Subject:
Date

تمرین DA 1 — محمد عرفان زارع زردینی

سؤال 1:

برای حل این سؤال ابتدا الگوی بازگشتی را که Top-Down است را توضیح می دهم و از روی آن و با توضیحات خود، الگوریتم DP خود را توصیف نمایم. در الگوی بازگشتی، ما از مقصد شروع می کنیم و می دانیم حداکثر ۸ حالت اسب می تواند حرکت کند. (دلیل واژه حداکثر این است که اگر در گوشه های برد بازی باشد، بعضی حرکات آن غیرقابل انجام می شود) از خانه های یافته شده نیز حداکثر ۸ حرکت قابل انجام است که این طریق همین طور ادامه دارد تا m حرکت از مقصد انجام شود و سپس بررسی می کنیم که به مبدأ رسیده ایم یا خیر. حال اما در این روش ما تعداد زیادی مراحل تکراری داریم. و این در تضاد با بهینگی است و علاوه بر این روش بازگشتی است.

حال بنابراین برای برطرف کردن این نواقص باید در الگوریتم تغییر ایجاد کنیم و الگوریتم جدیدی بدهیم.

در الگوریتم حل اصلی ما باید با استفاده از memorization و اینکه در مراحل برخی خانه ها و مسیرهای پیموده شده تکراری اند، شروع به حل می کنیم و این بار به جای اینکه هی برای پیمودن نیز به بازگشتی و در پی آن عقب ها باشیم، از نقطه مبدأ بررسی را انجام می دهیم و به جلو می رویم و در واقع از مسائل کوچکتر شروع به حل می کنیم و به سراغ مسیرهای بزرگ می رویم و در حل هر حرکت در خونه ها از مسیرهای قبلی و نتایج آنها استفاده می کنیم تا بدین وسیله چک کنیم که آیا در m حرکت می شود به مقصد رسید و از چند راه. برای شفاف تر شدن روش حل مثالی می آییم با توجه به اینکه هر حرکت حداکثر ۸ حالت دارد و همچنین پیموده شدن از یک خونه به خونه دیگر ایجاد یک مسیری کند، به حل سؤال می پردازیم. و بررسی می کنیم که در مسیر پیموده شدن از

یک نقطه به نقطه دیگر، چه مسیری با چند حرکت طی شده و برای محاسبه کردن هر کدام از زیرمسیرهایی که قبلاً محاسبه شده

پیموده شده است. در واقع با بررسی زیرمسیرها و پیدا کردن مقدار گام هر حالت ما می توانیم بررسی کنیم که

آیا با m گام می رود (به مقصد رسید یا خیر) و سایز تخته بازی (چون) روی الگوریتم ما تأثیری ندارد و برای سایزهای

کوچک و بزرگ هم پاسخ دهنده است.

پیچیدگی زمانی: برای ایجاد محیط زمین شطرنج و مختصات خانه ها $O(n^2)$ نیاز و همچنین $O(n^2)$ دیگر نیز برای تولید

$O(mn^2)$

آرایه ذخیره آنکه در خود یاد شده با چند مسیر و اینکه چند گام به آن خانه رسیده ایم ذخیره می شود. و برای بررسی و چک کردن

حرکت با m حرکت است. پس پیچیدگی $O(mn^2)$ است

پیچیدگی و سایز حافظه: $O(n^2)$ برای دو آرایه یاد شده در بخش قبل جا گرفته و بقیه موارد با اردر

$O(1)$ تولید می شوند

پایان سوال یک

سوال ۲:

$(a+1)\times(b+1)$

برای این سوال برای حل از dp که یک آرایه دو بعدی a تعداد شمش ها و b هم حداکثر وزن قابل قبول است

را تعیین می کنیم و بعد پیش می رویم که سطر و ستون صفر را همش را صفر می نویسیم. چون بی معنی است و فقط برای

آنکه آرایه همیشه از صفر شماره گذاری می شود و ما می خواهیم از قاطی شدن و گنگی جلوگیری کنیم به دست به این روش

پر کردن می‌کنیم) راه حل بررسی می‌کنیم با حساب این آتاشی‌ها، چه مقدار شمش با وزن کمتر مساوی w را می‌توان در کوله جای داد.

که برای این کار را از زیرمسئله‌هایی که در قبل پر شده استفاده می‌کنیم. و برای مثال وقتی وزن قابل تحمل ۱۰ است

و ما می‌خواهیم $K[3][9]$ را پر کنیم و با ۳ آتاشی توانستیم ۶ کیلو را پوشش دهیم، چک می‌کنیم تنها کیلیا شمش ۳ = ۶ - ۹ داریم یا

و اگر بود آن را برای حالت‌های زیرمسئله‌ی دیگر بررسی می‌کنیم تا در نهایت به خانه‌ای که ماکسیمم ستون آخر

است $K[0][b]$ برسیم.

پیچیدگی زمانی: برای پر کردن آرایه $O(n^2)$ و همچنین برای بررسی حالات و کمک گرفتن از زیرمسئله‌ها هم

دو تا for احتیاج می‌خواهد که از $O(n^2)$ در است. پس پیچیدگی زمانی از $O(n^2)$ می‌باشد.

پیچیدگی حافظه‌ای: آرایه دوبعدی جدول داریم که با ابعاد سؤال $O((a+1)(b+1)) = O(ab)$ یا $O((n+1)(w+1)) = O(nw)$ است

و متغیرهای دیگر هم از $O(1)$ اند پس آن در همان $O(nW)$ است.

پایان سؤال ۲

Subject: ۹۸۴۱۱۴۳۲
Date

محمدعرفان زارع زردینی

سوال ۳)

در حل این ماتریس $m \times n$ و یافتن مستطیل مدنظر، از یک آرایه هم اندازه آرایه اصلی استفاده می‌کنیم.
بدین گونه که آرایه کمکی بدین گونه پر می‌شود که به جز سطر اول که مانند سطر اول ماتریس اصلی پر می‌شود، الگوریتم
مقداردهی شدن آرایه کمکی بدین گونه است که در صورتی که مقدار خونه در آرایه اصلی صفر باشد، نظیر همان
خانه در آرایه کمکی نیز صفر قرار می‌گیرد. حال اما در صورتی که مقدار خانه در آرایه اصلی، یک باشد، باید به خونه
نظیر آن در آرایه کمکی (خانه هم آدرسش) برویم و خانه بالا سر آن را چک کرده و به مقدار آن یک واحد بیافزاییم. که این
روش در واقع مقدار خانه هایی که در یک ستون پشت هم یک اند تا آن سطر را نمایش می‌دهد. حال برای محاسبه
ماکزیمم سایز آرایه‌ی (زیر) مستطیل شکل که همش یک باشد کافی است در هر سطر ببینیم مقدار عدد غیر
گرفته و به شکل آرایه ۱ تا k که k مقدار متوالی است حساب کرد max را
صفر را که شامل همه مقادیر آن سطر است (اگر صفر نباشد در سطر) ها ببینیم مقدار غیر صفر را یافت چک می‌کنیم در یک
ردیف آن چند عدد غیر صفر متوالی است که ببینیم آن ضرب کرده و با متغیری که به عنوان max گرفته شده و در
مقدار را — آن را به شکل آرایه 1 تا k و
ابتدا صفر است (پیش از ساختن آرایه کمکی) مقایسه و در نهایت متغیر max به عنوان جواب باز گردانده می‌شود که
سایز مستطیل زیر ماتریس مدنظر است. توجه: سوال از ما محدوده شروع و پایان زیرآرایه و (در) $a \times b$ خود $b \times a$
را خواسته (مقدار حاصل ضرب ab که همان تعداد 1 در زیرآرایه و سایز است) اما می‌شود با توجه به جدول کمکی
(البته محاسبات در شکل وسط به طور توضیح دادم)

PAPCO

Scanned with CamScanner

Subject:
Date

و سطر یافته شده عنصر max و ذخیره index سطر را در آخر به سراغش رفته و عنصر مینیمم را یافته و براساس مقدار آن

منهای یک ها به سمت بالا حرکت می کنیم که سطر شروع و مقدار مینیمم (حفره آن) را اگر بیابیم و به مقدار بدست آمده بالا رویم

index استارت (اگر سمت راست باشه مینیمم جفت آخر زیر آرایه) را یافته ایم.

مثال برای نمایش پیاده سازی

$$A=\begin{array}{c|cccccccc|}
 & ۰ & ۱ & ۲ & ۳ & ۴ & ۵ & ۶ & ۷ \\
۰ & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\
۱ & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
۲ & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
۳ & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\
۴ & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
۵ & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
۶ & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1
\end{array}$$

روش پر کردن جدول کمکی: سطر اول که یکسانه با اصلی و برای سطرهای دیگر

از روش توضیح داده شده در صفحه قبل استفاده می کنیم. مثال اگر $A[i][j]=1$ باشد

به سراغ جدول کمکی می رویم و آن را بدین گونه و با فرمول زیر پر می کنیم.

$$B[i][j] = A[i][j] + B[i-1][j]$$

سپس پس از پر کردن جدول کمکی (B) از الگوریتم است استفاده کرده و سطر max را می یابیم. پس مقدار مینیمم در سطر یافته

و نقطه شروع می شود اولین خونه از سمت چپ که غیر صفره و شامل مقدار مینیمم است (برابر یا مساوی مینیمم) و نقطه پایان ستون

ستون هم آخرین خونه ای که متوالی غیر صفر را پوشش داده و شامل مقدار مینیمم باشد. جدول زیر جدول کمکی است

$$B=\begin{array}{c|cccccccc|}
 & ۰ & ۱ & ۲ & ۳ & ۴ & ۵ & ۶ & ۷ \\
۰ & ۰ & ۱ & ۱ & ۱ & ۰ & ۱ & ۰ & ۰ \\
۱ & ۱ & ۲ & ۰ & ۰ & ۱ & ۲ & ۱ & ۰ \\
۲ & ۲ & ۳ & ۱ & ۱ & ۲ & ۳ & ۰ & ۰ \\
۳ & ۰ & ۴ & ۲ & ۲ & ۳ & ۰ & ۱ & ۱ \\
۴ & ۱ & ۵ & ۳ & ۳ & ۴ & ۱ & ۲ & ۲ \\
۵ & ۰ & ۰ & ۴ & ۴ & ۵ & ۲ & ۰ & ۳ \\
۶ & ۱ & ۱ & ۵ & ۵ & ۰ & ۳ & ۱ & ۴
\end{array}$$

max=۱x۳ maxSatr=۰
max=۳ maxSatr=۰
max=۱x۶ maxSatr=۲
max=۴x۲ maxSatr=۳
max=۳x۴ maxSatr=۴
max=۳x۴ maxSatr=۵
max=۱۲ maxSatr=۵ ⇒ max=۳x۴

سطر پایانی = سطر ۵

PAPCO

Scanned with CamScanner

Subject:
Date

تایم زمانی: اردر ساخت آرایه کمکی $O(m \times n)$ و اردر یافتن جواب $O(m \times n \times k)$

پیچیدگی حافظه‌ای: دو تا آرایه $m \times n$ که $O(m \times n)$ است. همچنین چک کردن برای یافتن max با $t1$ با k آرایه که پشت هم اند به مقدار $O(k)$ است.

درستی الگوریتم: توضیحات درستی الگوریتم در حل سوال توضیح داده شده. اما برای توضیحات بیشتر به دلیل آنکه ما بر اساس این که اعداد سطر و ستون متوالی یک است پیش می‌رویم که توضیحات آن در مراحلش در حل موجود است و همچنین ما سطر شروع زیرماتریس و اندازه آن را داریم به دست می‌آوریم و بدین شکل ایندکس شروع و پایان و سایز و ابعاد ماتریس مدنظر را بدست می‌آوریم و بدین گونه جواب صحیح بازگردانده می‌شود.

پایان سوال ۳

سوال ۴)

در این سوال الگوریتم بدین گونه است که بعد از گرفتن جدول و دو آرایه که دو تا مسیر اولیه را بررسی کنیم (خانه‌های اول آرایه‌ها) که حالت پایه اند حال برای خانه‌های جلوتر به بررسی می‌کنیم که خانه‌های هم ایندکس که در کدام یک از دو آرایه بزرگتره و (همچنین دو آرایه داریم که خودمان ایجاد می‌کنیم که شماره های آرایه مسیر را در آن [illegible] می‌کنیم) سپس آرایه [illegible] کننده

PAPCO

شماره: آرایه شده دم یک از آرایه 1 یا 2) را چک می‌کنیم که در گام قبلی آن مسیر که در کدام آرایه رفته است. اگر انتخاب کمینه گام

قبلی مینیمم بوده که در آرایه مسیر ثبت می‌شود (شماره آرایه) حال اما اگر یکی نبوده، باید چک کنیم که تغییر مسیر با هزینه وجود

حتماً در آرایه مخالف نسبت به مسیر مستقیم صرفه دارد (یعنی $A[i] < B[i] + P$) که در آن صورت به مسیر دیگر می‌رویم و

در غیر این صورت همان مسیر مستقیم را ادامه می‌دهیم. بجای ورودی هم اگر مستقیم بره خود مقادیر و اگر تغییر رفت،

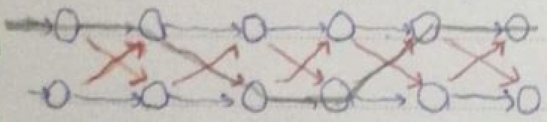
هزینه P از آن کم می‌شود. پیچیدگی زمانی: $O(n)$ که همان ساخت دو آرایه و بعدی و دو آرایه مسیر انتخابی است

پیچیدگی مکانی: $O(n)$ که ۲ آرایه‌ها اندازه یک بعدی اند

Subject:
Date

اثبات درستی سوال ۴: ما می خواهیم نشان دهیم که اگر با DP بردیم جواب درست و بهینه را می یابد. فرض کنید دایره

ها مقادیر انرژی اند. بین هر خط جابجا می شوم و هزینه ای است. شکل زیر را رسم می نماییم و داریم:



حال فرض می کنیم روش ما جواب را خط مشکی بدست آورده ولی جواب یک مسیر دیگه است و

مسیر ما جواب درست را نداده است. ما جوابمان تا یک قسمتی با جواب بهینه یکی بوده و یک جاهایی با آن

فرق می کرده است. (از راه حل ما فرض کردیم مقدار کمتری نسبت به روش فرضی بهینه می یابد). حال ما می رویم

اون تیکه قبل یکی بودن دو راه حل را برداشته و به جای روش فرضی بهینه، روش خودمان را می گذاریم که می بینیم جواب ما

بیشتر می شود و طبق آن جواب بزرگ‌تر یعنی تیکه مشترک یکی است و ما دقیقا داشتیم حل می کردیم زیرمسئله از (t, n) که یافتیم بیشترین

مقدار را داشته. حال یا مسیر فرضی جوابش با مسیر ما یکی است یا مسیر ما جواب بالاتری بدست می آید.

همچنین ما با جایگزین کردن دو در خصوص بالا نشان دادیم که مسیر فرضی مقدار کمتری از مسیر ما دارد و در نتیجه

راه حل مان را درستی اش را اثبات نمودیم (ما با جایگزینی مسیر خود با مسیر بهینه به مقدار بیشتر از

بهینه رسیدیم پس که روش مان را هم اثبات نمودیم.)

پایان سوال ۴

سوال ۵

در دنباله فیبوناچی ها می دانیم $f(0)=0$ و $f(1)=1$ و مقادیر دیگر از جمع دو مقدار قبلی حاصل می شوند.

حال با توجه به نکته بیان شده ساختار خروجی به شکل زیر می شود. چون ما رقم یکان را می خواهیم و مقدار عددی

خانه های دنباله پس از مدتی بسیار زیاد می شود، ما تنها یکان آن را ذخیره می نماییم.

$$f[i]\begin{cases} 0 & \text{if } n=0 \\ 1 & \text{if } n=1 \\ (f[i-2]+f[i-1])\%10 & \text{if } n>1 \end{cases} \qquad sum += f[i]$$

خب برای حل سوال همانطور که مشخص بیان شده مجموعه F را ساخته که سایز آن n است که

از ورودی می گیریم و براساس جدول بالا مقادیر را درونش می ریزیم. و همچنین به متغیر sum نیز می افزاییم

و در نهایت یکان مقدار sum را برمی گردانیم و چاپ می کنیم

پیچیدگی زمانی: یک حلقه برای آرایه F داریم. پس از اردر $O(n)$ می باشد و شرطها و حلقه سازنده

پیچیدگی حافظه: آرایه F از اردر $\underline{\underline{O(n)}}$ است و بقیه متغیرها هم چون sum و $O(1)$ است.

اثبات درستی روش: توضیحات و روش حل دلیل درستی در توضیحات حل موجود است اما برای توضیحات بیشتر

با توجه به آنکه سوال یکان مجموع n عدد اول فیبوناچی را می خواهد (0 تا n-1) و آنکه اعداد فیبوناچی از یک مقدار n بسیار بزرگ

می شود نمی توان آنرا ذخیره نمود. برای آنکه این مشکل را حل کنیم، با توجه به آنکه مجموع دو عدد، یکانش حاصل تولید جمع یکان های اشان

که باقی مانده آن را بر 10 می گیریم، بهترین روشی که مشکلی برای ذخیره عدد ندارد همین است که حتی اگر جمع یکان ها را هم بزنیم و در نهایت

sum % 10 بگیریم، باز جواب را می یابیم.

پایان سوال پنج

سوال ۶ (امتیازی)

الگوریتم مورد نظر برای حل این سوال این است که ابتدا مشخصات سوال را می نویسیم. n تعداد اعداد و n-1 عملگر

ورودی هست. حال ما برای حل و یافتن بیشترین مقدار قابل تولید با پرانتز گذاری، مانند سوال (matrix-chain multiplication)

مانند بخش از a_i تا a_j را با پرانتز به دو بخش $A_i \ldots A_k \; op_k \; A_{k+1} \ldots A_j$ تقسیم می کنیم و مقدار آن ها را محاسبه می کنیم.

هرکدام از این دو بخش ها خود نیز می توانند به بخش های دیگر تقسیم شوند که البته مقادیر شکل در قبل در مراحل قبل تر محاسبه شده و ما از

آن استفاده می کنیم و max و min که دلیل min نگه داشتن این است که حاصل ضرب دو عدد منفی مثبت می شود.

Subject:
Date

در واقع ما با استفاده از روش حل مسئله matrix chain multiplication مقادیر هر پرانتز را محاسبه و ایجاد در خروجی

max جواب را می‌یابیم.

اردر زمانی: $O(n^2)$ برای پرانتز گذاری و n بخش دارند اعداد که در نتیجه می‌شود اوردر $O(n^3)$

اردر حافظه: از اردر $O(n)$ برای ذخیره مقادیر در آرایه.

اثبات درستی:

در حل این سؤال با استفاده از این که ما یکسری بخش‌ها را قبلا محاسبه کردیم یا ماکسیمم یک تکه را داریم که از حل دوباره آن

ها خودداری می‌کنیم (همان memorization) و از آنها برای حل ادامه سؤال استفاده می‌کنیم که این کار سبب

بهینه شدن زمان مصرفی می‌شود. روش حل این سؤال بر پایه‌ی memorization برای بهینه سازی آن می‌باشد.

سودوکد آن هم مانند زیر است:

```
Max(a)
n ← Length[a] - 1
For i ← 0 to n
    do for j ← i to n
        m[i,j] ← -∞
        For k ← i to j - 1
            do b ← find(m[i,k] , m[k+1,j])
        max برای - If b > m[i,j]
                Then m[i,j] ← b
                f1[i,j] ← k
        min برای - If c < m[i,j]
                Then m[i,j] ← c
                f2[i,j] ← k
```

پایان سؤال شش و تمرین سری اول

PAPCO