تحلیل و طراحی الگوریتم (فاز سوم)

نیم سال ۴۰۴۲

دانشکده‌ی مهندسی کامپیوتر – آزاد شیراز

مدرس: دکتر اسکندری

طراحان:

امیر حسین همتی ، حمید رضا نامجومنش،

ویانا حسین بیگی ، مریم میدانشاهی ،

علی نیکوان ، رضا رضایی ،نیما سلطانی،

بابک جانفزا

زمان تحویل: ۲۵ خرداد

- پروژه انفرادی ویا نهایتا ۲ نفره است.
- فقط در موارد ذکرشده مجاز به استفاده از کتابخانه‌های آماده هستید.
- هر گروه باید کد پروژه خود را در مهلت اعلام شده ارسال نماید.
- علاوه بر گزارش پروژه، در انتهای ترم یک ارائه و دفاع نیز از این پروژه گرفته خواهد شد. مهلت ارسال پاسخ تا ساعت ۲۳:۵۹ روز مشخص شده است.
- همکاری و هم فکری شما در انجام پروژه مانع ندارد اما پاسخ ارسالی هر گروه حتما باید توسط خود او نوشته شده باشد.
- در صورت هم فکری و یا استفاده از هر منابع خارج درسی، نام هم فکران و آدرس منابع مورد استفاده برای حل سوال مورد نظر را ذکر کنید.

1 مقدمه

در فاز سوم پروژه **فرار از فاکس ریور** ،مایکل اسکافیلد و تیمش از زندان فاکس ریور فرار کرده‌اند، اما اکنون با شبکه گسترده‌تری از موانع امنیتی، جاده‌های تحت نظارت و مخفی‌گاه‌های نزدیک به هم مواجه هستند. برای عبور موفق از این مرحله، باید مسائل بهینه‌سازی ترکیبیاتی و تصمیم‌گیری در شرایط محدودیت زمانی را حل کنند

در این فاز، تیم فرار باید پنج ایستگاه عملیاتی را پشت سر بگذارد:

۱. **کوتاه‌ترین مسیر بازدید از شهرها** (Held-Karp)
۲. **یافتن سیکل همیلتونی** (Backtracking)
۳. **رنگ‌آمیزی مخفی‌گاه‌ها** (Graph Coloring)
۴. **مفاهیم P و NP** (NP-Complete)
۵. **تقریب مسیر فرار** (Approximation for TSP)

همه پیاده‌سازی‌ها باید از صفر (from scratch) انجام شود، اما استفاده از کتابخانه‌های استاندارد زیر برای اهداف کمکی مجاز است:

توضیح	کاربرد مجاز	کتابخانه
فقط برای مشخص کردن نوع داده‌ها	Type hints (List, Tuple, Dict, Optional,)	Typing
ساختار داده صف برای پیمایش	اختیاری (با ذکر دلیل)	collections
فقط برای عملیات ساده	توابع پایه ریاضی (.inf, floor, etc)	math
برای مسائل heap-based	فقط در صورت نیاز و با ذکر دلیل	heapq

2 توضیح تکمیلی

این فاز از پروژه را در ۵ مرحله قرار است تکمیل کنید، ولی قبل از هرچیزی ابتدا باید پروژه را از طریق کوئرا دانلود کنید و آن را از حالت فشرده خارج کنید.
به موارد زیر توجه کنید:

- از تغییر دادن نام کلاس ها و همچنین قرار دادن فایل های پروژه خود در دایرکتوری های مختلف خودداری کنید. در انتها این فاز، پروژه شما با اجرای کد autograder نمره دهی خواهد شد و این کد از نام کلاس های مشخص شده استفاده می‌کند و همچنین فرض می‌کند که تمام فایل ها در کنار این فایل قرار گرفته‌اند.
- سه دسته تابع در کدها وجود دارد:

  - دسته‌ی اول تابع هایی هستند که با کامنت TODO مشخص شده‌اند. این قسمت ها باید توسط شما تکمیل شوند.
  - دسته‌ی دوم تابع هایی هستند که با کامنت PROVIDED (keep) مشخص شده‌اند. این قسمت ها معمولا

helper method هایی هستند که برای اجرای بهتر و راحت تر کد مورد نیاز است. بهتر است از تغییر آن ها خودداری کنید.

- ➢ دسته سوم تابع های هستند که کامنت هایی شبیه DO NOT CHANGE یا leave as-is مشخص شده‌اند.

از تغییر دادن این توابع بپرهیزید چرا که اجرای کد autograder را با مشکل مواجه می‌کند.

- پس از اتمام پروژه حتما کد autograder را خودتان نیز یک بار اجرا کنید تا از درست همه چیز مطمئن شوید. از

تغییر این کد خودداری کنید چرا که برای نمره دهی این کد به صورت بدون تغییر در کنار پروژه شما جایگزین و اجرا می شود.

- شما می‌توانید هر تعدادی helper method که خواستید به کلاس ها اضافه کنید یا هر بخش که DO NOT CHANGE نخورده است را تغییر دهید. فقط در نهایت از درستی عملکرد autograder اطمینان حاصل کنید.
- استفاده از کتابخانه‌های آماده برای پیاده‌سازی مستقیم الگوریتم‌ها غیر مجاز است و باید تمام بخش‌ها را از صفر (from scratch) پیاده‌سازی کنید. کتابخانه‌های typing، collections.deque و math (تنها برای اهداف کمکی) مجاز هستند.
- توجه کنید که آن چیزی که در نهایت برای ارزیابی پروژه شما بررسی می‌شود، کارکرد آن در بازیابی درست اسناد است؛ به همین علت فارغ از انجام تمام مراحل ذکر شده در ادامه، از بازگرداندن اسناد درست با یک کوئری مشخص اطمینان حاصل کنید.

حال برای تکمیل پروژه کافی است مراحل که در ادامه آورده شده است را به درست و به ترتیب انجام دهید.

3 مرحله صفر –دریافت فایل‌های پروژه

فایل‌های اولیه پروژه شامل:

```
project/
├── tsp_dp.py
├── hamiltonian.py
├── graph_coloring.py
├── tsp_approximation.py
├── autograder.py
├── utils.py
└── report.md یا report.pdf  # پاسخ سوالات نظری
```

4 مرحله اول – چالش شبکه شهرها(TSP)

🗀 **فایل** tsp_dp.py **:**

```
def held_karp_tsp(distances: List[List[float]]) -> Tuple[float, List[int]]:
    """
    TODO: الگوریتم پیاده‌سازی Held-Karp (برای پویا برنامه‌ریزی TSP)
    distances: ماتریس فاصله متقارن/زمان بین شهرها (کامل گراف)
    return: (هزینه حداقل، ترتیب شهرها از شروع [۰ شهر])
    """
```

راهنما:

- حالت DP: $dp[mask][i]$ = کمترین هزینه برای بازدید از مجموعه mask (که شامل شهر i است) و پایان در شهر i.
- بازگشت: $dp[mask][i] = min(dp[mask \wedge (1 << i)][j] + distances[j][i])$ به ازای همه j در mask.
- پیچیدگی $O(n^2 \cdot 2^n)$.
- از بیت‌ماسک برای نمایش زیرمجموعه‌ها استفاده کنید.
- شرط شروع و پایان: فرض کنید شهر شماره ۰ نقطه شروع و پایان است (می‌توانید بعداً برگردانید).

5مرحله دوم – چالش عبور از مرز امنیتی(Hamilton path)

🗀 **فایل** hamiltonian.py **:**

```
def hamiltonian_cycle(graph: List[List[bool]]) -> List[int]:
    """
    TODO: الگوریتم با همیلتونی سیکل یافتن Backtracking
    graph: ماتریس مجاورت (یال وجود)
    return: ترتیب رأس‌ها (رأس ۰ شروع و به ختم شود) یا [] در صورت نبود
    """
```

راهنما:

- از بازگشت و عقبگرد استفاده کنید.
- آرایه path را نگه دارید و هر بار رأس جدیدی که مجاور با آخرین رأس و استفاده نشده است اضافه کنید.
- در پایان، شرط بازگشت به رأس اول را چک کنید.
- تحلیل پیچیدگی: O(n!) در بدترین حالت.
- توضیح دهید چرا مسئله سیکل همیلتونی NP-Complete است (قسمت نظری).

6 مرحله سوم – چالش شبکه مخفی‌گاه‌ها (graph coloring)

🗀 **فایل** graph_coloring.py **:**

```
def graph_coloring(graph: List[List[bool]], max_colors: int = None) -> Dict[int, int]:
    """
    TODO: Greedy یا Backtracking با گراف رنگ‌آمیزی
    graph: مجاورت ماتریس
    max_colors: مجاز رنگ تعداد حداکثر (صورت در None، حداقل تعداد را پیدا کنید)
    return: دیکشنری {رأس: رنگ} (رنگ‌ها از ۰ شروع می‌شوند)
    """
```

راهنما :

- روش Greedy: ترتیب رأس‌ها (مثلاً بر اساس درجه نزولی) و اختصاص اولین رنگ مجاز. تعداد رنگ‌ها تضمین شده بهینه نیست ولی سریع است.
- روش Backtracking: تمام ترکیب‌ها را بررسی کنید تا کمترین تعداد رنگ بیابید (مناسب گراف‌های کوچک).
- پیاده‌سازی هر دو روش مجاز است، اما در مستندات خود ذکر کنید کدام را استفاده کرده‌اید.
- در صورت استفاده از Backtracking، می‌توانید با max_colors جستجو را محدود کنید.
- تفسیر: هر رنگ یک بازه زمانی است.

7 مرحله چهارم – چالش محدودیت زمان (NP , P)

🗀 **فایل نظری :این بخش در گزارش پاسخ داده می‌شود (بدون کدنویسی).**

سوالات :

۱. تعریف کلاس P و کلاس NP را بنویسید.
۲. مفهوم NP-Complete چیست؟
۳. ثابت کنید (با ذکر دلیل) چرا مسائل زیر NP-Complete هستند:
   - سیکل همیلتونی
   - رنگ‌آمیزی گراف (با ۳ رنگ)
   - TSP (نسخه تصمیمی: آیا مسیری با طول ≤ K وجود دارد؟)
٤. چرا برای مسائل NP-Complete معمولاً از الگوریتم‌های تقریبی یا هیوریستیک استفاده می‌شود؟

8مرحله پنجم – چالش فرار نهایی (TSP)

🗀 **فایل** tsp_approximation.py **:**

```
def approx_tsp(distances: List[List[float]]) -> Tuple[float, List[int]]:
    """
    TODO: الگوریتم تقریبی – ۲ برای تقریبی TSP متریک
    distances: ماتریس متقارن فاصله (باید مثلث شرط برقرار است)
    return: (طول مسیر تقریبی، ترتیب رأس‌ها)
    """
```

راهنما:

- یک MST از گراف کامل بسازید (با الگوریتم پریم یا کراسکال).
- یک پیمایش پیش‌ترتیب (preorder) از درخت انجام دهید (از رأس ۰ شروع کنید).
- ترتیب حاصل از پیمایش، یک مسیر TSP می‌دهد که حداکثر ۲ برابر جواب بهینه است (برای گراف‌های متریک).

- در انتها به رأس اول برگردید.

نکته: طول مسیر تقریبی را با جمع فواصل بین رأس‌های متوالی محاسبه کنید.

سوال نظری:
نسبت تقریب الگوریتم فوق را اثبات کنید (چرا حداکثر ۲ برابر اپتیمم است؟).

9 سوالات نظری (باید در گزارش پاسخ دهید)

پاسخ این سوالات را در فایل گزارش (PDF یا Markdown) به همراه تحلیل پیچیدگی‌ها و مستندات کد خود ارائه دهید:

۱. تحلیل پیچیدگی الگوریتم Held-Karp – چرا $O(n^2 \cdot 2^n)$ است؟ برای n=20 چند ثانیه تقریباً زمان می‌برد؟

۲. علت NP-Complete بودن مسائل – برای یکی از مسائل (مثلاً سیکل همیلتونی) توضیح دهید چرا احراز جواب سریع است ولی یافتن جواب احتمالاً نمایی است.

۳. نسبت تقریب الگوریتم MST برای TSP – اثبات کنید این الگوریتم یک ۲-تقریبی است. آیا برای گراف‌های غیرمتریک هم جواب می‌دهد؟

٤. مقایسه Greedy Coloring با Backtracking – چه زمانی Greedy جواب بهینه می‌دهد؟ یک مثال بزنید که Greedy جواب بهینه ندهد.

٥. کاربرد مفاهیم P و NP در پروژه – کدام یک از مسائل فاز سوم در P هستند و کدام‌ها در NP-Complete؟ آیا الگوریتم دقیق برای TSP (Held-Karp) را می‌توان «کارا» دانست؟ چرا؟

10 ارزیابی پروژه

در این بخش از پروژه شما باید از درستی عملکرد پروژه خود اطمینان حاصل کنید. برای راحتی شما در ارزیابی درست پروژه کافی است که فایل autograder را اجرا کنید و مطمئن شوید تمام موارد passed می‌شوند و هیچ failed ای اتفاق نمی‌افتد. توجه کنید که این فایل فقط خروجی نهایی شما را نمی‌سنجد بلکه تمام بخش های پروژه را بررسی می‌کند.

نحوه ران کردن در صورتی که از cmd ران می کنید :

```
python autograder.py
```

خروجی مورد انتظار:

```
Testing tsp_dp... PASSED
Testing hamiltonian... PASSED
Testing graph_coloring... PASSED
Testing tsp_approximation... PASSED
All tests passed!
```