باسمه تعالی

Class examination 3

نیمسال دوم ۱۴۰۱-۱۴۰۰

مدت امتحان :۶۰ دقیقه

۱- پس از شرح جستجوهای حریصانه و A* هر دو را با هم مقایسه کنید و بگویید کدامیک عملکرد بهتری دارد؟

۲- شرایط تابع هیوریستیک قابل پذیرش را با ذکر مثال بیان کنید؟

۳- کامل بودن و بهینه بودن A* را همراه با مرتبه زمانی و مرتبه حافظه آن بررسی کنید؟

۴- انواع جستجوی A* با حافظه محدود را نام برده و شرح دهید؟

۵- منظور از تغییر عقیده در جستجوی RBFS را شرح دهید؟

۶- تابع هیوریستیک را همرا با کیفیت تابع هیوریستیک ، با ذکر یک مثال شرح دهید؟

۷-ابداع تابع هیوریستیک قابل پذیرش را از طریق نسخه ساده شده از مساله relax version شرح دهید؟

۸- انواع جستجوی محلی را نام برده و دو مورد را به دلخواه با ذکر مثال شرح دهید؟

جستجوی حریصانه	جستجوی A^*	
1) - کامل نیست (حلقه تکرار)	- کامل است	پس A^* به دلیل بهینه بودن و کامل بودن عملکرد بهتری دارد.
- بهینه نیست (کامل نیست)	- بهینه است.	
- مرتبه زمانی $O(b^m)$	- مرتبه زمانی از مرتبه نمایی است	
- مرتبه حافظه $O(b^m)$	- حافظه به مراتب مشکل بزرگتری است	

2) تابع هیوریستیک

1) قابل پذیرش است (خوش بینانه است.) هیچ گاه تخمین اضافه از هزینه تا هدف ارائه ندهد.

1) $h(n) <= h^*(n)$ where $h^*(n)$ is the true cost from n

2) $h(n) >= 0$ so $h(G)=0$ for any goal G

3) A^* کامل و بهینه است و مرتبه زمانی از مرتبه نمایی است اما حافظه به مراتب مشکل دارد. A^* با استفاده از درخت جستجو بهینه است و اگر مسئله دارای پاسخ باشد A^* همواره آن را در کوتاه ترین مسیر خواهد یافت. A^* در حالی که بهترین جواب را پیدا می کند مشکل حافظه دارد و حافظه زیادی لازم دارد؛ زمان محاسبه قطعاً ضعف اصلی A^* از آنجا که A^* ناگزیر است تمامی گره های تولید شده را در حافظه نگاه دارد، معمولاً قبل از اتمام زمان، حافظه تمام خواهد کرد. پس A^* برای بسیاری از مسائل بزرگ عملی نیست.

4) عبور مشکل حافظه A^* با حفظ ویژگی کامل و بهینه بودن

1) IDA^*: تعیین یک مرتبه محدود f-cost (g+h) به جای عمق محدود

2) RBFS: الگوریتم بازگشتی برای داشتن حافظه خطی

3) MA^*: بسط بهترین گره جدید و حذف بدترین گره قدیمی در زمان پر شدن حافظه

(5) RBFS جستجوی بهتر و موثرتری از IDA^* است. تغییر عقیده (هنوز گسترش اضافی گره‌ها وجود دارد). تعیین پیچیدگی زمانی مشکل است. IDA^* و RBFS هر دو از میزان بسیار کم حافظه رنج می‌برند (با اینکه حافظه زیادی وجود دارد اما نمی‌توانند از آن استفاده کنند)

(6) تابع هیوریستیک خوب می‌تواند فضای حالت را به طرز چشمگیری کاهش دهد.

- عمق جواب بهینه به طور متوسط ← ۲۲ - فاکتور انشعاب به طور متوسط ← ۳
- تعداد فضای حالت ۳.۱×۱۰^{۱۰}

7	2	4
5		6
8	3	1

	1	2
3	4	5
6	7	8

h_1 ← تعداد کاشی‌هایی که سر جای خود نمی‌باشند $h_1(s)=8$

کیفیت تابع هیوریستیک ←

h_2 ← مجموع فاصله افقی - عمودی هر کاشی تا جای واقعی $h_2(s)=۳+۱+۲+۲+۲+۳+۳+۲=۱۸$

h_3 ← مجموع فاصله افقی - عمودی - قطری هر کاشی تا جای واقعی $h_3(s)=۲+۱+۱+۱+۲+۲+۲+۲=۱۳$

فاکتور انشعاب موثر (b^*): فاکتور انشعاب درخت یکنواخت

$$N+1 = 1 + b^* + (b^*)^2 + \dots + (b^*)^d$$

بهترین مقدار برای b^*؟

(کیفیت تابع ← b^* فاکتور انشعاب موثر فاکتور انشعاب درخت یکنواخت)

(7) از طریق نسخه ساده‌شده از مسئله (relax version)

h_1 ← هر کاشی می‌تواند به هر جایی منتقل شود. h_2 ← هر کاشی می‌تواند به هر خانه همسایه منتقل شود.

ABSolver هزینه راه حل برای مکعب روبیک را تخمین می‌زند.

(8)

- الگوریتم تپه‌نوردی

Simulated annealing (SA) ← اجتناب از گیر کردن در بیشینه‌های محلی با اجازه دادن به انجام حرکات نوعی (نامناسب) که در حین گذشت زمان احتمال و تعداد آن کم می‌شود.

- بیشینه الگوریتم به مفهوم متالورژی برمی‌گردد ← مثال Bouncing ball analogy

- ژنتیک Genetic Algorithms

- پرتوی محلی Local beam search ← از k حالت شروع به جای یک حالت بهره می‌برد. - حالت شروع: k حالت تصادفی
- حالت بعدی: انتخاب k تا بهترین حالت‌ها از بین تمام برگ‌ها
- حالت خاتمه ← پیدا شدن هدف یا بررسی تمام حالات
- ممکن است الگوریتم از عدم تنوع کافی برخوردار باشد. - تفاوت با تپه‌نوردی ← اطلاعات به اشتراک گذاشته می‌شود.