Maximization on University of British Columbia's Undergraduate Tuition Revenue

MATH 340 Project

Ting Guo
Frank Han

Introduction

For every organization in society, the efficiency of operation is key to its development and improvement in its reputation in the industry. For a university like the University of British Columbia (UBC), although its revenue relies heavily on government grants and contracts and slightly on donations, the revenue it generates from student tuition is also essential to determine whether it can operate efficiently. Keeping all the expenses and other revenue constant, maximizing tuition revenue would lead to a higher profit. As a result, more money can be invested into the operation of UBC for the next year which improves the growth of the organization. Investments can be put into research in different areas as well as to improve campus facilities and infrastructures. With better facilities, researchers and students are more likely to conduct better research, which improves UBC's reputation, and in turn brings more talented researchers and students to the university, generating a beneficial cycle. With the strike of COVID-19, many students have changed their plan at UBC, which impacts UBC's tuition revenue. Maximization of tuition revenue is needed to help UBC remain profitable as well as operationally efficient.

Objective

To maximize the profit of an organization, one can either increase its revenue, or decrease its cost, or do both. For UBC Vancouver campus, tuition and student fees have made up about 30% of its revenue in recent years [1]; how it maximizes its tuition income will greatly affect its overall revenue, hence impacting the financial report for the university. This research aims to maximize the University of British Columbia (UBC) Vancouver campus's undergraduate student tuition revenue by investigating the factors and constraints that influence the tuition revenue using linear programming techniques.

Background

1. Domestic/International Student Ratio

   The UBC student population consists of both local and international students. The difference in tuition between domestic and international students is massive (almost 10 times).

Therefore, investigating the tuition revenue with respect to different types of students can be considered similar to a linear problem.

2. Faculty tuition fees should also be considered in the project. Many engineering courses tend to have higher cost-per-credit than arts courses. Therefore, focusing on specific faculty by calculating and comparing their annual tuition fee would also help us to see if there are rooms to improve/maximize its profit margin.

3. COVID-19

   As COVID-19 strikes the whole world, different countries are having different regulations and approaches to cope with the pandemic. Therefore, student's responses to travelling to UBC will differ compared to previous years [2]. For example, most lab-focused courses have been made online, which makes it less ideal for students to learn related concepts and lab techniques; hence, some students choose not to take these courses. Also, some prerequisite courses being delayed might also affect students' plans for their academic year. Similarly, many other aspects, such as a drop in the general interest in studying, may result in students taking less courses, hence impacting tuition revenue.

Problem Modelling

The problem can be expressed and solved as a linear problem with decision variables, objective function and constraints described as below.

Decision Variables

The decision variables of the linear program represent the undergraduate headcount ratio of each faculty with respect to the grand total undergraduate headcount of UBC Vancouver campus. Each faculty is further separated into domestic and international headcount ratios. For example, ARTS_D = domestic arts student headcount ratio, ARTS_I = international arts student headcount ratio; both ratios are with respect to the grand total headcount. According to the Fact Sheet, there are 13 faculties in undergraduate program, therefore a total of $13 * 2 = 26$ decision variables: $x_1, x_2, ..., x_{26}$ [5]. All of the decision variables should add up to 1 since the decision variables are partition of the undergraduate student headcount.

The decision variables are represented as ratios instead of headcounts because headcounts are uncertain values that vary from year to year, which would make it harder to manipulate through the linear program.

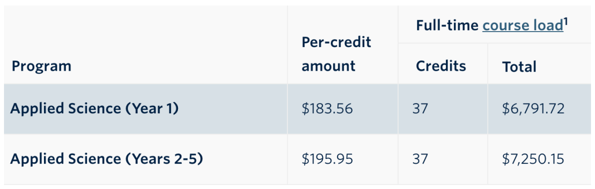
Objective Function

The linear program intends to maximize the objective function, where the objective function represents the total undergraduate tuition revenue of UBC Vancouver campus:

$$z = H \sum_{i=1}^{26} c_i x_i \tag{1}$$

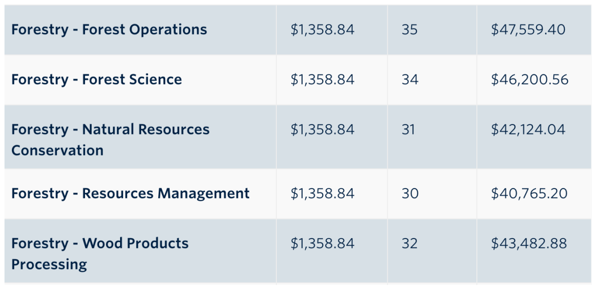
where c_i's represent the expected tuition of a domestic/international student in corresponding faculty, and H represents the grand total headcount of the undergraduate program.

The expected tuition, c_i's, are based on undergraduate tuition fees on the official UBC website [4]. For the faculties that provide different tuition based on students' year level (eg: domestic applied science and domestic commerce) or students' specialty in the faculty (eg: forestry, both domestic and international) (see Figure 1), uniform distribution of the student headcount on year level or on program specialization is assumed.



Program	Per-credit amount	Full-time course load[1]	
		Credits	Total
Applied Science (Year 1)	$183.56	37	$6,791.72
Applied Science (Years 2-5)	$195.95	37	$7,250.15

(a) Domestic Applied Science



Forestry - Forest Operations	$1,358.84	35	$47,559.40
Forestry - Forest Science	$1,358.84	34	$46,200.56
Forestry - Natural Resources Conservation	$1,358.84	31	$42,124.04
Forestry - Resources Management	$1,358.84	30	$40,765.20
Forestry - Wood Products Processing	$1,358.84	32	$43,482.88

(b) International Forestry

Figure 1: Undergraduate Tuition

For example, the expected tuition of a domestic applied science student is:

$$\frac{1}{5} \cdot 6791.21 + \frac{4}{5} \cdot 7250.15 = 7158.46$$

and the expected tuition of an international forestry student is:

$$\frac{1}{5}(47559.40 + 46200.56 + 42124.04 + 40765.20 + 43482.88) = 44026.416 \approx 44026.42$$

Since there is no tuition information on the faculty NONE (represents no faculties), its domestic (or international) expected tuition is estimated to be the mean of the domestic (or international) expected tuition of all other faculties.

The grand total undergraduate headcount, H, is estimated based on historical undergraduate headcount from year 2009 to 2020 [3]. For the estimation of H, see section PuLP Implementation.

Constraints

The constraints of the linear program composes two parts: the constraint on the sum of the decision variables, and the constraints (both upper and lower bounds) on each decision variable.

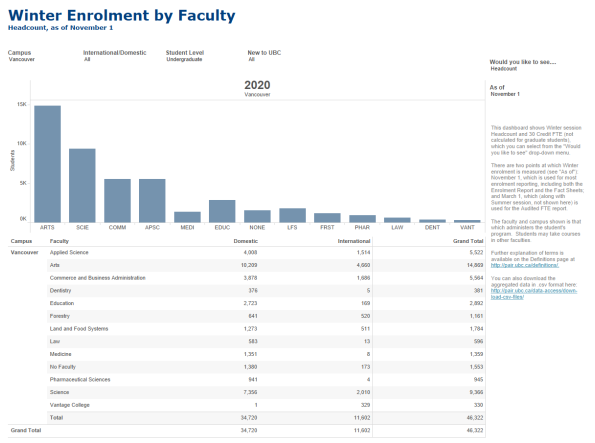
First, all of the decision variables should sum up to exactly 1, as explained in the subsection Decision Variables.

Next, each decision variable is set to have both an upper and a lower bound. These represent the upper and lower limits of how many domestic/international students a faculty can enroll with respect to the total undergraduate headcount. Without such constraints, a faculty may enroll as many international students and enroll very few domestic students, or the undergraduate program may enroll as many science students and enroll very few arts students, which makes the faculty student ratio or domestic/international student ratio lose balance.

To give the upper and lower bounds for each variable, the estimated mean for each variable is calculated first. Figure 2a gives the domestic/international headcount within each faculty. The domestic/international ratios within each faculty are calculated using Excel spreadsheet:

$$\text{Domestic}(\%) = \frac{\text{Domestic}}{\text{Grand Total}} \tag{2}$$

the calculation of International(%) is similar. The calculated results are presented in Figure 2b. Moreover, the ratios of each faculty with respect to the entire undergraduate program is presented in Figure 3.



Winter Enrolment by Faculty
Headcount, as of November 1

Campus: Vancouver
International/Domestic: All
Student Level: Undergraduate
New to UBC: All

Would you like to see....
Headcount

As of
November 1

Campus	Faculty	Domestic	International	Grand Total
Vancouver	Applied Science	4,008	1,514	5,522
	Arts	10,209	4,660	14,869
	Commerce and Business Administration	3,878	1,686	5,564
	Dentistry	376	5	381
	Education	2,723	169	2,892
	Forestry	641	520	1,161
	Land and Food Systems	1,273	511	1,784
	Law	583	13	596
	Medicine	1,351	8	1,359
	No Faculty	1,380	173	1,553
	Pharmaceutical Sciences	941	4	945
	Science	7,356	2,010	9,366
	Vantage College	1	329	330
	Total	34,720	11,602	46,322
Grand Total		34,720	11,602	46,322

This dashboard shows Winter session Headcount and 30 Credit FTE (not calculated for graduate students), which you can select from the "Would you like to see" drop-down menu.

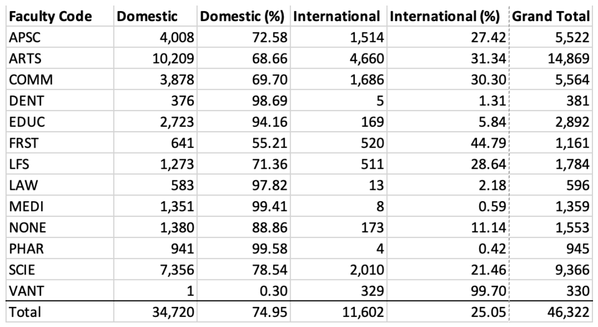
There are two points at which Winter enrolment is measured (see "As of"): November 1, which is used for most enrolment reporting, including both the Enrolment Report and the Fact Sheets; and March 1, which (along with Summer session, not shown here) is used for the Audited FTE report.

The faculty and campus shown is that which administers the student's program. Students may take courses in other faculties.

Further explanation of terms is available on the Definitions page at http://pair.ubc.ca/definitions/.

You can also download the aggregated data in .csv format here: http://pair.ubc.ca/data-access/download-csv-files/

(a) Original data [3]



Faculty Code	Domestic	Domestic (%)	International	International (%)	Grand Total
APSC	4,008	72.58	1,514	27.42	5,522
ARTS	10,209	68.66	4,660	31.34	14,869
COMM	3,878	69.70	1,686	30.30	5,564
DENT	376	98.69	5	1.31	381
EDUC	2,723	94.16	169	5.84	2,892
FRST	641	55.21	520	44.79	1,161
LFS	1,273	71.36	511	28.64	1,784
LAW	583	97.82	13	2.18	596
MEDI	1,351	99.41	8	0.59	1,359
NONE	1,380	88.86	173	11.14	1,553
PHAR	941	99.58	4	0.42	945
SCIE	7,356	78.54	2,010	21.46	9,366
VANT	1	0.30	329	99.70	330
Total	34,720	74.95	11,602	25.05	46,322

(b) Modified data with ratios

Figure 2: Enrolment by Faculty

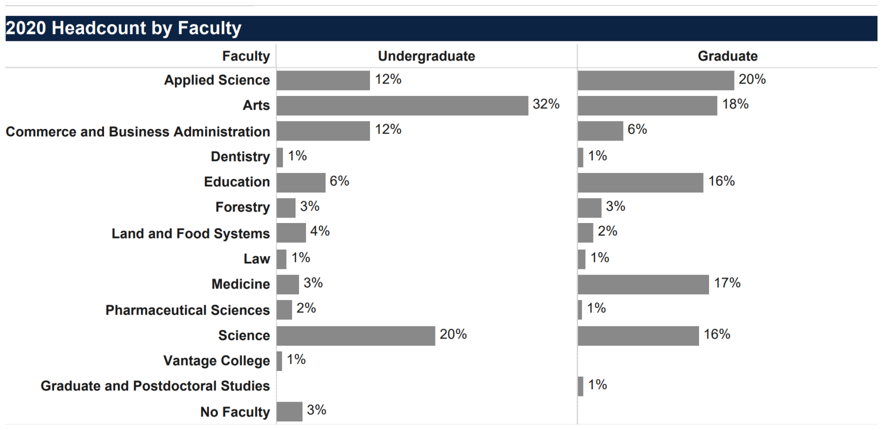
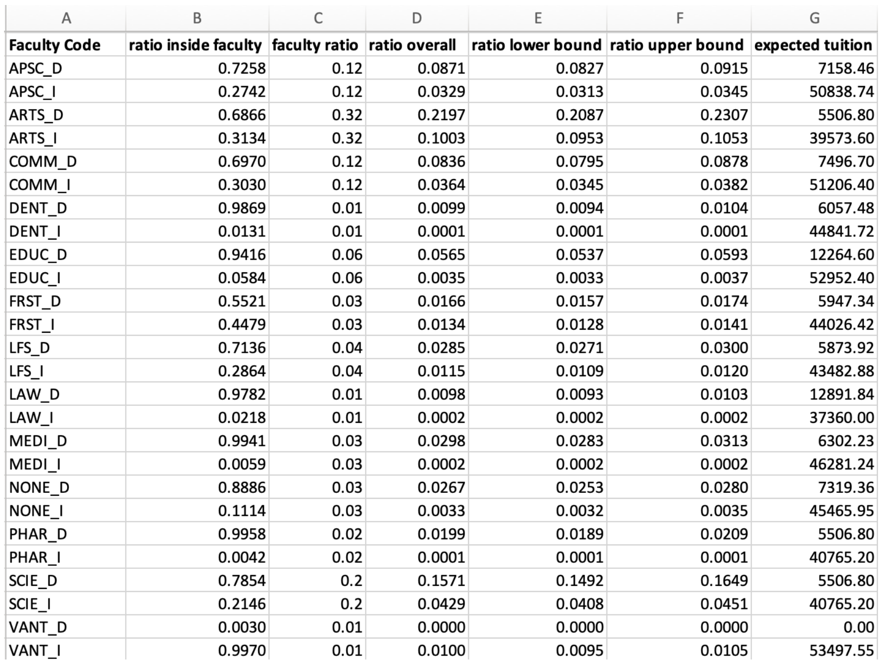


Figure 3: Faculty Ratios

Now each faculty's domestic/international student ratio with respect to the entire undergraduate program can be calculated: multiply the ratio of domestic or international student ratio within each faculty with the ratio of that faculty with respect to the undergraduate program.

Example calculation for APSC_D: the domestic student ratio within the faculty is 0.7258 and the ratio of APSC with respect to UBC is 0.12. So the ratio of APSC domestic students with respect to UBC is $0.7258 \cdot 0.12 \approx 0.0871$. This means that about 8.7% of all undergraduate UBC students are domestic APSC students.

The calculated ratios are based on previous year's data. But the statistics will always vary slightly from year to year. A $\pm 5\%$ bound is added to these ratios, which allows fluctuations in enrolment ratio of each faculty and make the linear program more realistic and flexible. Thus the lower bound is set to be 0.95 times the calculated ratio, and the upper bound is set to be 1.05 times the calculated ratio. The resulting bounds of each decision variable are displayed in columns E and F of Figure 4. For example, the bounds of APSC_D is $0.0827 \leq \text{APSC\_D} \leq 0.0915$.



A	B	C	D	E	F	G
Faculty Code	**ratio inside faculty**	**faculty ratio**	**ratio overall**	**ratio lower bound**	**ratio upper bound**	**expected tuition**
APSC_D	0.7258	0.12	0.0871	0.0827	0.0915	7158.46
APSC_I	0.2742	0.12	0.0329	0.0313	0.0345	50838.74
ARTS_D	0.6866	0.32	0.2197	0.2087	0.2307	5506.80
ARTS_I	0.3134	0.32	0.1003	0.0953	0.1053	39573.60
COMM_D	0.6970	0.12	0.0836	0.0795	0.0878	7496.70
COMM_I	0.3030	0.12	0.0364	0.0345	0.0382	51206.40
DENT_D	0.9869	0.01	0.0099	0.0094	0.0104	6057.48
DENT_I	0.0131	0.01	0.0001	0.0001	0.0001	44841.72
EDUC_D	0.9416	0.06	0.0565	0.0537	0.0593	12264.60
EDUC_I	0.0584	0.06	0.0035	0.0033	0.0037	52952.40
FRST_D	0.5521	0.03	0.0166	0.0157	0.0174	5947.34
FRST_I	0.4479	0.03	0.0134	0.0128	0.0141	44026.42
LFS_D	0.7136	0.04	0.0285	0.0271	0.0300	5873.92
LFS_I	0.2864	0.04	0.0115	0.0109	0.0120	43482.88
LAW_D	0.9782	0.01	0.0098	0.0093	0.0103	12891.84
LAW_I	0.0218	0.01	0.0002	0.0002	0.0002	37360.00
MEDI_D	0.9941	0.03	0.0298	0.0283	0.0313	6302.23
MEDI_I	0.0059	0.03	0.0002	0.0002	0.0002	46281.24
NONE_D	0.8886	0.03	0.0267	0.0253	0.0280	7319.36
NONE_I	0.1114	0.03	0.0033	0.0032	0.0035	45465.95
PHAR_D	0.9958	0.02	0.0199	0.0189	0.0209	5506.80
PHAR_I	0.0042	0.02	0.0001	0.0001	0.0001	40765.20
SCIE_D	0.7854	0.2	0.1571	0.1492	0.1649	5506.80
SCIE_I	0.2146	0.2	0.0429	0.0408	0.0451	40765.20
VANT_D	0.0030	0.01	0.0000	0.0000	0.0000	0.00
VANT_I	0.9970	0.01	0.0100	0.0095	0.0105	53497.55

Figure 4: Decision Variables Data

The percentage of fluctuation can be modified to explore whether the change will have an affect on the outcome of the linear program.

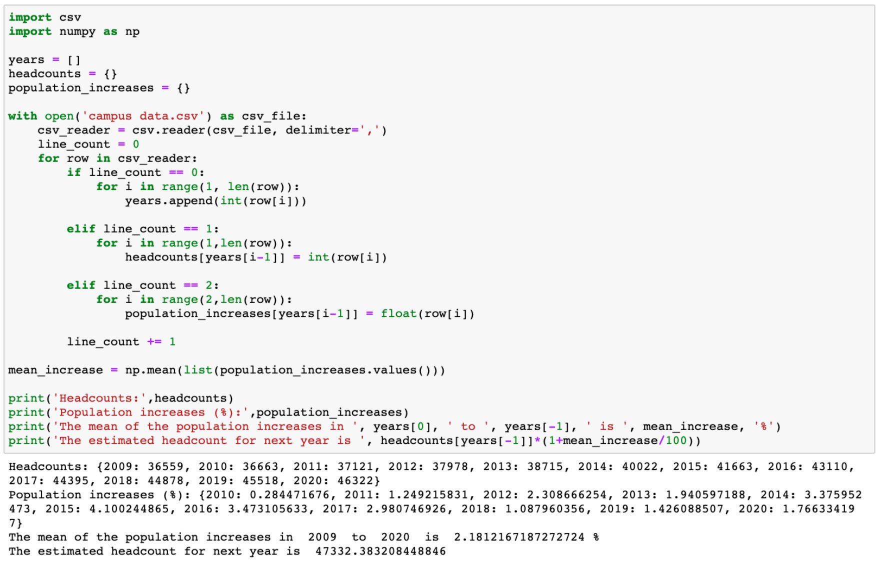
PuLP Implementation

Because of the large amount of decision variables and constraints as well as large coefficients, computer technology is used to solve the linear program. In particular, the linear program is solved using Python's PuLP package in Jupyter Notebook.

The PuLP linear program is set up as section Problem Modelling suggests, expect that the objective function is simply

$$z = \sum_{i=1}^{26} c_i x_i \tag{3}$$

The constant yet uncertain term H is excluded here because the estimated maximum total tuition revenue can be obtained simply by multiplying the solved optimal value with H. H may also be altered to see its effect on the overall tuition revenue.

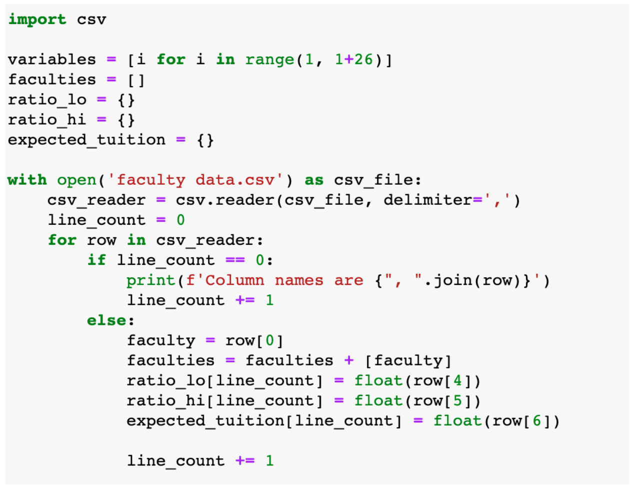


```
import csv
import numpy as np

years = []
headcounts = {}
population_increases = {}

with open('campus data.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0
    for row in csv_reader:
        if line_count == 0:
            for i in range(1, len(row)):
                years.append(int(row[i]))

        elif line_count == 1:
            for i in range(1,len(row)):
                headcounts[years[i-1]] = int(row[i])

        elif line_count == 2:
            for i in range(2,len(row)):
                population_increases[years[i-1]] = float(row[i])

        line_count += 1

mean_increase = np.mean(list(population_increases.values()))

print('Headcounts:',headcounts)
print('Population increases (%):',population_increases)
print('The mean of the population increases in ', years[0], ' to ', years[-1], ' is ', mean_increase, '%')
print('The estimated headcount for next year is ', headcounts[years[-1]]*(1+mean_increase/100))
```

```
Headcounts: {2009: 36559, 2010: 36663, 2011: 37121, 2012: 37978, 2013: 38715, 2014: 40022, 2015: 41663, 2016: 43110,
2017: 44395, 2018: 44878, 2019: 45518, 2020: 46322}
Population increases (%): {2010: 0.284471676, 2011: 1.249215831, 2012: 2.308666254, 2013: 1.940597188, 2014: 3.375952
473, 2015: 4.100244865, 2016: 3.473105633, 2017: 2.980746926, 2018: 1.087960356, 2019: 1.426088507, 2020: 1.76633419
7}
The mean of the population increases in  2009  to  2020  is  2.1812167187272724 %
The estimated headcount for next year is  47332.383208448846
```

Figure 5: PuLP Implementation on H

Although the constant H is excluded from the PuLP linear program, the estimate of H is still calculated in Jupyter Notebook. Using the historical headcount of undergraduate students from year 2009 to 2020, the percentage increase in headcount each year can be obtained, thus the mean percentage increase can be calculated [3]. The estimated headcount for the next year can thus be obtained, which is calculated to be 47332. Python computation is shown in Figure 5. Here 'campus data.csv' contains the year, headcount and percentage increase in headcount information for each year from 2009 to 2020.



```
import csv

variables = [i for i in range(1, 1+26)]
faculties = []
ratio_lo = {}
ratio_hi = {}
expected_tuition = {}

with open('faculty data.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0
    for row in csv_reader:
        if line_count == 0:
            print(f'Column names are {", ".join(row)}')
            line_count += 1
        else:
            faculty = row[0]
            faculties = faculties + [faculty]
            ratio_lo[line_count] = float(row[4])
            ratio_hi[line_count] = float(row[5])
            expected_tuition[line_count] = float(row[6])

            line_count += 1
```

Figure 6: PuLP Implementation on `faculties` and Decision Variables

Each decision variable $x_i, i \in \{1, ..., 26\}$, represents the ratio of the students represented by column Faculty Code in Figure 4, in that order. So x_1 = APSC_D, x_2 = APSC_I and x_{26} = VANT_I. In the PuLP implementation, for $i \in \{1, ..., 26\}$, the result of `faculties[i-1]` represents the interpretation of decision variable x_i. The implementation of `faculties` array is shown in Figure 6.

Besides solving the linear program and estimating the value of H as described above, the dual value and slack value of each constraint are also computed in Jupyter Notebook.

The full PuLP implementation of the linear program, including the computation of estimated H value, is in Appendix.

Results

Optimal Solution and Optimal Value

The linear program, whose objective function is as described in section PuLP Implementation, is computed to have an optimal solution. The optimal solution and optimal value are obtained in cell [7] of Jupyter Notebook (see Appendix). The value for each decision variable is the optimal headcount ratio of each decision variable with respect to the total headcount, H.

For example, $x_1^* = 0.0898$, which means that within the constraints of the possible fluctuation, the optimal ratio of APSC_D's headcount with respect to the overall headcount H is 0.0898. In other words, enrolling 8.98% of all of undergraduate UBC students to be domestic APSC students would maximize UBC's tuition revenue. The overall headcount in the undergraduate program is estimated to be $H = 47322$ (see PuLP Implementation), thus about $0.0898 \cdot 47332 \approx 4250$ domestic applied science students are suggested to get enrolled. The same calculation applies to all decision variables, and the summation of all the optimal solutions after the above calculation is equal to 47322.

The optimal value is $z^* \approx 16729.14$, which represents the maximum expected tuition that UBC receives from one student on average. Therefore the maximum total tuition revenue is $47322 \cdot 16729.14 \approx 791656363.08$.

Slack and Surplus of the Constraints

The slack value of a constraint is the amount of resources as represented by the less-or-equal-to parts of the constraint that is not being used. The surplus is the extra amount of resources over the greater-or-equal-to constraint that is being produced or utilized. When the slack value and surplus values are zero, all of the resources are being used with no leftovers and there is no need for extras [6].

In cell [8] in Jupyter Notebook, the values displayed after "slack:" are the slack value of each constraint. When the slack value is positive, there is a slack. When the slack value is negative, the constraint has a surplus.

For example, constraints `_C1` and `_C2` are the lower and upper bounds of the decision variable x_1, respectively. `_C1` has a slack value of -0.0071, which means that this constraint has a surplus. That is, the optimal solution of the corresponding decision variable (i.e.: x_1) is greater than its lower bound by 0.0071. On the other hand, constraint `_C2`, which has a slack value of 0.0017, has a slack. The optimal solution of the corresponding decision variable (i.e.: x_1) is less than its upper bound by 0.0017.

Since the absolute value of the slack value of the constraint that represents APSC_D's lower bound (i.e.: `_C1`) is greater than the absolute value of the slack value of the constraint that represents APSC_D's upper bound (i.e.: `_C2`), Therefore the data indicates that the optimal solution is closer to the upper bound. Hence it is more efficient at maximizing revenue to enroll more domestic students into applied science next year compare to this year. Again, the optimal solution proofs the analogy. The original APSC_D's overall ratio calculated was 0.0871, whereas from the optimal solution for APSC_D (i.e.: x_1) is 0.0898 which is greater than the original overall ratio. Therefore it is shown that the result of the forecast (optimal solution) indicates that to maximize revenue, applied science faculty should enroll more domestic students into next year compare to this year.

The same analogy applies to all other decision variables. By monitoring the absolute values of slack and surplus values of each constraint, there are three conclusions:

- $|\text{Slack}| < |\text{Surplus}| \Rightarrow$ Closer to upper bound, therefore more enrolment next year.
- $|\text{Slack}| > |\text{Surplus}| \Rightarrow$ Closer to lower bound, therefore less enrolment next year.
- $|\text{Slack}| = |\text{Surplus}| \Rightarrow$ Remains at center, therefore no change in enrolment next year.

Furthermore, when either slack or surplus equals to zero, the enrolment for next year is at the upper or lower bound. See cell [9] of the Jupyter Notebook.

Duality of the Constraints

The linear program can be turned into standard form:

$$\max\ z = \sum_{i=1}^{26} c_i x_i$$
$$\text{Subject to } A\vec{x} \leq \vec{b},$$
$$\vec{x} \geq \vec{0}$$

where c_i's and x_i's are defined same as the previous sections, and $\vec{b} \in \mathbb{R}^{54}$ (the = constraint has now turned into two constraints) represents the ratio bounds. Call this linear program (P).

The dual program of (P), denoted as (D), is:

$$\min\ w = \sum_{j=1}^{54} b_j y_j$$
$$\text{Subject to } A^T\vec{y} \geq \vec{c},$$
$$\vec{y} \geq \vec{0}$$

where y_j's represent the tuition per headcount ratio. The objective function of D is the least amount of revenue that UBC should be earning next year.

Since all optimal solutions $x_i^*, i \in \{1, ..., 26\}$ are greater than zero except x_{25}^*, then by complementary slackness, the majority of the constraints of the dual problem are equalities (i.e.: $A^T\vec{y} = \vec{c}$) instead of inequalities except the constraint corresponding to x_{25}.

In cell [8] in Jupyter Notebook, the values displayed after "shadow:" are the dual value of each constraint. It means that adding 1 to the value of the constraint will increase (or decrease) the optimal value by the amount of the dual value.

By Strong duality theorem, the optimal solution of (P) $\vec{x}^*$ in cell [7] of Jupyter Notebook and the objective value corresponding to the optimal solution z^* is equal to the objective value w^* of (D) with optimal solution $\vec{y}^*$ printed in cell [8] of Jupyter Notebook.

Sensitivity Analysis

There are two major ways that might have an effect on the optimal solution and optimal value of a linear problem: uncertainty and parameter changes. Since the linear program in this research is a forecast based on many assumptions, there are many possible factors which may affect the actual outcome that may differ from the optimal solution of the forecast.

Uncertainties

Uncertainties may occur in the linear program in many different areas.

For the expected tuition, the modelling of the linear program assumes that the distribution of student headcount in each year level (or in different programs in the same faculty) is uniform. However, the actual distribution may differ between each year. As a result, there are uncertainties to the expected tuition, which would affect the objective coefficients c_i's, for $i \in \{1, ..., 26\}$, hence affect the optimal solution and optimal value.

There is no tuition information for the students registered with no faculties, hence its expected tuition will change if assumptions are made differently, which will in turn affect the outcome of the linear program. Nevertheless, students with no faculties make up only a small portion of the total headcount (3%), therefore the effect should be minimal.

Transfer students may also bring uncertainties to the grand total headcount. However, since only the percentage ratio is considered instead of headcount, the effect is again minimal.

Parameter Changes

Parameter changes can be made to investigate shadow prices. For example, if a change is applied to the upper bound of one decision variable, that is allowing more up-ward fluctuation. (Consider the example above for APSC_D, now increase upper bound by 0.01.

Consider a situation where there is a sudden increase in investment into Applied Science faculty, which allows more resources and provides more seats for domestic students in the faculty of applied science. This increases the upper bound on the decision variable that represents APSC_D. There are two possible outcomes and one of them is more likely to occur:

- The first possible outcome is that the new optimal solution would result in the value of APSC_D to increase from the previous optimal solution. However, as the summation of all decision variable values is equal to 1, therefore all other values of the optimal solution would have to change in order to balance the equation. As a result, the new optimal solution might be completely different to the previous one, with an increase for some decision variables and a decrease for others.

- The second possibility is that the optimal solution remains unchanged. This is because although the upper bound has increased, it is still most efficient to remain in the same ratio for APSC_D. Therefore the optimal solution may remain unchanged.

For both possibilities, the impact on the objective value z depends on the change in $x_i, i \in \{1, ..., 26\}$ compare to the original linear program.

Another parameter change can be applied as changing multiple inequalities at once. This is to simulate that UBC is making a reform of enrolment structure. Then there will certainly be an all new optimal solution that would look significantly different to what we obtained in the solution displayed in Jupyter notebook. However, it is predictable that only raising the amount of international students will result in a possible increase in objective value z, otherwise z is less likely to be impacted significantly. Since international tuition is 10 times that of domestic students.

Consider if there is an external donation towards a specific faculty or program allowing that particular faculty to increase its seating capacity. In this case, all other faculties' overall ratio towards grand total remains unchanged, since the available seats are added strictly to that single program or faculty. As a result, the optimal solution remains unchanged. Whereas the objective function would have a new constant: the amount of seat increase multiplying the expected tuition of that program which the result gives the additional revenue generated by the added seats. Therefore there would only be an increase in the objective value z.

Due to COVID-19, there is a possibility that there will be less incoming international students for the next academic year. Therefore all decision variables corresponding to international students may have their lower bounds decreased. Hence the university may decide to enroll more domestic students, which increases the upper bound for decision variables corresponding to domestic students. As a result, the optimal solution will differ from the original optimal solution. Furthermore, the objective value will likely also decrease, since the tuition of international students is significantly higher than domestic students'. In fact, losing 1 international student will require about 10 domestic students to fulfill the revenue loss. Therefore, it is unlikely that UBC will continue earning a high revenue as the lower bound and upper bound shifts down due to COVID.

Lastly, seeing something UBC has been doing over the last few years, raising the cost-per-credit for international student. In terms of the linear program, it means raising the objective coefficient of all decision variables corresponding to international students (e.g.: APSC_I, ARTS_I, etc.). The resulting impact of the parameter change would only lead to an increase in the final objective value z, all constraints will not be effected. Therefore UBC will earn more revenue.

References

[1] Consolidated Financial Statements 2019-20

[2] Coronavirus disease (COVID-19): Travel restrictions, exemptions and advice

[3] The Planning and Institutional Research Office

[4] Undergraduate tuition fees

[5] Fact Sheet Winter 2020

[6] Slack and Surplus Definition

Appendix

The full PuLP implementation of the linear program as described in section PuLP Implementation, including the outputs, is attached in the next few pages.

project

April 15, 2021

[1]:
```
import sys
!{sys.executable} -m pip install pulp
import pulp
from pulp import *
```

```
Collecting pulp
  Using cached PuLP-2.4-py3-none-any.whl (40.6 MB)
Collecting amply>=0.1.2
  Using cached amply-0.1.4-py3-none-any.whl (16 kB)
Requirement already satisfied: pyparsing in /opt/conda/lib/python3.8/site-
packages (from amply>=0.1.2->pulp) (2.4.7)
Collecting docutils>=0.3
  Using cached docutils-0.17-py2.py3-none-any.whl (575 kB)
Installing collected packages: docutils, amply, pulp
Successfully installed amply-0.1.4 docutils-0.17 pulp-2.4
```

[2]:
```
import csv

variables = [i for i in range(1, 1+26)]
faculties = []
ratio_lo = {}
ratio_hi = {}
expected_tuition = {}

with open('faculty data.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0
    for row in csv_reader:
        if line_count == 0:
            print(f'Column names are {", ".join(row)}')
            line_count += 1
        else:
            faculty = row[0]
            faculties = faculties + [faculty]
            ratio_lo[line_count] = float(row[4])
            ratio_hi[line_count] = float(row[5])
            expected_tuition[line_count] = float(row[6])
            line_count += 1

print(ratio_lo)
print(ratio_hi)
print(expected_tuition)
print(faculties)
```

```
Column names are Faculty Code, ratio inside faculty, faculty ratio, ratio
overall, ratio lower bound, ratio upper bound, expected tuition
{1: 0.0827, 2: 0.0313, 3: 0.2087, 4: 0.0953, 5: 0.0795, 6: 0.0345, 7: 0.0094, 8:
0.0001, 9: 0.0537, 10: 0.0033, 11: 0.0157, 12: 0.0128, 13: 0.0271, 14: 0.0109,
15: 0.0093, 16: 0.0002, 17: 0.0283, 18: 0.0002, 19: 0.0253, 20: 0.0032, 21:
0.0189, 22: 0.0001, 23: 0.1492, 24: 0.0408, 25: 0.0, 26: 0.0095}
{1: 0.0915, 2: 0.0345, 3: 0.2307, 4: 0.1053, 5: 0.0878, 6: 0.0382, 7: 0.0104, 8:
0.0001, 9: 0.0593, 10: 0.0037, 11: 0.0174, 12: 0.0141, 13: 0.03, 14: 0.012, 15:
0.0103, 16: 0.0002, 17: 0.0313, 18: 0.0002, 19: 0.028, 20: 0.0035, 21: 0.0209,
22: 0.0001, 23: 0.1649, 24: 0.0451, 25: 0.0, 26: 0.0105}
{1: 7158.46, 2: 50838.74, 3: 5506.8, 4: 39573.6, 5: 7496.7, 6: 51206.4, 7:
6057.48, 8: 44841.72, 9: 12264.6, 10: 52952.4, 11: 5947.34, 12: 44026.42, 13:
5873.92, 14: 43482.88, 15: 12891.84, 16: 37360.0, 17: 6302.23, 18: 46281.24, 19:
7319.36, 20: 45465.95, 21: 5506.8, 22: 40765.2, 23: 5506.8, 24: 40765.2, 25:
0.0, 26: 53497.55}
['APSC_D', 'APSC_I', 'ARTS_D', 'ARTS_I', 'COMM_D', 'COMM_I', 'DENT_D', 'DENT_I',
'EDUC_D', 'EDUC_I', 'FRST_D', 'FRST_I', 'LFS_D', 'LFS_I', 'LAW_D', 'LAW_I',
'MEDI_D', 'MEDI_I', 'NONE_D', 'NONE_I', 'PHAR_D', 'PHAR_I', 'SCIE_D', 'SCIE_I',
'VANT_D', 'VANT_I']
```

[3]:
```
z = expected_tuition

constraints_single = []
for i in variables:
    constraint = {}
    for j in variables:
        constraint[j] = 0
    constraint[i] = 1
    constraints_single.append(constraint)

constraints_equal = {}
for i in variables:
    constraints_equal[i] = 1
```

[4]:
```
prob = LpProblem("Maximization_of_Tuition_Revenue", LpMaximize)
decision_vars = LpVariable.dicts("x", variables, 0)
```

[5]:
```
prob += lpSum([z[i]*decision_vars[i] for i in variables]), "z"

for j in range(len(constraints_single)):
    prob += lpSum([constraints_single[j][i] * decision_vars[i] for i in␣
↪variables]) >= ratio_lo[j+1]
    prob += lpSum([constraints_single[j][i] * decision_vars[i] for i in␣
↪variables]) <= ratio_hi[j+1]

prob += lpSum([constraints_equal[i] * decision_vars[i] for i in variables]) == 1
```

[6]:
```
print(prob)
```

```
Maximization_of_Tuition_Revenue:
MAXIMIZE
7158.46*x_1 + 52952.4*x_10 + 5947.34*x_11 + 44026.42*x_12 + 5873.92*x_13 +
43482.88*x_14 + 12891.84*x_15 + 37360.0*x_16 + 6302.23*x_17 + 46281.24*x_18 +
7319.36*x_19 + 50838.74*x_2 + 45465.95*x_20 + 5506.8*x_21 + 40765.2*x_22 +
5506.8*x_23 + 40765.2*x_24 + 53497.55*x_26 + 5506.8*x_3 + 39573.6*x_4 +
7496.7*x_5 + 51206.4*x_6 + 6057.48*x_7 + 44841.72*x_8 + 12264.6*x_9 + 0.0
SUBJECT TO
_C1: x_1 >= 0.0827

_C2: x_1 <= 0.0915

_C3: x_2 >= 0.0313

_C4: x_2 <= 0.0345

_C5: x_3 >= 0.2087

_C6: x_3 <= 0.2307

_C7: x_4 >= 0.0953

_C8: x_4 <= 0.1053

_C9: x_5 >= 0.0795

_C10: x_5 <= 0.0878

_C11: x_6 >= 0.0345

_C12: x_6 <= 0.0382

_C13: x_7 >= 0.0094

_C14: x_7 <= 0.0104

_C15: x_8 >= 0.0001
```

```
_C16: x_8 <= 0.0001

_C17: x_9 >= 0.0537

_C18: x_9 <= 0.0593

_C19: x_10 >= 0.0033

_C20: x_10 <= 0.0037

_C21: x_11 >= 0.0157

_C22: x_11 <= 0.0174

_C23: x_12 >= 0.0128

_C24: x_12 <= 0.0141

_C25: x_13 >= 0.0271

_C26: x_13 <= 0.03

_C27: x_14 >= 0.0109

_C28: x_14 <= 0.012

_C29: x_15 >= 0.0093

_C30: x_15 <= 0.0103

_C31: x_16 >= 0.0002

_C32: x_16 <= 0.0002

_C33: x_17 >= 0.0283

_C34: x_17 <= 0.0313

_C35: x_18 >= 0.0002

_C36: x_18 <= 0.0002

_C37: x_19 >= 0.0253

_C38: x_19 <= 0.028

_C39: x_20 >= 0.0032
```

```
_C40: x_20 <= 0.0035

_C41: x_21 >= 0.0189

_C42: x_21 <= 0.0209

_C43: x_22 >= 0.0001

_C44: x_22 <= 0.0001

_C45: x_23 >= 0.1492

_C46: x_23 <= 0.1649

_C47: x_24 >= 0.0408

_C48: x_24 <= 0.0451

_C49: x_25 >= 0

_C50: x_25 <= 0

_C51: x_26 >= 0.0095

_C52: x_26 <= 0.0105

_C53: x_1 + x_10 + x_11 + x_12 + x_13 + x_14 + x_15 + x_16 + x_17 + x_18
 + x_19 + x_2 + x_20 + x_21 + x_22 + x_23 + x_24 + x_25 + x_26 + x_3 + x_4
 + x_5 + x_6 + x_7 + x_8 + x_9 = 1

VARIABLES
x_1 Continuous
x_10 Continuous
x_11 Continuous
x_12 Continuous
x_13 Continuous
x_14 Continuous
x_15 Continuous
x_16 Continuous
x_17 Continuous
x_18 Continuous
x_19 Continuous
x_2 Continuous
x_20 Continuous
x_21 Continuous
x_22 Continuous
x_23 Continuous
x_24 Continuous
```

```
x_25 Continuous
x_26 Continuous
x_3 Continuous
x_4 Continuous
x_5 Continuous
x_6 Continuous
x_7 Continuous
x_8 Continuous
x_9 Continuous
```

[7]:
```
prob.solve()

print("Status:", LpStatus[prob.status])
for i in prob.variables():
    print("{0} = {1}".format(i.name, i.varValue))
print("Objective function z = {0}".format(value(prob.objective)))
```

```
Status: Optimal
x_1 = 0.0898
x_10 = 0.0037
x_11 = 0.0157
x_12 = 0.0141
x_13 = 0.0271
x_14 = 0.012
x_15 = 0.0103
x_16 = 0.0002
x_17 = 0.0283
x_18 = 0.0002
x_19 = 0.028
x_2 = 0.0345
x_20 = 0.0035
x_21 = 0.0189
x_22 = 0.0001
x_23 = 0.1492
x_24 = 0.0451
x_25 = 0.0
x_26 = 0.0105
x_3 = 0.2087
x_4 = 0.1053
x_5 = 0.0878
x_6 = 0.0382
x_7 = 0.0094
x_8 = 0.0001
x_9 = 0.0593
Objective function z = 16729.137522999998
```

```
[8]: for name, C in list(prob.constraints.items()):
         print(name, " slack: ", prob.constraints[name].slack, "; shadow: ", prob.
      ↪constraints[name].pi)
```

```
_C1  slack:  -0.007100000000000009 ; shadow:  -0.0
_C2  slack:  0.0016999999999999932 ; shadow:  -0.0
_C3  slack:  -0.0032000000000000015 ; shadow:  -0.0
_C4  slack:  -0.0 ; shadow:  43680.28
_C5  slack:  -0.0 ; shadow:  -1651.66
_C6  slack:  0.021999999999999992 ; shadow:  -0.0
_C7  slack:  -0.010000000000000009 ; shadow:  -0.0
_C8  slack:  -0.0 ; shadow:  32415.14
_C9  slack:  -0.008300000000000002 ; shadow:  -0.0
_C10  slack:  -0.0 ; shadow:  338.24
_C11  slack:  -0.0036999999999999995 ; shadow:  -0.0
_C12  slack:  -0.0 ; shadow:  44047.94
_C13  slack:  -0.0 ; shadow:  -1100.98
_C14  slack:  0.0009999999999999992 ; shadow:  -0.0
_C15  slack:  -0.0 ; shadow:  -0.0
_C16  slack:  -0.0 ; shadow:  37683.26
_C17  slack:  -0.005600000000000001 ; shadow:  -0.0
_C18  slack:  -0.0 ; shadow:  5106.14
_C19  slack:  -0.00040000000000000002 ; shadow:  -0.0
_C20  slack:  -0.0 ; shadow:  45793.94
_C21  slack:  -0.0 ; shadow:  -1211.12
_C22  slack:  0.0017000000000000001 ; shadow:  -0.0
_C23  slack:  -0.0012999999999999999 ; shadow:  -0.0
_C24  slack:  -0.0 ; shadow:  36867.96
_C25  slack:  -0.0 ; shadow:  -1284.54
_C26  slack:  0.0029 ; shadow:  -0.0
_C27  slack:  -0.0011000000000000003 ; shadow:  -0.0
_C28  slack:  -0.0 ; shadow:  36324.42
_C29  slack:  -0.0010000000000000009 ; shadow:  -0.0
_C30  slack:  -0.0 ; shadow:  5733.38
_C31  slack:  -0.0 ; shadow:  -0.0
_C32  slack:  -0.0 ; shadow:  30201.54
_C33  slack:  -0.0 ; shadow:  -856.23
_C34  slack:  0.0030000000000000027 ; shadow:  -0.0
_C35  slack:  -0.0 ; shadow:  -0.0
_C36  slack:  -0.0 ; shadow:  39122.78
_C37  slack:  -0.002700000000000001 ; shadow:  -0.0
_C38  slack:  -0.0 ; shadow:  160.9
_C39  slack:  -0.00029999999999999999 ; shadow:  -0.0
_C40  slack:  -0.0 ; shadow:  38307.49
_C41  slack:  -0.0 ; shadow:  -1651.66
_C42  slack:  0.0019999999999999983 ; shadow:  -0.0
_C43  slack:  -0.0 ; shadow:  -0.0
```

```
_C44  slack:  -0.0 ; shadow:  33606.74
_C45  slack:  -0.0 ; shadow:  -1651.66
_C46  slack:  0.015699999999999992 ; shadow:  -0.0
_C47  slack:  -0.004299999999999998 ; shadow:  -0.0
_C48  slack:  -0.0 ; shadow:  33606.74
_C49  slack:  -0.0 ; shadow:  -0.0
_C50  slack:  -0.0 ; shadow:  -0.0
_C51  slack:  -0.0010000000000000009 ; shadow:  -0.0
_C52  slack:  -0.0 ; shadow:  46339.09
_C53  slack:  -0.0 ; shadow:  7158.46
```

[9]:
```python
for i in prob.variables():
    var = i.name.replace('x_', '')
    if (i.varValue == ratio_hi[int(var)]):
        print('The optimal value of ', i.name, ' is its upper bound')
```

```
The optimal value of  x_10  is its upper bound
The optimal value of  x_12  is its upper bound
The optimal value of  x_14  is its upper bound
The optimal value of  x_15  is its upper bound
The optimal value of  x_16  is its upper bound
The optimal value of  x_18  is its upper bound
The optimal value of  x_19  is its upper bound
The optimal value of  x_2  is its upper bound
The optimal value of  x_20  is its upper bound
The optimal value of  x_22  is its upper bound
The optimal value of  x_24  is its upper bound
The optimal value of  x_25  is its upper bound
The optimal value of  x_26  is its upper bound
The optimal value of  x_4  is its upper bound
The optimal value of  x_5  is its upper bound
The optimal value of  x_6  is its upper bound
The optimal value of  x_8  is its upper bound
The optimal value of  x_9  is its upper bound
```

[10]:
```python
import csv
import numpy as np

years = []
headcounts = {}
population_increases = {}

with open('campus data.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0
    for row in csv_reader:
        if line_count == 0:
            for i in range(1, len(row)):
                years.append(int(row[i]))

        elif line_count == 1:
            for i in range(1,len(row)):
                headcounts[years[i-1]] = int(row[i])

        elif line_count == 2:
            for i in range(2,len(row)):
                population_increases[years[i-1]] = float(row[i])

        line_count += 1

mean_increase = np.mean(list(population_increases.values()))

print('Headcounts:',headcounts)
print('Population increases (%):',population_increases)
print('The mean of the population increases in ', years[0], ' to ', years[-1],␣
 ↪' is ', mean_increase, '%')
print('The estimated headcount for next year is ',␣
 ↪headcounts[years[-1]]*(1+mean_increase/100))
```

```
Headcounts: {2009: 36559, 2010: 36663, 2011: 37121, 2012: 37978, 2013: 38715,
2014: 40022, 2015: 41663, 2016: 43110, 2017: 44395, 2018: 44878, 2019: 45518,
2020: 46322}
Population increases (%): {2010: 0.284471676, 2011: 1.249215831, 2012:
2.308666254, 2013: 1.940597188, 2014: 3.375952473, 2015: 4.100244865, 2016:
3.473105633, 2017: 2.980746926, 2018: 1.087960356, 2019: 1.426088507, 2020:
1.766334197}
The mean of the population increases in  2009  to  2020  is  2.1812167187272724
%
The estimated headcount for next year is  47332.383208448846
```