Bloom Filters

מוטיבציה: מבני נתונים גדולים

- הזכרון מוגבל בגודלו (ו"מתאפס" בכיבוי המחשב).
- מבני נתונים גדולים נשמרים בהתקני אחסון דוגמת **דיסק**.
- נתמקד ב**מילון** (ללא מחיקות) עם מפתחות גדולים (מחרוזות).
  - למשל, בהרצאה 5 ראינו כיצד עצי B+ מתאימים לתרחיש זה.
  - לדוגמה, אם המפתח הוא כתובת אתר אינטרנט, (נניח 20 תוים), ובמילון יש 500 מיליון אתרים, סך הזכרון הנחוץ לשמירת המפתחות בלבד (ללא כיווץ) הוא 10G.

- נרצה לשמור על **מינימום גישות לדיסק** (יקרות! $$$).
- לא ניתן להמנע מגישות לדיסק בעת הוספת איבר.
- ננסה לחסוך גישות לדיסק בפעולת החיפוש, ובפרט בחיפוש כושל.

Approximate Membership Query (AMQ)

נגדיר מבנה נתונים חדש:

- יצירת מבנה נתונים ריק (Create) - הקבוצה S ריקה.
- הוספת איבר לקבוצה - Add(S,x).
- שאילתת שייכות IsMember(S,x) - האם x שייך לקבוצה S?

נחזיק AMQ שיסומן ב F בזכרון **בנוסף** למילון השמור בדיסק.

- בעת הוספת איבר x למילון, נוסיף את x ל- F.
- לפני חיפוש x במילון, נבדוק האם x נמצא ב- F. אם לא, **נעצור**!

סיבוכיות מקום ודיוק התשובות

מימוש AMQ בעזרת מילון בעייתי - המפתחות גדולים מכדי לשמור בזכרון;

אך ללא שמירת המפתחות לא נוכל להבטיח תשובה נכונה לכל שאילתא...

גישה חלופית:

נתפשר על דיוק התשובות לשאילתות בתמורה לחיסכון במקום.

בפרט, נרשה למבנה הנתונים לטעות במקרה של חיפוש כושל (ורק במקרה כזה!).

סוגי שגיאות

תשובה "x שייך לקבוצה S" כאשר x אינו שייך לקבוצה נקראת False Positive.

תשובה "x **אינו** שייך לקבוצה S" כאשר x שייך לקבוצה נקראת False Negative.

אנו נרשה רק False Positives. מדוע?

- לאחר False Positive מבוצע חיפוש של x במילון (בדיסק), ו**השגיאה תתגלה**.
- לאחר False Negative לא ניגשים למילון (דיסק) ולכן **השגיאה לא תתגלה!**

מימוש פשוט ל AMQ מבוסס טבלת ערבול

- נחזיק טבלת ערבול עם פונקציית ערבול h, אך ללא שמירת המפתחות בטבלה.
- במקום מפתחות, בכל תא יהיה ביט בודד, מאופס באתחול.
- בהכנסת איבר x לתא h(x) נדליק את הביט.
- בחיפוש איבר x ניגש לתא h(x) ונחזיר "שייך" אם ורק אם הביט דלוק.

דוגמה: נניח שגודל טבלת הערבול 5, ונבחרה פונקצית ערבול mod 5.

לאחר שנכניס את האיברים 7, 13, 23, 42, 11 הטבלה תראה כך:

0	1	1	1	0
0	1	2	3	4

אם כעת נחפש את 30, התשובה שתתקבל תהיה "אינו שייך"; אם נחפש 23 התשובה תהיה "שייך". אך נקבל תשובה "שייך" גם אם נחפש את 17...

מימוש פשוט ל AMQ מבוסס טבלת ערבול

נכונות: אם איבר הוכנס למבנה, מובטח שנחזיר "שייך" (כלומר אין False Negatives).

סיבוכיות זמן: O(1) **במקרה הגרוע**.

סיבוכיות מקום: m **ביטים** (m גודל הטבלה).

במימוש כזה, התנגשויות בפונקציית הערבול גורמות לשגיאות מסוג False Positive.

שימו לב: התנגשויות בין איברים שכבר הוכנסו אינן מהוות בעייה!

שגיאה מתרחשת בשאילתא לאיבר **שלא הוכנס** וממופה לאותו תא כמו **איבר שקיים במבנה**.

מדוע מימוש זה אינו תומך במחיקות?

ניתוח הסיכוי לשגיאת False Positive

נתבונן באיבר x **שלא** הוכנס למבנה. מה ההסתברות להחזיר "שייך" (False Positive)?

- זו ההסתברות שהביט h(x) דלוק.
- נסמן ב- n את מספר האיברים שהוכנסו למבנה עד כה.
- נניח שפונקצית הערבול h מקיימת את **הנחת הפיזור האחיד הפשוט**.
- לכל איבר שהוכנס, ההסתברות שלא התנגש עם x היא:

$$(1 - 1/m)$$

ולכן ההסתברות שאף איבר לא התנגש עם x (והביט כבוי) היא:

$$(1 - 1/m)^n$$

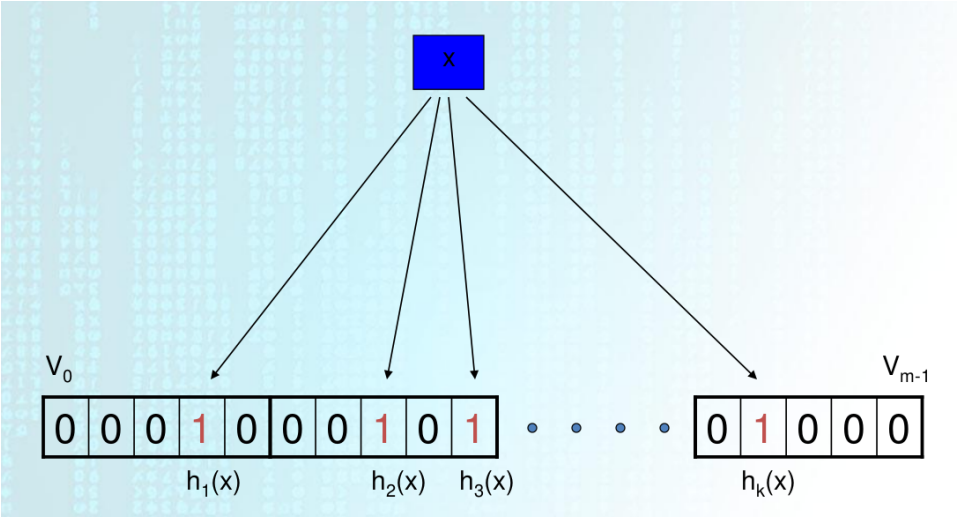
ומכאן שההסתברות לשגיאה (הביט דלוק) היא:

$$1 - (1 - 1/m)^n \sim 1 - e^{(-n/m)}$$

Bloom Filters

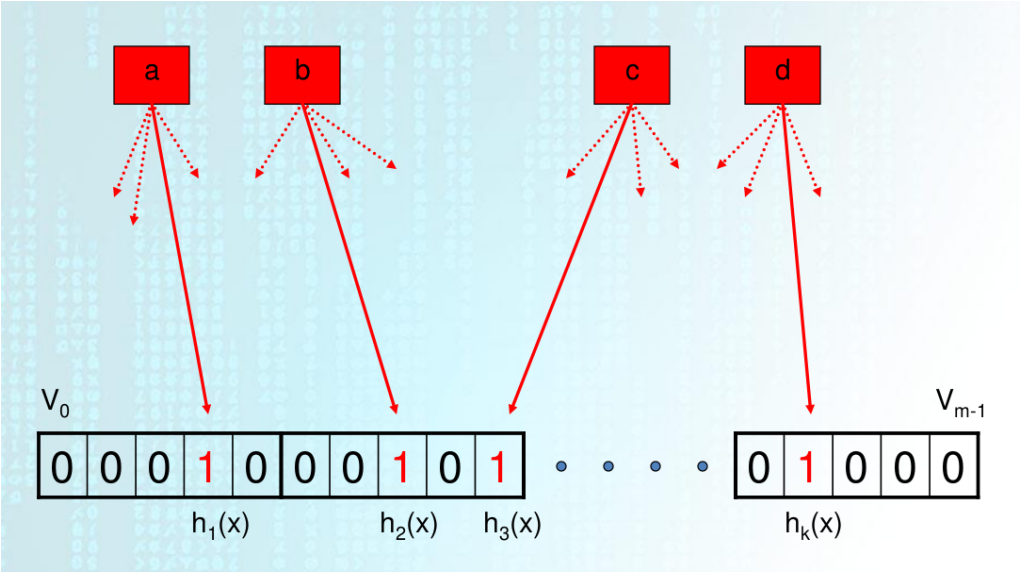
מימוש נוסף של AMQ נקרא Bloom Filter ומהווה הרחבה למימוש הפשוט.

- נשתמש ב k פונקציות ערבול: $h_1, h_2, \ldots h_k$.
- בעת הוספת איבר x נדליק את k הביטים המתאימים: $h_i(x)$.
- בחיפוש איבר x ניגש ל k התאים $h_i(x)$ ונחזיר "שייך" אם ורק אם **כל k הביטים דלוקים**.



שימו לב: המימוש הראשון הוא מקרה פרטי בו k=1.

False Positives in Bloom Filters



דוגמה לשגיאה:
המפתח x לא
הוכנס למבנה
הנתונים אך
השאילתא
תחזיר "שייך".

באופן דומה למקרה שבו k=1, מתקבל כי הסיכוי לשגיאה הוא בערך:

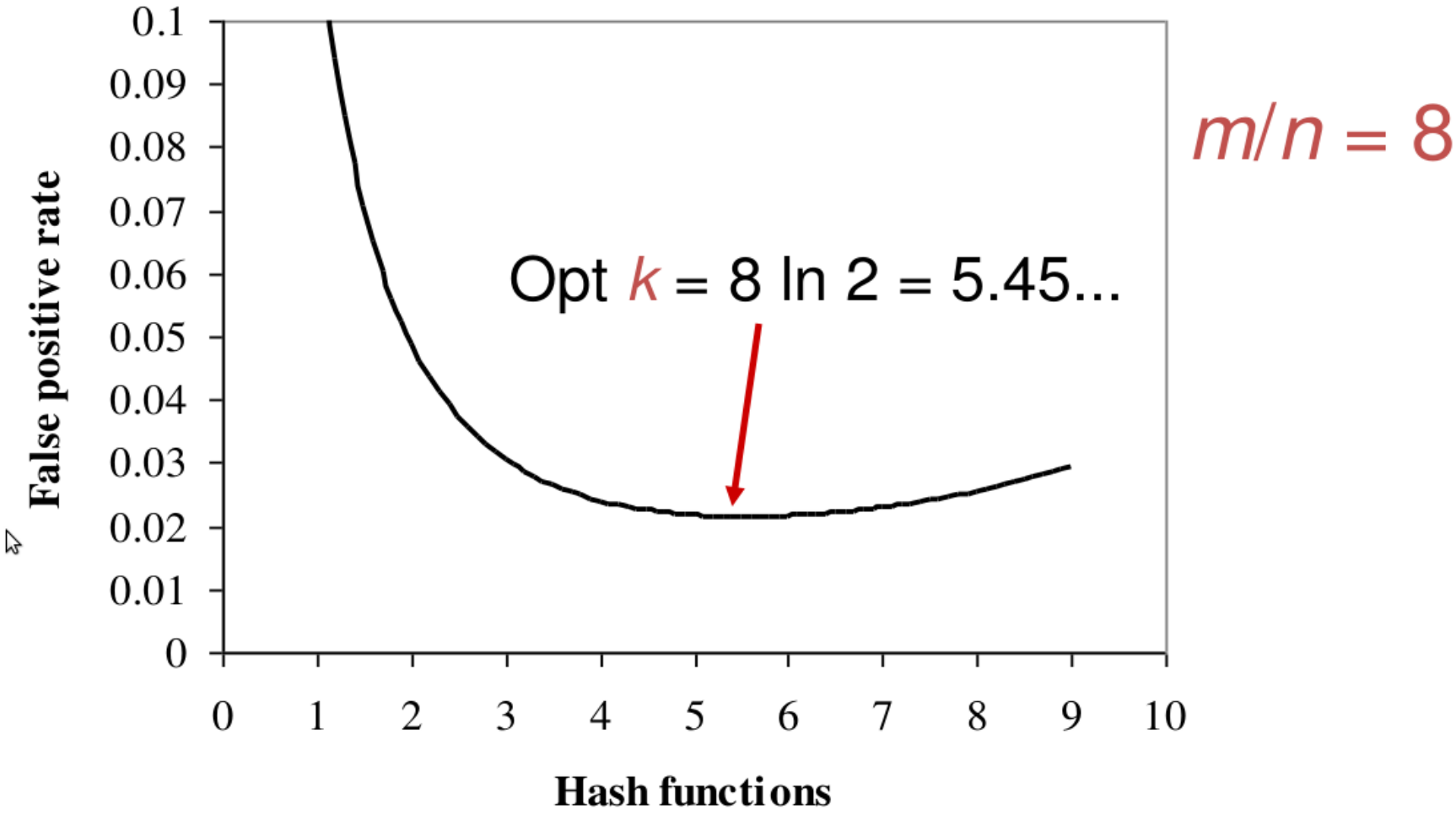
$$(1 - e^{(-kn/m)})^k$$

מכאן ניתן לחשב את הערך האופטימלי של k בהנתן m ו- n, ומתקבל:

$$k_{min} = \ln(2)\, m/n$$

הסיכוי לשגיאה עבור k זה, בערך כ- $0.6185^{m/n}$.

סיכוי שגיאה כפונקציה של k כש- m/n קבוע



שימושים והרחבות

- על ידי הקצאת ~8 ביטים למפתח בממוצע (היחס בין n לבין m), ובחירת k = 5, קיבלנו ~3% שגיאה בלבד.
- במצב האופטימלי כחצי מהביטים דלוקים.
- עבור הדוגמה משקף המוטיבציה, המקום הדרוש יהיה 0.5GB בזכרון.

- ניתן להרחיב את המימוש לתמיכה במחיקות על ידי שמירת מונה בכל כניסה בטבלת הערבול במקום ביט בודד (במחיקת x נחסיר 1 מהמונים בכניסות המתאימות).

- Bloom Filters נמצאים בשימוש בתעשייה (למשל, ב- BigTable של Google).
- הומצאו ב 1970 על ידי Bloom, וחזרו לקבל תשומת לב במגוון מחקרים בעשור האחרון שעסקו בהרחבות נוספות ושימושים שונים.