**אלגוריתמים 1 – התכוננות למועד ב'**

1. **בעיית החנייה:**

תיאור הבעיה:

נתונה חנייה מעגלית עם סימנים בכל מכונית וקיים חוקר שעליו לספור את מספר המקומות בחנייה.

החוקר יכול ללכת צעד אחד קדימה או צעד אחד אחורה בכל פעם, ובאפשרותו לשנות את סימן החנייה.

פתרון:

* החוקר יסמן ב- V את הרכב הראשון ויתקדם תוך כדי ספירה, כשיראה שוב את הסימן V ישנה אותו ל- W ויחזור אחורה כמספר הצעדים שספר.
* במידה והגיע ל- V (לאחר שחזר אחורה), ימשיך למקום של ה- W וימשיך להתקדם בהתאם.
* בסופו של דבר החוקר ליגיע ל- V ההתחלתי, ישנה אותו ל- W, יחזור אחורה כמספר הצעדים שספר ואם יראה W (שקודם היה V) סימן שסיים.

סיבוכיות:

במקרה הטוב- אם יש רק V אחד בהתחלה נבצע 2n צעדים, כלומר O(2n)=**O(n)**

במקרה הגרוע – אם יש V בכל הרכבים בחנייה, הסיבוכיות היא סכום סדרה חשבונית, כלומר O(n\*(n+1)/2)O().

1. **משחק המספרים:**

תיאור הבעיה:

נתון מערך בגודל n (מס' זוגי) ויש 2 שחקנים במשחק.

כל אחד מהשחקנים בתורו בוחר מספר מהקצה השמאלי או מהקצה הימני של המערך.  
המנצח הוא בעל סכום המספרים הגדול ביותר.

פתרון:

1. **אלגוריתם חמדני** – בוחר מבלי לקחת בחשבון את צעדי ההמשך, לא תמיד מוביל לניצחון.
2. אסטרטגיית ניצחון לשחקן A – אינה נותנת רווח מקסימלי.  
   חישוב סכום האיברים הנמצאים במקומות הזוגיים במערך, וכן את הסכום במקומות האי זוגיים במערך, **יש לבחור את הסכום הגדול מביניהם.**
3. **אסטרטגיה** **אדפטיבית** – אינה נותנת רווח מקסימלי.  
   נקבעה בתהליך של פתרון הבעיה על בסיס הצטברות של המידע החדש.

בתחילה נקבעים סכומים זוגיים ואי זוגיים, לאחר הבחירה הראשונה מחסירים מהסכומים את שנלקח ובוחרים לקחת מהסכום הגדול יותר, וכך ממשיכים הלאה באותו האופן.

1. **חיפוש שלם** – נבנה עץ של סדרת המספרים וכדי לחשב את הרווח המקסימלי נטייל על העץ מלמטה למעלה.
2. **תכנות דינאמי-**

האלגוריתם בודק את כל האפשרויות שניתן לבחור.

נבנה מטריצת עזר ובאלכסון שלה נשים את איברי המערך הנתון.

נרוץ מהפינה הימנית התחתונה עד לתחילת המטריצה, בכל תא יהיה את מקסימום הרווח שניתן לקבל.

לאחר בניית המטריצה נרוץ מהפינה הימנית העליונה פעמיים ובכל פעם נבדוק מאיפה הגענו על סמך החישוב (פעם בשביל האלגוריתם ופעם בשביל השחקן השני).

I = אינדקס של איבר פנוי שמאלי,

J= אינדקס של איבר פנוי ימני,

F(I,j) = הרווח המקסימלי של שחקן A במצב (I,j)

סיבוכיות האלגוריתם: O(n\*n)=O(n^2)

1. **בעיית החציון:**

**בעיה א'**

בהינתן מערך לא ממוין של מספרים אקראיים, יש למצוא את האיבר הגדול מהחציון  
(חציון = מספר שחצי מאיברי המערך גדולים ממנו וחצי מאיברי המערך קטנים ממנו).

פתרון:

פתרון אופטימלי: אם ניקח את האיבר הראשון a[0], ההסתברות שהוא גדול מהחציון היא 50% (או שיהיה בחצי השמאלי, או שיהיה בחצי הימני של המערך).

אם ניקח את שני האיברים הראשונים של המערך a[0], a[1] ההסתברות שהמקסימום מביניהם יהיה גדול מהחציון היא 75% (25% לטעות).

אם ניקח את שלושת האיברים הראשונים של המערך a[0], a[1], a[2] ההסתברות שהמקסימום מביניהם יהיה גדול מהחציון היא 7/8 (1/8 לטעות).

אם ניקח את 64 האיברים הראשונים של המערך ההסתברות שהגדול ביניהם נמצא במחצית הימנית של המערך היא 1, מכיוון שמספר כל תתי הקבוצות של 64 איברים הוא .

ולכן

כלומר, מספיק לחשב מקסימום של 64 האיברים הראשונים של המערך ונקבל איבר שגדול מהחציון.

בנוסף, המספר 64 אינו תלוי בגודל המערך, ולכן סיבוכיות האלגוריתם היא O(1) .

**בעיה ב'**

בהינתן שני מערכים שווי אורך ממוינים מהקטן לגדול יש למצוא את כל האיברים הגדולים מחציון במערך מאוחד.

פתרון:

1. פתרון נאיבי : מיזוג 2 המערכים, נקבל מערך ממוין באורך 2n והאיבר במקום ה n הוא החציון. נצטרך לבצע 2n השוואות.
2. פתרון אופטימלי: אלגוריתמים מקבילים – בהינתן 2 מערכים a,b שניהם באורך n, נחזיר מערך c שגם הוא באורך n שמכיל את האיברים הגדולים מהחציון.

Max(a1,bn),max(a2,bn-1),…,(max(an-1,b2),max(an,b1)

הוכחת נכונות האלגוריתם באינדוקציה:

בפעם הראשונה בוחרים max(a1,bn) – האיבר הראשון במערך a והאיבר האחרון במערך b.

חלוקה למקרים:

1. A1>bn – מכיוון ש-b הוא מערך ממוין, לפי תכונות טרנזטיביות a1 גדול מכל איברי b, ולכן נמצא במחצית הימנית של המערך וכל איברי a גדולים מהחציון.
2. A1<bn – יש n איברים קטנים מ bn, לכן זה המקרה ההפוך, כלומר bn נמצא במחצית הימנית של המערך הממוזג.

באותו אופן נמשיך לבדוק את המערכים ללא a1 ו bn.

1. **חישוב חזקה**

**סיבוכיות:**

*כדי לחשב חזקה בסיבוכיות נעביר את החזקה להצגה בינארית.*

*בהתאם למיקום לספרה 1 במס' הבינארי נדע באיזו חזקה להשתמש. נצטרך לשמור בכל שלב את החזקה של x, אך לבצע כפל רק כאשר בהצגה בינארית של המספר .*

*צריך להעלות מספר ממשי x בחזקת n.*

*\*מימוש בלולאה  
\*מימוש ברקורסיה*

*4.1* ***חישוב של סדרת פיבונאצ'י בסיבוכיות***

*כדי לחשב סדרת פיבונאצ'י בסיבוכיות (איבר מהסדרה) נשתמש בכפל מטריצות.*

*נגדיר מטריצה ראשונית להיות ערכי ההתחלה של הסדרה ונכפיל את המטריצה n-1 פעמים.*

1. **בעיית האסירים**

תיאור הבעיה:

בכלא מסוים יש אסירים שכולם נידונים למוות.

מנהל הכלא הסכים לאשר להם חנינה במידה ויצליחו באתגר מסוים.

בכלא יש חדר מיוחד ובו נורה אחת בלבד, כל אסיר יכול להיכנס לחדר מספר פעמים או פעם אחת בלבד באופן אקראי, ולהדליק או לכבות את הנורה.

אם נציג מהאסירים יאמר בזמן הנכון שכל אחד מהאסירים היה בחדר לפחות פעם אחת – כולם יקבלו חנינה והם יצליחו באתגר.

בזמן ההתכנסות האסירים בוחרים נציג שיצטרך לומר לסוהרים כאשר יידע שכולם כבר נכנסו לחדר, ובנוסף *כל אחד מהאסירים יודע כמה אסירים ישנם סה"כ.*

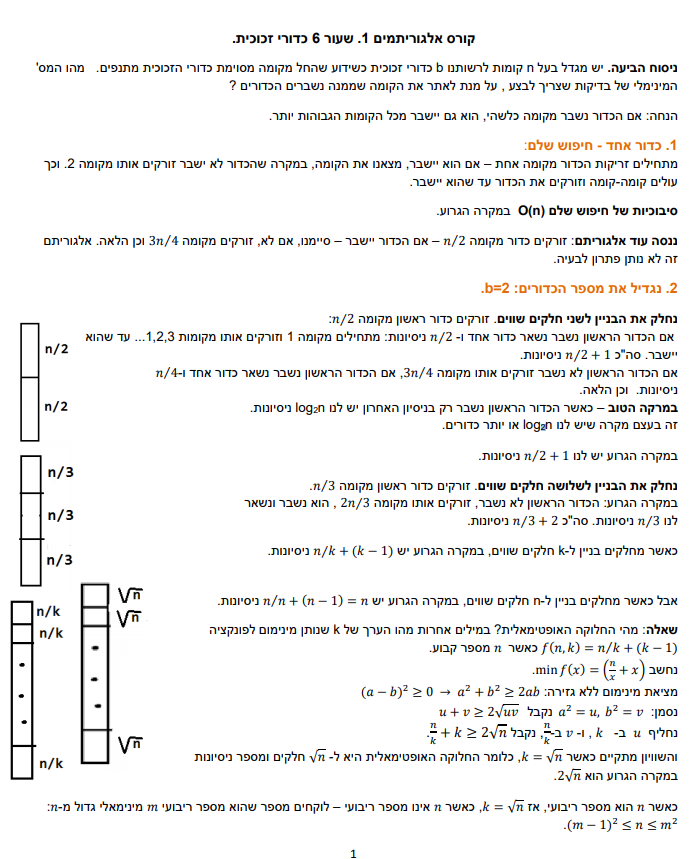
פתרון:

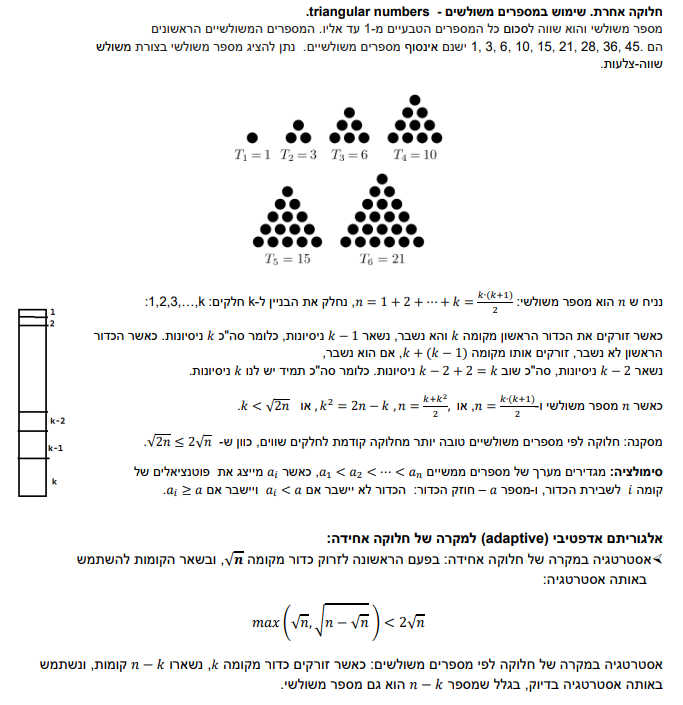
במקרה ומצב הנורה לא ידוע האסירים יצטרכו לכבות את האור פעמיים מכיוון שייתכן ונכנסו לפני הנציג "הסופר".

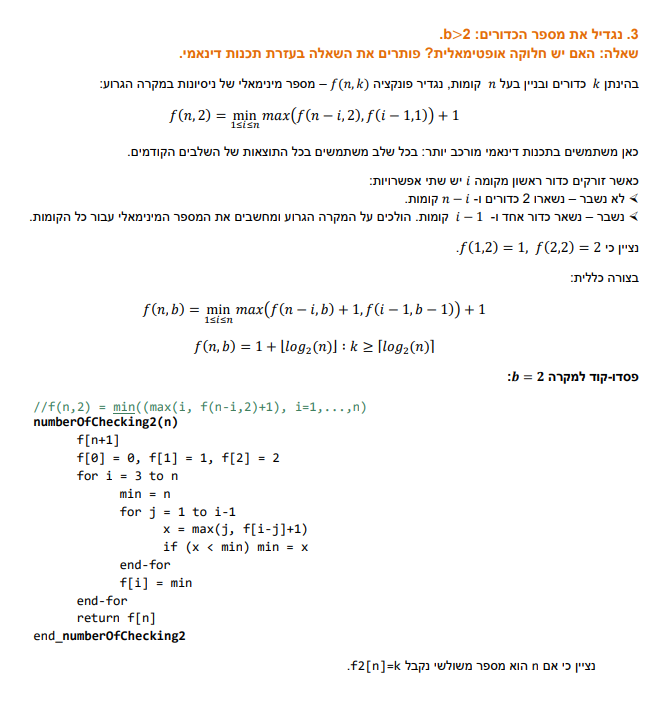
ברגע שכיבו פעמיים – הם יודעים שהנציג הסופר כבר נכנס כי רק הוא מדליק את הנורה.

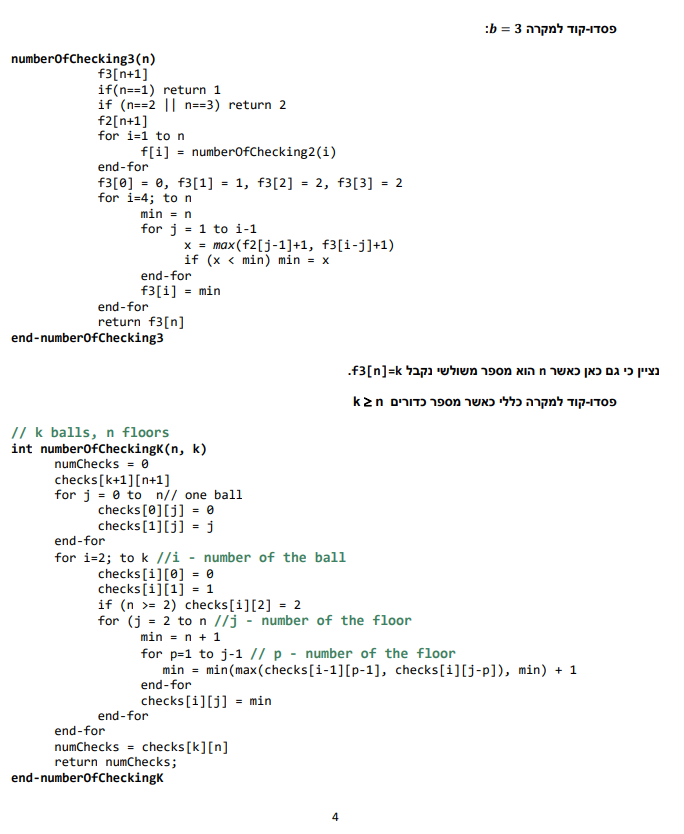
האסירים מיוצגים ע"י מערך של int, והנציג הסופר יודע שעליו לספור 2n אסירים (כל אסיר נכנס ויוצא פעמיים)

1. **בעיית כדורי הזכוכית – לחזור על זה טוב!!!!!!**









1. **LCS – Longest Common Subsequence**

תיאור הבעיה:

בהינתן 2 מחרוזות יש למצוא את תת המחרוזת המשותפת הארוכה ביותר (הסדר חשוב, הרצף לא).

פתרון:

1. אלגוריתם חמדני – לא נותן פתרון אופטימלי, סיבוכיות O(m\*n) כאשר אורך מחרוזת אחת הוא m והשנייה n.
2. חמדן משופר – נבחר בסדרה הקצרה ביותר מבין השניים, לסדרה זו נבנה מערך מונים אשר ימנה את כמות האותיות בסדרה. לאחר מכן נעבור על הסדרה הארוכה יותר וכאשר נמצא אות אשר קיימת במערך המונים – נחסר מהכמות שלה ונוסיף את האות לפתרון.

סיבוכיות: O(m+n)+O(min(m,n)) כלומר מעבר על המחרוזות + מערך עזר.

1. חיפוש שלם – מחפשים את כל תתי המחרוזות האפשריות ומבצעים השוואה  
   סיבוכיות
2. **תכנות דינאמי** -   
   נבנה מטריצת עזר שהשורה והעמודה הראשונה שלה הן 2 המחרוזות (נשמור שורה ועמודה ראשונה לאפסים).  
   עבור כל תא במטריצה:  
   אם האותיות הנמצאות באותו האינדקס זהות – נחשב את התא שנמצא באלכסון העליון + 1  
   אחרת – ניקח את המקסימום מבין התא העליון לתא השמאלי.

סיבוכיות : O(m\*n) + O(m+n) = O(m+n) , מציאת אורך + מציאת המחרוזת.

1. **LIS – תת סדרה עולה ארוכה ביותר**

תיאור הבעיה:

בהינתן סדרה בעלת איברים, יש למצוא את תת הסדרה העולה הארוכה ביותר (הסדר חשוב, הרצף לא).

פתרון:

1. אלגוריתם חמדני – אינו מתחשב בשלבים הבאים ולכן לא יעבוד, סיבוכיות O(n).
2. חיפוש שלם – מחפשים את כל תתי הסדרות של סדרה נתונה ובוחרים בארוכה ביותר. סיבוכיות
3. שימוש ב LCS – נמיין את הסדרה המקורית ונחפש תת מחרוזת משותפת בין המקורית לממוינת,

סיבוכיות:   
האלגוריתם תקין מכיוון שלכל תת סדרה עולה יש התאמה בסדרה ממוינת.

1. **תכנות דינאמי-**

* נבנה מטריצת עזר בגודל המערך, בתא הראשון יהיה האיבר הראשון.
* נבדוק האם האיבר הבא במערך גדול מהאיבר הקודם.
* אם כן – נוסיף אותו לתא הבא באלכסון, נעתיק את השורה ונוסיף אותה לשורה מתחת (אותה השורה עם האיבר הגדול יותר).
* אם האיבר הבא במערך קטן יותר – נסתכל על האלכסון ונמצא בין אילו שני איברים באלכסון הוא מתאים, נחליף בין האיבר (שאותו אנו בודקים) לבין הגדול יותר (מבין השניים) ונוריד למטה את השורה שמעליו.

1. **בעיית המזכירה**

תיאור הבעיה:

משרד מסוים נותן שירות ל-n לקוחות, מטרת המזכירה היא להקטין ככל האפשר את הזמן הממוצע שהלקוחות נמצאים במשרד.

הזמן שלקוח נמצא במשרד מורכב מזמן ההמתנה שלו וזמן הטיפול שלו (עבור כל לקוח i).

זמן ההמתנה הממוצע הוא:

מכאן, מספיק למצוא את המינימום של הסכום שייתן את התוצאה הטובה ביותר.

1. **בעיית הפיצה**

תיאור הבעיה:

אלי ובני הזמינו פיצה משפחתית. מהירות האכילה של אלי גדולה פי X ממהירות האכילה של בני (X>1), וניתן לחלק את הפיצה ל- N משולשים שווים.

במהלך הארוחה כל אחד לוקח משולף נוסף לאחר שסיים את הקודם, אסור ששניהם יגיעו למשולש האחרון בו זמנית, יש למצוא את החלוקה האופטימאלית כך שאלי יאכל כמה שיותר משולשים.

1. **בעיית הצב והארנב**

תיאור הבעיה:

נתון מסלול וקיימים שני רובוטים: אחד מהיר (ארנב) והשני איטי (צב), מהירות הארנב גדולה פי 2 ממהירות הצב. שניהם נעים באותו הכיוון וזוכרים את נקודת ההתחלה.

יש להוכיח כי המסלול הוא מעגלי.

אם הארנב והצב ייפגשו בשלב כלשהו – הרשימה מעגלית, אחרת – היא לא מעגלית.

נסמן:

|  |  |
| --- | --- |
| n | מספר איברי הרשימה |
| k | מס' האיברים מנקודת ההתחלה ועד נקודת המפגש |
| p | מספר סיבובי הצב |
| q | מספר סיבובי הארנב |
| i | מספר צעדים שעשה הצב |
| i\*2 | מספר צעדים שעשה הארנב |

I=np + k

2i = nq+k

נכפיל את המשוואה הראשונה ב – 2 ונחסר בין 2 המשוואות:

2np + 2k = nq + k

K= n(q-2p)

מכאן נובע ש k היא כפולה של n, כלומר הצב והארנב נפגשים בנקודת ההתחלה.

נציב p=1, q=2 ונקבל k=0, כלומר מספר סיבובים מינימלי שמקיים את המשוואה.

* 1. **בעיית הצב והארנב – עם זרוע**

תיאור הבעיה:

נתונה חנייה מעגלית המחוברת לכביש גישה, מעין זרוע וקיימים צב וארנב הנעים במסלול כאשר מהירות הארנב גדולה פי 2 ממהירות הצב.

נקודת ההתחלה של הצב והארנב היא על זרוע, הם אינם יודעים מתי התחילו את הספירה או מתי הסתיימה וזוכרים את נקודת ההתחלה.

יש לבדוק האם יש זרוע ברשימה המקושרת ומהו אורכה.

פתרון:

* נרוץ עד לנקודת המפגש ולאחר שייפגשו, נשים את הארנב בתחילת הרשימה, והצב יישאר בנקודת המפגש.
* שניהם מתחילם לזוז במהירות של הצב (לשניהם יש לעשות m צעדים עד נקודת ההתחלה של המעגל), כלומר נקודת המפגש שלהם היא נקודת ההתחלה של המעגל – מספר צעדים שהם עשו הוא אורך הזרוע.
* כאשר הם עומדים בנקודת ההתחלה של המעגל – הארנב יישאר בנקודת ההתחלה של המעגל, והצב הולך וכאשר יגיע לארנב אורך המעגל יהיה מספר צעדים שהצב עשה.
* אורך הרשימה = אורך הזרוע + אורך המעגל.

ראשית, נוכיח כי הם נפגשים:

נסמן:

|  |  |
| --- | --- |
| n | אורך המעגל |
| m | אורך הזרוע |
| k | המרחק מנקודת כניסת המעגל עד לנקודת המפגש |
| p | מספר הסיבובים השלמים של הצב |
| q | מספר הסיבובים השלמים של הארנב |
| i | מספר צעדים שעשה הצב |
| i\*2 | מספר צעדים שעשה הארנב |

I = m + np + k

2i = m + nq + k

נכפיל את המשוואה ב – 2, ונחסר בין 2 המשוואות:

2m + 2np + 2k = m + nq + k

K = n(q-2p) – m  
מכאן נובע ש- k= n – m (לא משנה כמה סיבובים הם עשו), כלומר נקודת המפגש נמצאת במרחק

n-m מתחילת המעגל.

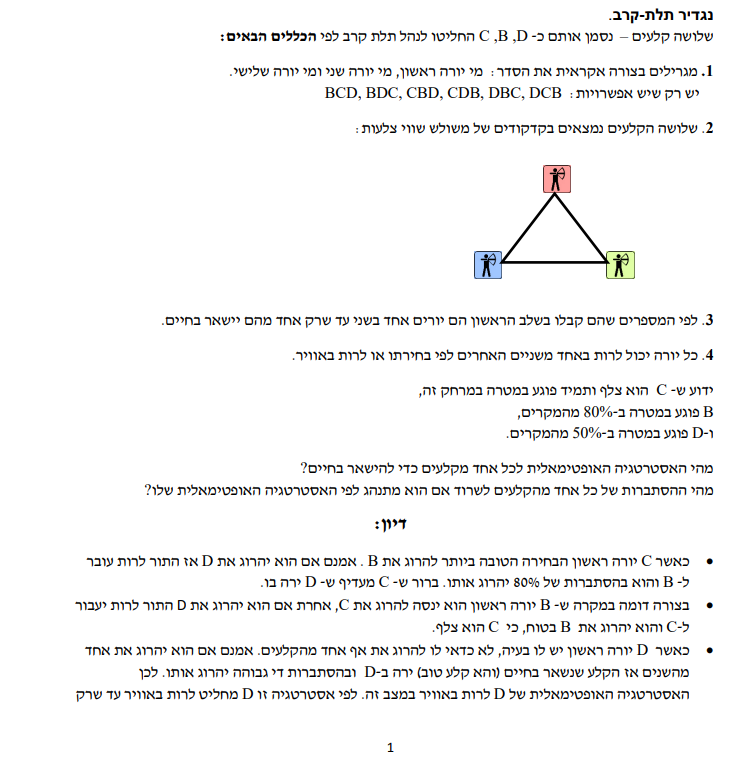
1. **תלת קרב**

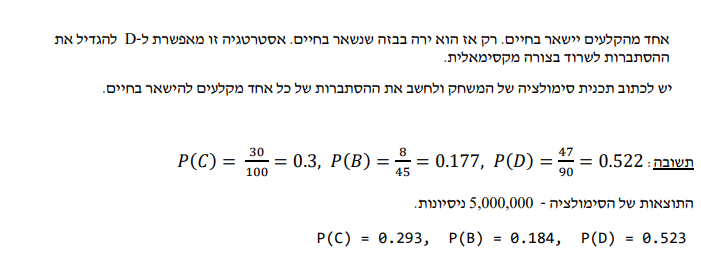
תיאור הבעיה:

ישנם 3 ציידים הנלחמים זה בזה. לצייד הראשון 100 אחוזי פגיעה, לשני 80 אחוזי פגיעה, לשלישי 50 אחוזי פגיעה. סדר השחקנים נקבע באופן רנדומלי.

בהתאם לעדיפות שהציידים מקבלים בשלב הראשון הם יורים אחד בשני עד שרק אחד מהם יישאר בחיים, כאשר כל צייד יכול לירות באחד מהשניים האחרים לפי בחירתו, או לחילופין – לירות באוויר.

פתרון:





1. **בעיית המטוס**

תיאור הבעיה:

* נתונה רשת MXN עם M קודקודים על ציר ה- X ו-N קודקודים על ציר ה-Y.
* על כל צלע רשומה עלות מעבר על הצלע.
* המטריצה בנויה מקודקודים, וכל אחד מהקודקודים הוא Node שמכיל את:

1. העלויות של שתי הצלעות היוצאות ממנו לכיוונים ימינה ומטה:

X – עלות של הצלע אופקית (ימינה)

Y – עלות של הצלע האנכית (למטה),

1. Price – העלות המינימלית של הגעה לקודקוד
2. nPaths – מספר המסלולים עם העלות המינימלית המאפשרים הגעה לקודקוד.

* יש להגיע מנקודה (0,0) לנקודה (m,n) בעלות המינימלית, ניתן לזוז רק כלפי מטה וימינה.
* לא ניתן לחזור שמאלה או כלפי מעלה.

פתרון:

1. אלגוריתם חמדני – לא עובד מכיוון שלא חושב על הצעדים הבאים.
2. חיפוש שלם – מציאת כל המסלולים. סיבוכיות
3. תכנות דינמי – נשתמש במטריצת עזר:

* כל תא במטריצה יהיה Node המייצג את הקודקוד.
* ראשית, נמלא את השורה והעמודה הראשונה של המטריצה:

nPaths יהיה 1 (מכיוון שיש רק מסלול יחיד אליהם).

Price יהיה העלות המינימלית של הקודקוד הקודם + הצלע שממנה הגענו אליו.

* לאחר מכן, נתחיל לרוץ על המטריצה מקודקוד (1,1) ועבור כל תא נחשב את העלות המינימלית אליו ואת כמות המסלולים בעלי העלות המינימלית (סכום של כמות המסלולים המינימליים).
* בקודקוד (m,n) נמצא את ה- price של העלות המינימלית (של המסלול הקצר ביותר) וכמות המסלולים המינימליים הקיימים.
* כדי למצוא את ערך המסלול נחזור אחורה מקודקוד (m,n) לכל תא שמקיים שוויון.

*mat(i,j).price=min(mat(i-1,j).price + mat(i-1,j).y, mat(i,j-1).price +mat(i,j-1).x)*

מימוש האלגוריתם:

1. מספר מסלולים בעלי עלות מינימלית –
2. חישוב מסלול אחד טוב ביותר –
3. חישוב כל המסלולים הטובים ביותר – פונקציה רקורסיבית

1. **בעיית הסופגניות**

תיאור הבעיה:

* במחבת יש מקום ל- P סופגניות, ויש K סופגניות סה"כ.
* זמן ההכנה לכל סופגנייה הוא 2 דקות (דקה לכל צד).
* מהו הזמן המינימלי להכנת K סופגניות?

פתרון:

טענה: הכנת K סופגניות דורשת K דקות.

**הוכחה א'**: אם K הוא מספר זוגי : K=2\*m, כאשר m הוא מספר הזוגות של סופגניות במחבת. במקרה זה נכין את הסופגניות בזוגות, 2 דקות הכנה עבור כל זוג סופגניות.

אחרת, K מספר אי זוגי כלומר K=2\*m + 1, במקרה זה יהיו m-1 זוגות שעבורם יהיו 2 דקות הכנה והזוג האחרון יצטרף לסופגנייה הבודדת והם יהיו שלישייה שעבורם יידרשו 3 דקות הכנה.

K=2\*(m-1)+3 = 2\*m + 1.

**הוכחה ב'**:

אם K מתחלק ב-3: K=3\*m, נכין בשלשות, כל שלישייה 3 דקות הכנה (הראינו בהוכחה של א').

אם K=3\*m + 2 : יש m שלישיות וזוג אחד.

אם K=3\*m + 1: יש m-1 שלישיות שיידרשו 3 דקות הכנה עבור כל אחת, ואז תישאר שלישייה אחת ועוד סופגנייה אחת שאותם נחלק ל- 2 זוגות, כל אחד מהזוגות ייקח 2 דקות הכנה.   
K=3\*(m-1)+4 .

1. **בעיית הקומפיילר**

תיאור הבעיה:

כל תוכנה בקומפיילר מורכבת מאורך li והסתברות שימוש pi, כדי להגיע לתוכנה כלשהי הקומפיילר צריך לעבור את כל התוכנות הנמצאות לפניה.

יש לסדר את התוכנות כך שזמן הריצה הממוצע יהיה הנמוך ביותר. יש n! אפשרויות לסדר את התוכנות.

יש זמן ריצה ממוצע T ויש למצוא לו זמן מינימלי:

פתרון:

1. חיפוש שלם – לסדר המורכב מ- n איברים יש n! > אפשרויות . לא יעיל!
2. נניח שכל ההסתברויות שוות – במקרה זה:

קיבלנו בעצם את בעיית המזכירה וכדי שזמן הריצה יהיה מינימלי יש למיין את התוכנות (מערך של li) בסדר עולה.

... המשך הוכחה בספר.