מסדי נתונים - תרגיל 4 - Join Algorithms

רועי בן יוסף - 307920116

19 בדצמבר 2020

שאלה 1

1.
א. עלות החישוב של BNL היא:

$$B(Movies) + B(PlaysIn) \cdot \left\lceil \frac{B(Movies)}{M-2} \right\rceil$$

נחשב את כל אחד מהאלמנטים:

$$B(Movies) = \frac{T(Movies)}{\left\lfloor \frac{8192}{36} \right\rfloor} = \left\lceil \frac{10000}{227} \right\rceil = 45$$

$$B(PlaysIn) = \frac{T(PlaysIn)}{\left\lfloor \frac{8192}{18} \right\rfloor} = \left\lceil \frac{100000}{445} \right\rceil = 220$$

$$M = 15$$

ולכן עלות החישוב היא:

$$45 + 220 \cdot \left\lceil \frac{45}{13} \right\rceil = 925$$

כלומר 925 פעולות IO
ב. התנאי לקיום אלגוריתם $Hash - Join$ הוא :

$$\left\lceil \frac{B(Movies)}{14} \right\rceil \leq 13 \quad \vee \quad \left\lceil \frac{B(PlaysIn)}{14} \right\rceil \leq 13$$

ומתקיים:

$$\left\lceil \frac{45}{14} \right\rceil = 4 \leq 13$$

כלומר ניתן לבצע את האלגוריתם.
עלות החישוב של $Hash - Join$ היא:

$$3B(Movies) + 3B(PlaysIn) = 3 \cdot 45 + 3 \cdot 220 = 795$$

ג. התנאי לקיום אלגוריתם $Sort - Merge - Join$ הוא:

$$\left\lceil \frac{B(Movies)}{15} \right\rceil < 15 \quad \wedge \quad \left\lceil \frac{B(PlaysIn)}{15} \right\rceil < 15$$

ומתקיים:

$$\left\lceil \frac{B(Movies)}{15} \right\rceil = 3 < 15$$

אך:

$$\left\lceil \frac{B(PlaysIn)}{15} \right\rceil = 15 \not< 15$$

כלומר לא ניתן לממש אלגוריתם $Sort - Merge - Join$ על הטבלאות הבאות.
2.
א. כעת $M - 2 = 14$ לכן עלות החישוב החדשה תהיה:

$$45 + 220 \cdot \left\lceil \frac{45}{14} \right\rceil = 925 = 45 + 220 \cdot \left\lceil \frac{45}{13} \right\rceil$$

כלומר אינה משתנה כלל
ב. נוסחת עלות החישוב של HJ אינה כוללת התייחסות לM. ומכיוון שהאלגוריתם היה בר מימוש
כאשר $M = 15$ הוא גם יהיה בר מימוש כאשר $M = 16$. לכן אין שינוי.
ג. התנאי לקיום אלגוריתם $Sort - Merge - Join$ הוא:

$$\left\lceil \frac{B(Movies)}{16} \right\rceil < 16 \quad \wedge \quad \left\lceil \frac{B(PlaysIn)}{16} \right\rceil < 16$$

ואכן מתקיים:

$$\left\lceil \frac{B(Movies)}{16} \right\rceil = 3 < 16$$

$$\left\lceil \frac{B(PlaysIn)}{16} \right\rceil = 14 < 16$$

נבדוק אם מתקיים התנאי לאלגוריתם SMJ **יעיל**:

$$\left\lceil \frac{B(Movies)}{16} \right\rceil + \left\lceil \frac{B(PlaysIn)}{16} \right\rceil < 16$$

$$3 + 14 = 17$$

כלומר לא קיים אלגוריתם **יעיל**, לכן עלות האלגוריתם תהיה:

$$5B(Movies) + 5B(PlaysIn) = 5 \cdot 45 + 5 \cdot 220 = 1325$$

.3
א. בכל מצב, גודל החוצץ הנדרש ל־BNL הוא 3. בלוק אחד ליחס החיצוני, בלוק אחד ליחס הפנימי,
ובלוק אחד לפלט.

ב. עבור HJ נדרוש שיתקיים:

$$\left\lceil \frac{B(Movies)}{M-1} \right\rceil \leq M-2 \ \vee \ \left\lceil \frac{B(PlaysIn)}{M-1} \right\rceil \leq M-2$$

הטבלה $Movies$ כוללת פחות בלוקים, ולכן ניתן להתייחס רק אליה, מכיוון שאם $\left\lceil \frac{B(PlaysIn)}{M-1} \right\rceil \leq M-2$ מתקיים, אז גם $\left\lceil \frac{B(Movies)}{M-1} \right\rceil \leq M-2$ מתקיים:

$$\left\lceil \frac{B(Movies)}{M-1} \right\rceil = \left\lceil \frac{45}{M-1} \right\rceil \leq M-2$$

נפתור ונראה כי עבור $M = 9$ מתקיים:

$$\left\lceil \frac{45}{M-1} \right\rceil = 6 \leq 7 = M - 2$$

ג. עבור SMJ נדרוש שיתקיים:

$$\left\lceil \frac{B(Movies)}{M} \right\rceil < M \quad \wedge \quad \left\lceil \frac{B(PlaysIn)}{M} \right\rceil < M$$

הטבלה $PlaysIn$ כוללת יותר בלוקים, לכן ניתן להתייחס רק אליה.
כלומר נפתור רק את:

$$\left\lceil \frac{B(PlaysIn)}{M} \right\rceil < M$$

כבר בסעיף הקודם ראינו כי עבור $M = 15$ לא מתקיים התנאי, ועבור $M = 16$ מתקיים.
לכן $M = 16$.
ד. התנאי לקיום אלגוריתם SMJ יעיל הוא:

$$\left\lceil \frac{B(Movies)}{M} \right\rceil + \left\lceil \frac{B(PlaysIn)}{M} \right\rceil < M$$

ראינו בסעיף הקודם כי עבור $M = 16$ התנאי אינו מתקיים.
עבור $M = 17$:

$$\left\lceil \frac{B(Movies)}{17} \right\rceil + \left\lceil \frac{B(PlaysIn)}{17} \right\rceil = 13 + 3 = 16 < 17 = M$$

כלומר הגודל המינימלי הוא $M = 17$

שאלה 2

נתון:

$$B(R) = 300,\ B(S) = 1000$$

$$T(R) = 300 \cdot 100 = 30,000$$

$$T(S) = 1000 \cdot 50 = 50,000$$

$$V(R,B) = 100,\ V(S,C) = 200$$

$$M = 10$$

א. למדנו שהערכת מספר השורות המתאימות לביטוי זה היא:

$$\frac{T(S)}{V(S,C)} = \frac{50000}{200} = 250$$

ולכן מספר הבלוקים יהיה:

$$\left\lceil \frac{250}{50} \right\rceil = 5$$

כלומר נזדקק ל־5 בלוקים.
ב. במקרה זה אנו לא יודעים את הטווח של A , ולכן הכלל לחישוב מספר השורות המתאימות לביטוי הוא:

$$\frac{T(R)}{3} = \frac{30000}{3} = 10000$$

כלומר מספר הבלוקים הוא:

$$\left\lceil \frac{10000}{100} \right\rceil = 100$$

כלומר נזדקק ל־100 בלוקים
ג.
לצורך החישוב ניזכר כי B הוא מפתח ב־S, לכן:

$$V(S,B) = T(S) = 50000$$

בסך הכל מספר השורות המתאימות לביטוי $\sigma_{A<10 \wedge C=8}(R(A,B) \bowtie S(B,C))$ יהיה:

$$\frac{T(R) \cdot T(S)}{V(S,C) \cdot \max\{V(R,B), V(S,B)\} \cdot 3} =$$

$$\frac{50000 \cdot 30000}{200 \cdot 50000 \cdot 3} = 50$$

$$a_b$$

כלומר 50 שורות שונות.
ד. נחשב את עלות החישוב עבור כל אלגוריתם צירוף שלמדנו עליו, נבצע את הבחירה לפני הצירוף בכל אחד מהם ע"מ לצמצם עלויות:

קודם כל נגדיר את היחסים המסוננים:

$$R_A = \sigma_{A<10} R(A,B)$$

$$S_C = \sigma_{C=8} S(B,C)$$

עלות הביצוע של $\sigma_{C=8} S(B,C)$ היא או $B(S) = 1000$ או עלות הגישה באמצעות האינדקס.
מכיוון שהשימוש באינדקס עצמו זניח, עלות הגישה היא בסך הכל עלות שליפת הבלוק לכל שורה המתאימה לתנאי, כלומר $T(S_C) = 250$, לכן נשתמש באינדקס.
ע"מ ליצור את R_A נשתמש ב־$full\,table\,scan$ מכיוון שאין אינדקס על A , לכן עלות החישוב של היחס R_A היא $B(R) = 300$

כעת נחשב עבור כל אלגוריתם:
$\boldsymbol{BNL}$:

$$Read(S_C) + Read(R_A) \cdot \left\lceil \frac{B(S_C)}{M-2} \right\rceil$$

$$250 + 300 \cdot \left\lceil \frac{5}{8} \right\rceil = 550$$

כלומר העלות היא 550 .

$\boldsymbol{INL}$:

עלות החישוב של INL תלויה במספר השורות ביחס הפנימי (עליו יש אינדקס), עלות החישוב תהיה:

$$Read(R_A) + T(R_A) \cdot (cost\, of\, select) =$$

מזניחות הגישה לאינדקס:

$$300 + 10000 \cdot (1) = 10300$$

לכן עלות החישוב תהיה 10300

$\boldsymbol{HJ}$:

תחילה נבדוק אם מתקיים התנאי לשימוש באלגוריתם:

$$\left\lceil \frac{B(R_A)}{M-1} \right\rceil \leq M-2 \;\; \vee \;\; \left\lceil \frac{B(S_C)}{M-1} \right\rceil \leq M-2$$

$$\left\lceil \frac{100}{9} \right\rceil \leq 8 \;\; \vee \;\; \left\lceil \frac{5}{9} \right\rceil \leq 8$$

התנאי מתקיים ולכן ניתן להשתמש.

עלות החישוב של HJ היא :

$$Read(R) + Read(S) + 2 \cdot B(R_A) + 2 \cdot B(S_C) =$$

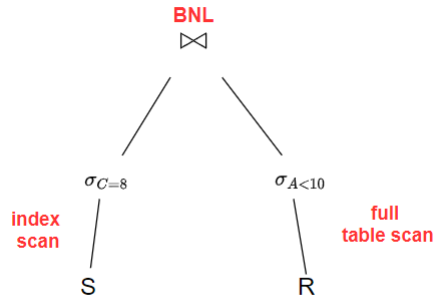
$$300 + 250 + 200 + 10 = 760$$

כלומר עלות החישוב היא 760.

$\boldsymbol{SMJ}$:

לא נחשב מכיוון שגם במקרה הטוב ביותר, עלות החישוב תהיה זהה לזאת של HJ, ולכן לא אופטימלית.

ראינו שהתכנית היעילה ביותר היא BNL. להל"ן תכנית השאילתה:



ה.
כמו שחישבנו בסעיף הקודם, עלות החישוב של תכנית השאילתה היא 550 פעולות IO

שאלה 3

נתון:

$$B(S) = 1200,\ \ B(R) = 4000$$

גודל בלוק הוא 2000 בתים
גודל שורה ב־S הוא 30 בתים
גודל שורה ב־R הוא 20 בתים
בלוק יחיד בS מכיל $\left\lfloor \frac{2000}{30} \right\rfloor$ שורות, כלומר 66 שורות. לכן:

$$T(S) = 1200 \cdot 66 = 79200$$

בלוק יחיד בR מכיל $\left\lfloor \frac{2000}{20} \right\rfloor$ שורות, כלומר 100 שורות. לכן:

$$T(R) = 4000 \cdot 100 = 400000$$

$$V(S, A) = 1000,\ V(R, A) = 400000$$

$$V(R, B) = 10$$

$$M = 70$$

א. מספר השורות בתוצאת הביטוי $\pi_{A,D}\sigma_{B=20\wedge D<5}(R(A,B) \bowtie S(A,C,D))$ יהיה:

$$\frac{T(S)\cdot T(R)}{\max\{V(R,A),V(S,A)\}\cdot V(R,B)\cdot 3} =$$

$$\frac{79200\cdot 400000}{400000\cdot 10\cdot 3} = 2640$$

כלומר יהיו 2640 שורות.
ב. כל שורה מכילה את האטריביוטים A, D . כל אחד מהם שוקל 10 בתים. כלומר גודל שורה בביטוי
יהיה 20 בתים.
לכן כל בלוק יכיל $\lfloor \frac{2000}{20} \rfloor = 100$ שורות, ולכן נידרש ל $\lceil \frac{2640}{100} \rceil = 27$ בלוקים.
כלומר גודל התוצאה יהיה 27 בלוקים.
ג.נחשב את עלות החישוב עבור כל אלגוריתם צירוף שלמדנו עליו, נבצע את הבחירה לפני הצירוף בכל אחד מהם ע"מ לצמצם עלויות, בנוסף, נבצע הטלות מוקדמות במידת הצורך:

קודם כל נגדיר את היחסים המסוננים לאחר הטלות:

$$R_B = \pi_A\sigma_{B=20}R(A,B)$$

$$S_D = \pi_{A,D}\sigma_{D<5}S(A,C,D)$$

נחשב את מספר הבלוקים עבור R_B, S_D:

$$B(R_B) = \frac{400000}{10\cdot\lceil\frac{2000}{10}\rceil} = 200$$

$$B(S_D) = \frac{79200}{3\cdot\lceil\frac{2000}{20}\rceil} = \frac{79200}{300} = 264$$

כעת נחשב עבור כל אלגוריתם:
$\boldsymbol{BNL}$:

$$Read(R_B) + Read(S_D) \cdot \left\lceil \frac{B(R_B)}{M-2} \right\rceil$$

$$4000 + 1200 \cdot \left\lceil \frac{200}{70-2} \right\rceil = 7600$$

כלומר העלות היא 7600 .
$\boldsymbol{INL}$:
לא קיים אינדקס ולכן לא נחשב.
$\boldsymbol{HJ}$:
תחילה נבדוק אם מתקיים התנאי לשימוש באלגוריתם:

$$\left\lceil \frac{B(R_B)}{M-1} \right\rceil \leq M-2 \quad \vee \quad \left\lceil \frac{B(S_D)}{M-1} \right\rceil \leq M-2$$

$$\left\lceil \frac{200}{69} \right\rceil \leq 68 \quad \vee \quad \left\lceil \frac{264}{69} \right\rceil \leq 68$$

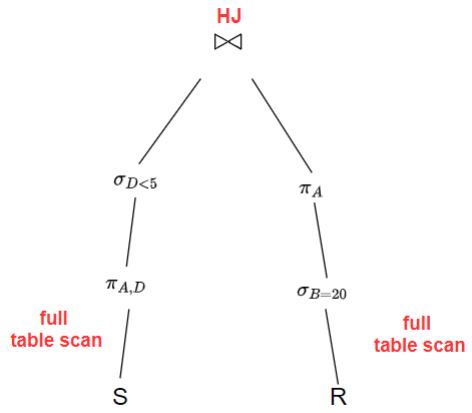
התנאי מתקיים ולכן ניתן להשתמש.
עלות החישוב של HJ היא :

$$Read(R) + Read(S) + 2 \cdot B(R_B) + 2 \cdot B(S_D) =$$

$$4000 + 1200 + 2 \cdot 200 + 2 \cdot 264 = 6128$$

כלומר עלות החישוב היא 6128.
$\boldsymbol{SMJ}$:
גם במידה ומתקיים התנאי לחישוב יעיל של SMJ, עלות החישוב שלו תהיה זהה לHJ , ולכן נעדיף להשתמש ב־HJ.

כלומר ראינו שהדרך האופטימלית לחישוב השאילתה היא HJ , להלן תכנית השאילתה:



ד.
כמו שחישבנו בסעיף ג' , עלות החישוב היא 6128.

שאלה 4

א. לשאילתה לקח יותר מ־2 דקות לרוץ.
תכנון השאילתה להלן:

```
                                    QUERY PLAN
-----------------------------------------------------------------------------------------
--
 Unique  (cost=54725977.96..54725982.33 rows=250 width=44)
   ->  Sort  (cost=54725977.96..54725978.58 rows=250 width=44)
         Sort Key: m1.movieid, m1.title, m1.rating, m1.year, m1.duration, m1.genre
         ->  Seq Scan on movies m1  (cost=0.00..54725968.00 rows=250 width=44)
               Filter: (duration = (SubPlan 1))
               SubPlan 1
                 ->  Aggregate  (cost=1094.49..1094.50 rows=1 width=4)
                       ->  Seq Scan on movies m2  (cost=0.00..1093.00 rows=595 width=4
)
                             Filter: (year = m1.year)
 JIT:
   Functions: 10
   Options: Inlining true, Optimization true, Expressions true, Deforming true
(12 rows)
```

ב.

```
                                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------
 Unique  (cost=2461.30..2464.36 rows=175 width=44) (actual time=150.179..150.508 rows=116 loops=1)
   ->  Sort  (cost=2461.30..2461.74 rows=175 width=44) (actual time=150.177..150.299 rows=116 loops=1)
         Sort Key: m2.year, m2.duration, m2.movieid, m2.title, m2.rating, m2.genre
         Sort Method: quicksort  Memory: 36kB
         ->  Hash Join  (cost=1224.25..2454.78 rows=175 width=44) (actual time=74.685..149.989 rows=116 loops=1)
               Hash Cond: ((m2.year = m3.year) AND (m2.duration = m3.duration))
               ->  Seq Scan on movies m2  (cost=0.00..968.00 rows=50000 width=44) (actual time=0.009..36.244 rows=50000 loops=1)
               ->  Hash  (cost=1222.99..1222.99 rows=84 width=8) (actual time=74.666..74.667 rows=88 loops=1)
                     Buckets: 1024  Batches: 1  Memory Usage: 12kB
                     ->  Subquery Scan on m3  (cost=1221.52..1222.99 rows=84 width=8) (actual time=74.258..74.562 rows=90 loops=1)
                           ->  Unique  (cost=1221.52..1222.15 rows=84 width=8) (actual time=74.255..74.444 rows=90 loops=1)
                                 ->  Sort  (cost=1221.52..1221.73 rows=84 width=8) (actual time=74.253..74.311 rows=90 loops=1)
                                       Sort Key: (min(m1.duration)), m1.year
                                       Sort Method: quicksort  Memory: 29kB
                                       ->  HashAggregate  (cost=1218.00..1218.84 rows=84 width=8) (actual time=74.110..74.173 rows=90 loops=1)
                                             Group Key: m1.year
                                             ->  Seq Scan on movies m1  (cost=0.00..968.00 rows=50000 width=8) (actual time=0.004..33.584 rows=50
 Planning Time: 0.192 ms
 Execution Time: 150.658 ms
(19 rows)
```

זמן הריצה כולל תכנון הוא כ150.9 מילישניות.
בשאילתה הראשונה ערכנו $full\,table\,scan$ לכל איבר ב־$M1$, כלומר למעשה עשינו מכפלה קרטזית, ורק אז סיננון.
בשאילתה השנייה, קודם כל סיננו, ורק אז התאמנו את השורות המאימות לאורך הסרט ולשנה.
ג. ניתן להשתמש באינדקסים הבאים:

$$(year)$$
$$(duration)$$
$$(year, duration)$$
$$(duration, year)$$

מתוכם, נראה שאלה שיועילו יותר יהיו

$$(year, duration), (duration, year)$$

מכיוון שהם מספקים גישה מהירה גם לשנה וגם למשך הסרט, המידע הדרוש לנו בעצם.
שימוש באינדקס $(year, duration)$ האיץ את ביצוע השאילתה בערך פי 2 באופן עקבי:

```
                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------
 Unique  (cost=1949.02..1952.09 rows=175 width=44) (actual time=72.318..72.561 rows=116 loops=1)
   ->  Sort  (cost=1949.02..1949.46 rows=175 width=44) (actual time=72.316..72.390 rows=116 loops=1)
         Sort Key: m2.year, m2.duration, m2.movieid, m2.title, m2.rating, m2.genre
         Sort Method: quicksort  Memory: 36kB
         ->  Nested Loop  (cost=1221.81..1942.50 rows=175 width=44) (actual time=71.386..72.209 rows=116 loops=1)
               ->  Unique  (cost=1221.52..1222.15 rows=84 width=8) (actual time=71.366..71.536 rows=90 loops=1)
                     ->  Sort  (cost=1221.52..1221.73 rows=84 width=8) (actual time=71.365..71.420 rows=90 loops=1)
                           Sort Key: (min(m1.duration)), m1.year
                           Sort Method: quicksort  Memory: 29kB
                           ->  HashAggregate  (cost=1218.00..1218.84 rows=84 width=8) (actual time=71.227..71.289 rows=90 loops
                                 Group Key: m1.year
                                 ->  Seq Scan on movies m1  (cost=0.00..968.00 rows=50000 width=8) (actual time=0.010..32.185 r
               ->  Index Scan using year_dur on movies m2  (cost=0.29..8.55 rows=2 width=44) (actual time=0.003..0.004 rows=1 l
                     Index Cond: ((year = m1.year) AND (duration = (min(m1.duration))))
 Planning Time: 0.184 ms
 Execution Time: 72.685 ms
(16 rows)
```

שימוש באינדקס $(duration, year)$ האיץ את ביצוע השאילתה באופן דומה:

```
                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------
 Unique  (cost=2052.17..2055.23 rows=175 width=44) (actual time=71.795..72.027 rows=116 loops=1)
   ->  Sort  (cost=2052.17..2052.61 rows=175 width=44) (actual time=71.794..71.865 rows=116 loops=1)
         Sort Key: m2.year, m2.duration, m2.movieid, m2.title, m2.rating, m2.genre
         Sort Method: quicksort  Memory: 36kB
         ->  Nested Loop  (cost=1221.81..2045.65 rows=175 width=44) (actual time=70.918..71.685 rows=116 loops=1)
               ->  Unique  (cost=1221.52..1222.15 rows=84 width=8) (actual time=70.898..71.069 rows=90 loops=1)
                     ->  Sort  (cost=1221.52..1221.73 rows=84 width=8) (actual time=70.897..70.952 rows=90 loops=1)
                           Sort Key: (min(m1.duration)), m1.year
                           Sort Method: quicksort  Memory: 29kB
                           ->  HashAggregate  (cost=1218.00..1218.84 rows=84 width=8) (actual time=70.759..70.820 rows=90 loops=
                                 Group Key: m1.year
                                 ->  Seq Scan on movies m1  (cost=0.00..968.00 rows=50000 width=8) (actual time=0.010..32.090 r
               ->  Index Scan using dur_year on movies m2  (cost=0.29..9.77 rows=2 width=44) (actual time=0.003..0.004 rows=1 l
                     Index Cond: ((duration = (min(m1.duration))) AND (year = m1.year))
 Planning Time: 0.182 ms
 Execution Time: 72.159 ms
(16 rows)
```

נציין גם כי שימוש באינדקסים $(year),(duration)$ האיץ רק במידה מועטת את ביצוע השאילתה (סדר גודל של כ־10 אחוזים לכל אחד מהם)

נוכל להסביר את ההפרש בזמני הריצה בצורה הבאה:
האינדקס $(year, duration)$ (שחשבנו שיהיה היעיל ביותר) מאפשר לנו לגשת בקלות לסרט שיצא בשנה נתונה, ומכיוון שהסידור המשני יהיה ע"פ $duration$, לכל שנה הסרט בעל האורך המינימלי, יהיה הראשון בכל קבוצה של שנה מסוימת. כלומר אם נחפש סרט שיצא בשנה נתונה, הסרט הראשון שניתקל בו, יהיה הסרט בעל האורך המינימלי.

האינדקס $(duration, year)$ מאפשר לנו לגשת בצורה מהירה לסרט בעל האורך המינימלי בשנה נתונה, מכיוון שאנו מסדרים את האינקס קודם כל לפי משך הסרט, נצפה למצוא את כל הסרטים בעלי האורך המינמלי בקיצון הקבוצה. ביצועיו היו טובים באופן מפתיע, חזינו שהוא יתפקד משמעותית פחות טוב מהאינדקס על $(year, duration)$.