دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)
دانشکده مهندسی کامپیوتر

تمرین چهار

آزمون نرم‌افزار

نگارش
محمدرضا اخگری زیری
محمدعلی کشاورز
علی نظری

استاد درس
دکتر معصومه طارمی‌راد

بهار ۱۴۰۰

فهرست مطالب

صفحه

سوال اول

برای پیدا کردن مسیرهای اصلی از کد موجود در لینک گیت‌هاب کمک گرفته شده است. برای اجرای کد لازم است که مورد آزمون را در فایلی با فرمت زیر بنویسیم.

- خط اول فایل نام گره‌های گراف است.
- خط دوم فایل نام گره‌های آغازین گراف است.
- خط سوم فایل نام گره‌های انتهایی گراف است که با فضای خالی جدا شده‌اند.
- خطوط بعدی نشان‌دهنده‌ی یال‌ها هستند، برای مثال خط اول نشان می‌دهد که گره‌ی اول به کدامین گره‌ها وصل است و میان آن‌ها یالی است (یال جهت‌دار).

پس از پیدا کردن مسیرهای اصلی، برای پیدا کردن موارد آزمون (موارد بهینه نیستند)، از الگوریتم پیمایش سطحی گراف استفاده کردیم. یعنی به گونه‌ای که از گره آخر مسیر اصلی پیمایش سطحی انجام دادیم تا به یکی از نقاط پایانی برسیم و همچنین از نقاط شروع پیمایشی انجام دادیم تا به ابتدای مسیر اصلی برسیم. در صورتی که این تست قبلا یافته نشده بود یا زیر مجموعه‌ی تست دیگری نبود، این تست را به موارد آزمون اضافه کردیم. با توجه به حجیم بودن در شکل ۱ فقط به نمایش نتایج اکتفا کرده‌ایم و کد را به همراه تمرین پیوست کرده‌ایم.

```
Enter input file: test_cases/2
Graph:
            nodes: [1, 2, 3, 4, 5],
            initial_nodes: [1],
            end_nodes: [5],
            edges: {1: [2, 3], 2: [4], 3: [4], 4: [2, 5], 5: []}

prime paths are:
[2, 4, 2]
[4, 2, 4]
[1, 2, 4, 5]
[1, 3, 4, 2]
[1, 3, 4, 5]
test paths are:
[1, 3, 4, 5]
[1, 3, 4, 2, 4, 5]
[1, 2, 4, 5]
[1, 2, 4, 2, 4, 5]
```

شکل ۱- نمایش خروجی کد برای مسیرهای اصلی و مسیرهای آزمون متناظر

سوال دوم

ابزاری که ما برای پوشش کد انتخاب کردیم Jest نام دارد. این ابزار، یک ابزار متن‌باز برای زبان JS و مشتقات آن مانند TypeScript است که بیشتر در پروژه‌های سمت مشتری[1] استفاده می‌شود.

برای انجام تنظیمات آن، ما یک فایل jest.config.js داخل پروژه قرار می‌دهیم که مواردی مانند پوشه‌هایی که برای پوشش کد مورد بررسی قرار می‌گیرد و هم‌چنین پسوندهایی که باید مورد بررسی قرار بگیرند و هم‌چنین پوشه‌هایی که نباید در گزارش پوشش کد مورد بررسی قرار بگیرند را مشخص می‌کنیم.

برای مثال، در قطعه کد ۱ نمونه‌ای از این تنظیمات که برای پروژه خود که یک پروژه پیامرسان است استفاده کردیم، نمایش داده شده است:

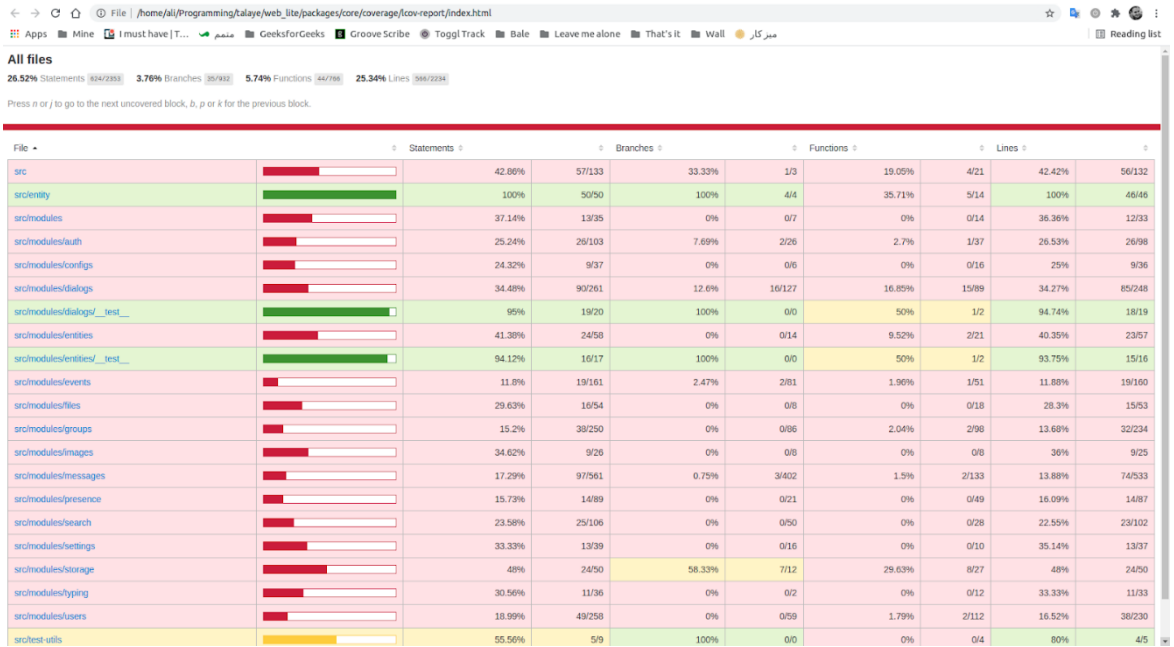
```
module.exports = {
 preset: "ts-jest",
 roots: ["<rootDir>/src"],
 collectCoverageFrom: ["src/**/*.{js,jsx,ts,tsx}", "!src/**/*.d.ts"],
 setupFiles: ["react-app-polyfill/jsdom"],
 setupFilesAfterEnv: ["<rootDir>/src/setupTests.ts"],
 testMatch: [
   "<rootDir>/src/**/*.{spec,test}.{js,jsx,ts,tsx}",
 ],
 testEnvironment: "jest-environment-jsdom-fourteen",
 transform: {
   "^.+\\.(js|jsx|ts|tsx)$": "<rootDir>/node_modules/babel-jest",
   "^.+\\.css$": "<rootDir>/config/jest/cssTransform.js",
   "^(?!.*\\.(js|jsx|ts|tsx|css|json)$)":
     "<rootDir>/config/jest/fileTransform.js",
 },
 transformIgnorePatterns: [
   "[/\\\\]node_modules[/\\\\].+\\.(js|jsx|ts|tsx)$",
   "^.+\\.module\\.(css|sass|scss)$",
 ],
 modulePaths: ["<rootDir>/src/test-utils"],
 moduleNameMapper: {
   "^react-native$": "react-native-web",
   "^.+\\.module\\.(css|sass|scss)$": "identity-obj-proxy",
   "~/(.*)": "<rootDir>/src/$1",
   "~/test-utils": "<rootDir>/src/test-utils",
 },
 moduleFileExtensions: [
   "web.js",
   "js",
```

[1] Client side

```
    "web.ts",
    "ts",
    "web.tsx",
    "tsx",
    "json",
    "web.jsx",
    "jsx",
    "node",
  ],
  watchPlugins: [
    "jest-watch-typeahead/filename",
    "jest-watch-typeahead/testname",
  ],
  coveragePathIgnorePatterns: ["/node_modules/", "api/proto/.*"],
};
```

کد ۱- تنظیمات اعمال شده برای ابزار

سپس ما یک هسته‌ی پیام‌رسانی را با کمک این ابزار مورد بررسی قرار دادیم که برخی از این گزارشات در ادامه آورده شده اند. شکل ۲ نشان‌دهنده‌ی گزارش کلی از میزان پوشش معیارهای مختلف (مانند پوشش شاخه، پوشش دستور، پوشش تابع و یا پوشش خطوط) به ما می‌دهد.



شکل ۲- نمایش گزارش کلی از پوشش کد

```
163 1x   public findDialog = (peer: ApiPeer): Observable<Dialog | undefined> => {
164        return this.db.getTable<Dialog>("dialogs").get(peer.id);
165      };
166
167      public chatOpened(peer?: ApiPeer): void {
168        this.currentPeer = peer;
169      }
170
171 1x   private putDialogs = (dialogs: Dialog[]): Observable<Dialog[]> => {
172 1x     E if (dialogs.length > 0) {
173          // TODO: PUT BUCK
174 1x       return zip(
175            ...dialogs.map((dialogs) =>
176 1x           this.db.getTable("dialogs").put(dialogs, dialogs.peer?.id)
177            )
178          ).pipe(mapTo(dialogs));
179        } else {
180          return of([]);
181        }
182      };
183
184      public putDialog(dialog: Dialog): Observable<Dialog> {
185        return this.putDialogs([dialog]).pipe(map((dialogs) => dialogs[0]));
186      }
187
```

شکل ۳- نمایش جزئی‌تر پوشش هر خط

در شکل ۳ گزارش به صورت دقیق‌تری نمایش داده شده است، یعنی در این شکل قسمت‌هایی که توسط آزمون‌ها پوشش داده‌نشده‌اند با رنگ قرمز مشخص شده است.

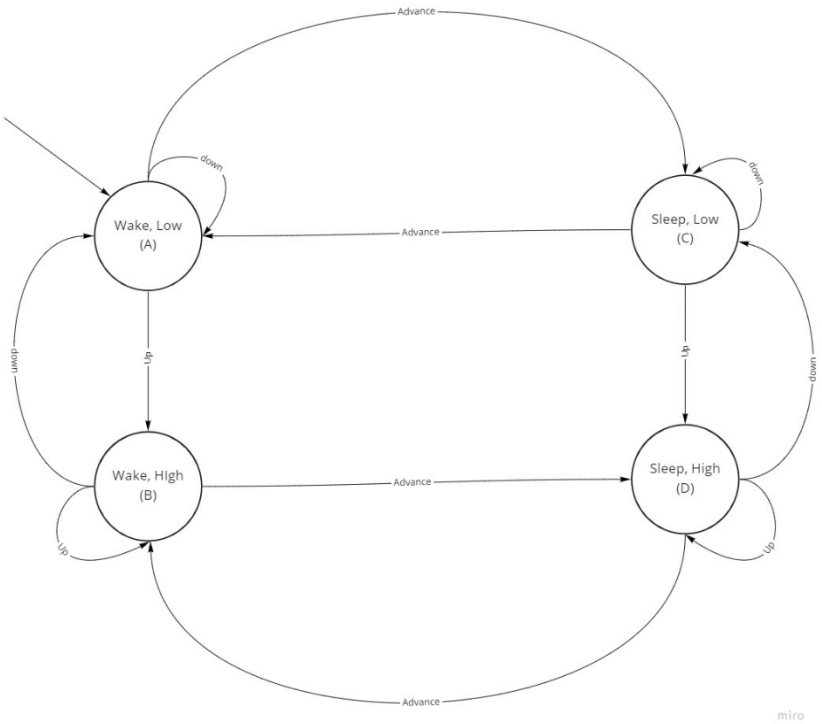
برخی از مشـــکلاتی که ما در اســـتفاده از این ابزار داریم به خود تنظیمات js برمی‌گردد که این ابزار با بقیه‌ی ابزارهای پکیج‌بندی ما با اختلال رو به رو می‌شـد و برخی از کانفیگ‌های ها نادیده گرفته می‌شـد. مثلا برای انتخاب دایرکتوری‌هایی که می‌خواستیم مورد بررسی قرار گیرند با مشکلاتی رو به رو شدیم.

سوال سوم

الف) این مدل دارای ۴ حالت هست که در ادامه به ذکر نام آن‌ها می‌پردازیم:

A- Wake, Low
B- Wake, High
C- Sleep, Low
D- Sleep, High

ب) مدل رسم شده در شکل ۴ نمایش داده شده است.



ج) مورد آزمونی که پوشش یال را نیز برآورده کند، در ادامه نشان داده شده است:

A Up B

B Down A

A Down A

A Advance C

C Advance A

A Advance C

C Down C

C Up D

D Up D

D Advance B

B Up B

B Advance D

D Down C

C Down C

D Down C

سوال چهارم

به طور کلی سناریوهای پوشش داده شده، دو سناریو پرداخت قبوض و استعلام موجودی هستند.

الف)

- **کاربری:** پرداخت قبوض
- **پیش‌شرایط:** آماده به کار بودن دستگاه
- **سناریو:**

۱- کاربر کارت را وارد می‌کند.

۲- زبان را انتخاب می‌کند.

۳- رمز را وارد می‌کند.

٤- پرداخت قبوض را انتخاب می‌کند.

٥- شناسه قبض را وارد می‌کند.

٦- شناسه پرداخت را وارد می‌کند.

۷- مشخصات را تایید می‌کند.

۸- رسید را دریافت می‌کند.

۹- خروج را می‌زند.

۱۰- کارت را دریافت می‌کند.

- **نتیجه و حالت نهایی:** قبض پرداخت شود.
- **استثناها:**

- اگر کارت معتبر نباشد، کارت خارج شود.
- اگر رمز اشتباه وارد شود با اعلام خطا به کاربر (تا ۳ بار) فرصت مجدد داده می‌شود.
- اگر رمز بیش از سه بار اشتباه وارد شود، حساب مسدود و کارت ذخیره شود.
- اگر شناسه قبض یا شناسه پرداخت نامعتبر باشد به کاربر اعلام شود و به مرحله‌ی بعدی نرود.
- اگر موجودی کاربر برای پرداخت قبض کافی نبود، با اعلام این موضوع به صفحه‌ی اول برگردد.

- **تست:** پرداخت قبض وقتی که موجودی کافی نیست:
  - **شرایط موجود و ورودی:** کاربری می‌خواهد قبضی به شناسه قبض ۳۴۵۳۴۵۴۵۶ و شناسه پرداخت ۹۸۲۹۴۸۶۸۲ که هزینه‌ی آن ۹۰,۰۰۰ تومان است را هنگامی که ۵۰,۰۰۰ تومان موجودی دارد، پرداخت کند.
  - **خروجی:** در این حلت پولی از حساب او کم نمی‌شود و موجودی همان مقدار ۵۰ هزارتومان باقی می‌ملند و نمی‌تولند قبض را پرداخت کند زیرا موجودی از هزینه کمتر است.
- **تست:** پرداخت قبض در زمانی که موجود کافی است:
  - **شرایط موجود و ورودی:** کاربری می‌خواهد قبضی به شناسه قبض ۳۴۵۳۴۵۴۵۶ و شناسه پرداخت ۹۸۲۹۴۸۶۸۲ که هزینه‌ی آن ۹۰,۰۰۰ تومان است را هنگامی که ۱۰۰,۰۰۰ تومان موجودی دارد، پرداخت کند.
  - **خروجی:** در این حالت پولی از حساب او کم می‌شود و قبض او پرداخت می‌شود و موجودی او در پایان به ۱۰,۰۰۰ تومان می‌رسد.

** (مراحل اجرای تست به همان ترتیب سناریو است)

ب)

- **کاربری:** دریافت موجودی
- **پیش‌شرایط:** آماده به کار بودن دستگاه
- **سناریو:**

۱- کارت را وارد می‌کند.
۲- زبان را انتخاب می‌کند.
۳- رمز را وارد می‌کند.
٤- دریافت موجودی را انتخاب می‌کند.
٥- رسید را دریافت می‌کند.
٦- خروج را می‌زند.
۷- کارت را دریافت می‌کند.

- **نتیجه و حالت نهایی:** فقط کارمزد (۱۰۰ تومان) از حساب شخص کاسته شود.
- **استثناها:**
  - اگر کارت معتبر نباشد، کارت خارج شود.
  - اگر رمز اشتباه وارد شود با اعلام خطا به کاربر (تا ۳ بار) فرصت مجدد داده می‌شود.
  - اگر رمز بیش از سه بار اشتباه وارد شود، حساب مسدود و کارت ذخیره شود.
  - اگر موجودی کاربر برای کارمزد کافی نبود، با اعلام این موضوع به صفحه‌ی اول برگردد.
- **تست:** دریافت موجودی وقتی که موجودی برای کسر کارمزد کافی نیست:
  - **شـــرایط موجود و ورودی:** کاربری کارت را وارد می‌کند و رمزش را نیز وارد می‌کند و موجودی او ۵۰ تومان است.
  - **خروجی:** رمز کاربر درســـت اســـت ولی موجودی او کمتر از کارمزد ۱۰۰ تومانی برای موجودی است و عملیات انجام نمی‌شود و به صفحه‌ی اول بازمی‌گردد.
- **تست:** دریافت موجودی وقتی که موجودی برای کسر کارمزد کافی است:
  - **شـــرایط موجود و ورودی:** کاربری کارت را وارد می‌کند و رمزش را نیز وارد می‌کند و موجودی او ۲۰,۰۰۰ تومان است.
  - **خروجی:** رمز کاربر درست است و موجودی او نیز از ۱۰۰ تومان بیشتر است و کارمزد می‌تواند از حساب او کسر شود و موجودی او به ۱۹,۹۰۰ تومان می‌رسد و رسید را هم دریافت می‌کند.

گزارش انجام تمرین به صورت گروهی:

برای انجام این تمرین با توجه به کمبود وقت، تقسیم به صورت نگاشت سوال به شخص بود و سوال اول را آقای اخگری، سوال دوم را آقای نظری و سوال سوم را آقای کشاورز انجام دادند، پس از آن در جلسه‌ی برخطی به توضیح سوالات برای افراد گروه پرداختیم و سوال چهارم را با همفکری هم حل کردیم. گزارش سوالات در جلسه نوشته شد و پس از اتمام جلسه اقدام به بازنویسی آن کردیم و پس از تایید اعضای گروه ارسال کردیم.