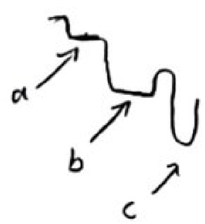
علی آقایاری ٤٠٠١٠٢٧١٨

۱) الف) غلط – عاملی که نمی‌تواند محیط خود را درک کند، لزوماً نمی‌تواند نتیجه و یا رفتاری عقلانی از خود نشان دهد.

ب) درست – $a \to p^*$: که را یک عامل می‌گیریم اگر این عامل دانش پیشین (p^*) بهتری داشته باشد قطعاً رفتار بهتری از خود نشان می‌دهد.

ج) درست – حرکت sideways همیشه ما را به مینیمم سراسری نمی‌رساند، برای مثال نمودار روبه‌رو را ملاحظه کنید: مینیمم سراسری در قسمت c است، اگر مینیمم نسبی



a را پیدا کرده باشیم با sideways به b برسیم، اما نمی‌توان از b به c رسید برای همین لازم است از حالت‌های اولیه مختلف شروع کنیم تا در مکانی قفل نشویم. (البته اگر جواب ۱۰۰٪ نخواهیم، با احتمال ۹۴٪ به جواب می‌رسیم با sideways) برای همین گزاره غلط است

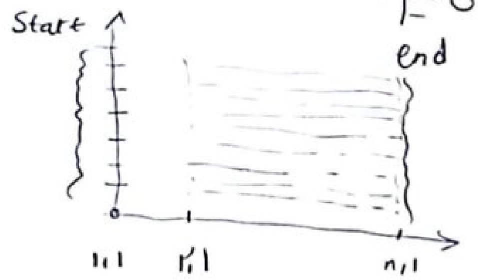
د) الگوریتم UCS عملاً همان الگوریتم دایجسترا است که در این الگوریتم برای پیدا کردن بهترین مسیر نیاز داریم که یال منفی نداشته باشیم، نداشتن یال منفی معادل است با کاهش نیافتن تابع هزینه است! به همین دلیل، با شروط داده شده، الگوریتم ما بهینه و کامل خواهد بود «با فرض b محدود و داشتن کران پایین ε»

ه) درست. این الگوریتم همواره فقط یک state و مقداری value سیو می‌کند برای همین مموری اشغال شده از "O" عدد ثابت است ولی این الگوریتم در دام مینیمم نسبی نمی‌افتد زیرا اول jump‌ها را با احتمال بیشتری انجام می‌دهد و کم‌کم احتمال جهش‌ها را کمتر می‌کند در الگوریتم ژنتیک، همین کار با جهش اتفاق می‌افتد. «O مموری ثابت بود چون مسیر state‌ها را نگه نداریم»

علی آمایاری ٤٠٠١٠٤٧١٥

۲)

State ها را اینگونه تعریف می‌کنیم که در هر State n تا location داریم که هر کدام به ترتیب location ربات i ام را نشان می‌دهد، location ها هم به شکل (x_i, y_i) نشان می‌دهیم.



Second bot

init_state : $((1,1), \overbrace{(1,2)}, \text{——}, (1,n))$

finish_state : $((n,n), (n,n-1), \text{——}, (n,1))$

action ها را اینگونه تعریف می‌کنیم که در هر action group، هر کدام از ربات‌ها، یکی از عمل‌های بالا/پایین/چپ/راست/هیچ را انتخاب می‌کند.

third bot action

action (group action) : $(U, D, L, R, \emptyset, \text{——})$

همسایه‌های هر State اینگونه خواهند بود که هر ربات می‌تواند در لحظه بعد یکی از موقعیت‌های زیر را داشته باشد،

$(a_i, b_i) \xrightarrow{action_i} (a_i \pm 1, b_i)$ or $(a_i, b_i \pm 1)$ or (a_i, b_i)

حال ← n ربات داریم که هر کدام ۵ نوع اکشن دارند، پس از هر State می‌توانیم به 5^n State مجاور برویم چون 5^n نوع action group خواهیم داشت، البته بهتر است حالتی که همه ربات‌ها هیچ کاری نکنند را از action group های ولید حذف کنیم. ← $5^n - 1$ نوع اکشن

هزینه هر کنش را اینگونه می‌گذاریم که هر action group، cost = 1 دارد.

ب) مسئله stateهای invalid ای خواهد داشت برای همین لازم است با استفاده از روش بهتری (اندازه مسئله) O را حساب کنیم. می‌خواهیم که n×n تا موقعیت ولید داریم که در n تا از این خانه‌ها، جایگشتی از ربات‌ها قرار خواهد گرفت ← $\binom{n^2}{n}$ برای انتخاب خانه‌ها و n! برای جایگشت

$$\boxed{\binom{n^2}{n}\times n!}$$

$$O\left(\binom{n^2}{n} n!\right) = O\left((n^2)^n\right) = O\left(n^{2n}\right)$$

ج) این عدد در قسمت الف پیدا کرده branching-fac نمی‌تواند بیشتر از 5^n شود پس این عدد می‌تواند کران بالای مناسبی باشد. از برای بهتر کردن جواب، حالت الکی ای که هیچ رباتی کاری نمی‌کند را حساب نمی‌کنیم:

$$\text{branching-fac} \le 5^n - 1$$

د) از جواب بدیهی ای که به ذهنمان می‌رسد استفاده می‌کنیم و قابل قبول بودن آن را تست می‌کنیم.
h یک موقعیت (x_i, y_i) برابر است با فاصله منهتن آن با goal
از آنجایی که الگوریتم ما حرکت‌های ۴ جهته + توقف است پس به نظر فاصله منهتن h خوبی است.

$$h = \underbrace{|n - x_i|}_{\text{فاصله افقی}} + \underbrace{|n-i-1-y_i|}_{\text{فاصله عمودی}}$$

(با توجه به جدول مثال نقش الف)

این یک h، admissible است چون در هر حالت $h \le h^*$ است.
و اگر شروط مسئله زیاد شوند، تنها باعث افزایش h^* می‌شوند $\xrightarrow{\text{پس همچنان}}$ $\boxed{h \le h^*}$

هـ) h برابر hهای admissible ای هستند ← $h = \min(h_1, \dots, h_n)$

برای اثبات $h \le h^*$ خواهیم داشت

$h = \max(h_1, \dots, h_n)$

$h = \sum_{i=1}^{n} h_i / n$

$\min(h_1, \dots, h_n) \le h^*$ ← بدیهی است چون همیشه $h_i \le h^*$

$\max(\dots) \le h^*$ ← این هم بدیهی است چون end goal ما رسیدن همه ربات‌ها به مکان مورد نظر است پس h^* از همه hها بزرگتر است

$\sum_{i=1}^{n} h_i / n \le h^*$ ← میانگین hها است و بین min و max وجود دارد پس از h^* کوچکتر است.

برای توضیح بیشتر admissible بودن ۳ h قبل، باید توجه کرد که هر h_i حداقل مقدار مورد نیاز برای رسیدن به هدف است ((حالت نسبتا مقید)) و ما h^* را تعداد action group هایی گرفته‌ایم. یعنی اگر همه h_i ها ۰ باشند و فقط یک h_i برابر m باشد، ما حداقل m action group نیاز داریم پس h^* حداقل m است.

min() که بدیهی است، max() هم با استدلال بالا بدیهی می‌شود، $\sum h_i / n$ هم میانگین آن‌ها است پس درست است! ((همه اینها بدون احتساب شروط مسئله بوده، اگر شروط حساب شوند h^* بیشتر می‌شود و اثبات ما قوی‌تر می‌شود))

Start

a	-
b	-

$\longrightarrow$

end

-	b
-	a

برای نقض h ها مثال نقض می‌آوریم:

first step:

b	a
-	-

second step:

-	b
-	a

پس h^* ما برابر است با ۲

حال محاسبه h ها:

$h = n \times \min(h_1, h_2) = ۴$ ✗

$h = n \times \max(h_1, h_2) = ۴$ ✗

$h = \sum_{i=1}^{۲} h_i = ۴$ ✗

دلیل اشتباه بودن این h ها واضح است.

در هر مرحله (در صورتی که شرط‌های مسئله اهمیت نکنند)
از هر h_i یک عدد کم می‌شود برای همین نمی‌توان انتظار
چنین چیزی ((با هر گونه فعل مشابه)) از h_i ها را داشت!

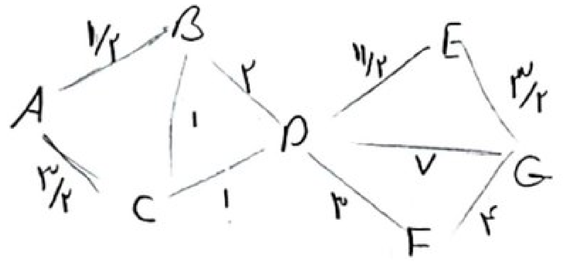
علی آقایاری ۴۰۰۱۰۲۷۱۸

۳) الف)

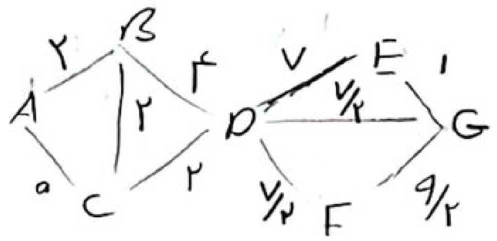
monotonic : $h(n) \le h'(n) + C(n, a, n')$

$\xrightarrow{\text{برای هر یال}}$ $|h_b - h_a| \le w_{ab}$

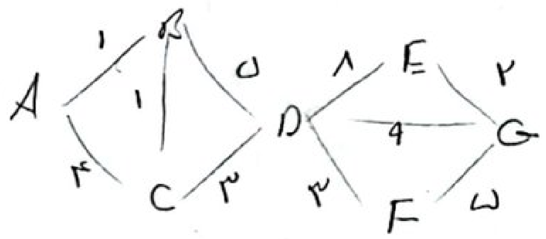
$\Delta h1$:



$\Delta h2$:



w :



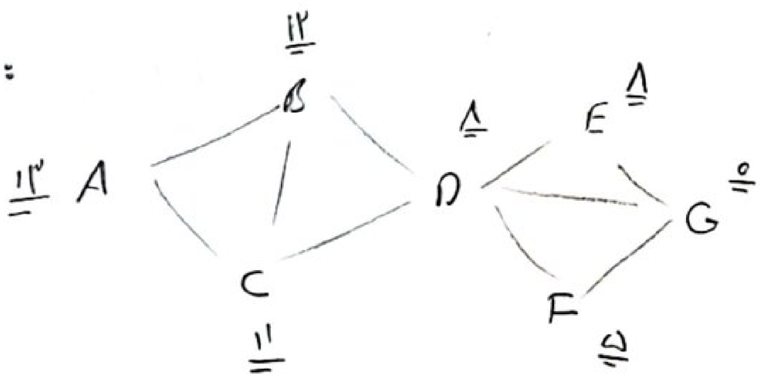
برای h_1، همه یال‌های h_1 از یال‌های متناظر w کمتر هستند ⟵ h_1 consistent

برای h_2 ⟵ یال‌های AB، BC و DF مثال نقض هستند ⟵ h_2 <u>not</u> consistent

---

برای قابل قبول بودن: اگر همواره شرط $h \le h^*$ برقرار باشد، h قابل قبول خواهیم داشت.

dist :



$h1$ {
۹/۵ ≤ ۱۳
۹ ≤ ۱۲
۸ ≤ ۱۱
۷ ≤ ۸
۱/۵ ≤ ۸
۴ ≤ ۵

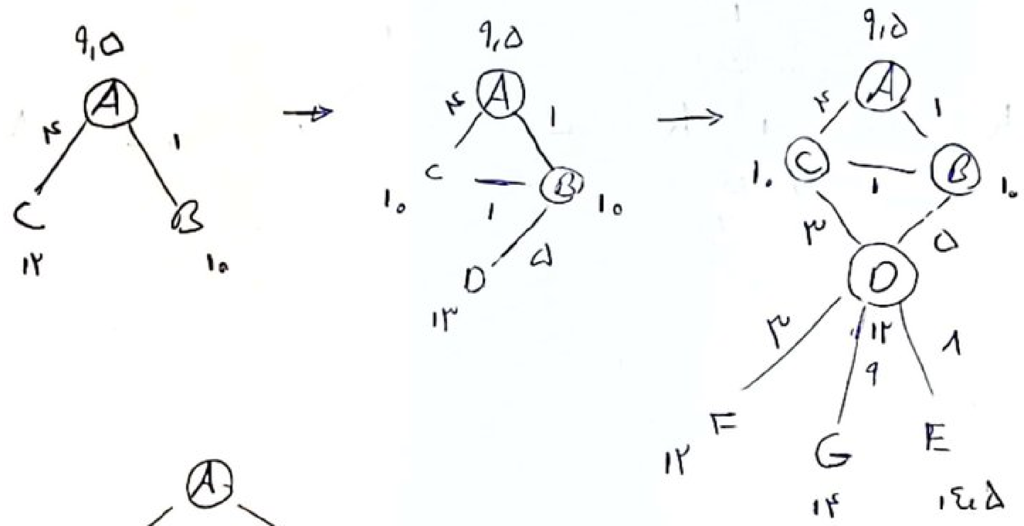
h_2 {
۱۰ ≤ ۱۳
۱۲ ≤ ۱۲
۱۰ ≤ ۱۱
۸ ≤ ۸
۱ ≤ ۸
۲/۵ ≤ ۵

$\boxed{h_1 \ \& \ h_2 \ \text{admissible} \ \checkmark}$ ⟵

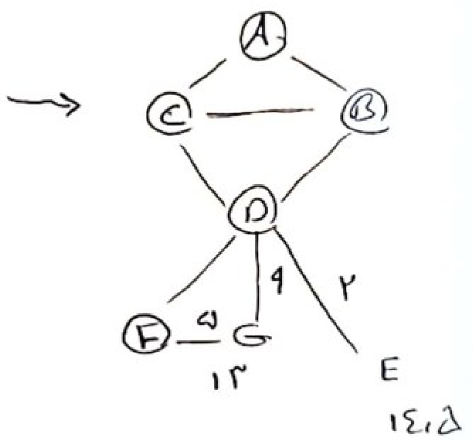
ب)

	ABDG	ACDG	ABCDFG
DFS	✓	✓	✓
BFS	✓	✓	✗
دایکسترا	✗	✗	✓
$A^*: h_1$	✗	✗	✓
$A^*: h_2$	✗	✗	✓

توضیح $A^*(h_1)$: «همیشه کمترین $f=g+h$ باز می‌شود!»



اول C انتخاب می‌شود
بعد تنها State
بعدی D است بنابراین
همین D انتخاب می‌شود
- یک مرحله جابجا شده -

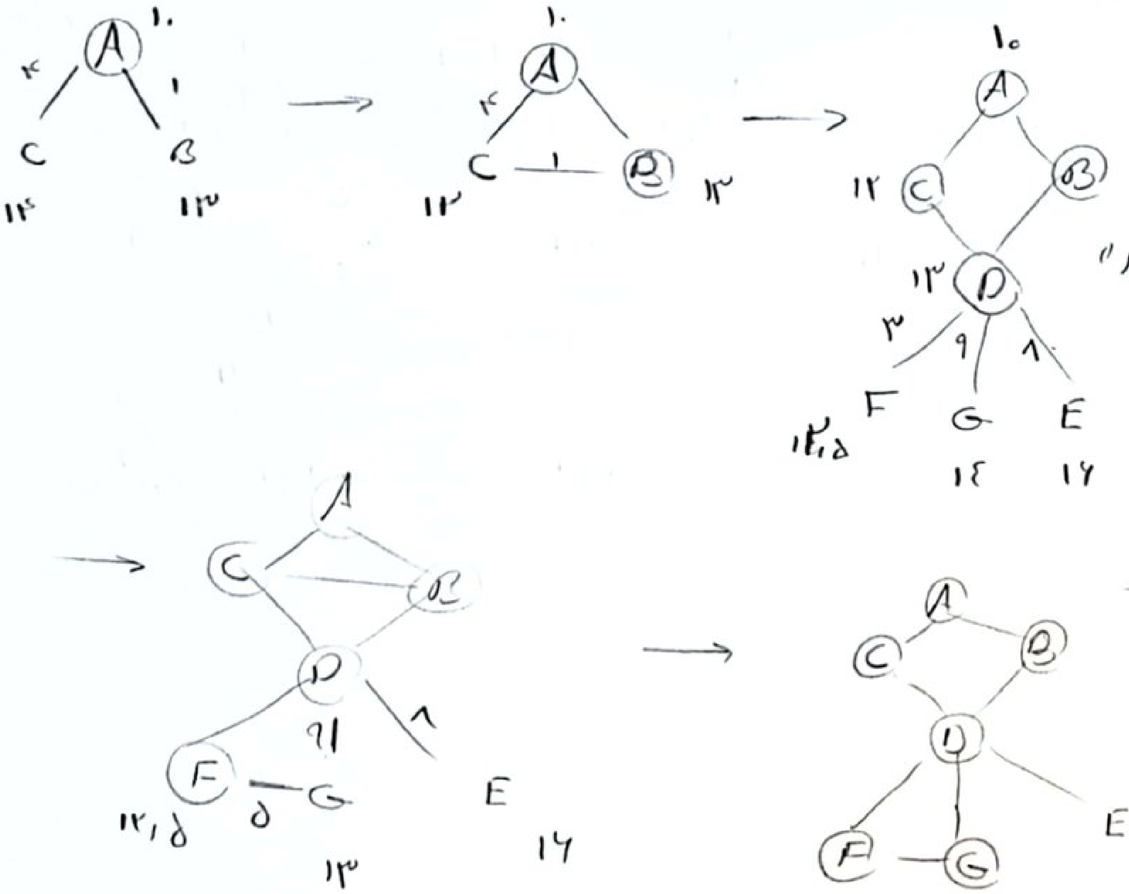


G open می‌شه و چون مقصد open شده، مسیر opt را
پیدا کرده‌ایم ⟵ مسیر را با backtrack به دست می‌آوریم.

P = ABCDFG ⟵

لازم به ذکر است که در اینجا جستجوی گرافی انجام دادیم «در اینکه تکراری چک نکردیم» زیرا monotonic (h_1)

توضیح A^* (h_2)



C را open می کنیم و فقط D ی ما [illegible] در انتخاب نداریم!! پس در اینجا یک مرحله جلو رفتیم!

به برای بدیهی بودن جواب!!

مقصد open شد و به جواب رسیدیم

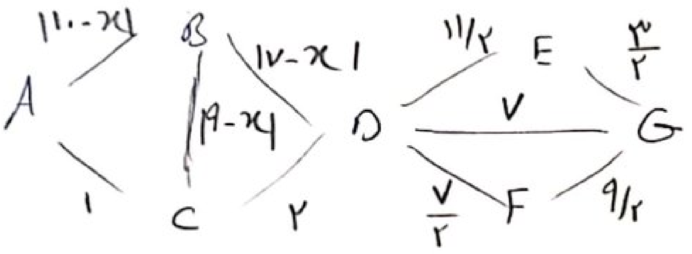
توجه شود که با این ها نباید جستجوی گرافی زد بلکه باید جستجوی درختی زد!

ج) با توجه به گراف dist در بخش الف، سوال را حل می کنیم.

$$\longrightarrow h_3 \begin{cases} 10 \le 13 \\ x \le 12 \quad \checkmark \\ 9 \le 9 \\ 7 \le 8 \\ 1.5 \le 8 \\ 4.5 \le 5 \end{cases} \longrightarrow h_3(B) \le 12$$

$0 < h_3(B) \le 12$ و $h_3(B) \neq 0$

د) گراف Δh_3 را می سازیم و با توجه به گراف w در بخش اول سوال را حل می کنیم.



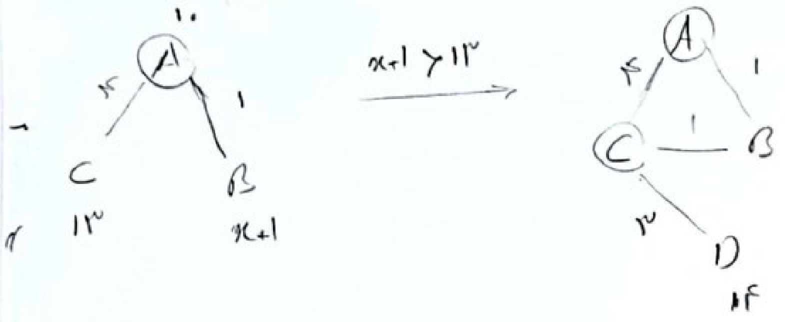
همه یال ها از یال های متناظر w کمتر هستند اگر:

$\underbrace{|7-x| \le 5}_{2 \le x \le 12}$ و $\underbrace{|9-x| \le 1}_{8 \le x \le 10}$ و $\underbrace{|10-x| \le 1}_{9 \le x \le 11}$

$$\xrightarrow{\text{اشتراک}} 9 \le x \le 10$$

CamScanner

ج)



در این مرحله، صورت سوال از ما می‌خواهد که از C به B برویم یعنی $parent^B$ برابر با C باشد. اما اینکار ممکن نیست چون

«$h + g = f$»

h_B ثابت است اما f «(B-A)» برابر یک است و f «B-C-A» برابر ۵ است.

پس ممکن نیست که $h_B + 5 < h_B + 1$ $\therefore$

و) چون مسئله یک را حل کرده، صرفا تغییرات را می گوییم

الف) در اینجا state ها مانند قبل اند - action group نخواهیم داشت بلکه فقط یک ربات حرکت می کند

← یعنی هر action ≡ انتخاب یک ربات و حرکت آن « حالت اعلام ندادن هیچ کار را حذف می کنیم

unvalid state ← همان است با این تغییر که نباید bot n به مقصد برسد در حالی که bot m به مقصد نرسیده است با فرض $m<n$.

start/goal state ← مثل قبل ! $\xrightarrow{\text{هزینه}}$ مثل قبل

ب) state ها همان اوردر را خواهند داشت چون عملا همان state ها هستند به جز unvalid state هایی که در بخش و-الف توضیح داده شد ← $O(n^{2n})$

ج) در اینجا branching fac عوض می شود چون n action عوض شده است.

برای اینکه max تعداد همسایه ها را حساب کنیم شرط را اعمال نمی کنیم، n تا ربات داریم که یکی از آنها انتخاب می شود و به ۴ جهت می رود ← $\text{branching-fac} = 4 \times \binom{n}{1}$

$= 4n$

علی آقایی ٤٠٠١٠٤٧١٥

۴) الف) فضای حالت مسئله پایان. تعداد stateها برابر است با جایگشت‌های مختلف جمله n کلمه‌ای یعنی n!

ب) هر action را اینگونه تعریف می‌کنیم که ۲ کلمه را انتخاب می‌کنیم و آن ۲ کلمه را جابجا می‌کنیم. پس هر state

$\binom{n}{2}$ state مجاور خواهد داشت. مثال:

هر چه زودتر خواهش می‌کنم آزاد کنید من را

مثال ۱ ← زودتر هرچه خواهش می‌کنم آزاد کنید من را

مثال ۲ ← هرچه زودتر آزاد می‌کنم خواهش کنید من را

ج) این الگوریتم یک الگوریتم greedy است و اگر به min نسبی برسیم ممکن است در آنجا قفل شویم یعنی همسایه بهتری نداشته باشیم تا به آن برویم و نتوانیم min اصلی را پیدا کنیم. برای همین، این الگوریتم به خودی خود complete نیست اما اگر روش های احتمالی را به این روش اضافه کنیم می‌توانیم در مدت زمانی محدود جمله‌ای معنادار تولید کنیم. الگوریتم های احتمالی به ما کمک می‌کنند تا قفل نشویم.

د) $\underbrace{\text{init population}}_{۱} \rightarrow \underbrace{\text{fitness function}}_{۲} \rightarrow \underbrace{\text{selection}}_{۳} \rightarrow \underbrace{\text{cross}}_{۴} \rightarrow \underbrace{\text{mutation}}_{۵}$

① ← در این مرحله، k جمله جنریت می‌کنیم "random generate"

② ← حال با استفاده از تابع تعریف شده در صورت سوال fitness-valueهای k جمله را به دست می‌آوریم.

"این فیتنس ولیوها را با f_i نشان می‌دهیم یعنی مقدار fitness جمله i ام"

③ ← قرار است k جمله اصلی انتخاب کنیم، به هر یک از k جمله جنریت شده احتمال $\frac{f_i}{\sum_{i=1}^{k} f_i}$ می‌دهیم، حال به‌طور رندوم با توجه به احتمال انتخاب‌ها، در هر مرحله یک جمله انتخاب می‌کنیم.

"در k مرحله - واضحاً می‌توانیم جمله‌های تکراری داشته باشیم."

④ ← حال، k جمله را به‌طور رندوم به k/2 گروه دوتایی تقسیم می‌کنیم. در هر گروه دوتایی عملیات (OX1) را انجام می‌دهیم. دلیل انتخاب OX1 به این دلیل بود که جملات ما حالت جایگشتی داشتند و هر کلمه چندبار

نمی‌آید ← توضیح $OX1$: ۲ عدد sample داریم که از sample 1 تعدادی خانه انتخاب می‌کنیم (در نمودار)

حال همه آنها را نگه می‌داریم و باقی را با ? پر می‌کنیم. حال به دنبال sample 2 می‌رویم، از sample 2

حرف‌هایی که در sample 1 انتخاب نشده‌اند را نگه می‌داریم. در آخر به جای ? ها به ترتیب order

دیده شده در sample 2 کلمه‌های مناسب را می‌گذاریم!.

در اینگونه جواب (Cross(sample 1 را پیدا می‌کنیم و نیاز است با همین روش (Cross(sample2 را هم حساب کنیم!)

5 ← از Swap استفاده می‌کنیم و با احتمال کم به طور رندوم بین دو کلمه یک جمله Swap انجام می‌دهیم.

CS CamScanner

علی اتابکی ۸۱۰۵۰۱۰۴۰

$$f_1: \mathbb{R}^n \to \mathbb{R},\ f_1(x) = \|x\|_2^2 = \sum_{i=1}^{n} x_i^2$$ (۵

$$f_2: \mathbb{R}^n \to \mathbb{R},\ f_2(x) = \|Ax\|_2^2 = (Ax \cdot Ax) = (Ax)^t Ax = x^t A^t A x$$

$$f_3: \mathbb{R}^n \to \mathbb{R},\ f_3(x) = \|Ax-b\|_2^2 + \gamma \|x\|_2^2$$

$$= (Ax-b)\cdot(Ax-b) + \gamma \|x\|_2^2$$

$$= (Ax-b)^t (Ax-b) + \gamma \|x\|_2^2$$

$$= (x^t A^t - b^t)(Ax-b) + \gamma \|x\|_2^2$$

$$= x^t A^t A x - 2x^t A^t b + b^t b + \gamma \|x\|_2^2$$

و $$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ | \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

الف)

$$\nabla f_1 = \begin{bmatrix} 2x_1 \\ | \\ 2x_n \end{bmatrix} = \boxed{2x}$$

$$\nabla f_2 = \frac{\partial f_2}{\partial x} = \frac{\partial x^t A^t A x}{\partial x} = \boxed{2A^t A x}$$

$$\nabla f_3 = \frac{\partial f_3}{\partial x} = \frac{\partial (x^t A^t A x - 2x^t A^t b + b^t b + \gamma \|x\|_2^2)}{\partial x}$$

$$= 2A^t A x - 2A^t b + 0 + 2\gamma x$$

$$= 2A^t(Ax-b) + 2\gamma x$$

CamScanner

ب) الگوریتم کاهش گرادیان اینگونه است که در هر مرحله $\vec{x}$ را در خلاف جهت گرادیان تغییر داده
تا به مینیمم برسد.

۱) $f: \quad x \leftarrow x - 2\alpha x$

۲) $f: \quad x \leftarrow x - 2\alpha A^t A x$

۳) $f: \quad x \leftarrow x - \alpha\left(2A^T(Ax-b) + 2\gamma x\right)$

ج) طبق اسلایدها داریم .

در هنگام اثبات، تابع در حال اثبات را f فرض کرده ایم. و وقتی از تابع دیگری در اثبات استفاده می کنیم آن را g فرض کرده ایم.

$f: \mathbb{R}^n \to \mathbb{R}$ is convex

if : $f(\alpha x + (1-\alpha) y) \leq \alpha f(x) + (1-\alpha) f(y)$

$\mapsto 0 \leq \alpha \leq 1$ and $x, y \in \mathbb{R}^n$

$$f(x) = \|x\|_2^2 = x^t x$$

$$f(\alpha x + (1-\alpha) y) = (\alpha x^t + (1-\alpha) y^t)(\alpha x + (1-\alpha) y)$$

$$= \alpha^2 x^t x + \alpha(1-\alpha) x^t y + (1-\alpha)\alpha y^t x + (1-\alpha)^2 y^t y$$

$$= \alpha^2 f(x) + (1-\alpha)^2 f(y) + 2\alpha(1-\alpha)(x \cdot y)$$

$$\xrightarrow{x \cdot y \leq \|x\|\|y\| \text{ داریم}} \leq \alpha^2 f(x) + (1-\alpha)^2 f(y) + 2\alpha(1-\alpha)(f(x))^{1/2}(f(y))^{1/2}$$

$$= (\alpha (f(x))^{1/2} + (1-\alpha)(f(y))^{1/2})^2$$

$g(x) = x^2$

g is convex ($\star$)

so $g(\alpha\sqrt{f_x} + (1-\alpha)\sqrt{f_y}) \leq \alpha g(\sqrt{f_x}) + (1-\alpha) g(\sqrt{f_y})$

$$\longrightarrow \boxed{\leq \alpha f(x) + (1-\alpha) f(y)}$$ اثبات شد!

$\star$: $f(\alpha x + (1-\alpha) y) \leq \alpha f(x) + (1-\alpha) f(y)$

$$(\alpha x + (1-\alpha) y)^2 \leq \alpha x^2 + (1-\alpha) y^2$$

$$\longrightarrow \alpha^2 x^2 + (1-\alpha)^2 y^2 + 2\alpha(1-\alpha) xy \leq \alpha x^2 + (1-\alpha) y^2$$

$$\longrightarrow (\alpha^2 - \alpha) x^2 + \underbrace{((1-\alpha)^2 - (1-\alpha))}_{(\alpha^2 - \alpha)} y^2 + 2\alpha(1-\alpha) xy \leq 0$$

CamScanner

$(\alpha^2-\alpha)=A \longrightarrow \quad Ax^2+Ay^2-2Axy \leq 0$

$$\underbrace{A}_{\text{منفی}}\underbrace{(x-y)^2}_{\text{مثبت}} \leq 0$$

چون $0 \leq \alpha \leq 1$

$\overbrace{\alpha^2 \leq \alpha} \longrightarrow \alpha^2-\alpha \leq 0$

تمامی مراحل را برگشت پذیر انجام دادیم پس

convex بودن x^2 و در نتیجه f را ثابت کردیم.

(د)

$f: R^n \to R$ is convex

so: $g: R^n \to IR \quad g=f(Ax-b)$ is convex

پس باید نشان دهیم.

$$g(\alpha x+(1-\alpha)y) \leq \alpha g(x)+(1-\alpha)g(y)$$

$0 \leq \alpha \leq 1$ and $x,y \in IR^n$

$$\longrightarrow g(\alpha x+(1-\alpha)y)=f(A(\alpha x+(1-\alpha)y)-b)$$

$$=f(A\alpha x+Ay-A\alpha y-b)$$

$$=f(\alpha(Ax-b)+(1-\alpha)(Ay-b))$$

$$\xrightarrow{f \text{ کانوکس بود}} \leq \alpha f(Ax-b)+(1-\alpha)f(Ay-b)$$

$$\xrightarrow{g(x)=f(Ax-b)} = \alpha g(x)+(1-\alpha)g(y)$$

اثبات شد.

$$\xrightarrow{\text{پس}} g(\alpha x+(1-\alpha)y) \leq \alpha g(x)+(1-\alpha)g(y)$$

هـ) ج ← اثبات می‌کند $f(x) = \|x\|_2^2$ convex است!

د ← اثبات شد $g(x) = f(Ax-b)$ convex است!

پس ← $f(x) = \|Ax-b\|_2^2$

نیز convex است!