دانشکده مهندسی
کامپیوتر و فناوری اطلاعات

12/3/2020

Homework 4

Dynamic programming & Greedy Algorithms

ALGORITHM DESIGN

۱))الگوریتمی با زمان اجرای $O(n^2)$ ارائه دهید که با استفاده از برنامه نویسی پویا بلندترین زیردنباله یکنوای صعودی را از یک دنباله اعداد نامرتب پیدا کند . (توضیح کافی است و نیازی به شبه کد نیست) (۱.۵ نمره)

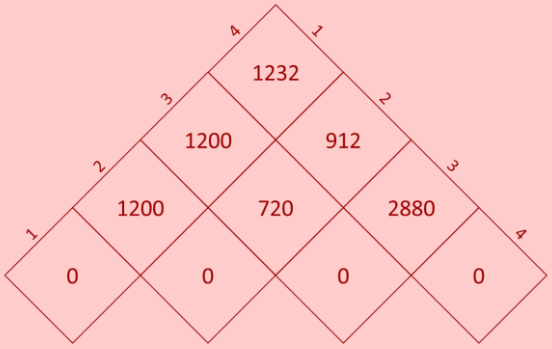
اگر فرض کنیم دنباله ورودی A باشد، از آنجا که گفته شده باید این زیر دنباله صعودی باشد، یک آرایه دیگر (A_s) درست میکنیم و در آن، دنباله ورودی را کپی میکنیم. حال باید این آرایه را بصورت صعودی sort کنیم. (برای این کار از الگوریتمی استفاده میکنیم که زمان آن از n^2 کمتر باشد.) برای این که یک زیر دنباله یکنوای صعودی از دنباله ورودی داشته باشیم، کافی است زیر دنباله ای مشترک بین آرایه ورودی و آرایه کپی شده (که آن را مرتب کردیم) را به کمک الگوریتم LCS پیدا کنیم. مطمئن هستیم که این زیر دنباله یکنوای صعودی خواهد بود، زیرا زیردنباله‌ی دنباله یکنوای صعودی که خودمان ایجاد کردیم میباشد. زمان انجام این کار از مرتبه حاصل ضرب اندازه دو دنباله ای است که الگوریتم LCS را روی آن ها اجرا کردیم. و از آنجا که هر دو دنباله طول یکسانِ n داشتند میتوان گفت که زمان اجرا برابر خواهد بود با:

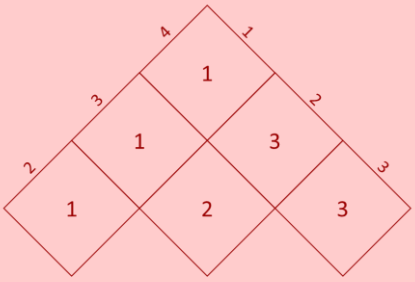
$$O(|A_s| \times |A|) = O(n \times n) = O(n^2)$$

۲) با استفاده از برنامه نویسی پویا بهترین حالت ضرب ماتریس های زیر را بدست آورید . (۲ نمره)

$$A(20 \times 2) \times B(2 \times 30) \times C(30 \times 12) \times D(12 \times 8)$$

$P = <20,2,30,12,8>, n = P.length - 1 = 5 - 1 = 4$





$m[1,1] = m[2,2] = m[3,3] = m[4,4] = 0$

$\mathbf{m[1,2]} = m[1,1] + m[2,2] + P_0P_1P_2 = 0 + 0 + (20 \times 2 \times 30) = \mathbf{1200}, \mathbf{s[1,2]} = \mathbf{1}$

$\mathbf{m[2,3]} = m[2,2] + m[3,3] + P_1P_2P_3 = 0 + 0 + (2 \times 30 \times 12) = \mathbf{720}, \mathbf{s[2,3]} = \mathbf{2}$

$\mathbf{m[3,4]} = m[3,3] + m[4,4] + P_2P_3P_4 = 0 + 0 + (30 \times 12 \times 8) = \mathbf{2880}, \mathbf{s[3,4]} = \mathbf{3}$

$$\mathbf{m[1,3]} = \begin{cases} m[1,1] + m[2,3] + P_0P_1P_3 = 0 + 720 + (20 \times 2 \times 12) = \mathbf{1200}, \mathbf{s[3,4]} = \mathbf{1} \\ m[1,2] + m[3,3] + P_0P_2P_3 = 1200 + 0 + (20 \times 30 \times 12) = 8400 \end{cases}$$

$$\mathbf{m[2,4]} = \begin{cases} m[2,2] + m[3,4] + P_1P_2P_4 = 0 + 2880 + (2 \times 30 \times 8) = 3360 \\ m[2,3] + m[4,4] + P_1P_3P_4 = 720 + 0 + (2 \times 12 \times 8) = \mathbf{912}, \mathbf{s[2,4]} = \mathbf{3} \end{cases}$$

$$\mathbf{m[1,4]} = \begin{cases} m[1,1] + m[2,4] + P_0P_1P_4 = 0 + 912 + (20 \times 2 \times 8) = \mathbf{1232}, \mathbf{s[1,4]} = \mathbf{1} \\ m[1,2] + m[3,4] + P_0P_2P_4 = 1200 + 2880 + (20 \times 30 \times 8) = 8880 \\ m[1,3] + m[4,4] + P_0P_3P_4 = 1200 + 0 + (20 \times 12 \times 8) = 3120 \end{cases}$$

$$\boxed{\mathbf{(A)((BC)(D))}}$$

۳) مسئله کوله پشتی زیر را که در آن گنجایش کوله ۳۰ کیلوگرم است در نظر بگیرید. به سه روش گفته شده مسئله را حل کنید. (۳ نمره)

قطعه	ارزش	وزن
۱	۵۰	۵
۲	۶۰	۱۰
۳	۱۴۰	۲۰

الف) با استفاده از الگوریتم حریصانه و با انتخاب صفر و یکی اشیا

ب) با استفاده از الگوریتم حریصانه و امکان انتخاب کسری از اشیا

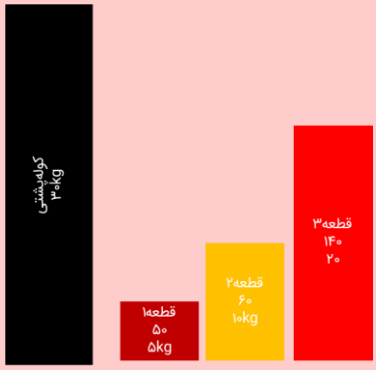
ج) با استفاده از روش برنامه نویسی پویا

نسبت ارزش به وزن هر یک از قطعات را محاسبه میکنیم:

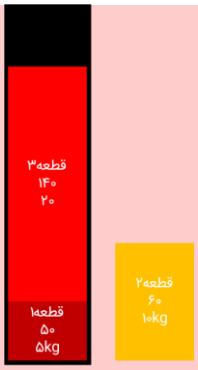
قطعه ۱ : $\frac{50}{5} = 10$ ☆☆☆

قطعه ۲ : $\frac{60}{10} = 6$ ☆

قطعه ۳ : $\frac{140}{20} = 7$ ☆☆



الف) الگوریتم حریصانه و با انتخاب صفر و یکی اشیا



ب) با استفاده از الگوریتم حریصانه و امکان انتخاب کسری از اشیا

ج) با استفاده از روش برنامه نویسی پویا

از آنجایی که وزن ها مقدار 5k دارند، فقط وزن های مضارب ۵ را برای حل بازگشتی مینویسیم.

ظرفیت ←	۰	۵	۱۰	۱۵	۲۰	۲۵	۳۰
قطعه ۱	۰	۵۰	۵۰	۵۰	۵۰	۵۰	۵۰
قطعه ۲	۰	۵۰	۶۰	۱۱۰	۱۱۰	۱۱۰	۱۱۰
قطعه ۳	۰	۵۰	۶۰	۱۱۰	۱۴۰	۱۹۰	**۲۰۰**

۴) فرض کنید در مسئله activity-selection به جای انتخاب اولین کاری که میخواهیم تمام کنیم آخرین کاری را میخواهیم شروع کنیم مشخص کنیم. توضیح دهید که چرا این روش، روش حریصانه هست و چرا یک راه حل بهینه تولید می کند . (۱.۵ نمره)

اگر فرض کنیم همان مسئله اصلی در جهت برعکس (از نظر زمانی) اجرا شود به این مسئله میرسیم. پس همانند مسئله اصلی راه حل بهینه را ایجاد میکند. این روش چون در هر مرحله بهترین انتخاب را انجام میدهد میتوان گفت که روشی حریصانه است.

۵) فرض کنید که یک فایل حاوی دنباله ای از کاراکترهای ۸ بیتی داریم طوری که همه‌ی ۲۵۶ کاراکتر تقریبا به میزان یکسانی متداول هستند و در واقع بیشترین تعداد دفعات تکرار کاراکترها کمتر از دو برابر کمترین دفعات

تکرار کاراکترها میباشد. توضیح دهید چرا کارایی الگوریتم کدگذاری هافمن در این حالت بیشتر از کارایی استفاده از یک کد معمولی با طول ثابت ۸ بیت برای هر کاراکتر نیست. (۲ نمره)

برای هر دو کاراکتر، مجموع تعداد دفعات تکرار آنها از نعداد دفعات تکرار هر کاراکتر دیگری بیشتر است. بخاطر همون موضوع در ابتدا کد گذاری هافمن ۱۲۸ درخت با تعداد ۲ برگ تولید میکند. در مرحله بعد، هیچ کدام از گره ها لیبلی ندارد که از دو برابر لیبل دیگری بیشتر باشد. پس تفاوت چندانی نخواهیم داشت.با ادامه دادن همین روش، کدگذاری هافمن یک درخت کامل دودویی با ۲۵۶ گره میسازد که ارتفاعی برابر با ۸ خواهد داشت که کارایی بیشتری نسبت به کدهای معمول 8 بیتی ندارد.

- مهلت ارسال تمرین ساعت ۲۳:۵۵ روز **جمعه ۷** آذرماه می‌باشد.
- سوالات خود را می‌توانید از طریق ایمیل از تدریسیاران بپرسید.
  - parsa.abdollahi.pa@gmail.com
  - faridi.mina.1@gmail.com
- ارائه پاسخ تمرین به دو روش ممکن است:
  ۱) تایپ داخل همین فایل و ارائه فایل Pdf
  ۲) چاپ تمرین و پاسخ دهی به صورت دستنویس خوانا
- ارائه تمرین به روش اول شامل ۱۰٪ نمره امتیازی می‌گردد.
- فایل پاسخ تمرین را تنها با قالب **HW3-9531747.pdf** در مودل بارگزاری کنید.
- فایل زیپ ارسال **نکنید**.