מטלה 9 - פתרון

יש לענות על שאלה אחת לבחירתכם. הגשה בזוגות, עד תחילת ההרצאה הבאה.

שאלה 1: שידוך מקסימלי וכיסוי מינימלי

בהרצאה הזכרנו את המשפט האומר שבגרף דו-צדדי, גודל השידוך המקסימלי שווה לגודל הכיסוי המינימלי. המשפט הזה נקרא **משפט קניג.** ההוכחה שלו נותנת גם אלגוריתם שבעזרתו ניתן לבנות, מכל שידוך מקסימלי, את הכיסוי המינימלי. קיראו את ההוכחה כאן:

https://en.wikipedia.org/wiki/K%C5%91nig%27s_theorem_(graph_theory)#Proof

תארו במילים שלכם את האלגוריתם ותנו דוגמת הרצה.

- **פתרון**: היו כמה פתרונות נכונים. ראו למשל בפתרון של יינון וסמואל.
- שימו לב - יש לוודא שבדוגמת ההרצה שלכם, מספר הצמתים בכיסוי המינימלי זהה למספר הקשתות בשידוך המקסימלי (זה משפט קניג).

שאלה 2: חלוקת שכר-דירה אגליטרית

בכל השמה של דיירים לחדרים, ישנן דרכים רבות לקבוע את מחירי החדרים. בכל וקטור מחירים, התועלת של כל דייר היא ערך החדר שקיבל עבורו פחות מחיר החדר. בכל וקטור מחירים, יש דייר אחד (או יותר) שהתועלת שלו הכי קטנה. החלוקה ה**אגליטרית** (egalitarian) היא החלוקה שבה התועלת הקטנה ביותר היא גדולה יותר מכל שאר החלוקות. החלוקה **האגליטרית ללא קנאה** היא החלוקה שבה התועלת הקטנה ביותר היא גדולה יותר מכל שאר החלוקות ללא קנאה.

א. תנו דוגמה עם שלושה חדרים ושלושה דיירים, שבה בחלוקה האגליטרית יש קנאה.

ב. מיצאו את החלוקה האגליטרית-ללא-קנאה בדוגמה זו.

ג. האם מצב כזה (שבו בחלוקה האגליטרית יש קנאה) אפשרי גם עם שני חדרים ושני דיירים?

- **פתרון**: הפתרון של אריה וניסן.

שאלה 3: חישוב חלוקה הוגנת עם תשלום מקסימלי

נתונה טבלה המתארת ערכים של שלושה דיירים לשלושה חדרים:

	חדר א	**חדר ב**	**חדר ג**
דייר א	**33**	22	11
דייר ב	66	**55**	44
דייר ג	99	88	**77**

ההשמה שנבחרה מסומנת בהדגשה. כיתבו פקודה בשפת Mathematica (או בשפה אחרת לבחירתכם) המחשבת מחיר לכל חדר, כך ש:

א. ההשמה הנבחרת, עם וקטור המחירים שחושב, היא ללא קנאה.

ב. סכום הכסף שנאסף מהדיירים הוא מקסימלי מבין כל החלוקות ללא קנאה.

- **פתרון**:

```
FindMaximum[{p1 + p2 + p3,    (* מיקסום סכום המחירים *)
  33 - p1 >= 22 - p2, 33 - p1 >= 11 - p3,    (* א לא מקנא *)
  55 - p2 >= 66 - p1, 55 - p2 >= 44 - p3,    (* ב לא מקנא *)
  77 - p3 >= 88 - p2, 77 - p3 >= 99 - p1,    (* ג לא מקנא *)
  33 - p1 >= 0, 55 - p2 >= 0, 77 - p3 >= 0}, (* השתתפות מרצון *)
 {{p1, 0}, {p2, 0}, {p3, 0}}]
```

שאלה 4: בעיית מכפלת הצריחים

נתון לוח שחמט (8 על 8) שבו בכל משבצת כתוב מספר חיובי כלשהו.

תארו אלגוריתם יעיל (פולינומיאלי) להצבת שמונה צריחים על הלוח, כך שאף צריח לא מאיים על אף צריח אחר, ומכפלת המספרים הרשומים תחת הצריחים היא הגדולה ביותר.

- **פתרון**: כל סידור של צריחים שאינם מאיימים זה על זה הוא למעשה שידוך בין טורים לשורות. ניצור גרף דו-צדדי שבו בצד אחד הטורים, בצד שני השורות, ומשקל הקשת בין טור לשורה הוא **מינוס הלוגריתם** של המספר שכתוב בחיתוך הטור והשורה. נפעיל את האלגוריתם ההונגרי. מצאנו שידוך שבו מינוס סכום הלוגריתמים הוא מינימלי --> סכום הלוגריתמים הוא מקסימלי --> לוגריתם המכפלה הוא מקסימלי --> המכפלה מקסימלית.
- **שימו לב**: כל הפותרים פספסו את הלוגריתם כי הניחו שמכפלה מקסימלית שקולה לסכום מקסימלי. ההנחה הזאת אינה נכונה - בדקו ותראו...

שאלה 5: תיכנות אלגוריתם

רוצים לחלק n חדרים ל-n דיירים. כל דייר מיוצג ע"י המחלקה הבאה:

```
class Agent {
     int bestRoom(int[] prices);
     // INPUT:   the prices of the n rooms, in shekels.
     // OUTPUT:  the index of a room that the agent most prefers
in these prices. Index is between 0 and n-1.
};
```

כיתבו בשפה לבחירתכם, או בפסאודו-קוד, אלגוריתם המקבל כקלט n שחקנים ואת מחיר הדירה הכולל בשקלים, ומוצא השמת-חדרים ללא קנאה עד כדי שקל אחד. אם זה מקל עליכם - אפשר להניח ש n=3.

כותרת הפונקציה:

```
void findEnvyFreeAssignment(Agent[] agents, int totalRent)
```

פלט לדוגמה:

```
Agent 0 receives room 2 for 163 shekels.
Agent 1 receives room 1 for 274 shekels.
Agent 2 receives room 0 for 343 shekels.
```

פתרון: נשים לב שאין לנו נתונים על ערך של כל חדר עבור כל דייר ("העדפות קרדינליות") - אלא רק איזה חדר כל דייר מעדיף בכל וקטור מחירים ("העדפות אורדינליות"). האלגוריתם היחיד שאנחנו מכירים למציאת חלוקה ללא קנאה במצב זה הוא אלגוריתם הסימפלקסונים של סו. לא נתון בשאלה שמתקיימת "הנחת הדיירים העניים", ולכן חלוקה ללא קנאה לא בהכרח קיימת. אבל אפשר לנסות למצוא אחת כזאת.

כיוון שאין דרישה לגבי יעילות, נחפש חלוקה ללא קנאה ע"י בדיקת כל האפשרויות. מצורף פסאודו-קוד לפתרון הבעיה. הקוד מניח שיש לנו פונקציה *permutations(n)* המאפשרת לעבור את כל הפרמוטציות של המספרים בין 0 ל-*n-1*.

```
// returns true if the assignment of agents[i] to room i in
prices[i] is envy-free.
bool isEnvyFree(Agent[] agents, int[] prices) {
     int n = agents.length;
     for (int i=0; i<n; ++i) {
         if (agents[i].bestRoom(prices)!=i)
     }
     }
}

 // Try to find an envy-free assignment in the given prices.
 map<int,Agent> findEnvyFreeAssignment
                (Agent[] agents, int[] prices) {
     int n = agents.length;
```

```
    map<int,Agent> mapRoomToAgentWhoWantsIt;
    for (int i=0; i<n; ++i) {
        int bestRoomOfI = agents[i].bestRoom(prices);
         int currentAgentWhoWantsThisRoom =
            mapRoomToAgentWhoWantsIt[bestRoomOfI];
         if (currentAgentWhoWantsThisRoom == null) {
            mapRoomToAgentWhoWantsIt[bestRoomOfI] = agents[i];
         } else {
            return null;
              // two agents want the same room - no EF assignment
         }
    }
}

void findEnvyFreeAssignment(Agent[] agents, int totalRent) {
    // Loop over all price-vectors in whole shekels:
    for (int p1=0; p1<=totalRent; ++p1) {
       for (int p2=0; p2<=totalRent-p1; ++p2) {
          int p3 = totalRent-p1-p2;
          int[] prices = {p1, p2, p3};
           map<int,Agent> envyFreeAssignment =
               findEnvyFreeAssignment(agents, prices);
           if (envyFreeAssignment) {
               cout <<  envyFreeAssignment << endl;
               return;
           }
       }
    }
    cout <<  "There is no envy-free assignment" << endl;
```

}