به نام خدا

محمد سجاد علیپور گزارش کار پروژه ۱ ۹۸۳۱۰۴۳

مدل سازی مسئله :

از آنجایی که مکان مانع ها و مشتری ها ثابت است و همچنین هزینه های هر خانه ثابت است و تغییر نمی کند این نقشه را به صورت یک آرایه دو بعدی استاتیک تعریف می کنیم .

استیت ها یا همان رئوس گراف ما در این مسئله حاوی اطلاعات زیر می باشد:

۱- مختصات ربات

۲- یک آرایه از کره ها که هر کره دارای مختصات و اینکه آیا حق جابجایی دارد یا نه ، می باشد .

۳- استیت پدر (برای نشان دادن مسیر و همچنین expand نکردن دوباره ی پدر به درد میخورد).

استیت هدف ما در واقع استیتی است که همه ی مشتری ها دارای کره شده باشند . یعنی در مختصات تمام مشتری ها کره باشد .

تابع شهودی انتخاب شده :

تابع شهودی من برای الگوریتم A* به شکل زیر می باشد :

$$\sum_{i=1}^{n} min(manhattan\_distance(butter(\ i), purchasers))$$

بررسی قابل قبول بودن تابع هیوریستیک بالا :

از آنجا که فاصله ی منهتنی هر کره تا نزدیک ترین مشتری از فاصله ی واقعی آن تا نزدیک ترین مشتری کمتر مساوی است و همچنین از آنجا که باید همه ی کره ها به یک مشتری برسند می توانیم ادعا کنیم که اگر فاصله ی منهتن هر کره تا نزدیک ترین مشتری را جمع کنیم قطعا از فاصله ی واقعی کمتر خواهد بود . زیرا علاوه بر دلایل ذکر شده در بالا ما فاصله

ی ربات تا هر کره را نیز نادیده گرفته ایم پس قطعا کمتر از فاصله ی واقعی خواهد بود . به عبارت دیگر :

$$for\ all\ state\ k: \cdot \leq h(k) \leq h*(k)$$

بنابراین تابع هیوریستیک ما قابل قبول (admissible) است .

توضیح کلی توابع و کلاس های تعریف شده در کد :

هر ۳ الگوریتم دارای کلاس استیت می باشند و نقشه نیز به صورت استاتیک در کلاس استیت می باشد .

هر کدام از ۳ الگوریتم دارای فایل جاوای خود می باشند و در آن تمام کلاس ها و تابع هایشان نوشته شده است و کافیست هر فایل را جداگانه اجرا کرد .

در پوشه ی ***output*** نیز خروجی الگوریتم برای ورودی ها نامگذاری و مشخص گردیده است .

خروجی همچنین در کنسول نیز نشان داده می شود .

مقایسه ی روش های پیاده سازی شده در موارد زیر :

۱- زمان صرف شده :

برای ورودی ***test1.txt*** :

الگوریتم ***IDS*** :

۹۸ میلی ثانیه

الگوریتم ***Bidirectional BFS*** :

۳۳ میلی ثانیه

الگوریتم ***A**** :

۲۷ میلی ثانیه

۲- پیچیدگی زمانی :

الگوریتم IDS :

O(b^d)

که *b* همان *branching factor* و *d* همان ماکسیمم عمق می باشد .

الگوریتم Bidirectional BFS :

O(b^(d/۲))

الگوریتم *A :

کمترین : *O(bd)*

بیشترین : *O(b^d)*

(بیشتر به تابع هیوریستیک بستگی دارد)

۳- تعداد گره های تولید شده :

برای ورودی *test1.txt* :

الگوریتم *IDS* : ۷۸۶۱

الگوریتم *Bidirectional BFS* : ۵۵

الگوریتم *A* : ۱۷۷

۴- تعداد گره های گسترش داده شده :

برای ورودی *test1.txt* :

الگوریتم *IDS* : ۳۶۲۲

الگوریتم *Bidirectional BFS* : ۲۴

الگوریتم *A** : ۱۰۹

۵- عمق راه حل :

برای ورودی *test1.txt* :

الگوریتم *IDS* : ۱۰

الگوریتم *Bidirectional BFS* : ۱۰

الگوریتم *A** : ۱۰