Homework 1

Matini Gabriele 1934803

1 Problem 1

1.1

Let's analyze the three elements of the probability space, which are the event space $\mathcal{E}$, the sample space Ω and the probability function $\mathcal{F}$. Since we have 52! possible permutations, the sample space Ω will be the sample space containing all the 52! permutations, so a finite sample space like this:

$$\Omega = \{P_1, P_2, ..., P_{52!}\} \tag{1}$$

Where each P_i is a single possible permutation of the shuffled deck. It's also clear that $|\Omega| = 52!$. The probability of each element P_i is:

$$Pr(P_i) = \frac{1}{52!} \tag{2}$$

for each P_i. The event space $\mathcal{E}$ will be all the possible subsets of permutations, and the probability function $\mathcal{F}$ is the function assigning to each element of $\mathcal{E}$ a probability (number between 0 and 1). The probability space is the triple $(\Omega, \mathcal{E}, \mathcal{F})$

1.2

(a) $E =$ "First four cards include at least one club"
Let's consider our deck to have in total 13 clubs. When we pick our first four cards, event E doesn't happen if and only if none of them is a club. Therefore, we can reduce this to the problem of calculating when the event doesn't happen. Ruling out 13 cards out of 52, we get:

$$Pr(E) = 1 - Pr(\text{"}None\ of\ the\ first\ four\ cards\ include\ one\ club\text{"}) = 1 - \frac{39}{52}*\frac{38}{51}*\frac{37}{50}*\frac{36}{49} \approx 1 - 0.304 = 0.696 \tag{3}$$

(b) $E =$ "First seven cards include exactly one club"
For this event, we need the probability of six cards to of not being a club (given extractions without replacement), and the other card needs to be extracted out of the 13 clubs. This can happen in 7 ways (the club can be extracted in the first or the second extraction, and so on...):

$$Pr(E) = \frac{39}{52} * \frac{38}{51} * \frac{37}{50} * \frac{36}{49} * \frac{35}{48} * \frac{34}{47} * \frac{13}{46} * 7 \approx 0.317 \tag{4}$$

(c) $E =$ "First three cards all of the same suit"
We are looking for an event where all cards are extracted from the same 13 ones. This can happen in 4 different ways, since they can be clubs, diamonds, spades or hearts:

$$Pr(E) = \frac{13}{52} * \frac{12}{51} * \frac{11}{50} * 4 \approx 0.052 \tag{5}$$

(d) $E =$ "First three cards are all sevens"
We have in total four sevens to choose from:

$$Pr(E) = \frac{4}{52} * \frac{3}{51} * \frac{2}{50} \approx 0.00018 \tag{6}$$

(e) $E =$ "First five cards form a straight"
A straight can have as highest value 5, 6, 7, 8, 9, 10, Ace, King, Jack or Queen. So given any combination of suites there are ten ways to get a straight. There are also $5! = 120$ possible orders of the 5 cards' permutations. We first get the probability that they form either a straight or a straight flush (call this event E_a):

$$Pr(E_a) = \frac{4}{52} * \frac{4}{51} * \frac{4}{50} * \frac{4}{49} * \frac{4}{48} * 10 * 120 \approx 0.00394 \tag{7}$$

Then, we calculate the probability to get a straight flush (let's call the event E_b) and subtract it from $Pr(E_a)$:

$$Pr(E_b) = \frac{1}{52} * \frac{1}{51} * \frac{1}{50} * \frac{1}{49} * \frac{1}{48} * 4 * 120 * 10 \approx 0.000015 \tag{8}$$

$$Pr(E) = Pr(E_a) - Pr(E_b) \approx 0.003925 \tag{9}$$

1.3

The program was developed in Problem1.py

2 Problem 2

2.1

The sample space of question 3 has as events the tuples $< gender - day,\ gender - day >$, with all the possible combinations of gender and day. Thus, we have in total $2 * 2 * 7^2 = 196$ possible combinations, since we have 2 genders, 2 kids, 7 days that can change combination for the 2 kids. For each point we have 1/196 probability of the event happening.

2.2

E = "The other kid is a girl" G = "The first kid is a girl" The sample space Ω is
$\Omega = \{Boy - Boy, Boy - Girl, Girl - Boy, Girl - Girl\}$
Since the sample space reduces to 3 choices (we rule out Boy-Boy) we end up with: $Pr(E|G) = 1/3$

2.3

E = "The other kid is a girl too"
G = "One kid is a girl born on Sunday"
First we need to rule out 1/4 of all outcomes (49 outcomes) where they're all boys. Secondly we rule out the all outcomes where one girls isn't born on Sunday. Such outcomes are 14, all the $< Girl - Sunday,\ gender - day >$ combinations, and 13 for $< gender - day,\ Girl - Sunday >$ (not counting $< Girl - Sunday,\ Girl - Sunday >$ twice). We have thus restricted the sample space to 27 elements. We count 7 outcomes for when they are both girls and one is born on Sunday. We also count another 6 outcomes for the reverse order of the tuple (when the girl-sunday is the second element of the tuple), thus getting 13 favourable outcomes to event E. So we have:
$Pr(E|G) = \frac{13}{27}$

3 Problem 3

E = "Aris has Data-Miningitis"
G = "The test is positive"
$Pr(E) = \frac{1}{100000}$
The sample space Ω is divided between E and E^c, and between G and G^c with the events G and E overlapping in every combination ($E\ and\ G,\ E^c\ and\ G\ and\ so\ on...$). We are interested in the overlap between E and G, since we want first to restrict the sample space to the slice where G has happened. To calculate $Pr(E|G)$ we need to account for two macro events' probabilities: first, the probability that Aris has Data-Miningitis and that at the same time the test is positive given that Aris has the

illness, and the probability that Aris does not have Data-Miningitis and the test resulted positive given that he doesn't have the illness. From the test accurateness, we know that $Pr(G|E) = 0.999$ and $Pr(E) = \frac{1}{100000}$. We can use Bayes' theorem to get our result, partitioning Ω and considering only the sample space where G happens and calculating the probability (over the sample space slice of G) of E happening:
$Pr(D) = Pr(E|G) = \frac{Pr(G|E)Pr(E)}{Pr(G|E)Pr(E)+Pr(G|E^c)Pr(E^c)} = \frac{0.999*\frac{1}{100000}}{0.999*\frac{1}{100000}+0.001*\frac{99999}{100000}} \approx 0.0989$

4 Problem 4

Aris and Gianluca cannot choose the same number and we need to guess if Aris or Gianluca have picked the smallest number. Given a certain number x, we can define the numbers chosen by Aris and Gianluca with respect to x as follows:
$\Omega = \{G < x < A,\ A < x < G,\ G < A < x,\ A < G < x,\ x < A < G,\ x < G < A\}$
Where G represents the number chosen by Gianluca and A the number chosen by Aris. We have 6 possible positions of the three numbers with respect to eachother. Assume we ask Aris, the same reasoning can be applied to Gianluca: if Aris' number is bigger than x, then we rule out 3 out of these 6 possibilities, restraining the sample space to three elements. Choosing Gianluca, we have thus 2 out of 3 favourable outcomes. *Restricted* $\Omega = \{G < x < A,\ x < A < G,\ x < G < A\}$
Of these, if we choose G as lowest number we have $G < x < A$ and $x < G < A$ as favourable outcomes so $\frac{2}{3}$ of the total outcomes. So if the person we asked has picked a number $y > x$, we can always choose the other person to get a better probability of guessing the smallest number between A and G.
Knowing a-priori that x is between the two numbers, winning becomes even easier: the sample space Ω is: $\Omega = \{A < x < G,\ G < x < A\}$
If $A > x$, the sample space will be restricted to $\{G < x < A\}$, so if we choose Gianluca in this case we win with certain probability (1).
Now, in reality we cannot know such an x a-priori, and also we need to account for the fact that x might be equal to A or G.
The latter fact is not concerning, since, as the number N of pages increases, the probability of x being equal to either A or G diminishes. To prove this, it's sufficient to point out that, in our sample space, the probability of x being equal to A is $\frac{1}{N}$, and the same goes for G. Thus, in the context of having events such as $G < x < A$ to account for, events like $A = x < G$ ($x = A \wedge x < G$) will be very rare in comparison, for N that grows to infinite.
But not only that: as proven in the "Problem_4_experiment.py" file, when we take a smaller N, the probability of getting the answer right increases when accounting for equalities (for numpages=10 the difference is visible, for numpages=100 it's already negligible). Thus equalities would skew the probability in our favour. We can also prove this theoretically by making a simple consideration: if the number A Aris tells us is such that $x < A$, then the sample space is reduced to four possibilities:
$\{\ G < x < A,\ G = x < A,\ x < A < G,\ x < G < A\}$
The probabilities of these four events are not uniform because of the equality, but as we can see the equality case helps our strategy.
Even though such a sample space is more difficult to study (as we would have to account for cases like $A = x < G, G < A = x$ and so on, which do not have the same probability as cases like $A < x < G$), in the worst case we still get a probability of $\frac{2}{3}$ (just push N to the infinite to nullify equality effects and get back to the simpler case where x cannot be neither equal to A nor G).
So, we can devise the final strategy as follows: since we don't know x, let's choose it at random. Every time we can ask either Aris or Gianluca for their number(let's pick Aris, the reasoning is the same for Gianluca): if $A > x$, then choosing Gianluca will award us the same victory probability of $\frac{2}{3}$ because of the aforementioned reasoning. If $A < x$, then we know for the same reason that we need to choose Aris (Ω is again restricted).

5 Problem 5

5.1

Let's analyze the probability space $(\Omega, \mathcal{E}, \mathcal{F})$, where Ω is the sample space, $\mathcal{E}$ the event space and $\mathcal{F}$ the probability function. Given a graph of n nodes and being p the probability of each edge to be

initialized, we can describe the sample space as follows:
$\Omega = \{G1, G2, ...\}$ where each event is the initialization of a particular graph. The size of Ω is $|\Omega| = 2^{\binom{n}{2}}$, since we take into account that all the combinations of two vertices (edges) can be chosen in $\binom{n}{2}$ times. The event space $\mathcal{E}$ is thus defined as all possible subsets of graphs of n nodes, and the probability function $\mathcal{F}$ is the function associating to every event of $\mathcal{E}$ a probability.

5.2

Given n nodes and m edges, the probability of a particular graph Gi with these numbers of nodes and edges are:
$Pr(Gi) = p^m(1-p)^{\binom{n}{2}-m}$
where $\binom{n}{2}$ are all the ways to possibly pick edges combinations, and we subtract from $\binom{n}{2}$ the m edges that have actually been initialized.

5.3

Assuming n even, these types of graphs have exactly n edges in total, $\frac{n}{2}$ for every subgraph, and form two separated subgraphs representing the cycles. Notice how we cannot have these subgraphs be connected in any way. Let's call G, the event where we have exactly two disjoint cycles of length $\frac{n}{2}$ and no other edges. Thus, we need first to calculate all the possible combinations of such graphs.
Let's name the vertices $v_1, v_2, etc...$ First, we need to get all the combinations of the entire graph we can do: one of the subgraph might be composed of $v_1, .., v_{\frac{n}{2}}, v_1$ and the other of all the vertices from $v_{\frac{n}{2}}, ..., v_n$, but we can also have other combinations. These in total are $\binom{n}{\frac{n}{2}}$. Also, given a certain subset of $\frac{n}{2}$ nodes, there are $(\frac{n}{2}-1)!$ possible permutations. And given that we have 2 cycles to create, my initial guess was:
$Pr(G) = \binom{n}{\frac{n}{2}}((\frac{n}{2}-1)!)^2 p^n(1-p)^{\binom{n}{2}-n}$
The problem was that, computing by hand all possible graphs for n = 6, these graphs where 10, while my formula to compute all possible permutation computes way more than 10 graphs. It was clear also that the reason $\binom{n}{\frac{n}{2}}(\frac{n}{2}!)^2$ doesn't work is that there are more ways of computing the same graph: the two subsets of nodes could be switched, we could have $v_1, v_2, v_3, v_1 and v_4, v_5, v_6, v_4$ as initialization permutation of the cycles, or v_4, v_5, v_6, v_4 and v_1, v_2, v_3, v_1. Also, notice how the permutations like $v_1, v_2..., v_{\frac{n}{2}}, v_1$ and $v_{\frac{n}{2}}, v_{\frac{n}{2}-1}..., v_1, v_{\frac{n}{2}}$ form the same cycle, so we also need to account for that, dividing by 2 (these two sequences create the same edges). In conclusion, the final probability is:
$Pr(G) = \frac{\binom{n}{\frac{n}{2}}}{2}(\frac{(\frac{n}{2}-1)!}{2})^2 p^n(1-p)^{\binom{n}{2}-n}$

5.4

We analyze two cases: in the first case, we can have the 2 cycles and then some isolated nodes, in the second no leftover isolated nodes are allowed. For this we need to build upon the last exercise, accounting for all possible combinations of different lengths in the cycles. Let's call E the event where we have two disjoint cycles of any length and at the same time the remaining nodes have no edges. So let us take n nodes: we have a cycle of length i ($i \geq 3$), another cycle of length j ($j \geq 3$ too), and a final subset of $n-i-j$ nodes that do not form any cycle and have no edges. i and j are dependent from each other, since it must hold $i + j \leq n$.
The final probability is thus:
$Pr(E) = \sum_{i=3}^{n-3}\sum_{j=3}^{n-i}\frac{1}{2}\binom{n}{n-i-j, i, j}\frac{(i-1)!}{2}\frac{(j-1)!}{2}p^{i+j}(1-p)^{\binom{n}{2}-i-j}$
Where the sums account for all the possible values i and j can take. In the case where for E there can be no leftover nodes out of the two cycles, we instead have a simpler formula, accounting for only the cycles (it holds always $i + j = n$, so $j = n - i$):
$Pr(E) = \sum_{i=3}^{n-3}\frac{1}{2}\binom{n}{i, n-i}\frac{(i-1)!}{2}\frac{(n-i-1)!}{2}p^n(1-p)^{\binom{n}{2}-n}$

5.5

For a given node v, the initialization of any edge involving v happens with probability p and doesn't happen with probability $1-p$. Thus, the degree of a node is the number of such successful initializations

involving said node. The maximum number of such initialization is $n-1$ (we don't count a self edge). Let's call X the number of edges connected to v. We have a binomial distribution for the degree of each vertex: $Pr(X=i)=\binom{n-1}{i}p^i(1-p)^{n-1-i}$
By definition of the expected value we have:
$E[X]=\sum_{i=1}^{n-1} Pr(X=i)\ i=\sum_{i=1}^{n-1}\binom{n-1}{i}p^i(1-p)^{n-1-i}\ i=\sum_{i=1}^{n-1}(n-1)p\binom{n-2}{i-1}p^{i-1}(1-p)^{n-1-i}=(n-1)p\sum_{j=0}^{n-2}\binom{n-2}{j}p^j(1-p)^{n-2-j}=(n-1)p(p+1-p)^{n-2}=(n-1)p$

5.6

Again, an edge is present with probability p and not present with probability $1-p$. Let's call X the number of edges.
$X=X_1+X_2+...+X_{\binom{n}{2}}$
where each X_i is a random variable that takes the value 1 if the edge e_i was instantiated, 0 otherwise. We have thus:
$E[X_i]=p+0(1-p)=p\ for\ i=1,...,\binom{n}{2}$
$E[X]=\sum_{i=1}^{\binom{n}{2}}E[X_i]=\binom{n}{2}p$

5.7

A papillon is a subgraph of 5 nodes and 6 edges arranged in that particular form. It's clear then that the probabilty of a certain papillon P_i to be initialized are:
$Pr(P_i)=p^6(1-p)^4$
Where 6 are the edges we need for the papillon, and at the same time we need 4 edges not to be initialized.
Let us now call X_i the indicator variable that takes the value of 1 if papillon P_i was initialized, otherwise it takes the value of 0.
For n nodes, the maximum number of subsets of 5 nodes is $\binom{n}{5}$. Notice also that, for each fixed subset of 5 nodes, we can form up to 15 different papillons. To prove this, let's call "center" of the papillon the central node connected to the other 4 nodes. It's clear that in a fixed subset of 5 nodes there are 5 different possible centers, each of which defines the edges needed to connect the center. By hand, one can also verify easily that, fixing the center, there are at most 3 different papillons that can be obtained by switching the non-center nodes among each other. Thus, we have $3*5=15$ different papillons given any specific subset of 5 nodes. In conclusion, the number of all possible papillons is $\binom{n}{5}15$.
So we can define the number X of papillon subgraphs as $X=X_1+X_2+...+X_{\binom{n}{5}15}=\sum_{i=1}^{\binom{n}{5}15}X_i$.
This means that, by linearity of the expected value, it holds:
$E[X]=\sum_{i=1}^{\binom{n}{5}15}E[X_i]=\sum_{i=1}^{\binom{n}{5}15}p^6(1-p)^4=\binom{n}{5}15p^6(1-p)^4$

This result was tested by the script "Problem_5.7_papillon_experiment.py", where the formula was tested for $n=6$ and $p=1/2$: the above formula gives an expected value of ≈ 0.087, which is confirmed by the estimation done by the script, which generates one million graphs and calculates the total number of papillons found, to then estimate the expected value by doing $\frac{\#papillons\_found}{\#graphs}$ $(\#graphs=1000000)$.

6 Problem 6

First, we need to download the file and wget helps us:
wget –user=USER –password=REDACTED http://aris.me/contents/teaching/data-mining-2024/protected/beers.txt

Where we specify user and password. This downloaded the file directly into the folder. The file presents itself as a file with beer names and review scores that go from 1 to 20. Next, we start to see what commands like awk, sort and uniq do. For example, awk is good to isolate a field of the file with the {print $n} option to get the nth column. uniq eliminates the duplicate lines of a file, and can be used together with -c specification to count the eliminated duplicates. sort sorts the file on a

specific recognizable field, which can be specified with the -km,mnr where m is the m-th field, and 'r' is present if we want to reverse the sorting order. A thing I discovered after a bit of testing, is that uniq eliminates only adjacent duplicates, so we need to sort the file with the beer names first before counting. Thus, the final solution is concatenating awk with sort, with the count operation, with sort again to present the results in the right order. Finally, we can use the head command to get the first 10 elements in descending order of number of reviews:
$awk\ -F'\backslash t'\ '\{print\ \$1\}'\ './beers.txt'\ |\ uniq\ -c\ |\ sort\ -k1,1nr\ |\ head\ -n\ 10$
However, this command still didn't work properly, showing this result:

```
3 Newcastle Brown Ale
3 Rogue Imperial India Pale Ale &#40;IPA&#41;
2 1809 Berliner Style Weisse
2 Abbaye dAulne Val de Sambre 6 &#40;Ambre&#41;
2 Abita Fleur-de-Lis Restoration Ale
2 Abita SOS &#40;Save Our Shore&#41;
2 Aecht Schlenkerla Rauchbier Urbock
2 Aecht Schlenkerla Rauchbier Weizen
2 Alaskan Smoked Porter
2 AleSmith Anvil ESB
```

The result is clearly wrong since a simple check on the file with sort reveals that there are more than 3 reviews per beer. This is because we need to sort the output of awk before counting because of the aforementioned functioning of the uniq instruction. We also need to specify that '
tab' is the real separator in awk. Thus the final command becomes:
$awk\ -F'\backslash t'\ '\{print\$1\}'\ './beers.txt'\ |\ sort\ |\ uniq\ -c\ |\ sort\ -k1,1nr\ |\ head\ -n\ 10$
where with -F we specify the separator.
The final result is:

```
3696 Guinness Draught
3662 Pabst Blue Ribbon
3230 Dogfish Head 90 Minute Imperial IPA
3126 Budweiser
3119 Sierra Nevada Pale Ale &#40;Bottle&#41;
3110 Samuel Adams Boston Lager
3056 Chimay Bleue &#40;Blue&#41; / Grande Rserve
2904 North Coast Old Rasputin Russian Imperial Stout
2872 Stone Arrogant Bastard Ale
2813 Orval
```

7 Problem 7

Solution available on the python script, the final result is written on the file ”Problem_7_results.txt”. The script makes use of two dictionary structures to store review count and score sum, and, after removing all beers with less than 100 reviews, it then calculates the averages. Finally, we sort the dictionary by average score and write the result.