Year: Month: Day: Subject:

مرتضی نعلبندی توضیحات پروژه ۲ ۹۸۳۵۹۸۳

((DP))

پروژه اول: مسئله ی پارکراف

برنامه از ۳ بخش اصلی تشکیل شده است.

۱- گرفتن ورودی ها و یافتن طول کلمات

۲- پیدا کردن کمترین هزینه

۳- چاپ کردن کمترین هزینه و پارکراف آن

در بخش اول در تابع main انجام می شود و هر بار کلمات گرفته شده و همزمان طول آن ها حساب می شود و مقدار M نیز که تعداد کل کاراکترهای یک خط است نیز نوشته می شود.

سپس برای محاسبه کمترین هزینه وارد تابع findMinCost می شویم

این تابع ورودی های زیر را دارد (به صورت شبه کد):

طول کلمات: long[] Length [0→n-1]

NOOR

Year: Month: Day: Subject:

String[] words[0->n-1] خود کلمات

int n; طول آرایه ها

long M; حداکثر کلمات هر سطر

در این تابع دو آرایه دو بعدی مرتبه [n+1][n+1] آرایه یک بعدی کلی مرتبه n+1

را به که همانطور که در کد گفته شده برای:

① extra spaces [n+1][n+1] long از نوع

این تابع دو بعدی برای این تعریف شده که برنامه در صورت قرار گرفتن

تعدادی کلمه پشت سر هم در یک خط، چند خانه خالی در انتهای خط

باقی می ماند، حداقل انتظار ما قرار گرفتن چند کلمه کنار هم و

طولشان از M بزرگ تر نشود، این مقدار منفی است

② costs [n+1][n+1] long از نوع

برای هزینه ناشی از قرار گرفتن چند کلمه کنار هم است

③ leastCost [n+1] long برای کمترین هزینه ممکن برای اگر اف

در صورت تشکیل آن از کلمه اول تا کلمه ای خاص

NOOR

Year: Month: Day: Subject:

۴- برای تعیین محل [illegible] ابتدایی و انتهایی ها، گراف جهت چپ ... print[n+1] int

داریم:

① extraSpaces[i][j] = es[i][j]

- $es[i][j] \geq 0$: امکان تشکیل خط برای کلمات i تا j
- $es[i][j] < 0$: عدم ...
- $es[i][j] = 0$: پر شدن خط با تشکیل کلمات i تا j

② costs[i][j]

- $(es)^3$: ① امکان تشکیل خط برای کلمات i تا j (extraspaces[i][j] ≥ 0)
- 0 : ② ... و $j = n$
- $+\infty$: ③ عدم امکان تشکیل خط برای کلمات i تا j (extraspaces[i][j] < 0)

max ← همان ممکن

برای leastCost فرمول زیر را داریم

$$
leastCost[j] = \begin{cases} 0 & j = 0 \\ \min\limits_{1 \leq i \leq j} \left(c[i-1] + costs[i][j] \right) & j > 0 \end{cases}
$$

Year: Month: Day: Subject:

در تابع ، در صفحه دو مجموعه حلقه ی تو در تو ، به ترتیب مقادیر

costs ، را ایجاد کرده و سپس درون یک حلقه تو در تو تمامی

مقادیر leastcosts را ایجاد می کنیم ، سپس مقدار least cost[n] که همان

مقدار درخواستی است را چاپ خواهیم کرد.

در حلقه نهایی ، هر بار ، ابتدای مکانی را که پاراگراف در آن شروع

می شود را به این صورت نگه می داریم

print[j] = i

آخرین ← اولین

سپس درون تابع ، printsolution به صورت بازگشتی مقدار خواسته شده

را چاپ می کنیم

Year: Month: Day: Subject:

پیچیدگی زمانی

(۱) for[i → 0 to n-1] گرفتن ورودی ها → $\Theta(n)$

(۲) پیدا کردن edges ها با n بار for[i → 1 to n]

for[j → i+1 to n] تشخیص → $\Theta(n^2)$

(۳) پیدا کردن costs

for[i → 1 to n]

for[j → 1 to n] → $\Theta(n^2)$

(۴) پیدا کردن lestcosts و pal ها

for[i → 1 to n]

for[j → 1 to n] → $\Theta(n^2)$

(۵) تابع بازگشتی به اندازه n بار فراخوانی می‌شود → $\Theta(n)$

پس $T(n) = \Theta(n)$

Scanned with CamScanner

Year: Month: Day: Subject:

string words[n] → $an = \Theta(n)$ پیچیدگی حافظه

long Length[n] → $bn = \Theta(n)$

long m → $b = \Theta(1)$ ، int n → $c \rightarrow \Theta(1)$

long es[n+1][n+1] $= (n+1)(n+1)b = n^2 + 2(n+1)b \rightarrow \Theta(n^2)$

long costs [n+1][n+1] $= (n^2 + 2(n+1))b \rightarrow \Theta(n^2)$

long leastCost [n+1] $= (n+1)b \rightarrow \Theta(n)$

int print [n+1] $= (n+1)c \rightarrow \Theta(n)$

(1)

پس: $S_T(n) = \Theta(n^2)$

به این دلیل آرایه ها (1) از 1 تا n+1 گرفته شده از ۰ مفهوم را حذف

تر شود → es[i][j] → (i+1) تا (j+1) کلمه از 1 تا n بوده

از 1 تا n+1 بوده → es[i][j] → i تا j کلمه

در واقع خانه [0] برای تک یعنی ها و خانه های ستون و ردیف اول

(x[i][0] و x[0][i]) بی استفاده هستند

NOOR