**שאלה 1**

הוכח/הפרך עבור :

הטענה לא נכונה. לפי הנלמד בהרצאה f1(n)=Ɵ(1) אם ורק אם f1(n)=O(1) וגם f1(n)=Ω (1). לפיכך, הכיוון הוא נכון. נציג דוגמא נגדית לכיוון , ובכך נפריך את הטענה:

יהא . נראה שמתקיים , כלומר לכל c > 0 קיים n0>0 כך שיתקיים לכל n≥n0 ש- f­1(n)≤c. כדי להוכיח זאת, נשתמש בהגדרה השקולה להגדרת o:

מכיוון ש- היא פונקציה ידועה, וידוע כי , הטענה מתקיימת. לכן f1(n)=o(1), ולכן בפרט f1(n)=O(1) (כי ). בנוסף, לפי הגדרת o, לכל c>0 קיים n0>0 כך שלכל n>n0 מתקיים f1(n)<c. לכן, **לא קיים** c>0 עבורו קיים n0>0 כך שלכל n>n0 מתקיים f1(n)>c, לכן f1(n)≠Ω(1), ולכן f1(n)≠Ɵ(1).



הטענה נכונה. נוכיח זאת:

***הסבר המעברים:***

1. *לפי הגדרת* Ω*.*
2. *חילוק ב- שאפשרי כי* f1 *פונקציה חיובית לפי הנתונים.*
3. *העלאה בריבוע, שמכיוון שהיא פונקציה אי שלילית, ובתחום שלנו חיובית ממש ומונוטונית עולה, אינה משנה את סימן אי השוויון.*
4. *חלוקה ב-*c2 *(מספר חיובי) לא משנה את אי השוויון.*
5. *הצבה - .*
6. n0 *נתון מהגדרת Ω, ו-*d *שהצבנו קודם התלוי ב-*c *הנתונה מהגדרת Ω.*
7. *לפי הגדרת* O*.*
8. *אם וגם אז*

*הטענה לא נכונה. נוכיח באמצעות דוגמא נגדית:*

*נגדיר את הפונקציות הבאות: .*

: *יהא* c=1, n0=2, *לכל* n≥n0 *מתקיים: .*

*הוכחה: היא פונקציה מונוטונית עולה לכל ולכל (מאינפי 1מ'). לכן לכל 1≤x<y מתקיים .*

1. *אם אז*

*הטענה לא נכונה. נוכיח באמצעות דוגמא נגדית:*

*מתקיים:*

*אך לא מתקיים:*

*נניח כי קיימים* c, n0 >0 *כך ש:*

*קיבלנו סתירה עבור:*

1. *אם וגם אז*

*הטענה נכונה. נוכיח זאת:*

*מהנתונים נסיק כי:*

*כעת נסתכל על המנה:*

*לכן:*

*קיבלנו:*

*באופן דומה:*

*ולכן*

*נסמן:*

*מ-(\*) ומ-(\*\*) נקבל כי:*

1. *אם וגם*

*אז*

*הטענה נכונה. נוכיח זאת:*

*לפי הנתון, קיימים כך שלכל מתקיים כי:*

*כמו כן, קיימים כך שלכל מתקיים כי:*

*נסתכל כעת על:*

*לכן קיימים כך שלכל מתקיים כי:*

*לכן:*

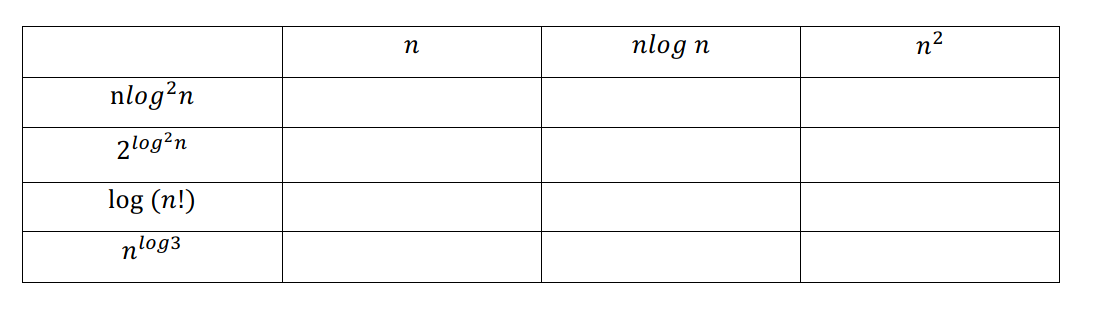
**שאלה 2**

1. לכל זוג פונקציות 𝑓(𝑛), 𝑔(𝑛) בטבלה למטה, כתבו את הסימנים: Ω, 𝜃, 𝑂

בהתאם לפי היחס: 𝑓(𝑛)= 𝑂(𝑔(𝑛)) , 𝑓(𝑛) = Θ(𝑔(𝑛)) , 𝑓(𝑛) = Ω(𝑔(𝑛))

אם יש יותר מיחס מתאים, תבחרו את היחס הכי חזק בין ו-.

הוכיחו את תשובתכם.



הסבר ⊛: ידוע כי log(n) מונוטונית עולה, וידוע כי log2(2)=1, וכן n≥n0.

***הסבר המעברים:***

1. *לפי הגדרת אומגה*
2. *חילוק ב-*n>0 *לא משנה את סימן אי השוויון.*
3. *חילוק ב-*log(n)>0 *(כי* n>2*) לא משנה את סימן אי השוויון.*
4. *ממונוטוניות* 2x*.*

*לפיכך נובע כי עבור* n0=2 *ועבור* c=1 *מתקיים התנאי, כי* 2c *היא פונקציה קבועה, וספציפית עבור ה-*c *שבחרנו מתקיים כי לכל* n≥2=n0 *יתקיים* n≥2c*. לפיכך, מכיוון שמתקיים , הטענה נכונה.*

*נבחר , ונקבל כי לכל הטענה מתקיימת (⊛).*

***הסבר המעברים:***

1. *לפי הגדרת* O*.*
2. *חילוק ב-*n>0 *לא משנה את סימן אי השוויון.*
3. *הוצאת שורש לא משנה את סימן אי השוויון ממונוטוניות (ושני האגפים חיוביים).*
4. *לפי הגדרת* O*, עבור* c, n0 *של .*

*⊛ (טענה מהתרגול), לכן בפרט עבור הטענה מתקיימת. לפיכך, מכיוון שמתקיים , הטענה נכונה.*

*נשים לב כי היא הרכבת פונקציות מונוטוניות עולות, ולכן פונקציה עולה. לפיכך, ומכיוון שמתקיים , אזי לכל* n>n0 *אי השוויון מתקיים.*

*נציב ונקבל כי מה שעלינו להוכיח הוא :*

*נשים לב כי עבור* logn-1 > 0 *יתקיים אי השוויון השני לפי טענה מההרצאה*:

*לכן, בפרט, לכל קיים* c>0 *עבורו קיים* n0>0 *כך שלכל* n>n0 *מתקיים , ולכן לפי הגדרה . עבור , אי השוויון ⊛ מתקיים, ומקיום יחס האמ"מ, נקבל כי הטענה נכונה.*

***הסבר המעברים:***

1. *לפי הגדרת Ω.*
2. *חילוק ב-*n>0 *לא משנה את סימן אי השוויון.*
3. *לפי ההגדרה השניה של* o.
4. *משפט ידוע מאינפי 2.*
5. *לפי ההגדרה השניה של .*
6. *לפי הגדרת .*

***הסבר המעברים:***

1. *לפי הגדרת אומגה.*
2. *הוכחנו כבר כי* nlog2n=O(n2)*, כלומר* **\****קיים* c1>0 *עבורו קיים* n0>0 *כך שלכל* n>n0 *יתקיים* nlog2n ≤ c1·n2*, ומכיוון שכך, בפרט יתקיים עבור* :n≥1log2n ≤ nlog2n ≤ c1·n2.

*לכן, ממונוטוניות יתקיים כי מכיוון ש- , מתקיים .*

*בנוסף, מכיוון שלכל מתקיים , עבור c=c1 אי השוויון מתקיים תמיד, ולכן מצאנו* c>0 *עבורו , ומכיוון שמתקיים*:

*, הטענה נכונה.*

*נבחר* c=1, n0=2*, ואז לכל* n≥2=n0 *מתקיים כי* (n-1)! ≥ (n0-1)! = (2-1)! = 1! = 1 = c*.*

: *הוּכַח בתרגול*

***הסבר המעברים:***

*⊛ הוכחנו זאת – זו הייתה הטענה הקודמת.*

1. *לפי הגדרת Ɵ, , ולכן נובע המעבר לפי הגדרת* O*.*
2. *לכל מתקיים*
3. *לפי הגדרת* O*.*

*נשים לב כי מתקיים , וכיוון ש-*nx *פונקציה מונוטונית עולה, מתקיים כי*

*. לפיכך עקב הביטוי הטריוויאלי , בפרט , כלומר קיים* c>0 *עבורו קיים* n0>0 *כך שלכל* n>n0 *יתקיים* n>c·n*. לכן, עבור אותם* c,n0 *יתקיים* nlog3>n>c·n0*.*

*לכן לפי הגדרה מתקיים .*

*נשים לב כי* log(3)-1>0*, לכן ניתן לסמן , ואז, כפי שכבר הוכחנו מסעיפים קודמים, מכיוון שמתקיים , אי השוויון* ***\**** *מתקיים, ולכן, עקב קיום , הטענה מתקיימת.*

***הסבר המעברים:***

1. *לפי הגדרת Ω.*
2. *חלוקה ב-*n>0 *לא משנה את סימן השוויון.*

*נשים לב כי מתקיים , ומכיוון ש-*nx *פונקציה מונוטונית עולה****\*****, ומתקיים כי*

*, מתקיים אי השוויון . לכן נבחר* c=1, n0=1*, ועבורם מתקיים*

***הסבר המעברים:***

1. *לפי הגדרת* O*.*
2. *חלוקה ב-*n2>0 *לא משנה את סימן השוויון.*

*⊛* xa *מונוטונית יורדת לכל* x>0 *עבור* a<0.

**המשך שאלה 2**

1. הראו כי לכל מתקיים .

***פתרון:***

*נסמן* log(n)=t *ונציב במה שנתבקשנו להוכיח: .*

*נשים לב שטענה זו הוכחה בתרגול, ולכן מש"ל.*

1. נתונה פונקציה כך ש-. הראו כי .

***פתרון:***

***הסבר המעברים:***

1. *לפי הגדרת* o*.*
2. 2x *מונוטונית עולה, ולכן אי השוויון נשמר.*
3. *חוקי חזקות.*
4. *.*

*עתה, יהא* c1>0*. נבחר . מכיוון שאי-השוויון מתקיים לכל* c > 0 *ולכל* n ≥ n0 > 0*, נציב* c=logn(c1·n)*, שכן מתקיים* c1·n1>1*,* *ולכן* logn(c1·n)>0.

*לפיכך מתקיים:*

*נציב , ולפיכך ( ):*

**שאלה 3**

1. נתונה משוואת הנסיגה הבאה כאשר a הוא קבוע ו-T(n)=1 לכל n<10. האם קיים a קבוע עבורו (𝑇(𝑛) = 𝜔(𝑛? אם כן מהו ה-a המינימאלי עבורו זה מתקיים ומה ערכה האסימפטוטי של (T(n במקרה זה?

*נפתור באמצעות עץ רקורסיות, נסמן:*

*ונקבל:*

*עץ רקורסיות:*

*נשים לב כי הסכום בשורה ה-k הוא מהצורה הבאה:*

*לכן סה"כ הסיבוכיות (סכום השורות):*

*לכל נקבל סכום של קבוע כלשהו כפול* n*, מצב זה אינו מתאים לנו כי אנחנו מחפשים* a *עבורו נקבל .*

*נדרוש על מנת שהסכום 'יתפוצץ' ולכן נקבל*

*במקרה שבו*

*נחסום את בצורה אסימפטוטית ע"י:*

1. מצאו חסם עליון וחסם תחתון אסימפטוטיים פשוטים (כלומר, ללא שימוש בסכימה, סימן עצרת וכו') עבור 𝑇(𝑛) בכל אחת מנוסחאות הנסיגה שלהלן. מצאו חסמים הדוקים ככל שתוכלו והוכיחו תשובתכם.

*נסמן:*

*נבדוק האם מתקיים:*

*מתקיים עבור:*

*נבדוק האם מתקיים:*

*נציב:*

*חישוב עזר:*

*לכן ניתן להזניח את החלק של* log *בחיפוש אחר* c *מתאים:*

*מתקיים עבור* c *כלשהו קטן מ-1*

*ולכן:*

*עבור:*

*נקבל:*

*מצד שני:*

*עבור:*

*נקבל:*

*בסה"כ:*

*נסמן:*

*מתקיים:*

*למשל עבור*

*מתקיים:*

*לכן:*

*נסמן:*

*נקבל:*

*נסמן:*

*נקבל:*

*נסמן:*

*נקבל:*

*נפתור לפי שיטת המאסטר כאשר:*

*מתקיים כי:*

*כמו כן מתקיים:*

*לכן מתקיים:*

*ולכן לפי שיטת המאסטר נקבל:*

**שאלה 4**

בעקבות הקורונה, מדינות רבות בעולם המיוצגות ע"י האינדקסים 1 עד 𝑁 רוצים לאמץ שיטת האקורדיון עם השינוי הבא: בשיטת האקורדיון המשק יפתח וייסגר בהתאם לקצב התחלואה במדינה, אבל, במטרה לצמצם את אי הודאות, במיוחד בתקופה זו, כל מדינה החליטה שהיא נכנסת לסגר כל מספר קבוע של ימים, כלומר במדינה 𝑖 נכנסים לסגר כל 𝐷i ימים אלא אם מקדימים את הסגר בעקבות מצב חירום (ע״י פעולת ()𝑃𝑟𝑒𝑐𝑒𝑑𝑒\_𝑙𝑜𝑐𝑘𝑑𝑜𝑤𝑛).

בתחילת כל יום, אנו מבצעים הדפסה של כל המדינות שיש בהן סגר (ע״י פעולת ()𝑙𝑜𝑐𝑘\_𝑑𝑜𝑤𝑛\_𝑛𝑜𝑤 בדיוק פעם אחת ליום, כלומר פעולה זו מסמנת את סוף היום הקודם)

נסמן

הציעו מבנה נתונים למימוש הפעולות הבאות:

* 𝐼𝑛𝑖𝑡(𝑁, 𝐷) - אתחול מבנה הנתונים עם הפרמטרים 𝑁 ו- 𝐷.

**סיבוכיות זמן: (𝟏)𝑶.**

* 𝐼𝑛𝑠𝑒𝑟𝑡(𝑖, 𝐷𝑖, 𝑑𝑎𝑦𝑠) - הוספת המדינה 𝑖 למבנה שהיא עוד לא במבנה, עם קבוע סגר 𝐷𝑖 כך שנותרו 𝑑𝑎𝑦𝑠 ימים לסגר הבא מיום ביצוע פעולה זו.

**סיבוכיות זמן: (𝟏)𝑶.**

* 𝑃𝑟𝑒𝑐𝑒𝑑𝑒\_𝑙𝑜𝑐𝑘𝑑𝑜𝑤𝑛(𝑖, 𝑑𝑎𝑦𝑠) – הקדמת הסגר במדינה 𝑖 ב – 𝑑𝑎𝑦𝑠 ימים בעקבות מצב חירום.

**סיבוכיות זמן: (𝟏)𝑶.**

* ()𝑙𝑜𝑐𝑘\_𝑑𝑜𝑤𝑛\_𝑛𝑜𝑤 – הדפסת האינדקסים של כל המדינות שבהן יש סגר ביום הפעלת הפונקציה.

**סיבוכיות זמן: (𝒌) 𝑶כאשר 𝒌 זהו מספר המדינות המודפסות.**

סיבוכיות המקום הנדרשת עבור המבנה הינה 𝑂(𝑁 + 𝐷).

*הערה: D ו- N אינם קבועים לצורכי ניתוח סיבוכיות.*

*בבואנו לפתור את התרגיל, נשים לב כי סיבוכיות הזמן של אתחול המבנה מוכרח להיות O(1). לפיכך, לפי הנלמד בהרצאה, נשתמש במבנה נתונים מסוג* ***מערך רב מימדי****.*

*ראשית, נגדיר את המערכים בהם נשתמש, אשר יהוו את המבנה שלנו:*

***Countries*** *– מערך ראשי בגודל* N *בו התא ה-*i *יחזיק מצביע ל-*node *של המדינה ה-*i.

***B*** *– מערך "ברוך" – מערך בגודל* N *הנועד לוודא האם כל ערך במערך* C *הוא "ערך זבל".*

***C*** *– מערך "אינדקסים מאותחלים" באורך* N*.*

***remainingDays*** *– מערך בגודל* D *בו התא ה-*i *מכיל רשימה מקושרת דו כיוונית של מדינות שייכנסו לסגר ביום ה-*i(modD) *מיום יצירת המבנה. במבנה שלנו נחזיק מצביע למערך זה בשם* lockdown\_today*, שיסמן באיזה יום אנו נמצאים מבחינת סגר (יתקדם* +1 *ע"י הקריאה לפונקציה lock\_down\_now()).*

*נציג את מבנה ה-*nodes *בהם נשתמש:*

int **country\_id** - (i)

int **between\_lockdowns** - (Di)

int **starting\_days** - (days)

int **list\_index**

country\_node\* **previous** - (i-1)’th country node pointer

country\_node\* **next** – (i+1)’th country\_node pointer

typedef struct country\_node:

int **index**

country\_node\* **next**

typedef struct head\_node:

*המקום הנדרש למימוש המבנה, אם כן, הוא בגודל* 3N+D *עבור המערכים, ועוד* Ncountry\_nodes*, ועוד* Dhead\_nodes*. סה"כ:* 3N+D+N+D=4N+2D=O(N+D)*.* *לכן המבנה שלנו עומד בסיבוכיות המקום הנדרשת.*

*כעת נציג את אופן מימוש הפונקציות:*

***Init:*** *ניצור 3 מערכים בגודל* N *ומערך אחד בגודל* D*. נאתחל את 3 המערכים הראשונים ע"י שימוש בשיטה שנלמדה בהרצאה, כאשר המערכים* Countries, B, C *תואמים בדיוק למערכים* A, B, C *מההרצאה. בנוסף, ניצור מצביע בשם* lockdown\_today *לתא 0 של המערך* remainingDays*. לא ניצור את ה-*head\_nodes *(זה יקרה ב-*Insert *בפעם הראשונה שנידרש לכך.*

סיבוכיות זמן: *מכיוון שהקצאת מערך ויצירת מצביע הן פעולות שסיבוכיות הזמן שלהן היא מסדר גודל* O(1)*, סה"כ סיבוכיות הזמן של* Init *היא* 5·O(1)=O(1)*, כנדרש.*

***Insert:*** *ניצור* country\_node *עם הפרטים הרלוונטיים עבור השדות* country\_id, between\_lockdowns, starting days *ונקרא לו מעתה "החוליה החדשה". נעדכן את ערך* list\_index*, להיות* (lockdown\_today + starting\_days)(modD)*, וזהו יהיה ערך האינדקס הרלוונטי במערך* remainingDays*.*

*במערך* A *נשמור מצביע לחוליה החדשה, ונעדכן את המערכים* B *ו-*C *ממש כפי שנלמד בהרצאה: נכניס את* i *בתא ה-*top*, נכניס את הערך הרלוונטי בתא המתאים ב-*B *ונקדם את* top*.*

*במערך* remainingDays*, בתא עם אינדקס* list\_index*, נכניס את החוליה החדשה שיצרנו לרשימה המקושרת שבתא ע"י קישור של ה-*next *של החוליה ה-*i *לחוליה הראשונה הנוכחית אחרי ה-*head *ברשימה הרלוונטית, וקישור ה-*previous *של החוליה החדשה שיצרנו ל-*head. *אם התא הרלוונטי במערך* remainingDays *עוד ריק, ניצור* head\_node *מהתא הרלוונטי וה*-next *שלו יצביע לחוליה החדשה. ה-*head\_node *שיצרנו יהיה ה-*previous *של החוליה שיצרנו.*

סיבוכיות זמן: *מכיוון שיצירה של כל אחת מהחוליות (*country\_node/head\_node*) היא פעולה שמתבצעת ב-*O(1) *(כי בכל אחד מהם יש מס' קבוע של פעולות השמה ללא הקצאה בגודל משתנה), סיבוכיות הזמן של* Insert *היא גם* O(1)*, כנדרש.*

***Precede\_Lockdown:***

*ניגש לחוליה של המדינה ה-*i *(הכתובת שלה שמורה בתא ה-*i *במערך* countries*). נעדכן את* list\_index *כך:* list\_index-=days *(*days *התקבל בקריאה לפונקציה).*

*נוציא את חוליה* i *מהרשימה שלה ע"י חיבור* previous *ו-*next *אחד לשני, ונחבר את החוליה* i *לרשימה* list\_index *המעודכן (ישר אחרי ה-*head*, כמפורט בפונקציה* Insert*). אם* days>list\_index-lockdown\_today*, תיזרק הודעת שגיאה.*

סיבוכיות זמן: *מכיוון שלא עברנו על איברים באופן סדרתי, ורק ביצענו מס' סופי של פעולות חישוב/השמה מסדר* O(1)*, ומכיוון שמתקיים , זוהי גם סיבוכיות הזמן של הפונקציה:* O(1)*, כנדרש.*

***Lockdown\_now:***

ניגש למצביע lockdown\_today. הוא מצביע על תא במערך remainingDays, כך שכל המדינות שנמצאות ברשימה המקושרת של התא הזה נכנסות לסגר באותו היום. לפיכך, נעבור על כל איברי הרשימה המקושרת, שלפי נוסח השאלה, מספרם הוא k. לאחר הדפסה של מדינה, נוציא אותה לרשימה מקושרת temp, כאשר בחוליה של כל מדינה list\_index יתעדכן להיות lockdown\_today+between\_lockdowns(modD). כאשר ברשימה לא יוותרו חוליות מלבד ה-head, נעבור על האיברים שהכנסנו לtemp, ונקשר כל חוליה לרשימה המקושרת המתאימה לה (כבר עדכנו את list\_index של כולן). בסוף התהליך נקדם את lockdown\_today כך: lockdown\_today++(modD) (אריתמטיקה של מצביעים).

סיבוכיות זמן: *המעבר הסדרתי על* k *איברי הרשימה מתבצע ב-*O(k)*, וכל שאר הפעולות הן פעולות חישוב/השמה שהסיבוכיות שלהן היא* O(1)*, לכן סה"כ הסיבוכיות היא* O(k+x·1)=O(k), *כנדרש.*

*לפיכך הצגנו מבנה נתונים מסוג מערך רב מימדי, שסיבוכיות המקום שלו היא O(N+D) וסיבוכיות הזמן של ביצוע הפעולות עליו עומדות בדרישות השאלה, כנדרש.*

**שאלה 5**

לקראת פתיחת השמיים וסוף סוף נוכל לטייל בעולם, חברת CoroFly החליטה לנהל סקרים על מנת להתארגן ולהסדיר טיסות עתידיות.

לחברת CoroFly יש הסכם עם N מדינות המיוצגות ע"י האינדקסים 1 עד N.

ממשו מבנה נתונים אשר תומך בפעולות הבאות:

* 𝑰𝒏𝒊𝒕(𝑵) - מאתחלת את המבנה עם N מדינות.

**סיבוכיות זמן: (𝟏)𝑶.**

* 𝑭𝒍𝒚(𝒋, 𝒊) - מוסיפה לתוצאות הסקר כי אדם מסוים ממדינה j רוצה לטוס למדינה i.

**סיבוכיות זמן: (𝟏)𝑶.**

* 𝑨𝒓𝒓𝒊𝒗𝒂𝒍𝒔(𝒊) - מחזירה את מספר האנשים שאמרו שרוצים לטוס למדינה ה-i.

**סיבוכיות זמן: (𝟏)𝑶.**

* 𝑫𝒆𝒑𝒂𝒓𝒕𝒖𝒓𝒆𝒔(𝒋) - מחזירה את מספר האנשים שאמרו שרוצים לטוס מהמדינה ה-j.

**סיבוכיות זמן: (𝟏)𝑶.**

* (𝑭𝒂𝒗𝒐𝒓𝒆𝒅(𝒌 - מדפיסה את 𝑘 המדינות המועדפות (בסדר יורד) שאנשים רוצים לטוס

אליהן ברגע הקריאה לפעולה.

במקרה שישנם כמה פלטים אפשריים העומדים בתנאי הפונקציה, די בהדפסת אחד מהם.

**סיבוכיות זמן: (k)𝑶.**

* ()𝑨𝒗𝒐𝒊𝒅𝒆𝒅 - מדפיסה את כל המדינות שאף אחד לא רוצה לטוס אליהן.

**סיבוכיות זמן: (𝑂(𝑟, כאשר 𝑟 הוא מספר המדינות שיש להדפיס.**

סיבוכיות המקום הנדרשת עבור המבנה הינה 𝑂(𝑁).

**הערות:** שימו לב שאסור לאדם להצביע שהוא רוצה לטוס למדינה שלו.

נשים לב כי גם בתרגיל זה, ישנה דרישה לסיבוכיות אתחול O(1), לכן, כמו בתרגיל 4, נשתמש במערך רב מימדי: נשתמש במערך אחד באורך N בשם countries, (ובעוד שני מערכים לצורכי אתחול, לא נפרט עליהם) וברשימה מקושרת דו כיוונית של מדינות. כל איבר ברשימה יכיל את אינדקס המדינה, מספר בקשות לטוס אליה, מספר בקשות לטוס ממנה, מצביע קדימה ומצביע אחורה.

***Init:***נאתחל מערך (ב- O(1) לפי הנלמד בהרצאה) שיכיל זוגות של [מצביע, דגל] לאיבר ברשימה מקושרת דו כיוונית של המדינות שלא ביקשו לטוס אליהן. בעת האתחול נאפס את ערכי הבקשות (אל ומ-), נתחזק מונה שיאמר כמה מדינות יש לנו ברשימה.  
כל מדינה שתתווסף נוסיף לתחילת הרשימה (O(1)) ונעדכן את המצביע לראש הרשימה (נשמור אותו במערך באיבר מספר 0 (אין מדינה באינדקס כזה).

***Fly:*** אם i==j נחזיר שגיאה. נבדוק מה הדגל של i (המדינה שאנו רוצים לטוס אליה), אם הדגל הוא אפס נוציא את המדינה מרשימת המדינות שלא רוצים לטוס אליהן בO(1), נעביר לרשימת המדינות שבדיוק אדם אחד רוצה לטוס אליה (O(1)), נעדכן את ערכי אל ומ- המתאימים (O(1)), נעדכן את הדגל של המדינה להיות 1.

***Arrivals***:

אם הדגל של המדינה הוא 0, נחזיר 0.  
אחרת – נחזיר את הערך בתא המתאים באיבר ברשימה המקושרת שמחזיק את המדינה. ניתן לגשת למידע הדרוש ב-O(1) באמצעות המצביע שנמצא במערך.

***Departures:***

נחזיר את הערך של "מ-" באיבר שעליו מצביע הפוינטר במערך המדינות במקום ה-j.

***Favored(k):***

לצורך חישוב זה, נחזיק רשימה מקושרת שבה כל איבר הוא מצביע לראש רשימה מקושרת שמייצג ערך מספרי עבור כל ערכי הרשימה (כל המדינות שרוצים לטוס אליהן d אנשים).  
בעת הרצת הפונק' נרוץ על רשימת המצביעים מהמצביע שמייצג את הערך הכי גבוה, עבור כל מצביע נדפיס את כל המדינות שנמצאות ברשימה שלו. אם סיימנו לעבור על כל המצביעים והרשימות ועדיין לא הגענו ל-k, נמשיך להדפיס מרשימת המדינות שאליהן אף אחד לא ביקש לטוס עד להגעה ל-k מדינות.  
סיבוכיות המיקום O(N), סיבוכיות הזמן – O(k) כאשר k מספר המדינות המבוקשות.

***Avoided():***

נקרא לאיבר הראשון ברשימת המדינות שיצרנו ב-Init (ברשימה זו כל המדינות שעדיין לא ביקשו לטוס אליהן), נעבור על כל הרשימה ונדפיס את האינדקסים – O(r) כאשר r הוא מספר המדינות ברשימה.

איבר ברשימה המקושרת הדו-כיוונית של כל מדינה:

int **country\_id** - (i)

int **from**

int **to**

country\_node\* **previous** - (i-1)’th country node pointer

country\_node\* **next** – (i+1)’th country\_node pointer

typedef struct country\_node:

איבר ברשימה המקושרת של מצביעים סופרים (כל מצביע הוא ראש רשימה של המדינות בעלות ערך to שווה:

country\_node\*\* **head** – first country in the list

counter\_node\* **previous** - (i-1)’th counter\_node pointer

counter\_node\* **next** – (i+1)’th counter\_node pointer

typedef struct counter\_node:

***המערך* countries**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | · · · |  |  |  | **[FLAG, ]** |
| *N-1* | *N-2* | *N-3* | *N-4* |  | *3* | *2* | *1* | *0* |