אלגוריתמים ומנגנונים

האלגוריתמים שנלמד בקורס הזה יהיו שונים מהאלגוריתמים שלמדתם בקורסים קודמים.

עד עכשיו, למדתם על אלגוריתמים שמקבלים קלט ומוציאים פלט. אבל מאיפה מגיע הקלט הזה?

במקרים רבים, הקלט הזה מגיע מבני-אדם. הנה כמה דוגמאות.

1. אנחנו רוצים להעביר הודעה ברשת מחשבים, במסלול המהיר ביותר. לכאורה אפשר להשתמש באלגוריתם מוכר למציאת מסלול עם משקל קל ביותר - אלגוריתם דייקסטרה (תיכנות דינמי). אבל מה קורה אם כל מחשב ברשת שייך לאדם אחר - מי אמר שהאנשים האלה בכלל רוצים שנעבור דרכם? אם הם לא רוצים שנעבור דרכם, הם יכולים לדווח מהירות גבוהה מהמהירות האמיתית כך שהאלגוריתם לא יבחר בהם. איך נגרום להם להגיד את האמת?

2. הממשלה רוצה לתת חפץ מסויים (למשל, זכות שידור בתדר מסויים) למי שהחפץ הזה שווה עבורו הכי הרבה. אפשר פשוט לשאול כל אחד כמה החפץ שווה עבורו ולהשתמש באלגוריתם לחישוב מקסימום, אבל מי אמר שאנשים יגידו את האמת? הרי כל מי שרוצה לקבל את החפץ יכול להגיד שהוא שווה עבורו המון...

3. יש לנו גרף דו-צדדי - גרף שהצמתים בו מתחלקים לשתי קבוצות והקשתות הן רק בין קבוצה אחת לשניה. אנחנו רוצים למצוא "שידוך" בגרף הזה - התאמה בין צמתים מקבוצה אחת לצמתים מהקבוצה השניה. קיימים אלגוריתמים המוצאים שידוכים מקסימליים בגרפים דו-צדדיים.אבל עכשיו נניח שהצמתים בגרפים מייצגים בני-אדם. לדוגמה, צד אחד בגרף הוא סטודנטים המעוניינים להתקבל לאוניברסיטאות מסויימות, והצד השני בגרף הוא האוניברסיטאות, והקשתות מציינות רצון - איזה סטודנט רוצה להתקבל לאיזו אוניברסיטה, ואיזו אוניברסיטה רוצה לקבל איזה סטודנט. האם אנחנו בטוחים שהצדדים המעורבים יגלו לנו את ההעדפות האמיתיות שלהם?

במהלך הקורס נלמד על אלגוריתמים שהקלט שלהם הוא העדפות ורצונות של בני-אדם. בספרות המקצועית, אלגוריתם כזה נקרא גם **מכניזם** mechanism או **מנגנון** (בהתאם לאות מ בשם הקורס...).

שיבוצים

האלגוריתמים הראשונים שנלמד יהיו אלגוריתמים של שיבוץ. נתחיל מדוגמה המוכרת לרבים מכם - שיבוץ סטודנטים למעונות. אני לא יודע איך בדיוק זה עובד באריאל. אבל באוניברסיטאות אחרות בעולם זה עובד בערך כך.

הקלט למנגנון הוא - כאמור - ההעדפות של הסטודנטים. לשם כך כל סטודנט ממלא רשימה ובה שלושה סוגים של מעונות: עדיפות ראשונה, שניה ושלישית. בנוסף, משרד המעונות מגדיר תור בין סטודנטים לפי קריטריוני זכאות.

האלגוריתם:
1. עוברים על הסטודנטים לפי התור.
2. נותנים לכל סטודנט את העדיפות הכי גבוהה הפנויה.
3. אם כל העדיפויות תפוסות – נותנים לסטודנט חדר באקראי.

האם המנגנון הזה טוב? ומה זה בכלל מנגנון "טוב"?

אנחנו נגדיר שתי תכונות של מנגנונים ונראה אם אלגוריתם השיבוץ שלנו מקיים אותן.

תכונה א: כנות

הגדרה: אלגוריתם נקרא **כֵּנָה** (באנגלית **truthful**, מילים נרדפות: **strategyproof, incentive compatible**) אם כל משתתף משיג את התוצאה הטובה ביותר עבורו כאשר הוא מדווח את ההעדפות האמיתיות שלו, וזאת בלי תלות בפעולות של המשתתפים האחרים.

למה זו תכונה טובה? מכמה סיבות:

- היא מקלה על המשתתפים. זה מספיק קשה לחשוב איזה חדר אתם רוצים - חדר נוח ויקר או חדר פחות נוח וזול, בקרוואן או בבניין, רחוק או קרוב וכו'... אם המנגנון לא כנה, אתם צריכים להוסיף לזה גם שיקולים אסטרטגיים - באיזה מקום בתור תהיו, ומה יבחרו המשתתפים האחרים. אם המנגנון כנה, אתם לא צריכים לעשות שיקולים אסטרטגיים אלא רק לדווח מה שאתם באמת חושבים.
- היא מונעת חרטה וצער. במנגנון לא כנה אתם עלולים להצטער אם טעיתם בשיקולים האסטרטגיים שלכם; במנגנון כנה אין מה להצטער.

האם האלגוריתם "שיבוץ שלוש עדיפויות" שהוצג למעלה כנה? התשובה היא לא!
הוכחה: ניקח לדוגמה סטודנט ששלושת העדיפויות הראשונות שלו הן 101, 102, 103. נניח לצורך הדוגמה שבכל בניין יש 100 חדרים, והסטודנט נמצא במקום 301 בתור. אם הסטודנט יודע ש-300 הראשונים בתור רוצים את הבניינים האלה, כדאי לו לומר שהוא רוצה בעדיפות ראשונה את העדיפות הרביעית שלו – זה טוב יותר מלקבל חדר באקראי. מש"ל.

האם בכלל קיים אלגוריתם שיבוץ כנה? התשובה היא כן! מאד קל למצוא אלגוריתם כזה: התעלם מהקלט ושבץ את הסטודנטים באקראי (אם לא רוצים **לגמרי** להתעלם מהקלט, יש פתרון אחר: בוחרים סטודנט אחד באקראי ונותנים לו את העדיפות הראשונה שלו, ואת כל השאר משבצים באקראי).

אם האלגוריתם הזה נראה לכם מוזר, אתם צודקים, הוא באמת מוזר, אבל מה **בדיוק** הבעיה בו? כדי להגדיר את הבעיה נגדיר תכונה נוספת.

תכונה ב: יעילות פארטו

כנות היא תכונה של מנגנון; **יעילות** היא תכונה של תוצאה של מנגנון (במקרה שלנו: שיבוץ).
התכונה שנגדיר הוגדרה פורמלית ע"י וילפרדו פארטו (Pareto) - כלכלן איטלקי שחי לפני כ-100 שנה.
אולם הרעיון קיים כבר בתלמוד בעיקרון "זה נהנה וזה לא חסר". הנה ההגדרות:

- תוצאה א נקראת **שיפור פארטו** (Pareto improvement) של תוצאה ב, אם תוצאה א טובה יותר לחלק מהמשתתפים ("זה נהנה"), וטובה לפחות באותה מידה לכל השאר ("וזה לא חסר").
- תוצאה נקראת **יעילה פארטו** (Pareto efficient / Pareto optimal) אם לא קיימת תוצאה אחרת שהיא שיפור פארטו שלה.
- מנגנון נקרא **יעיל פארטו** אם כל תוצאה שלו היא יעילה פארטו.

למה זו תכונה טובה? כי אם התוצאה לא יעילה פארטו, זה אומר שהמנגנון לא עשה את המקסימום שהוא יכול כדי להשביע את רצונם של המשתתפים. בכל שיבוץ תמיד יהיו אנשים מרוצים יותר ופחות, אבל אם אפשר לשפר את מצבם של חלק מהמשתתפים בלי לפגוע באחרים - ודאי ראוי לעשות זאת.

האם המנגנון האקראי שהצגנו קודם הוא יעיל פארטו? ודאי שלא. הרי ייתכן שיהיו שני סטודנטים שכל אחד מהם יקבל את העדיפות הראשונה של השני שהיא העדיפות האחרונה שלו.
האם מנגנון "שיבוץ שלוש עדיפויות" יעיל פארטו? גם כאן התשובה לא. הוכחה: ניקח לדוגמה שני סטודנטים, א ו-ב, שכל שלושת העדיפויות שלהם נתפסו כשהגיע תורם. כל אחד מהם משובץ באקראי.

ייתכן ש-א ישובץ לעדיפות הרביעית של ב שהיא העדיפות האחרונה שלו, ולהיפך. תוצאה זו אינה יעילה פארטו. מש"ל.

האם קיים אלגוריתם שיבוץ יעיל פארטו? התשובה היא כן! ניקח את האלגוריתם הקודם ונבצע בו שינוי אחד קטן:
כל סטודנט יסמן את **כל** רשימת העדיפויות שלו, ולא רק שלוש עדיפויות גבוהות ביותר.
האלגוריתם המתקבל נקרא **דיקטטורה סדרתית** (serial dictatorship). מדוע? כי יש **סדרה** של סטודנטים, וכל אחד מהם בתורו הוא דיקטטור - הוא בוחר את השיבוץ הטוב ביותר עבורו מהחדרים שנשארו.
אנחנו נוכיח שהאלגוריתם החדש לא רק יעיל פארטו אלא גם כנה.

משפט: אלגוריתם "דיקטטורה סדרתית" הוא כנה.
הוכחה: נניח שמקומך בתור הוא k. עד שמגיע תורך, k-1 חדרים כבר תפוסים, וקבוצת החדרים הנשארים לא תלויה בדיווח שלך.
המנגנון בוחר עבורך, מתוך החדרים הנשארים, את החדר הטוב ביותר עבורך – בהתאם לרשימה שדיווחת. לכן, התוצאה הטובה ביותר עבורך תתקבל ע"י דיווח הרשימה האמיתית. מש"ל.

משפט: אלגוריתם "דיקטטורה סדרתית" הוא יעיל פארטו.
הוכחה: כיוון שהאלגוריתם כנה, ניתן להניח שכל הסטודנטים מדווחים את העדיפויות האמיתיות.
בהינתן קלט מסויים, נגדיר:
שיבוץ א – השיבוץ של המנגנון.
שיבוץ ב – שיבוץ אחר כלשהי.
נניח בשלילה ששיבוץ ב הוא שיפור פארטו של שיבוץ א.
נניח שהסטודנט הראשון הנהנה מהשיפור הוא הסטודנט שמקומו בתור הוא k.
כל הסטודנטים שמקומם 1 עד k-1 לא חסרים, כלומר – קיבלו את אותו סוג חדר בשני השיבוצים.
מכאן: כשמגיע תורו של הסטודנט ה-k, בשני השיבוצים, אוסף החדרים הפנויים הוא זהה.
אבל המנגנון בוחר עבור סטודנט k את החדר הטוב ביותר עבורו מבין החדרים הפנויים – סתירה. מש"ל.

מה המסקנה מכל הדיון הזה? ששינוי קטן באלגוריתם יכול ליצור שינוי גדול באיכות-החיים של המשתתפים. כשמתכננים אלגוריתמים שהקלט שלהם הוא רצונות של אנשים, צריך לשים לב לתכונות האלו. המסר הזה יחזור פעמים רבות לאורך הקורס.

מקורות

- הקורס של טים: http://theory.stanford.edu/~tim/f16/f16.html הרצאה 1.
- אתר המעונות של אוניברסיטת אריאל.

סיכם: אראל סגל-הלוי.