

Universidad Politecnica Salesiana

Examen

Nombre: Erika Morocho

Asignatura: Simulación

Objetivo:

- Consolidar los conocimientos adquiridos en clase para desarrollar simulaciones de eventos.

Introducción:

El golpe económico de la crisis sanitaria del corona virus no va a ser cosa de semanas, sino de meses. Dentro de una de las etapas importantes que están a la vuelta de la esquina son las elecciones presidenciales y asambleístas del Ecuador. Para ello se plantea realizar un sistema de regresión que permita identificar cual es la tendencia de los votos en base al manejo de las redes sociales (Twitter y/o Facebook) Las regresiones lineales pueden aprender por sí mismos y en este caso obtener automáticamente esa "recta" que buscamos con la tendencia de predicción. Para hacerlo se mide el error con respecto a los puntos de entrada y el valor "y" de salida real [3].

Diseña y desarrolla un modelo y/o script que permita simular el siguiente caso real:

1. Obtener datos de tendencia de twitter para ello se puede obtener a través del API

- Title: Título del Post/Twitter
 - Word count: la cantidad de palabras del artículo.
 - numero de Links: los enlaces externos que contiene,
 - numero of comments: cantidad de comentarios,
 - numero Shares: compartidos.
 - HashTag
2. Posteriormente se debe seguir un procesos de votación de eventos discretos que se describe a continuación:
- Solo se va a tener en cuenta las elecciones de los asambleístas por el Azuay.
 - Las personas solo tiene un recinto electoral para realizar el proceso.
 - Las personas sólo pueden realizar un proceso de elección por asambleísta del Azuay.
 - La persona se acerca a la mesa electoral y hacen fila en caso de ser necesario.
 - Realiza el voto en un tiempo aleatorio de un partido específico.
 - La persona recibe su certificado votación.

```
In [1]: import tweepy
from time import sleep
from datetime import datetime
from textblob import TextBlob
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import tweepy as tw
import random
import pylab as py
import simpy
import numpy as np
import pandas as pd
from tweepy import OAuthHandler
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from scipy.optimize import curve_fit
import seaborn as sns
%matplotlib inline

In [2]: consumer_key = 'CLxAMh7h8Tgr9H35Uu67V4wDL'
consumer_secret = 'wuiVj5DwQZ8YLPaQm3KtJjXOT1wtnoHQm8KVzyJ0K67aB622H'
access_token = '1330313209114544640-fsKAhyL6g5XVfB6bFLJtqTGTq4PQy'
access_token_secret = 'TqqaXiUvAuRryaR1JzvHP2awZAgogs1JPFU1eOB57w361'
```

```
In [3]: auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)
print(api.me().name)
```

Lisseth

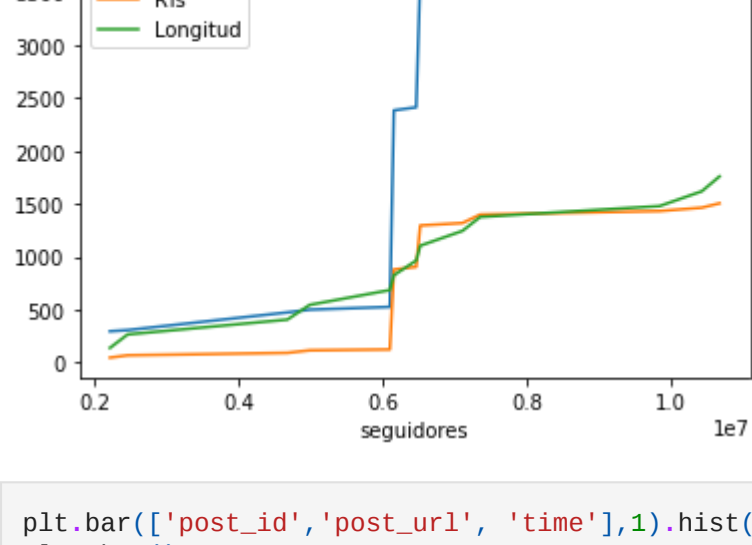
In [4]: Buscar = 'Azuay'

```
In [5]: ts = api.search(Buscar, lang="es", count=100, result_type='popular')
```

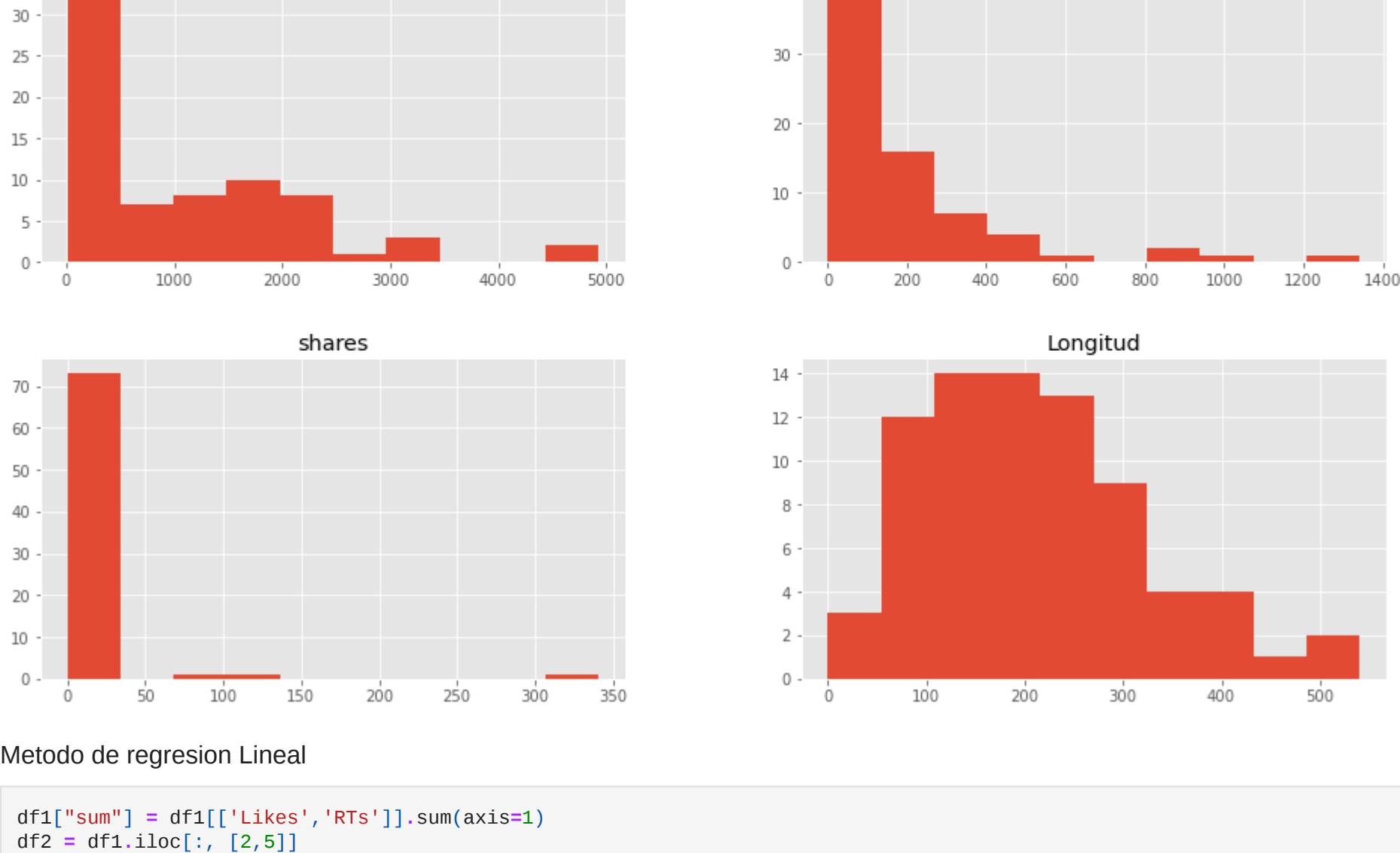
```
In [19]: df = pd.DataFrame({'Usuario': list(map(lambda twee: twee.author.name , ts)),
                        'Fuente': list(map(lambda twee: twee.source, ts)),
                        'Longitud': list(map(lambda twee: len(twee.text), ts)),
                        'RTs': list(map(lambda twee: twee.retweet_count, ts)),
                        'Likes': list(map(lambda twee: twee.favorite_count, ts)),
                        'seguidores': list(map(lambda twee: twee.author.followers_count, ts)),
                        'Fecha': list(map(lambda twee: twee.created_at, ts))})
df['Fecha'] = pd.to_datetime(df.Fecha)
df.sort_values(by='seguidores')
```

	Usuario	Fuente	Longitud	RTs	Likes	seguidores	Fecha
0	El Universo	Hootsuite Inc.	139	47	296	2214479	2020-12-20 13:40:03
1	cnegobec	Twitter for Android	127	22	11	244715	2020-12-20 13:50:33
2	El Universo	Hootsuite Inc.	140	23	167	2214479	2020-12-20 17:01:24
3	Movimiento Alianza PAIS	Twitter for iPhone	139	24	26	309652	2020-12-19 21:57:46
4	Policia Ecuador	Twitter for Android	140	8	26	1111892	2020-12-20 23:11:19
5	Andrés Arauz	Twitter for iPhone	140	755	1856	55903	2020-12-16 17:44:42
6	Movimiento Alianza PAIS	Twitter for Android	139	25	29	309652	2020-12-18 22:29:34
7	Andrés Arauz	Twitter for iPhone	140	391	1236	55903	2020-12-18 03:55:04
8	Riesgos Ecuador	Twitter Web App	140	23	11	585984	2020-12-18 21:03:42
9	cnegobec	Twitter for Android	131	80	43	244715	2020-12-16 19:10:21
10	Ecuavisa	Twitter Media Studio - LiveCut	103	33	46	2487847	2020-12-17 19:02:34
11	Riesgos Ecuador	Twitter Web App	140	32	21	585984	2020-12-14 12:45:07
12	cnegobec	Twitter Web App	140	41	19	244715	2020-12-14 14:13:53

```
In [7]: df1 = df.iloc[:, 2:7].cumsum()
df1.plot(x='seguidores', y=['Likes', 'RTs', 'Longitud'])
```



```
In [92]: plt.bar(['post_id', 'post_url', 'time'],1).hist()
plt.show()
```



Metodo de regresion Lineal

```
In [8]: df1["sum"] = df1[['Likes', 'RTs']].sum(axis=1)
df2 = df1.iloc[:, [2,5]]
df2
```

	Likes	sum
0	296	343
1	307	376
2	474	566
3	500	616
4	526	650
5	2382	3261
6	2411	3315
7	3647	4942
8	3658	4976
9	3701	5099
10	3747	5178
11	3768	5231
12	3787	5291

```
In [9]: x = list(df2.iloc[:, 0]) # Likes
y = list(df2.iloc[:, 1]) # Total

def promedio(x,y):
    return sum(x) / len(y)

def operacion1(x,y):
    #obtiene x menos el promedio de x
    a = x-np.average(x)
    b = y-np.average(y)
    promxy = sum(a*b)
    promxx = sum(a*a)
    result = promxy/promxx
    return result

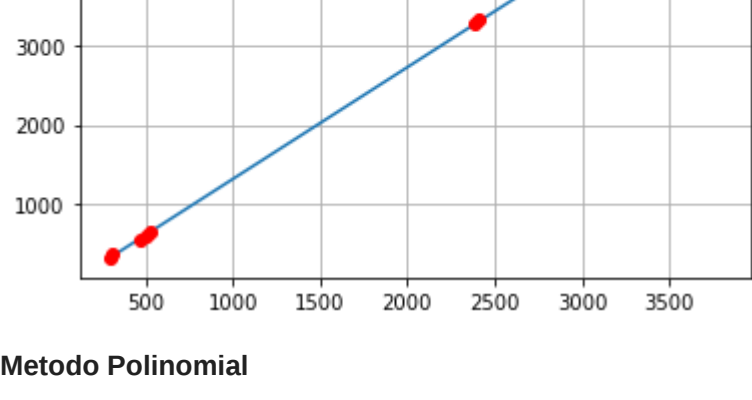
def operacion0(x,y):
    res = np.average(y)-operacion1(x,y)*np.average(x)
    return res

z = api.get_user('XimenaPenap').followers_count

def graficar(x,y,z):
    b1 = operacion1(x,y)
    b0 = operacion0(x,y)
    predecir= b1*z+b0
    puntos_x = np.linspace(x[0],x[-1],6)
    puntos_y = b0+b1*puntos_x
    print("Ecuacion: 'Z=' ,b1,'z','+',b0)
    print("Prediccion: ",predecir)
    plt.plot(puntos_x,puntos_y)
    plt.plot(x,y,"o", color='red')
    plt.grid()
    if __name__=="__main__":
        graficar(x,y,z)
```

Ecuacion: Z= 1.39839329110204 * 236 + -76.51366718030567

Prediccion: 253.5671495197758

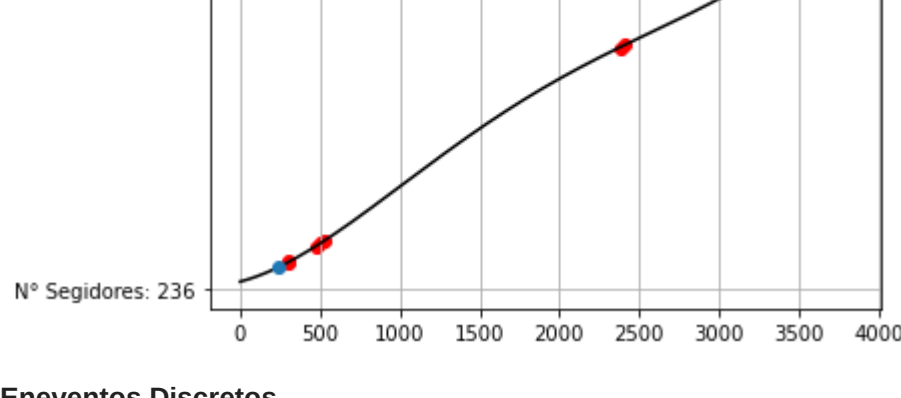


Metodo Polinomial

```
In [56]: pf = PolynomialFeatures(degree = 4)
X = pf.fit_transform(np.array(x).reshape(-1, 1))
regression_lineal = LinearRegression()
regression_lineal.fit(X, y)
pred_x = list(range(0, max(x)+30))
prediccion = regression_lineal.predict(puntos)

def graficar(x,y,z):
    plt.plot(pred_x, prediccion, color='black')
    plt.scatter(x,y,color="red")
    plt.plot(z, prediccion[z], 'o')
    plt.grid()
    plt.plot("N Seguidores: " + str(z))

if __name__=="__main__":
    graficar(x,y,z)
```



Eventos Discretos

```
In [165]: def votacion():
    i=0
    votantes = []
    tiempos_llegada = []
    tiempos_espera = []
    tiempos_votando = []
    tiempos_dando_papeletas = []
    salida = []
    random.seed(1)
    numero_votantes_mesa = 350
    lista_total=[['Usuario','Tiempo llegada','Tiempo de espera','Votando.....','Termina de votar']]
    random.seed(1)
    lasso_digits = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    carrasco_digits = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    lasso_win_pct = 69.13
    carrasco_win_pct = 29.28
    number_of_sims = 1000
    total_wards = 0
    total_carrasco_votes = 0
    total_lasso_votes = 0
    total_votes = 0

    while(i<numero_votantes_mesa):
        if(i==0):
            votantes.append(i)
            tiempos_llegada.append(i)
            tiempos_espera.append(random.random())
            tiempos_votando.append(random.random())
            tiempos_dando_papeletas.append(i)
            tiempo_salida = tiempos_llegada[i] + tiempos_espera[i] + tiempos_votando[i]
            salida.append(tiempo_salida)
            if(i==1):
                votantes.append(i)
                tiempos_llegada.append(tiempos_llegada[i-1]+random.random())
                tiempos_espera.append((salida[i-1]-tiempos_llegada[i])-tiempos_llegada[i])
                tiempos_votando.append(random.random())
                tiempos_dando_papeletas.append(max(salida[i-1], tiempos_llegada[i])-salida[i-1])
                tiempo_salida = tiempos_llegada[i] + tiempos_espera[i] + tiempos_votando[i]
                salida.append(tiempo_salida)
                i=i+1

        i=0

    while(i<numero_votantes_mesa):
        lista=[str(votantes[i]).format(2/3),str(tiempos_llegada[i]).format(10/3),str(tiempos_espera[i]).format(10/3),str(tiempos_votando[i]).format(10/3),str(
        lista_total.append(lista)
        i=i+1
    print(type(lista_total))
    data=pd.DataFrame(lista_total)
    print(data)
    votacion()

    for i in range(number_of_sims):
        my_input = open('shares.txt')
        for line in my_input:
            total_wards += 1
            fields = line.strip().split()
            num_voters = int(fields[0])
            carrasco_votes_in_ward = 0
            lasso_votes_in_ward = 0
            for j in range(num_voters):
                random_num = random.random() * 100
                if random_num <= lasso_win_pct:
                    lasso_votes_in_ward += 1
                    total_lasso_votes += 1
                elif random_num <= (lasso_win_pct + carrasco_win_pct):
                    carrasco_votes_in_ward += 1
                    total_carrasco_votes += 1
                total_votes += 1
            carrasco_digit = int(str(carrasco_votes_in_ward)[0])
            lasso_digit = int(str(lasso_votes_in_ward)[0])
            carrasco_digits[carrasco_digit] += 1
            lasso_digits[lasso_digit] += 1
            if i % 100 == 0:
                print('Simulacion con ', i)

    carrasco_win_pct = 100.0 * total_carrasco_votes / total_votes
    lasso_win_pct = 100.0 * total_lasso_votes / total_votes
    print('Guillermo Lazo gana con el:', carrasco_win_pct, '%')
    print('Paul Carrasco gana con el:', lasso_win_pct, '%')

<class 'list'>
0 0
1 0
2 0
3 0
4 0
5 0
6 0
7 0
8 0
9 0
10 0
11 0
12 0
13 0
14 0
15 0
16 0
17 0
18 0
19 0
20 0
21 0
22 0
23 0
24 0
25 0
26 0
27 0
28 0
29 0
30 0
31 0
32 0
33 0
34 0
35 0
36 0
37 0
38 0
39 0
40 0
41 0
42 0
43 0
44 0
45 0
46 0
47 0
48 0
49 0
50 0
51 0
52 0
53 0
54 0
55 0
56 0
57 0
58 0
59 0
60 0
61 0
62 0
63 0
64 0
65 0
66 0
67 0
68 0
69 0
70 0
71 0
72 0
73 0
74 0
75 0
76 0
77 0
78 0
79 0
80 0
81 0
82 0
83 0
84 0
85 0
86 0
87 0
88 0
89 0
90 0
91 0
92 0
93 0
94 0
95 0
96 0
97 0
98 0
99 0
100 0
101 0
102 0
103 0
104 0
105 0
106 0
107 0
108 0
109 0
110 0
111 0
112 0
113 0
114 0
115 0
116 0
117 0
118 0
119 0
120 0
121 0
122 0
123 0
124 0
125 0
126 0
127 0
128 0
129 0
130 0
131 0
132 0
133 0
134 0
135 0
136 0
137 0
138 0
139 0
140 0
141 0
142 0
143 0
144 0
145 0
146 0
147 0
148 0
149 0
150 0
151 0
152 0
153 0
154 0
155 0
156 0
157 0
158 0
159 0
160 0
161 0
162 0
163 0
164 0
165 0
166 0
167 0
168 0
169 0
170 0
171 0
172 0
173 0
174 0
175 0
176 0
177 0
178 0
179 0
180 0
181 0
182 0
183 0
184 0
185 0
186 0
187 0
188 0
189 0
190 0
191 0
192 0
193 0
194 0
195 0
196 0
197 0
198 0
199 0
200 0
201 0
202 0
203 0
204 0
205 0
206 0
207 0
208 0
209 0
210 0
211 0
212 0
213 0
214 0
215 0
216 0
217 0
218 0
219 0
220 0
221 0
222 0
223 0
224 0
225 0
226 0
227 0
228 0
229 0
230 0
231 0
232 0
233 0
234 0
235 0
236 0
237 0
238 0
239 0
240 0
241 0
242 0
243 0
244 0
245 0
246 0
247 0
248 0
249 0
250 0
251 0
252 0
253 0
254 0
255 0
256 0
257 0
258 0
259 0
260 0
261 0
262 0
263 0
264 0
265 0
266 0
267 0
268 0
269 0
270 0
271 0
272 0
273 0
274 0
275 0
276 0
277 0
278 0
279 0
280 0
281 0
282 0
283 0
284 0
285 0
286 0
287 0
288 0
289 0
290 0
291 0
292 0
293 0
294 0
295 0
296 0
297 0
298 0
299 0
300 0
301 0
302 0
303 0
304 0
305 0
306 0
307 0
308 0
309 0
310 0
311 0
312 0
313 0
314 0
315 0
316 0
317 0
318 0
319 0
320 0
321 0
322 0
323 0
324 0
325 0
326 0
327 0
328 0
329 0
330 0
331 0
332 0
333 0
334 0
335 0
336 0
337 0
338 0
339 0
340 0
341 0
342 0
343 0
344 0
345 0
346 0
347 0
348 0
349 0
350 0
351 0
352 0
353 0
354 0
355 0
356 0
357 0
358 0
359 0
360 0
361 0
362 0
363 0
364 0
365 0
366 0
367 0
368 0
369 0
370 0
371 0
372 0
373 0
374 0
375 0
376 0
377 0
378 0
379 0
380 0
381 0
382 0
383 0
384 0
385 0
386 0
387 0
388 0
389 0
390 0
391 0
392 0
393 0
394 0
395 0
396 0
397 0
398 0
399 0
400 0
401 0
402 0
403 0
404 0
405 0
406 0
407 0
408 0
409 0
410 0
411 0
412 0
413 0
414 0
415 0
416 0
417 0
418 0
419 0
420 0
421 0
422 0
423 0
424 0
425 0
426 0
427 0
428 0
429 0
430 0
431 0
432 0
433 0
434 0
435 0
436 0
437 0
438 0
439 0
440 0
441 0
442 0
443 0
444 0
445 0
446 0
447 0
448 0
449 0
450 0
451 0
452 0
453 0
454 0
455 0
456 0
457 0
458 0
459 0
460 0
461 0
462 0
463 0
464 0
465 0
466 0
467 0
468 0
469 0
470 0
471 0
472 0
473 0
474 0
475 0
476 0
477 0
478 0
479 0
480 0
481 0
482 0
483 0
484 0
485 0
486 0
487 0
488 0
489 0
490 0
491 0
492 0
493 0
494 0
495 0
496 0
497 0
498 0
499 0
500 0
501 0
502 0
503 0
504 0
505 0
506 0
507 0
508 0
509 0
510 0
511 0
512 0
513 0
514 0
515 0
516 0
517 0
518 0
519 0
520 0
521 0
522 0
523 0
524 0
525 0
526 0
527 0
528 0
529 0
530 0
531 0
532 0
533 0
534 0
535 0
536 0
537 0
538 0
539 0
540 0
541 0
542 0
543 0
544 0
545 0
546 0
547 0
548 0
549 0
550 0
551 0
552 0
553 0
554 0
555 0
556 0
557 0
558 0
559 0
560 0
561 0
562 0
563 0
564 0
565 0
566 0
567 0
568 0
569 0
570 0
571 0
572 0
573 0
574 0
575 0
576 0
577 0
578 0
579 0
580 0
581 0
582 0
583 0
584 0
585 0
586 0
587 0
588 0
589 0
590 0
591 0
592 0
593 0
594 0
595 0
596 0
597 0
598 0
599 0
600 0
601 0
602 0
603 0
604 0
605 0
606 0
607 0
608 0
609 0
610 0
611 0
612 0
613 0
614 0
615 0
616 0
617 0
618 0
619 0
620 0
621 0
622 0
623 0
624 0
625 0
626 0
627 0
628 0
629 0
630 0
631 0
632 0
633 0
634 0
635 0
636 0
637 0
638 0
639 0
640 0
641 0
642 0
643 0
644 0
645 0
646 0
647 0
648 0
649 0
650 0
651 0
652 0
653 0
654 0
655 0
656 0
657 0
658 0
659 0
660 0
661 0
662 0
663 0
664 0
665 0
666 0
667 0
668 0
669 0
670 0
671 0
672 0
673 0
674 0
675 0
676 0
677 0
678 0
679 0
680 0
681 0
682 0
683 0
684 0
685 0
686 0
687 0
688 0
689 0
690 0
691 0
692 0
693 0
694 0
695 0
696 0
697 0
698 0
699 0
700 0
701 0
702 0
703 0
704 0
705 0
706 0
707 0
708 0
709 0
710 0
711 0
712 0
713 0
714 0
715 0
716 0
717 0
718 0
719 0
720 0
721 0
722 0
723 0
724 0
725 0
726 0
727 0
728 0
729 0
730 0
731 0
732 0
733 0
734 0
735 0
736 0
737 0
738 0
739 0
740 0
741 0
742 0
743 0
744 0
745 0
746 0
747 0
748 0
749 0
750 0
751 0
752 0
753 0
754 0
755 0
756 0
757 0
758 0
759 0
760 0
761 0
762 0
763 0
764 0
765 0
766 0
767 0
768 0
769 0
770 0
771 0
772 0
773 0
774 0
775 0
776 0
777 0
778 0
779 0
780 0
781 0
782 0
783 0
784 0
785 0
786 0
787 0
788 0
789 0
790 0
791 0
792 0
793 0
794 0
795 0
796 0
797 0
798 0
799 0
800 0
801 0
802 0
803 0
804 0
805 0
806 0
807 0
808 0
809 0
810 0
811 0
812 0
813 0
814 0
815 0
816 0
817 0
818 0
819 0
820 0
821 0
822 0
823 0
824 0
825 0
826 0
827 0
828 0
829 0
830 0
831 0
832 0
833 0
834 0
835 0
836 0
837 0
838 0
839 0
840 0
841 0
842 0
843 0
844 0
845 0
846 0
847 0
848 0
849 0
850 0
851 0
852 0
853 0
854 0
855 0
856 0
857 0
858 0
859 0
860 0
861 0
862 0
863 0
864 0
865 0
866 0
867 0
868 0
869 0
870 0
871 0
872 0
873 0
874 0
875 0
876 0
877 0
878 0
879 0
880 0
881 0
882 0
883 0
884 0
885 0
886 0
887 0
888 0
889 0
890 0
891 0
892 0
893 0
894 0
895 0
896 0
897 0
898 0
899 0
900 0
901 0
902 0
903 0
904 0
905 0
906 0
907 0
908 0
909 0
910 0
911 0
912 0
913 0
914 0
915 0
916 0
917 0
918 0
919 0
920 0
921 0
922 0
923 0
924 0
925 0
926 0
927 0
928 0
929 0
930 0
931 0
932 0
933 0
934 0
935 0
936 0
937 0
938 0
939 0
940 0
941 0
942 0
943 0
944 0
945 0
946 0
947 0
948 0
949 0
950 0
951 0
952 0
953 0
954 0
955 0
956 0
957 0
958 0
959 0
960 0
961 0
962 0
963 0
964 0
965 0
966 0
967 0
968 0
969 0
970 0
971 0
972 0
973 0
974 0
975 0
976 0
977 0
978 0
979 0
980 0
981 0
982 0
983 0
984 0
985 0
986 0
987 0
988 0
989 0
990 0
991 0
992 0
993 0
994 0
995 0
996 0
997 0
998 0
999 0
1000 0
1001 0
1002 0
1003 0
1004 0
1005 0
1006 0
1007 0
1008 0
1009 0
1010 0
1011 0
1012 0
1013 0
1014 0
1015 0
1016 0
1017 0
1018 0
1019 0
1020 0
1021 0
1022 0
1023 0
1024 0
1025 0
1026 0
1027 0
1028 0
1029 0
1030 0
1031 0
1032 0
1033 0
1034 0
1035 0
1036 0
1037 0
1038 0
1039 0
1040 0
1041 0
1042 0
1043 0
1044 0
1045 0
1046 0
1047 0
1048 0
1049 0
1050 0
1051 0
1052 0
1053 0
1054 0
1055 0
1056 0
1057 0
1058 0
1059 0
1060 0
1061 0
1062 0
1063 0
1064 0
1065 0
1066 0
1067 0
1068 0
1069 0
1070 0
1071 0
1072 0
1073 0
1074 0
1075 0
1076 0
1077 0
1078 0
1079 0
1080 0
1081 0
1082 0
1083 0
1084 0
1085 0
1086 0
1087 0
1088 0
1089 0
1090 0
1091 0
1092 0
1093 0
1094 0
1095 0
1096 0
1097 0
1098 0
1099 0
1100 0
1101 0
1102 0
1103 0
1104 0
1105 0
1106 0
1107 0
1108 0
1109 0
1110 0
1111 0
1112 0
1113 0
1114 0
1115 0
1116 0
1117 0
1118 0
1119 0
1120 0
1121 0
1122 0
1
```