1. Compute the probability that a SkipList with \( n \) elements has at least \((\log_2 n)^2\) many levels. (15 points)

**Solution:** On page 8 of the lecture slides on Skip Lists it was found that the expected number of elements on level \( i \), in notation \( E[\#(i)] \), is given by

\[
E[\#(i)] = \frac{n}{2^i}
\]

It was also found that the Markov inequality implies that the probability of having at least one element on level \( i \), i.e., \( P(\#(i) \geq 1) \), satisfies

\[
P(\#(i) \geq 1) \leq \frac{E[\#(i)]}{1} = \frac{n}{2^i}
\]

Letting \( i = (\log_2 n)^2 \) we get

\[
P(\#((\log_2 n)^2) \geq 1) \leq \frac{E[\#((\log_2 n)^2)]}{1} = \frac{n}{2^{(\log_2 n)^2}}
\]

\[
= \frac{n}{2^{\log_2 n \log_2 n}} = \frac{n}{n^{\log_2 n}} = \frac{1}{n^{\log_2 n - 1}}
\]

2. Assume that in the deterministic algorithm for Order Statistic we split numbers into groups of \( 2k + 1 \) many elements (so, the particular case we did in class is for \( k = 2 \) because we split numbers into groups of \( 2 \times 2 + 1 = 5 \) elements.) Derive an estimate of the run time \( T_k(n) \) of the algorithm for general \( k \) and show that \( T_k(n) \) satisfies \( T_k(n) \leq 10 C_k n \) for all \( k > 1 \) where \( C_k \) is a constant such that the overhead of the recursion is bounded by \( C_k n \). (20 points)
Solution: See the picture below:

Assume we have $n$ elements, all distinct. We split them into groups containing $2k + 1$ elements each and sort element WITHIN each group by brute force in linear time (each group can be sorted by any algorithm in constant time; here $k$ is a fixed parameter and only $n$ is the variable). We sort them so that elements increase from top going to the bottom. We then call recursively the algorithm applied to the middle row to find its median. The row contains $n/(2k + 1)$ many elements so this part takes $T(n/(2k + 1))$ time.

Having found the median of the middle row (the orange dot) we now permute the columns so that all columns with the middle element smaller equal to the pivot are on the left and all elements larger than the pivot are on the right (this is not necessary but it helps explain the picture). We now split the original array into two parts, one being all elements smaller or equal than the pivot and the other all elements larger than the pivot. We now estimate the minimal and the maximal sizes of the two sides of the partition.

Referring to the picture, we now notice that all elements in the left small pink top left box are smaller than the pivot, because they are all smaller than the middle element of their column which is in turn smaller than the pivot. This box has the bottom side containing $\frac{n}{2(2k+1)}$ many elements (one half of the middle row). the vertical side contains $(k+1)$ many elements so in total the box contains $\frac{n(k+1)}{2(2k+1)}$ many elements all smaller than the pivot. The same
number of elements are also contained in the bottom right box and they are all bigger or equal to the pivot. Thus, if we split all of \( n \) elements into two subsets, one of all elements smaller than the pivot and another larger than the pivot, each side contains at least \( \frac{n(k+1)}{2(2k+1)} \) and at most \( n - \frac{n(k+1)}{2(2k+1)} \) many elements. So after calling the algorithm recursively on one side of the partition, the run time will be at most
\[
T\left(n - \frac{n(k+1)}{2(2k+1)}\right) = T\left(n \frac{3k+1}{2(2k+1)}\right)
\]
So the recursion satisfied by \( T \) is
\[
T(n) \leq T\left(\frac{n}{2k+1}\right) + T\left(n \frac{3k+1}{2(2k+1)}\right) + C_k n
\]
where \( C_k n \) is the cost of splitting the array and sorting the columns by brute force. We are now looking for a constant \( K \) such that \( T(n) \leq K C_k n \). Assume we have such a constant; then to be able to prove such a bound by induction we need the following property
\[
K C_k \frac{n}{2k+1} + K C_k n \frac{3k+1}{2(2k+1)} + C_k n \leq K C_k n
\]
Dividing both sides by \( n C_k \) we get the following condition for \( K \):
\[
K \frac{1}{2k+1} + K \frac{3k+1}{2(2k+1)} + 1 \leq K
\]
Multiplying both sides by \( 2k+1 \) we obtain
\[
K + K \frac{3k+1}{2} + 2k + 1 \leq K (2k+1)
\]
This is equivalent to
\[
K \geq \frac{4k + 2}{k-1}
\]
Thus, the run time satisfies
\[
T(n) \leq \frac{4k + 2}{k-1} C_k n
\]
We can now prove that fact by induction on \( n \). Assume the statement is true for \( \frac{n}{2k+1} < n \) and \( \frac{n(3k+1)}{2(2k+1)} < n \), then, substituting in
\[
T\left(\frac{n}{2k+1}\right) + T\left(\frac{n(3k+1)}{2(2k+1)}\right) + C_k n
\]
we obtain
\[ T\left(\frac{n}{2k+1}\right) + T\left(n\frac{3k+1}{2(2k+1)}\right) + C_k n \leq C_k \frac{4k+2}{k-1} \frac{n}{2k+1} + \frac{4k+2}{k-1} C_k n \frac{3k+1}{2(2k+1)} + C_k n \]
\[ = C_k n \left( \frac{1}{2k+1} \frac{4k+2}{k-1} + \frac{4k+2}{k-1} \frac{3k+1}{2(2k+1)} + 1 \right) \]
\[ = C_k n \frac{4k+2}{k-1} \]
which proves induction step. Thus \( T(n) \leq C_k n \frac{4k+2}{k-1} \) for all \( n \) IF \( k > 1 \). The algorithm does not work for \( k = 1 \), i.e., if we split the numbers into groups of \( 2 \cdot 1 + 1 = 3 \) many elements because for \( k = 1 \) inequality 1 becomes
\[ K + K \frac{4}{2} + 2 + 1 \leq 3K \]
i.e, \( 3K + 3 \leq 3K \) which is not satisfied by any \( K \).

3. Assume that you draw independently and uniformly randomly \( n \) points \( x_1, \ldots, x_n \) from the unit ball in \( n \) dimensional space \( \mathbb{R}^n \). Show that with probability of at least \( 1 - 1/n \) it holds that the sum \( \sum_{k=1}^{n} |x_k| \) of lengths of the corresponding vectors satisfies
\[ n \geq \sum_{k=1}^{n} |x_k| \geq n - 2 \ln n \quad \text{(15 points)} \]

**Solution:** On page 13 of the lecture slides on the Johnson Lindenstrauss Theorem we proved that if we draw independently and uniformly \( n \) points \( x_1, \ldots, x_n \) from the unit ball in a vector space of dimension \( d \) and if \( \varepsilon \) is an arbitrary number such that \( 0 < \varepsilon < 1 \), then the probability that \( |x_i| < 1 - \varepsilon \) satisfies
\[ \mathcal{P}(|x_i| < 1 - \varepsilon) = \frac{V(d)(1-\varepsilon)^d}{V(d)} = (1-\varepsilon)^d < e^{-\varepsilon d} \]
Thus, taking \( \varepsilon = \frac{2 \ln n}{d} \) we obtain that for each \( i \)
\[ \mathcal{P} \left( |x_i| < 1 - \frac{2 \ln n}{d} \right) < e^{-\frac{2 \ln n}{d} d} = e^{-\ln n^2} = \frac{1}{n^2} \]
In our case \( d = n \); thus, the probability that \( |x_i| < 1 - \frac{2 \ln n}{n} \) for at least one \( i \) is less than \( n \cdot \frac{1}{n^2} = \frac{1}{n} \). Consequently, with probability of at least \( 1 - \frac{1}{n} \) all points
satisfy $|x_i| \geq 1 - \frac{2 \ln n}{n}$ and consequently

$$\sum_{k=1}^{n} |x_i| \geq n \left(1 - \frac{2 \ln n}{n}\right) = n - 2 \ln n$$

Since all the points are drawn from a unit ball, we have $|x_i| \leq 1$ and thus

$$n \geq \sum_{k=1}^{n} |x_i| \geq n - 2 \ln n$$

with probability of at least $1 - \frac{1}{n}$, as required.

4. You happen to be a very good student receiving only Distinctions and High Distinctions. To get a High Distinction you have to score between 85 and 100; to receive a Distinction you have to score between 75 and 84. You noticed that on average after each High Distinction in the next exam you received another High Distinction in 2 out of every 3 cases, and after each Distinction you are equally likely to get a High Distinction as you are likely to get a Distinction. You also notice that whenever you get a Distinction all scores between 75 and 84 are equally likely, and similarly whenever you get a High Distinction it is equally likely you got every score between 85 and 100. Estimate your average mark. (15 points)

Solution: This is a simple Markov Chain with states $D$ and $HD$. We are given that $\mathcal{P}(HD \rightarrow HD) = 2/3$ which implies that $\mathcal{P}(HD \rightarrow D) = 1/3$. We are also given that $\mathcal{P}(D \rightarrow D) = 1/2$ and $\mathcal{P}(D \rightarrow HD) = 1/2$. Thus the frequency $hd$ of your $HD$’s and frequency $d$ of your $D$ satisfy the stationary distribution property

$$(d, hd) = (d, hd) \begin{pmatrix} 1/2 & 1/2 \\ 1/3 & 2/3 \end{pmatrix}$$

From there we obtain the following equations. First $d$ and $hd$ must sum up to 1 being probabilities of the only two states, and also using 2 we obtain

$$d + hd = 1$$

$$\frac{1}{2}d + \frac{1}{3}hd = d$$

$$\frac{1}{2}d + \frac{2}{3}hd = hd$$
It is easy to see that the solutions to this equations are \( d = 2/5 \) and \( hd = 3/5 \).
If you got an \( hd \) you are equally likely to have gotten any score between 85 and a 100, your average score per each \( hd \) is \((100 + 85)/2 = 92.5\), similarly, since you are equally likely to have gotten any score between 75 and 84 for each \( d \), your average score for each \( d \) is \((75 + 84)/2 = 79.5\). Thus the best estimate of your WAM is given by \( 92.5 \times 3/5 + 79.5 \times 2/5 = 87.3 \).

5. Use the PageRank algorithm to rank Twitter users according to their popularity. If a person A follows person B, person B is important to person A proportionally to the number of times A retweets messages of person B plus the number of times person A liked messages of person B. (15 points)

**Solution:** This is a typical problem to which the PageRank applies. The vertices of the graph are Twitter users and there exists a directed edge from a user \( A \) to a user \( B \) just in case user \( A \) has retweeted messages of user \( B \) or/and person \( A \) liked messages of user \( B \). The weight \( w^*_{AB} \) of edge \( \vec{AB} \) is the ratio \( w^*_{AB} = (r(A,B) + \ell(A,B))/\sum_{u \in U}(r(A,u) + \ell(A,u)) \) where \( r(A,B) \) is the number of the of \( A \)'s retweets of \( B \)'s tweets and \( \ell(A,B) \) is the number of likes of \( A \) of \( B \)'s tweets. \( \sum_{u \in U}(r(A,u) + \ell(A,u)) \) is the sum of the total number of \( A \)'s retweets and the total number of likes of \( A \) of any user \( u \). We then resolve the problem of dangling nodes in the same way as in the original PageRank by giving every dangling node an outgoing edge to every other vertex of weight \( 1/|U| \) where \( |U| \) is the number of all users. We also introduce weak edges from every vertex to every other vertex not already connected of weight \( (1 - \alpha)/|U| \) and where the weights of all existing edges are adjusted accordingly in the same manner as in the original PageRank, namely \( w_{AB} = \alpha w^*_{AB} + (1 - \alpha)/|U| \). Here \( \alpha \) is a parameter of value close to 1, usually \( \alpha = 0.85 \). Now the standard PageRank algorithm applies to obtain the popularity rank of a user.

6. Explain in your own words why Spectral Clustering algorithm works for clusters which are not center based. (20 points)

**Solution:** Spectral clustering operates on the graph whose vertices correspond to the points (usually vectors \( \tilde{a}_i \)) to be clustered, with edges between pairs of points having a weight proportional to the proximity (or more generally, similarity) of the two vertices. The similarity measure depends inversely to the distance between points; such similarity weight is usually given by a Gaussian of the form

\[
 w_{ij} = e^{-\frac{||\tilde{a}_i - \tilde{a}_j||^2}{2\sigma^2}}
\]
Here $\sigma$ is a design parameter which controls how fast the similarity of vertices decreases as their distance $\|\vec{a}_i - \vec{a}_j\|$ increases. Informally speaking, Spectral Clustering attempts to break up the points into clusters such that points which are close together (sufficiently similar) stay in the same cluster. The clusters are “approximate connected components” in the sense that between any two points in the same approximate component (cluster) there is a path between them where all the edges have high weights; on the other hand, points from different connected components can be connected only with paths where at least one edge has a low weight. Thus, the clusters, being such “approximate connected components”, do not necessarily have centers.