Question: Cost of Gas

```
The cost of gas is insane! Can you believe node traversal is so expensive? I
wish I had some sort of map or matrix describing the cheapest cost to any node
from any node ...

NodeA -> NodeC 32324
NodeB -> NodeA 26786
NodeC -> NodeB 77458 NodeC -> NodeD 19905 NodeC -> NodeG 19455
NodeD -> NodeA 64678 NodeD -> NodeE 57878
NodeE -> NodeF 29999 NodeE -> NodeA 82356
NodeF -> NodeC 77777 NodeF -> NodeA 33333 NodeF -> NodeD 88888 NodeF -> NodeG
88888
NodeG -> NodeA 1

Example submission: A -> B 1 B -> A 1
Resulting matrix: A B A 0 1 B 1 0

THIS IS WHAT YOUR FLAG SHOULD LOOK LIKE: STOUTCTF{0110}
.No spaces
.One line
.No letters
```
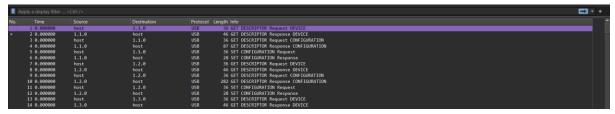
This is from the description

solve.py

```python
import numpy as np

# Define nodes and edges
nodes = ['A', 'B', 'C', 'D', 'E', 'F', 'G']
edges = [
    ("A", "C", 32324),
    ("B", "A", 26786),
    ("C", "B", 77458), ("C", "D", 19905), ("C", "G", 19455),
    ("D", "A", 64678), ("D", "E", 57878),
    ("E", "F", 29999), ("E", "A", 82356),
    ("F", "C", 77777), ("F", "A", 33333), ("F", "D", 88888), ("F", "G", 88888),
    ("G", "A", 1)
]

# Create adjacency matrix
node_count = len(nodes)
INF = float('inf')
adj_matrix = np.full((node_count, node_count), INF)

# Set diagonal to 0 (self-costs)
np.fill_diagonal(adj_matrix, 0)

# Map node names to indices
```
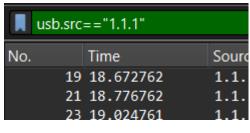
```python
node_index = {node: i for i, node in enumerate(nodes)}

# Fill adjacency matrix with given edges
for src, dst, cost in edges:
    src_idx = node_index[src]
    dst_idx = node_index[dst]
    adj_matrix[src_idx][dst_idx] = cost

# Floyd-Warshall Algorithm for all-pairs shortest paths
for k in range(node_count):
    for i in range(node_count):
        for j in range(node_count):
            adj_matrix[i][j] = min(adj_matrix[i][j], adj_matrix[i][k] + adj_matrix[k][j])

# Replace INF with 0 (for unconnected paths)
shortest_paths_matrix = np.where(adj_matrix == INF, 0, adj_matrix).astype(int)

# Flatten the matrix and format as required
shortest_path_one_line = ''.join(map(lambda x: str(int(x)),
shortest_paths_matrix.flatten()))

# Create the final flag
final_flag = f"STOUTCTF{{{shortest_path_one_line}}}"

# Output the flag
print(final_flag)
```

I create the script run it and got the flag !

STOUTCTF{010978232324522291101071401065177926786059110790151368931668

927856519456774580199057778310778219455646781744609700205787887877116

457633321731149565611556102999911511133333143115656578556214344008511

21109783323255223011010814010 70}

Question: Hackers Keyboard

We were given this file and since it says that his keyboard was hacked in the description so I guess it had something to do with an usb device for keyboards ?? so I filtered it

After we filter it we can see all the HID Data contains the keyboard's buttons.

apply as column and export the data as csv

Solve.py

```
#!/usr/bin/python
# coding: utf-8
from __future__ import print_function
import sys, os
```

```
# declare -A lcasekey
lcasekey = {}
# declare -A ucasekey
ucasekey = {}

# associate USB HID scan codes with keys
# ex: key 4 can be both "a" and "A", depending on if SHIFT is held down
lcasekey[4] = "a"; ucasekey[4] = "A"
lcasekey[5] = "b"; ucasekey[5] = "B"
lcasekey[6] = "c"; ucasekey[6] = "C"
lcasekey[7] = "d"; ucasekey[7] = "D"
lcasekey[8] = "e"; ucasekey[8] = "E"
lcasekey[9] = "f"; ucasekey[9] = "F"
lcasekey[10] = "g"; ucasekey[10] = "G"
lcasekey[11] = "h"; ucasekey[11] = "H"
lcasekey[12] = "i"; ucasekey[12] = "I"
lcasekey[13] = "j"; ucasekey[13] = "J"
lcasekey[14] = "k"; ucasekey[14] = "K"
lcasekey[15] = "l"; ucasekey[15] = "L"
lcasekey[16] = "m"; ucasekey[16] = "M"
lcasekey[17] = "n"; ucasekey[17] = "N"
lcasekey[18] = "o"; ucasekey[18] = "O"
lcasekey[19] = "p"; ucasekey[19] = "P"
lcasekey[20] = "q"; ucasekey[20] = "Q"
lcasekey[21] = "r"; ucasekey[21] = "R"
lcasekey[22] = "s"; ucasekey[22] = "S"
lcasekey[23] = "t"; ucasekey[23] = "T"
lcasekey[24] = "u"; ucasekey[24] = "U"
lcasekey[25] = "v"; ucasekey[25] = "V"
lcasekey[26] = "w"; ucasekey[26] = "W"
lcasekey[27] = "x"; ucasekey[27] = "X"
lcasekey[28] = "y"; ucasekey[28] = "Y"
lcasekey[29] = "z"; ucasekey[29] = "Z"
lcasekey[30] = "1"; ucasekey[30] = "!"
lcasekey[31] = "2"; ucasekey[31] = "@"
lcasekey[32] = "3"; ucasekey[32] = "#"
lcasekey[33] = "4"; ucasekey[33] = "$"
lcasekey[34] = "5"; ucasekey[34] = "%"
lcasekey[35] = "6"; ucasekey[35] = "^"
lcasekey[36] = "7"; ucasekey[36] = "&"
lcasekey[37] = "8"; ucasekey[37] = "*"
lcasekey[38] = "9"; ucasekey[38] = "("
lcasekey[39] = "0"; ucasekey[39] = ")"
lcasekey[40] = "Enter"; ucasekey[40] = "Enter"
lcasekey[41] = "esc"; ucasekey[41] = "esc"
lcasekey[42] = "del"; ucasekey[42] = "del"
```
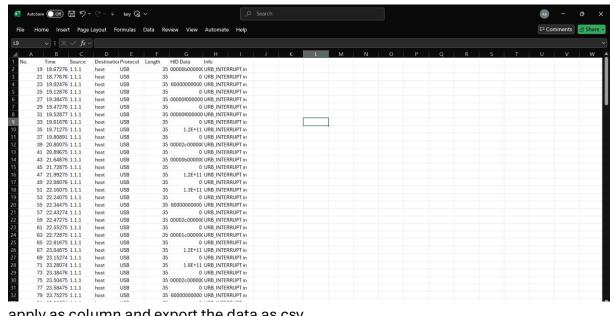
```python
lcasekey[43] = "tab"; ucasekey[43] = "tab"
lcasekey[44] = "space"; ucasekey[44] = "space"
lcasekey[45] = "-"; ucasekey[45] = "_"
lcasekey[46] = "="; ucasekey[46] = "+"
lcasekey[47] = "["; ucasekey[47] = "{"
lcasekey[48] = "]"; ucasekey[48] = "}"
lcasekey[49] = "\\"; ucasekey[49] = "|"
lcasekey[50] = " "; ucasekey[50] = " "
lcasekey[51] = ";"; ucasekey[51] = ":"
lcasekey[52] = "'"; ucasekey[52] = "\""
lcasekey[53] = "`"; ucasekey[53] = "~"
lcasekey[54] = ","; ucasekey[54] = "<"
lcasekey[55] = "."; ucasekey[55] = ">"
lcasekey[56] = "/"; ucasekey[56] = "?"
lcasekey[57] = "CapsLock"; ucasekey[57] = "CapsLock"
lcasekey[79] = "RightArrow"; ucasekey[79] = "RightArrow"
lcasekey[80] = "LeftArrow"; ucasekey[80] = "LeftArrow"
lcasekey[84] = "/"; ucasekey[84] = "/"
lcasekey[85] = "*"; ucasekey[85] = "*"
lcasekey[86] = "-"; ucasekey[86] = "-"
lcasekey[87] = "+"; ucasekey[87] = "+"
lcasekey[88] = "Enter"; ucasekey[88] = "Enter"
lcasekey[89] = "1"; ucasekey[89] = "1"
lcasekey[90] = "2"; ucasekey[90] = "2"
lcasekey[91] = "3"; ucasekey[91] = "3"
lcasekey[92] = "4"; ucasekey[92] = "4"
lcasekey[93] = "5"; ucasekey[93] = "5"
lcasekey[94] = "6"; ucasekey[94] = "6"
lcasekey[95] = "7"; ucasekey[95] = "7"
lcasekey[96] = "8"; ucasekey[96] = "8"
lcasekey[97] = "9"; ucasekey[97] = "9"
lcasekey[98] = "0"; ucasekey[98] = "0"
lcasekey[99] = "."; ucasekey[99] = "."

# modify this line to open the file directly, without command-line arguments
filename = "hid_data.txt"

# open the file and process the key codes
with open(filename, 'r') as keycodes:
    for line in keycodes:
        # Strip out any whitespace characters and check if the line contains valid hex
        line = line.strip()
        if not all(c in '0123456789ABCDEFabcdef' for c in line):
            print(f"Skipping invalid line: {line}")
            continue

        try:
```

```
        # dump line to bytearray
        bytesArray = bytearray.fromhex(line)
        # see if we have a key code
        val = int(bytesArray[2])
        if val > 3 and val < 100:
            # see if left shift or right shift was held down
            if bytesArray[0] == 0x02 or bytesArray[0] == 0x20:
                print(ucasekey[int(bytesArray[2])], end='')  # single line output
            else:
                print(lcasekey[int(bytesArray[2])], end='')  # single line output
    except ValueError:
        print(f"Error processing line: {line}")
```

```
  solve.py    ×

  solve.py > ...
 95              print(f"Skipping invalid line: {line}")
 96              continue
 97
 98         try:
 99             # dump line to bytearray
 100            bytesArray = bytearray.fromhex(line)
 101            # see if we have a key code
 102            val = int(bytesArray[2])
 103            if val > 3 and val < 100:
 104                # see if left shift or right shift was held down
 105                if bytesArray[0] == 0x02 or bytesArray[0] == 0x20:
 106                    print(ucasekey[int(bytesArray[2])], end='')  # single line output
 107                else:
 108                    print(lcasekey[int(bytesArray[2])], end='')  # single line output
 109        except ValueError:
 110            print(f"Error processing line: {line}")
 111
```

```
PROBLEMS   OUTPUT   TERMINAL   PORTS   DEBUG CONSOLE

PS C:\Users\akmal\OneDrive\Desktop\ctf> python solve.py
Skipping invalid line: HID Data
hellospacehopespaceyouspacecanspacegetspacethisspaceflag:)spacestoutctf{by3yfbitp3vs1ecx6ery}goodspaceluck
PS C:\Users\akmal\OneDrive\Desktop\ctf> 
```

I used this script and here I got the flag

stoutctf{by3yfbitp3vs1ecx6ery}

Question: Who Said 30 Times

```
00000000  56 6d 30 77 64 32 51 79 55 58 6c 56 57 47 78 57  |Vm0wd2QyUXlVWGxW|
00000010  56 30 64 34 56 31 59 77 5a 44 52 57 4d 56 6c 33  |V0d4V1YwZDRWMVl3|
00000020  57 6b 52 53 56 30 31 57 62 44 4e 58 61 31 4a 54  |WkRSV01WbDNXa1JT|
00000030  56 6a 41 78 56 32 4a 45 54 6c 68 68 4d 55 70 55  |VjAxV2JETlhhMUpU|
00000040  56 6d 70 42 65 46 59 79 53 6b 56 55 62 47 68 6f  |VmpBeFYySkVUbGho|
00000050  54 56 56 77 56 56 5a 74 63 45 4a 6c 52 6c 6c 35  |TVVwVVZtcEJlRll5|
00000060  55 32 74 57 56 57 4a 48 61 47 39 55 56 6c 5a 33  |U2tWVWJHaG9UVlZ3|
00000070  56 6c 5a 61 64 47 4e 46 53 6d 78 53 62 47 77 31  |VlZadGNFSmxSbGw1|
00000080  56 54 4a 30 56 31 5a 58 53 6b 68 68 52 7a 6c 56  |VTJ0V1ZXSkhhRzlV|
00000090  56 6d 78 61 4d 31 5a 73 57 6d 46 6b 52 30 35 47  |VmxaM1ZsWmFkR05G|
000000a0  55 32 31 34 55 32 4a 48 64 7a 46 57 56 45 6f 77  |U214U2JHdzFWVEow|
000000b0  56 6a 61 46 61 57 46 4e 72 61 47 68 53 65 6d 78 57  |VjFaWFNraGhSemxW|
000000c0  56 6d 31 34 59 55 30 78 57 6e 4e 58 62 55 5a 72  |Vm14YU0xWnNXbUZr|
000000d0  55 6a 41 31 52 31 55 79 4d 54 52 56 4d 6b 70 49  |UjA1R1UyMTRVMkpI|
000000e0  5a 48 70 47 56 31 5a 46 62 33 64 57 61 6b 5a 68  |ZHpGV1ZFb3dWakZh|
000000f0  56 30 5a 4f 63 6d 46 48 61 46 4e 6c 62 58 68 58  |V0ZOcmFHaFNlbXhX|
00000100  56 6d 30 78 4e 46 6c 56 4d 48 68 58 62 6b 35 59  |Vm0xNFlVMHhXbk5Y|
00000110  59 6c 56 61 63 6c 56 71 51 54 46 53 4d 56 56 35  |YlVaclVqQTFSMVV5|
00000120  54 56 52 53 56 6b 31 72 63 45 6c 61 53 48 42 48  |TVRSVk1rcElaSHBH|
00000130  56 6a 46 61 52 6d 49 7a 5a 46 64 68 61 31 70 6f  |VjFaRmIzZFdha1po|
00000140  56 6a 42 61 54 32 4e 74 52 6b 68 68 52 6b 35 73  |VjBaT2NtRkhhRk5s|
00000150  59 6c 68 6f 57 46 5a 74 4d 48 68 4f 52 6d 78 57  |YlhoWFZtMHhORmxW|
00000160  54 55 68 6f 57 47 4a 72 4e 56 6c 5a 62 46 5a 68  |TUhoWGJrNVlZbFZh|
00000170  59 32 78 57 63 56 46 55 52 6c 4e 4e 56 6c 59 31  |Y2xWcVFURlNNVlY1|
00000180  56 46 5a 53 55 31 5a 72 4d 58 4a 6a 52 57 78 68  |VFZSU1ZrMXJjRWxh|
00000190  55 30 68 43 53 46 5a 71 52 6d 46 53 62 55 6c 36  |U0hCSFZqRmFSbUl6|
000001a0  57 6b 5a 6b 61 47 45 78 63 47 39 57 61 6b 4a 68  |WkZkaGExcG9WakJh|
000001b0  56 44 4a 4a 4f 64 46 4a 72 61 47 68 53 61 7a 56 7a  |VDJOdFJraGhSazVz|
000001c0  57 57 78 6f 62 31 64 47 57 6e 52 4e 53 47 68 50  |WWxob1dGWnRNSGhP|
000001d0  55 6d 31 34 56 31 52 56 61 47 39 58 52 30 70 79  |Um14V1RVaG9XR0py|
000001e0  54 6c 5a 73 57 6d 4a 47 57 6d 68 5a 4d 6e 68 58  |TlZsWmJGWmhZMnhX|
000001f0  59 31 5a 47 56 56 4a 73 54 6b 35 57 62 46 6b 78  |Y1ZGVVJsTk5WbFkx|
00000200  56 6b 5a 61 55 31 55 78 57 6e 4a 4e 57 45 70 71  |VkZaU1UxWnJNWEpq|
00000210  55 6c 64 34 61 46 55 77 61 45 4e 54 52 6c 70 78  |Uld4aFUwaENTRlpx|
00000220  55 6d 31 47 55 32 4a 56 62 44 5a 58 61 31 70 72  |Um1GU2JVbDZXa1pr|
00000230  59 55 64 46 65 47 4e 48 4f 56 64 68 61 30 70 6f  |YUdFeGNHOVdha0po|
00000240  56 6b 52 4b 54 32 52 47 53 6e 4a 68 52 32 68 54  |VkRKT2RGSnJhR2hT|
00000250  59 58 70 57 65 6c 64 58 65 47 39 69 4d 57 52 48  |YXpWeldXeG9iMWRH|
00000260  56 32 35 53 54 6c 4e 48 61 46 42 56 62 54 45 30  |V25STlNHaFBVbTE0|
```

The file given so I open it with notepad and extract the hex.

```python
def extract_strings_from_file(filename):
        content = file.read()

        # Define the regular expression pattern to match text between '| |'
        pattern = r'\|([^\|]+)\|'

        # Find all matches of the pattern
        matched_strings = re.findall(pattern, content)

        if matched_strings:
            # Print all the extracted strings on a single line
            print(" ".join(matched_strings))
        else:
            print("No strings found within | |")

    except Exception as e:
        print(f"An error occurred: {e}")

# Usage: specify the filename where you want to filter the strings
filename = 'Who_Said_30.txt'  # Replace this with your actual file path
extract_strings_from_file(filename)
```

z YUZkTmJtaHlXVmN4 UzFOV1JuSmhSbVJY Wld0YU5sZFdWbFpr TWxGNFZHNVNVMkpI VWs5V2FrWkhUVEZa ZUZKdFJsVmlWbHBZ VmxjMVMxbFdTblJo Um14V1lXdGFURll5 ZUd0V01WWnlUMVph VGxadGR6QldiVEF4 VlRKS1 IxUnJhRlpo ZW14WFdXeFNSbVF4 VWxWU2JVWnJVbFJH VmxWWGVHdFViRnBa VVd4a1YxWjZSWGRY VmxwYVpWwk9jMVpz WkdoaGVsWlpWMVpr TUZJd05VZGhNMmhZ WkxWYWNWUldXbUZs YkdSeVYyczVWV0Y2 UmpCWlZWcExWak pL UjFOclVScFdSVnBM V2xWYVQxZFhSa2Rh UjJ4b1pXeGFXbFl4 WkRCaE1WVjRWWmhv Vm1KSFVsaFpiVEZU WTJ4c2NsZHVaR3hp UjFKNVdWVmtNRlpy TVVwU2JHaFhZbGhD VkZacVFYaFhSMFpK Vj34b1YySklRbEZX VjNCS FZESlNWMVJ1 VGxoaVYyaHZWRlJD U21WV1dYaFZhMDVh Vm14c05WkdhSE5W TWtWNFkwWldXbUpI YUZSV1JFWmhZMVpL Y21SRk9XbFNia0Yz VmtaV2IySXHwWGRO V0VwWVlUSlNWbFp0 ZUhkamJHeFlaVVYw YTFJd1draFpWV nBY VmpKV2MxZHJWbGhX YkZwb1YxWmtUMVl4 VW5WVGJYUlRZWHBX VUZadGVGTldiVkY0 Vj34b1RsWkdTbTlV VjNSaFYwWlZlVTVX WkZaaVIxSktWWmQ0 YzFkdFJYbFZiRTVW Vmxad1lWcFdWWGhX TVU1MFpVWk9UbEpH V2ts V2Frb3dZVEZK ZUZaWWJGTmhiSEJZ V1cweFUxZEdiSE5W YTFwclRWaENSbFZ0 Y3pWV01ERlpVV3Rv V0dFeGNISlpWbHBo VmpKT1IxcEdaRk5p U0VKdlZsaHdTMVl5 VFhoYVJteHBVbFJz V0Zsc1drdFhSbVJZ WlVaa1YwMlVWw bnBX TW5odllrWktObUpH YkZwaVJscFlwR3hh wVZkSFzrWlViR1JY VFVaWk1GZFVRbUZW TVdSeVRWaFdhRk5G Y0dGYVYzUjNWVVpW ZDFwR1pGTk5Wa293 V1d0YWEyRkZNSGRU YmxwWFlsaENTRmRX V2xwbFIwcEhXa1pr V0Z JemFGVlhWM1Jy VlRGYVYySklVbXBs YTFweFZtMTRZVTFH WWxkVmJFNVlVbXh3 Zwxrd1VrTlpWbHBY VjFoa1YyRXhjRXhh UlZwWFl6SkdSMXBG TldoaE1WbDZWbTB4 ZDFGdFZraFVibEpW WVRGd2NWVnRlSGRV TVd4MFRwYzV UbUpI WWxoV01qRXdWa1V4 VjJKRVVsZFNlbFpZ VmpCa1MxTkdWbk5o UjBaWFRURktXRmRy VWtkaE1rNXpWbTVL YTFKdFVrOVdiR1F6 VFVaYWNWTlVSbGRO Vm5Bd1ZUSjRjMVp0 U2tkWGJHUmFWak5T YUZaRlduZFNWa3Aw VW 14b1UxWkZXbGRX Vm1ONFRVWlNkRk5y WkdwU2JYaFlXV3Rr VG1WR1drVlRhMlJU VFZaS2VwbFZXbUZo VmtsNVlVWmtWMkpZ UWxCV1ZFRjNaVVpr ZFZWdGVGTmlWa3A1 VmxkNFUxWXhaSE5h U0VwWVlsaFNWbGxy V25kTlZsWl laRWM1 V0ZKc2NFZFZNbmh2 VjJ4YVJsZHVjRmRo YTFweVdYcEtSMUl4 WkhSaVJrNVhWMFZL V2xacVJtRmhNREZJ VlZob1ZXRXlVbGha YlhSaFZrWnNkRTFX VGxkV2JGcDRWWMx YTFaR1NuUmxSbVJY VFc1Tk1WbFZXa3RX V mtaeFZHeGFUbUp0 YURKV2JYQkxVekZh VjFOc1ZsSmlSMmhZ V1d4YVMxTkdaSE5Y YlVaYVZqRktXRlpI TlV0VU1WcDBWVzVD VjJKWVVqTldNbmhh WlVacmVtRkhlR2hs YTFwWlZtMTRiMkl5 UmxkVWExcHFVbTE0 WVZscldtR mhSbHAw WlVaT1YwMVhVakJW YlRFMFZqSktTVkZz Y0ZoaE1WcFlWRlZh WVdNeFRuVlViR2hw VW01Q1YxWnRlR0ZU TwxaWFZtdGtVMVpI ZUZKV1ZsRjNUMUU5 UFE9PQ==
PS C:\Users\akmal\OneDrive\Desktop\ctf>
```

Code used to extract the base64

```python
# Filter content between the `||` markers
def filter_content_between_markers(file_path):
    with open(file_path, "r") as file:
        content = file.readlines()

    # Extracting only the content within the || markers
    filtered_content = []
    for line in content:
        start = line.find('|')
        end = line.find('|', start + 1)
        if start != -1 and end != -1:
            filtered_content.append(line[start + 1:end])

    # Save the filtered content to a new file
    output_path = "filtered_content.txt"
    with open(output_path, "w") as output_file:
        output_file.write("\n".join(filtered_content))

    return output_path
```

Now decode the base64 30 times

```python
import base64

# Decode base64 content multiple times
def decode_base64_multiple_times(encoded_str, times=30):
    decoded = encoded_str
    for _ in range(times):
        decoded = base64.b64decode(decoded).decode('utf-8', errors='ignore')
    return decoded

# Usage example: Assuming the content is read from a file
def decode_file_content(file_path, times=30):
    # Read the file content
    with open(file_path, 'r') as file:
        encoded_content = file.read()

    # Decode the content multiple times
    decoded_content = decode_base64_multiple_times(encoded_content, times)

    # Save the decoded content to a new file
    decoded_file_path = "decoded_content.txt"
    with open(decoded_file_path, 'w') as file:
        file.write(decoded_content)

    return decoded_file_path
```

Flag : STOUTCTF{djfRQP4yBWjbcnEEixBHvUvta8iZd5Fm}