	Quantum Pratical A	ssignment	-
	Grupo 23 Elementos: • a87982 Bruna Dias Carvalho • a82124 Maria Luísa Faria Silva		=
	Conteúdo		
	 Algoritmo de Procura Simulação de Ruído Execução em Computador Qua IGNIS 	ântico	
In [1]:	<pre>from qiskit import execute</pre>	Circuit, ClassicalRegister, , transpile	
In [2]:	<pre>import matplotlib.pyplot a %matplotlib inline</pre>	_	, plot_state_city, plot_state_hir
In [3]:	<pre>s = 23 % 8 print(s) 7 wb = bin(s)[2:]</pre>		
In [4]:	<pre>print(wb) 111 x = len(wb) print('number of qubits: '</pre>	, x)	
In [5]:	number of qubits: 3 Primeiro começamos por criar os q qr_x = QuantumRegister(x,	ubits necessário para representar c	o nosso número, ou seja, x qubits
	<pre>backend = Aer.get_backend(cr=ClassicalRegister(x,'cr qc_Grover= QuantumCircuit(</pre>	"qasm_simulator")	
	Algoritmo Este algoritmo de procura(Grover's inicialização, oracle e amplificação. Inicialização do sistema com a mes		da encontra-se dividido em 3 fases, tates possíveis
In [6]:	# Init	$\sum_{x_i} x_i angle$	Edites possivels
Out[6]:	qc_Grover.h(0) qc_Grover.h(1) qc_Grover.h(2) <qiskit.circuit.instruction Aplica \sqrt{N} vezes a seguinte opera</qiskit.circuit.instruction 	nset.InstructionSet at 0x17c.	
	,	operador é responsável por identif $-lpha_m x_m angle+eta\sum_{x_i eq x_m} x_i angle$	ficar a solução para o problema e
In [7]:	Com esta implementação, a fase do sofrem alterações. import math as m		ns, enquanto os outros estados não
	<pre>times= round(m.sqrt(2**x)) print(times) 3 Posteriormente, aplicamos um diffu</pre>	user com intuito de uma amplificaçã	ão, ou seja, para que a probabilidade
In [8]:	do x_m aumente e as dos outros es def select_w(circuit, qr_x # Estamos à procura do # Logo não é preciso a pass	:): o estado 111>	
	<pre>def phase_oracle(circuit, select_w(circuit, qr_x circuit.h(0) circuit.ccx(2, 1, 0) circuit.h(0) select_w(circuit, qr_x</pre>	.)	
	<pre>def diffuser(circuit, qr_x</pre>):	
	<pre>circuit.x(qr_x) circuit.h(qr_x) #oracle and diffuser for t in range(2): # phase oracle 2</pre>		
	<pre>phase_oracle(qc_Grover # diffuser diffuser(qc_Grover,qr_ qc_Grover.measure(qr_x,cr)</pre>	(x)	
Out[8]:	qc_Grover.draw(output='mpl x ₀ - H - H - H - H - X - X ₁ - H - X	H — H — X — H — H — H — H — H — H — H —	н — х — х
In [9]:		ckend('statevector_simulator	x H 1 2 0
Out[9]:	result = execute(qc_Grover psi1 = result.get_stateve plot_state_city(psi1)	r, backend_state).result() cctor(qc_Grover)	
		1.0 0.8 0.6 9 0.4 0.2 0.0	o [6]ml
	90,90,01,00,01,00,10,00,00,00,00,00,00,00,00	0.0 1111 1000	
In [10]: Out[10]:	plot_state_hinton(psi1) Re[ρ]	Im[ρ]	
	000 - 001 - 010 - 011 - 100 -	000 - 001 - 010 - 011 - 100 -	
	100 000 000 1111 - 001 001 001 001 001 0	101 -	110 - 111
In [11]:	<pre>backend_state = Aer.get_ba shots=1024</pre>	<pre>ackend('qasm_simulator') # th c, backend, shots=shots).resu</pre>	he device to run on
Out[11]:	1.00	0.94	42
	Probabilities 0.75		
	0.25 0.007 0.008 0.009 00 00 00 00 00 00 00 00 00 00 00 00	0.007 0.009 0.009 0.010	7
In [12]: Out[12]:	qc_Grover.depth()	10 00 10 10 10 10 10 10 10 10 10 10 10 1	
	Simulação de Ru Para poder utilizar o simulador de r como tal necessitamos de carregar	ruído, é necessário recorrer a comp	outadores quânticos verdadeiros, e
In [13]:	IBMQ.save_account('778d50a	1f7e64f3d7b4ceb3066c93fed504	414df05f8cc64e7cf0f9f554ab4ecab57, 171: Credentials already presen
In [14]: Out[14]:	ain')>,		<pre>ibm-q', group='open', project='m open', project='main')>.</pre>
	<pre><ibmqbackend('ibmq_16_melb n')="">,</ibmqbackend('ibmq_16_melb></pre>	oourne') from IBMQ(hub='ibm-q', grown of the state of the	<pre>q', group='open', project='mai oup='open', project='main')>, oup='open', project='main')>, group='open', project='main')>,</pre>
	<pre><ibmqbackend('ibmq_quito'))="" <ibmqsimulator('simulator_="main">,</ibmqbackend('ibmq_quito')></pre>	from IBMQ(hub='ibm-q', growstatevector') from IBMQ(hub='mps') from IBMQ(hub='ibm-q') extended_stabilizer') from interpretation of the stabilizer of the stab	<pre>up='open', project='main')>, ='ibm-q', group='open', project , group='open', project='main') IBMQ(hub='ibm-q', group='open',</pre>
In [15]:	<pre><ibmqsimulator('simulator_ ='main')="">, <ibmqbackend('ibmq_manila'< pre=""></ibmqbackend('ibmq_manila'<></ibmqsimulator('simulator_></pre>) from IBMQ(hub='ibm-q', gro	<pre>'ibm-q', group='open', project oup='open', project='main')>]</pre>
P	Analisando as propriedades dos dir optamos por escolher o backend _i que os <i>qubits</i> percam as suas propriño ser muito elevado.	ibmq_16 <i>melbourne</i> por terem valor	res baixos de <i>AVG T1/T2</i> , de modo a
In [16]:	<pre>my_provider_ibmq = IBMQ.ge # Define backend</pre>	et_provider(hub='ibm-q', grouder_ibmq_1, grouder_ibmq_1)	
	<pre>backend_device = my_provid</pre>		
In [17]: In [18]:	%qiskit_job_watcher	ice configuration() coupling	a man
	<pre>%qiskit_job_watcher coupling_map = backend_dev from qiskit.providers.aer.</pre>		g_map
In [18]:	<pre>%qiskit_job_watcher coupling_map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis gates: ['cx', 'id', Instructions with noise: Qubits with noise: [0, 1,</pre>	noise import NoiseModel from_backend(backend_device) 'rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10	, 'cx']
In [18]:	<pre>%qiskit_job_watcher coupling_map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis gates: ['cx', 'id', Instructions with noise: Qubits with noise: [0, 1, Specific qubit errors: [([4]), ('id', [5]), ('id', [1]), ('id', [11]), ('id', [12]), x', [2]), ('sx', [3]), ('sx', [8]), ('sx', [9]), ('sx', [1]) [6]), ('x', [7]), ('x', [8] [13]), ('x', [14]), ('cx',</pre>	noise import NoiseModel from_backend(backend_device) ['xz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', [0]), ('id', [1]), ('id', [0]), ('id', [1]), ('id', [1]), ('sx', [4]), ('sx', [5]), ('sx', [10]), ('sx', [11]), ('sx', ('x', [2]), ('x', [3]), ('x', [10]), ('x', [10]), (14, 0]), ('cx', [0, 14]),	, 'cx'] , 11, 12, 13, 14] id', [2]), ('id', [3]), ('id',]), ('id', [9]), ('id', [10]), , ('sx', [0]), ('sx', [1]), ('s , [6]), ('sx', [7]), ('sx', [12]), ('sx', [13]), ('sx', [1 'x', [4]), ('x', [5]), ('x', ('x', [11]), ('x', [12]), ('x', ('cx', [14, 13]), ('cx', [13, 1
In [18]:	<pre>%qiskit_job_watcher coupling_map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis gates: ['cx', 'id', Instructions with noise: Qubits with noise: [0, 1, Specific qubit errors: [(4]), ('id', [5]), ('id', [4]), ('id', [11]), ('id', [12]), x', [2]), ('sx', [3]), ('sx', [4]), ('x', [9]), ('x', [1]) [6]), ('x', [0]), ('x', [1]) [6]), ('x', [6, 8]), ('cx', [1, 2]), ('cx', [12, 11]), ('cx', [12, 11]), ('cx', [12, 11]), ('cx', [2, 3]), ('cx', [3, [0, 1]), ('measure', [0]),</pre>	rom_backend(backend_device) 'rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', [0]), ('id', [1]), ('id', [6]), ('id', [7]), ('id', [8]) ('id', [13]), ('id', [14]) ('id', [13]), ('id', [14]) ('x', [4]), ('sx', [5]), ('sx', [10]), ('sx', [11]), ('sx', [10]), ('x', [9]), ('x', [10]), ('cx', [5, 9]), ([8, 6]), ('cx', [3, 11]), ('cx', [10, 14]), ('cx', [10, 11]), ('cx', [11, 10]), ('cx', [11, 10]), ('cx', [10, 11]), ('cx', [11, 10]), ('cx', [11]), ('cx', [11]), ('c	, 'cx'] , 11, 12, 13, 14] id', [2]), ('id', [3]), ('id',]), ('id', [9]), ('id', [10]), , ('sx', [0]), ('sx', [1]), ('s , [6]), ('sx', [7]), ('sx', [12]), ('sx', [13]), ('sx', [1 'x', [4]), ('x', [5]), ('x', ('x', [11]), ('x', [12]), ('x', ('cx', [14, 13]), ('cx', [13, 1 'cx', [9, 5]), ('cx', [4, 10]), x', [12, 2]), ('cx', [2, 12]), cx', [12, 13]), ('cx', [11, 1 0]), ('cx', [9, 10]), ('cx', [1 , ('cx', [8, 7]), ('cx', [5, 'cx', [4, 3]), ('cx', [3, 4]), [2, 1]), ('cx', [1, 0]), ('cx', ', [2]), ('measure', [3]), ('mea
In [18]: In [20]: In [21]:	<pre>%qiskit_job_watcher coupling_map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis gates: ['cx', 'id', Instructions with noise: Qubits with noise: [0, 1, Specific qubit errors: [(4]), ('id', [5]), ('id', [4]), ('id', [11]), ('id', [12]), x', [2]), ('sx', [3]), ('sx', [4]), ('x', [9]), ('x', [1]) [6]), ('x', [0]), ('x', [1]) [6]), ('x', [10]), ('cx', [14]), ('cx', [10, 4]), ('cx', [11]) ('cx', [13, 1]), ('cx', [11]) ('cx', [13, 1]), ('cx', [11]) ('cx', [12, 11]), ('cx', [12]), ('cx', [12], [12]), ('cx', [12], [12]), ['cx', [2], 3]), ('cx', [3], [0, 1]), ('measure', [0]), sure', [4]), ('measure', [5])</pre>	rom_backend(backend_device) 'rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', [0]), ('id', [1]), ('id', [8]), ('id', [13]), ('id', [14]) ('id', [13]), ('id', [14]), ('sx', [4]), ('sx', [5]), ('sx', [10]), ('x', [9]), ('x', [10]), ('cx', [0, 14]), [8, 6]), ('cx', [3, 11]), ('cx', [10, 14]), ('cx', [10, 14]), ('cx', [10, 14]), ('cx', [10, 14]), ('cx', [10, 11]), ('cx', [11, 10]), ('cx', [10, 11]), ('cx', [11, 10]), ('cx', [10, 11]), ('cx', [10, 11]), ('cx', [10, 11]), ('cx', [11, 10]), ('cx', [10, 11]), ('cx', [11, 10]), ('cx', [10, 11]), ('measure', [1]), ('measure', [1]), ('measure', [10]), ('measure', [11])	, 'cx'] , 11, 12, 13, 14] id', [2]), ('id', [3]), ('id',]), ('id', [9]), ('id', [10]), , ('sx', [0]), ('sx', [1]), ('s , [6]), ('sx', [7]), ('sx', [12]), ('sx', [13]), ('sx', [1 'x', [4]), ('x', [5]), ('x', ('x', [11]), ('x', [12]), ('x', ('cx', [14, 13]), ('cx', [13, 1 'cx', [9, 5]), ('cx', [4, 10]), x', [12, 2]), ('cx', [2, 12]), cx', [12, 13]), ('cx', [11, 1 0]), ('cx', [9, 10]), ('cx', [1 , ('cx', [8, 7]), ('cx', [5, 'cx', [4, 3]), ('cx', [3, 4]), [2, 1]), ('cx', [1, 0]), ('cx',
In [18]: In [20]:	<pre>%qiskit_job_watcher coupling_map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis gates: ['cx', 'id', Instructions with noise: Qubits with noise: [0, 1, Specific qubit errors: [([4]), ('id', [5]), ('id', [12]), x', [2]), ('sx', [3]), ('sx', [4]), ('x', [0]), ('x', [1]) [6]), ('x', [7]), ('x', [8] [13]), ('x', [14]), ('cx', [14]), ('cx', [14]), ('cx', [15]), ('cx', [10, 4]), ('cx', [11]), ('cx', [12, 11]), ('cx', [2, 3]), ('cx', [2, 3]), ('cx', [3, [0, 1]), ('measure', [0]), sure', [4]), ('measure', [5]), ('measure', [14])] basis_gates = noise_model. result_noise = execute(qc_noicounts_noise = result_noise</pre>	rom_backend(backend_device) 'rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', [0]), ('id', [1]), ('id', [8] ('id', [13]), ('id', [14]) ('id', [13]), ('id', [14]) ('id', [13]), ('ix', [3]), ('sx', [10]), ('sx', [10]), ('x', [10]), ('x', [10]), ('cx', [5, 9]), ('cx', [3, 11]), ('cx', [13]), ('cx', [13, 12]), ('cx', [10, 11]), ('cx', [11, 10]), ('cx', [10, 11]), ('measure', [1]), ('measure', [1]), ('measure', [1]), ('measure', [10]), ('measure', [11]) basis_gates Grover, backend, se_model=noise_model, se_model=soise_model, se_ing_map=coupling_map, sis_gates=basis_gates).result se.get_counts(qc_Grover)	, 'cx'] , 11, 12, 13, 14] id', [2]), ('id', [3]), ('id',]), ('id', [9]), ('id', [10]), , ('sx', [0]), ('sx', [1]), ('s , [6]), ('sx', [7]), ('sx', [12]), ('sx', [13]), ('sx', [1 'x', [4]), ('x', [5]), ('x', ('x', [14, 13]), ('cx', [13, 1 'cx', [9, 5]), ('cx', [4, 10]), x', [12, 2]), ('cx', [2, 12]), cx', [12, 13]), ('cx', [11, 1 0]), ('cx', [9, 10]), ('cx', [1, 1 0]), ('cx', [8, 7]), ('cx', [3, 4]), [2, 1]), ('cx', [1, 0]), ('cx', ', [2]), ('measure', [3]), ('measure', [7]), ('measure', [8]), , ('measure', [12]), ('measure',
In [18]: In [20]: In [21]:	<pre>%qiskit_job_watcher coupling_map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis gates: ['cx', 'id', Instructions with noise: Qubits with noise: [0, 1, Specific qubit errors: [([4]), ('id', [5]), ('id', [12]), x', [2]), ('sx', [3]), ('sx', [8]), ('sx', [9]), ('xx', [1]) [6]), ('x', [0]), ('x', [1]) [6]), ('x', [0]), ('x', [1]) [6]), ('x', [6, 8]), ('cx', ('cx', [10, 4]), ('cx', [11]), ('cx', [13]), ('cx', [14]), ('cx', [2, 3]), ('cx', [3, [0, 1]), ('measure', [9]), ('measure', [9]), yure', [4]), ('measure', [14])] basis_gates = noise_model. result_noise = execute(qc_</pre>	rom_backend(backend_device) 'rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', [0]), ('id', [1]), ('id', [8] ('id', [13]), ('id', [14]) ('id', [13]), ('id', [14]) ('id', [13]), ('ix', [3]), ('sx', [10]), ('sx', [10]), ('x', [10]), ('x', [10]), ('cx', [5, 9]), ('cx', [3, 11]), ('cx', [13]), ('cx', [13, 12]), ('cx', [10, 11]), ('cx', [11, 10]), ('cx', [10, 11]), ('measure', [1]), ('measure', [1]), ('measure', [1]), ('measure', [10]), ('measure', [11]) basis_gates Grover, backend, se_model=noise_model, se_model=soise_model, se_ing_map=coupling_map, sis_gates=basis_gates).result se.get_counts(qc_Grover)	<pre>, 'cx'] , 11, 12, 13, 14] id', [2]), ('id', [3]), ('id',]), ('id', [9]), ('id', [10]), , ('sx', [0]), ('sx', [1]), ('s , [6]), ('sx', [7]), ('sx', [12]), ('sx', [13]), ('sx', [1 'x', [4]), ('x', [5]), ('x', ('x', [14]), ('x', [12]), ('x', ('cx', [14, 13]), ('cx', [13, 1 'cx', [9, 5]), ('cx', [4, 10]), x', [12, 2]), ('cx', [2, 12]), cx', [12, 13]), ('cx', [11, 1 0]), ('cx', [9, 10]), ('cx', [1 , ('cx', [4, 3]), ('cx', [3, 4]), [2, 1]), ('cx', [1, 0]), ('cx', ', [2]), ('measure', [3]), ('measure', [7]), ('measure', [7]), ('measure', [8]), , ('measure', [12]), ('measure', t()</pre>
In [18]: In [19]: In [20]: In [21]:	<pre>coupling_map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis gates: ['cx', 'id', Instructions with noise: Qubits with noise: [0, 1, Specific qubit errors: [(</pre>	'rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', [0]), ('id', [1]), ('id', [8] ('id', [13]), ('id', [14]) (', [4]), ('sx', [5]), ('sx', ('x', [2]), ('x', [3]), () ('x', [9]), ('x', [10]), ('cx', [14]), [8, 6]), ('cx', [3, 11]), ('cx', [13]), ('cx', [14, 5]), ('cx', [14, 5]), ('cx', [14, 5]), ('measure', [1]), ('measure', [1]), ('measure', [10]), ('measure', [10]), ('measure', [11]) basis_gates Grover, backend, se_model=noise_model, upling_map=coupling_map, ois_gates=basis_gates).resulting_map=coupling_map, ois_gates=basis_gates).resulting_map=coupling_map=coupling_map=coupling_map=coupling_map=coupling_map=coupling_map=coupling_map=coupling_map=co	<pre>, 'cx'] , 11, 12, 13, 14] id', [2]), ('id', [3]), ('id',]), ('id', [9]), ('id', [10]), , ('sx', [0]), ('sx', [1]), ('s , [6]), ('sx', [7]), ('sx', [12]), ('sx', [13]), ('sx', [1 'x', [4]), ('x', [5]), ('x', ('x', [11]), ('x', [12]), ('x', ('cx', [14, 13]), ('cx', [13, 1 'cx', [9, 5]), ('cx', [4, 10]), x', [12, 2]), ('cx', [2, 12]), cx', [12, 13]), ('cx', [11, 1 0]), ('cx', [9, 10]), ('cx', [1, , ('cx', [4, 3]), ('cx', [3, 4]), [2, 1]), ('cx', [1, 0]), ('cx', ', [2]), ('measure', [3]), ('measure', [7]), ('measure', sure', [7]), ('measure', [8]), , ('measure', [12]), ('measure', </pre>
In [18]: In [19]: In [20]: In [21]:	#qiskit_job_watcher coupling_map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis gates: ['cx', 'id', Instructions with noise: Qubits with noise: [0, 1, Specific qubit errors: [([4]), ('id', [5]), ('id', [2]), x', [2]), ('sx', [3]), ('sx', [4]), ('sx', [9]), ('sx', [1]) [6]), ('x', [0]), ('x', [1]) [6]), ('x', [10, 4]), ('cx', [11, 4]), ('cx', [10, 4]), ('cx', [10, 4]), ('cx', [10, 4]), ('cx', [12, 11]), ('cx', [2, 3]), ('cx', [3, 0, 1]), ('measure', [0]), sure', [4]), ('measure', [14])] basis_gates = noise_model. result_noise = execute(qc_noise_model) 0.00 0	rom_backend(backend_device) 'rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', [0]), ('id', [1]), ('id', [8]), ('id', [13]), ('id', [14]) (', [4]), ('sx', [5]), ('sx', [10]), ('sx', [11]), ('sx', ('x', [2]), ('x', [10]), [14, 0]), ('cx', [5, 9]), (13]), ('cx', [3, 11]), ('cx', [10, 14]), ('cx', [10, 14]), ('cx', [10, 14]), ('cx', [10, 14]), ('cx', [10, 11]), ('cx', [14, 5]), ('cx', [4, 5]), ('cx', [4, 5]), ('cx', [10]), ('measure', [1]), ('measure', [1]), ('measure', [10]), ('measure', [11]) basis_gates Grover, backend, se_model=noise_model, upling_map=coupling_map, uis_gates=basis_gates).result ge.get_counts(qc_Grover) ee, title="# Dense Coding with upwith depolarizing noise model 0.26 0.114 0.117 0.094 0.082	<pre>, 'cx'] , 11, 12, 13, 14] id', [2]), ('id', [3]), ('id',]), ('id', [9]), ('id', [10]), , ('sx', [0]), ('sx', [1]), ('s , [6]), ('sx', [7]), ('sx', [12]), ('sx', [13]), ('sx', [1 'x', [4]), ('x', [5]), ('x', ('x', [11]), ('x', [12]), ('x', ('cx', [14, 13]), ('cx', [13, 1 'cx', [9, 5]), ('cx', [4, 10]), x', [12, 2]), ('cx', [2, 12]), cx', [12, 13]), ('cx', [11, 1 0]), ('cx', [9, 10]), ('cx', [1, , ('cx', [4, 3]), ('cx', [3, 4]), [2, 1]), ('cx', [1, 0]), ('cx', ', [2]), ('measure', [3]), ('measure', [7]), ('measure', sure', [7]), ('measure', [8]), , ('measure', [12]), ('measure', </pre>
In [18]: In [20]: In [21]: In [22]:	*qiskit_job_watcher coupling_map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis gates: ['cx', 'id', Instructions with noise: Qubits with noise: [0, 1, Specific qubit errors: [([4]), ('id', [5]), ('id', [12]), x', [2]), ('sx', [9]), ('sx', [1]), [6]), ('x', [7]), ('x', [1]), [6]), ('x', [7]), ('x', [8], ('cx', [13]), ('cx', [14]), ('cx', [14]), ('cx', [13, 1]), ('cx', [14], ('cx', [13, 1]), ('cx', [14], ('cx', [2, 3]), ('cx', [3, [0, 1]), ('measure', [0]), sure', [4]), ('measure', [14])] basis_gates = noise_model. result_noise = execute (qc_ noi cou bas counts_noise = result_nois plot_histogram(counts_nois) # Dense Coding 0.32 # Dense Coding 0.32 # Dense Coding 0.096	rom_backend(backend_device) 'rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', [0]), ('id', [1]), ('id', [8] ('id', [13]), ('id', [14]) (', [4]), ('sx', [5]), ('sx', [10]), ('sx', [11]), ('sx', [10]), ('x', [2]), ('x', [3]), (('x', [2]), ('x', [3]), ('cx', [13, 12]), ('cx', [13, 12]), ('cx', [10, 11]), ('cx', [11, 1]) [13, ('cx', [13, 12]), ('x', [10, 11]), ('cx', [11, 1]) [14, 0]), ('cx', [13, 12]), ('x', [10, 11]), ('cx', [11, 1]) [15, 4]), ('cx', [1, 2]), ('cx', [7, 8]) [10, 1]), ('measure', [6]), ('measure', [10]), ('measure', [11]) basis_gates Grover, backend, se_model=noise_model, upling_map=coupling_map, uis_gates=basis_gates).result se_get_counts(qc_Grover) se, title="# Dense Coding with with depolarizing noise model 10.26	, 'cx'] , 11, 12, 13, 14] id', [2]), ('id', [3]), ('id',]), ('id', [9]), ('id', [10]), , ('sx', [0]), ('sx', [1]), ('sx', [12]), ('sx', [13]), ('sx', [1 'x', [41)), ('x', [5]), ('x', ('x', [14, 13]), ('cx', [13, 1 'cx', [9, 5]), ('cx', [2, 12]), cx', [12, 2]), ('cx', [2, 12]), cx', [12, 13]), ('cx', [1, 1 0]), ('cx', [9, 10]), ('cx', [1, 1 0]), ('cx', [9, 10]), ('cx', [5, 1 'cx', [4, 3]), ('cx', [3, 4]), [2, 1]), ('measure', [3]), ('measure', [7]), ('measure', [8]), , ('measure', [12]), ('measure', [8]), , ('measure', [12]), ('measure', [8]) t()
In [18]: In [19]: In [20]: In [21]:	# giskit_job_watcher coupling_map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis gates: ['cx', 'id', Instructions with noise: Qubits with noise: [0, 1, Specific qubit errors: [([4]), ('id', [5]), ('id', [12]), ('x', [2]), ('sx', [3]), ('sx', [4]), ('x', [2]), ('sx', [3]), ('x', [1]) [6]), ('x', [7]), ('x', [8]] [13]), ('x', [14]), ('cx', [1, [2]), ('cx', [6, 8]), ('cx', [1, [2]), ('cx', [12, 11]), ('cx, [1, [2]), ('cx', [12, 11]), ('cx, [1, [2]), ('cx', [2, 3]), ('cx', [3, [0, 1]), ('measure', [0]), sure', [1]), ('measure', [1])] basis_gates = noise_model. result_noise = execute(qc_ noi cou bas counts_noise = result_nois plot_histogram(counts_nois # Dense Coding 0.32 # Dense Coding 0.34 # Dense Coding 0.35 # Dense Coding 0.36 # Dense Coding 0.37 # Dense Coding 0.38 # Dense Coding 0.39 # Dense Coding 0.39 # Dense Coding 0.30 # Dense Coding	rom_backend(backend_device) 'rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', [0]), ('id', [1]), ('id', [8] ('id', [13]), ('id', [14]) ('id', [13]), ('id', [14]), ('x', [4]), ('sx', [5]), ('sx', [10]), ('sx', [10]), ('x', [9]), ('x', [10]), ('cx', [13, 12]), ('cx', [10, 11]), ('cx', [13, 12]), ('cx', [10, 11]), ('cx', [10, 11]), ('cx', [11, 12]), ('cx', [10, 11]), ('cx', [11, 12]), ('cx', [10, 11]), ('cx', [11, 2]), ('cx', [10, 11]), ('measure', [11]) basis_gates Grover, backend, se_model=noise_model, pling_map=coupling_map, dis_gates=basis_gates).result puth depolarizing noise model numputador Quânt one one one one one one one o	, 'cx'] , 11, 12, 13, 14] id', [2]), ('id', [3]), ('id', [1]), , ('sx', [0]), ('sx', [1]), ('sx', [1]), ('sx', [12]), ('sx', [12]), ('xx', [12]), ('x', ('x', [14]), ('x', [12]), ('x', [14], 13]), ('cx', [13, 1], ('cx', [14, 13]), ('cx', [14, 10]), x', [12, 2]), ('cx', [2, 12]), (cx', [12, 13]), ('cx', [14, 13]), ('measure', [12]), ('measure
In [18]: In [20]: In [21]: In [22]:	*qiskit_job_watcher coupling_map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis gates: ['cx', 'id', Instructions with noise: [0, 1, Specific qubit errors: [([4]), ('id', [5]), ('id', [12]), x', [2]), ('sx', [3]), ('sx', [3]), ('sx', [3]), ('sx', [3]), ('x', [1]), ('cx', [1], ('cx', [10, 4]), ('cx', [12, 11]), ('cx', [2, 3]), ('cx', [3, 4]), ('cx', [2, 3]), ('cx', [3, 4]), ('measure', [0]), sure', [4]), ('measure', [14])] basis_gates = noise_model. result_noise = execute(qc_noi cou bas counts_noise = result_nois plot_histogram(counts_nois) # Dense Coding 0.32 # Dense Coding 0.32 # Dense Coding 0.34 # Dense Coding 0.35 # Dense Coding 0.36 # Dense Coding 0.37 # Dense Coding 0.38 # Dense Coding 0.39 # Dense Coding 0.39 # Dense Coding 0.31 # Dense Coding 0.32	rom_backend(backend_device) 'rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', [0]), ('id', [1]), ('id', [8] ('id', [13]), ('id', [14]) ('id', [13]), ('id', [14]), ('x', [4]), ('sx', [5]), ('sx', [10]), ('sx', [10]), ('x', [9]), ('x', [10]), ('cx', [13, 12]), ('cx', [10, 11]), ('cx', [13, 12]), ('cx', [10, 11]), ('cx', [10, 11]), ('cx', [11, 12]), ('cx', [10, 11]), ('cx', [11, 12]), ('cx', [10, 11]), ('cx', [11, 2]), ('cx', [10, 11]), ('measure', [11]) basis_gates Grover, backend, se_model=noise_model, pling_map=coupling_map, dis_gates=basis_gates).result puth depolarizing noise model numputador Quânt one one one one one one one o	, 'cx'] , 11, 12, 13, 14] id', [2]), ('id', [3]), ('id',]), ('id', [9]), ('sx', [1]), ('sx', [6]), ('sx', [7]), ('sx', [12]), ('sx', [13]), ('sx', [1 'x', [4]), ('x', [12]), ('x', ('x', [14], 13]), ('cx', [14, 10]), x', [12, 2]), ('cx', [2, 12]), cx', [12, 13]), ('cx', [1, 1] , ('cx', [8, 7]), ('cx', [3, 4]), [2, 1]), ('cx', [1, 0]), ('cx', [2, 1]), ('cx', [1, 0]), ('cx', [2]), ('measure', [3]), ('measure', [2]), ('measure', [3]), , ('measure', [12]), ('measure', t() th depolarizing noise model") t() th depolarizing noise model")
In [18]: In [19]: In [20]: In [21]: In [22]:	coupling_map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis gates: ['cx', 'id', Instructions with noise: [0, 1, Specific qubit errors: [(4]), ('id', [5]), ('id', [12]), x', [2]), ('sx', [3]), ('sx, [8]), ('sx', [9]), ('sx', [14]), ('x', [17]), ('x', [8], [13]), ('x', [17]), ('x', [8], [13]), ('cx', [14]), ('cx', [10, 4]), ('cx', [12, 11]), ('cx', [12, 11]), ('cx', [12, 11]), ('cx', [2, 3]), ('cx', [2, 3]), ('cx', [2, 3]), ('cx', [2, 3]), ('measure', [9]), ('measure', [9]), ('measure', [14])] basis_gates = noise_model. result_noise = execute(qc_noi	rom_backend(backend_device) 'rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', [0]), ('id', [1]), ('id', [8] ('id', [13]), ('id', [14]) ('y, [4]), ('sx', [5]), ('sx', [10]), ('sx', [11]), ('sx', [14]), ('x', [10]), ('x', [9]), ('x', [0], (4]), [8, 6]), ('cx', [0, 14]), [8, 6]), ('cx', [13, 12]), ('x', [10, 11]), ('cx', [13, 12]), ('x', [10, 11]), ('cx', [13, 12]), ('x', [10, 11]), ('cx', [14, 5]), (2]), ('cx', [1, 2]), ('cx', [1, 2]), ('cx', [1, 2]), ('measure', [1]), ('measure', [1]), ('measure', [1]), ('measure', [1]), ('measure', [1]), ('measure', [1]) basis_gates Grover, backend, se_model=noise_model, pling_map=coupling_map, nis_gates=basis_gates).result se.get_counts(qc_Grover) se, title="# Dense Coding with set.get_counts(qc_Grover) se, title="# Dense Coding with set.get_counts(qc_Grover) se, title="# Dense Coding with set.get_counts(qc_Grover) set.get_	, 'cx'] , 11, 12, 13, 14] id', [2]), ('id', [3]), ('id', [1)], ('ia', [9]), ('id', [10]), , ('sx', [0]), ('sx', [1]), ('sx', [12]), ('sx', [12]), ('sx', [12]), ('xx', [14]), ('xx', [14]), ('xx', [14], 10], ('cx', [14, 10]), ('cx', [14, 10]), ('cx', [2, 12]), cx', [12, 13]), ('cx', [1, 1], ('cx', [4, 3]), ('cx', [1, 1], ('cx', [4, 3]), ('cx', [1, 0]), ('cx', [2]), ('measure', [3]), ('measure', [12]), ('measure
In [18]: In [19]: In [20]: In [21]: In [22]:	coupling_map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis gates: ['cx', 'id', Specific qubit errors: [([4]), ('id', [5]), ('id', [12]), x', [2]), ('sx', [3]), ('sx', [4]), ('id', [1]), ('id', [12]), x', [2]), ('sx', [9]), ('sx', [1], [6]), ('x', [1], ('cx', [1], ('cx', [10, 4]), ('cx', [11, 4]), ('cx', [10, 4]), ('cx', [11, 4]), ('cx', [12, 11]), ('cx', [13, 12, 11]), ('cx', [13, 12, 12]), ('cx', [14, 12]), ('cx', [14, 12]), ('measure', [6]), sure', [4]), ('measure', [6]), sure', [4]), ('measure', [6]), ('measure', [6]), ('measure', [6]), ('measure', [6]), ('measure', [14])] basis_gates = noise_model. result_noise = execute(qc_noious_noise_model) result_noise = execute(qc_noious_noise_model) 0.32 # Dense Coding 0.34 # Dense Coding 0.35 # Dense Coding 0.36 # Dense Coding 0.37 # Dense Coding 0.38 # Dense Coding 0.39 # Dense Coding 0.31 # Dense Coding 0.32 # Dense Coding 0.32 # Dense Coding 0.33 # Dense Coding 0.34 # Dense Coding 0.35 # Dense Coding 0.36 # Dense Coding 0.37 # Dense Coding 0.38 # Dense Coding 0.39 # Dense Coding 0.31 # Dense Coding 0.32 # Dense Coding 0.32 # Dense Coding 0.33 # Dense Coding 0.34 # Dense Coding 0.35 # Dense Coding 0.36 # Dense Coding 0.37 # Dense Coding 0.38 # Dense Coding 0.39 # Dense Coding 0.31 # Dense Coding 0.32 # Dense Coding 0.32 # Dense Coding 0.33 # Dense Coding 0.34 # Dense Coding 0.35 # Dense Coding 0.36 # Dense Coding 0.36 # Dense Coding 0.37 # Dense Coding 0.38 # Dense Coding 0.39 # Dense Coding 0.30 # Dense Coding 0.31 # Dense Coding 0.32	"rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', [0]), ('id', [1]), ('sk', [6]), ('id', [13]), ('id', [14]), ('sx', [5]), ('sx', [10]), ('sx', [2]), ('x', [9]), ('x', [9]), ('x', [10]), ('sx', [10]), ('cx', [13, 12]), ('cx', [10, 1]), ('cx', [11, 1]), ('cx', [11, 1]), ('cx', [11, 1]), ('cx', [13, 12]), ('cx', [13, 12]), ('cx', [14, 2]), ('cx', [14, 2]), ('cx', [14, 2]), ('cx', [17, 2]), ('cx', [18, 9]), ('cx', [17, 2]), ('cx', [18, 9]), ('measure', [1]), ('measure', [1]), ('measure', [1]), ('measure', [1]), ('measure', [10]), ('measure', [11]) basis_gates Grover, backend, se_model=noise_model, pling_map=coupling_map, sis_gates=basis_gates).result se_get_counts(qc_Grover) se, title="# bense Coding wit with depolarizing noise model import backend_overview, backend, se_get_counts(qc_Grover) se, title="# bense Coding wit with depolarizing noise model import backend_overview, backend, se_get_counts(qc_Grover) se, title="# bense Coding wit import backend_overview, backend, se_get_counts(qc_Grover) se, title="# bense Coding wit import backend_overview, backend, se_get_counts(qc_Grover) se, title="# bense Coding wit import backend_overview, backend, se_get_counts(qc_Grover) se, title="# bense Coding wit import backend_overview, backend, se_get_counts(qc_Grover) se_get_get_get_get_get_get_get_get_get_ge	(ckend_monitor tico (ckend_monitor ibmq_belem ., 'cax', [1], ('massure', [12]), ('massure', [13]), ('ex', [1]), ('ex', [14]), ('ex', [14]), ('ex', [14]), ('ex', [14]), ('ex', [14]), ('ex', [14], [15]), ('ex', [14], [15]), ('ex', [14], [15]), ('ex', [14], [15]), ('ex', [16]), ('ex', [17]), ('ex', [18]), ('ex', [18]), ('ex', [19]), ('ex', [19]), ('ex', [19]), ('massure', [19]), ('measure',
In [18]: In [19]: In [20]: In [21]: In [22]:	coupling_map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis gates: ['cx', 'id', Instructions with noise: Qubits with noise: [0, 1, Specific qubit errors: [(4]), ('id', [5]), ('id', [4]), ('id', [1]), ('id', [1]), ('id', [1]), ('sx', [3]), ('sx', [9]), ('sx', [9]), ('x', [1]), ('cx', [1], ('cx', [2, 3]), ('cx', [2, 3]), ('cx', [3], ('cx', [2, 3]), ('cx', [3], ('cx', [2], [3], ('measure', [9]), ('measure', [9]), ('measure', [14])] basis_gates = noise_model. result_noise = execute(qc_noi	noise import NoiseModel "rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', [0]), ('id', [1]), ('d', [6]), ('id', [7]), ('id', [8]) ('id', [13]), ('id', [14]), ('sx', [11]), ('sx', [11]), ('sx', [21]), ('x', [21]), ('x', [21]), ('x', [3]), ('cx', [3, 11]), ('cx', [13]), ('cx', [13, 12]), ('x', [10, 11]), ('cx', [11, 2]), ('cx', [1, 2]), ('measure', [1]),	<pre> / 'cx'] , 11, 12, 13, 14] id', [2]), ('id', [3]), ('id',]), ('id', [9]), ('id', [10]), , ('sx', [9]), ('sx', [1]), ('sx', [12]), ('sx', [13]), ('sx', [12]), ('x', ('x', [11]), ('x', [12]), ('x', ('cx', [14, 13]), ('cx', [14, 10]), x', [12, 2]), ('cx', [4, 10]), x', [12, 2]), ('cx', [1, 1], 0]), ('cx', [8, 7]), ('cx', [1, 1], 0]), ('cx', [8, 7]), ('cx', [3, 4]), (2, 1]), ('cx', [1, 0]), ('cx', ('2]), ('measure', [3]), ('measure', sure', [7]), ('measure', [8]), , ('measure', [12]), ('measure', t() th depolarizing noise model") t() tt) tt) tico ibmq_belem</pre>
In [18]: In [19]: In [20]: In [21]: In [22]:	# Backend overview import qiskit.tools.jul, ('cx', [10, 11), ('cx', [13, 12), ('cx', [14, 12), ('cx', [10, 12), ('measure', [10]), ('measure',	rom_backend(backend_device) 'rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', [0]), ('id', [1]), ('id', [1]), ('id', [1]), ('id', [1]), ('sx', [1]), ('sx', [1]), ('sx', [1]), ('sx', [1]), ('x', [1]), ('x', [1]), ('x', [1]), ('x', [1]), ('cx', [1], [1]), ('cx', [1], [1]), ('cx', [1], [1], ('x', [1], [1]), ('cx', [1], [1]), ('x', [1], [1]), ('measure', [1]), ('meas	(ckend_monitor tico 66: Credentials are already in ueled. (ckend_monitor tico 66: Credentials are already in ueled. 66: Credentials are already. 66: Credentials are already in ueled. 66: Credentials are already. 67: ('x', [1], ('x', [2]), ('x', [1], [1], ('x', [1], ('x', [1], [1], ('x', [1], ('x', [1], [1], ('x', [1], [1], ('x',
In [18]: In [19]: In [20]: In [21]: In [22]:	*giskit_job_watcher coupling_map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis gates: ['cx', 'id', Instructions with noise: Qubits with noise: [0, 1, Specific qubit errors: [(4]), ('id', [5]), ('id', [1]), ('id', [1]), ('id', [1]), ('x', [2]), ('xx', [9]), ('xx', [9]), ('xx', [9]), ('xx', [1]), ('xx', [1], ('xx', [1], ('xx', [1], ('cx', [1], (1], ('measure', [0]), sure', [4]), ('measure', [5], ('measure', [9]), ('measure', [9]), ('measure', [1])) basis_gates = noise_model. result_noise = execute(qc	rom_backend(backend_device) 'rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', [0]), ('id', [1]), ('id', [1]), ('id', [1]), ('sx', [1]), ('sx', [1]), ('sx', [1]), ('sx', [1]), ('sx', [1]), ('x', [1], ('x', [x', [x], ('x', [x], ((ckend_monitor tico 66: Credentials are already in uelease below the depolarizing noise model") t() t() t() t() t() t() t() t
In [18]: In [20]: In [21]: In [22]: In [23]:	*giskit_job_watcher coupling_map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis_gates: ['cx', 'id', Instructions with noise: (0, 1, Specific qubit errors: [([4]), ('id', [5]), ('id', [12]), ('a', [1]), ('sx', [3]), ('sx', [3]), ('sx', [3]), ('sx', [3]), ('x', [1]), ('x', [1]), ('x', [1]), ('x', [1]), ('x', [1]), ('x', [1], ('x', [1]	rom_backend(backend_device) 'rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', [0]), ('id', [1]), ('id', [6]), ('id', [13]), ('id', [14]) ('y, [4]), ('sx', [5]), ('sx', [10]), ([xs', [13]), ('sx', [10]), ([x', [2]), ('x', [6]), ('cx', [6, 14]), ('cx', [13]), ('cx', [13]), ('cx', [13]), ('cx', [13]), ('cx', [13]), ('cx', [14]), ('2x', [14]), ('2x', [14]), ('2x', [14]), ('2x', [14]), ('2x', [14]), ('xx', [(cx') (1, 12, 13, 14) (1', (2]), ('id', [3]), ('id', [1]), ('(id', [9]), ('id', [1]), ('sx', [1]), ('sx', [1]), ('sx', [1]), ('x', [4]), ('x', [5]), ('x', ('x', [1]), ('x', [1], ('x', [1], 1]), ('cx', [14, 13]), ('measure', [3]), ('measure', [3]), ('measure', [3]), ('measure', [3]), ('measure', [3]), ('measure', [12]), ('measure', [3]), ('measure', [12]), ('measure', [3]), ('measure', [3]), ('measure', [3]), ('ax', [14, 12]), ('ax', [14, 12
In [18]: In [20]: In [21]: In [22]: In [23]:	**Backend overview import giskit_backend_overview (13), ("measure", [9]), ("cx', [2, 3]), ("cx	rom_backend(backend_device) 'rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', [0]), ('id', [1]), ('id', [6]), ('id', [7]), ('id', [8]) ('id', [13]), ('id', [14]), ('sx', [13]), ('sx', [13]), ('x', [13]), ('x', [9]), ('x', [9]), ('x', [9]), ('x', [9]), ('x', [13]), ('cx', [13, 12]), ('cx', [13, 12]), ('cx', [13, 12]), ('x', [10, 1]), ('cx', [11, 1]), ('x', [13, 1]), ('x', [10, 1]), ('x', [10, 1]), ('x', [11, 1]),	11, 12, 13, 14 id', [2]), ('id', [3]), ('id',)), ('id', [9]), ('sx', [1]), ('s, , [6]), ('sx', [1]), ('sx', , [2]), ('sx', [1]), ('sx', , [1]), ('sx', [1]), ('xx', , [4]), ('xx', [1]), ('cx', [1], , ('xx', [14], 13]), ('cx', [1], 10]), ('cx', [1], 10]), ('cx', [1], 10]), ('cx', [1], ('cx', [1], 1]), ('cx', [1], ('cx', [1], 4]), ('yx', [1], ('1]), ('xx', [1], 4]), ('yx', [1], ('1]), ('measure', [2]), ('measure', [3]), ('measure', [2]), ('measure',
In [18]: In [20]: In [21]: In [22]: Out [22]: Out [25]:	*giskit_job_watcher coupling_map = backend_dev from giskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis gates: ['cx', 'id', Instructions with noise: Qubits with noise: [0, 1, specific qubit errors: [(i4)], ('id', [5]), ('id', [1]), ('x', [1], ('x', [2], [3], ('x', [4]), ('measure', [5]), ('x', [4]), ('measure', [1]) basis_gates = noise_model. Fexecução em Co # Backend overview import qiskit.tools.jupyte provider: [1], ('x', [1], ('	noise import NoiseModel 'rom_backend (backend_device) 'rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' [2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', [0]), ('id', [1]), ('id', [6]), ('id', [1]), ('sx', [1]), ('sx', [1]), ('sx', [1]), ('sx', [1]), ('x', [1]), ('x', [1]), ('x', [1]), ('x', [1]), ('x', [1]), ('cx', [1], 1'cx', [8, 9]), ('cx', [1, 1]), ('cx', [1], 1'cx', [8, 9]), ('cx', [4, 5]), ('cx', [1, 2]), ('cx', [1], 1'cx', [1]), ('measure', [1]), ('measure', [1]), ('measure', [1]), ('measure', [1]), ('measure', [1]), ('measure', [1]) basis_gates Grover, backend, se_model_mois_model, [1], (1], (1], (1], (1], (1], (1], (1], (11, 12, 13, 14 id', [2]), ('id', [3]), ('id',)), ('id', [9]), ('sx', [1]), ('s, , [6]), ('sx', [1]), ('sx', , [2]), ('sx', [1]), ('sx', , [1]), ('sx', [1]), ('xx', , [4]), ('xx', [1]), ('cx', [1], , ('xx', [14], 13]), ('cx', [1], 10]), ('cx', [1], 10]), ('cx', [1], 10]), ('cx', [1], ('cx', [1], 1]), ('cx', [1], ('cx', [1], 4]), ('yx', [1], ('1]), ('xx', [1], 4]), ('yx', [1], ('1]), ('measure', [2]), ('measure', [3]), ('measure', [2]), ('measure',
In [18]: In [19]: In [20]: In [21]: In [22]: Out [22]: Out [23]:	coupling map = backend dev from qiskit.providers.aer. noise model = NoiseModel.f print (noise model) NoiseModel: Basis qates: ['cx', 'id', Instructions with noise: [0, 1, Specific qubit errors: [(d1), ('id', [1]), ('id', [1]), ('x', [1], ['cx', [t,x',	noise import NoiseModel 'rom_backend(backend_device) 'rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', [0]), ('id', [1]), ('id', [8] ('id', [13]), ('id', [11]), ('sx', [10]), ('sx', [11]), ('sx', [21]), ('x', [21]), ('x', [21]), ('x', [21]), ('x', [21]), ('cx', [3, 3]), ('cx', [3, 1]), ('cx', [3, 3]), ('cx', [3, 1]), ('cx', [13]), ('cx', [13]), ('cx', [11], 1] (x', [8, 9]), ('cx', [11], 1] (x', [8, 9]), ('cx', [12]), ('x', [10]), ('cx', [12]), ('measure', [11]) (y', [10, 11]), ('cx', [12]), ('measure', [11]) (measure', [6]), ('measure', [6]), ('me	11, 12, 13, 14 id', [2]), ('id', [3]), ('id',)), ('id', [9]), ('sx', [1]), ('s, , [6]), ('sx', [1]), ('sx', , [2]), ('sx', [1]), ('sx', , [1]), ('sx', [1]), ('xx', , [4]), ('xx', [1]), ('cx', [1], , ('xx', [14], 13]), ('cx', [1], 10]), ('cx', [1], 10]), ('cx', [1], 10]), ('cx', [1], ('cx', [1], 1]), ('cx', [1], ('cx', [1], 4]), ('yx', [1], ('1]), ('xx', [1], 4]), ('yx', [1], ('1]), ('measure', [2]), ('measure', [3]), ('measure', [2]), ('measure',
In [18]: In [19]: In [20]: In [21]: In [22]: Out [22]: In [26]: In [26]: In [27]:	*qiskit_job_watcher coupling_map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis gates: ['cx', 'id', Instructions with noise: Qubits with noise: [0, 1, Specific qubit errors; ('id', [11]), ('s'd, [12]), ('s', [3]), ('sx', [3], ('xx', [4]), ('xx', [6], 5]), ('cx', [12, 11]), ('cx', [12, 11]), ('cx', [13, 1]), ('cx', [13, 1]), ('cx', [2, 3]), ('measure', [41]) basis_gates = noise_model. result_noise = execute(qc_noise_model_noise_	"rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 7, 11], ('id', 7, 12], ('id', 12], ('id', 12]), ('id', 13]), ('id', [1]), ('sx', [1]), ('cx', [1]), ('cx', [1]), ('cx', [1]), ('cx', [1], 1], ('measure', [1]), ('measu	('ex')
In [18]: In [19]: In [20]: In [21]: In [22]: Out [22]: Out [23]: In [24]:	*qiskit_job_watcher coupling_map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis gates: ['cx', 'id', Instructions with noise: Qubits with noise: [0, 1, Specific qubit errors; ('id', [11]), ('s'd, [12]), ('s', [3]), ('sx', [3], ('xx', [4]), ('xx', [6], 5]), ('cx', [12, 11]), ('cx', [12, 11]), ('cx', [13, 1]), ('cx', [13, 1]), ('cx', [2, 3]), ('measure', [41]) basis_gates = noise_model. result_noise = execute(qc_noise_model_noise_	"rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', (0)), ('id', [1]), ('2, (14)), ('id', [1]), ('sx', [1]), ('measure', [1])	11, 12, 13, 14 id', [2]), ('id', [3]), ('id',)), ('id', [9]), ('sx', [1]), ('s, , [6]), ('sx', [1]), ('sx', , [2]), ('sx', [1]), ('sx', , [1]), ('sx', [1]), ('xx', , [4]), ('xx', [1]), ('cx', [1], , ('xx', [14], 13]), ('cx', [1], 10]), ('cx', [1], 10]), ('cx', [1], 10]), ('cx', [1], ('cx', [1], 1]), ('cx', [1], ('cx', [1], 4]), ('yx', [1], ('1]), ('xx', [1], 4]), ('yx', [1], ('1]), ('measure', [2]), ('measure', [3]), ('measure', [2]), ('measure',
In [18]: In [19]: In [20]: In [21]: In [22]: In [23]: In [24]: In [26]: In [27]:	# Backend overview import giskit.backend overview import giskit.tools.monitor backend_overview import giskit.tools.monitor backend_overview import giskit.tools.monitor backend_overview () # Basis gates: ['cx', 'id', 'id', ('id', [11]), ('id', [13]), ('id', [12]), ('s', [13]), ('sx', [14]), ('sx', [14]), ('x', [16]), ('x', [17]), ('x', [18]), ('x', [17]), ('x', [18]), ('cx', [14]), ('cx', [15]), ('cx', [16]), ('cx', [1	"rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', (0)), ('id', [1]), ('2, (14)), ('id', [1]), ('sx', [1]), ('measure', [1])	(cx'] (11, 12, 13, 14) (12', (21), ('id', [3]), ('id', [10]), ('('a', [10]), ('ax', [1]), ('ax', [12]), ('ax', [13]), ('ax', [12]), ('x', [13]), ('x', [13]), ('x', [13]), ('x', [14, 10]), ('x', [14, 13]), ('x', [12, 2]), ('x', [2, 12]), ('x', [2, 12]), ('x', [3, 4]), ('cx', [4, 3]), ('cx', [3, 4]), ('cx', [4, 3]), ('cx', [3, 4]), ('cx', [4, 3]), ('cx', [3, 4]), ('cx', [4, 3]), ('ax', [2, 1]), ('measure', [3]), ('measure',
<pre>In [18]: In [19]: In [20]: In [21]: In [22]: In [23]: In [24]: In [26]: In [27]:</pre>	## Coupling map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis gates: ['cx', 'id', Instructions with noise: (0, 1, specific qubit errors: (id'), ('id', [11]), ('sd', [12]), ('sx', [13]), ('sx', [14]), ('sx', [1]), ('sx', [1]), ('sx', [1]), ('sx', [1]), ('x', [1]), ('x', [1]), ('x', [1]), ('x', [1]), ('x', [1]), ('x', [1], ('x', [1], 1]), ('cx', [2, 3]), ('cx', [3, 8]), ('cx', [2, 3]), ('cx',	rom_backend(backend_device) 'rz', 'sx', 'x'] ('sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('id', [01), ('id', [1]), ('id', [8]), ('id', [7]), ('id', [8]), ('sx', [1]), ('sx', [1]), ('sx', [1]), ('sx', [1]), ('sx', [1]), ('cx', [1], [1], ['cx', [1], [1], ('cx', [1], [1]), ('anasure', [1]), ('measure', [1]), ('anasure', [1])	'cx'
In [18]: In [19]: In [20]: In [21]: In [22]: In [23]: In [24]: In [26]: In [27]:	### Coupling map = backend dev from qiskit.providers.aer. Doise model = NoiseModel.f print(noise model) NoiseModel: Basis gates: ['cx', 'id', Basis gates: ['cx', 'id', Sapeific qubit errors: [((41), ('id', [51), ('ix', [3]), ('sx', [3]), ('sx', [4]), ('x', [4], ('x', [2, 3]), ('cx', [2, 3]), ('cx', [2, 3]), ('x', [2, 4]), ('measure', [4])] Basis gates = noise model. #### Dense Coding O.32	rom_backend(backend_device) 'rz', 'sx', 'x'] ('sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('sx', [01), ('id', [1]), ('sd', [6]), ('id', [7]), ('id', [8]), ('sx', [1]), ('sx',	'cx'
<pre>In [18]: In [19]: In [20]: In [21]: In [22]: Out [22]: Out [22]: In [24]: In [24]: In [28]:</pre>	coupling_map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.f print(noise_model) NoiseModel: Basis_gates: ('cx', 'id',	rom_backend(backend_device) 'rz', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 5, 7, 8, 9, 10 (id', (1)), ('id', [1]), ('id', [1]), ('id', [1]), ('sx', [(ax')
In [20]: In [21]: In [22]: Out [22]: Out [22]: In [24]: In [26]: In [27]: In [27]:	coupling map = backend_dev from qiskit.providers.aer. noise_model = NoiseModel.fprint(noise_model) NoiseModel: Basis gates: [(w.,'iii, 10)] Bill, ('s', [9]), ('s', [1]) Bill, ('s', [9]), ('s', [1]) Bill, ('s', [9]), ('s', [1]) Bill, ('x', [1]), ('x', [1]) Bill, ('x', [1]), ('x', [1]) Bill, ('x', [1]), ('x', [1]) Basis gates Roise Ro	noise import NoiseModel rom_backend(backend_device) 'rz', 'sx', 'x'] ['sx', 'sx', 'x'] ['sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 9, 10 ('d', [0]), ('sx', [1]), ('ta', [1]), ('measure', [1]), ('meas	(a)
In [20]: In [21]: In [22]: Out [22]: In [24]: In [26]: In [26]: In [27]: In [28]:	### Square	noise import NoiseModel rom_backend(backend_device) 'rz', 'sx', 'x'] ('sx', 'measure', 'x', 'id' 2, 3, 4, 5, 6, 7, 8, 9, 10 ('la', [13]), ('la', [14]), ('sx', [5]), ('sx', [14]), ('sx', [13]), ('sx', [14]), ('sx', [13]), ('sx', [14]), ('sx', [14]), ('sx', [14]), ('sx', [14]), ('sx', [15]), ('cx', [15]), ('sx', [16]), ('	### Comparison of the contract
In [23]: In [23]: In [24]: In [24]: In [27]: In [27]: In [27]: In [27]: In [27]:	### Special State of the complete of the compl	noise import NoiseModel rom_backend (backend_device) 'rz', 'sx', 'x', 'i' 'zx', 'sx', 'x', 'i' 'z, ', 'sx', 'x', 'i' 'z, ', 's, 's, 's, 's, 'id' 'z, ', 's, 's, 's, 's, 's, 'id' 'z, ', 's, 's, 's, 's, 's, 's, 's, 's, 's,	### Comparison of the contract
<pre>In [18]: In [19]: In [20]: In [21]: In [22]: Out [22]: Out [22]: In [24]: In [26]: In [26]: In [27]: In [27]:</pre>	### Sacked overview ### Sacked overview ### Dense Coding #### Dense Coding ##### Dense Coding ##### Dense Coding ##### Dense Coding ##### Dense Coding ###### Dense Coding ######## Dense Coding ###################################	noise import NoiseModel 'rom_backerd(backend_device) 'rr', 'sx', 'x'] 'rsx', 'measure' 's', 'id' 'ss', '5, 7, 7, 910 (10), ('10', '11', '16', '11', '16', '11', '16', '11', '16', '11', '16', '11', '16', '11', '16', '11	1. 1. 1. 1. 1. 1. 1. 1.
In [23]: In [23]: In [24]: In [24]: In [27]: In [27]: In [27]: In [27]: In [27]:	### Sackend overview import giskit.tools.moitor basis gates ('cx', 'id',	noise import NoiseModel 'rom_backerd(backend_device) 'rr', 'sx', 'x'] 'rsx', 'measure' 's', 'id' 'ss', '5, 7, 7, 910 (10), ('10', '11', '16', '11', '16', '11', '16', '11', '16', '11', '16', '11', '16', '11', '16', '11	tico 66: Credentials are already in u elimq_belam Num. (Qubits: 5 Pending Jobe: 1 Least Luy: True Chay. (12); (*m*, [13]); (*m*, [12]); (*m*, [13]); (*m*, [13]); (*m*, [13]); (*m*, [13]); (*m*, [12]); (*m*, [13]); (*m*, [13]
<pre>In [18]: In [19]: In [20]: In [21]: In [22]: Out [22]: Out [23]: In [24]: In [24]: In [27]: In [27]: In [27]: In [27]: In [27]: In [27]: In [27]: In [27]: In [27]: In [28]: In</pre>	### Coupling map = backend_dev from qiskt.providers.aer. moise_model = MoiseModel.f print(noise_model) NoiseModel:	noise import NoiseModel rom_backend (backend_device) 'tra', 'sa', 'a', 'id', 'g', 'g', 'g', 'g', 'g', 'g', 'g', '	tico 66: Credentials are already in u elimq_belam Num. (Qubits: 5 Pending Jobe: 1 Least Luy: True Chay. (12); (*m*, [13]); (*m*, [12]); (*m*, [13]); (*m*, [13]); (*m*, [13]); (*m*, [13]); (*m*, [12]); (*m*, [13]); (*m*, [13]
<pre>In [18]: In [19]: In [20]: In [21]: In [22]: Out [22]: Out [23]: In [24]: In [24]: In [27]: In [27]: In [27]: In [27]: In [27]: In [27]: In [27]: In [27]: In [27]: In [28]: In</pre>	### Special Company of the service o	Trit Sax' X' 1 1 1 1 1 1 1 1 1	'car' 13, 140 13)
<pre>In [18]: In [19]: In [20]: In [21]: In [22]: Out [22]: Out [23]: In [24]: In [24]: In [30]: In [31]: In [31]: In [31]: In [31]: In [33]:</pre>	### State Section	moise import NoiseModel "ray, 'ex', 'x'] "ray, 'ex', 'x'] "ray, 'ex', 'x'] "(sx', 'measure, 'x', 'id' "(s1), ('id', (11), 'id',	'car' 13, 140 13)
In [18]: In [19]: In [20]: In [21]: In [22]: In [23]: In [23]: In [24]: In [26]: In [26]: In [31]: In [31]: In [31]: In [33]: In [33]:	Squart job_watcher coupling_map = backend_dev from giskit.providers.aer. noise_model = KoiseModel.f printincise_model = KoiseModel.f (1) 1 (14 1	romise import NoiseModel rom_backend (backend_device) 'rr', 'sx', 'x'] ('rx', 'sx', 'x'] ('sx', 'sy', 's', 'id' 'rs', ('st', 's'), '(s', 's', 'id' 'rs', ('st', 's'), '(s', 's'), 'id' 'rs', ('st', 's'), '(s', 's'), 's', 's', 's', 's', 's', 's',	'car' 13, 140 13)
<pre>In [18]: In [20]: In [20]: In [21]: In [22]: Out [22]: In [24]: In [24]: In [30]: In [31]: In [31]: In [31]: In [31]:</pre>	deskit_jeb_watches coupling_map = backend_dev from giskt.providers.aer. noise_model = NoiseNodel.f printinoise_model NoiseNodel.f Dasis_gates: ('ex', 'id', Instructions with noise: ('o, 'id', I'is', ('11), ('id', 'id', 'id', I'is', ('11), ('id', 'id', 'id', I'is', ('id', 'id', 'id', 'id', I'is', ('id', 'i	rom_backend (backend_device) /**** /**** /**** /*** /*** /*** /**	'ma'
In [18]: In [19]: In [20]: In [21]: In [22]: In [23]: In [23]: In [24]: In [26]: In [26]: In [31]: In [31]: In [31]: In [33]: In [33]:	### Square Square Square ### Square ### Square ### Square Square ### Square ### Square S	moise import boiseModel rom_backend(backend_device) 'rr', 'rs', 's', 's', 's', 's', 's', 's'	tico 66: Credentials are already in u ed. (12); "issue (13); "issue (
In [18]: In [19]: In [20]: In [21]: In [22]: In [23]: In [23]: In [24]: In [26]: In [26]: In [31]: In [31]: In [31]: In [33]: In [33]:	### Square Square Square ### Square ### Square ### Square Square ### Square ### Square S	moise import boiseModel rom_backend (biseModel rom_rom_backend (biseModel rom_rom_rom_backend (biseModel rom_rom_rom_backend (biseModel rom_rom_rom_backend (biseModel rom_rom_rom_rom_backend (biseModel rom_rom_rom_rom_rom_rom_rom_rom_rom_rom_	## A control of the c