Assignment 2: Probability/Scheduling
Submitted by Abdul Derh

1. (a) First, let's analyze our probabilities of winning here:
P(selecting a car the first time) = $\frac{1}{4}$
P(selecting a goat the first time) = $\frac{3}{4}$

$\therefore$ We are more likely to select a goat the first time. These probabilities can be carried over to not switching. So P(winning a car by not switching) = $\frac{1}{4}$ and P(winning a goat by not switching) = $\frac{3}{4}$. Now let's analyze the possibility of winning by switching:

P(selecting a goat the first time) = $\frac{3}{4}$
P(selecting a car after a door has been identified) = $\frac{1}{2}$
This is because we have selected one door and another has been identified, so either of the two remaining doors could potentially have the car, assuming we first picked a goat and not a car.

P(winning a car by switching) = $\frac{3}{4} \cdot \frac{1}{2} = \frac{3}{8}$
This is because after you have selected 1 of 3 doors (containing a goat), you have a 50% chance of actually selecting the door with a car behind it, because there are two doors left and a $\frac{1}{2}$ chance of picking the car. This means, your final probability is reduced by half.

$\therefore$ since $\frac{3}{8} > \frac{1}{4}$ we have a better chance of winning if we switch doors.

(b) In this problem, the only variation is in the $\frac{1}{2}$ factor which is the probability of selecting a car after a door has been identified and two doors are remaining. If Monty was to open *two* doors instead of one (and assuming I have picked wrong the first time), then the P(selecting a car after two doors have been identified) = 1 or 100%. $\therefore$ if I select a goat the first time, I am certain to win by switching.

$\therefore$ P(win a car by switching) = $\frac{3}{4}$ vs P(win a car by not switching) = $\frac{1}{4}$. So switching is a much better choice,

(c) We know that there are n doors. In those n doors, $n-1$ will have a goat. $\therefore$ P(picking a goat first) is $\frac{n-1}{n}$, and P(picking a car after a goat is revealed) = $\frac{1}{n-2}$ because there are $n-2$ doors that are left and one will have the car in it (assuming you picked a goat first). Finally, the P(winning a car by switching) = $\frac{n-1}{n} \cdot \frac{1}{n-2}$. To answer the second part:

Looking for a value n such that:
P(winning a car by switching) = $\frac{n-1}{n} \cdot \frac{1}{n-2} < \frac{1}{n}$ = P(winning a car by not switching)

$$\frac{n-1}{n-2} < 1$$
$$n-1 < n-2$$

That is never true. A number minus one cannot less than the same number minus 2. So the answer is no, there is no threshold value n where it is better for you to not switch.

(d) Again, n doors and $n-1$ doors have a goat, but this time $n-2$ doors will be opened. So we select one door which may or may not have a car, and another door which may or may not have a car, but all other doors are shown to not have a car. If we choose wrong the first time, we are guaranteed to win because one of the two unknown doors is a goat. Thus, the other door, which is the one we switch to, *must* be a car ergo we are certain to win. In math:

P(win by not switching) = $\frac{1}{n}$
P(win by switching) = P(selecting a goat first) = $\frac{n-1}{n}$

Since $\frac{n-1}{n} > \frac{1}{n}\ \forall\ n > 2$, where n $\in \mathbb{Z}$ (n represents the number of doors), there is no value n such that $\frac{n-1}{n} <= \frac{1}{n}$. For that reason, switching is always the best decision, thus there is no number of doors (threshold of n) where it is better not to switch.

2. (a) Some Notation:

$$B_2 : |\text{2 people have the same birthday}|$$
$$B_3 : |\text{3 people have the same birthday}|$$
$$S = \text{sample space}$$
$$|S| = 365^3 \text{ (365 ways for one, 365 for the other and 365 for the last)}$$
$$P(E) = \frac{B_2 + B_3}{|S|}$$

To count the number of possibilities that two people have the same birthday and the third person does not, I first choose two numbers. Now, I have to arrange these numbers (with repeats) on 3 places where two numbers can repeat and one cannot: $P(3; 2, 1)$. I also notice that that for each number I select, there is a reverse case (1, 1, 2) and (2, 2, 1), so I multiply my arrangement by two.

$$B_2 = \frac{365}{363! \cdot 2!} \cdot \frac{3!}{2! \cdot 1!} \cdot 2$$

To count the number of possibilities that three people have the same birthday, I notice there are 365 ways for three people to all have the same birthday |(1,1,1), (2,2,2), (3,3,3) ... (n,n,n)| = same thing as counting from 1 to 365 = 365.

$$B_3 = 365$$

Therefore:

$$P(E) = \frac{398580 + 365}{365^3}$$
$$= \frac{398945}{48627125}$$

Now to check my answer using $P(E) = 1 - P(\overline{E})$. I know $P(\overline{E})$ is simply the event of all birthdays being different. That means you have 365 permute 3 different ways to make an ordered 3-tuple, and divide that value by the total number of possibilities (which is 365 for one person $\times$ 365 for the other person $\times$ 365 for the last person $= 365^3$). That is equivalent to $P(\overline{E}) = \frac{P(365,3)}{365^3}$:

$$\begin{aligned}
P(E) &= 1 - \frac{P(365,3)}{365^3} \\
&= 1 - \frac{365 \cdot 364 \cdot 363}{365^3} \\
&= 1 - \frac{48228180}{48627125} \\
&= \frac{48627125 - 48228180}{48627125} \\
&= \frac{398945}{48627125}
\end{aligned}$$

Therefore my answer is correct.

(b)

$$\begin{aligned}
B_2 &: |\text{2 people have the same birthday}| \\
B_{2_2} &: |\text{2 sets of 2 people have the same birthday}| \\
B_3 &: |\text{3 people have the same birthday}| \\
B_4 &: |\text{4 people have the same birthday}| \\
S &= \text{sample space} \\
|S| &= 365^4 \\
P(E) &= \frac{B_2 + B_3 + B_4}{|S|}
\end{aligned}$$

Immediately I know $B_4 = 365$ (for $n = 4$) for the same reason $B_3 = 365$ when $n = 3$ above. For B_2, I must choose three numbers, one for the two repeats and two others for the two non-repeats, and I multiply it with the number of arrangements (with repetitions) for 4 numbers (3 distinct categories: 2 repeat, 1 does not, 1 does not). Finally, I multiply it by three because there are three cases: (1,1, 2, 3) or (2, 2, 1, 3) or (3, 3, 1, 2).

$$\begin{aligned}
B_4 &= 365 \\
B_2 &= \frac{365!}{363! \cdot 3!} \cdot \frac{4!}{2! \cdot 1! \cdot 1!} \cdot 3 \\
&= 289369080
\end{aligned}$$

B_{2_2} is the case where only two numbers are chosen and there are two repeats with it (e.g: (1,1, 2, 2)). Obviously, I choose 2 from 365, I arrange them (with repetition) on four spots (knowing that both have the potential to repeat) and I multiply by the reverse case (2 this time).

$$
\begin{aligned}
B_{2_2} &= \frac{365!}{363! \cdot 2!} \cdot \frac{4!}{2! \cdot 2!} * \cdot 2 \\
&= 797160
\end{aligned}
$$

For B_3, I choose 2 numbers - one that repeats and one that does not. Then, I arrange them on 4 positions (with repetition: 3 can repeat, one does not), and I know there are 2 cases for this as well ((1, 1, 1, 2) and (2, 2, 2, 1)).

$$
\begin{aligned}
B_3 &= \frac{365!}{363! \cdot 2!} \cdot \frac{4!}{3! \cdot 1!} * \cdot 2 \\
&= 132860
\end{aligned}
$$

Finally,

$$
\begin{aligned}
P(E) &= \frac{289369080 + 797160 + 132860 + 365}{365^4} \\
&= \frac{290299465}{365^4}
\end{aligned}
$$

(Checking my Answer):

$$
P(E) = 1 - P(\overline{E}) = 1 - \frac{365 \cdot 364 \cdot 363 \cdot 362}{365^4} = \frac{290299465}{365^4} \quad \textit{Correct!}
$$

As n increases, you have to consider another term and you have to change the various combinations of how these people can be arranged (alongside accounting for the combinations of the new term, and the "in-between" cases like B_{2_2} that arise). This means complexity increases (by more than just one term because every term also has to have changes done to it) exponentially for greater n's, compared to the relatively easier method of finding $P(E)$ by $P(E) = 1 - P(\overline{E})$.

3. (a) Sample Space, S = {RGB, RGW, RGY, RBW, RBY, RWY, GBW, GBY, GWY, BWY}. The order does not matter. $|S| = 10$.

   (b) i. P(Y included): (RGY, RBY, RWY, GBY, GWY, BWY) $\frac{6}{10} = \frac{3}{5} = 60\%$

   ii. P(Y excluded): 1 - P(Y included) $= 1 - \frac{3}{5} = \frac{2}{5} = 40\%$

   iii. P(B and G included) (RGB, WBG, GBY): $\frac{3}{10} = 30\%$

   iv. P(B or G included) (RGB, RGW, RGY, RBW, WBG, RBY, YWB, YWG, WBY, WBG) $\frac{9}{10} = 90\%$

4. We are simply selecting three things from five (5 choose 3): C(5, 3) $= \frac{5!}{(5-3)! \cdot 3!} = 10$ total ways to select three balls from five. (This is confirming the sample space. I am in no way referring to the sample space, rather, I am checking to make sure the answers in question 3 are not wrong.).

(a) Let any selection have Y in it (order does not matter). $1 \cdot$ C(4, 2) $= 1 \cdot \frac{4!}{(4-2)! \cdot 2!} = 6$ ways to have Y in it. Therefore, there is a $\frac{6}{10} = \frac{3}{5}$ probability of having a Y included. This confirms my answer for 3b.

(b) Take the total number of selections subtracted by the value in part a. $10 - 6 = 4$ ways to exclude Y. $\therefore \frac{4}{10} = \frac{2}{5}$ is the probability of Y being excluded. This confirms my answer for 3b.

(c) Treat B and G as one unit, and look for only one from the remaining 3. C(3,1) $= \frac{3!}{2! \cdot 1!} = 3$. $\therefore \frac{3}{10}$ is the probability of B and G being included. This confirms my answer for 3b.

(d) The probability of B or G occurring is the same thing as finding out when B is included and when G is included. However, we we must subtract instances when both B and G are included (found that above) as they are included in the individual probabilities of B and G. P(B) and P(G) both have to be $= \frac{6}{10}$ because the process is exactly the same as part A — remove 1 from the total and choose one less.

$$\begin{aligned} P(B \text{ } or \text{ } G) &= P(B) + P(G) - P(B \text{ } and \text{ } G) \\ &= \frac{6}{10} + \frac{6}{10} - \frac{3}{10} \\ &= \frac{9}{10} \end{aligned}$$

$\therefore$ the probability for a B or a G to appear is $\frac{9}{10}$. This confirms my answer for 3b.

5. (a) c^1, c^2, c^3: 100000 + 100000·1.1^1 + 100000·1.3^2 = \$282048293.10

c^1, c^3, c^2: 100000 + 100000·1.3^1 + 100000·1.1^2 = \$1352018.454

c^2, c^1, c^3: 100000 + 100000·1.2^1 + 100000·1.3^2 = \$282058293.10

c^2, c^3, c^1: 100000 + 100000·1.3^1 + 100000·1.2^2 = \$16078931.92

c^3, c^1, c^2: 100000 + 100000·1.2^1 + 100000·1.1^2 = \$1342018.454

c^3, c^2, c^1: 100000 + 100000·1.1^1 + 100000·1.2^2 = \$1605893192.00

Immediately, one can see a difference in cost due differences in scheduling.

(b) Initial Notation:

C: the set of properties to renovate
C_i: the ith property in C
c: the cost appreciation factor of the corresponding of a property in C
t: the month in which property in C is renovated. It is also the index of that property in it's containing subset.

$f_c(t)$: a function of cost in terms of months, t, representing the total cost of renovation.
$f_c(t) = 100000 * (c_i)^t$ for some cost appreciations rate c_i.
$S = (S_1, S_2, ..., S_n)$: the set of properties created by the following algorithm.
n: represents the total number of properties in set C.

Let O represent the set of all optimal solutions.
Let $B = \{B_1, B_2, ..., B_n\}$ represent a set of one of the optimal solutions in O, which is assumed to have a total cost less than S.

Algorithm:
Sort the jobs by decreasing cost appreciation factor, c_i, therefore a property C_i is before C_j in the set S if and only if $c_i > c_j$. In set S (which is created by this algorithm), the value at S_n is last and it corresponds to some property in the set C which has the lowest appreciation rate.

My algorithm should minimize the total cost because as time progresses, (month after month) the cost of renovating a property with a greater c_i is more than that of a property with a lower c_i. This can be seen above with the permutation c_3, c_1, c_2 (how this algorithm would sort) vs c_2, c_1, c_3 (the opposite of this algorithm). The former is significantly less than the latter, and all other permutations also show this relation.

(c) I must assume that set S is not an optimal solution but set B is.

For some values i and k, assume it is true that $B_i = S_i \ \forall \ i < k$. This means underlying properties (C_i and it's respective c_i) are the same for both sets B and S in this range. This also means that the total cost of both sets are the same until $i = k$. This implies that underlying properties at k must be completely different in both sets, and "different" is defined as having a different cost factor.

B_k must have a lesser cost factor, c_b, $< c_s$ of S_k — by definition of my algorithm, the only way S_k and B_k don't represent the same property is if c_b, $< c_s$.

This means the total cost of $B_k = f_b(k) < f_s(k) =$ the total cost of S_k (for $t = k$). But this does not prove B is an optimal set because this is only looking at index k and not considering the entire set B.

The following must be true about set B and the underlying property represented by S_k:

$$B = \{B_1, B_2, ..., B_k, ..., B_s = S_k, ..., B_n\}$$

This means that the property represented by S_k can be found in the set B, call it B_s, and it can be found after B_k because everything before k in set B is the same as set S. I have to prove that if I were to swap the values B_s and B_k in the set B, the total cost of set B would not change because B is optimal. If I am unable to prove this mathematically, then B is not the most optimal set (rather set S would be optimal).

The mathematical attempt of a proof of this change is below:

More Notation

Let T_B = the total cost of swapping properties in set B.
Let s = the original index/month of B_s (which is the same underlying property as that of S_k) in the set B.
Let k = the original index/month of B_k in the set B.
Note: s is greater than k because B_s occurs after B_k in set B as explained above. Here we are scrutinizing the supposed optimal set B.

$$f_b(t) = 100000 * (c_b)^t \tag{1}$$
$$f_s(t) = 100000 * (c_s)^t \tag{2}$$
$$\Delta f_b = f_b(new) - f_b(initial) = f_b(t_s) - f_b(t_k) \tag{3}$$
$$\Delta f_s = f_s(new) - f_s(initial) = f_s(t_k) - f_s(t_s) \tag{4}$$
$$\Delta T_B = \Delta f_b + \Delta f_s \tag{5}$$
$$\Delta T_B >= 0 \tag{6}$$

Equation 5 is correct because the change in the total, T_B, is the sum of the change in cost of moving both elements in set B. $\Delta T_B >= 0$ must be true if and only if B is an optimal set. Changing anything should either increase T_B or leave it the same, but the total cost computed by set B should never decrease (because B is assumed optimal). This is contradicted, substituting (3) and (4) into (5) yields:

$$f_b(t_s) - f_b(t_k) + f_s(t_k) - f_s(t_s) > 0$$
$$100000 \cdot c_b^s - 100000 \cdot c_b^k + 100000 \cdot c_s^k - 100000 \cdot c_b^s > 0$$
$$c_b^s - c_b^k + c_s^k - c_b^s > 0$$
$$c_b^s - c_s^s + c_s^k - c_b^k > 0 \tag{7}$$
$$c_b^s - c_s^s = (c_b - c_s)(c_b^{s-1} + c_b^{s-2} \cdot c_s + ... + c_b \cdot c_s^{s-2} + c_s s - 1) \tag{8}$$
$$c_s^k - c_b^k = (c_s - c_b)(c_s^{k-1} + c_s^{k-2} \cdot c_b + ... + c_s \cdot c_b^{k-2} + c_b k - 1)$$
$$= -(c_b - c_s)(c_s^{k-1} + c_s^{k-2} \cdot c_b + ... + c_s \cdot c_b k - 2 + c_b^{k-1}) \tag{9}$$

Equations 8 and 9 are simply the factor of a difference of $a^n - b^n$ which is $(a-b)(a^{n-1} + a^{n-2} \cdot b + a^{n-3} \cdot b^2 + ... + a^2 \cdot b^{n-3} + a \cdot b^{n-2} + b^{n-1})$. There are always n terms in the second factor (the long one), and this is important: from (8) and (9), $(c_b - c_s)$ can be factored out and (7) can be written as

$$\begin{aligned} &(c_b - c_s)[c_b^{s-1} + ... + c_s^{s-1} + -(c_s^{k-1} + ... + c_b^{k-1})] \\ =&(c_b - c_s)[c_b^{s-1} + ... + c_s^{s-1} - c_s^{k-1} - ... - c_b^{k-1}] \\ =&(<0) \cdot (>0) \\ =&<0 \end{aligned} \tag{10}$$

The first factor (c_b - c_s) is less than 0 because $c_s > c_b$. The second factor is greater than zero, because there are more terms (s terms compared to k terms that are subtracting from s) that are positive. In addition, the first set of positive terms have a higher exponent, up to s (which is greater than k that the negative's side has). This means a positive value will result from evaluating all of the terms in the long factor. Finally, a positive value multiplied by a negative value is always negative, and this means that the change of cost is less than zero and total cost has decreased! Also, the set B is made more optimal by making it resemble set S. $\Rightarrow\Leftarrow$

So $\Delta T_B < 0$ (therefore T_B has decreased) when S_k in set B is swapped with B_k in set B, and this is a contradiction because B is supposed to be optimal and swapping any value in B should increase (or not change at all) the total cost, T_B.

In conclusion, B is not the optimal solution, and, by the proof above, the most optimal set is the set created by the algorithm in 5(b) — my algorithm.

o