نام مدرس: دکتر محمدی دستیاران تمرین: محمدپویا تراشی، محمدمهدی شریف‌بیگی، مهرشاد فلاح	تمرین سری اول هوش مصنوعی و سیستم‌های خبره	
مهلت تحویل: چهارشنبه 16م مهر 1404		

بخش تئوری (60 نمره):

1- یک عامل در یک شهربازی قرار دارد. این شهربازی شامل دو ماشین سکه‌ای است. بازی کردن هر یک از این ماشین‌ها یک دلار هزینه دارد. بازده مورد انتظار (منظور امید ریاضی است) ماشین اول a و بازده مورد انتظار ماشین دوم b است. (12 نمره)

- شرایط PEAS را با فرض آن که پاداش ماشین‌ها نامشخص است، برای محیط معرفی شده مشخص کنید.
- اگر فرض کنیم عامل منطقی و خردمند باشد، در هر یک از حالات زیر چه واکنشی نشان می‌دهد؟

الف) عامل می‌داند که $a = 2$ و $b = 0.5$ است.

ب) عامل می‌داند $a = 2$ اما b ناشناخته است.

ج) عامل می‌داند $a = 0.999$ اما b ناشناخته است.

د) a و b هر دو ناشناخته هستند.

2- به سوالات زیر در خصوص الگوریتم‌های مطرح شده پاسخ دهید. (7 نمره)

الف) Breadth-First Search (جستجوی اول سطح):

- در چه صورتی الگوریتم BFS بهینه خواهد بود؟
- مشکل اصلی این الگوریتم نسبت به الگوریتم اول عمق چیست؟

ب) Depth-First Search (جستجوی اول عمق):

- سه پیاده‌سازی مختلف از این الگوریتم آمده است. پیچیدگی فضایی را برای هر کدام بر اساس b و m بنویسید. (b بیشینه ضریب انشعاب و m بیشینه عمق درخت جستجو است).

پیاده‌سازی اول:

```python
def dfs(root):
    if root is None:
        return

    stack = [root] # The frontier, storing entire nodes/states

    while stack:
        node = stack.pop()
        print(node.value) # Process the node

        # Push all children onto the stack
        for child in reversed(node.children): # reversed to process left-first
            stack.append(child)
```

پیاده‌سازی دوم (پیاده‌سازی بازگشتی):

```python
def recursive_dfs(node):
    if node is None:
        return

    print(node.value) # Process the node
    # For each child, dive deeper. The system remembers our place via the call stack.
    for child in node.children:
        recursive_dfs(child) # Recursive call

# Invoke:
recursive_dfs(root)
```

پیاده‌سازی سوم(جستجوی گراف - دارای closed list):

```python
def dfs_graph_search(root):
    if root is None:
        return

    stack = [root]              # Frontier
    visited = set()             # Closed list
    visited.add(root.state)     # Use a unique state identifier, not the node object itself

    while stack:
        node = stack.pop()
        print(node.value) # Process the node

        for child in node.children:
            # CRITICAL CHECK: Only add if not visited
            if child.state not in visited:
                visited.add(child.state)  # Mark state as visited
                stack.append(child)       # Add to frontier

```

ج) Iterative Deepening Search:

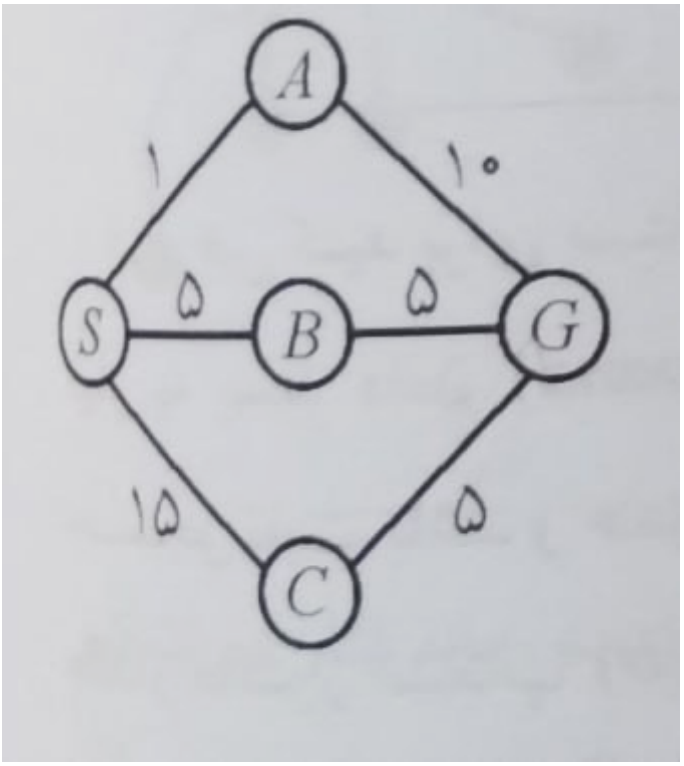
- شخصی ادعا می‌کند که این الگوریتم در شرایطی خاص بهینه است. این ادعا در چه صورتی صحت دارد؟

د) iterative Lengthening Search:

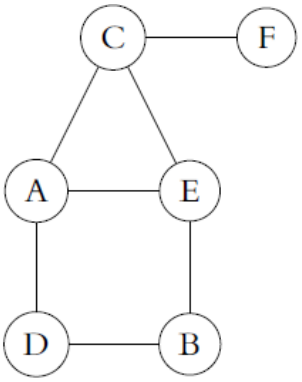
توضیحات: این الگوریتم UCS را در چندین مرحله اجرا می‌کند. در هر تکرار سقفی را (تحت عنوان مقدار برش) در نظر می‌گیرد. اگر در هنگام جستجو گرهی تشکیل شد که هزینهٔ مسیر به آن از مقدار برش بیش‌تر بود، این گره توسعه داده نمی‌شود. لینک توضیحات بیشتر

- این الگوریتم را از نظر پیچیدگی زمانی و فضایی، کامل و بهینه بودن بررسی کنید.

3- فرض کنید شکل زیر نقشهٔ شهرهای کشور لدیس‌وشهروم را نشان می‌دهد. با فرض آن که هیچگونه تخمینی از فاصلهٔ وضعیت فعلی تا هدف نداریم، برای رسیدن از S به G چه الگوریتم جستجوی بهینه‌ای پیشنهاد می‌دهید؟ ترتیب قرار گرفتن گره‌ها در frontier این الگوریتم به چه صورت خواهد بود؟ (8 نمره)



4– نمودار زیر گراف محدودیت یک CSP را نشان می‌دهد که فقط محدودیت‌های باینری دارد و در ابتدا هیچ متغیری مقداردهی نشده است. (12 نمره)



- اگر متغیر A را مقداردهی کنیم، دامنهٔ کدام متغیرها بعد از اعمال تغییر forward checking روی A تغییر خواهد کرد؟

- اگر متغیر A را مقداردهی و forward checking را روی آن اجرا کنیم سپس متغیر B را مقداردهی کنیم، با اجرای forward checking روی B دامنهٔ کدام متغیرها تغییر خواهد کرد؟
- اگر متغیر A را مقداردهی کنیم، دامنهٔ کدام متغیرها بعد از اعمال arc consistency تغییر خواهد کرد؟
- اگر متغیر A را مقداردهی و arc consistency را اجرا کنیم سپس متغیر B را مقداردهی کنیم، با اجرای arc consistency دامنهٔ کدام متغیرها تغییر خواهد کرد؟

5- یک شرکت تولیدی می‌خواهد 5 پروژهٔ مختلف (P1, P2, P3, P4, P5) را به سه تیم مختلف (T1, T2, T3) واگذار کند. محدودیت‌های زیر وجود دارد: (21 نمره)

- هر تیم حداکثر می‌تواند 2 پروژه انجام دهد.
- پروژه‌های P1 و P2 نمی‌توانند به یک تیم واگذار شوند (تداخل منابع).
- پروژه‌های P3 و P4 باید به یک تیم واگذار شوند (وابستگی تکنولوژی).
- تیم T1 تنها می‌تواند پروژه‌های P1, P3, P5 را انجام دهد.
- تیم T2 تنها می‌تواند پروژه‌های P1, P2, P4 را انجام دهد.
- تیم T3 همه پروژه‌ها را می‌تواند انجام دهد.

- این مسئله را به عنوان یک CSP مدل‌سازی کنید. متغیرها، دامنه‌ها و قیود را مشخص کنید.
- گراف قیود این مسئله را رسم کنید.
- با استفاده از الگوریتم backtracking همراه با forward checking، مسئله را حل کنید. تمام مراحل backtracking را نشان دهید.
- آیا ترتیب متفاوتی برای انتخاب متغیرها وجود دارد که منجر به backtrack کمتر شود؟ دلیل خود را بیان کنید.
- اگر محدودیت "تیم T3 نمی‌تواند بیش از یک پروژه انجام دهد" اضافه شود، آیا همچنان جواب وجود دارد؟ بدون حل کامل، تنها با تحلیل منطقی پاسخ دهید.

بخش عملی (40 نمره):

1- فرض کنید در یک دانشگاه قصد دارید با توجه به شرایط زیر برنامه امتحانی بچینید: (20 نمره)

- لیستی از درس‌ها داریم.
- هر درس توسط تعدادی دانشجو انتخاب شده است.

- اگر دو درس دانشجوی مشترک داشته باشند، امتحان آن‌ها نباید در یک زمان مشترک برگزار شود.
- تعداد اسلات‌های زمانی محدود است.

وظیفه شما به شرح زیر است:

1. مسئله را به صورت یک مدل CSP کنید. (متغیرها، دامنه‌ها و محدودیت‌ها را به درستی تعریف کنید).
2. الگوریتم‌های زیر را پیاده‌سازی کنید:
   - ابتدا Backtracking معمولی
   - در مرحله بعد Backtracking + Forward Checking + MRV.
3. خروجی را به صورت تخصیص نهایی زمان‌بندی برای هر درس برنامه‌ریزی کنید.
4. یک فایل PDF به عنوان Document (گزارش) ارسال کنید و در رابطه تفاوت زمانی دو الگوریتم پیاده‌سازی شده تحلیل انجام دهید (می‌توانید با زمان یا هر پارامتر دیگری که نمایش‌دهنده زمان انجام الگوریتم باشد این موضوع را گزارش کنید).
5. در فایل template داده شده که دارای TODO هست این کار را انجام دهید. دقت کنید وظیفه شما این است که مکان‌های TODO را کامل کرده و یک معیار برای سنجش زمان طول کشیدن هر الگوریتم بزنید.

2- یک ربات در یک هزارتوی بزرگ گیر کرده است و شما باید به او با شرایط زیر کمک کنید تا از این هزارتو خارج شود:

- بعضی موانع سر جای خودشان ثابت می‌مانند.
- برخی دیگر از موانع در طول زمان حرکت می‌کنند و ممکن است برای مسیر اولیه اختلال به وجود بیاورند.
- بعضی سلول‌ها هزینه بیشتری دارند و باید از این جلوگیری کرد.

شما باید برای این ربات الگوریتم‌های جستجوی مختلف را پیاده‌سازی کرده و در نهایت تحلیل کنید که به صورت میانگین کدام الگوریتم به جواب سریع‌تری می‌رسد و بهترین مسیر را بدون هزینه زیاد پیدا می‌کند. دقت کنید که الگوریتم‌ها عبارتند از:

- الگوریتم BFS
- الگوریتم DFS
- الگوریتم UCS
- الگوریتم A*

در نهایت این الگوریتم‌ها را از نظر تعداد حرکت و زمان و تعداد گره‌های گسترش یافته و هزینه کل مسیر مقایسه کنید (به صورت دستی محاسبه کردن و بدون اضافه کردن کد هم این عمل ممکن است و موردی ندارد).