به نام خدا

نظریه زبان‌ها و ماشین‌ها- بهار 1403

پاسخ تمرین شماره 9

دستیار آموزشی این مجموعه: شهنام فیضیان
shahnamecfman@gmail.com

تاریخ تحویل : 6 خرداد (صفحه درس)

1. الف) پشته‌های اتوماتون پشته‌ای را A و B می‌نامیم و فرض می‌کنیم A عناصر سمت چپ head نوار را نگهداری می‌کند و B عناصر سمت راست آن را. عنصری که head به آن اشاره می‌کند را نیز عنصر اول پشته B در نظر می‌گیریم. حال برای شبیه سازی نوشتن و حرکت به سمت راست اول از پشته B مقدار بالایی آنرا pop می‌کنیم و سپس مقداری که باید نوشته می‌شد را به پشته A اضافه (push) می‌کنیم. برای نوشتن و رفتن به چپ نیز مقدار بالایی B را pop می‌کنیم، مقدار جدید را push می‌کنیم و بعد مقدار بالایی A را pop می‌کنیم و آن را به B اضافه می‌کنیم. برای حالت برعکس نیز کافی است یک ماشین تورینگ دو نواره را در نظر بگیرید، از آنجایی که قدرت نوار از پشته بیشتر است پس قطعا می‌توان این ماشین را با ماشین تورینگ استاندارد شبیه سازی کرد.

ب) برای شبیه‌سازی ماشین تورینگ با این ماشین به ازای هر حرف از الفبای زبان یک حرف دیگر اضافه می‌کنیم که نماینده همان حرف است در صورتی که head ماشین تورینگ استاندارد روی آن قرار دارد برای مثال می‌توان همان حرف با یک نقطه بالای آن را در نظر گرفت. همچنین به یک جدا کنند مثل # نیز نیاز داریم. از آنجایی که در هر خانه فقط یک بار می توانیم مقدار معنا دار بنویسیم پس برای اعمال هر تغییری روی نوار یک بار محتوای آن را به صورت کامل در قسمت خالی سمت چپ نوار کپی می‌کنیم و در هنگام کپی کردن تغییر را اعمال می‌کنیم. در روند کپی کردن مقدار اولین المانی نسوخته است را می‌خوانیم، آن را می‌سوزانیم و به اولین خانه خالی نوار اضافه می‌کنیم. در این حین برای گم نکردن مکان head از حروف نقطه داری که اضافه کرده‌ایم کمک می‌گیریم. همچنین برای جدا سازی بخش قبلی که در حال کپی کردن آن هستیم با بخش جدید ابتدا یک # در انتهای بخش قبلی می‌گذاریم. برای حالت برعکس نیز یک نوار استاندارد آزادی عمل بیشتری نسبت به این نوار دارد. پس قطعا می‌توان این ماشین را با ماشین تورینگ استاندارد شبیه سازی کرد.

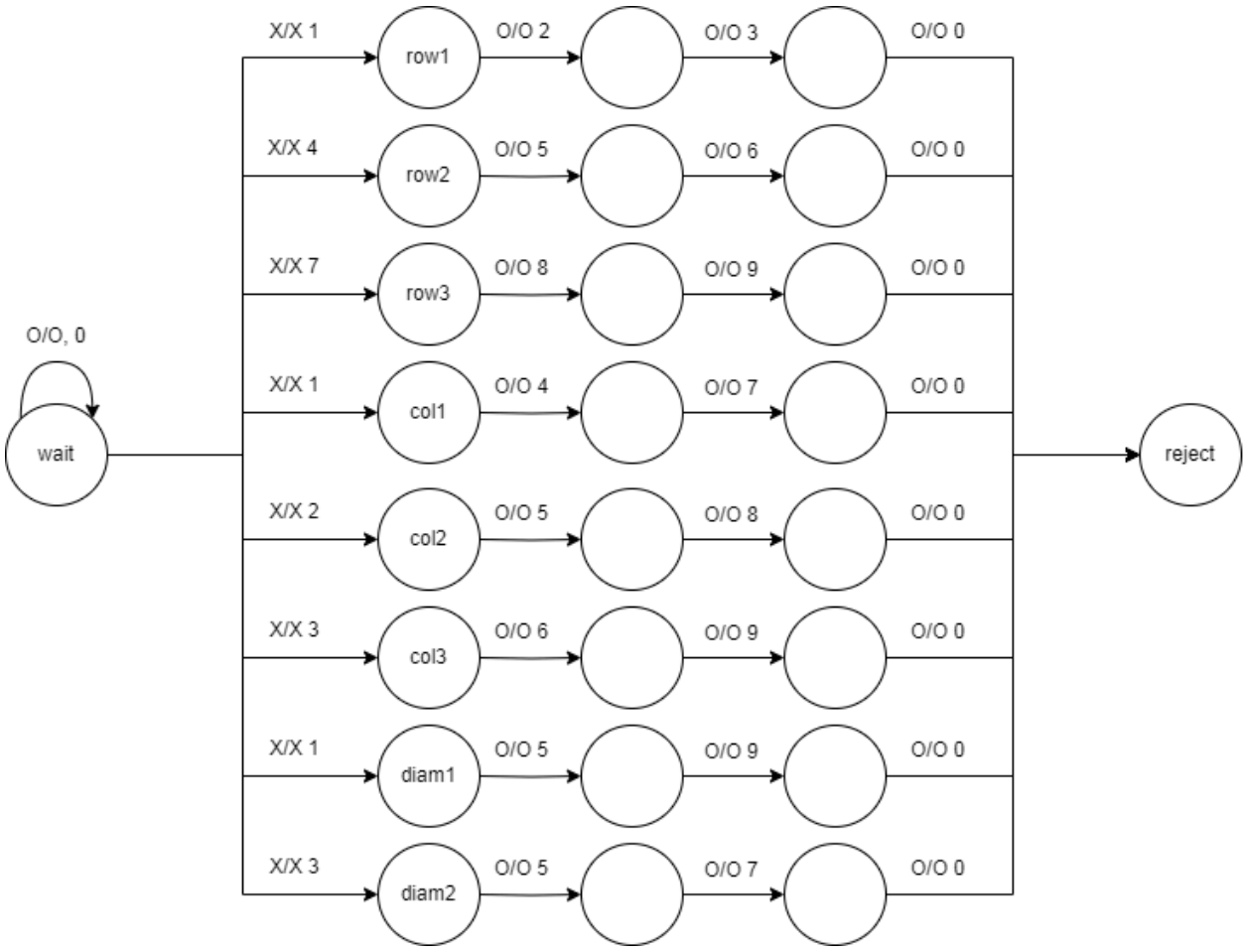
پ) کافی است یک جدا کننده مانند # در نظر بگیریم. حال به ازای نوشتن در سطر nام ماشین تورینگ صفحه‌ای کافی است نوار ماشین تورینگ استاندارد را به n بخش تقسیم کنیم و حرف مورد نظر را در بخش nام آن بنویسیم. واضح است که اگر بخش nامی که به آن اشاره شد قبلا ساخته شده باشد صرفا اندازه آن را بزرگتر می‌کنیم و حرف را در آن می‌نویسیم. توجه داشته باشید از آنجایی که تعداد سطرها بینهایت است نمی‌توانیم از اول نوار را تقسیم بندی کنیم یا حتی یک ماشین تورینگ با بینهایت نوار در نظر بگیریم. بلکه در هنگام اجرای برنامه باید دید که در چه سطری مقدار نوشته می‌شود تا به همان تعداد بخش به نوار اصلی اضافه کنیم. برای حالت برعکس آن نیز کافی است فقط از سطر اول ماشین صفحه‌ای استفاده کنیم تا بتوانیم رفتار ماشین تورینگ استاندارد را شبیه سازی کنیم.

ت) یک ماشین تورینگ دو نواره در نظر می‌گیریم. هنگامی که می‌خواهیم به خانه‌ای با ایندکس n برویم، n را در نوار دوم می‌نویسیم و head نوار اول را به خانه اول بر می‌گردانیم. حال یک واحد به سمت راست می‌رویم و یک واحد از عددی که در نوار دوم است کم می‌کنیم. این کار را آنقدر ادامه می‌دهیم تا عدد نوار دوم صفر شود (جزئیات نحوه انجام عملیات کم کردن و مقایسه با صفر نمره‌ای ندارد). برای حالت برعکس نیز ابتدا باید یک مکانیزمی برای نگهداری ایندکس فعلی در نظر بگیریم (می‌توانیم آن را در خانه اول نوار ذخیره کنیم یا هر مکانیزم دیگر). حال برای رفتن به سمت راست یا چپ کافی است ایندکسی که می خواهیم به آن برویم یک واحد بیشتر یا کمتر از ایندکس فعلی باشد. در آخر نیز مقدار ایندکس جدید را در مکانی ذخیره می‌کنیم تا برای حرکت‌های بعدی از آن استفاده کنیم.

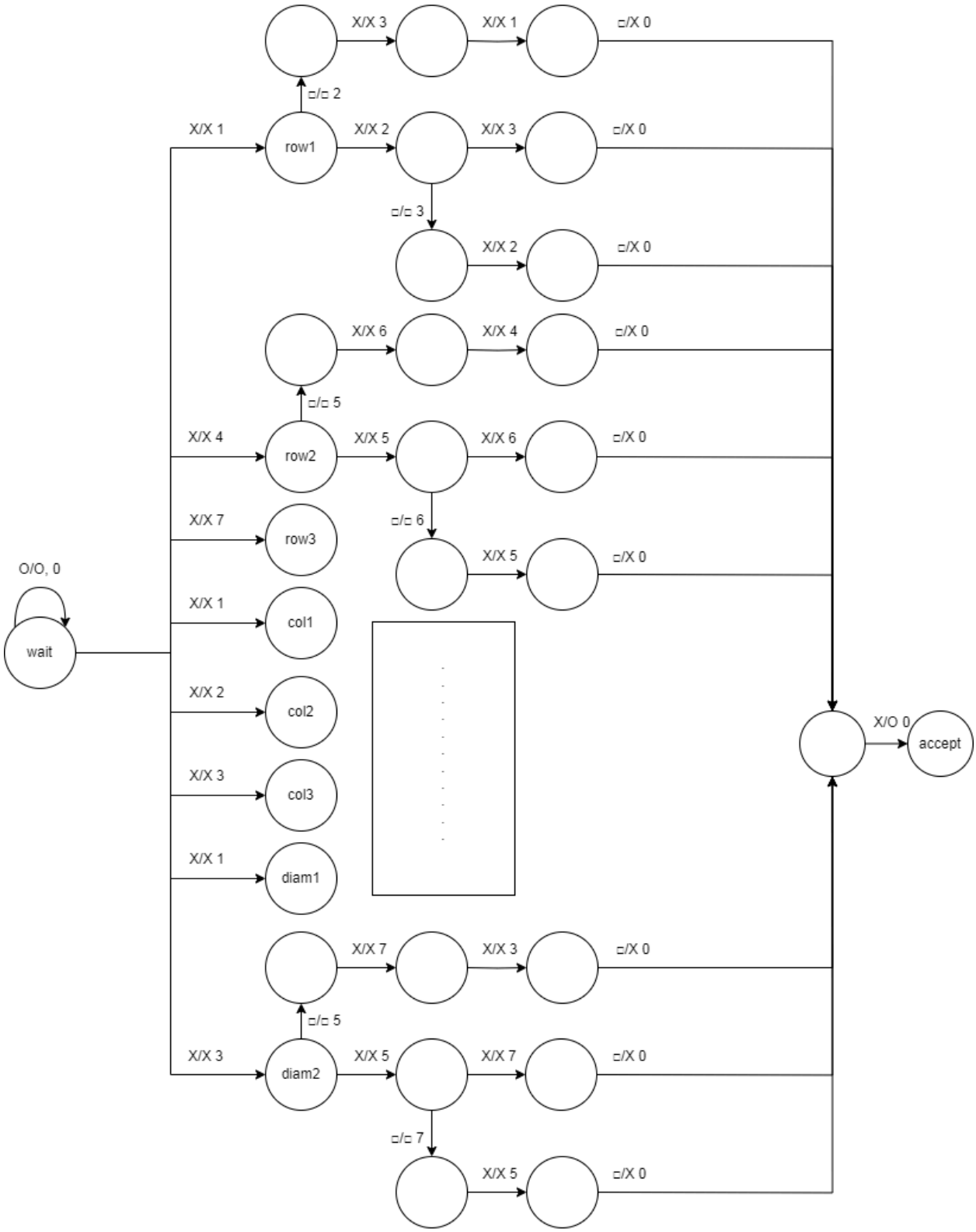
ث) در اینجا نیاز به تغییری در ماشین تورینگ نیست، بلکه باید الفبای زبان را به گونه‌ای با صفر و یک کد کنیم. از آنجایی که مجموعه الفبا نمی‌تواند نامحدود باشد پس حروف هر الفبایی را می‌توان با log_2^n کاراکتر صفر و یک تشخیص داد. n تعداد حروف الفبا است (هر نوع کد گذاری دیگر نیز صحیح است). برای حالت برعکس نیز کافی است حروف الفبای ماشین تورینگ را به صفر و یک محدود کنیم تا ماشین ذکر شده را شبیه سازی کند.

2. یک ماشین تورینگ غیر قطعی با n نوار در نظر می‌گیریم. شروع به پیمایش ورودی می‌کنیم و هر جفت key, value را در یک نوار می‌نویسیم. در آخر به صورت همزمان همه نوار ها را بررسی می‌کنیم تا ببینیم مقدار گفته شده در دستور آخر بین مقادیر وجود دارد یا نه و با توجه به نوع دستور جواب مناسب را می‌نویسیم و یا ریجکت می‌کنیم. اگر n از تعداد key, value ها کمتر بود باید در دو مرحله بررسی را انجام بدیم.

3. الف)


O/O, 0
wait
X/X 1
row1
O/O 2
O/O 3
O/O 0
X/X 4
row2
O/O 5
O/O 6
O/O 0
X/X 7
row3
O/O 8
O/O 9
O/O 0
X/X 1
col1
O/O 4
O/O 7
O/O 0
X/X 2
col2
O/O 5
O/O 8
O/O 0
X/X 3
col3
O/O 6
O/O 9
O/O 0
X/X 1
diam1
O/O 5
O/O 9
O/O 0
X/X 3
diam2
O/O 5
O/O 7
O/O 0
reject

ب)


X/X 3
X/X 1
□/X 0
□/□ 2
X/X 1
row1
X/X 2
X/X 3
□/X 0
□/□ 3
X/X 2
□/X 0
X/X 6
X/X 4
□/X 0
□/□ 5
X/X 4
row2
X/X 5
X/X 6
□/X 0
X/X 7
row3
□/□ 6
X/X 5
□/X 0
O/O, 0
wait
X/X 1
col1
X/X 2
col2
X/X 3
col3
X/X 1
diam1
X/O 0
accept
X/X 7
X/X 3
□/X 0
□/□ 5
X/X 3
diam2
X/X 5
X/X 7
□/X 0
□/□ 7
X/X 5
□/X 0

پ) مانند بخش ب است. با این تفاوت که به جای خواندن X در تمامی مراحل **چک کردن** مقدار O می‌خوانیم. و در نهایت بجای accept به wait بر می‌گردیم.

ت) چهار مسیر پیروزی از خانه وسط، سه مسیر از خانه‌های گوشه و دو مسیر از خانه‌هایی کناری می‌گذرد. پس اولویت انتخاب بر اساس این ترتیب است.

ث) هر شبیه سازی که حرکت اول وسط باشد، سه حرکت بعدی در گوشه‌ها و در نهایت به تساوی ختم شود درست است.

4. برای حل این سوال از ایده سوال یک بخش ب کمک می‌گیریم و مثل آن عمل می‌کنیم. با این تفاوت که نمی‌توانیم خانه‌ها را بسوزانیم، پس هنگام کپی کردن بین هر دو خانه یک فاصله خالی قرار می‌دهیم. حال در سری بعدی کپی کردن کافی است خانه بعدی مقداری که داریم کپی می کنیم (که فعلا خالی است و تا به حال چیزی در آن نوشته نشده) را با کاراکتری خاص و خارج از الفبای اصلی پر کنیم تا هنگام برگشت برای کپی کردن خانه بعدی آخرین خانه‌ای که کپی شده را گم نکنیم.

توجه: در صورت سوال ذکر نشده که خانه‌هایی که روی آنها ورودی نوشته شده را می‌توان تغییر داد یا خیر. اگر تغییر خانه‌هایی که ورودی روی آنها قرار دارد مجاز نباشد نمی‌توانیم اولین کپی را انجام دهیم و قدرت آن از ماشین تورینگ استاندارد کمتر می‌شود. ولی اگر تغییر آنها مجاز باشد بعد از انجام اولین کپی به روش سوال یک بخش ب، مابقی کار مانند توضیح بالا خواهد بود. سوال با توجه به فرض شما صحیح می‌شود.

5. سه روتر را می‌توان به شش حالت کنار یکدیگر قرار داد و هر کدام 26 تنظیم اولیه دارند. یک ماشین تورینگ قطعی باید این حالات را به ترتیب چک کند تا به تنظیمی برسد که با گرفتن ورودی معمولی خروجی کد شده را تحویل دهد. در صورتی که ماشین تورینگ غیر قطعی در یک لحظه می‌تواند در چند استیت مختلف باشد و هر کدام از شاخه‌های اجرا، به صورت مستقل و بدون اینکه روی نوار شاخه‌ای دیگر تغییری ایجاد کند با نوار خود کار کند در صورتی در اصل فقط یک نوار وجود دارد. به همین علت می‌تواند همزمان تمامی حالات مختلف چیدمان را تست کند بدون آنکه هزینه‌ای برای افزایش قدرت پردازشی به روش‌های مرسوم شده باشد.