#QML-HEP GSoC 2023 Task Solutions

##Installing Required Package

```
!pip install cirq
!pip install qiskit
!pip install pylatexenc
!pip install pennylane
!pip install -U tensorflow-addons
# # !pip install -q tensorflow==2.3.1
# !pip install -q tensorflow-quantum
```

```
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Collecting cirq
  Downloading cirq-1.1.0-py3-none-any.whl (7.7 kB)
Collecting cirq-aqt==1.1.0
  Downloading cirq_aqt-1.1.0-py3-none-any.whl (27 kB)
Collecting cirq-rigetti==1.1.0
  Downloading cirq_rigetti-1.1.0-py3-none-any.whl (66 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 66.4/66.4 KB 3.9 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 577.4/577.4 KB 17.7 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 57.6/57.6 KB 7.0 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 594.6/594.6 KB 48.4 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 66.9 MB/s eta
0:00:00
ent already satisfied: requests~=2.18 in
/usr/local/lib/python3.9/dist-packages (from cirq-aqt==1.1.0->cirq)
(2.27.1)
Requirement already satisfied: numpy<1.24,>=1.16 in
/usr/local/lib/python3.9/dist-packages (from cirq-core==1.1.0->cirq)
(1.22.4)
Requirement already satisfied: sortedcontainers~=2.0 in
/usr/local/lib/python3.9/dist-packages (from cirq-core==1.1.0->cirq)
(2.4.0)
Requirement already satisfied: matplotlib~=3.0 in
/usr/local/lib/python3.9/dist-packages (from cirq-core==1.1.0->cirq)
(3.7.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-
packages (from cirq-core==1.1.0->cirq) (4.65.0)
Collecting duet~=0.2.7
  Downloading duet-0.2.7-py3-none-any.whl (28 kB)
Requirement already satisfied: pandas in
/usr/local/lib/python3.9/dist-packages (from cirq-core==1.1.0->cirq)
(1.4.4)
Requirement already satisfied: sympy in /usr/local/lib/python3.9/dist-
```

```
packages (from cirq-core==1.1.0->cirq) (1.11.1)
Collecting networkx~=2.4
  Downloading networkx-2.8.8-py3-none-any.whl (2.0 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 66.8 MB/s eta
0:00:00
ent already satisfied: typing-extensions in
/usr/local/lib/python3.9/dist-packages (from cirq-core==1.1.0->cirq)
(4.5.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-
packages (from cirq-core==1.1.0->cirq) (1.10.1)
Requirement already satisfied: protobuf<4,>=3.15.0 in
/usr/local/lib/python3.9/dist-packages (from cirq-google==1.1.0->cirq)
(3.20.3)
Collecting google-api-core[grpc]<2.0.0dev,>=1.14.0
  Downloading google_api_core-1.34.0-py3-none-any.whl (120 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 120.2/120.2 KB 16.8 MB/s eta
0:00:00
ent already satisfied: proto-plus>=1.20.0 in
/usr/local/lib/python3.9/dist-packages (from cirq-google==1.1.0->cirq)
(1.22.2)
Collecting pyquil>=3.2.0
  Downloading pyquil-3.3.4-py3-none-any.whl (221 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 221.6/221.6 KB 28.3 MB/s eta
0:00:00
ent already satisfied: googleapis-common-protos<2.0dev,>=1.56.2 in
/usr/local/lib/python3.9/dist-packages (from google-api-
core[grpc]<2.0.0dev,>=1.14.0->cirq-google==1.1.0->cirq) (1.59.0)
Requirement already satisfied: google-auth<3.0dev,>=1.25.0 in
/usr/local/lib/python3.9/dist-packages (from google-api-
core[grpc]<2.0.0dev,>=1.14.0->cirq-google==1.1.0->cirq) (2.17.0)
Requirement already satisfied: grpcio-status<2.0dev,>=1.33.2 in
/usr/local/lib/python3.9/dist-packages (from google-api-
core[grpc]<2.0.0dev,>=1.14.0->cirq-google==1.1.0->cirq) (1.48.2)
Requirement already satisfied: grpcio<2.0dev,>=1.33.2 in
/usr/local/lib/python3.9/dist-packages (from google-api-
core[grpc]<2.0.0dev,>=1.14.0->cirq-google==1.1.0->cirq) (1.53.0)
Requirement already satisfied: pillow>=6.2.0 in
/usr/local/lib/python3.9/dist-packages (from matplotlib~=3.0->cirq-
core==1.1.0->cirq) (8.4.0)
Requirement already satisfied: importlib-resources>=3.2.0 in
/usr/local/lib/python3.9/dist-packages (from matplotlib~=3.0->cirq-
core==1.1.0->cirq) (5.12.0)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.9/dist-packages (from matplotlib~=3.0->cirq-
core==1.1.0->cirq) (2.8.2)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.9/dist-packages (from matplotlib~=3.0->cirq-
core==1.1.0->cirq) (23.0)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.9/dist-packages (from matplotlib~=3.0->cirq-
```

```
core==1.1.0->cirq) (1.0.7)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.9/dist-packages (from matplotlib~=3.0->cirq-
core==1.1.0->cirq) (3.0.9)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.9/dist-packages (from matplotlib~=3.0->cirq-
core==1.1.0->cirq) (4.39.3)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.9/dist-packages (from matplotlib~=3.0->cirq-
core==1.1.0->cirq) (1.4.4)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.9/dist-packages (from matplotlib~=3.0->cirq-
core==1.1.0->cirq) (0.11.0)
Collecting lark<0.12.0,>=0.11.1
  Downloading lark-0.11.3.tar.gz (229 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 229.9/229.9 KB 26.0 MB/s eta
0:00:00
etadata (setup.py) ... ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
147.4/147.4 KB 13.3 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45.6/45.6 KB 4.8 MB/s eta
0:00:00
etadata (setup.py) ... ent already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.9/dist-packages (from requests~=2.18->cirq-
aqt==1.1.0->cirq) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/usr/local/lib/python3.9/dist-packages (from requests~=2.18->cirq-
aqt==1.1.0->cirq) (1.26.15)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.9/dist-packages (from requests~=2.18->cirq-
aqt==1.1.0->cirq) (2.0.12)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.9/dist-packages (from requests~=2.18->cirq-
aqt==1.1.0->cirq) (2022.12.7)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.9/dist-packages (from pandas->cirq-core==1.1.0-
>cirq) (2022.7.1)
Requirement already satisfied: mpmath>=0.19 in
/usr/local/lib/python3.9/dist-packages (from sympy->cirq-core==1.1.0-
>cirq) (1.3.0)
Requirement already satisfied: six>=1.9.0 in
/usr/local/lib/python3.9/dist-packages (from google-
auth<3.0dev,>=1.25.0->google-api-core[grpc]<2.0.0dev,>=1.14.0->cirq-
google==1.1.0->cirq) (1.16.0)
Requirement already satisfied: rsa<5,>=3.1.4 in
/usr/local/lib/python3.9/dist-packages (from google-
auth<3.0dev,>=1.25.0->google-api-core[grpc]<2.0.0dev,>=1.14.0->cirq-
google==1.1.0->cirq) (4.9)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/usr/local/lib/python3.9/dist-packages (from google-
auth<3.0dev,>=1.25.0->google-api-core[grpc]<2.0.0dev,>=1.14.0->cirq-
```

```
google==1.1.0->cirq) (5.3.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.9/dist-packages (from google-
auth<3.0dev,>=1.25.0->google-api-core[grpc]<2.0.0dev,>=1.14.0->cirq-
google==1.1.0->cirq) (0.2.8)
Requirement already satisfied: zipp>=3.1.0 in
/usr/local/lib/python3.9/dist-packages (from importlib-
resources>=3.2.0->matplotlib~=3.0->cirq-core==1.1.0->cirq) (3.15.0)
Requirement already satisfied: toml<0.11.0,>=0.10.2 in
/usr/local/lib/python3.9/dist-packages (from qcs-api-
client<0.22.0,>=0.21.0->pyquil>=3.2.0->cirq-rigetti==1.1.0->cirq)
(0.10.2)
Collecting PyJWT<3.0.0,>=2.4.0
  Downloading PyJWT-2.6.0-py3-none-any.whl (20 kB)
Collecting attrs<22.0.0,>=21.3.0
  Downloading attrs-21.4.0-py2.py3-none-any.whl (60 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 60.6/60.6 KB 8.2 MB/s eta
0:00:00
ent already satisfied: pydantic<2.0.0,>=1.7.2 in
/usr/local/lib/python3.9/dist-packages (from qcs-api-
client<0.22.0,>=0.21.0->pyquil>=3.2.0->cirq-rigetti==1.1.0->cirq)
(1.10.7)
Collecting rfc3339<7.0,>=6.2
  Downloading rfc3339-6.2-py3-none-any.whl (5.5 kB)
Collecting httpx<0.24.0,>=0.23.0
  Downloading httpx-0.23.3-py3-none-any.whl (71 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 71.5/71.5 KB 10.7 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.7/98.7 KB 11.0 MB/s eta
0:00:00
ent already satisfied: decorator>=3.4.2 in
/usr/local/lib/python3.9/dist-packages (from retry<0.10.0,>=0.9.2-
>pyquil>=3.2.0->cirq-rigetti==1.1.0->cirq) (4.4.2)
Requirement already satisfied: msgpack<2.0,>=0.6 in
/usr/local/lib/python3.9/dist-packages (from rpcq<4.0.0,>=3.10.0-
>pyquil>=3.2.0->cirq-rigetti==1.1.0->cirq) (1.0.5)
Collecting python-rapidjson
  Downloading python_rapidjson-1.10-cp39-cp39-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.7 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 68.0 MB/s eta
0:00:00
ent already satisfied: pyzmq>=17 in /usr/local/lib/python3.9/dist-
packages (from rpcq<4.0.0,>=3.10.0->pyquil>=3.2.0->cirq-
rigetti==1.1.0->cirq) (23.2.1)
Collecting ruamel.yaml
  Downloading ruamel.yaml-0.17.21-py3-none-any.whl (109 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 109.5/109.5 KB 11.3 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 69.6/69.6 KB 8.8 MB/s eta
0:00:00
```

```
ent already satisfied: pyasn1<0.5.0,>=0.4.6 in
/usr/local/lib/python3.9/dist-packages (from pyasn1-modules>=0.2.1-
>google-auth<3.0dev,>=1.25.0->google-api-core[grpc]<2.0.0dev,>=1.14.0-
>cirq-google==1.1.0->cirq) (0.4.8)
Collecting ruamel.yaml.clib>=0.2.6
  Downloading ruamel.yaml.clib-0.2.7-cp39-cp39-
manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_24_x86_64.whl
(519 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 519.4/519.4 KB 38.8 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 KB 5.4 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 80.6/80.6 KB 7.3 MB/s eta
0:00:00
e=lark-0.11.3-py2.py3-none-any.whl size=99646
sha256=3fbdf5a3c50fb279ede331680bb10c13be520ad53fc8a3f5d4fa94c64566a4c
3
  Stored in directory:
/root/.cache/pip/wheels/ec/6a/24/f8eeaf52fee56bfe54309621b59c41bb7f1df
56f4bfbcdb0ce
  Building wheel for rpcq (setup.py) ... e=rpcq-3.11.0-py3-none-
any.whl size=45985
sha256=809c584a2b129cd3ca635d99b95ecbbfe80158d7c9f044bfb38492ef836c513
4
  Stored in directory:
/root/.cache/pip/wheels/a6/c4/42/34581dfe489802146924ad802b13aa7fe3820
f9e8c15f67afc
Successfully built lark rpcq
Installing collected packages: types-retry, types-python-dateutil,
rfc3986, rfc3339, lark, sniffio, ruamel.yaml.clib, retrying, python-
rapidjson, PyJWT, py, networkx, iso8601, h11, duet, attrs,
ruamel.yaml, retry, anyio, rpcq, httpcore, google-api-core, cirq-core,
httpx, cirq-web, cirq-pasqal, cirq-ionq, cirq-aqt, qcs-api-client,
cirq-google, pyquil, cirq-rigetti, cirq
  Attempting uninstall: networkx
    Found existing installation: networkx 3.0
    Uninstalling networkx-3.0:
      Successfully uninstalled networkx-3.0
  Attempting uninstall: attrs
    Found existing installation: attrs 22.2.0
    Uninstalling attrs-22.2.0:
      Successfully uninstalled attrs-22.2.0
  Attempting uninstall: google-api-core
    Found existing installation: google-api-core 2.11.0
    Uninstalling google-api-core-2.11.0:
      Successfully uninstalled google-api-core-2.11.0
Successfully installed PyJWT-2.6.0 anyio-3.6.2 attrs-21.4.0 cirq-1.1.0
cirq-aqt-1.1.0 cirq-core-1.1.0 cirq-google-1.1.0 cirq-ionq-1.1.0 cirq-
pasqal-1.1.0 cirq-rigetti-1.1.0 cirq-web-1.1.0 duet-0.2.7 google-api-
core-1.34.0 h11-0.14.0 httpcore-0.16.3 httpx-0.23.3 iso8601-1.1.0
```

```
lark-0.11.3 networkx-2.8.8 py-1.11.0 pyquil-3.3.4 python-rapidjson-
1.10 qcs-api-client-0.21.3 retry-0.9.2 retrying-1.3.4 rfc3339-6.2
rfc3986-1.5.0 rpcq-3.11.0 ruamel.yaml-0.17.21 ruamel.yaml.clib-0.2.7
sniffio-1.3.0 types-python-dateutil-2.8.19.11 types-retry-0.9.9.3
```

{"pip_warning":{"packages":["google"]}}

```
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Collecting qiskit
  Downloading qiskit-0.42.1.tar.gz (14 kB)
  Preparing metadata (setup.py) ...
anylinux_2_17_x86_64.manylinux2014_x86_64.whl (5.1 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.1/5.1 MB 33.8 MB/s eta
0:00:00
anylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.8 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 46.0 MB/s eta
0:00:00
q-provider==0.20.2
  Downloading qiskit_ibmq_provider-0.20.2-py3-none-any.whl (241 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 241.5/241.5 KB 23.2 MB/s eta
0:00:00
ent already satisfied: scipy>=1.0 in /usr/local/lib/python3.9/dist-
packages (from qiskit-aer==0.12.0->qiskit) (1.10.1)
Requirement already satisfied: numpy>=1.16.3 in
/usr/local/lib/python3.9/dist-packages (from qiskit-aer==0.12.0-
>qiskit) (1.22.4)
Requirement already satisfied: python-dateutil>=2.8.0 in
/usr/local/lib/python3.9/dist-packages (from qiskit-ibmq-
provider==0.20.2->qiskit) (2.8.2)
Requirement already satisfied: requests>=2.19 in
/usr/local/lib/python3.9/dist-packages (from qiskit-ibmq-
provider==0.20.2->qiskit) (2.27.1)
Collecting websockets>=10.0
  Downloading websockets-11.0-cp39-cp39-
manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux
2014_x86_64.whl (129 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 129.5/129.5 KB 6.6 MB/s eta
0:00:00
ent already satisfied: urllib3>=1.21.1 in
/usr/local/lib/python3.9/dist-packages (from qiskit-ibmq-
provider==0.20.2->qiskit) (1.26.15)
Collecting websocket-client>=1.5.1
  Downloading websocket_client-1.5.1-py3-none-any.whl (55 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 55.9/55.9 KB 6.4 MB/s eta
0:00:00
<=1.1.0
  Downloading requests_ntlm-1.1.0-py2.py3-none-any.whl (5.7 kB)
Collecting symengine>=0.9
  Downloading symengine-0.10.0-cp39-cp39-
manylinux_2_12_x86_64.manylinux2010_x86_64.whl (37.5 MB)
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 37.5/37.5 MB 12.0 MB/s eta
0:00:00
anylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.9 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.9/1.9 MB 13.7 MB/s eta
0:00:00
ent already satisfied: sympy>=1.3 in /usr/local/lib/python3.9/dist-
packages (from qiskit-terra==0.23.3->qiskit) (1.11.1)
Collecting dill>=0.3
  Downloading dill-0.3.6-py3-none-any.whl (110 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 110.5/110.5 KB 7.7 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.6/49.6 KB 5.4 MB/s eta
0:00:00
ent already satisfied: psutil>=5 in /usr/local/lib/python3.9/dist-
packages (from qiskit-terra==0.23.3->qiskit) (5.9.4)
Collecting ply>=3.10
  Downloading ply-3.11-py2.py3-none-any.whl (49 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.6/49.6 KB 2.1 MB/s eta
0:00:00
ent already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-
packages (from python-dateutil>=2.8.0->qiskit-ibmq-provider==0.20.2-
>qiskit) (1.16.0)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.9/dist-packages (from requests>=2.19->qiskit-
ibmq-provider==0.20.2->qiskit) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.9/dist-packages (from requests>=2.19->qiskit-
ibmq-provider==0.20.2->qiskit) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.9/dist-packages (from requests>=2.19->qiskit-
ibmq-provider==0.20.2->qiskit) (2022.12.7)
Requirement already satisfied: cryptography>=1.3 in
/usr/local/lib/python3.9/dist-packages (from requests-ntlm<=1.1.0-
>qiskit-ibmq-provider==0.20.2->qiskit) (40.0.1)
Collecting ntlm-auth>=1.0.2
  Downloading ntlm_auth-1.5.0-py2.py3-none-any.whl (29 kB)
Collecting pbr!=2.1.0,>=2.0.0
  Downloading pbr-5.11.1-py2.py3-none-any.whl (112 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 112.7/112.7 KB 14.1 MB/s eta
0:00:00
ent already satisfied: mpmath>=0.19 in /usr/local/lib/python3.9/dist-
packages (from sympy>=1.3->qiskit-terra==0.23.3->qiskit) (1.3.0)
Requirement already satisfied: cffi>=1.12 in
/usr/local/lib/python3.9/dist-packages (from cryptography>=1.3-
>requests-ntlm<=1.1.0->qiskit-ibmq-provider==0.20.2->qiskit) (1.15.1)
Requirement already satisfied: pycparser in
/usr/local/lib/python3.9/dist-packages (from cffi>=1.12-
>cryptography>=1.3->requests-ntlm<=1.1.0->qiskit-ibmq-
provider==0.20.2->qiskit) (2.21)
Building wheels for collected packages: qiskit
```

```
  Building wheel for qiskit (setup.py) ... e=qiskit-0.42.1-py3-none-
any.whl size=12938
sha256=727022bb6132caec7f0462c7106119f07cf8d8628e4794f8476eae0d7ada4b2
0
  Stored in directory:
/root/.cache/pip/wheels/40/64/74/29c046bda04fd60f3f6b2e244fa85b70f219e
363fc3373f541
Successfully built qiskit
Installing collected packages: ply, websockets, websocket-client,
symengine, rustworkx, pbr, ntlm-auth, dill, stevedore, requests-ntlm,
qiskit-terra, qiskit-ibmq-provider, qiskit-aer, qiskit
Successfully installed dill-0.3.6 ntlm-auth-1.5.0 pbr-5.11.1 ply-3.11
qiskit-0.42.1 qiskit-aer-0.12.0 qiskit-ibmq-provider-0.20.2 qiskit-
terra-0.23.3 requests-ntlm-1.1.0 rustworkx-0.12.1 stevedore-5.0.0
symengine-0.10.0 websocket-client-1.5.1 websockets-11.0
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Collecting pylatexenc
  Downloading pylatexenc-2.10.tar.gz (162 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 162.6/162.6 KB 5.0 MB/s eta
0:00:00
etadata (setup.py) ... e=pylatexenc-2.10-py3-none-any.whl size=136831
sha256=00a6990882fd4dfb2054e571f5bf5d8203f74229e3a0b4068c07f6a2893a8a7
c
  Stored in directory:
/root/.cache/pip/wheels/a3/68/66/2f15abd0673d83c02f354115feedeb89c3dae
d2ac319b11090
Successfully built pylatexenc
Installing collected packages: pylatexenc
Successfully installed pylatexenc-2.10
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Collecting pennylane
  Downloading PennyLane-0.29.1-py3-none-any.whl (1.3 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 17.2 MB/s eta
0:00:00
ent already satisfied: networkx in /usr/local/lib/python3.9/dist-
packages (from pennylane) (2.8.8)
Collecting semantic-version>=2.7
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
Requirement already satisfied: appdirs in
/usr/local/lib/python3.9/dist-packages (from pennylane) (1.4.4)
Requirement already satisfied: toml in /usr/local/lib/python3.9/dist-
packages (from pennylane) (0.10.2)
Collecting autoray>=0.3.1
  Downloading autoray-0.6.3-py3-none-any.whl (48 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 48.3/48.3 KB 6.3 MB/s eta
0:00:00
anylinux_2_17_x86_64.manylinux2014_x86_64.whl (16.5 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.5/16.5 MB 60.2 MB/s eta
```

```
0:00:00
ent already satisfied: autograd in /usr/local/lib/python3.9/dist-
packages (from pennylane) (1.5)
Requirement already satisfied: cachetools in
/usr/local/lib/python3.9/dist-packages (from pennylane) (5.3.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-
packages (from pennylane) (1.10.1)
Requirement already satisfied: numpy<1.24 in
/usr/local/lib/python3.9/dist-packages (from pennylane) (1.22.4)
Requirement already satisfied: requests in
/usr/local/lib/python3.9/dist-packages (from pennylane) (2.27.1)
Requirement already satisfied: future>=0.15.2 in
/usr/local/lib/python3.9/dist-packages (from autograd->pennylane)
(0.18.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.9/dist-packages (from requests->pennylane)
(2.0.12)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/usr/local/lib/python3.9/dist-packages (from requests->pennylane)
(1.26.15)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.9/dist-packages (from requests->pennylane)
(3.4)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.9/dist-packages (from requests->pennylane)
(2022.12.7)
Requirement already satisfied: rustworkx==0.12.1 in
/usr/local/lib/python3.9/dist-packages (from retworkx->pennylane)
(0.12.1)
Installing collected packages: semantic-version, autoray, retworkx,
pennylane-lightning, pennylane
Successfully installed autoray-0.6.3 pennylane-0.29.1 pennylane-
lightning-0.29.0 retworkx-0.12.1 semantic-version-2.10.0
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Collecting tensorflow-addons
  Downloading tensorflow_addons-0.19.0-cp39-cp39-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.1 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 14.8 MB/s eta
0:00:00
ent already satisfied: packaging in /usr/local/lib/python3.9/dist-
packages (from tensorflow-addons) (23.0)
Requirement already satisfied: typing-extensions>=4.4.0 in
/usr/local/lib/python3.9/dist-packages (from typeguard>=2.7-
>tensorflow-addons) (4.5.0)
Requirement already satisfied: importlib-metadata>=3.6 in
/usr/local/lib/python3.9/dist-packages (from typeguard>=2.7-
>tensorflow-addons) (6.1.0)
Requirement already satisfied: zipp>=0.5 in
/usr/local/lib/python3.9/dist-packages (from importlib-metadata>=3.6-
```

```
>typeguard>=2.7->tensorflow-addons) (3.15.0)
Installing collected packages: typeguard, tensorflow-addons
Successfully installed tensorflow-addons-0.19.0 typeguard-3.0.2
```

TASK III

Please comment on quantum computing or quantum machine learning. You can also comment on one quantum algorithm or one quantum software you are familiar with. You can also suggest methods you think are good and you would like to work on. Please use your own understanding. Comments copied from the internet will not be considered.

###General Views on quantum computing and quantum machine learning

As student of computer science with deep interest in physics i find the field of qunatum computing quite fascinating. While there are thousands of article detailing the potential of quantum computing and by extension quantum machine learning to revolutionize the way we solve complex problems and the impact will have in our day to day life. What i found most interesting aspects of quantum computing is its ability to perform certain calculations much faster than classical computers. This is due to the fact that quantum computers use qubits, which can be in a state of superposition, allowing for many calculations to be performed simultaneously. Additionally, quantum computers can also use entanglement, which allows for the manipulation of multiple qubits at the same time. These unique features make quantum computers well-suited for solving problems that would take classical computers exponentially longer.

###Quantum Algorithms

One of the classical quantum algorithms that I find particularly fascinating is the Deutsch-Jozsa algorithm. This algorithm solves the problem of determining whether a function is constant or balanced, which is a problem that is difficult to solve using classical computing. The algorithm uses a quantum computer to evaluate the function in a single step, which is much faster than the classical approach of evaluating the function multiple times. I find this algorithm to be a great example of the power of quantum computing and how it can be used to solve problems that are not feasible with classical computing.

Another important quantum algorithm is Shor's algorithm, which is used for factoring large numbers and solving the discrete logarithm problem. Shor's algorithm is one of the most well-known quantum algorithms and has significant implications for cryptography because many classical cryptographic protocols rely on the difficulty of factoring large numbers.Shor's algorithm uses the principles of quantum superposition and entanglement to find the factors of a large composite number much faster than any known classical algorithm. This algorithm has the potential to break RSA encryption

Apart form the classical Quantum Algorithms. Error correcting code is Something that captivates my thoughts particularly surface codes. Introduced by Alexei Kitaev in 1997, and have become a popular choice for quantum error correction due to their relatively simple structure and high error threshold.

The basic idea of surface codes is to encode qubits in a two-dimensional array of physical qubits (usually implemented using superconducting qubits or ion traps), such that each physical qubit is connected to its neighboring qubits. The encoding scheme ensures that any errors that occur can be detected and corrected by measuring the state of certain groups of qubits in the array.

One of the challenges however, is the implementing surface codes is the need for a large number of physical qubits. To achieve a high level of error correction, the array of physical qubits must be quite large (typically on the order of thousands or tens of thousands of qubits). This makes it difficult to implement surface codes with current quantum technologies, but there has been significant progress in recent years towards building larger and more reliable quantum computers that can support error-correction using surface codes.

###Quantum software

From my personal experience, working with quantum software . I've had the opportunity to explore various quantum software platforms, each with their own unique features and capabilities.

The IBM Quantum Experience is a powerful platform that provides access to cloud-based quantum hardware and simulators. I've found the Qiskit programming language to be quite intuitive, allowing me to easily define quantum circuits and run simulations of quantum algorithms. The visual tools provided by the platform also make it easy to analyze the results of my simulations, helping me gain a deeper understanding of the behavior of quantum systems.

The Microsoft Quantum Development Kit is another platform that I've had the chance to work with. While I initially found the Q# programming language to be a bit challenging, it has some powerful features that make it well-suited for certain types of quantum programming tasks, such as quantum error correction. The suite of tools included with the platform, such as the quantum simulator and the quantum chemistry library, also make it a valuable tool for researchers and developers working in quantum computing.

Looking to the future, I believe that quantum computing and quantum machine learning will have a significant impact on many areas of science and industry. For example, quantum computing can be used to simulate the behavior of complex systems, such as molecules, which could have important applications in drug development and materials science. Additionally, quantum machine learning can be used to improve data analysis and pattern recognition, which could have important applications in fields such as finance and cybersecurity.

In terms of future research, I am interested in exploring new quantum algorithms and developing new quantum software tools. Specifically, I am interested in developing algorithms for solving optimization problems, which are important in fields such as logistics and transportation. I am also interested in exploring the potential applications of quantum machine learning in areas such as natural language processing and image recognition.

Deutsch-Jozsa algorithm

The Deutsch-Jozsa algorithm is a quantum algorithm that determines whether a given function is constant or balanced. The algorithm uses a quantum computer to evaluate the function in a single step, which is much faster than the classical approach of evaluating the function multiple times.

The algorithm works by using a quantum computer to perform a series of operations on a set of qubits. The algorithm requires one input qubit and n ancillary qubits, where n is the number of input bits to the function. The input qubit is initially set to $|1\rangle$ and the ancillary qubits are initially set to $|0\rangle$. The algorithm then applies a Hadamard gate to each qubit, putting them all in a superposition state. The resulting state is:

$$(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes ... \otimes (|0\rangle + |1\rangle)$$

The algorithm then applies a function oracle to the input qubit and the ancillary qubits. The oracle flips the phase of the state if the function is balanced and leaves the state unchanged if the function is constant. The resulting state is:

$$(|0\rangle + (-1)^f(0)|1\rangle) \otimes (|0\rangle + (-1)^f(1)|1\rangle) \otimes ... \otimes (|0\rangle + (-1)^f(2^n-1)|1\rangle)$$

where f(x) is the value of the function for input x.

The algorithm then applies another set of Hadamard gates to each qubit, which puts the input qubit in the state:

$$(|0\rangle - |1\rangle) / \sqrt{2}$$

If the function is constant, then all of the ancillary qubits will be in the same state after the first set of Hadamard gates, and so the second set of Hadamard gates will leave the input qubit in the state $|1\rangle$. If the function is balanced, then the ancillary qubits will be in different states after the first set of Hadamard gates, and so the second set of Hadamard gates will leave the input qubit in a state other than $|1\rangle$.

###The implementation of the Deutsch-Jozsa algorithm involves several steps:

1. **Prepare the input state:** The input to the algorithm is an n-qubit register, initialized to the state $|0\rangle^(n)|1\rangle$. The first n qubits are in the state $|0\rangle$, and the last qubit is in the state $|1\rangle$.

2. **Apply the Hadamard gate:** Apply the Hadamard gate to each qubit in the input register. This creates a superposition of all possible input states.

3. **Apply the oracle:** Apply the oracle function to the input register. The oracle function is a black box that takes as input the n-qubit register and returns either 0 or 1. If the function is constant, it returns the same value for all inputs. If the function is balanced, it returns 0 for half of the inputs and 1 for the other half.

4. **Apply the Hadamard gate again:** Apply the Hadamard gate to each qubit in the input register. This transforms the superposition of input states into a superposition of output states.

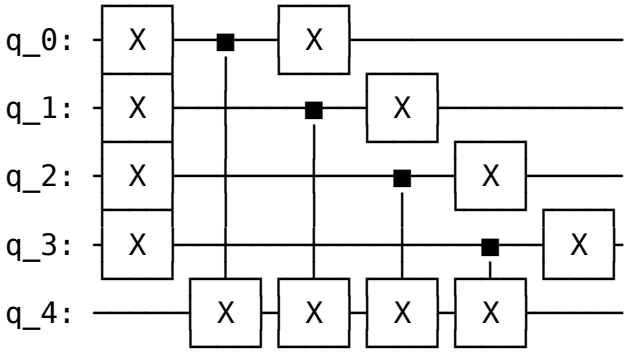
5. **Measure the output:** Measure the output qubits to obtain the result of the algorithm. If the function is constant, the output will be |0⟩^(n). If the function is balanced, the output will be some other state.

```
import numpy as np
from qiskit import execute, Aer
from qiskit.providers.ibmq import least_busy
from qiskit import QuantumCircuit, assemble, transpile
from qiskit.visualization import plot_histogram,plot_bloch_multivector

def deutsch_jozsa_oracle(case: str, num_input_qubits: int) ->
QuantumCircuit:
    """
    Creates a Deutsch-Jozsa oracle circuit for a given case.

    Args:
        case (str): The case for the oracle. Can be "balanced" or
"constant".
        num_input_qubits (int): The number of input qubits to the
oracle.

    Returns:
        A quantum circuit for the Deutsch-Jozsa oracle corresponding
to the given case.
    """
    # Create a quantum circuit with num_input_qubits + 1 qubits, one
for the output of the oracle
    oracle_circuit = QuantumCircuit(num_input_qubits + 1)

    # Case in which the oracle is balanced
    if case == "balanced":
        # Choose a random number to determine which CNOTs to wrap in
X-gates
        random_int = np.random.randint(1, 2 ** num_input_qubits)
        # Convert random_int to a binary string of length
num_input_qubits
        random_str = format(random_int, '0' + str(num_input_qubits) +
'b')
        # Apply X-gates to the qubits corresponding to 1s in
random_str
        for qubit in range(len(random_str)):
            if random_str[qubit] == '1':
                oracle_circuit.x(qubit)
        # Apply CNOT gates from each input qubit to the output qubit
        for qubit in range(num_input_qubits):
            oracle_circuit.cx(qubit, num_input_qubits)
```

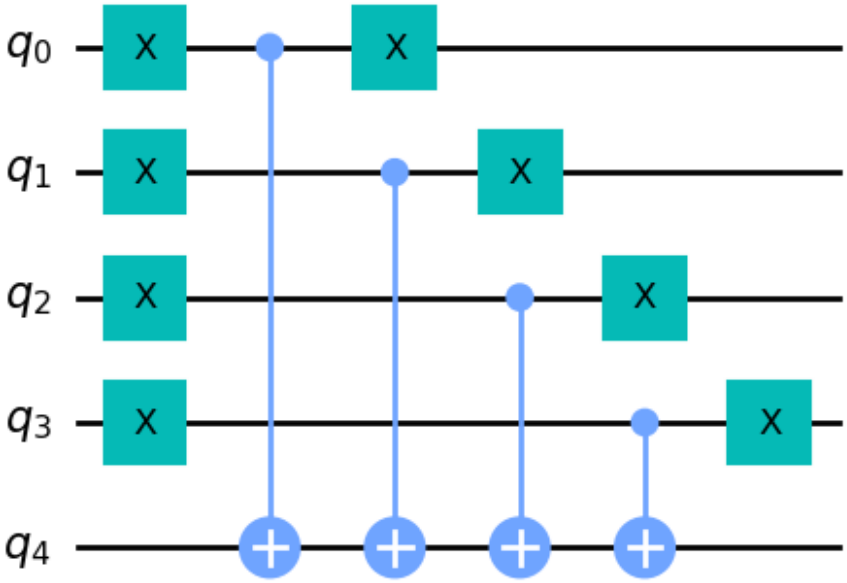
```
        # Apply X-gates again to the qubits corresponding to 1s in
random_str
        for qubit in range(len(random_str)):
            if random_str[qubit] == '1':
                oracle_circuit.x(qubit)

    # Case in which the oracle is constant
    if case == "constant":
        # Choose a random bit to be the fixed output of the oracle
        output = np.random.randint(2)
        # If the output is 1, apply an X-gate to the output qubit
        if output == 1:
            oracle_circuit.x(num_input_qubits)

    # Return the oracle circuit
    return oracle_circuit

from qiskit import QuantumCircuit

def deutsch_jozsa_algorithm(oracle_circuit: QuantumCircuit,
input_size: int) -> QuantumCircuit:
    """Runs the Deutsch-Jozsa algorithm on the given oracle circuit.

    Args:
        oracle_circuit (QuantumCircuit): The oracle circuit to test.
        input_size (int): The number of input qubits to the oracle
circuit.

    Returns:
        QuantumCircuit: The quantum circuit that implements the
Deutsch-Jozsa algorithm.
    """

    # Create the quantum circuit with n+1 qubits for input and output
    dj_circuit = QuantumCircuit(input_size + 1, input_size)

    # Set up the output qubit to |1>
    dj_circuit.x(input_size)
    dj_circuit.h(input_size)

    # Set up the input register to a superposition of all possible
states
    for qubit in range(input_size):
        dj_circuit.h(qubit)

    # Append the oracle gate to our circuit
    dj_circuit.append(oracle_circuit, range(input_size + 1))

    # Perform the H-gates again on the input register
```

```
    for qubit in range(input_size):
        dj_circuit.h(qubit)

    # Measure the input qubits
    for i in range(input_size):
        dj_circuit.measure(i, i)

    return dj_circuit
```

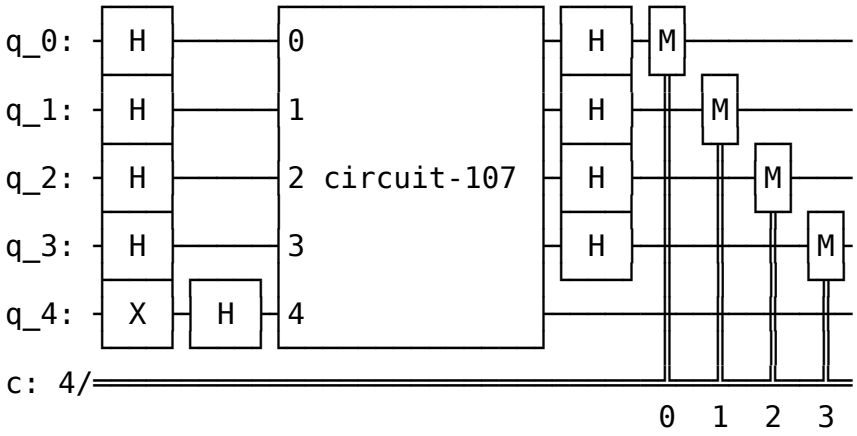
```
input_size = 4
oracle_gate = deutsch_jozsa_oracle('balanced', input_size)
oracle_gate.draw()
```



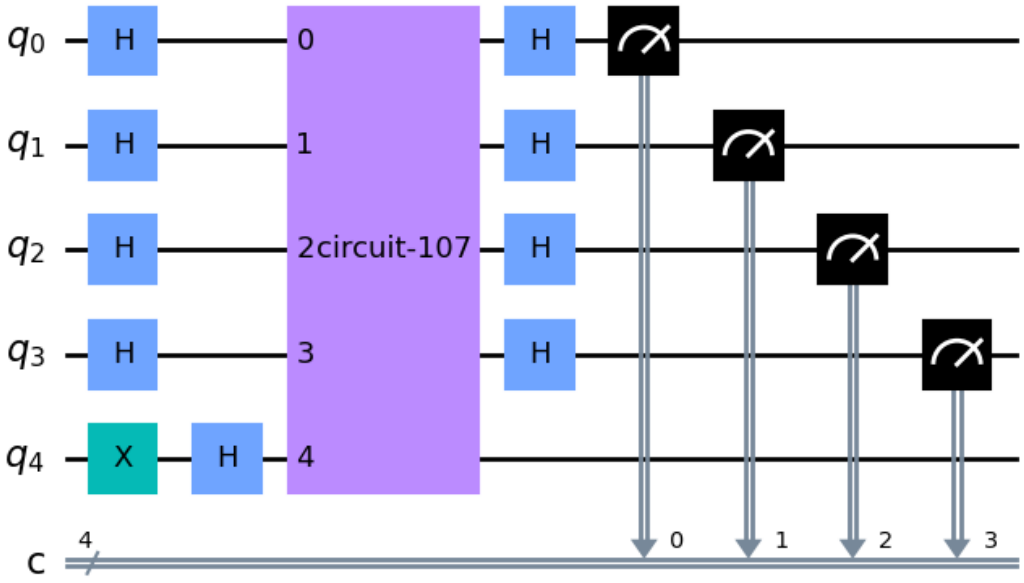

```
import matplotlib.pyplot as plt
oracle_gate.draw(output='mpl')
```



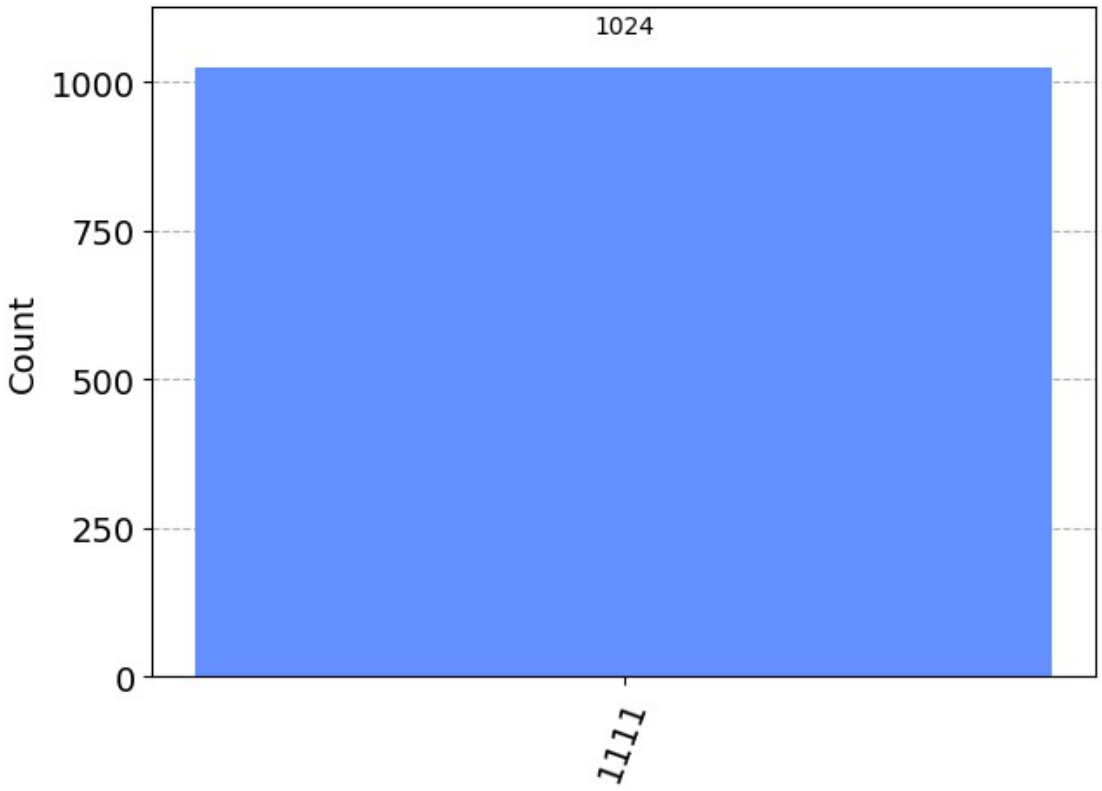
```
dj_circuit = deutsch_jozsa_algorithm(oracle_gate,input_size)
dj_circuit.draw()
```




```
dj_circuit.draw(output='mpl')
```




```
backend = Aer.get_backend('qasm_simulator')
counts = execute(dj_circuit, backend).result().get_counts()

plot_histogram(counts)
```


1024
1000
750
500
250
0
Count
1111