choice means both the vertices are from A or from B. Let E_i denote such an event ie., there is a good choice in the i^{th} step. Probability that we get a min-cut at the end of the algorithm is:

$Pr[\cap_{i=1}^{n-2}] = Pr[E_i] * Pr[E_2|E_1] * Pr[E_2|E_2 \cap E_1]...$

Let after i steps the first group contains a vertices and the second group contains b vertices and assume that $a \leq b$. At this step, the probability of making a good choice is:

$$\begin{aligned} Pr[E_i | \cap_{j=1}^{i-1} E_j] &= \tfrac{a(a-1)}{2} + \tfrac{b(b-1)}{2} / \tfrac{(a+b)(a+b-1)}{2} \\ &\leq 2b(b-1)/(4b(b-1)) \\ &\leq \tfrac{1}{2} \end{aligned}$$

So, when considered for $n-1$ steps the probability of making a good choice is exponentially small.

(9) $(\frac{1}{e})^p \leq \frac{1}{100} => p \approx 5$

(10) We label each edge with both the end points. Each element in the adjacency matrix contains the number of edges between the corresponding vertices in the graph and also the labels of the edges. Whenever there is a contraction, all the edge labels of the 2nd vertex is copied into the matrix element of the first vertex, other than those between the two vertices in consideration and make the other vertex invalid in the adjacency matrix. This operation takes only $O(n)$ time.

To get back the edge from the contracted graph H, just see the label on the edge and it will give the edge in the original graph.