جامعة دمشق

كلية الهندسة المعلوماتية

السنة الرابعة

قسم الذكاء الصنعي

خوارزميات البحث الذكية

الوظيفة الفصلية

تقديم الطلاب:

رغد الحلبي

سدره ميرخان

عبد العليم السيد

عمر القبطان

بإشراف:

م.عبير الكجك

الخميس 6/1/2022

المسألة الأولى:

Class Position:

وذلك لتجريد الموقع بحد ذاته حيث يحتوي object منه على 2 attributes، هما x و y .

Class Truck:

وذلك لتجريد الشاحنة حيث يحتوي object منه على object من Position للاحتفاظ بموقع الشاحنة بالإضافة إلى مصفوفة parcels من Ids البضائع.

Class GridInfo:

وذلك لتجريد معلومات الـ Grid حيث يحتوي object منه على طول وعرض الرقعة بالإضافة إلى الموقع البدائي للشاحنة والذي هو الموقع الهدف (الذي يجب أن تعود إليه بعد انتهاء جميع عمليات الاستلام والتسليم).

Class Grid:

وذلك لقراءة أمثلة مختلفة من الرقعات كـ static code بالإضافة إلى قراءة دخل من المستخدم والذي يتم تنظيمه بالشكل التالي:

```
Enter truck starting position as (x,y):
(3,6)
Enter n: 4
Enter m: 7
Enter buildings positions as (x1,y1), (x2,y2), ...:
(0,3), (1,3), (2,2), (2,3), (2,5), (3,5)
Enter receipts positions and parcelsIDs as ((x1,y1), parcelID1), ((x2,y2), parcelID2), ...:
((2,1), 0)
Enter delivers positions and parcelsIDs as ((x1,y1), parcelID1), ((x2,y2), parcelID2), ...:
((1,0), 0)
```

Class Game:

لتهيئة اللعبة تبعا لدخل المستخدم أو تبعا للـ static code.

Class State:

تم تمثيل الـ state كمصفوفات من أجل buildings، receipts، delivers بالإضافة إلى objects: truck, gridInfo.

التوابع المستخدمة:

checkInput():

وهو تابع يتحقق من كون الدخل valid وذلك من خلال إعادة نتيجة استدعاء تابع inLimits(positions) على كل position تم إدخاله و إعادة استدعاء تابع checkReceiptsAndDelivers().

inLimits(*args):

وهو تابع يتحقق من اجل كل parameter من أجل كل الـ positions أنها مواقع صالحة وذلك بحسب gidInfo حيث أنه يجب ألا يتجاوز أي موقع من المواقع حدود الرقعة.

checkReceiptsAndDelivers():

وهو تابع يتحقق من وجود مكان تسليم لكل طرد (id) له مكان استلام والعكس، أي مكان استلام لكل طرد (id) له مكان تسليم.

canMoveTruckTo(self, position):

للتحقق من إمكانية انتقال الشاحنة إلى موقع معين حيث يجب أن يكون هذا الموقع ضمن حدود الرقعة وألا يوجد بناء في هذا الموقع.

canReceive(self, position):

للتحقق من وجود مكان استلام في الموقع المحدد.

getReceivedParcelID(self, position):

للحصول على id البضاعة التي يمكن استلامها في موقع محدد.

canDeliverPP(self, parcelID, position):

للتحقق من إمكانية تسليم طرد معين في موقع معين.

canDeliverP(self, position):

للتحقق من وجود مكان تسليم في موقع محدد.

getDeliveredParcelID (self, position):

للحصول على id البضاعة التي يمكن تسليمها في موقع محدد.

endState (self):

للتحقق من كون الحالة نهائية أي حالة فوز وذلك من خلال التحقق من أن الموقع الحالي للشاحنة هو الموقع الهدف وأن جميع البضائع قد تم تسليمها (لا داعي للتحقق من كون جميع البضائع قد تم استلامها لأن هذا مشمول بحالة التسليم).

receive(self, parcelID):

لا يمكن استلام طرد ما إلا في حال كانت الشاحنة في نفس الموقع الذي تريد الاستلام منه لذلك يكفي أن نعلم أي طرد نريد استلامه من هذا الموقع ونقوم بحذف موقع الاستلام من مصفوفة الاستلام بالإضافة إلى إضافة id الطرد الذي تم استلامه إلى طرود الشاحنة.

deliver(self, parcelID):

لا يمكن تسليم طرد ما إلا في حال كانت الشاحنة في نفس الموقع الذي تريد التسليم فيه لذلك يكفي أن نعلم أي طرد نريد تسليمه في هذا الموقع ونقوم بحذف موقع التسليم من مصفوفة التسليم بالإضافة إلى حذف id الطرد الذي تم تسليمه من طرود الشاحنة.

moveTruckTo(self, to):

تغيير موقع الشاحنة الحالي إلى الموقع to.

generatePossibleNextIntStates(self):

في هذا التابع نولد الانتقالات الممكنة من الحالة الحالية، حيث نولد المواقع الممكنة تبعا للاتجاهات الأربعة(أعلى، أسفل، يمين، يسار) ثم نتحقق من إمكانية انتقال الشاحنة لهذه المواقع عندها نخزن في مصفوفة intStates رقم يدل على الـ action وهو 0 في حالة الـ move بالإضافة إلى info تخص هذا الـ action وهي الـ position الذي ستنتقل إليه الشاحنة في حالة الـ move ثم نتحقق من إمكانية الاستلام في موقع الشاحنة عندها نخزن في المصفوفة الرقم 1 في حالة الاستلام والـ info هنا هي id الطرد ثم نتحقق من إمكانية التسليم في موقع الشاحنة عندها نخزن في المصفوفة الرقم 2 في حالة التسليم والـ info هنا هي id الطرد.

generateState(self, action, info):

تبعا للـ action الموجود نستدعي إما تابع move أو receive أو deliver من أجل الـ info الموجودة وذلك بعد نسخ الـstate الحالية.

show(self,num):

يتم طباعة الرقعة تبعا لتنسيق معين.

ucs(self):

نقوم بالتعامل مع priority queue يتم ترتيبها تصاعديا حسب كلفة الطريق من أجل كل state ويتم حساب كلفة الطريق حسب الـ action الذي نقوم به فإن كلفة الـ move أي الانتقال من موقع إلى آخر هي عبارة عن كلفة الطريق حتى الآن مضافا إليها 1+حجم مصفوفة الطرود للشاحنة، أما بالنسبة لكلفة الاستلام فهي عبارة عن كلفة الطريق حتى الآن مضافا إليها 1 أي وزن الطرد الذي تم استلامه في هذا الموقع، أم بالنسبة لكلفة التسليم فهي عبارة عن كلفة الطريق حتى الآن مطروحا منها 1 أي وزن الطرد الذي تم تسليمه وذلك بعد حساب جميع الحالات الممكنة من الحالة الحالية وتوليدها واحدة تلو الأخرى من أجل السرعة، ومن أجل كل حالة حتى يتم إدخالها إلى الـ priority queue نبحث عنها ضمن الـ priority queue في حال عدم وجودها نضيفها مع totalCost الخاص بها إلى priorirty queue أو في حال وصلنا إلى كلفة أقل من تلك الموجودة سابقا نستبدل الحالة حالية مع totalCost الخاص بها بتلك الموجودة مسبقا.

Class A_star:

Heuristic1:

يقوم التابع التجريبي الأول بحساب المسافة (كفروق إحداثية) ما بين موقع الشاحنة الحالي والموقع الهدف.

Heuristic2:

يقوم التابع التجريبي الثاني بحساب المسافة (كفروق إحداثية) ما بين موقع الشاحنة الحالي ومكان الاستلام مضروبا بـ (عدد البضائع أي وزن الطرود + 1 وهي كلفة الخطوة الواحدة) مضافا له المسافة ما بين موقع الاستلام وموقع التسليم مضروبا بـ (عدد البضائع في الشاحنة أي وزن الطرود + 2 وهي كلفة الخطوة الواحدة وكلفة الاستلام) وذلك من أجل كل طرد لم يتم استلامه وبالتالي لم يتم تسليمه بعد، أما في حال الطرود التي تم استلامها ولم يتم تسليمها

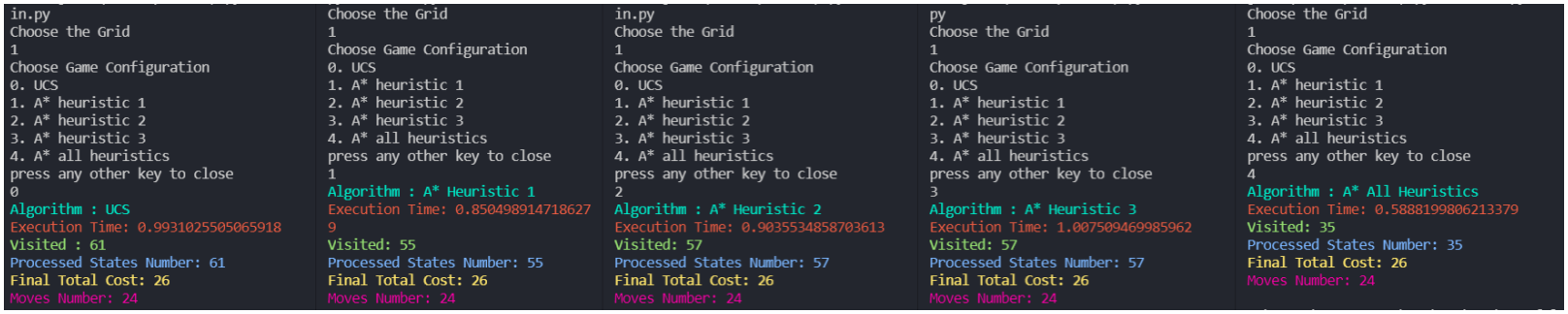
بعد فيكفي حساب المسافة ما بين موقع الشاحنة الحالي وموقع التسليم مضروبا بـ (عدد البضائع في الشاحنة أي وزن الطرود + 1 وهي كلفة الخطوة الواحدة)

Heuristic3:

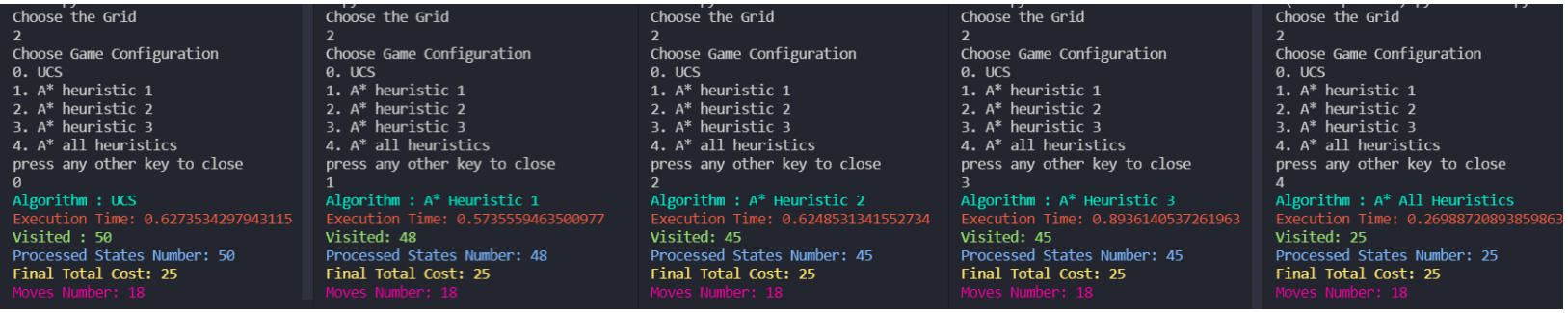
يقوم التابع التجريبي الثالث بنفس ما يقوم به التابع التجريبي الثاني ولكن من أجل قيم maximum.

مقارنات ونتائج:

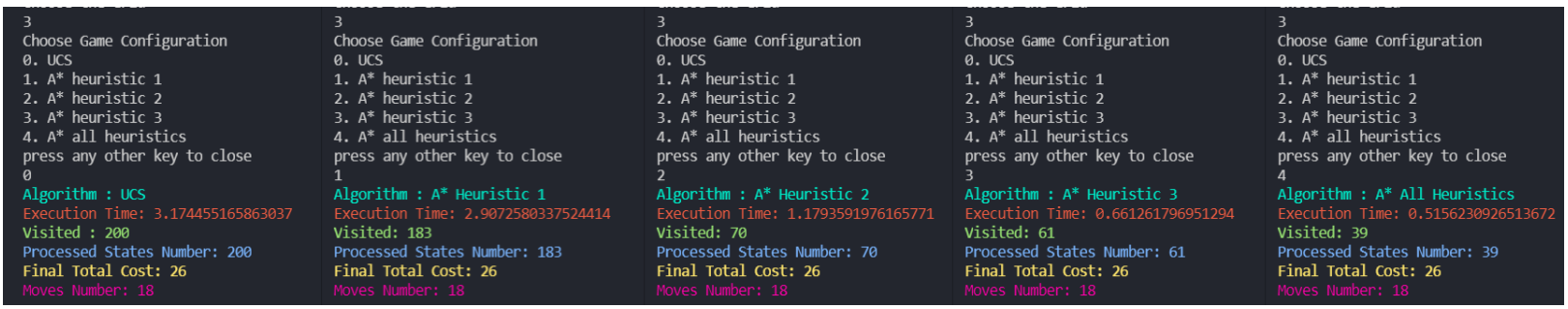
المثال الأول:



المثال الثاني:



المثال الثالث:



المسألة الثانية:

تمّ استخدام لغة بايثون في برمجة اللعبة.

يتكون المشروع من 3 كلاسات رئيسية:

1. BordWars
2. MenuGui
3. GameGui

BordWars:

تأخذ طول وعرض الرقعة، ونوع اللاعب الثاني (حاسب أو لاعب ثاني)، ويحوي رقعة اللعبة board، وهي عبارة عن مصفوفة ثنائية، تحتوي 'X' (player1)، 'O' (player2/PC) ، '-' (empty place).

X - - - - - - - - X
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
O - - - - - - - - O

شرح لأهم التوابع بهذا الكلاس:

Chick_win

يتحقق من فوز أحد اللاعبين أو تعادلهما في كل دور، حيث يحسب عدد X، وعدد O ويفوز اللاعب صاحب العدد الأكبر في حال كانت الرقعة ممتلئة، أو في حال لم تكن ممتلئة فيفوز اللاعب لم يتبقى من الخصم أي لاعب.

Is_full

يتحقق من امتلاء الرقعة.

Update_cell

يأخذ هذا التابع اندكسات cell، إذا كانت ضمن الفريق الخصم فتصبح ضمن فريق اللاعب.

Update_board

يأخذ هذا التابع اندكسات الـcell التي تم الانتقال/النسخ إليها، فيقوم بإيجاد مجاورات الـcell، واستدعاء update_cell من أجل كل مجاور، فإذا كان خصم يخزن بمصفوفة updated التي سترسل للواجهة لتحديثها.

Get_border

يأخذ هذا التابع cell ، ويقوم بإيجاد مجاوراتها (مجاور أول/ثاني) حسب عدد يمرر إليه.

Next_turn

يعيد اللاعب صاحب الدور التالي.

Is_end

يتحقق من انتهاء اللعبة.

First_move

يقوم بنسخ اللاعب للموقع الجديد

Second_move

يقوم بنقل اللاعب للموقع الجديد(حذفه من موقعه الحالي، وإضافته للموقع الجديد).

Move

يتحقق من أن الخلية التي أريد الانتقال إليها ضمن المجاور الأول أو الثاني للخلية التي أريد الخلية التي أريد الانتقال منها، وبالتالي يتم استدعاء first_move أو second_move، وإلا يقوم برمي Exception.

Next_state_at

يقوم بإيجاد كل الحركات المتاحة من أجل cell معينة ولتكن x، حيث يتم تخزينها ب dictionary حيث الـkey هو الخلية y التي تم الانتقال إليها من الخلية x .

Next_state

يقوم بإيجاد كل الحركات المتاحة من أجل كل خلايا اللاعب عن طريق استدعاء next_state_at لكل منها، حيث يتم تخزينها ب dictionary , الـkey هو خلايا اللاعب والـvalue عبارة عن dictionary يحوي الانتقالات المتاحة من أجل الخلية x.

Eval

يمثل تابع التقييم لـminmax، ويقوم بحساب ناتج طرح عدد خلايا اللاعب من عدد خلايا خصمه.

Minmax

يستقبل هذا التابع state من نمط BoardWar و depth للتوقف عند عمق معين و alpha,beta للقيام بالتحسين و enable:Boolean لتفعيل التحسين أو تعطيله .

State.next_states() يعيد dictionary يحوي على مجموعة من الـdictionaries مجموعها تمثل الـnext_states وبتطبيق الخوارزمية على كل واحدة منهم يتم إيجاد أفضل انتقال .

يعيد التابع tuple(val1,val2) بحيث تمثل val1 الانتقال من خانة أولى إلى خانة ثانية و القيمة val2 تمثل قيمة الـevaluation .

MenuGui:

يقوم بعرض الواجهة الأولى، والتي تتيح للاعب إدخال عرض وطول رقعة اللعب، (في حال لم يدخلهم فيتم أخذ قيم افتراضية)، وتحديد في حال نريد اللاعب مع الحاسب أو لاعبين.

GameGui:

يقوم بعرض الواجهة الثانية، والتي تمثل واجهة اللعب، حيث الرقعة عبارة عن أزرار، عند الضغط على زر من فريق اللاعب فيتم تلوين الأزرار المجاورة له والتي يمكنه النسخ أو الانتقال إليها.

وفي أعلى الواجهة يتم عرض عدد لاعبين بعد كل حركة.

فيما يلي شرح لأهم توابع هذا الـclass:

Game_board && initiliaze

مسؤلان عن إنشاء واجهة اللعبة.

Update_color_at && setColor && delete_border

مسؤولون عن تحديث أزرار الألوان

Update_gui

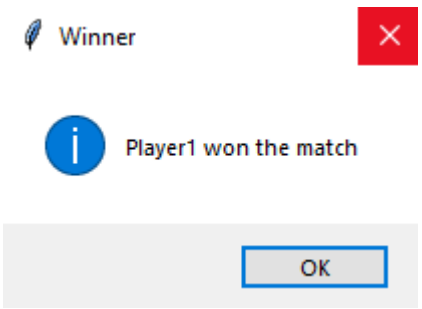
يتحقق إذا الزر المضغوط ضمن فريق اللاعب فيخزن ب pressed_cell، وإلا كان خلية فارغة فيتم استدعاء التابع move من الكلاس bordWars، ويحدث ألوان الواجهة.

Get_move

يتم استدعاؤه عند ضغط زر بالواجهة، وبدوره يستدعي التابع update_gui و إذا كانت اللعبة مع الحاسب، فبعد تمام دور اللاعب يتم استدعاء التابع minmax لاختيار أفضل حركة للحاسب، ويستدعي update_gui من أجل هذه الحركة.

Check_win

يتحقق من انتهاء اللعبة ليعرض messageBox بالنتيجة:



فيما يلي نتائج مقارنة بين minmax و minmax مع alpha_beta:

Level 1:

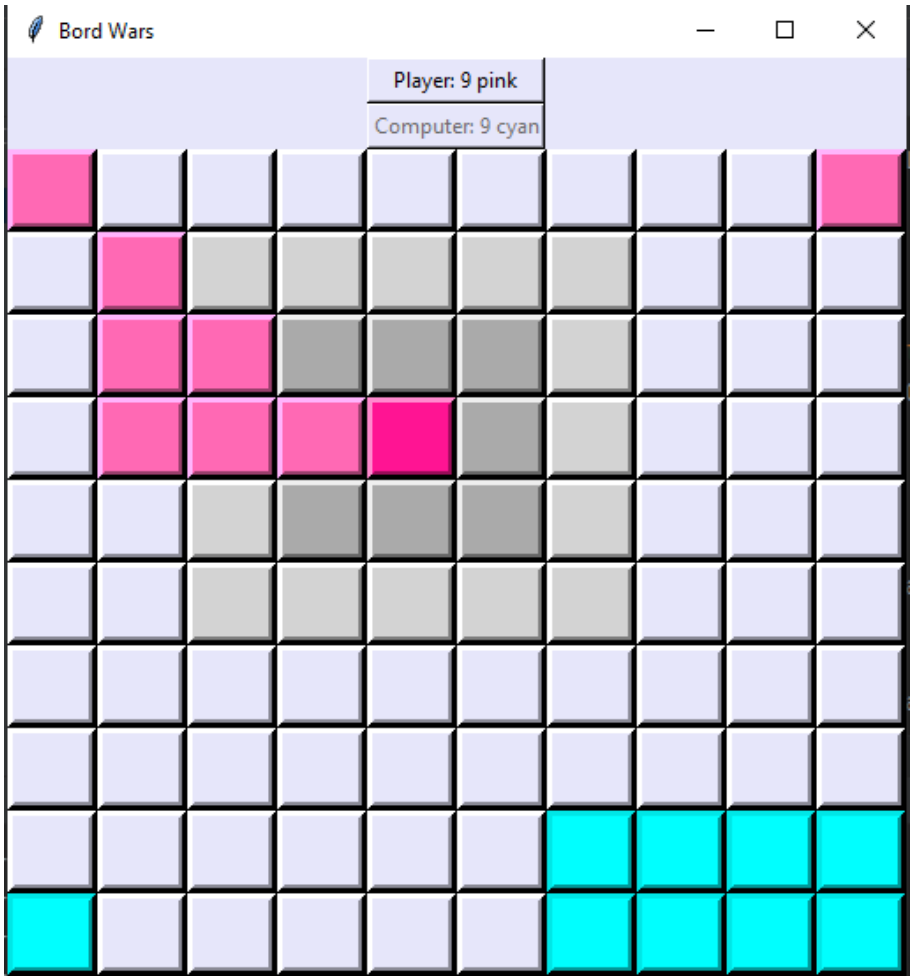
the time needed for minmax is: 0.004995822906494141

the time needed for minmax with alpha_beta is: 0.0029976367950439453

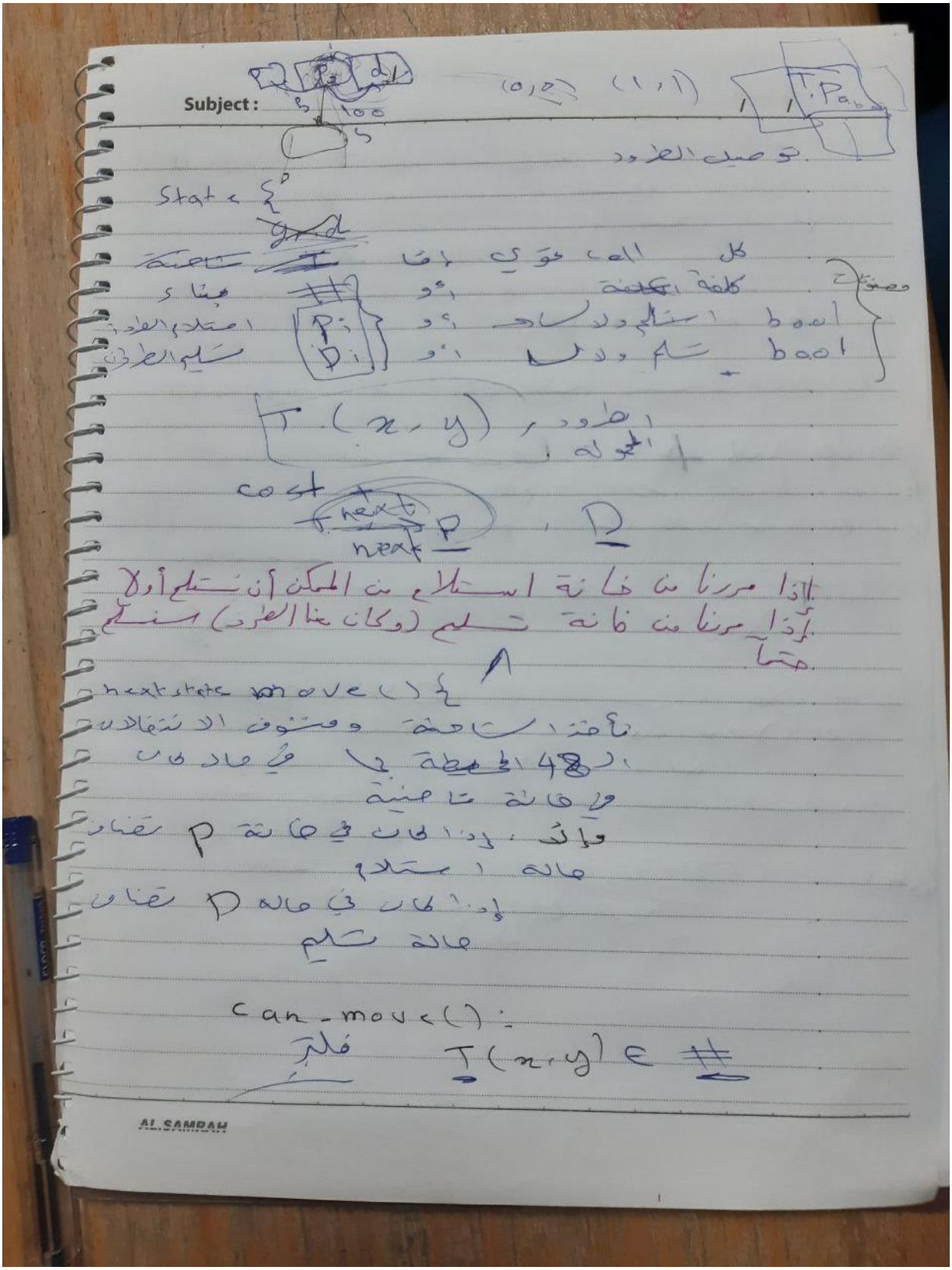
Level 3:

the time needed for minmax is: 9.240488767623901

the time needed for minmax with alpha_beta is: 1.316877841949463



أوراق المناقشات:



Subject: تشبيه نفقة البداية   / /

1) state: T عبارة عن x,y + طرود

P: عبارة عن مراكز التسليم / الاستلام

D: " " " الاستلام (~~التسليم~~) تسليم

التكلفة:

2) move: في حال خانة فارغة يكون 4

" " " استلام أو تسليم 5 أو 4

3) can_move: المرور على خلية ليست محكومة بناء

cost: المستقبل: ~~أ~~ كلفة الطرود

الاستلام 5 الطرد في حال وجوده

التسليم: - الطرد في حال عدم وجوده

الحالة الابتدائية ؟؟

الحالة النهائية: تسليم جميع الطرود + عودة T إلى مكان البداية

الهدف

التقريب الأول: حساب أقصر من نقطة البداية

الثانية: حساب كلفة توصيل الطرد

في حال لم يتم استلامه

أو الطرد الأول الموجود في الساحة في حال

تم استلامه

الثالثة: حساب ~~الكلفة لكل الطرود أقل كلفة~~

كلفة أي طرد

لا تسليم

Subject : / /

state : grid فيها 0 أو 1 (الأحمر / الأزرق) لتمثيل اللاعب
أو فارغة

~~next~~ move(خلية لاعب , أحد الخلايا (المجاورة أو التي بينها خانة))

next states{
لكل خلايا اللاعب
مع كل الخلايا المجاورة لخلايا اللاعب
أو التي بينها خانة
can move()
ما يكون في خليتين فوق بعض

end : امتلاء جميع الخلايا أو ~~نهاية~~
win : بقاء لون اللاعب و اختفاء لون الخصم
~~lost~~ lose : بالعكس

AL SAMRAH