Question 1

Randomized quick select: The expected number of comparisons is at most 3.5n.

```
Rand-QSelect(k, S)

Select a pivot element x uniformly randomly from set S.
Find its rank in the set S (by comparing x with every other element of set S).
Let r be the rank of x.
If r = k, we report x as the output. Otherwise proceed recursively as follows:
    If r > k, then Rand−QSelect(k, S<x)
    Else Rand−QSelect(k−r, S>x).
```

This problem is quite similar to the analysis in randomized quick sort. let e_i denote i^{th} smallest element, so e_1 = smallest element of S, e_n = largest element of S, and so on... Now let us examine, when are e_i and e_j coampared during execution of Rand-QSelect(k, S).

Case 1: $i < j < k$

Pivots are choosen randomly, if e_p is choosen, such that $p > k$, then size of set S reduces, but still e_i and e_j remains in the set. If $p < i$, then also just size of set reduces, e_i and e_j will both be present in $S_{>p}$. What if $p > i,\ p < k$ and $p \neq i,\ p \neq j$? In this case, set S reduces such a way that e_i is removed, or e_i and e_j will not be compared. Also if $p = i \ or \ p = j$ e_i, e_j are definately compared. So using the result in case of Randomized quick sort we get $P(i, j \ are \ compared/(i < j < k)) = \dfrac{2}{k - i + 1}$.

Case 2: $k < i < j$

This case is exactly similar to case 1, just that here the range of interest is (k, j). So $P(i, j \ are \ compared/(k < i < j)) = \dfrac{2}{j - k + 1}$

Case 3: $i < k < j$

In this case our range of interest is (i, j), hence it should be i or j, which get choosen first from this range. So $P(i, j \ are \ compared/(k < i < j)) = \dfrac{2}{j - i + 1}$

Linearity of Expectation!

Let X be the total number of comparisons and X_{ij} be a random varaible, denoting if i and j are compared duing execution of Rand-QSelect(k, S). So

$$X_{ij} = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are compared} \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} E(X_{ij}) &= 1 * P(X_{ij} = 1) + 0 * P(X_{ij} = 0) \\ &= P(X_{ij} = 1) \end{aligned}$$

So by linearity of expectation we can say that:

$$\begin{aligned} X &= X_{12} + X_{13} + \ldots . X_{(n-1)n} \\ E(X) &= E[X_{12}] + E[X_{13}] \ldots . E[X_{(n-1)n}] \\ \text{or } E(X) &= P(X_{12} = 1) + P(X_{13} = 1) + \ldots . P(X_{(n-1)n} = 1) \end{aligned}$$

Using results from 3 cases discussed we get:

$$E(X) = \sum_{i=1}^{k-2}\sum_{j=i+1}^{k-1}\frac{2}{k-i+1} + \sum_{i=k+1}^{n-1}\sum_{j=i+1}^{n}\frac{2}{j-k+1} + \sum_{i=1}^{k}\sum_{j=k+1}^{n}\frac{2}{j-i+1}$$

$$\text{let } E(X) = A + B \text{ where } A = \sum_{i=1}^{k-2}\sum_{j=i+1}^{k-1}\frac{2}{k-i+1} + \sum_{i=k+1}^{n-1}\sum_{j=i+1}^{n}\frac{2}{j-k+1},\ B = \sum_{i=1}^{k}\sum_{j=k+1}^{n}\frac{2}{j-i+1}$$

$A = a_1 + a_2$, where a_1 and a_2 are the 2 summations given above. Now lets simplify each of these summations:

$$a_1 = \sum_{i=1}^{k-2}\frac{2}{k-i+1} * (k-1-(i+1)+1) = \sum_{i=1}^{k-2}\frac{2*(k-i-1)}{k-i+1}$$

$$a_1 = 2 * \sum_{i=1}^{k-2}(1 - \frac{2}{k-i+1}) < 2(k-1)$$

Some observations on a_2 are: (i) j takes the value n, for all possible values of i, ie $(n-1-(k+1)+1) = n-k-1$ values. (ii) j takes value $n-1$, for all $i \in (k+1, n-2)$, ie $n-k-2$ values, and so on... Hence, we can write a_2 as:

$$a_2 = \sum_{j=k+2}^{n}\frac{2}{j-k+1} * (j-k-1)$$

$$a_2 = 2 * \sum_{j=k+2}^{n}(1 - \frac{2}{j-k+1}) < 2(n-(k+2)+1) = 2n-2k-2 < 2n-2k$$

We get $A = a_1 + a_2 < 2(k-1) + 2n - 2k < 2n$, now we need to show that $B < 1.5n$.

Lemma:

Maximum value of B occurs when $k = n/2$.

Proof. Let S_t denotes sum over all values of $j \in (k+1, n)$ given $i = t$. for $i = k$ and taking all values of j we get:

$$S_k = \frac{2}{2} + \frac{2}{3} + \ldots . \frac{2}{n-k+1}$$

$$S_{k-1} = \frac{2}{3} + \frac{2}{4} + \ldots . \frac{2}{n-k+2}$$

$$\ldots .$$

$$\ldots .$$

$$S_1 = \frac{2}{k+1} + \frac{2}{k+2} + \ldots . \frac{2}{n}$$

$$B = S_1 + S_2 + \ldots . + S_k$$

So if $k = n/2$ we get:

$$B_{n/2} = 2 * (\frac{1}{2} + \frac{2}{3} + \ldots . \frac{n/2}{n/2+1} + \frac{n/2-1}{n/2+2} + \ldots . \frac{1}{n})$$

If $k < n/2$ we get:

$$B_{<n/2} = 2 * (\frac{1}{2} + \frac{2}{3} + \ldots \frac{k}{k+1} + \frac{k}{k+2} + \ldots . \frac{k-1}{n-k+1} + \ldots . \frac{1}{n})$$

If $k > n/2$:

$$B_{>n/2} = 2 * (\frac{1}{2} + \frac{2}{3} + \ldots \frac{n-k}{n-k+1} + \frac{n-k}{n-k+2} + \ldots . \frac{1}{n})$$

So the only difference when $k \neq n/2$ occurs on the middle terms. For $k < n/2$ this numerator is k, while for $k > n/2$ this is $n-k$, both of these are $\leq n/2$. Hence maximum value of B occurs when $k = n/2$. ■

Now we just need to find value of B at $k = n/2$ that would give us an upper bound on B. Expression for B is:

$$
\begin{aligned}
&B = 2 * (\frac{1}{2} + \frac{2}{3} + \dots . \frac{x}{x+1} \dots \frac{n/2}{n/2+1} + \frac{n/2-1}{n/2+2} + \dots \frac{n-x}{x} + \frac{0}{n}) \\
&B = 2 * (\sum_{i=1}^{n/2} \frac{x}{x+1} + \sum_{i=n/2} n\frac{n-x}{x}) \\
&B \leq 2 * (\frac{n}{2} + n * (\sum_{i=n/2}^{n} \frac{1}{x}) - \frac{n}{2}) \\
&B \leq 2 * (n * (\sum_{i=1}^{n} \frac{1}{x} - \sum_{i=1}^{n/2} \frac{1}{x})) \\
&B \leq 2 * n(\log n - \log(n/2)) \\
&B \leq 2n(\log 2) \\
&B \leq 1.387n
\end{aligned}
$$

Hence $A + B \leq 3.387n < 3.5n$.

Question 2

(a) We have a function $F : \{0, ..., n-1\} \rightarrow \{0, ..., m-1\}$. For $0 \leq x, y \leq n-1$,

$$F((x + y) \ mod \ n) = (F(x) + F(y)) \ mod \ m$$

The only way we have for evaluating F is to use a lookup table that stores the values of F. Unfortunately, an Evil Adversary has changed the value of $1/5$ of the table entries when we were not looking. Describe a simple randomized algorithm that given an input z, outputs a value that equals $F(z)$ with probability at least $1/2$. Your algorithm should work for every value of z, regardless of what values the Adversary changed. Your algorithm should use as few lookups and as little computation as possible.
(b) Suppose I allow you to repeat your initial algorithm k times. What should you do in this case, and what is the probability that your enhanced algorithm returns the correct answer?

(a)

Suppose that entries of the table are given by $G(x)$, for $0 \leq x \leq n-1$ after the adversary changed the values.
Algorithm:

1. Select a number x randomly between 0 and $n-1$.
2. Define the mapping,

$$H(x) = \begin{cases} (z - x) & \text{if } 0 \leq x \leq z \\ (n - 1 + z - x) & \text{if } (z-1) \leq x \leq (n-1) \end{cases}$$

3. Set $y = H(x)$
4. Return $(G(x) + G(y)) \ mod \ m$.

Lemma 1: Probability that Algorithm returns wrong answer is less than or equal to $2/5$.

Proof. Let E be the event that Algorithm returns a wrong answer.
Let F be the event that adversary changed the value at index x in the table.
Let G be the event that adversary changed the value at index y in the table.
It is clear that if the values at indices x and y were not changed by the adversary then the Algorithm would return the correct answer. So,

$$P(E) \leq P(F \cup G)$$

By Union Theorem,

$$P(E) \leq (P(F) + P(G))$$

Now,

$$P(F) = 1/5$$

because x is selected randomly from 0 to $n-1$
Also,

$$P(G) = 1/5$$

as the value of y is uniquely determined by x.
Hence,

$$P(E) \leq 2/5$$

Thus, the Algorithm returns the correct answer with probability greater than or equal to $3/5$.
Proved. ■

Time Complexity : $O(1)$

(b)

Algorithm:

1. Declare an empty set S.
2. Repeat k times.
    - Select a number x uniformly randomly between 0 and $n-1$.
    - Define the mapping,

$$H(x) = \begin{cases} (z-x) & \text{if } 0 \leq x \leq z \\ (n-1+z-x) & \text{if } (z-1) \leq x \leq (n-1) \end{cases}$$

    - Set $y = H(x)$
    - Insert $(G(x) + G(y))\ mod\ m$ into S.
3. Return the element with frequency greater than or equal to $k/2$ from the set S.
4. If there is no such element then return any random number from S.

Lemma 1: Probability that Algorithm returns wrong answer is less than or equal to $\frac{3}{2}\left(\frac{24}{25}\right)^{k/2}$.

Proof. Consider a biased coin C which shows head with a probability of $2/5$.
Let E be the event that Algorithm returns a wrong answer. Then $P(E)$ is same as the probability that there are less than $k/2$ correct values in the set S.
By Lemma 1 we can say that this probability is less than or equal to the probability that more than $k/2$ heads appear in k tosses of coin C.
Call G to be the event that more than $k/2$ heads appear in k tosses of coin C.

$$P(E) \leq P(G)$$

$$P(E) \leq \sum_{r=k/2}^{k} \binom{k}{r} \left(\frac{2}{5}\right)^{r} \left(\frac{3}{5}\right)^{k-r}$$

$$P(E) \leq \binom{k}{k/2} \left(\frac{2}{3}\right)^{k/2} \sum_{r=k/2}^{k} \left(\frac{2}{5}\right)^{r} \left(\frac{3}{5}\right)^{k-r}$$

$$P(E) \leq \binom{k}{k/2} \left(\frac{2}{3}\right)^{k/2} \left(\frac{3}{5}\right)^{k} \sum_{r=k/2}^{k} \frac{2}{3}^{r}$$

$$P(E) \leq \binom{k}{k/2} \left(\frac{2}{3}\right)^{k/2} \left(\frac{3}{5}\right)^{k} \frac{1}{1-2/3}$$

Using Stirling's approximation,

$$P(E) \leq 3\frac{4^{k/2}}{2} \left(\frac{3}{5}\right)^{k} \left(\frac{2}{3}\right)^{k/2}$$

$$P(E) \leq \frac{3}{2} \left(\frac{24}{25}\right)^{k/2}$$

Proved. ∎

Thus the probability that the Algorithm returns the correct answer is greater than or equal to

$$1 - \frac{3}{2}\left(\frac{24}{25}\right)^{k/2}$$

For $k > 140$ algorithms returns correct answer with a probability greater than 0.9.
Time Complexity : Implement S as a hash table, and hence it takes $O(1)$ time to increase the count of an element into it. Also we can find the element with largest frequency in $O(k)$ time from S in Step 3.
Thus the overall time complexity of the algorithm is $O(k)$.

Question 3

(a) Design a randomized Monte Carlo algorithm that will take $O(n^2)$ time on word RAM model per update. Write pseudocode for handling edge insertion and edge deletions.
(b) Focus on any pair (u, v). Analyse the probability that $T[u, v]$ will be incorrect at any stage. How will you reduce this error probability to less than n^{-4}.
(c) Use some probability tool to ensure that T[] will be completely correct at any stage with probability at least $1 - 1/n^2$.

(a)

Some results that are deduced from questions given in problem are

- Paths $u \to v$, that pass through (x, y) are $N_{u,x} * N_{y,v}$ (*Since for each path from $u \to v$, there are $N_{y,v}$ paths to v*).
- If an edge (x, y) is deleted then all the paths, that pass through (x, y) will be deleted. So by above result $N_{u,v}$ is decreased by $N_{u,x} * N_{y,v}$.
- If an edge (x, y) is added then all paths from $u \to v$, that pass through (x, y) will be added, or $N_{u,v}$ increases by $N_{u,x} * N_{y,v}$.

In the algorithm suggested in question there was a **serious problem**, so first lets discuss that.

Lemma 1: Total number of paths from u to v is $\leq 2^n$, where n is total number of nodes.

Proof. A path is a sequence of vertices $\langle (u =)x_1, x_2, \ldots, x_k(= v)\rangle$, such that $(x_i, x_{i+1}) \in E$ for each $1 \leq i < k$ and no vertex is repeated in this sequence. Note that length of sequence can vary from $2 \to n$. So the total number of paths from $u \to v$ is given by: $N_{u,v} = \sum_{i=2}^{n} \binom{n}{i}$ *(there is only 1 way to order these i nodes, because if we consider 2 permutation, there would be atleast 1 pair of nodes (x, y) such that in one permutation x appears after y and in other y appears after x. That would mean that there is a cycle, as we have path $x \to y$ and $y \to x$. which contradicts to the fact that G is DAG.)*. Consider:

$$(1 + x)^n = \sum_{i=0}^{n} \binom{n}{i} x^i \textit{ substituting } x = 1 \textit{ we get :}$$
$$2^n = \sum_{i=0}^{n} \binom{n}{i}$$
$$\textit{or } N_{u,v} \leq 2^n$$

■

Result: On a word ram model of computation, product of $N_{u,x}, N_{y,v}$, would take O($\frac{n}{\log n}$) worst case time.

Proof. From Lemma 1, we know that $N_{a,b}$ can be close to 2^n in worst case. On a word ram model computation involving $\log n$ bits take $O(1)$ time. But here maximum number of bits can be $\log(2^n) = O(n)$ bits. which would take $O(\frac{n}{\log n})$, for each computation. ■

Serious Problem: Since computation of product in word ram model takes $O(\frac{n}{\log n})$ time, each update will take $O(\frac{n^3}{\log n})$ time, instead of just $O(n^2)$.

Solution Sketch

Main problem that we are facing is that $N_{a,b}$ can be very large, which wouldn't take $O(1)$ time on word ram model. So one thing we can do is: perform all operations in modulo p, *where p is a prime which is of order polynomial in n, now since it is polynomial in n, all computation involving p can be done in $O(1)$ time in word ram model*. More presisely perform all updated in $N_{a,b}$, weather addition, deletion, or computing product $N_{a,b} * N_{x,y}$ preform in $\mod p$.

Note:

$$N[u][v] = N_{u,v} \mod p$$

All the queries(deletion or addition of nodes) can now be handled by 3 results given on top on this answer, and using them in $\mod p$. Underlying algorithm is given by:

Algorithm:

```
p := prime choosen randomly and uniformly from (2,t)
N := n x n matrix storing number of edges modulo p
M := n x n matrix
q := query

if((x,y) deleted)
                                        # deletion
    for u := 1 to n
        for v := 1 to n
            M[u][v] = (N[u][v] − N[u][x]*N[y][v]) mod p
    N = M
else
                                        # addition
    for u := 1 to n
        for v := 1 to n
            M[u][v] = (N[u][v] + N[u][x]*N[y][v]) mod p
    N = M

procedure Query_for_reachability(u,v)
    if N[u][v] is 0
        return false
    else
        return true
```

Complexity analysis

For each update, we change the value of $N(a, b)$ for all nodes. and then finally update the matrix N to this new matrix M. Total computation required in single update is given by $O(n^2) * O(time\_for\_product)$ *(as there are n^2 pairs and on each pair we do a product operation)*. Now since we are doing computation in $\mod p$, it is guaranteed that time_for_product would be $O(1)$, since p is polynomial in n. Hence, $T = O(n^2)$.

(b)

The only case when the algorithm in (a) returns the wrong answer for pair (u, v) is when the number of paths between u, v are non-zero and the algorithm computes that they are 0. Call this event to be E.
Now, call the event that $(N[u, v] - N[u, x] * N[y, v])\ mod\ p = 0$ or $(N[u, v] + N[u, x] * N[y, v])\ mod\ p = 0$ to be F.
Let $\pi(x)$ denotes the no of primes less than or equal to x.
Let $X(a)$ be the no of primes which divide a.

Clearly,

$$a \geq 2^{X(a)}$$

Thus,

$$X(a) \leq \log(a)$$

Lemma 1: Probability that $A\ mod\ p = 0$ when p is a random prime no from $(2, t)$ is less than or equal to $\frac{log(A}{\pi(t)}$.

Proof. Since p is selected randomly uniformly from all primes from $(2, t)$,

$$P(A\ mod\ p = 0) = \frac{X(A)}{\pi(t)}$$

Thus,

$$P(A\ mod\ p = 0) = \frac{\log(A)}{\pi(t)}$$

■

Since,

$$N_{u,v} \pm N_{u,x} * N_{y,v} \leq 2^n$$

by Lemma 1 of Part (a).
Let $P(D)$ be the probability that one of $(N[u,v] - N[u,x] * N[y,v])\ mod\ p = 0$ or $(N[u,v] + N[u,x] * N[y,v])\ mod\ p = 0$.
By Lemma 1,

$$P(D) \leq \frac{log(2^n)}{\pi(t)}$$

Call the probability that $T[u, v]$ will be incorrect at any stage to be $P(T)$

$$P(T) = P(D)$$

$$P(T) \leq \frac{n}{\pi(t)}$$

Now,

$$\pi(t) \approx \frac{t}{log(t)}$$

Thus,

$$P(T) \leq \frac{n * log(t)}{t}$$

Take $t > 5 * n^5 log(n)$. Then,

$$P(T) \leq n^{-4}$$

Thus for $t > 5 * n^5 log(n)$, probability that $T[u, v]$ will be wrong at any stage is less than or equal to n^{-4}.

(c)

Let E be the event that $T[]$ will be completely correct at any stage.
Let $E_{u,v}$ be the event that $T[u, v]$ will be wrong at any stage.
Clearly,

$$P(E) = 1 - \bigcup_{u,v \epsilon V} P(E_{u,v})$$

By part (b) and Union Theorem,

$$P(E) \geq 1 - \sum_{u,v \epsilon V} n^{-4}$$

or,

$$P(E) \geq 1 - n^2 * n^{-4}$$

Thus,

$$P(E) \geq 1 - n^{-2}$$

for $t > 5 * n^5 \log(n)$