Akshaj Kammari
ak1990

Problem 1: To Absent Friends

1. B(1) = 1 because there is only one bagel, and it will be left for the person who ordered it. B(2) = 1/2 because there are two possible outcomes - either both people take their own bagels, or they swap, person 1 taking person 2's bagel, person 2 getting person 1's bagel.

2. Suppose person 1 takes person 2's bagel. Then there are two possibilities: either person 2 takes person 1's bagel, leaving the remaining (N-2) bagels to be distributed among the remaining (N-2) people, or person 2 takes someone else's bagel, say person 3's, leaving the remaining (N-2) bagels to be distributed among the remaining (N-2) people, with the knowledge that person 3 will take person 2's bagel. Therefore, we have:
   B(N) = (1/N) [B(1) + B(2) + ... + B(N-1)] where the factor of 1/N represents the probability that person 1 takes any particular bagel.

3. Using the result from part 2, we can write: B(N) = (1/N) [1 + B(1) + (1/2)B(2) + ... + (1/(N-1))B(N-1)]. Now, we can use mathematical induction to prove that: B(N) = 1/N for all N >= 1.
   Base case: B(1) = 1/1 = 1/N is true.
   Inductive step: Assume B(k) = 1/k for all k < N. Then: B(N) = (1/N) [1 + 1/1 + (1/2)(1/2) + ... + (1/(N-1))(1/(N-1))] = (1/N) [1 + 1/1 + (1/2)(1/2) + ... + (1/(N-1))(1/(N-1))] (by the inductive hypothesis) = 1/N for all N >= 1.

4. Bmax(1) = 1 because there is only one bagel, and it will be left for the person who ordered it. Bmax(2) = 1/2 + 1/2 * 1/2 = 3/4, where the first term represents the probability that person 1 takes their own bagel or Max's bagel, and the second term represents the probability that person 1 and person 2 swap bagels, leaving Max's bagel and person 2's original bagel for the last person.

5. We can again use mathematical induction to prove that: Bmax(N) = (1/N) [1 + Bmax(1) + (1/2)Bmax(2) + ... + (1/(N-1))Bmax(N-1)]= (1/N) [1 + (1-1/N)B(N-1)].

Bonus: In this scenario, the first person to arrive will always get their ordered bagel. For the second person, there are two possibilities: either the first person took their bagel, or they took the other one. The probability that the first person took their bagel is 1/N, since there are N bagels and the first person chooses one at random. If the first person took their bagel, then the second person gets their bagel with probability 1/(N-1), since there are N-1 bagels left and they choose one at random. If the first person took the other bagel, then the second person gets their bagel with probability 1/N, since there are still N bagels and they choose one at random. Therefore, the

probability that the second person gets their bagel is: P(second person gets bagel) = 1/N * 1/(N-1) + (1 - 1/N) * 1/N = 2/N(N-1). We can show that the probability that the nth person gets their bagel is: P(nth person gets bagel) = (n-1)! / N(N-1)...(N-n+1). Therefore, the probability that the last person gets their bagel (i.e., B(N)) is: B(N) = (N-1)! / N! = 1/N. So in this scenario, every person has the same probability of getting their bagel.

Problem 2: Security Concerns

1. Suppose that BagelBot positions MonitorBots on the vertices of a maximal independent set I. Assume that there is an unmonitored vertex v. Since I is a maximal independent set, v must be connected to at least one vertex in I, say u. However, this means that the bot stationed at u would also be monitoring v, which contradicts the assumption that v is unmonitored. Therefore, every vertex is monitored by at least one MonitorBot. Moreover, since I is an independent set, no bot will be monitoring a location where another bot is stationed, leaving no overlap between MonitorBots watching.

2. An example of a graph with 2 independent sets can be shown through {A,D} and {B,C}. Both of these are maximal and independent of one another, and also includes {A, B, C}, an independent set that is larger than the previous sets.

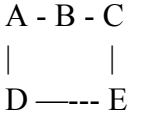
   A - B A
   | | / \
   C - D B C

3. For a complete graph on n vertices, every vertex is connected to all other vertices, so no independent set can contain more than one vertex. Therefore, $\alpha(G) = 1$. If n is odd, then $\alpha(G) = (n+1)/2$, as with n vertices, we select every other vertex on the cycle, which would leave us with an independent and maximal set. If n is even, $\alpha(G) = n/2$, as we would select every other vertex on the cycle from the beginning, leaving us with another independent and maximal set.

4. Suppose that the output I of the greedy algorithm is not a maximal independent set. Then there exists a vertex v that is not in I, but is adjacent to every vertex in I. Since v is adjacent to every vertex in I, it must have been processed before every vertex in I. Therefore, v was not added to I when its turn came up, which means that v is adjacent to some vertex in I that was already processed, contradicting the assumption that the algorithm adds v to I if it is not connected to any vertex already in I. Hence, I must be a maximal independent set.

5. Running this graph through A B C D E, we get an independent set {A}. A is not adjacent to any other vertex in the graph. Running this graph through E D C B A, then we get

maximal and independent set of {E, D, C, B}, as we add E to the set, then D, C, and E, as they are not adjacent to each vertex before them, making the set itself maximal independent.



4. The output of this algorithm is an independent set because in Step 2, if there are any edges with both vertices in I, one of the two vertices is deleted at random. This ensures that no two vertices in I are connected by an edge, and thus I is an independent set. However, it is not necessarily a maximal independent set because some vertices may not have been added to I in Step 1 due to the probability p. Therefore, there may be other vertices that could have been added to I to form a larger independent set.

5. Let X be the random variable that counts the number of vertices in I after Step 1. Each vertex is added to I independently with probability p, so the expected value of X is given by $E[X] = \Sigma x$ $P(X = x) * x$ where $P(X = x)$ is the probability that x vertices are added to I in Step 1. The probability that a single vertex is added to I is p, and there are $|V|$ vertices in the graph, so the expected number of vertices added to I in Step 1 is $p|V|$. Therefore, the expected value of X is $E[X] = p|V|$.

6. Let Y be the random variable that counts the number of edges in I after Step 1. Each edge is included in I independently with probability p^2, so the expected value of Y is given by $E[Y] = \Sigma y$ $P(Y = y) * y$, where $P(Y = y)$ is the probability that y edges are included in I in Step 1. There are $|E|$ edges in the graph, and the probability that a single edge is included in I is p^2, so the expected number of edges included in I in Step 1 is $p^2|E|$. Therefore, the expected value of Y is: $E[Y] = p^2|E|$.

7. Let Z be the random variable that counts the number of vertices in I after Step 2. The expected value of Z is E[Z] = E[X] - E[# vertices removed from I]. The expected number of vertices removed from I is the sum of the probabilities that each vertex is removed, which is equal to the sum over all vertices v in I of the probability that v is removed, denoted by P(v is removed). For a given vertex v, the probability that it is removed is equal to the probability that at least one of its neighbors is also in I and is removed. Since each vertex is added to I independently with probability p, the probability that a neighbor of v is also in I is p. Therefore, the probability that v is removed is at most p^2, since it requires both v and at least one of its neighbors to be in I. Thus, the expected number of vertices removed from I is at most p^2 times the number of vertices in I that have at least one neighbor in I, which is at most $p^2|E|$. Therefore, we have $E[Z] >= p|V| - p^2|E|$.

8. For-expected value, we can again use X as our random variables times the probability p, however we must account for an edge to be in set I, we need both vertices that make that edge. We can use O for X which means an edge is not in the set due to one or both vertices not being in that set, and 1 for X which means both vertices are present in the set to make that edge. Now we can use the same summation formula, where we are now iterating through all edges in the total number of edges in the graph. Probability p is being squared to account for both vertices that make the edge, and we are using (e) to show whether the edge is in set | or not, meaning both vertices are present in that set or not with O or 1. This expression can be simplified to p^2 * |E|, since we know the number of edges in set I and p^2 can be factored out since it is constant for each term in the linearity of expectation formula.

Bonus: For the 2D square grid, we can cover every vertex by selecting every other vertex in each row and column. This will cover half of the vertices in the grid, so the smallest fraction of vertices needed to cover is 1/2. For the regular triangular grid, we can cover every vertex by selecting every third vertex in each row. This will cover one-third of the vertices in the grid, so the smallest fraction of vertices needed to cover is 1/3.

Problem 3: If I Could Turn Back Time

1. Let X be the amount of time it takes BagelBot to catch up to the present. We can model X as the sum of two random variables: Y and Z, where Y is the time BagelBot spends looping back in time and Z is the time it takes to get from the last time eddy to the present. Y is a geometric distribution with parameter p, since each loop back in time has probability p of occurring. Therefore, E(Y) = 1/p. Z is a uniform distribution on the interval [0, T], since the time between time eddies is uniformly random between 1 and T. Therefore, E(Z) = (T + 0)/2 = T/2. So, by the linearity of expectation, we have E(X) = E(Y) + E(Z) = 1/p + T/2. To determine if BagelBot makes it back to the present, we need to calculate the probability that BagelBot catches up to the present in finite time. This is equivalent to the probability that BagelBot never loops back to a time before the last time eddy. The probability of this happening is (1-p)^n, where n is the number of time eddies that occur before BagelBot catches up to the present. So, BagelBot will make it back to the present if (1-p)^n > 0, which is true as long as p is not equal to 1.

2. Let X be the amount of time it takes BagelBot to catch up to the present, as before. Now, Y is a geometric distribution with parameter q, since each time jump is geometrically distributed with parameter q. Therefore, E(Y) = 1/q. Z is still a uniform distribution on [0, T]. Therefore, E(Z) = T/2, as before. To determine if BagelBot makes it back to the present, we need to calculate the probability that BagelBot catches up to the present in finite time. This is equivalent to the probability that BagelBot never jumps back in time

by an amount greater than the time since the last jump. We can model this as a Markov chain with states 0, 1, 2, ..., T, where state i represents being i minutes since the last jump. The transition probabilities are: P(i, j) = q, if j = i+1, P(i, 0) = 1 - q, if i < T, P(T, T) = 1. The probability of being in state i after k jumps is given by the kth row of the transition matrix raised to the ith power. To calculate the probability that BagelBot catches up to the present in finite time, we need to find the smallest k such that the probability of being in state T after k jumps is greater than 0. This can be done using matrix exponentiation. Once we have k, we can calculate the probability that BagelBot makes it back to the present as 1 - (1-q)^k.

3. Let X be the amount of time it takes BagelBot to catch up to the present. We can model X as the sum of two random variables: Y and Z, where Y is the time BagelBot spends looping back in time and Z is the time it takes to get from the last time eddy to the present. Y is a geometric distribution with parameter p, since each loop back in time has probability p of occurring. Therefore, E(Y) = 1/p. Z is a uniform distribution on the interval [0, T], since the time between time eddies is uniformly random between 1 and T. Therefore, E(Z) = (T + 0)/2 = T/2. So, by the linearity of expectation, we have E(X) = E(Y) + E(Z) = 1/p + T/2. To determine if BagelBot makes it back to the present, we need to calculate the probability that BagelBot catches up to the present in finite time. This is equivalent to the probability that BagelBot never loops back to a time before the last time eddy. The probability of this happening is (1-p)^n, where n is the number of time eddies that occur before BagelBot catches up to the present. So, BagelBot will make it back to the present if (1-p)^n > 0, which is true as long as p is not equal to 1.