بسم الله الرحمن الرحیم

پروژه تبدیل DFA به کد Python و C++

استاد راهنما: دکتر پارسا

ارایه دهنده: محمد حسین حسنی (99521199)

**توضیحات اولیه در مورد DFA**

**DFA متناظر با یک زبان، یک ماشین مجرد است که برای تشخیص و تولید رشته های زبان مشخص شده استفاده می شود. در زبان برنامه نویسی، DFA برای پردازش و تفسیر کدها به کار می رود و به عنوان یک مرحله از فرایند کامپایلر استفاده می شود.**

**DFA شامل یک مجموعه وضعیت ها است که از یک وضعیت شروع (یا استیت استارت) شروع می شود و یک مجموعه از وضعیت های پایانی (یا استیت های فاینال) دارد. هر وضعیت در ماشین با یک نام شناخته می شود و هر یالی که از یک وضعیت به وضعیت دیگری وجود دارد، با یک علامت مشخص شده است.**

**DFA با خواندن ورودی و حرکت به وضعیت های مختلف در داخل ماشین، تلاش می کند تا رشته ورودی را تولید کند. اگر رشته ورودی در زبان قابل قبول توسط DFA باشد، ماشین در یکی از وضعیت های پایانی قرار می گیرد و در غیر این صورت، به وضعیتی که هیچ یالی برای حرکت به آن وجود ندارد، می رود.**

**در کامپایلر، DFA برای تشخیص الفبا و تشخیص واژه ها (Tokens) در برنامه های مبتنی بر زبان، استفاده می شود. به عنوان مثال، یک DFA می تواند به منظور تشخیص واژه های کلیدی (Keyword) در زبان برنامه نویسی C مورد استفاده قرار گیرد. با این کار، کامپایلر می تواند بخشی از کار زمان اجرا را پیش از آنکه برنامه اجرا شود، انجام دهد و به شکل موثرتری کد را تفسیر کند.**

**تبدیل DFA به کد زبان های برنامه نویسی**

**موضوع این ارایه تبدیل یک DFA به کد Python و C++ می باشد. این تبدیل با استفاده از زبان پایتون انجام شده است. در ابتدا کلاس DFA را شرح می دهیم:**

class DFA:

    def \_\_init\_\_(self, alphabet, states, transitions, start\_state, accepting\_states):

        self.alphabet = alphabet

        self.states = states

        self.numeric\_states = [i for i in range(len(states))]

        self.transitions = transitions

        self.transitions\_with\_numeric\_states = {}

        for (state, symbol), next\_state in self.transitions.items():

            self.transitions\_with\_numeric\_states[(self.states.index(state), symbol)] = self.states.index(next\_state)

        self.start\_state = start\_state

        self.numeric\_start\_state = self.states.index(start\_state)

        self.accepting\_states = accepting\_states

        self.numeric\_accepting\_states = [self.states.index(state) for state in accepting\_states]

        self.dfa\_code = self.generate\_dfa\_code()

        self.dfa\_code\_cpp = self.generate\_dfa\_code\_cpp()

        self.dfa\_graph = self.generate\_dfa\_graph()

**در این کلاس، تمام اطلاعات مربوط به یک DFA شامل زبان الفبا، حالت ها، تابع انتقال، حالت شروع و حالت های پذیرفته شده به صورت ورودی داده می شود و سپس این اطلاعات در داخل کلاس ذخیره می شوند.**

**علاوه بر ذخیره اطلاعات DFA، این کلاس سه method نیز ارائه می دهد:**

* **generate\_dfa\_code : کد Python را برای یک DFA تولید می کند.**
* **generate\_dfa\_code\_cpp : کد C++ را برای یک DFA تولید می کند.**
* **generate\_dfa\_graph : نمودار یک DFA را تولید می کند.**

**توضیحات مربوط به متد generate\_dfa\_code:**

def generate\_dfa\_code(self):

    code = [

        f"class DFA:",

        f"    def \_\_init\_\_(self):",

        f"        self.alphabet = {self.alphabet}",

        f"        self.states = {self.states}",

        f"        self.transitions = {self.transitions}",

        f"        self.start\_state = '{self.start\_state}'",

        f"        self.accepting\_states = {self.accepting\_states}",

        f"",

        f"    def run(self, input\_string):",

        f"        current\_state = self.start\_state",

        f"        for symbol in input\_string:",

        f"            current\_state = self.transitions.get((current\_state, symbol))",

        f"            if current\_state is None:",

        f"                return False",

        f"        return current\_state in self.accepting\_states",

        f"",

        f"if \_\_name\_\_ == '\_\_main\_\_':",

        f"    dfa = DFA()",

        f"    print(dfa.run(input('Enter a string: ')))"

    ]

    code = "\n".join(code)

    with open("dfa.py", "w") as f:

        f.write(code)

    return code

**این تابع، یک کد Python برای یک DFA تولید می کند. کد تولید شده شامل تعریف یک کلاس به نام DFA است که دارای اعضای زیر است:**

* **:alphabet زبان الفبایی که DFA برای آن طراحی شده است**
* **:states مجموعه حالت های DFA**
* **transitions : تابع انتقال بین حالت ها با ورودی های مختلف**
* **start\_state : حالت شروع DFA**
* **accepting\_states : مجموعه حالت های پذیرفته شده توسط DFA**

**این کد تولید شده همچنین شامل یک تابع به نام run است که با گرفتن یک رشته ورودی، حالت نهایی DFA را با استفاده از تابع انتقال به دست می آورد و بررسی می کند که آیا حالت نهایی در مجموعه حالت های پذیرفته شده است یا خیر.**

**در انتها، در صورتی که کد به صورت مستقیم اجرا شود، یک DFA به نام dfa ایجاد شده و پس از گرفتن یک رشته ورودی، تابع run روی آن فراخوانی می شود. سپس نتیجه بررسی به صورت یک مقدار بولین True یا False چاپ می شود.**

**توضیحات مربوط به متد generate\_dfa\_code\_cpp:**

**این تابع یک کد C++ برای اجرای ماشین مجازی DFA تولید می‌کند. این کد به کاربر اجازه می‌دهد تا یک رشته ورودی را وارد کند، سپس رشته را در ماشین مجازی اجرا کند و بررسی کند که آیا رشته ورودی به عنوان ورودی معتبر شناخته شده توسط ماشین قابل قبول است یا خیر. کد تولید شده شامل یک switch-case برای پیاده سازی انتقالات از حالت کنونی به حالت بعدی با استفاده از جفت شماره حالت-نماد، و همچنین یک بخش بررسی برای تشخیص اینکه آیا ماشین در یکی از حالت‌های قبول قرار دارد یا خیر، است. در پایان، کد C++ تولید شده در یک فایل به نام dfa.cpp ذخیره می‌شود.**

**توضیحات مربوط به متد generate\_dfa\_graph:**

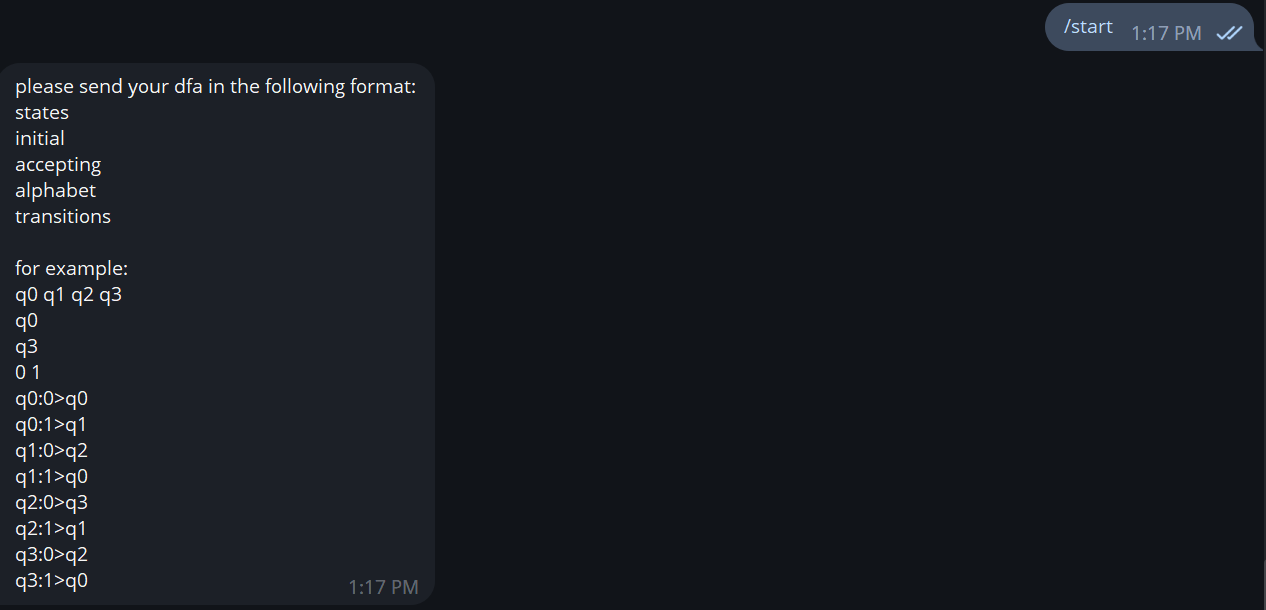
**این تابع یک گراف تصویری از دی‌اف‌ای را که به عنوان ورودی به کلاس داده شده است، تولید می‌کند. گراف تولید شده با استفاده از کتابخانه graphviz ایجاد می‌شود.**

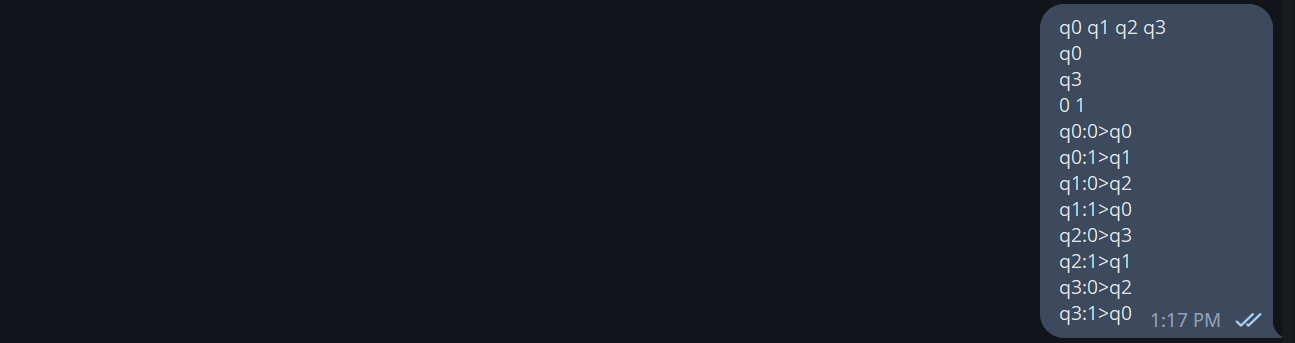
**در این گراف، هر حالت به عنوان یک گره نمایش داده می‌شود. حالت‌های پذیرفتنی با یک دایره دوتایی و حالت‌های غیر پذیرفتنی با یک دایره نمایش داده می‌شود. همچنین، یال‌ها نشان‌دهنده ی ترانزیشن‌های این دی‌اف‌ای هستند که با برچسبی مشخص شده‌اند که نشان دهنده ی نمادی از الفبای دی‌اف‌ای است.**

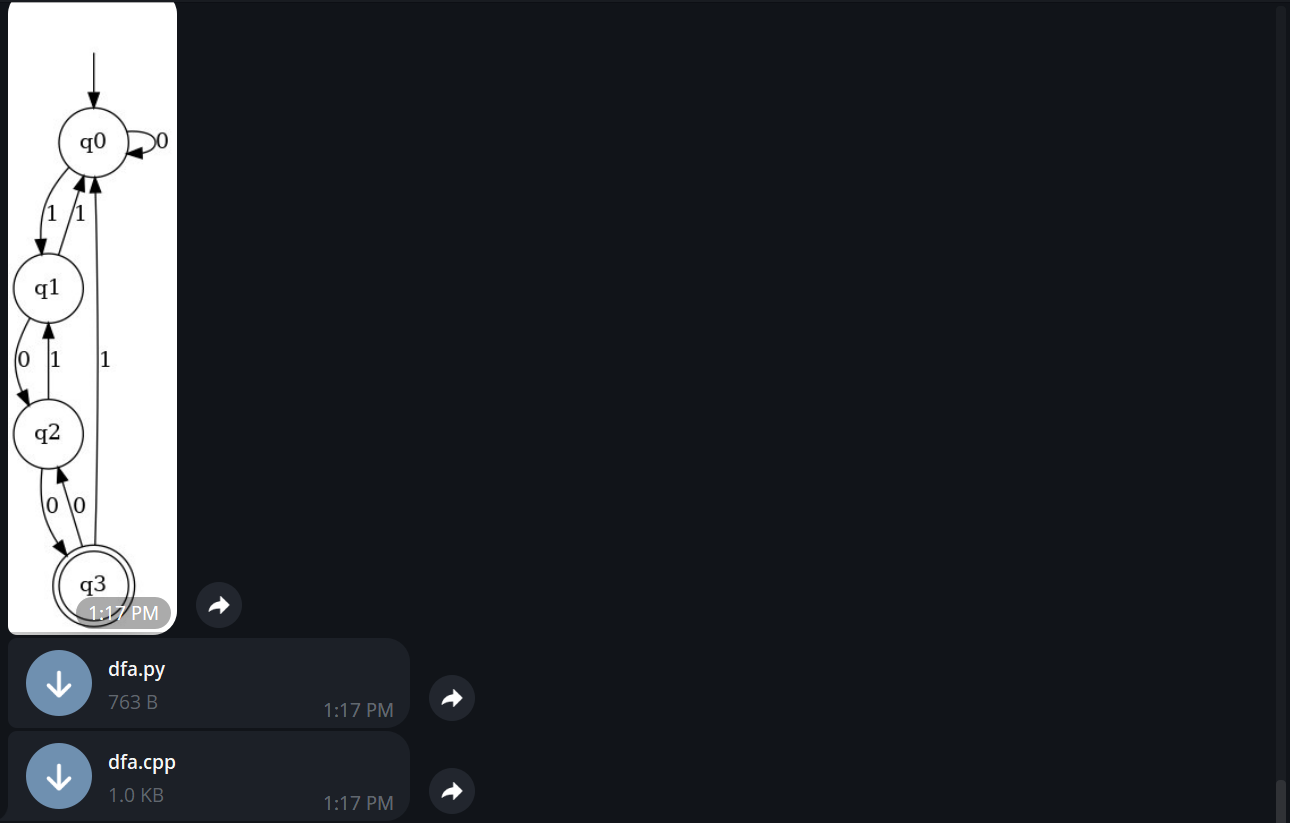
**در نهایت، این گراف در قالب یک فایل تصویری با پسوند png ذخیره می‌شود و نام آن به عنوان خروجی تابع بازگردانده می‌شود.**

**توضیحات مربوط به ربات تلگرام (**https://t.me/DFA2Codebot**):**

**در این ربات کاربر با فرستادن کامند start می تواند DFA خود را با فرمت مشخص شده ارسال کند. سپس ربات گراف و کد Python و C++ مربوط به آن DFA را برای کاربر ارسال می کند. همچنین با استفاده از کامند test می توان رشته ای برای ربات ارسال کرد و ربات آن رشته را در آخرین DFA ارسال شده توسط آن کاربر تست می کند و یکی از نتایج accepted یا rejected را ارسال می کند.**

****

****

****