Big Data Mining 2021-22 - Final Assignment

Instructions: ***Fill your ID Here:*** *205614845*
Work on the assignment and submit your solution *individually*.
No sharing of information on the assignment is allowed between student.

Format: *Fill code, text explanations and output (figures, tables ..) in the designated places. Fo r some questions, the code you fill should run in this .ipynb notebook and generate the output automatically after running (e.g. in* `google colab`*). For others, you will need to run comman ds in other environments(e.g. unix) - in this case, just copy the commands and the relevant out puts in the designated text blocks.*
Submit your filled solution by July 28th 23:59 your solution on moodle.

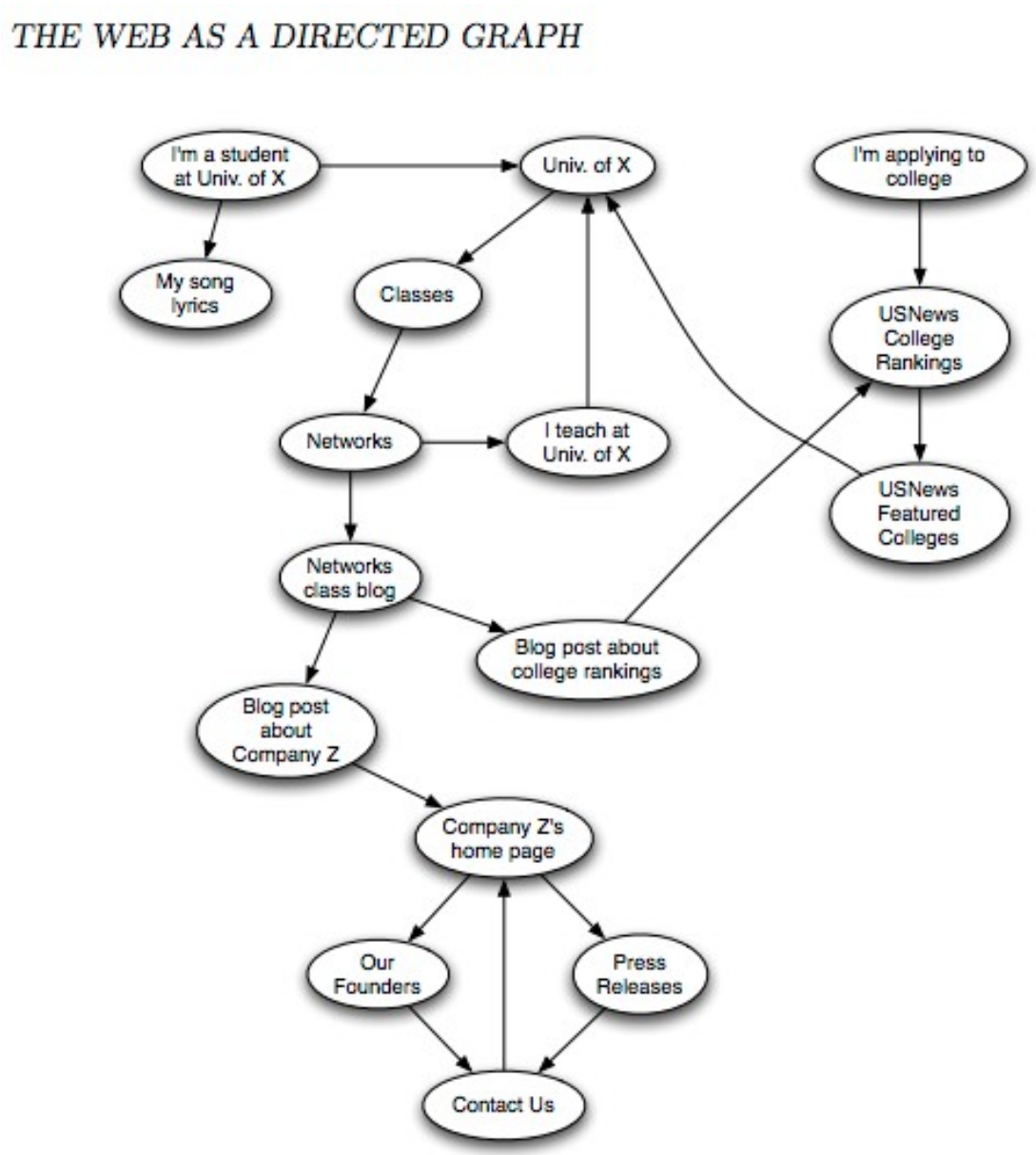
Grading: *There are overall three questions. The number of points is shown for each question and sub-question.* **Note:** Points from your grade may be deducted for submitting wrong/ missing parts of files OR if not submitting the complete generated/complied output.
* **Note:** Some parts of the code may take up to several hours to run.Be patient. However, do n't leave everything to run at the last minute but prepare in advance so that your entire sol ution runs and finishes on time.

- **Good luck!**

Question 1: Build a Network by Web-Crawling [30pt]

In this question we will build a network of web-pages that is part of the World-Wide-Web. This is a directed graph, where each node corresponds to a web-page, and each edge from node a to node b corresponds to an outgoing link from page a to page b. To do so, we will run a web-crawler (a similar well known related concept is web-scraper), using the unix command `scrapy`. The web-crawler starts at a certain starting page, finds all outgoing hyperlinks, and goes to the corresponding pages. It then does the same for each outgoing page, recuresively.

For example, suppose that we crawl the following network of webpages:



and suppose that we start the crawler at the page called `"Networks"`. Then at the first step, the crawler will extract the webpages `"I teach at Univ. of X"` and `"Networks class blog"`. Next, it will also extract the webpages `"Univ. of X"`, `"Blog post about Company Z"` and `"Blog post about college rankings"`. The crawler will never extract the pages `"My song lyrics"` and `"I'm applying to college"`, no matter how long it will run.

a. [6pt] Clone the PageRankScrapy repository from github to a local directory. Install the `scrapy` crawler using `pip install scrapy` in the unix command line. You may need to install additional software before like `pip` - follow the instructions in the Follow the `Readme` file of the `PageRankScrapy` for more details.

After installation, run the `scrapy` command from the unix command line, to start at the wikipedia page: `https://en.wikipedia.org/wiki/Statistics`, and extract only pages that are connected from it and belong to the domain `https://en.wikipedia.org/wiki` (i.e. we restrict ourselves only to `wikipedia` pages with url starting at this string). Set the number of pages crawled to **one million**. (**Hint:** You can copy the command available in the Readme file of the above repository, and change it to fullfil the question's instructions).

If successful, the `scrapy` command will fill two local files:

1. `keyvalue` with the node IDs and names (page urls)
2. `transition` with link information: each line contains a node ID for a source page and the node IDs of all pages that can be reached from the source by a single hyperlink.

After finishing/stopping the crawler, display the first five lines of the `keyvalue` and `transition` output files.

Note: The files are filled as the command is running, and running the crawler on the entire wikipedia may take too long. However, extracting the first **one million** distinct pages (nodes) should not take more than a couple of hours with a stable internet connection. The crawler may include many pages as nodes without their outgoing hyperlinks (edges). This is fine, we will just use the nodes and edges retrieved by the crawler as our network for further analysis.

b. [6pt] In unix, parse the resulting `transition` file to a file with the same information called `network`, but in which each line corresponds to a `single edge`, with two columns showing the IDs of the two corresponding nodes, such that for a directed edge $a \to b$ we have a in the first column and b in the second column. For example the line: 4 2,5,9

should be replaced by the following three lines:
4 2 4 5 4 9

For the resulting file, display the (ordered) pairs of urls corresponding to the first 10 rows (edges). You will need to use also the `keyvalue` file.

c. [6pt] The web-crawler works by parsing the text of a webpage, finding all the (outgoing) hyperlinks, and adding them as directed edges before moving to the corrsponding pages. Suppose that a webpage a has two or more *identical* hyperlinks, i.e. pointing to the *same* page. Will the crawler notice this and output only one edge, or will it display the target page twice or more? Write a unix command that checks using the file `transition` which of the two options is correct, and write your conclusion (we are just interested in `detection`. No need to change the file and remove multiple identical edges if there are any).

d. [12pt] Compute and report basic statistics for the network, using unix commands. You may use for each statistic either the `transition` file format or the `network` file format, or both, as desired.

1. Report the total number of nodes. What is the number of nodes with at least one outgoing edge? Does it match what you expected?
2. What is the total number of (directed) edges (corresponding to hyperlinks)
3. Compute the out-degree of each node, sort the nodes by their computed out-degree, and output the top 10 nodes with the highest out-degree (show both ID and url). **Note:** You may ignore here nodes with zero out-degree
4. Compute the in-degree of each node, sort the nodes by their computed in-degree, and output the top 10 nodes with the highest in-degree (show both ID and url)

Write the commands that you use and the corresponding outputs in the designated **text** blocks below.

Question 1 Solutions:

a. Unix command/s for Web Crawling: `sudo apt update`

```
sudo apt install python3-pip
```

```
pip install scrapy
```

```
cd /mnt/c/Users/orelb/Downloads/DataMiningQ1
```

```
cd scrapy
```

```
scrapy crawl pagerank -a
start='https://en.wikipedia.org/wiki/Statistics' -a
domain='en.wikipedia.org' -a maxpages=47000
```

```
cd /mnt/c/Users/orelb/Downloads/DataMiningQ1/scrapy
```

```
head -5 keyvalue
```

```
head -5 transition
```

First 5 lines of resulting file:

```
0       https://en.wikipedia.org/wiki/Category:Statistics 1
https://en.wikipedia.org/wiki/Help:Category 2
https://en.wikipedia.org/wiki/Portal:Mathematics 3
https://en.wikipedia.org/wiki/Help:Categories 4
https://en.wikipedia.org/wiki/Statistics
```

```
0       1,2,3,4,5,6,7,8,0,9,10,11,12,13,14,15,16 4
0,2,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,
39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,6
2,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85
,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,101,102,103,104,105,106
,107,108,109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,1
24,125,126,127,128,129,130,131,132,133,134,135,136,137,138,139,140,141
,142,143,144,145,146,147,148,149,150,151,152,153,154,155,156,157,158,1
59,160,161,162,163,164,165,166,167,168,169,170,171,172,173,174,175,176
,177,178,179,180,181,182,183,184,185,186,187,188,189,190,191,192,193,1
```

```
94,195,196,197,198,199,200,201,202,203,204,205,206,207,208,209,210,211
,212,213,214,215,216,217,218,219,220,221,222,223,224,225,226,227,228,2
29,230,231,232,233,234,235,236,237,238,239,240,241,242,243,244,245,246
,247,248,249,250,251,252,253,254,255,256,257,258,259,260,261,262,263,2
64,265,266,267,268,269,270,271,272,273,274,275,276,277,278,279,280,281
,282,283,284,285,286,287,288,289,290,291,292,293,294,295,296,297,298,2
99,300,301,302,303,304,305,306,307,308,309,310,311,312,313,314,315,316
,317,318,319,320,321,322,323,324,325,326,327,328,329,330,331,332,333,3
34,335,336,337,338,339,340,341,342,343,344,345,346,347,348,349,350,351
,352,353,354,355,356,357,358,359,360,361,362,363,364,365,366,367,368,3
69,370,371,372,373,374,375,376,377,378,379,380,381,382,383,384,385,386
,387,388,389,390,391,392,393,394,395,396,397,398,399,400,401,402,403,4
04,405,406,407,408,409,410,411,412,413,414,415,416,417,418,419,420,421
,422,423,424,425,426,427,428,429,430,431,432,433,434,435,436,437,438,4
39,440,441,442,443,444,445,446,447,448,449,450,451,452,453,454,455,1,4
56,457,8,4,458,9,10,11,12,13,14,15,16 19
459,173,25,460,75,461,462,463,464,465,466,467,468,469,470,471,178,366,
51,300,472,249,473,251,474,475,476,477,478,479,480,481,482,483,484,485
,486,487,488,489,490,491,492,493,494,495,496,497,498,499,500,501,502,5
03,504,505,506,507,508,509,510,511,512,513,514,515,516,517,518,519,1,8
,19,9,10,11,12,13,14,15,16 48
40,520,49,521,522,523,45,524,525,526,527,528,529,530,531,532,533,534,4
11,535,536,537,538,539,540,541,46,542,335,543,367,544,545,249,250,254,
247,1,8,48,546,9,10,11,12,13,14,15,16 49
40,441,520,547,548,445,269,549,550,551,552,533,553,554,523,555,556,557
,558,559,560,561,562,563,564,565,394,566,567,379,568,569,570,48,571,14
6,572,573,574,575,39,247,264,249,261,576,1,577,8,49,578,9,10,11,12,13,
14,15,16
```

b. Unix command/s for parsing the file and converting to a networks format:

Helpful source: https://www.folkstalk.com/2012/05/awk-command-to-split-column-into-row.html

```
awk '{n=split($2,s,",");for (i=1;i<=n;i++) {$2=s[i];print}}'
transition > network
```

Helpful source: https://unix.stackexchange.com/questions/113898/how-to-merge-two-files-based-on-the-matching-of-two-columns

First 10 lines of resulting file, corresponding to first 10 edges

```
cp network copynetwork cp keyvalue copykeyvalue
```

Helpful source: https://itecnote.com/tecnote/unix-properly-join-two-files-based-on-2-columns-in-common/

```
wk 'NR==FNR{a[$1]=$2;next} ($1) in a{print a[$1]}' copykeyvalue
copynetwork > merge1
```

Helpful source: https://stackoverflow.com/questions/32812916/how-to-delete-the-first-column-which-is-in-fact-row-names-from-a-data-file-in

```
cut -d" " -f2- copynetwork > cnetwork

awk 'NR==FNR{a[$1]=$2;next} ($1) in a{print a[$1]}' copykeyvalue
cnetwork > merge2

paste -d' ' merge1 merge2 > finalmerge

head -10 finalmerge

https://en.wikipedia.org/wiki/Category:Statistics
https://en.wikipedia.org/wiki/Help:Category
https://en.wikipedia.org/wiki/Category:Statistics
https://en.wikipedia.org/wiki/Portal:Mathematics
https://en.wikipedia.org/wiki/Category:Statistics
https://en.wikipedia.org/wiki/Help:Categories
https://en.wikipedia.org/wiki/Category:Statistics
https://en.wikipedia.org/wiki/Statistics
https://en.wikipedia.org/wiki/Category:Statistics
https://en.wikipedia.org/wiki/Correlates_of_crime
https://en.wikipedia.org/wiki/Category:Statistics
https://en.wikipedia.org/wiki/Hawkes_process
https://en.wikipedia.org/wiki/Category:Statistics
https://en.wikipedia.org/wiki/Scagnostics
https://en.wikipedia.org/wiki/Category:Statistics
https://en.wikipedia.org/wiki/Special:MyTalk
https://en.wikipedia.org/wiki/Category:Statistics
https://en.wikipedia.org/wiki/Category:Statistics
https://en.wikipedia.org/wiki/Category:Statistics
https://en.wikipedia.org/wiki/Main_Page
```

c. Unix command/s for checking if there are non unique edges: Helpful source: https://www.makeuseof.com/how-to-find-duplicate-data-in-a-linux-text-file-with-uniq/

```
sort network > sortednetwork uniq -D sortednetwork
```

Conclusion: I used the uniq command to print only repeated Lines. The command print duplicate lines from the text file, by using the -D flag. After using it there was not any line which returned. Therefore my conclusion is that the crawler notices multiple identical edges / hyperlinks and outputs only one time an edge.

d.

1. Commands for extracting numbers of nodes:

Helpful source: https://www.unix.com/unix-for-dummies-questions-and-answers/210445-how-do-i-merge-multiple-columns-into-one-column.html

```
awk '{ for(i=1;i<=NF;i++){ print $i}}' network >  one_column_network

uniq one_column_network > uniqnodes
```

```
wc -l uniqnodes

wc -l transition
```

Resulting numbers of nodes: `22755742 nodes in the network. There are 470000 number of nodes with at least one outgoing edge. This result is not surpriseng. I expected to get this number because in the first question I asked for 470000 sources pages.`

1. Command/s for extracting number of edges: `wc -l network`

Resulting number of edges: `11418578`

1. Command/s for computing&sorting by out-degrees: `awk -F" " '{print $1}' network| sort | uniq -c | sort -k1 -nr | head -10 > top_outedges`

```
awk -F" " '{print $2}' top_outedges > outedges

awk 'NR==FNR{a[$1]=$2;next} ($1) in a{print $1,a[$1]}' copykeyvalue outedges > mergeoutedges

head -10 mergeoutedges
```

Top 10 pages:

```
89778 https://en.wikipedia.org/wiki/Nitrous_oxide 81688
https://en.wikipedia.org/wiki/Ketamine 528502
https://en.wikipedia.org/wiki/1983_in_film 17886
https://en.wikipedia.org/wiki/Portal:World 20389
https://en.wikipedia.org/wiki/Antioch 79483
https://en.wikipedia.org/wiki/Catholics 79444
https://en.wikipedia.org/wiki/Catholic_church 26906
https://en.wikipedia.org/wiki/Catholic 23461
https://en.wikipedia.org/wiki/Catholicism 24050
https://en.wikipedia.org/wiki/Diphenhydramine
```

1. Command/s for computing&sorting by in-degrees: `cut -d" " -f2- network | sort | uniq -c | sort -k1 -nr | head -10 > top_inedges`

```
awk -F" " '{print $2}' top_inedges > inedges

awk 'NR==FNR{a[$1]=$2;next} ($1) in a{print $1,a[$1]}' copykeyvalue inedges > mergeinedges

head -10 mergeinedges
```

Top 10 pages:

```
9 https://en.wikipedia.org/wiki/Main_Page 8
https://en.wikipedia.org/wiki/Special:MyTalk 13
https://en.wikipedia.org/wiki/Wikipedia:Contact_us 12
https://en.wikipedia.org/wiki/Wikipedia:About 11
https://en.wikipedia.org/wiki/Special:Random 10
https://en.wikipedia.org/wiki/Wikipedia:Contents 14
```

https://en.wikipedia.org/wiki/Help:Contents 15
https://en.wikipedia.org/wiki/Help:Introduction 16
https://en.wikipedia.org/wiki/Special:SpecialPages 1
https://en.wikipedia.org/wiki/Help:Category

Question 2: Degrees, Communities, Clustering Coefficient, Random Networks [34pt]

a. [6pt] Extract the DBLP collaboration network file `com-dblp.ungraph.txt`, and load it to the python notebook. This **undirected** network describes collaboration between computer science authors, where each node corresponds to an author, and each edge corresponds to a joint publication of two authors. Compute and report the number of nodes, number of edges, number of triangles and global clustering coefficient of the network. The global clustering coefficient is defined here, as 3 times the number of triangles, divided by the number of triplets, where a triplet is defined as a set of two edges $(x,y),(x,z)$ centered at a joint vertex x. Alternatively, a formula for the global clustering coefficient in terms of the adjacency matrix E and the degrees vector d is:

$$\frac{\sum_{i,j,k=1}^{n} e_{ij}e_{ik}e_{jk}}{\sum_{i=1}^{n} d_i(d_i-1)}$$

b. [6pt] Compute the degree of every node in the network, and list the IDs of the five nodes (authors) with the highest degree. Next, compute and plot the `degree distribution` of the graph, i.e. the relative frequency f_i (out of n) of nodes having degree i for every i from 0 to the maximal degree. You should choose the type of plot that shows the degree distribution clearly.

c. [6pt] Simulate a random graph using the `Erdos-Renyi` model with the edge probability p chosen such that the same number of nodes n as the DBLP network, and such the `expected` number of overall edges in the simulated random graph is equal to the number of edges in the DBLP graph. Compute and plot the degree distribution for the random graph in similar to (b.), either on the same or on separate plots. Is the `Erdos-Renyi` model a good generative model for this graph, based on the degree distribution?

d. [8pt] Run the `asyn_lpa_communities` algorithm available in `networkx` for finding the community structure for the graph, implemented in the `networkx.algorithms.community` python module. Report the number of communities found.

Compare the division to communities found by the algorithm to the ground truth, given in the file `com-dblp.all.cmty.txt`, where each row corresponds to a cummunity. Report the Adjusted Rand Index (ARI). **Explanation:** ARI is an index that measures how well two partitions of the same set (e.g. two clustering results) agree with each other. For example ARI is equal to 1 when the two partitions completely agree with each other, and is equal to

approximately 0 for two random partitions. A python implementation of ARI is available in `scikit-learn`.

Note: When you compare the communities found by our algorithm to the true communities, there are some issues that make the comparison non-ideal. There are missing labels you need to take care of. In particular, put all the nodes that appear in the graph but do not appear in any true community, in a new, separate community. Also, the true communities are overlapping, so some nodes may be present in more than one community. For simplicity, assign them to one of the communities at random.

e. [8pt] Draw 100 random networks with the same degree distribution as the real DBLP network, but with edges between pairs of nodes drawn randomly, using the function `sample_random_graph_with_degree_seq` supplied to you, with at least $100,000$ iteration per random network. Start the randomization of each random network from the preious one. This implementation approximates sampling a graph from the configuration model that is implemented in `networkx`, but is too slow and memory-consuming for our purposes. Compute the global clustering coefficient for each random network, and plot the empirical distribution of the 100 clustering coefficients. How do they compare to the clustering coefficient of the real network computed in (a.)? Is the configuration model a good generative model for the network based on the clustering coefficient? Run also the community detection algorithm as in (d.) for **one** of the 100 random networks. Report the number of communities and the ARI value with respect to the ground truth. How do these values compare to the values reported in (d.)? Are the results surprising?

Write the commands that you use in the designated **code** blocks below. The output should be generated automatically when running the .ipynb notebook (e.g. in `google colab`). You may include additional text blocks with explanations and clarifications.

Importing relevant libraries

```
# importing relevant libraries (from excersice 5 solution)
from operator import itemgetter
import networkx as nx  # networkx - python library for dealing with
graphs
import numpy as np
from matplotlib import pyplot as plt
```

Question 2 Solutions:

a. Code for reading the DBLP network here:

```
# Question 2 Solutions:
# a. Code for reading the DBLP network here:

# Mount my drive so I can upload the network file (needed if working
in colab) (from excersice 5 solution)
from google.colab import drive
drive.mount('/content/drive/')

# Load data
```

```
dblp_route = "/content/drive/MyDrive/com-dblp.ungraph.txt"
G_dblp = nx.read_edgelist(dblp_route)
G_dblp = nx.relabel_nodes(G_dblp, {str(i):i for i in
range(G_dblp.number_of_nodes())}) # change node names to integers
G_dblp

# Code to output the number of nodes, edges, triangles and clustering
coefficient here:

# Compue the number of nodes
nodes_num = G_dblp.number_of_nodes()
print("The number of nodes is", nodes_num)

# Compute the number of edges
edges_num = G_dblp.number_of_edges()
print("The number of edges is", edges_num)

# Compute the number of triangles
# Helpful source:
https://stackoverflow.com/questions/60426256/finding-total-number-of-
triangles-using-networkx
triangles_num = sum(nx.triangles(G_dblp).values()) / 3
print("The number of triangles is", triangles_num)

# Compute the global clustering coefficient
# Helpful source:
https://pynetwork.readthedocs.io/en/latest/connectivity.html
Global_cc = nx.transitivity(G_dblp)
print("The global clustering coefficient", Global_cc)
```

```
Drive already mounted at /content/drive/; to attempt to forcibly
remount, call drive.mount("/content/drive/", force_remount=True).
The number of nodes is 317080
The number of edges is 1049866
The number of triangles is 2224385.0
The global clustering coefficient 0.3063766130023435
```

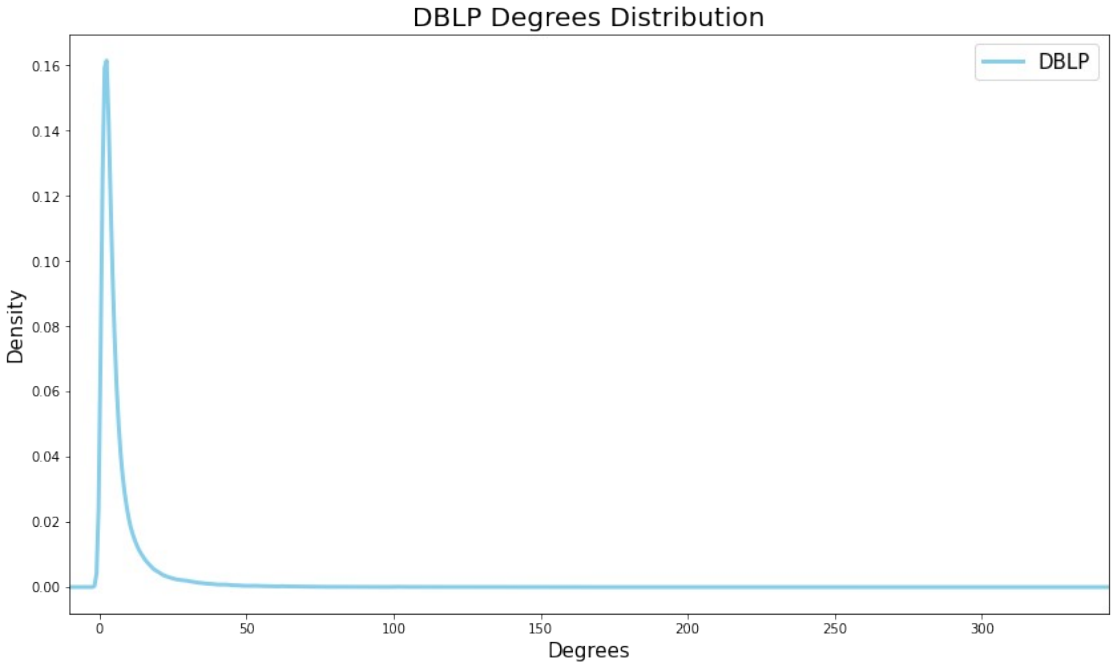
b. Code for Computing and plotting the degree distribution, and for printing the top highest degree nodes:

```
# b. Code for Computing and plotting the degree distribution, and for
printing the top highest degree nodes:

# Helpful source: https://stackoverflow.com/questions/613183/how-do-i-
sort-a-dictionary-by-value

degree_dict1 = dict(nx.degree(G_dblp))
degree_list1 = sorted(degree_dict1.items(), key=lambda x: x[1],
reverse=True)
print("The 5 top highest degree nodes are:")
```

```
for i in range(5):
  print("node", degree_list1[i][0], "with", degree_list1[i][1],
"degree")
```

```
The 5 top highest degree nodes are:
node 38868 with 343 degree
node 45479 with 296 degree
node 57571 with 290 degree
node 6737 with 269 degree
node 72210 with 264 degree
```

```
from matplotlib import pyplot as plt
import pandas as pd

fig = plt.figure(figsize=(14,8))
columns_name = ["Node Number", "Degrees"]
degrees_data1 = nx.degree(G_dblp)
dblp_dist_df = pd.DataFrame(data = degrees_data1, columns =
columns_name)
dblp_dist_df.Degrees.plot.density(color = 'skyblue', linewidth = 3,
label = "DBLP")

# Helpful source: https://stackoverflow.com/questions/3899980/how-to-
change-the-font-size-on-a-matplotlib-plot
plt.rc('legend', fontsize=15)    # legend fontsize

plt.title('DBLP Degrees Distribution',fontsize = 20)
plt.xlabel("Degrees", fontsize = 15)
plt.ylabel("Density", fontsize = 15)

plt.xlim(xmin = -10, xmax = 343)

plt.legend()
plt.show()
```



c. Simulation code for the Erdos-Renyi model

For the network we have $n=317080$ nodes and $M = 1049866 $ edges. The expected number of edges for a $G(n,p)$ Erdos-Renyi graph is:

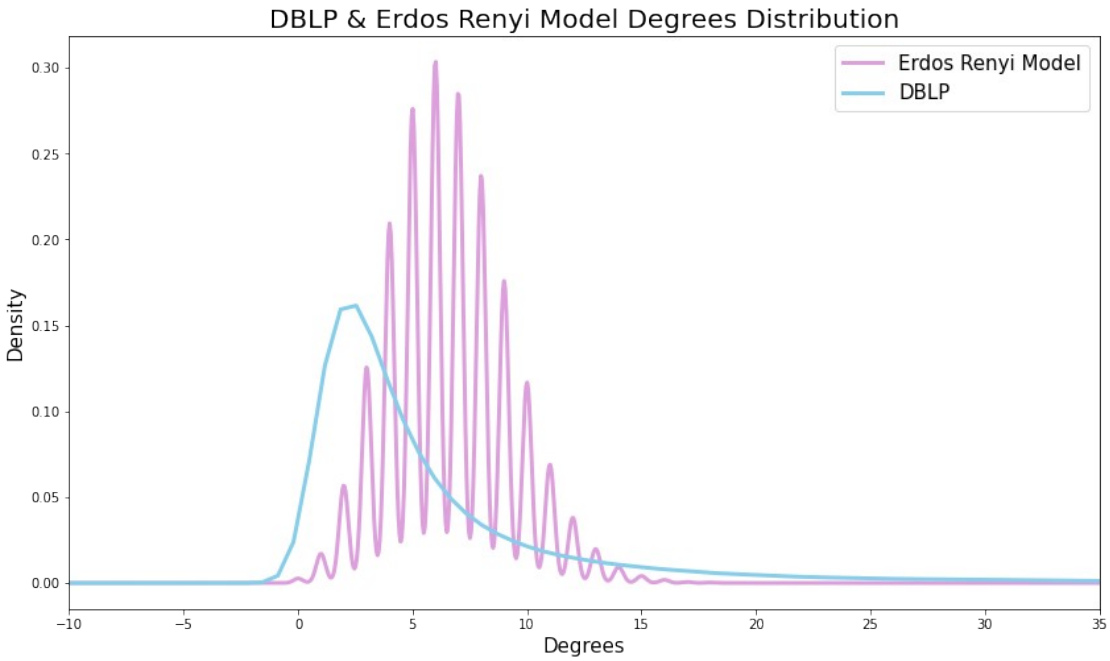
$$\binom{n}{2}p$$

Hence, to match the expectation we get:

$$p=\frac{M}{\binom{n}{2}}$$

```
# c. Simulation code for the Erdos-Renyi model
#
https://networkx.org/documentation/networkx-1.10/reference/generated/
networkx.generators.random_graphs.fast_gnp_random_graph.html#networkx.
generators.random_graphs.fast_gnp_random_graph

nodes_num = G_dblp.number_of_nodes()
edges_num = G_dblp.number_of_edges()

p = ((edges_num))/((nodes_num*(nodes_num-1))/2)

G_Erdos_Renyi = nx.fast_gnp_random_graph(nodes_num, p, seed = None,
directed = False)

# Code for Computing and plotting the degree distribution for the
random graph:
```

```
from matplotlib import pyplot as plt
import pandas as pd

fig = plt.figure(figsize=(14,8))
columns_name = ["Node Number", "Degrees"]
degrees_data2 = nx.degree(G_Erdos_Renyi)

Erdos_Renyi_dist_df = pd.DataFrame(data = degrees_data2, columns =
columns_name)
Erdos_Renyi_dist_df.Degrees.plot.density(color = 'plum', linewidth =
3, label = "Erdos Renyi Model")

dblp_dist_df.Degrees.plot.density(color = 'skyblue', linewidth = 3,
label = "DBLP")

plt.rc('legend', fontsize=15)    # legend fontsize

plt.title("DBLP & Erdos Renyi Model Degrees Distribution" ,fontsize =
20)

plt.xlabel("Degrees", fontsize = 15)
plt.ylabel("Density", fontsize = 15)

plt.xlim(xmin = -10, xmax = 35)

plt.legend()
plt.show()
```



the Erdos-Renyi model is not a good generative model for this graph, based on the degree distribution.

כפי שאנחנו יכולים לראות להתפלגות הדיגרי של המודל יש התפלגות נורמלית במידת מה, בעוד שהתפלגות הדיגרי של הגרף שלנו היא התפלגות גמא עם פרמטר שערכו נמוך.

כפי שניתן לראות מגרף ההתפלגויות - רוב ערכי הדיגרי של הגרף שלנו נמוכים יותר מערכי הדיגרי של המודל

על כן המודל לא מהווה מודל טוב לגרף שלנו, בהתבסס על התפלגות הדיגרי

d. Run the community detection algorithm and report the resulting number of commuunities

```
# d. Run the community detection algorithm and report the resulting
number of commuunities
# Helpful source:
http://cazabetremy.fr/Teaching/DCDclass/Intro_networkx.html
comLabelPropa =
list(nx.algorithms.community.asyn_lpa_communities(G_dblp,
weight=None))
expected_com_num = len(comLabelPropa)
print("The number of communities is", expected_com_num)
```

```
The number of communities is 46417
```

```
# Helpful source: https://www.geeksforgeeks.org/python-converting-all-
strings-in-list-to-integers/

expected_cmty = []
for i in range(expected_com_num):
  expected_cmty.append(list(map(int, comLabelPropa[i])))
```

Load the ground-trush communities and compute and report the ARI between the two

```
# Load the ground-trush communities and compute and report the ARI
between the two
from sklearn.metrics.cluster import adjusted_rand_score

# Mount my drive so I can upload the network file (needed if working
in colab) (from excersice 5 solution)
from google.colab import drive
drive.mount('/content/drive/')

# Load the ground truth file
cmty_route = "/content/drive/MyDrive/com-dblp.all.cmty.txt"

# Using readlines()
cmty_route_file = open(cmty_route, 'r')
Lines = cmty_route_file.readlines()
```

```
true_cmty = []
for line in Lines:
  community_nodes = list(int(x) for x in line.split())
  true_cmty.append(community_nodes)

cmty_route_file.close()
```

```
Drive already mounted at /content/drive/; to attempt to forcibly
remount, call drive.mount("/content/drive/", force_remount=True).
```

```
# Helpful source: https://stackoverflow.com/questions/952914/how-do-i-
make-a-flat-list-out-of-a-list-of-lists

# Convert the list of lists to one big list
flat_true_cmty = [x for xs in true_cmty for x in xs]
# Convert the list of lists to one big list
flat_expected_cmty = [x for xs in expected_cmty for x in xs]

# Helpful source: https://stackoverflow.com/questions/3462143/get-
difference-between-two-lists

# Drop duplicates by converting the list to a set
set_flat_true_cmty = set(flat_true_cmty)
# Drop duplicates by converting the list to a set
set_flat_expected_cmty = set(flat_expected_cmty)

# Finding items that appear in the true communities but not in the
expected communities
diff_cmty1 = [item for item in set_flat_true_cmty if item not in
set_flat_expected_cmty]
# I didnt find any item like these

# Finding items that appear in the expected communities but not in the
true communities
diff_cmty2 = [item for item in set_flat_expected_cmty if item not in
set_flat_true_cmty]
# There are items like these

# I append the list with the new items I found - to the real
communities huge list
true_cmty.append(diff_cmty2)

# creating a dict for the true communities, the key -> is the ID of
the node,
# the value -> is the number of the community

# Note that for duplicate nodes in differents communities, the
community that
# will be chosen is the last community that the node has been in it
```

```
# (this is how I handle duplictes, that way the community that was
choshen for this node is random)

dict_true_cmty = {}

for i in range(len(true_cmty)):
  for j in range(len(true_cmty[i])):
    dict_true_cmty[true_cmty[i][j]] = i + 1

# convert the values of the dictionary (the numbers of the different
# communities) to a list

new_dict_true_cmty = dict(sorted(dict_true_cmty.items()))
list_true_cmty = list(new_dict_true_cmty.values())

# creating a dict for the expected communities, the key -> is the ID
of the node,
# the value -> is the number of the community

# Note that for duplicate nodes in differents communities, the
community that
# will be chosen is the last community that the node has been in it
# (this is how I handle duplictes, that way the community that was
choshen for this node is random)

dict_expected_cmty = {}

for i in range(len(expected_cmty)):
  for j in range(len(expected_cmty[i])):
    dict_expected_cmty[expected_cmty[i][j]] = i + 1

# convert the values of the dictionary (the numbers of the different
# communities) to a list

new_dict_expected_cmty = dict(sorted(dict_expected_cmty.items()))
list_expected_cmty = list(new_dict_expected_cmty.values())

# Calculate the ARI value by using the 2 lists I created - list of
true comminities

ari = adjusted_rand_score(list_true_cmty, list_expected_cmty)
ari

0.0004109619816075436
```

e. Loop and generate 100 random networks, and compute the clustering coefficient for each one

```
# e. Loop and generate 100 random networks, and compute the clustering
coefficient for each one:
```

```
import random
import copy

def sample_random_graph_with_degree_seq(G, iters = 100000):
    G_rand = G # Note! we don't do deep copy here to save time. But
the function can modify G. It is better to keep a separate copy of G
outside the function
    edges = random.sample(list(G_rand.edges()), 2*iters)

    for i in range(0, iters, 2):
        if i%1000 == 0:
            print("Run: ", i)  # you can turn off/on internal printing
        if edges[i] in G_rand.edges() and edges[i+1] in G_rand.edges()
and len(set(edges[0] + edges[1])) == 4: # swap
            G_rand.remove_edges_from([edges[i], edges[i+1]])
            G_rand.add_edges_from([ (edges[i][0], edges[i+1][1]),
(edges[i][1], edges[i+1][0])  ])

    return G_rand

# Calculate the Global clustering coefficient of the first graph, i,e.
the DBLP graph
dblp_global_clustering_coefficient = nx.transitivity(G_dblp)

# Creating a list which will contain all the 100 random graphs' global
clustering coefficients
global_clustering_coefficient_list = []

# Initial the first graph to be input inside the for loop
Graph = G_dblp

# 100 iterations of the for loop
# any previous graph will be the base of creating a new graph inside
the for loop

for i in range(100):
  print(i)  # for help

  # Create a new graph by the helpful function above
  Graph = sample_random_graph_with_degree_seq(Graph)

  # Calculate the global clustering coefficient of the new graph, and
append it to the relevant list
  global_clustering_coefficient_list.append(nx.transitivity(Graph))
```

```
Streaming output truncated to the last 5000 lines.
Run:  49000
Run:  50000
Run:  51000
Run:  52000
```

```
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
51
Run:  0
Run:  1000
```

```
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
```

```
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
52
Run:  0
```

```
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
```

```
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
53
```

```
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
```

```
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
```

```
54
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
```

```
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
```

```
Run:  99000
55
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
```

```
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
```

```
Run:  98000
Run:  99000
56
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
```

```
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
```

```
Run:  97000
Run:  98000
Run:  99000
57
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
```

```
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
```

```
Run:  96000
Run:  97000
Run:  98000
Run:  99000
58
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
```

```
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
```

```
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
59
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
```

```
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
```

```
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
60
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
```

```
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
```

```
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
61
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
```

```
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
```

```
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
62
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
```

```
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
```

```
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
63
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
```

```
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
```

```
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
64
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
```

```
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
```

```
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
65
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
```

```
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
```

```
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
66
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
```

```
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
```

```
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
67
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
```

```
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
```

```
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
68
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
```

```
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
```

```
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
69
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
```

```
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
```

```
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
70
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
```

```
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
```

```
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
71
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
```

```
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
```

```
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
72
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
```

```
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
```

```
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
73
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
```

```
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
```

```
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
74
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
```

```
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
```

```
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
75
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
```

```
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
```

```
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
76
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
```

```
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
```

```
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
77
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
```

```
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
```

```
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
78
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
```

```
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
```

```
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
79
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
```

```
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
```

```
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
80
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
```

```
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
```

```
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
81
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
```

```
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
```

```
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
82
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
```

```
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
```

```
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
83
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
```

```
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
```

```
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
84
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
```

```
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
```

```
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
85
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
```

```
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
```

```
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
86
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
```

```
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
```

```
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
87
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
```

```
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
```

```
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
88
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
```

```
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
```

```
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
89
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
```

```
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
```

```
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
90
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
```

```
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
```

```
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
91
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
```

```
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
```

```
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
92
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
```

```
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
```

```
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
93
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
```

```
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
```

```
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
94
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
```

```
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
```

```
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
95
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
Run:  7000
```

```
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
Run:  57000
```

```
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
96
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
Run:  6000
```

```
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
Run:  56000
```

```
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
97
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
Run:  5000
```

```
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
Run:  55000
```

```
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
98
Run:  0
Run:  1000
Run:  2000
Run:  3000
Run:  4000
```

```
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
Run:  54000
```

```
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
99
Run:  0
Run:  1000
Run:  2000
Run:  3000
```

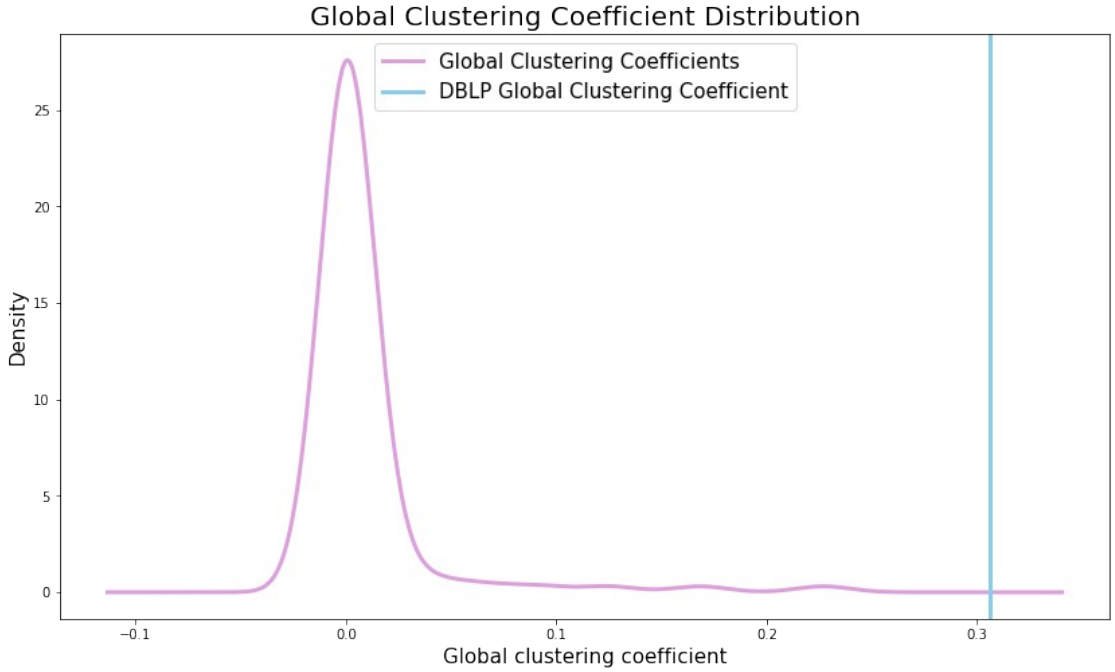
```
Run:  4000
Run:  5000
Run:  6000
Run:  7000
Run:  8000
Run:  9000
Run:  10000
Run:  11000
Run:  12000
Run:  13000
Run:  14000
Run:  15000
Run:  16000
Run:  17000
Run:  18000
Run:  19000
Run:  20000
Run:  21000
Run:  22000
Run:  23000
Run:  24000
Run:  25000
Run:  26000
Run:  27000
Run:  28000
Run:  29000
Run:  30000
Run:  31000
Run:  32000
Run:  33000
Run:  34000
Run:  35000
Run:  36000
Run:  37000
Run:  38000
Run:  39000
Run:  40000
Run:  41000
Run:  42000
Run:  43000
Run:  44000
Run:  45000
Run:  46000
Run:  47000
Run:  48000
Run:  49000
Run:  50000
Run:  51000
Run:  52000
Run:  53000
```

```
Run:  54000
Run:  55000
Run:  56000
Run:  57000
Run:  58000
Run:  59000
Run:  60000
Run:  61000
Run:  62000
Run:  63000
Run:  64000
Run:  65000
Run:  66000
Run:  67000
Run:  68000
Run:  69000
Run:  70000
Run:  71000
Run:  72000
Run:  73000
Run:  74000
Run:  75000
Run:  76000
Run:  77000
Run:  78000
Run:  79000
Run:  80000
Run:  81000
Run:  82000
Run:  83000
Run:  84000
Run:  85000
Run:  86000
Run:  87000
Run:  88000
Run:  89000
Run:  90000
Run:  91000
Run:  92000
Run:  93000
Run:  94000
Run:  95000
Run:  96000
Run:  97000
Run:  98000
Run:  99000
```

```
# Create a copy of the global clustering coefficients list
gcclist = global_clustering_coefficient_list
```

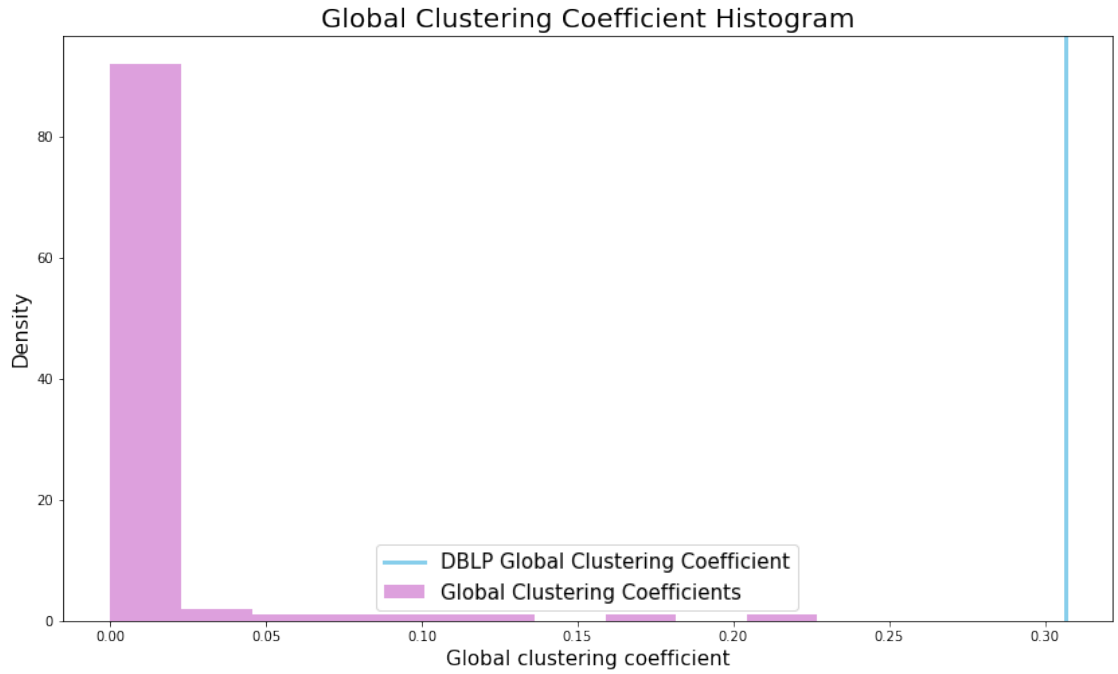
Display the empirical distribution and compare to the clustering coefficient of the real network

```
from matplotlib import pyplot as plt
import pandas as pd

fig = plt.figure(figsize=(14,8))
column_name = ["clustering"]

clustering_coefficients_df = pd.DataFrame(data = gcclist, columns =
column_name)
clustering_coefficients_df.clustering.plot.density(color = 'plum',
linewidth = 3, label = "Global Clustering Coefficients")

# only one line may be specified; full height
plt.axvline(x = dblp_global_clustering_coefficient, color =
'skyblue',linewidth = 3, label = 'DBLP Global Clustering Coefficient')

plt.rc('legend', fontsize=15)    # legend fontsize

plt.title("Global Clustering Coefficient Distribution" ,fontsize = 20)

plt.xlabel("Global clustering coefficient",fontsize = 15)
plt.ylabel("Density",fontsize = 15)

#plt.xlim(xmin = -1, xmax = 1)

plt.legend()
plt.show()
```



I made also an histogram, to see clearly if the DBLP's global clustering coefficient is respect to the different observation in the destribution.

```
from matplotlib import pyplot as plt
import pandas as pd

fig = plt.figure(figsize=(14,8))
column_name = ["clustering"]

#clustering_coefficients_df = pd.DataFrame(data = gcclist, columns =
column_name)
plt.hist(gcclist, color = 'plum', label = "Global Clustering
Coefficients")

# only one line may be specified; full height
plt.axvline(x = dblp_global_clustering_coefficient, color =
'skyblue',linewidth = 3, label = 'DBLP Global Clustering Coefficient')

plt.rc('legend', fontsize=15)    # legend fontsize

plt.title("Global Clustering Coefficient Histogram" ,fontsize = 20)

plt.xlabel("Global clustering coefficient",fontsize = 15)
plt.ylabel("Density",fontsize = 15)

#plt.xlim(xmin = -1, xmax = 1)

plt.legend()
plt.show()
```



My Conclustions:

After computing the global clustering coefficient for each random network and ploting the empirical distribution of the 100 clustering coefficients. I can say that they are smaller in compare to the clustering coefficient of the real network computed in (a.) -> 0.306.

Therfore, in my opinion the configuration model is not a good generative model for the network based on the clustering coefficient, because the global clustering coefficient of the DBLP is far from the rare values that appear in the histogram (The Global clustering coefficient of the DBLP is far from the average global clustering coefficients, more than 3 standard deviation).

Run community detection for a random network and report the number of communities and ARI with the ground truth

```
# I will create this on the last graph that was randomely created
comLabelPropa_rand =
list(nx.algorithms.community.asyn_lpa_communities(Graph, weight=None))
expected_com_rand_num = len(comLabelPropa_rand)
print("The number of communities is", expected_com_rand_num)
```

```
The number of communities is 73246
```

```
# Initial the list with the real communities

# Mount my drive so I can upload the network file (needed if working
in colab) (from excersice 5 solution)
from google.colab import drive
drive.mount('/content/drive/')

# Load the ground truth file
cmty_route = "/content/drive/MyDrive/com-dblp.all.cmty.txt"

# Using readlines()
cmty_route_file = open(cmty_route, 'r')
Lines = cmty_route_file.readlines()

true_cmty = []
for line in Lines:
  community_nodes = list(int(x) for x in line.split())
  true_cmty.append(community_nodes)

cmty_route_file.close()
```

```
Drive already mounted at /content/drive/; to attempt to forcibly
remount, call drive.mount("/content/drive/", force_remount=True).
```

```
expected_cmty_rand = []
for i in range(expected_com_rand_num):
  expected_cmty_rand.append(list(map(int, comLabelPropa_rand[i])))
```

```
# Convert the list to huge list
flat_true_cmty = [x for xs in true_cmty for x in xs]
# Convert the list to huge list
flat_expected_cmty_rand = [x for xs in expected_cmty_rand for x in xs]

# Helpful source: https://stackoverflow.com/questions/3462143/get-
difference-between-two-lists
# Droping duplicates by using set
set_flat_true_cmty = set(flat_true_cmty)
set_flat_expected_cmty_rand = set(flat_expected_cmty_rand)

diff_cmty1 = [item for item in set_flat_true_cmty if item not in
set_flat_expected_cmty_rand]
# There are not items

diff_cmty2 = [item for item in set_flat_expected_cmty_rand if item not
in set_flat_true_cmty]
# There are items

# Create new community in the true communities list
true_cmty.append(diff_cmty2)

# Create dictionary for the true communities
dict_true_cmty = {}

for i in range(len(true_cmty)):
  for j in range(len(true_cmty[i])):
    dict_true_cmty[true_cmty[i][j]] = i + 1

new_dict_true_cmty = dict(sorted(dict_true_cmty.items()))

list_true_cmty = list(new_dict_true_cmty.values())

# Create dictionary for the expected communities
dict_expected_cmty_rand = {}

for i in range(len(expected_cmty_rand)):
  for j in range(len(expected_cmty_rand[i])):
    dict_expected_cmty_rand[expected_cmty_rand[i][j]] = i + 1

new_dict_expected_cmty_rand =
dict(sorted(dict_expected_cmty_rand.items()))

list_expected_cmty_rand = list(new_dict_expected_cmty_rand.values())

# Compute the ARI value
ari = adjusted_rand_score(list_true_cmty, list_expected_cmty_rand)
ari

6.512679821486583e-06
```

Conclustions:

Base on the DBLP graph we got 46417 communities, and an ARI value of: 0.0004109619816075436.

Now, base on a random graph we got 73246 communities, and an ARI value of: 6.512679821486583e-06.

These results are not surprising:

1) Firstly, the file with the real communities contain 13477 communities. the DBLP graph has closer number of communities in compare to the random graph.

2) Secondely, the random graph was created after 100 iterations based on the DBLP graph, therefore its less similar to the DBLP graph and of course it will be less similar to the file with the real communitiese.

Therefore the value of ARI respect to the random graph is smaller than the ARI value respect to the DBLP graph. -> The random graph is less similar than the DBLP graph.

```
# Calculte the number of real communities
len(true_cmty)-1
```

```
13477
```