Теоретические модели вычислений ДЗ №3: Машины Тьюринга и квантовые вычисления

А-13б-19 Сергей Тимченко

17 мая 2022

1 Машины Тьюринга

Все решения представлены в папке TMV3.

1.1 Операции с числами

Реализуйте машины Тьюринга, которые позволяют выполнять следующие операции:

1. Сложение двух унарных чисел (1 балл)

Решение представлено в файле 2.1 1.yaml.

В моей реализации на вход подается два унарных числа, разделенных знаком +. Результатом является унарное число, являющееся суммой двух данным. К примеру:

Вход	Результат
11+111	11111
$1{+}1$	11
+1	1
1+	1

Можно заметить, что в данном случае сделал также реализацию, если отсутствует (равно 0) 1 или 2 число. Алгоритм работы прост: находим единицу в правом слагаемом, заменяем ее на 0 и переносим влево. В конце необходимо "почистить" строку (убрать + и 0). Учитываю уникальный случай, если отсутствует первое число.

2. Умножение унарных чисел (1 балл)

Решение представлено в файле 2.1 2.yaml.

В моей реализации на вход подается два унарных числа, разделенных знаком *. Результатом является унарное число, являющееся произведение двух данным. К примеру:

Вход	Результат
11*111	111111
1*1	1
*1	
1*	

Можно заметить, что в данном случае сделал также реализацию, если отсутствует (равно 0) 1 или 2 число. В реализации использую два дополнительных знака. Алгоритм работы:

- Находим *, путем прохода в правую сторону.
- Находим первую 1 правого множителя, заменяем ее на 0, уходим влево, пока не найдем первую 1 левого множителя.
- Заменяем найденную 1 на 0 и уходим в левый край, ставим дополнительный знак х.
- Снова находим * и повторяем копирование остальных единиц (если есть) левого множителя (ставим дополнительный знак х слева, заменяем 1 на 0). Так производим, пока не встретится х при поиске 1.

- Повторяем предыдущие три шага, пока не уйдем за границы второго множителя.
- Производим "чистку" и замену: удаляем 0 и *, вместо х пишем 1.

находим единицу в правом числе, заменяем ее на 0 и переносим влево. В конце необходимо "почистить" строку (убрать + и 0). Учитываю уникальный случай, если отсутствует первое число.

1.2 Операции с языками и символами

Реализуйте машины Тьюринга, которые позволяют выполнять следующие операции:

1. Принадлежность к языку $L = \{0^n 1^n 2^n\}, n > 0 \ (0.5 \ балла)$

Решение представлено в файле 2.2 1.yaml.

На вход подается строка из 0, 1, 2. Результат работы 1 или 0, если слово принадлежит или не принадлежит языку L соответственно.

Алгоритм:

- Нулевой шаг: учитываем пустое слово. В данном случае сразу даем ответ 1.
- На данном этапе мы точно имеем не пустое слово. Заменяем все первые вхождения 0, 1 и 2 на дополнительный знак х. Если не встречается хотя бы одного знака, то очищаем строку, даем ответ 0.
- После замены находим первое вхождение 0 и повторяем предыдущий шаг, пока не закончатся 0.
- Если в строке остаются 1 или 2, очищаем и выводим ответ 0. В ином случае очищаем строку и выводим ответ 1.

Вкратце представлю работу алгоритма:

Условный шаг	Значение в строке	Условный шаг	Значение в строке
1	001122	1	0011122
2	x0x1x2	2	x0x11x2
3	xxxxxx	3	xxx1xxx
4	1	4	0

Примечание: не стал расписывать конкретную реализацию шага 2. Постарался оптимизировать работу машины.

2. Проверка соблюдения правильности скобок в строке (минимум 3 вида скобок) (0.5 балла)

Решение представлено в файле 2.2 2.yaml.

На вход подается строка из $(,), \{,\}, [,]$. Результат работы 1 или 0, если слово является правильной скобочной последовательностью и нет соответственно. Алгоритм решения:

- Пустое слово правильная скобочная последовательность без скобок. Даем ответ 1.
- На данном этапе мы точно имеем не пустое слово. Проходим вправо и игнорируем любые открывающиеся скобки. Делаем до тех пор, пока не встретим первую любую закрывающуюся скобку, заменяем ее на дополнительный знак х, уходим влево.
- Так это первая закрывающаяся скобки, то мы в любом случае получим на этом этапе открывающуюся скобку. Если она соответствует найденной закрывающейся, то заменяем на доп. знак х и повторяем предыдущий шаг. Если не соответствует, то очищаем строку выводим ответ 0.
- При повторении предыдущего шага пропускаем дополнительный знак. Делаем до тех пор, пока не закончатся закрывающиеся скобки.
- Если в строке остались только дополнительные знаки, то очищаем строку и выводим ответ 1, в ином случае очищаем и выводим ответ 0.

Вкратце представлю работу алгоритма:

Условный шаг	Значение в строке		
1	{{}([])}		
2	{{x([])}}	Условный шаг	Значение в строке
3	$\{xx([])\}$	1 TODIIBII III III	(O/DI)
4	$\{xx([x)]\}$	1	
5	$\{xx(xx)\}$	2	{{x([}])}
6	{xx(xxx}	3	$\{xx([\}])\}$
7	{xxxxxx}	4	$\{xx([x])\}$
0	, ,	5	0
0	{xxxxxx		•
9	XXXXXXX		
10	1		

Примечание: опять же, в данном случае не досканально описал алгоритм, а только тезисно.

3. Поиск минимального по длине слова в строке (слова состоят из символов 1 и 0 и разделены пробелом) (1 балл)

Решение представлено в файле 2.2 3.yaml.

На вход строка из слов, состоящих из 1 и 0, разделенных пробелом. Результат работы - самое короткое слово в строке. Алгоритм решения построен на создании унарного числа, которое соответствует длине минимального слова в строке.

- Если введена пустая строка это и является ответом.
- Берем первое слово за основу минимального. Создаем для него слева унарное число, соответствующее длине этого слова. Заменяем 1 этого слова на доп. знак x, 0 на y. Также дополнительно использую знак p слева от длины слова для стопа.
- Если в строке больше нет слов, то удаляем длину и заменяем х на 1, у на 0. Иначе на следующий шаг
- В строке остались слова. Сопоставляем каждый символ нового слова с длиной минимального, заменяем 1 на z, 0 на w (для этого заменяется в длине 1 на l). Если выходит так, что достигается знак p, то слово является длинее минимального. Оно очищается, длина восстанавливается (с l на 1). Если слово по длине такое же или меньше, то это означает, что мы достигнем пробела после него. Получается это за счет того, что мы смотрим по одному символу из слова и постоянно "бегаем"влево-вправо от слова к длине. В данном случае мы удаляем минимальное слово и его длину, заменяем z на x, w на y, а также снова формируем длину нового минимального слова.
- Если в строке больше нет слов, то очищаем длину и заменяем х на 1, у на 0.

Примечание: аналогично описал тезисно весь алгоритм. Постарался оптимизировать процесс. Например, если нашел новое минимальное слово и больше после него нет слов, то сразу же удалял старое минимальное и его длину, z заменял на 1, а w на 0. Вкратце представлю работу алгоритма:

	Условный шаг	Значение в строке	Условный шаг	Значение в строке
_	1	101 11 101	1	101 11 10
	2	p111 xyx 11 101	2	p111 xyx 11 10
	3	p1ll xyx zz 101	3	p1ll xyx zz 10
	4	p11 xx 101	4	p11 xx 10
	5	pll xx zw1	5	pll xx zw
	6	p11 xx	6	ZW
	7	11	7	11

Во втором примере как раз показал работу двух пунктов: встречено слово минимальной длины и заменено и оно же является последним.

2 Квантовые вычисления

В данном задании представил решения на Python с использованием пакета qiskit. В данном отчете представляю кусочки кода основных процедур и функций, а для наглядности предлагаю перейти в ноутбук: Quantum task.ipynb.

2.1 Генерация суперпозиций 1 (1 балл)

Дано N кубитов ($1 \le N \le 8$) в нулевом состоянии $|0...0\rangle$. Также дана некоторая последовательность битов, которое задаёт ненулевое базисное состояние размера N. Задача получить суперпозицию нулевого состояния и заданного.

$$|S\rangle = \frac{1}{\sqrt{2}}(|0\dots0\rangle + |\psi\rangle)$$

То есть требуется реализовать операцию, которая принимает на вход:

- 1. Массив кубитов q_s
- 2. Массив битов bits описывающих некоторое состояние $|\psi\rangle$. Это массив имеет тот же самый размер, что и q_s . Первый элемент этого массива равен 1.

Код для основной процедуры:

```
def Solve_First(q, psi):
    global circuit
    circuit.h(0)  # Hadamar for first qubit
    circuit.barrier()
    for i in range (1, len(psi)):
        if psi[i]:  # for each 1 qubit in psi making CNOT
        circuit.cx(q[0], q[i])
```

2.2 Различение состояний 1 (1 балл)

Дано N кубитов ($1 \le N \le 8$), которые могут быть в одном из двух состояний:

$$|GHZ\rangle = \frac{1}{\sqrt{2}}(|0\dots0\rangle + |1\dots1\rangle)$$

$$|W\rangle = \frac{1}{\sqrt{N}}(|10\dots00\rangle + |01\dots00\rangle + \dots + |00\dots01\rangle)$$

Требуется выполнить необходимые преобразования, чтобы точно различить эти два состояния. Возвращать 0, если первое состояние и 1, если второе.

Код основной функции:

```
def Solve_Second(q):
      str = q.measure()[0]
                              # measuring qubits
      if (len(str) == 1):
                              # exception claimed
          if q == One:
              return 1
              return 0
      i = 0
      for qubit in str:
          if qubit == '1':
              i+=1
12
      if (i == 1):
13
14
          return 1
15
          return 0
      return i
```

2.3 Различение состояний 2 (2 балла)

Дано 2 кубита, которые могут быть в одном из четырех состояний:

$$\begin{split} |S_0\rangle &= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \\ |S_1\rangle &= \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle) \\ |S_2\rangle &= \frac{1}{2}(|00\rangle + |01\rangle - |10\rangle - |11\rangle) \\ |S_3\rangle &= \frac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle) \end{split}$$

Требуется выполнить необходимые преобразования, чтобы точно различить эти четыре состояния. Возвращать требуется индекс состояния (от 0 до 3).

Код основной функции:

```
def Solve_Third(q):
       global circuit
      circuit.initialize(q)
                                       # putting our statevector into circut
      circuit.h(range(2))
                                       # making Hadamar for both qubits
      state = Statevector(circuit)
                                       # measuring qubits
      res = state.measure()[0]
      if res[0] == '0':
    if res[1] == '0':
               return 0
           else:
12
               return 1
      else:
           if res[1] == '0':
              return 2
15
16
           else:
               return 3
```

2.4 Написание оракула 1 (2 балла)

Требуется реализовать квантовый оракул на N кубитах $(1 \le N \le 8)$, который реализует следующую функцию: $f(\boldsymbol{x}) = (\boldsymbol{b}\boldsymbol{x}) \mod 2$, где $\boldsymbol{b} \in \{0,1\}^N$ вектор битов и \boldsymbol{x} вектор кубитов. Выход функции записать в кубит \boldsymbol{y} . Количество кубитов N $(1 \le N \le 8)$.

Заготовка для кода: