Contents

1 Basic Test Results

```
Extracting Archive:
Archive:  /tmp/bodek.GbL7tu/db/ex4/reut.bruner/presubmission/submission
  inflating: ex4.pdf
 extracting: README

************************************************************
** Testing that all necessary files were submitted:
README:
    SUBMITTED
ex4.pdf:
    SUBMITTED

************************************************************
** Checking for correct README format:
```

2 README

```
hagai.y,reut.bruner
```

DB תרגיל 4

* Movies - כל שורה תופסת 36 בית ← 8192/36=227 שורות בבלוק
← 100000/227=45 בלוקים

* PlaysIn - כל שורה תופסת 18 בית ← 8192/18=455 שורות בבלוק
← 100,000/455 = 220 בלוקים

שאלה 1

1 (א) נשים את Maries בחסם התחתון: העלות תהיה:

$$B(Movies)+B(PlaysIn)\cdot\left\lceil\frac{B(Movies)}{15-2}\right\rceil=45+220\cdot\frac{45}{13}=925$$

(ב)

$$\left\lceil\frac{B(Movies)}{15-1}\right\rceil=\left\lceil\frac{45}{14}\right\rceil=4<15-1=14$$

← ניתן לבצע את האלגוריתם והעלות תהיה

$$3B(Movies)+3B(PlaysIn)=3\cdot45+3\cdot220=795$$

(ג)

$$\left\lceil\frac{B(PlaysIn)}{15}\right\rceil=15=M$$

← לא ניתן להפעיל את האלגוריתם

2 (א)

$$45+220\cdot\frac{45}{14}=925$$

← נשאר אותו דבר

(ב) כמובן שניתן לבצע גם כאן את האלגוריתם והעלות תישאר כפי- 795

(ג)

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

← לא ניתן לבצע את האלגוריתם היעיל.
ניתן לבצע את האלגוריתם הלא יעיל והעלות תהיה

$$5\cdot45+5\cdot220=1325$$

3 (א) 3

(ב) ניקח את היחס הקטן יותר - כלומר שיתקיים $\left\lceil \frac{45}{M-1} \right\rceil < M-1$

$$\frac{45}{M-1} < M-1 \Rightarrow 45 < M^2-2M+1 \Rightarrow M^2-2M-44 > 7.7$$

עבור $M=8$: $\left\lceil \frac{45}{7} \right\rceil = 7$ - התנאי לא מתקיים. עבור $M=9$: $\left\lceil \frac{45}{8} \right\rceil = 6$

- התנאי מתקיים. לכן הערך המינימלי הוא 9.

(ג) לפי החישוב בסעיף ב ניתן לבדוק שהגודל המינימלי הוא 16 - עבור גודל 15, אי אפשר לבלוק את הרלציות.

(ד)

$$\frac{45}{M} + \frac{220}{M} < M \Rightarrow 45+220 < M^2 \Rightarrow 265 < M^2 \Rightarrow 16.27 < M$$

עבור $M=17$

$$\left\lceil \frac{45}{17} \right\rceil + \left\lceil \frac{220}{17} \right\rceil = 3+13 = 16$$

- הגודל המינימלי הוא 17

שאלה 2

(א)

$$\frac{B(S)}{V(S,C)} = \frac{1000}{200} = 5$$

(ב) מכיוון שלטבלה ערכי A לא יגרום ניתוח שאלים מהשורות מקיימות את התנאי יתקבל $\frac{300}{3} = 100$

(ג) $\sigma_{C=8} S(B,C)$ - יש כ $5 \cdot 50 = 250$ שורות

$\sigma_{A<10} R(A,B)$ - יש כ $100 \cdot 100 = 10{,}000$ שורות

מספר השורות לאחר הצירוף הטבעי בין הטבלאות:

B מפתח זר ולכן $V(S,B)$ שווה למס' השורות ב-S

$$\frac{250 \cdot 10{,}000}{\max\{V(S,B), V(R,B)\}} = \frac{2{,}500{,}000}{\max\{1{,}000 \cdot 50, 100\}} = \frac{2{,}500{,}000}{50{,}000} = 50$$

(ד) עלות הקריאה של $\sigma_{C=8} S(B,C)$: לא אינדקס = קריאת כל הטבלה: 1,000

עם אינדקס = קריאת הבלוקים של השורות המתאימות - 250

=> יותר יעיל להשתמש באינדקס

BNL: כאשר R חיצוני

$$300 + 250 \cdot \left\lceil \frac{100}{10-2} \right\rceil = 3550$$

כאשר S חיצוני:

$$250 + 300 \cdot \left\lceil \frac{5}{10-2} \right\rceil = 5550$$

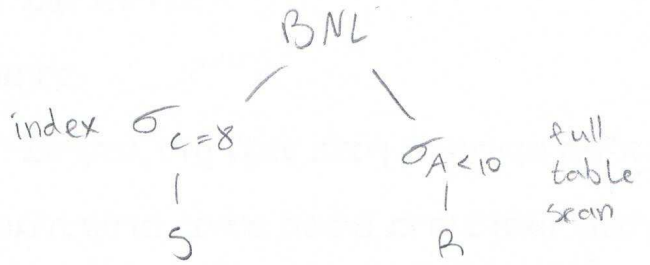
← כאשר S חיצוני קוראים את R פעם אחת ואז את S פעם אחת

ולכן זה יותר יעיל מ SM + HJ

INL:

$$300 + 10{,}000 \cdot 1 = 10300$$

⇐ האלגוריתם היעיל ביותר הוא BNL



ה) $$250 + 300 \cdot \left\lceil \frac{5}{10-2} \right\rceil = 550$$

שאלה 3

א) * $\sigma_{B=20} R(A,B)$ – $\frac{B(R)}{V(R,B)} = \frac{4000}{10} = 400$ בלוקים

כל שורה ב- R – 20 בלוקים, הבלוק נכנסות $\frac{2000}{20} = 100$ שורות של R

⇐ $\sigma_{B=20} R(A,B)$ – $400 \cdot 100 = 40{,}000$ שורות

* $\sigma_{D<5} S(A,C,D)$ – $\frac{1200}{3} = 400$ בלוקים

כל שורה ב- S – 30 בלוקים – הבלוק נכנסות $\frac{2000}{30} = 66$ שורות של S

⇐ $\sigma_{D<5} S(A,C,D)$ – $66 \cdot 400 = 26400$ שורות

⇐ לאחר הבירור מס' השורות יהיה:

$$\frac{26400 \cdot 40{,}000}{\max\{V(R,A), V(S,A)\}} = \frac{26400 \cdot 40{,}000}{\max\{400{,}000, 1000\}} = 2640$$

$V(R,A) = T(R) = 100 \cdot 4{,}000$

ב) בתוצאה יש 2 אטריבוטים - כל שורה 20 בית ⇐ בבלוק נכנסות 2000/20=100 שורות.

⇐ $\lceil 2640/100 \rceil = 27$ בלוקים

ג) נחשב את $B(E_R)$: $T(E_R)=40{,}000$, יש אטריבוט אחד (A) בכל בלוק יכנסו 2000/10=200 שורות ולכן $B(E_R)=40{,}000/200=200$

נחשב את $B(E_S)$: $T(E_S)=26{,}400$, יש שני אטריבוטים (A,D) בכל בלוק יכנסו 2000/20=100 שורות ולכן $B(E_S)=26400/100=264$

BNL: כאשר R חיצוני:

$$4{,}000+1200\cdot\frac{200}{70-2}=7{,}600$$

כאשר S חיצוני:

$$1200+4000\cdot\frac{264}{70-2}=17{,}200$$

SM

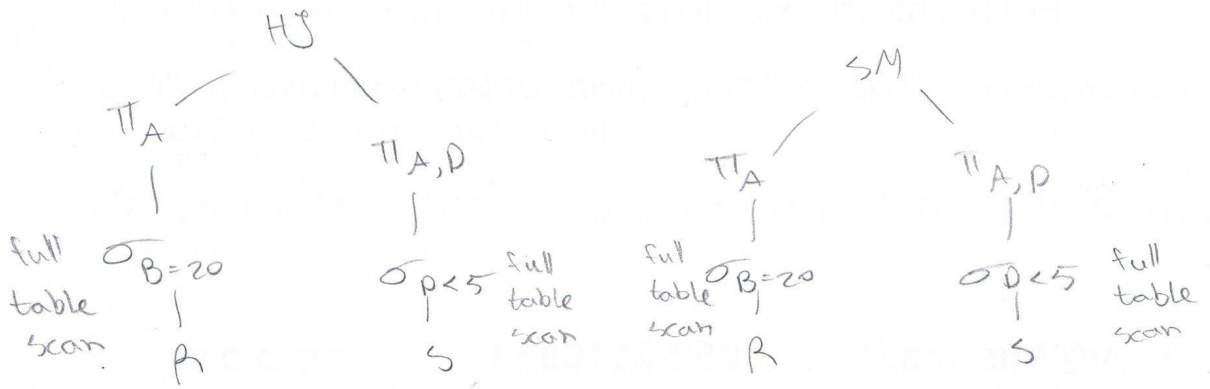
$$\left\lceil\frac{B(E_R)}{70}\right\rceil=\left\lceil\frac{200}{70}\right\rceil=3\ ,\ \left\lceil\frac{B(E_S)}{70}\right\rceil=\left\lceil\frac{264}{70}\right\rceil=4 \Rightarrow 3+4<70$$

=> ניתן לבצע את האלגוריתם היעיל והעלות תהיה:

$$4000+1200+2(200+264)=6128$$

HJ התנאים מתקיימים והעלות תהיה 6128 כמו ב-SM

⇐ האלגוריתמים היעילים הם SM ו- HJ



ג) 6128 כמו שחישבנו בסעיף הקודם.

שאלה 4

(א) יותר מ-2 דקות.

(ב) השאילתה:

```
select distinct *
from movies natural join (select year, min(duration) as duration
from movies
group by year) t
```

זמן הריצה: 142.178+0.181 = 142.359 ms

השיפור בזמן הריצה קרה כי בשאילתה החדשה מתשבים רק פעם אחת מה הסרט הקצר ביותר שהיה בכל שנה. לעומת זאת בשאילתה הקודמת, על כל שורה מתשבים את משך הסרט הכי קצר שהיה באותה שנה של הסרט הנוכחי.

(ג) יצרנו אינדקס על year ועל אינדקס על duration. האינדקס על duration הביא זמן ריצה טוב יותר - 3702.146+0.265 = 3702.411 ms

זמן הריצה השתפר כי מציאת המינימום של duration בשאילתה הפנימית יכלה להתבצע בעזרת.

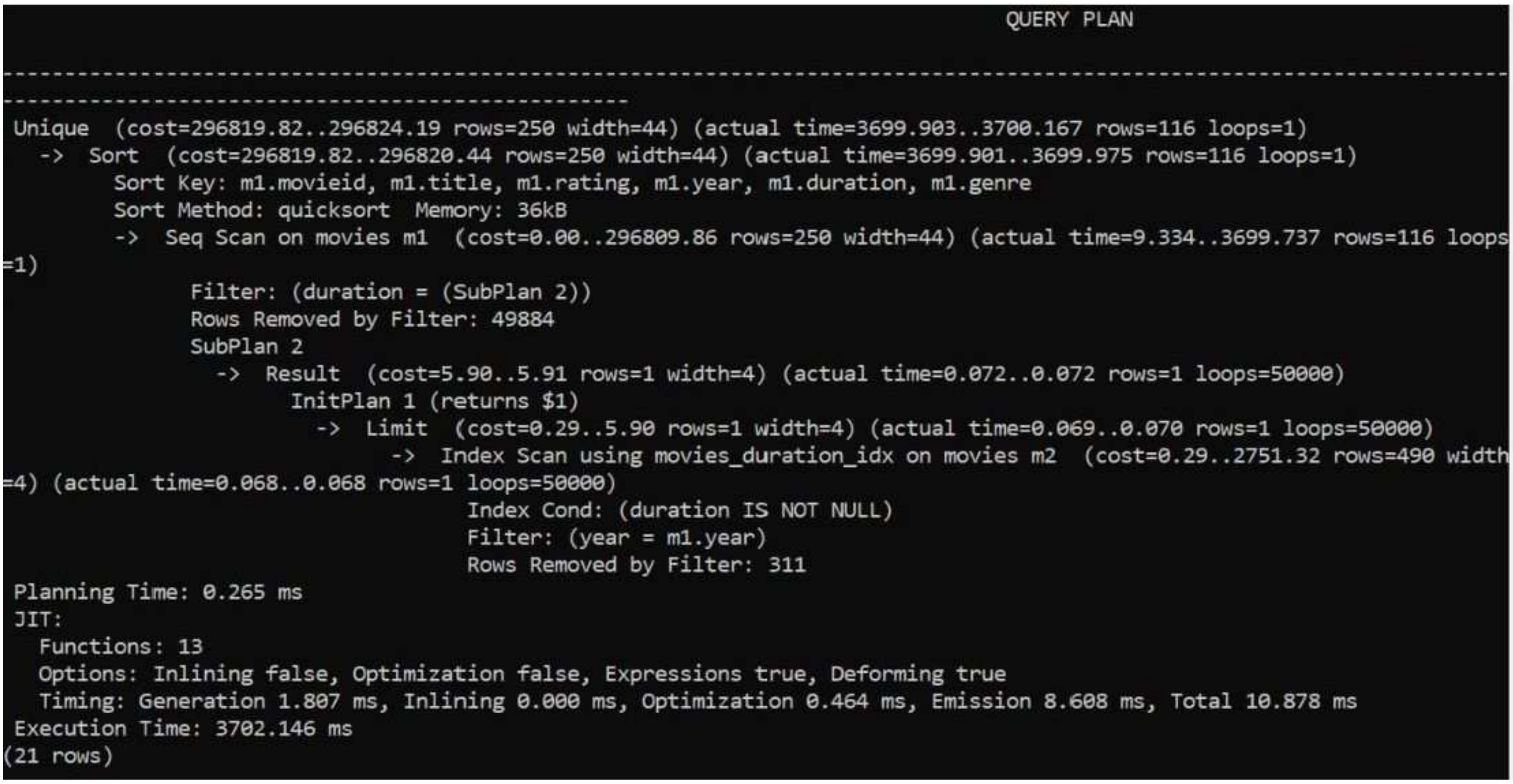
סעיף א'

```
 loops=1)
               Filter: (duration = (SubPlan 1))
               Rows Removed by Filter: 49884
               SubPlan 1
                 ->  Aggregate  (cost=1094.49..1094.50 rows=1 width=4) (actual time=4.207..4.208 rows=1 loops=50000)
                       ->  Seq Scan on movies m2  (cost=0.00..1093.00 rows=595 width=4) (actual time=1.156..3.658 rows=8
47 loops=50000)
                             Filter: (year = m1.year)
                             Rows Removed by Filter: 49153
 Planning Time: 0.630 ms
 JIT:
   Functions: 10
   Options: Inlining true, Optimization true, Expressions true, Deforming true
   Timing: Generation 1.458 ms, Inlining 70.328 ms, Optimization 87.297 ms, Emission 59.282 ms, Total 218.365 ms
 Execution Time: 210812.986 ms
(18 rows)

(END)    ->  Seq Scan on movies m2  (cost=0.00..1093.00 rows=595 width=4) (actual time=1.156..3.658 rows=847 loops=50000
)
                             Filter: (year = m1.year)
                             Rows Removed by Filter: 49153
 Planning Time: 0.630 ms
 JIT:
   Functions: 10
   Options: Inlining true, Optimization true, Expressions true, Deforming true
   Timing: Generation 1.458 ms, Inlining 70.328 ms, Optimization 87.297 ms, Emission 59.282 ms, Total 218.365 ms
 Execution Time: 210812.986 ms
(18 rows)
```

סעיף ב'

```
c         Sort Key: movies.year, movies.duration, movies.movieid, movies.title, movies.rating, movies.genre
          Sort Method: quicksort  Memory: 36kB
          ->  Hash Join  (cost=1220.94..2451.46 rows=174 width=44) (actual time=72.590..141.661 rows=116 loops=1)
                Hash Cond: ((movies.year = movies_1.year) AND (movies.duration = (min(movies_1.duration))))
                ->  Seq Scan on movies  (cost=0.00..968.00 rows=50000 width=44) (actual time=0.010..33.209 rows=50000 loo
ps=1)
                ->  Has                                                                    QUERY PLAN
)
-----------------------------------------------------------------------------------------------------------------------------
--------------------------
 Unique  (cost=2457.94..2460.98 rows=174 width=44) (actual time=141.797..142.042 rows=116 loops=1)
   ->  Sort  (cost=2457.94..2458.37 rows=174 width=44) (actual time=141.795..141.875 rows=116 loops=1)
          Sort Key: movies.year, movies.duration, movies.movieid, movies.title, movies.rating, movies.genre
          Sort Method: quicksort  Memory: 36kB
          ->  Hash Join  (cost=1220.94..2451.46 rows=174 width=44) (actual time=72.590..141.661 rows=116 loops=1)
                Hash Cond: ((movies.year = movies_1.year) AND (movies.duration = (min(movies_1.duration))))
                ->  Seq Scan on movies  (cost=0.00..968.00 rows=50000 width=44) (actual time=0.010..33.209 rows=50000 loo
ps=1)
                ->  Hash  (cost=1219.68..1219.68 rows=84 width=8) (actual time=72.564..72.565 rows=88 loops=1)
                      Buckets: 1024  Batches: 1  Memory Usage: 12kB
                      ->  HashAggregate  (cost=1218.00..1218.84 rows=84 width=8) (actual time=72.424..72.487 rows=90 loop
s=1)
                            Group Key: movies_1.year
                            ->  Seq Scan on movies movies_1  (cost=0.00..968.00 rows=50000 width=8) (actual time=0.004..3
2.744 rows=50000 loops=1)
 Planning Time: 0.181 ms
 Execution Time: 142.178 ms
(14 rows)

(END)
```

סעיף ג'



```
                                                                                QUERY PLAN

------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------
 Unique  (cost=296819.82..296824.19 rows=250 width=44) (actual time=3699.903..3700.167 rows=116 loops=1)
   ->  Sort  (cost=296819.82..296820.44 rows=250 width=44) (actual time=3699.901..3699.975 rows=116 loops=1)
         Sort Key: m1.movieid, m1.title, m1.rating, m1.year, m1.duration, m1.genre
         Sort Method: quicksort  Memory: 36kB
         ->  Seq Scan on movies m1  (cost=0.00..296809.86 rows=250 width=44) (actual time=9.334..3699.737 rows=116 loops
=1)
               Filter: (duration = (SubPlan 2))
               Rows Removed by Filter: 49884
               SubPlan 2
                 ->  Result  (cost=5.90..5.91 rows=1 width=4) (actual time=0.072..0.072 rows=1 loops=50000)
                       InitPlan 1 (returns $1)
                         ->  Limit  (cost=0.29..5.90 rows=1 width=4) (actual time=0.069..0.070 rows=1 loops=50000)
                               ->  Index Scan using movies_duration_idx on movies m2  (cost=0.29..2751.32 rows=490 width
=4) (actual time=0.068..0.068 rows=1 loops=50000)
                                     Index Cond: (duration IS NOT NULL)
                                     Filter: (year = m1.year)
                                     Rows Removed by Filter: 311
 Planning Time: 0.265 ms
 JIT:
   Functions: 13
   Options: Inlining false, Optimization false, Expressions true, Deforming true
   Timing: Generation 1.807 ms, Inlining 0.000 ms, Optimization 0.464 ms, Emission 8.608 ms, Total 10.878 ms
 Execution Time: 3702.146 ms
(21 rows)
```