در ابتدا لازم به ذکر است میتوانید مخزن گیت هاب این پروژه را بر روی لینک زیر ببینید :

<https://github.com/mmaghajani/AI_2>

مساله هشت وزیر

برای این مساله تابع هدفی به صورت زیر تعریف شده است :

تعداد جفت وزیر هایی که همدیگر را تهدید میکنند.باید توجه داشت که این میزان هنگامیکه به یک حالت پایانی برسیم برابر صفر می‌شود . در نتیجه تابع هدف ما مقدار بالا را در یک منفی ضرب میکندچرا که ارزش حالت‌هایی که تعداد بیشتری وزیر در آن‌ها تهدید می‌شود برای ما کمتر است و این تابع هدف بیانگر ارزش راه حل نیز میباشد.

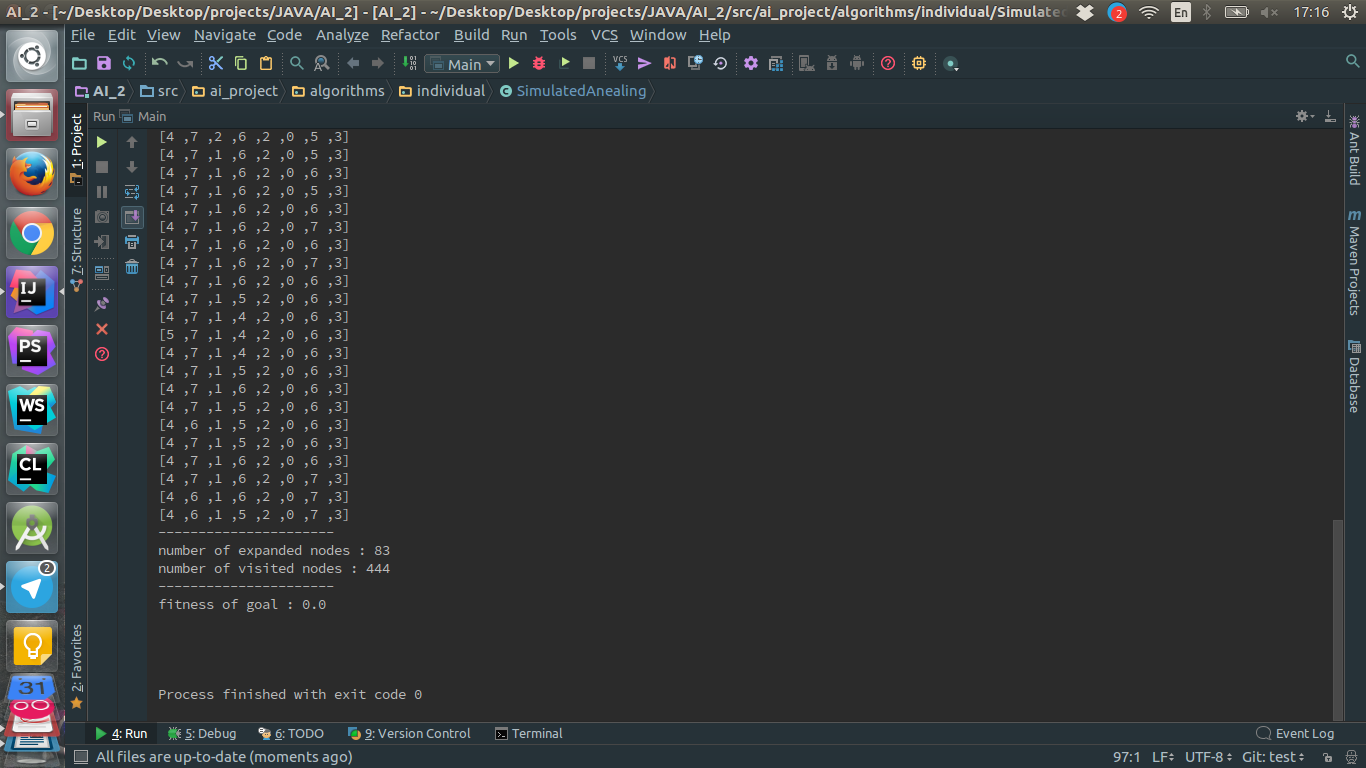
در این مساله حداکثر تعداد گام ها برابر با 10000000 گام در نظر گرفته شده است

روش ارایه حالت در این مساله به صورت یک آرایه هشت تایی میباشد که در هر درایه آن نماینده یک ستون در صفحه شطرنج میباشد و شماره سطری که وزیر در آن است در درایه مورد نظر قرار میگیرد.همچنین حالت اولیه برابر حالت زیر است :

[0,1,2,3,4,5,6,7]

الف )

برای این قسمت سه روش کاهش در نظر گرفتیم :

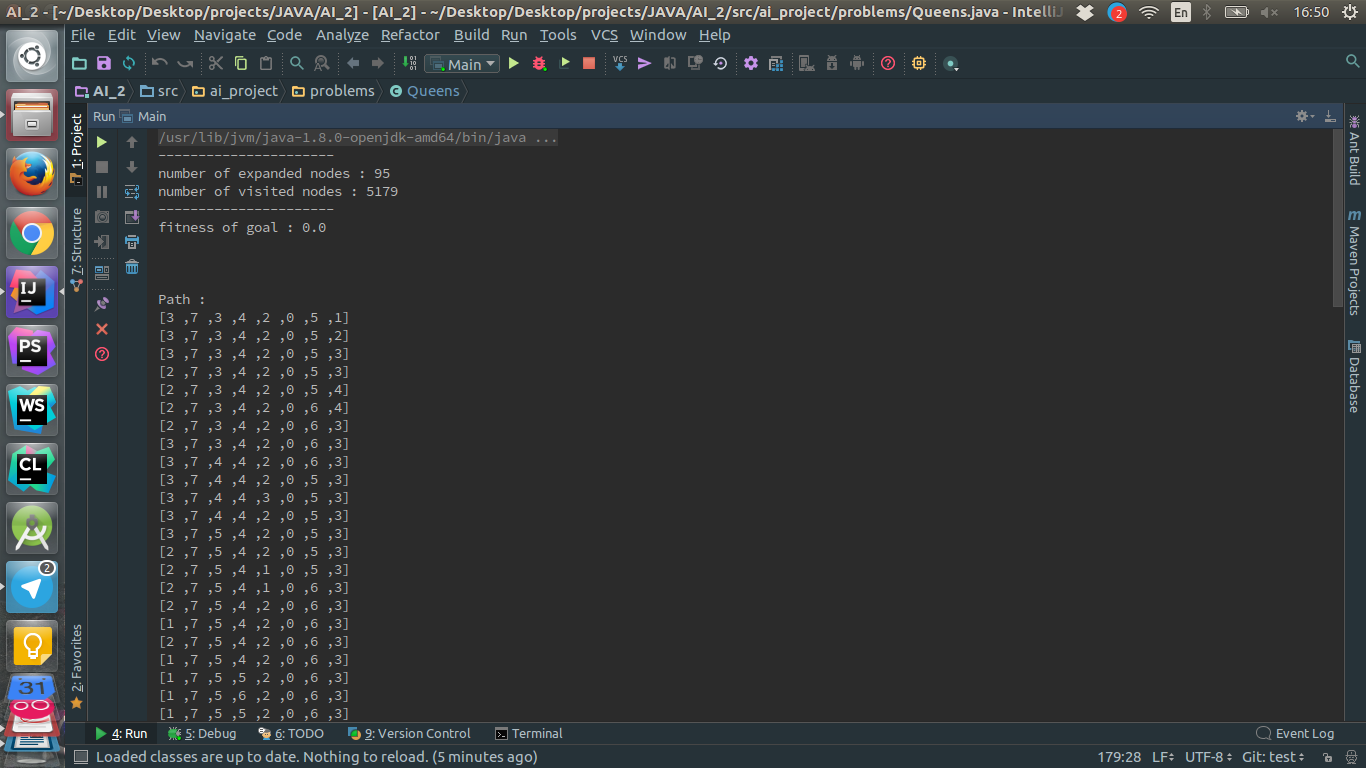
متغیر step بیانگر زمان ( تعداد گام مساله )‌ میباشد بنابراین روش‌های زیر درنظر گرفته شده است :

1. :

خروجی نمونه در بالا آورده شده است.

در این روش میانگین تعداد راس های expand شده حدود ۱۰۰۰ راس بود.همچنین همواره به جواب نهایی دست پیدا میکردیم

در این حالت خروجی نمونه به صورت زیر است :

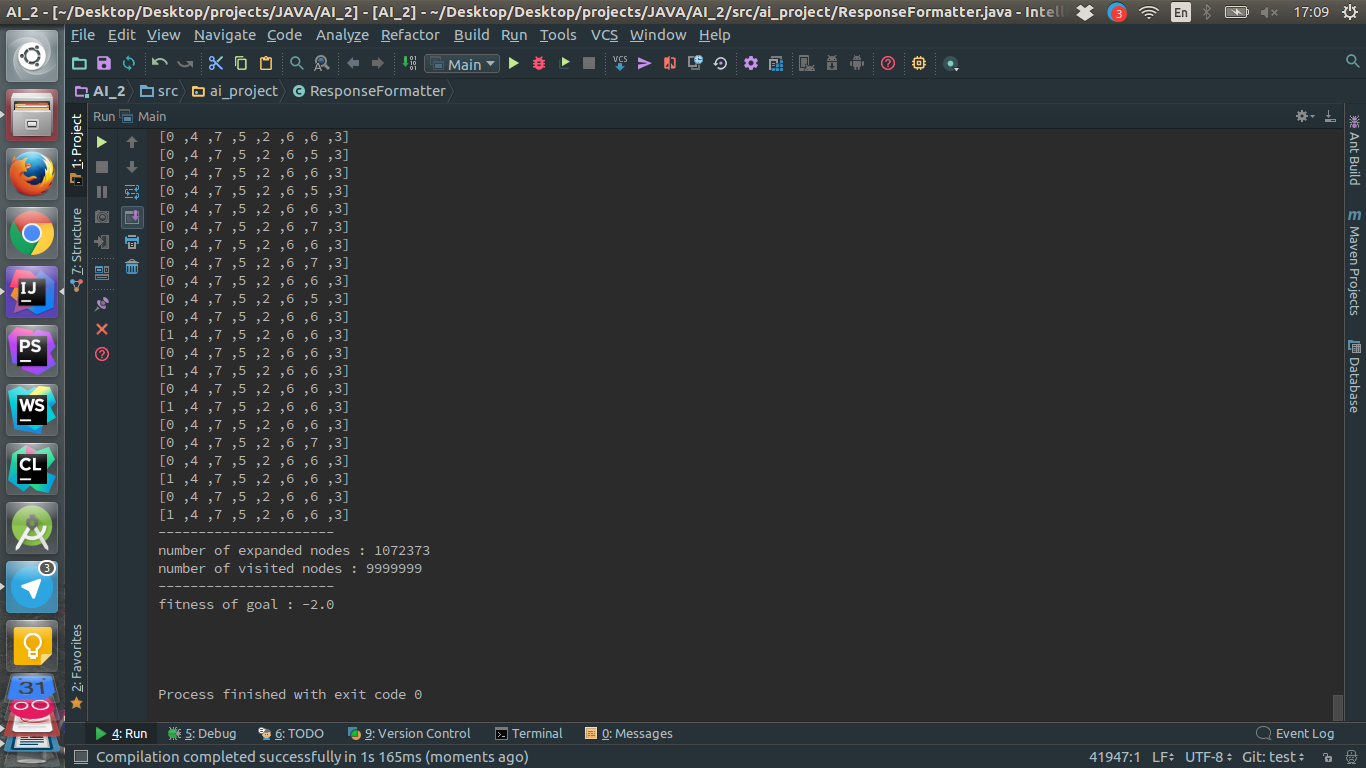


با این روش به حالت پایانی زیر ( در یکی از اجراها ) رسیدیم :

[2,6,1,7,4,0,3,5]

در این روش میانگین تعداد راس های expand شده حدود ۳۰۰۰ راس است و تعداد راس های مشاهده شده حدود ۲۰ الی ۳۰ هزار است.

خروجی نمونه :



در این روش با چندین بار اجرا هیچ گاه به جواب نرسیدیم و ارزشمندترین راه حلی که بدست آوردیم ارزش آن 1- بود و این کار با expand کردن حدود یک میلیون راس بدست میآمد که میزان بالایی ست.میانگین ارزش راخ حل‌ها حدود 2- بود.

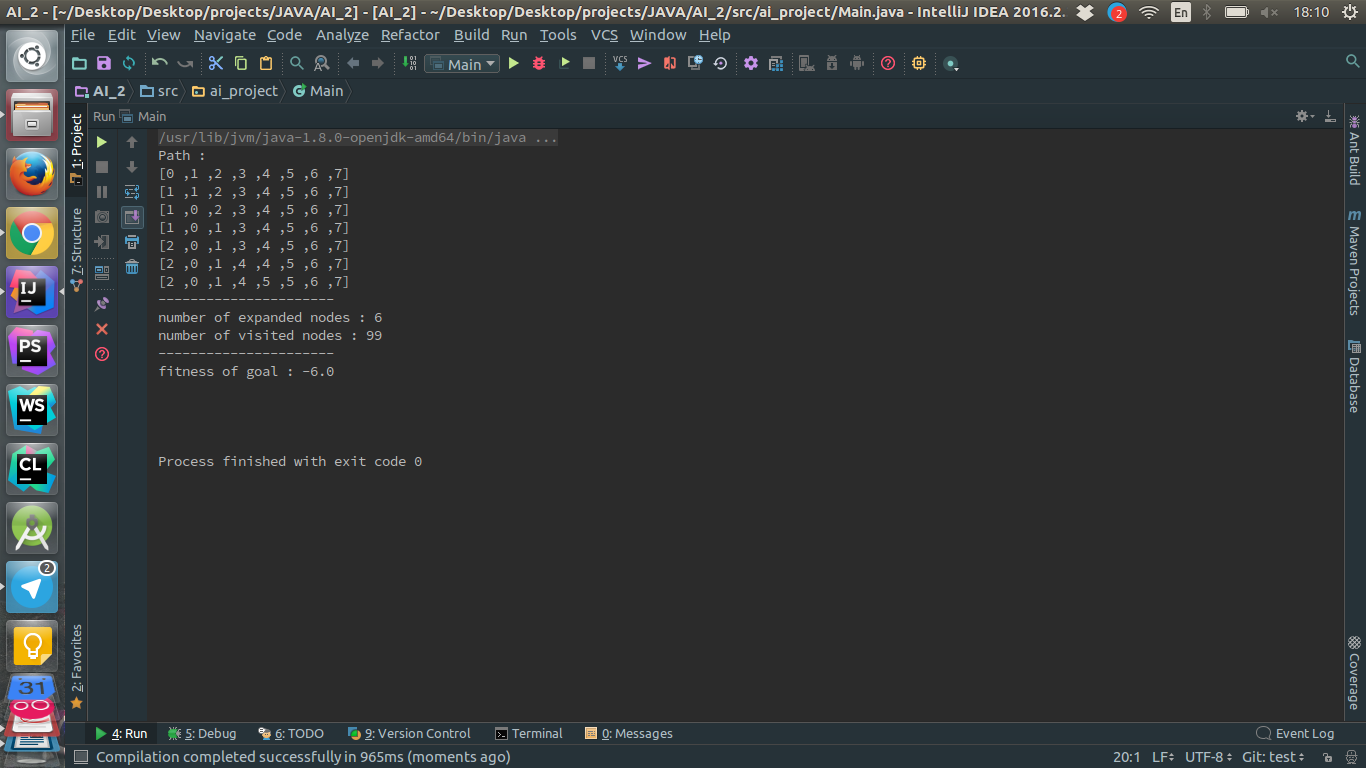
نتیجه : روش اول در مجموع بهترین روش میباشد زیرا تعداد راس های کمی را گسترش میدهد و همچنین همواره به جواب میرسد بعد از آن روش دوم بهتر است و سپس روش سوم با توجه به خطی بودن آن اصلاً مطلوب نیست.

ب )

حال به سراغ اجرای دیگر الگوریتم ها میرویم:

1. تپه نوردی معمولی :

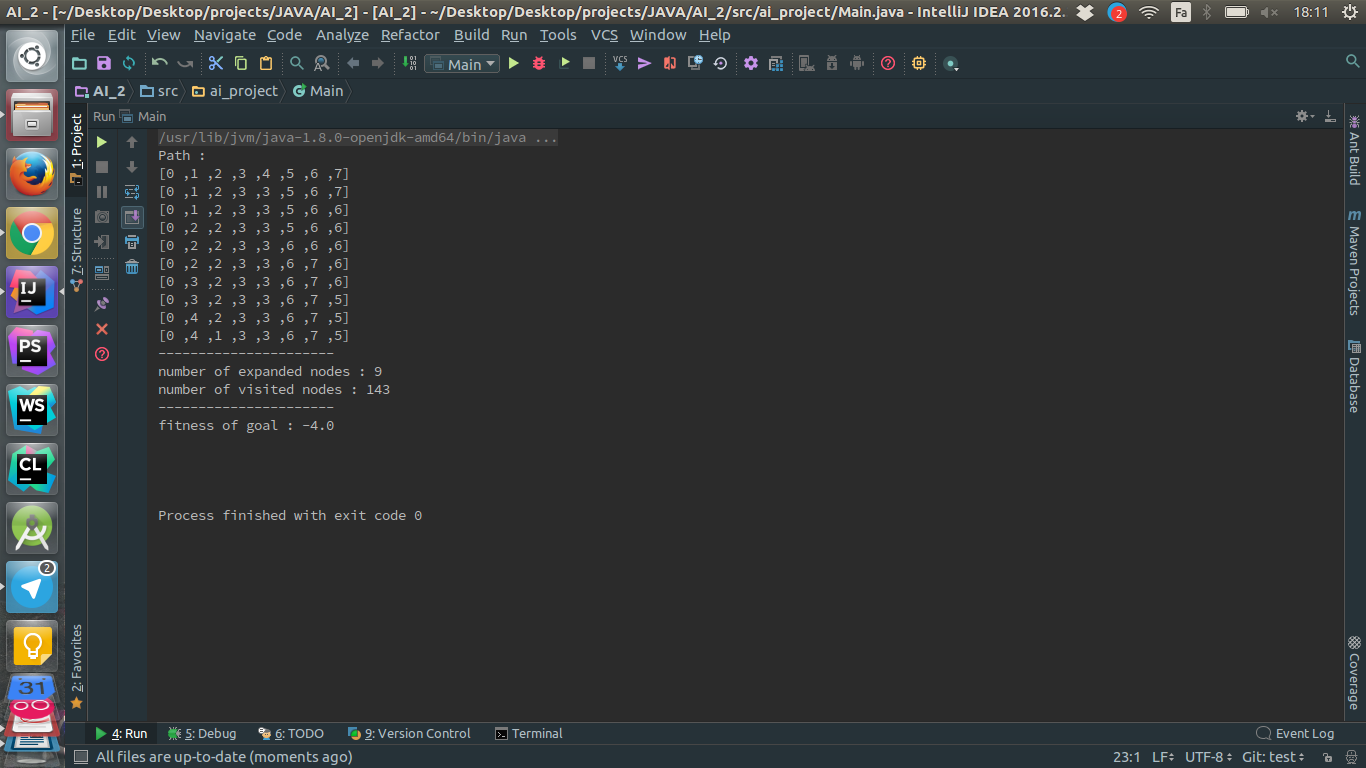
در این حالت خروجی به صورت زیر است :



همانطور که پیداست فوراً در یک بهینه محلی گیر میکند و جواب با‌ارزشی هم بدست نمی دهد.

1. تپه نوردی تصادفی

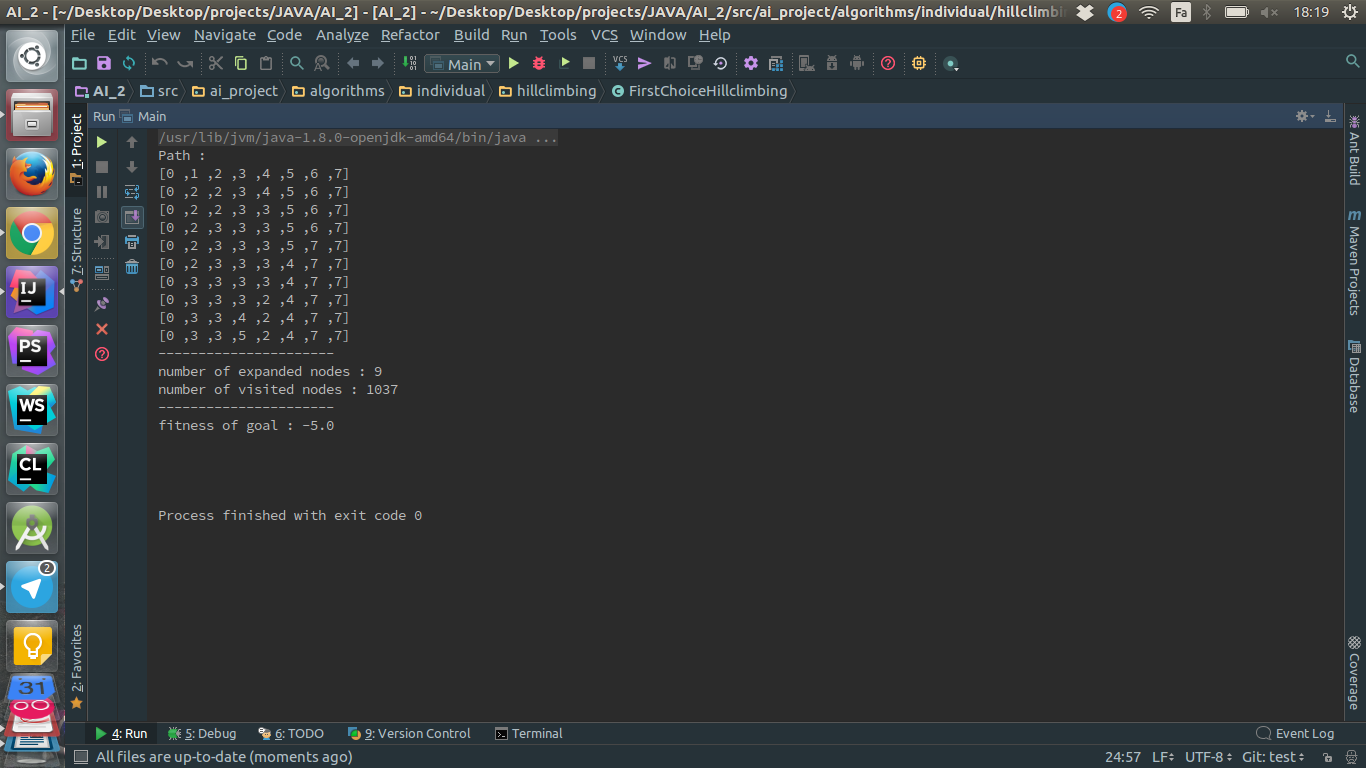
در این حالت یکی از خروجی ها به صورت زیر است :



همانطور که معلوم است گاهی اوقات نتایج ضعیف‌تر از حالت معمولی میدهد اما عموما به صورت میانگین ارزش جواب حدود 4- میباشد و بهترین نتیجه حاصله 3- بود.تعداد راس های گسترش داده شده نیز حدود ۹ الی ۱۰ میباشد.

1. تپه نوردی اولین انتخاب

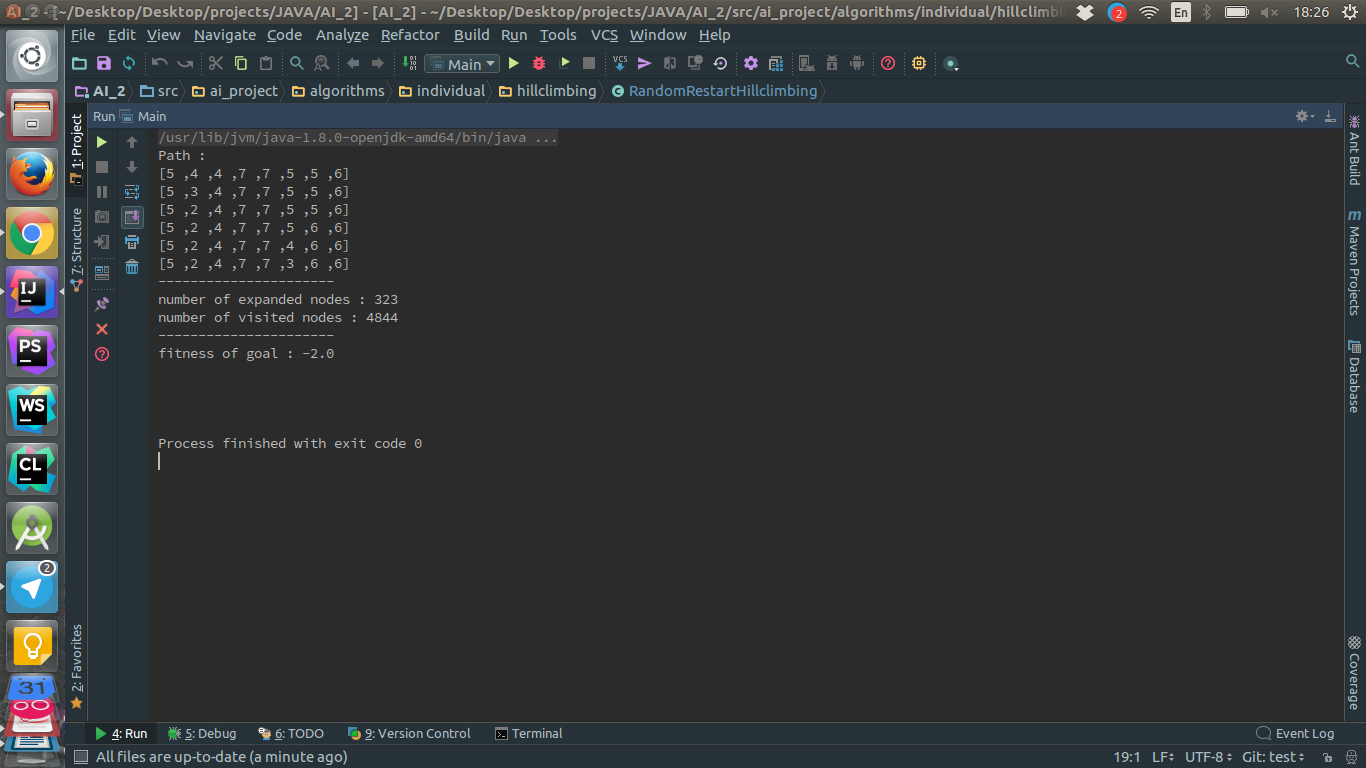
در این حالت تعداد حالات بعدی که به صورت تصادفی تولید می‌شود توسط ثابت RANDOM\_RATE مشخص میگردد که برای این الگوریتم برابر ۱۰۰۰ در نظر گرفته شده است و اگر تعداد همسایه های تولید شده از این تعداد بیشتر شد الگوریتم این حالت را به عنوان گیر افتادن در یک بهینه محلی تلقی میکند.میانگین تعداد گره‌های گسترش داده شده در این حالت حدود ۱۰ گره است و یک خروج ینمونه به صورت زیر است :

میانگین ارزش راه حل‌های تولید شده -۵ بود و بهترین جواب تولید شده ارزشی برابر با 4-داشت که در‌واقع از الگوریتم قبلی بدتر میباشد.

1. تپه نوردی با شروع مجدد تصادفی

در این حالت تعداد شروع مجدد ها توسط ثابت RESTART\_LIMIT کنترل می‌شود که برابر با ۱۰۰ است.

یکی از خروجی های نمونه به صورت زیر است :

در این روش ارزشمند ترین جوابی که تولید شد همان جوا بهینه بود و در بین تمامی الگوریتم های تپه نوردی تنها نسخه‌ای که موفق به تولید جواب بهینه شد همین الگوریتم بود .میانگین ارزش جواب‌ها نیز بین 2- تا 3- بود که از این حیث نیز بهترین الگوریتم تپه نوردی به شمار میرفت و همجنین میانگین گره‌های گسترش داده شده حدود ۳۰۰ گره بود که تعداد بیشتری در مقایسه با بقیه نسخه های تپه نوردی میباشد.

نتیجه : همانطور که دیدید در بین الگوریتم های تپه نوردی ، الگوریتم با شروع مجدد تصادفی در مجموع بهترین نتایج را بدست میداد و کمتر در بهینه های محلی گیر می‌افتاد اما الگوریتم سرد کردن تدریجی با تابع کاهش t = |sin(step)|/step حتی از این الگوریتم بهتر عمل میکرد زیرا همواره جواب بهینه را بدست میداد اما از لحاظ تعداد گره‌های گسترش داده حدوداً سه برابر الگوریتم تپه نوردی با شروع مجدد تصادفی گره گسترش میداد که اندکی آن را کند کرده بود.

مساله حل معادله :

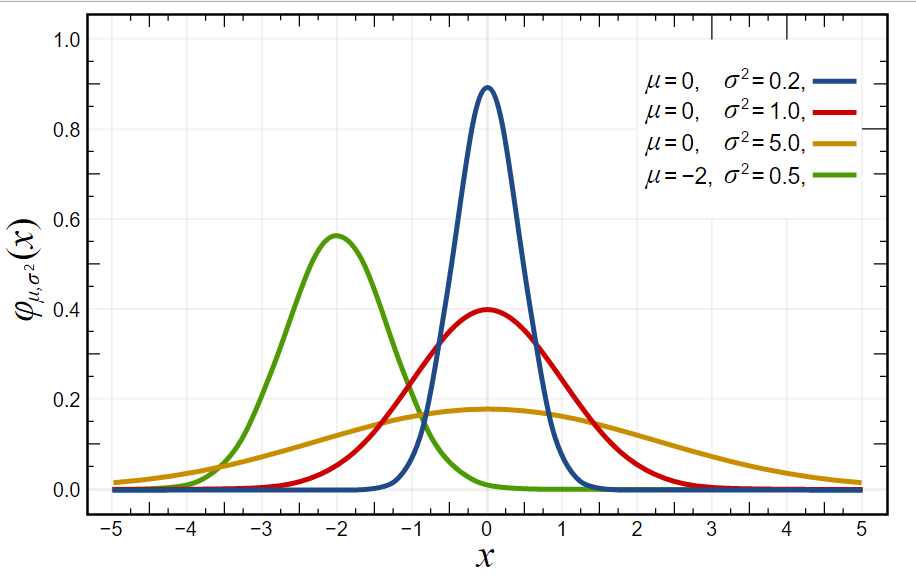
در این مساله ابتدا معادله را به حالت استاندارد درآوردیم یعنی همه جملات را به یک طرف تساوی بردیم در نتیجه برای پیدا کردن جواب باید به دنبال صفر کردن تابع بدست آمده باشیم.حال برای بدست آورن تابع هدف اینگونه عمل میکنیم که مقدار این تابع را ابتدا قدر مطلق میگیرم و سپس در یک منفی ضرب میکنیم با این کار در‌واقع میزان اختلاف ما با جواب بدست میآید و چون هر چه اختلاف کمتر باشد ارزش جواب بیشتر است پس باید آن را در یک منفی ضرب کردن تا تابع هدف با تابع شایستگی یکسان گردد.

در این مساله برای تولید حالات اولیه از تابع رندم به صورتی استفاده شده است که اعدادی در بازه گفته شده تولید کند.همچنین با توجه به پیوسته بودن فضای مساله برای تابع next state که حالات بعدی را تولید میکند در جهت گرادیان تابع داده شده حرکت می‌شود.برای تشخیص حالت نهایی نیز یک دقت جواب درنظر گرفته شده که توسط ثابت PRECISION\_SOLVING\_EQUATION کنترل میشود( این مقدار به صورت پیشفرض برابر ۰.01 است ) .

الف )

نمودار بهترین و بدترین و میانگین شایستگی :

ب‌)‌ با توجه به نمودار زیر همانطور که میبیند اگر واریانس از یک حدی بیشتر شود دیگرجواب های جهش داده شده در بازه قرار نمیگیرد در نتیجه همگرایی نسل به سمت جواب‌های بهینه کند کتر میشود.



ج ) در این بخش تعداد کل ارزیابی ها را ۱۰ گذاشتیم

برای بخش بهترین های هر نسل با افزایش تعداد جمعیت بهترین ها نیز بهتر میشدند و شایستگی بیشتری داشتند مثلاً وقتی جمعیت را برابر ۱۰۰ در نظر گرفتیم هیچ‌گاه در پایان ارزیابی های شایستگی ( ۱۰ مرحله ) ارزش بهترین نسل از 0.05- بیشتر نبود . اما از طرفی با افزایش جمعیت بدترین های هر نسل نیز بدتر می‌شدند و با افزایش جمعیت میانگین شایستگی در نسل ها نیز اندکی رو به بدتر شدن میرفت که البته خیلی قابل توجه نبود.