hw 2

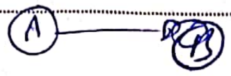
Subject

Year: Month: Date:

Sa | Su | Mo | Tu | We | Th

①

الف) برای بررسی Consistency باید اصل زیر را بررسی کنیم:



$h(A) - h(B) \leq c(A,B)$

تمام node هایی که با یک یال بهم متصل هستند را می‌نویسیم و بررسی می‌کنیم:

یال‌ها: AB _ AC _ BE _ BD _ CD _ DE _ DG _ DF _ EG _ FG

~~h1 : (۹.۵ - ۹ ≤ ۲)~~ ~~h2:~~   به طور مثال: h_1 = AB = ۹.۵ - ۹ = ۰.۵

(AB : h_1 = ۰.۵ ≤ ۲ , h_2 : ۱۰ - ۱۲ ≤ ۲)(AC : ۹.۵ - ۸ ≤ ۵ , ۱۰ - ۱۰ ≤ ۵)

(BC : ۹ - ۸ ≤ ۱ , ۱۲ - ۱۰ > ۱ ✗ => نیست Consistent h_2 در نتیجه

در ادامه فقط h_1 را بررسی می‌کنیم

(BD : ۹ - ۷ ≤ ۷)(CD : ۸ - ۷ ≤ ۱)(DE : ۷ - ۱.۵ > ۴ ✗

در نتیجه h_1 هم خاصیت Consistency را ندارد

Scanned with CamScanner

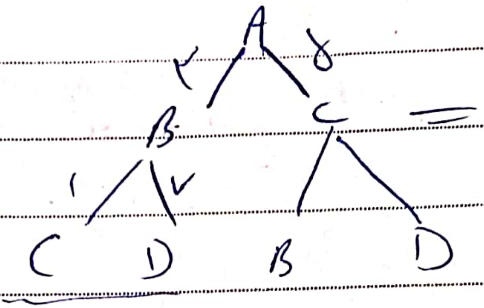
Subject: hw2

Year: Month: Date:

Sa Su Mo Tu We Th

① قسمت ب

ابتدا برای h1 می‌نویسیم

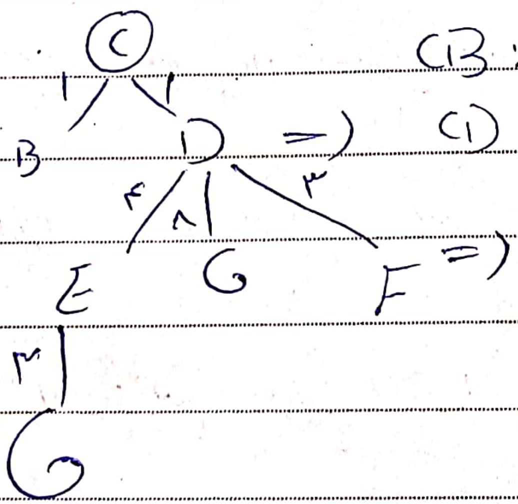


AB: ۲+۹=۱۱ ← AB را بررسی نمی‌کنیم اما به علت

⇒ AC: ۵+۸=۱۳ احتمال اشتباه C را هم expand می‌کنیم

ADC: ۲+۱+۸=۱۱ ✓

ABD: ۲+۷+۷=۱۴



⇒ CB: (۲+۱+۱)+۹=۱۳

CD: (۲+۱+۱)+۷=۱۱ ✓

⇒ DE: (۲+۱+۱+۴)+۱٫۵=۹٫۵ ✓

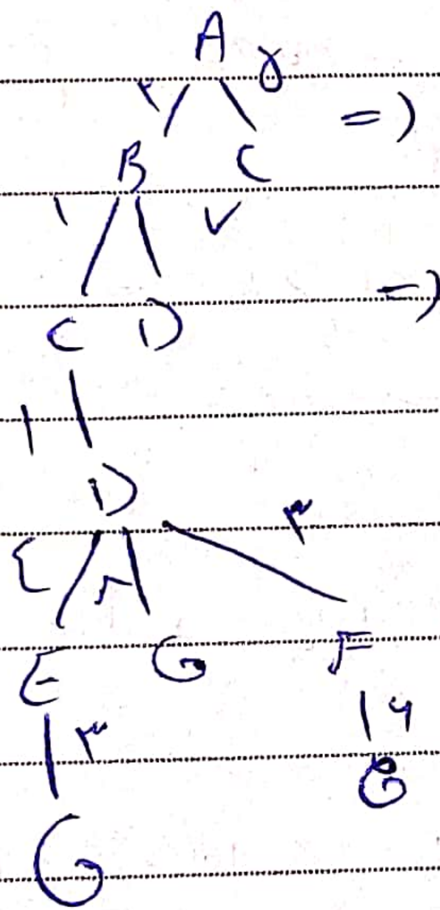
DG: (۲+۱+۱+۸)+۰=۱۲

DF: (۲+۱+۱+۳)+۴=۱۱

DEG: (۲+۱+۱+۴+۳)+۰=۱۱ ✓

مسیر ← ABCDEG

حال برای h۲ می‌نویسیم:



AB: ۲+۱۲=۱۴ ✓

AC: ۵+۱۰=۱۵

BC: (۲+۱)+۱۰=۱۳ ✓

BD: (۲+۷)+۸=۱۷

DE: (۲+۱+۱)+۴+۱=۹ ✓

DG: (۲+۱+۱)+۸+۰=۱۲

DF: (۲+۱+۱)+۳+۴,۵=۱۱,۵

EG: (۲+۱+۱+۴)+۳+۰=۱۱

ABDEG

بازهم مسیر

Subject: hw2
Year: Month: Date:

Sa Su Mo Tu We Th

(۲)

الف) خیر stochastic نیست. چون قدم بعدی دارای احتمالات می‌باشد و مطلق نیست

ب) درست. DFS از لحاظ memory نسبت به BFS بهینه‌تر می‌باشد. زیرا BFS

هر سطر از سطح را کامل چک می‌کند و اگر جواب ما در آخرین عمق باشد الگوریتم BFS باعث

هزینه زیاد فضایی (space) می‌شود

ج و د) طبق جزوه Consistency implies admissibility

=> اگر Consistent باشد قطعا admissible نیز هست
هیوریستیک

اما در بعضی موارد پیش می‌آید که عکس این موضوع صادق نباشد یعنی اگر

admissible باشد ممکن است (به ندرت) Consistent نباشد

(۳) بله می‌توان. یکی از روش‌های Informed search، A* search می‌باشد که از این

الگوریتم برای حل این نوع puzzle نیز استفاده می‌شود. به طور مثال A* tiles،

manhattan A* نمونه‌ای از این الگوریتم‌ها هستند. در الگوریتم manhattan

hw 2

Subject:

Year: Month: Date:

Sa | Su | Mo | Tu | We | Th

فاصله ای که هر عدد از جایگاه واقعی خود دارد را براساس x و y نوشته و اختلاف

آنها را حساب می کنیم

Time complexity: $O(n^2)$

به طور مثال در این سوال الگوریتم manhattan را پیاده کردیم.

بهتر : $O(n \log n)$

در ابتدا مختصات ابتدایی و انتهایی را بدست می آوریم

(مختصاتی که در آن مار به غذا می رسد)

متغیر Price را داریم که در آن هزینه f قرار داده می شود یعنی f حاصل از ابتدای حرکت تا به اینجا

$f = g + h$

least cost node هم مربوط به کمترین cost تا به این لحظه می باشد. اگر این node با final location برابر بود مار غذا را می گیرد. اگر خیر ادامه

سپس در neighbors همسایه least cost node را بدست می آوریم و سپس مختصات neighbors را چک می کنیم. متغیری داریم به نام furthercost که همان g در فرمول بالا است یعنی هزینه را تا به اینجا بررسی می کند. و booleanی به نام bestpath داریم که چک می کند آیا بهترین مسیر هست یا خیر. سپس در تابع Bestpathcheck می رویم

در این تابع از فاصله manhattan برای بدست آوردن neighbors استفاده می شود

و در نهایت bestpath را True می کند. اگر bestpath True باشد مقدار f را آپدیت می کند که از فرمول

$f = g(n) + h(n)$

(h: هیوریستیک، g: هزینه تا اینجا)

بدست می آید.

Scanned with CamScanner