المسألة الأولى:

في هذه المسألة قمنا بتمثيلها باعتبارها مسألة state space searsh

الذي يمثل في مسألتنا حالة (state) بنيتها هي :

مصفوفة board ثنائية من نوع string

مصفوفة بولیانية validP تعبر عن حالة استلام الطرد

مصفوفة بولیانية validD تعبر عن حالة تسليم الطرد

استخدمنا المصفوتين السابقتين من نوع بولياني لتقليل العبئ على الذاكرة و التسريع من التنفيذ

الكلاسات المتوفرة:

Class State

وهو الذي عرفنا ضمنه بنية المسألة

التوابع:

1- البواني الخاصة بالكلاس State() : هناك أكثر من باني مع اختلاف البارامیترات و ذلك حسب الحاجة لاستخدامها و يتم من خلالها عمل clone
2- تابع getcost() : يرد الكلفة فقط
3- تابع deletT() : هو تابع لتتغير موقع ال T (الشاحنة) مع معالجة الوضع السابق ضمن الرقعة
4- تابع move() : لحركة الشاحنة مع تخزين مسار الحركة ضمن Arraylist movement حيث يتم تخزين الحركة على شكل string مثال (0,1,m)
5- تابع recdev() : يحدد عمل الشاحنة إما استلام P أو تسليم D وذلك بفحص النقطة المرسلة كبارميتر إذا كانت Pفهذا يعني استلام وإذا Dهذا يعني أنه تسليم مع تخزين مسار الحركة ضمن ال movement
6- تابع process() :تابع يعالج عملية

أي إذا كانت # لانستطيع أن نقوم بأي عملية

إذا النقطة يلي أرسلتها هي نفسها ال truck هذا يعني يوجد عملية يا أما تسليم أو استلام إلا انتقل

7- تابع checkprocess() : أمرر له نقطة ويفحص إذا كان يوجد عملية أم لا وتكون العملية إما (حركة أو تسليم أو استلام)

8- تابع isfinish() : للتحقق من ثلاث أشياء:

-انتهاء الاستلام من جميع الأماكن

-انتهاء عملية التسليم

-التحقق من الوصول لنقطة البداية(أي نفحص ال validPو validD إذا امتلأت خانتهم ب true)

9- تابع nextallprocess() : يرد الأولاد وهم عبارة عن مصفوفة state مع تحديد الخطوة التالية (جنوب,شمال,أسفل,أعلى)أو عملية (استلام,تسليم)

10- تابع findcost() : القيمة الابتدائية لل cost=1 وهو لحساب الكلفة يمر التابع على مصفوفة validP ويرى إذا كان هناك طرد تم استلامه ولم يتم تسليمه فيتم زيادة ال cost بمقدار واحد

11- تابع findcost(oldcost) : تابع لحساب ال cost الحالية مع ال cost القديمة

12- تابع heuristic1(): نحسب البعد بين نقطة البداية والموقع الحالي للشاحنة

13- تابع heuristic2(): حساب الكلفة التقديرية لتسليم طرد معين آخذين بعين الاعتبار جميع المسافات التي يجب أن نقطعها ومنه يوجد ثلاث حالات لنعالجها :

الحالة الأولى: لم يتم استلام أو تسليم الطرد : و بهذ الحالة نحسب المسافة بين الشاحنة و موقع ال p0 ثم نحسب الفرق بين ال p0 و d0 و نجمعهم و نأخذ بعين الاعتبار قيمة ال cost عند الحركة أو عم الطرود المستلمة

الحالة الثانية : تم استلام الطرد و لم يتم التسليم : هنا يتم فقط حساب الفرق بين الشاحنة و موقع التسليم أي d0

الحالة الثالثة : في حال تم الاستلام و تم التسليم تكون قيمة التابع 0

14- تابع heuristic3():

نفس فكرة التجريبية الثانية لكن الاختلاف أننا نطبق عملية التجريبية الثانية على كل الطرود ومن ثم أخذ ال max بين القيم التجريبية

15- تابع topoint(): في مصفوفة ال movement نقوم بتخزين العمليات على شكل string وذلك لتخفيف العبئ على الذاكرة مثال (0,1,M) فهذا التابع يقوم باستخراج احداثيات النقطة من هذا الستريغ مثلا هنا x=0 و y=1

Class Point

يتم فيه إنشاء النقطة

Class Controller

يتضمن object من كلاس State

واستخدمنا فيه priority Queue من أجل خوارزمية Ucs و A* لإ

واستخدمنا فيه ArrayList نوعها hashcode لوضع جميع العقد المزارة فيها

توابع االكلاس هي :

1- البواني

2- خوارزمية UCS : لإيجاد الحل بالكلفة الأقل عن طريق استخدام تابع الكلفة findcost()

3- خوارزمية A* : لإيجاد الحل بشكل مسرع عن طريق استخدام التوابع التجريبية المذكورة سابقا

Class Main

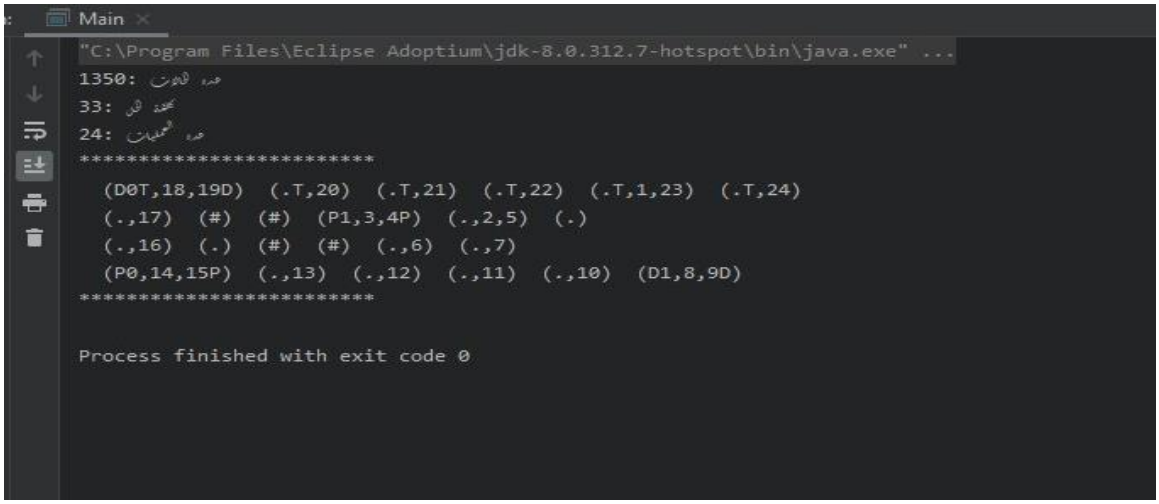
تم تعريف ثلاث رقع مختلفة بالشكل للتجريب , و أخذنا object من كلاس ال State وأيضا object من كلاس ال Controller لتنفيذ الخوارزميات وأخيرا استدعاء الخوارزمية المراد تنفيذها

Class HashCodes

أعرف فيه المتغيرات التي احتاجها لمقارنة حالتين وهم board ومصفوفتي ال validP و ال validD وموقع الشاحنة والكلفة

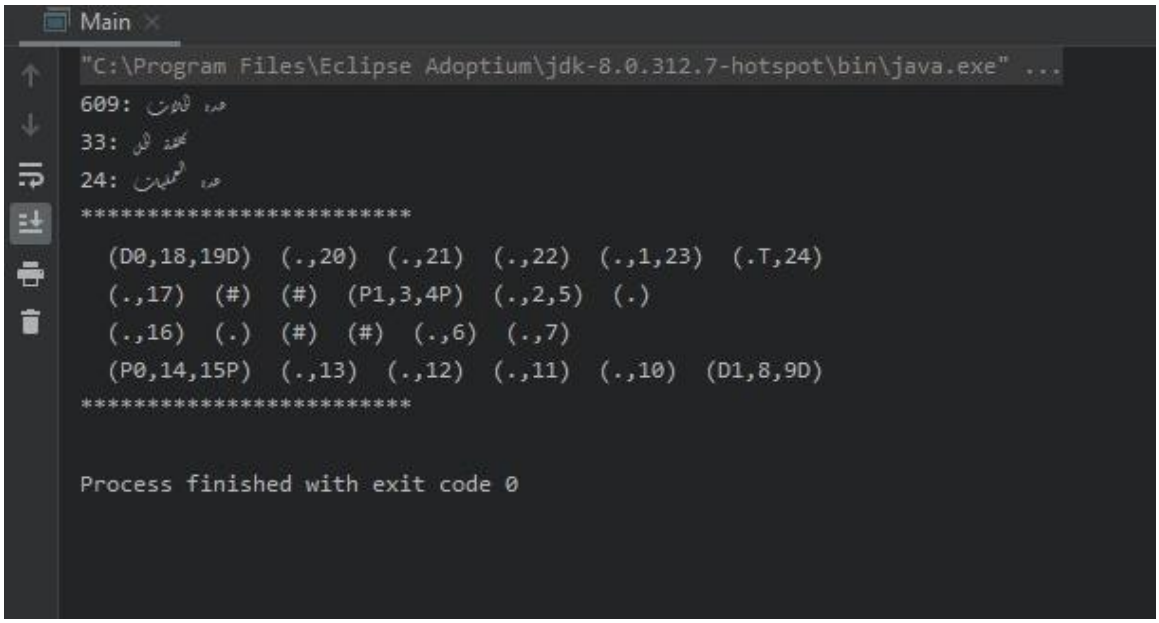
حيث نمرر للباني state ليأخذ منها القيم المهمة للمقارنة وهم (board,validP,validD) وعرفنا تابع boolean hashset(State state) وهنا يتم ال state الممرة كبارميتر مع ال state الموجودة في الباني

UCS حساب البعد بين الجذر والهدف مع حساب الكلفة

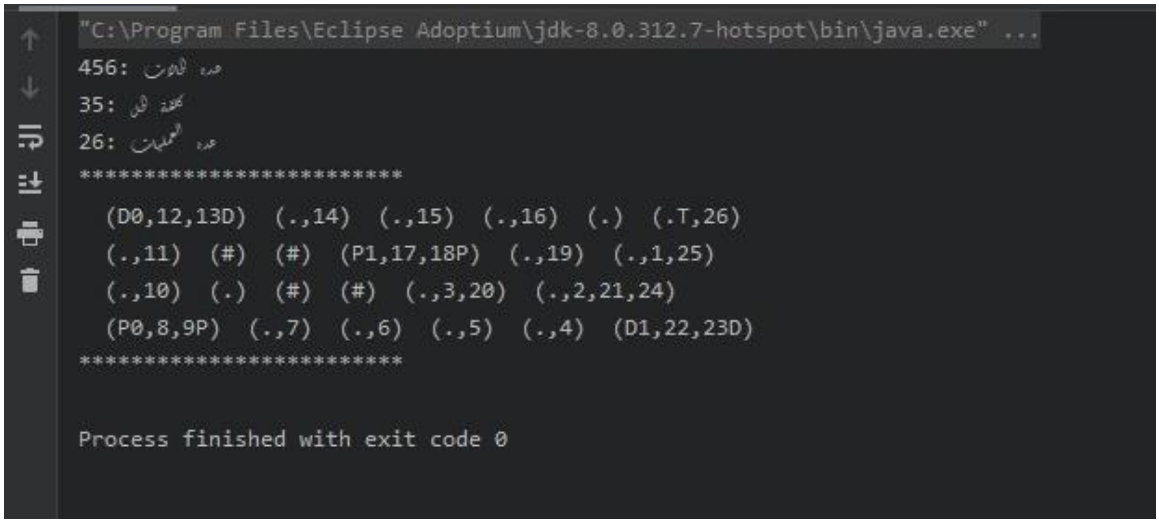


A* حساب البعد بين الجذر والهدف مع حساب الكلفة ومع وجود التابع التجريبي

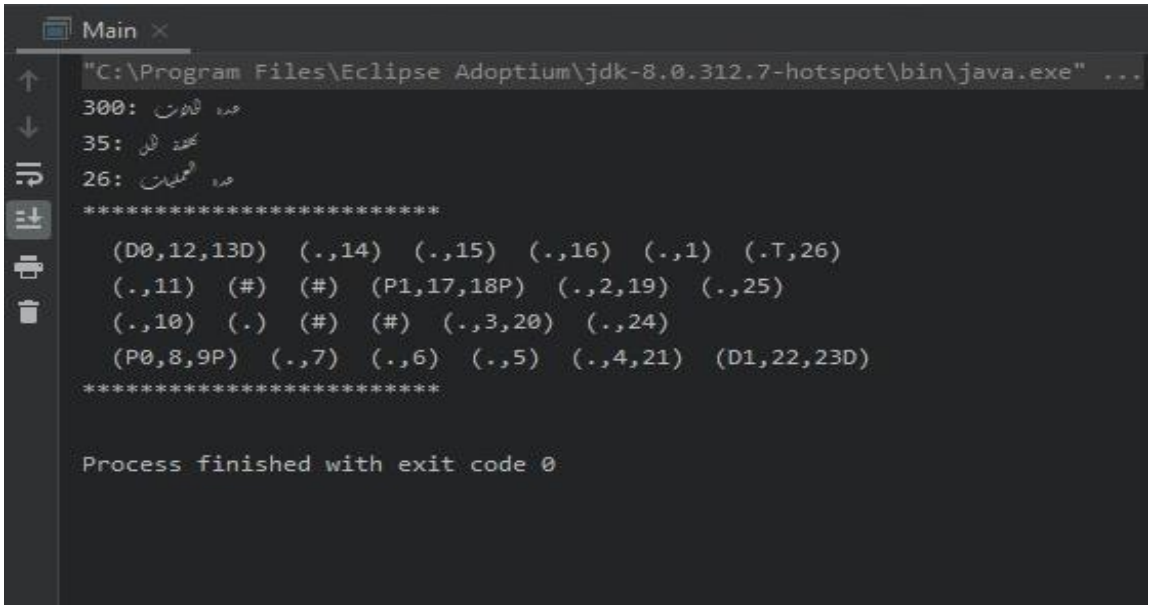
باستخدام التجريبية الأولى:



باستخدام التجريبية الثانية:



باستخدام التجريبية الثالثة:



المسألة الثانية: (BlopWars)

Class State

التوابع المستخدمة ضمن هذا الكلاس:

1. البواني:
- الباني الأول: أعطيه البعد وهو ينشأ لي مصفوفة فارغة
- الباني الثاني: أعطيه البعد مع مصفوفة وهو بيجبلي اياها
- الباني الثالث: أمرر له state بكامل معلوماتها فهذا الباني يرد لي copy عن هذه ال state
2. تابع ال ()move : أمرر له النقطة الحالية current والنقطة الهدف target
بداية نمثل عملية الانسحاب بتنقيص واحد أي عندما ينسحب الخصم تمتلئ كل المربعات الفارغة بالخصم المقابل مثلا إذا كنت اللاعب B وانسحبت فف عن طريق التابع pass يعبي كل المربعات R
أي امرر تابع ()move الخصم وإذا انسحب امرر له تابع ال ()pass وإلا
اختبر الخانة هل هي فارغة أي "-" ثم تتم عملية النقل ثم نختبر إذا تم الانتقال بخطوة أو خطوتين إذا خطوة ندع النقطة كما هي ثم نقوم بعملية النسخ
أما إذا خطوتين ادع النقطة الحالية فارغة ثم نستدعي تابع ()attack ثم نستدعي تابع ()score كي يحسب كم صار عدد ال (R,B)
3. تابع ال () attack : بهذا التابع يتم البث عن جميع الخانات المجاورة للعقد
4. تابع ال ()chechmove :
   - إذا انسحاب يرد true
   - إذا خرجت خارج حدود المصفوفة يرد false
   - إذا فراغ او هاشتاغ يرد false
   - إذا التارغيت كان غير فارغ أو كان يساوي الهاشتاغ يرد false
   - إذا تحركت أكثر من حركتين يرد false
5. تابع ()isFinish :

   هو تابع يحدد وقت انتهاء اللعبة

   إذا كانت هناك خانة فارغة هذا يعني أن اللعبة لم تنتهي

   وإلا إذا امتلأت جميع الخانات هذا يعني أن اللعبة انتهت

6. تابع ()eval :
   هو تابع يرد لي قيمة, عندما تنتهي اللعبة
   أحتاجه عند استخدام خوارزميات MinMax ,alphaBeta
7. تابع ()nextallmove :

8. تابع AllowMovement() :

Class Hashcodes

التوابع الموجودة ضمن هذا الكلاس :

1. الباني Hashcodes() : نمرر له state ليأخذ منها المصفوفة.
2. تابع ال Hashset() : أمرر له state وهو يرى إذا كانت المصفوفة وال state متساويين أو مختفتين

إذا كان كانوا متماثلتين يرد false .

وإلا يرد True .

Class Game

التوابع الموجودة ضمن هذا الكلاس:

1. التابع runGame(): التابع الذي يتم استدعاءه في ال main كي يتم تشغيل اللعبة.
2. التابع startGame(): لدي هنا arraylist visit أخزن فيها العقد المزارة حسب ال typeplayer وهو إما يكون B أو يكون R
   ثم أدخل بحلقة while حتى تنتهي اللعبة
3. التابع playerMove(): هنا ندخل إحداثيات العقدة الحالية current والعقدة الهدف target .
4. التابع computer() : له ثلاث مراحل (easy,medium,hard) .
5. التابع easy() : تابع random يختار مكان عشوائي من حاله
6. التابع hard(): تابع أستدعي فيه تابع ال max مع تمرير العمق له .
7. التابع medium() : نفس تابع ال hard() لكن بعمق أقل من العمق الممرر لتابع ال hard() .

التقسيم:

المسألة الأولى : لين وماريمار وهلا