The Postal Owl

Logged in: Zdeněk Tomis (home) Logout

Algoritmy a datové struktury 1

Back to the course

Bonusový DÚ 1: Největší podmatice

Deadline: 2022-03-10 10:40 (53 days ago)

Pavel Veselý — modified 2022-02-28 16:39 (63 days ago) — reply

Pro (celočíselnou) matici s M řádky a N sloupci nalezněte největší souvislou (obdélníkovou) nulovou podmatici, tedy největší obdélník, který obsahuje samé nuly. Jiná formulace: máte černobílý obrázek zadaný jako dvourozměrné pole nul a jedniček a chcete v něm najít největší černý obdélník.

Plný počet bodů je za lineární časovou složitost, ale i za asymptoticky horší řešení budou nějaké body. Zkuste nejprve přijít na řešení, které určitě funguje, i když pomalu a pak ho zlepšovat (při zachování správnosti).

Vhodný úkol, pokud se vám nechce programovat na RAMu :-)

Pavel Veselý — 2022-03-04 16:15 (59 days ago) — reply

PS: Doporučoval bych nejdřív přijít na řešení, které určitě funguje, byť třeba s horší než lineární časovou složitostí (jak jsem psal, i za to budou body), a teprve potom ho zkusit optimalizovat.

Zdeněk Tomis — 2022-03-06 22:02 (56 days ago) — edit — reply

Pomocná matice

Nechť matice má název a$ Při prvním průchodu propočtěme pomocnou matici $aux[][]$ o velkosti $m \times n$, pro kterou platí:

$$aux[i][j] = \begin{cases} 0 & \text{pokud } a[i][j] \neq 0 \\ \text{výška sloupce nul nad } a[i][j] \text{ včetně} & \text{pokud } a[i][j] = 0 \end{cases}$$

Toto jsme schopni spočítat v čase $O(mn)$, můžeme matici projít po sloupcích shora dolů, každý sloupec zabere čas $O(m)$, prostě ukládáme délku koncového úseku nul.

Plocha pod sloupci v každém řádku

Pro každý řadek matice aux vezememe jako sérii sloupců, histogram, pro který jsme schopni spočítat maximální plochu pod sloupci.

Brute force řešení by bylo na v čase $O(n^2)$ pro každý řádek: pro každou dvojici sloupců jako ohraničení nalezneme největší možný obdelník pod nimi, a to optimalizujeme tak, že pro každý jeden sloupec začneme sloupcem hned za ním a průběžné minimum všech sloupců mezi danou dvojcí si pamatujeme.

Existuje ale lepší algoritmus, který je každý řádek schopný spočítat v čase O(n): Uvažme, že každý kandidát na optimální obdelník má výšku nějakého sloupce, který pod kterým je. Kdyby byl nižší, mohli bychom ho zvýšit a jeho obsah zvětšit. Podívejme se tedy na každý sloupec a uvažme obdelník, který je shora omezen právě tímto sloupcem. Abychom zjistili jeho obsah, musíme zjistit jeho šířku, a to nalezením prvního nižšího sloupce napravo a nalevo od zkoumaného sloupce.

Optimálně sloupce projdeme takto: Udržujeme zásobník sloupců, u kterých ještě neznáme pravé ohraničení odpovídajících obdelníků. (Tento zásobník bude určitě sestupný, kdybychom nejdříve vytáhli menší sloupec A a pak větší sloupec B, pak by určitě A bylo pravým ohraničením obdelníku s výškou sloupce B). Když iterujeme přes nový sloupec S, vytahujeme ze zásobníku sloupce, dokud jsou větší, protože těm bude sloupec S tvořit ohraničení zprava, pro ty můžeme spočítat onen obsah, protože známe výšku, ohraničení zprava a ohraničení zleva bude další prvek zásobníku (nebo levý kraj, když je zásobník prázdný). Jakmile už žádný vyšší sloupec na zásobníku nemáme, přidáme tam S a jdeme dál. Když jsme na konci řádku, sloupce zbylé v zásobníku budou odpovídat obdelníkum, jejichž pravé ohraničení je pravý okraj, tak je opět postupně projdeme a zásobník vyprázníme. Takto projdeme všechny prvky a pamatujeme si maximální obsah.

Pro každý řádek projdeme lineárně všechny sloupce a udržujeme u toho zásobník, do kterého každý sloupec maximálně jednou přidáme a odebereme, tedy $O(n + n) = O(n)$

Pro každý řádek máme maximální obsah pod histogramem, maximum těchto maxim bude náš hledaný obdelník.

Závěr

Pomocnou matici vypočteme v $O(mn)$. Pro každý řádek pak v čase $O(n)$ najdeme maximum, celkem $O(mn + mn) = O(mn)$

Algoritmus je správný, protože určitě prozkoumá každý možný největší obdelník. Zavrhne pouze ty, o kterých víme, že jdou protáhnout nahoru, a ty optimální jistě nebudou (podrobněji rozvedeno v sekci výše).

Algoritmus je konečný, není možné, aby se někdě zacyklil.

Kódy pro ilustraci

Pro objasnění připojuji funkční kód v pythonu. Není nutný pro pochopení, ale v případě nejasnosti mého vysvětlení může pomoci.

Výpočet pomocné matice

```
aux = [[None for y in range(n)] for x in range(m)]

for row in range(m):
    for column in range(n):
        aux[row][column] = int(matrix[row][column] == 0)
        if row > 0 and matrix[row][column] == 0:
            aux[row][column] += aux[row-1][column]
```

Výpočet obsahů

```
maxArea = 0
stack = []

def updateMaxArea(leftLimit, rightLimit, height):
    global maxArea
    newArea = (rightLimit - leftLimit - 1) * height
    maxArea = max(maxArea, newArea)

for row in range(m):
    for column in range(n):

        currentHeight = aux[row][column]

        while len(stack) > 0 and aux[row][stack[-1]] > currentHeight:

            c = stack.pop(-1)
            updateMaxArea(leftLimit=stack[-1] if len(stack) > 0 else -1,
                          rightLimit=column,
                          height=aux[row][c] )


        stack.append(column)

    while len(stack) > 0:
        c = stack.pop(-1)

        updateMaxArea(leftLimit=stack[-1] if len(stack) > 0 else -1,
                      rightLimit=n,
                      height=aux[row][c])

print(maxArea)
```

Pavel Veselý — 2022-03-07 20:02 (56 days ago) — reply

Nejsem přesvědčen, že toto řešení funguje obecně, a myslím, že následující matice dává protipříklad:

```
1 0 1 0
0 0 0 0
1 0 0 0
0 0 0 0
```

Řešením má být matice 3×3, nicméně, pokud to chápu dobře, tak pole aux pro 4. řádek je `1 4 3 4`, takže se na zásobník přidá sloupec 1, pak sloupec 2, pak se odebere sloupec 2 a přidá se 3, dále se přidá sloupec 4, takže na konci je obsah zásobníku `1 3 4`, takže na řešení 3×3 algoritmus nepřijde. Nicméně dávám 5 bodů za řešení v $O(n^2 m)$.

Points: 5.00

Zdeněk Tomis — 2022-03-08 09:24 (55 days ago) — edit — reply

Podle mě to funguje, ten program to nalezl, respektive nalezl obsah 9 v posledním řádku, když jsem ho spustil. Možná jste přehlédl tuto větu:

> Když jsme na konci řádku, sloupce zbylé v zásobníku budou odpovídat obdelníkum, jejichž pravé ohraničení je pravý okraj, tak je opět postupně projdeme a zásobník vyprázníme.

Algoritmus tak projde opravdu všechny obdelníky dané sloupce pro každý sloupec toho pomyslného historgramu. Pro průchodu řádkem bude vždy zásobník prázdný, není možné, aby v zásobníku něco zůstalo.

Pavel Veselý — 2022-03-11 13:26 (52 days ago) — reply

Že se zásobník na konci postupně vyprázdní mi neuniklo, ale potřeboval jsem pochopit toto: když odebíráme sloupec c ze zásobníku, tak nechť c' je předchozí sloupec na zásobníku a pak mezi c' a c není žádný sloupec c'' nižší než sloupec c — jinak by byl na zásobníku (přesněji se to ukáže přes "minimální protipříklad"). Tohle mi v řešení chybělo nebo jsem to přehlédl, nicméně už vidím, že řešení je správně.

Points: 12.00

Pavel Veselý — 2022-03-11 13:31 (52 days ago) — reply

Hint: pro každé nulové políčko si předpočítáme, kolik je nad ním a pod ním nul, a potom si všimneme, že největší obdélníková podmatice musí být zleva (i z jiné strany) ohraničena jedničkou nebo okrajem matice.

(Toto je hint na jedno konkrétní řešení, existují ale i jiná lineární řešení.)

Return

New post (You can use Markdown with KaTeX math here)

Attachment (PDF or UTF-8 text): Durchsuchen… Keine Datei ausgewählt.

Submit Preview

The Owl is maintained by Martin Mareš. Send all suggestions, bug reports, and requests for new courses to mj@ucw.cz.