GREEDY

מה זה אלגוריתם גרידי?

- גרידי הוא סוג של אלגוריתם שמקבל החלטות מקומיות שנראות הכי טובות באותו רגע, מתוך תקווה שזה יוביל לפתרון אופטימלי בסוף.
- בכל שלב נבחר את האפשרות שנראית הכי "טובה" עכשיו – בלי להסתכל קדימה או לחשב את כל האפשרויות.

איך פותרים שאלה באופן גרידי

- הרבה בעיות לא ניתנות לפתרון ישיר באמצעות גרידי – האלגוריתם לא תמיד מוביל לפתרון הנכון. אז איך בכל זאת יודעים מתי גרידי כן מתאים?
- כדי לפתור שאלה באופן גרידי, צריך לזהות תכונה עקרונית של הבעיה – תובנה שמצדיקה את הבחירה המקומית בכל שלב.
כלומר, לרוב האתגר הוא לא לכתוב את האלגוריתם, אלא להגיע לתובנה שתפשט את הבעיה.
- במילים אחרות: הפתרון הגרידי מגיע רק אחרי שמבינים משהו עמוק על המבנה של השאלה.

אז איך משתפרים?

- גרידי הוא לא נושא שיש בו הרבה חומר תיאורטי, מה שצריך זה פשוט לתרגל ולראות שאלות כדי להשתפר בלהגיע לתובנות מתאימות, אז בואו נתרגל.

CSES – ANOTHER GAME

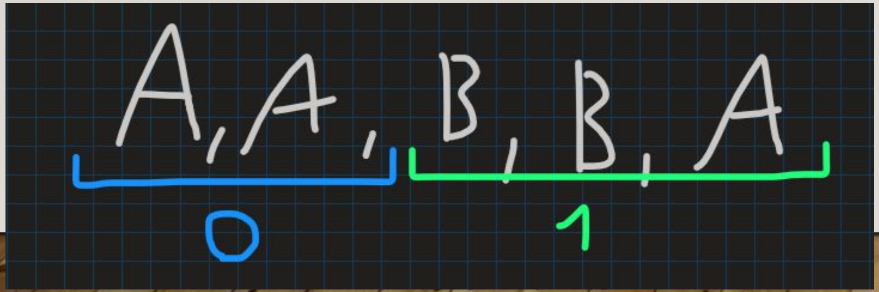
- ישנן n ערימות, בערימה ה-i יש a_i מטבעות.
- שני שחקנים משחקים משחק. כל אחד בתורות בוחר מספר ערימות לא ריקות גדול מ-0, ומוריד מכל אחת מהן מטבע אחד. המנצח הוא השחקן שלוקח את המטבע האחרון.
- בהינתן מספר n ומערך a שמייצג את מספר המטבעות בכל ערימה, מי מנצח אם שני השחקנים משחקים אופטימלית- השחקן הראשון או השני?
- סיבוכיות זמן – $O(n)$.

פתרון

- הבחנה- אם בתורי יש רק ערימות עם כמות מטבעות זוגית, אז אני מפסיד מכיוון שלא משנה איזה ערימות אבחר, היריב יחקה אותי, ויצא שהוא יוריד את המטבע האחרון, כי בתור הבא שלי שוב יהיו רק ערימות עם כמות זוגית, עד שלא יהיו עוד מטבעות.
- אם בתור שלי חלק מהערימות זוגיות וחלק אי זוגיות אני אוכל להפוך את כולן לזוגיות, ולכן אני במצב מנצח.
- השחקן הראשון יפסיד רק אם כל הערימות בהתחלה הן זוגיות, אחרת ינצח.
- פתרון- נעבור על כל הערימות, אם קיימת ערימה אי זוגית אז המנצח הוא הראשון, אחרת השני.

COMPETITIVE FISHING

- נתונים n דגים ממוספרים מ־1 עד n, וכל דג שייך או לאליס או לבוב. נרצה לחלק את הדגים ל־m קבוצות רציפות ולא ריקות, כך שהקבוצה הראשונה תקבל ערך 0, השנייה ערך 1, וכן הלאה, והערך של כל דג הוא ערך הקבוצה שהוא נמצא בה. ניקוד של שחקן הוא סכום ערכי הדגים שתפס. מהו המספר המינימלי של קבוצות m שיש לבחור, כך שהניקוד של בוב יעלה על זה של אליס בלפחותk נקודות? ואם אין כזה – יש להחזיר שאין פתרון. סיבוכיות O(n).
- דוגמה לפתרון עבור k=1, וכאשר A מסמן אליס ו-B מסמן בוב.



פתרון

- כמה בוב יקבל אם נבחר ליצור קבוצה חדשה החל מהדג ה-i?
- נסתכל על הוספת קבוצה כעל הוספת חוצץ. נשים אותו במקום מסוים, והתוצאות יעלו ב-1 על כל דג שמימינו. (זה בעצם מה שקורה בשאלה).
- הבחנה- לבחירות שלנו אין השפעה אחת על השנייה, וכל בחירה תוסיף את ההפרש בין כמות הדגים של בוב מימינה לכמות הדגים של אליס שמימינה. לכן נבחר את הנקודה שההפרש בין כמות הדגים של בוב מימינו לכמות הדגים של אליס מימינו הכי גדול, ואז נבחר את השני הכי גדול...
- פתרון- יהיה לנו מערך בגודל n שזוכר לכל i כמה מקומות שונים יש להרוויח בהם את ה-i הזה אם נשים בו חוצץ. נלך מימין לשמאל ונזכור לכל נקודה את ההפרש בין כמות הדגים של בוב מימינה לכמות הדגים של אליס מימינה, אם זה גדול מ-0 נעלה את המקום המתאים במערך ב-1, ובסוף נלך על המערך מימין לשמאל ונבחר את התוספות הכי גדולות, ברגע שעברנו את k הכמות שבחרנו היא ה-M שלנו.

CSES - MISSING COIN SUM

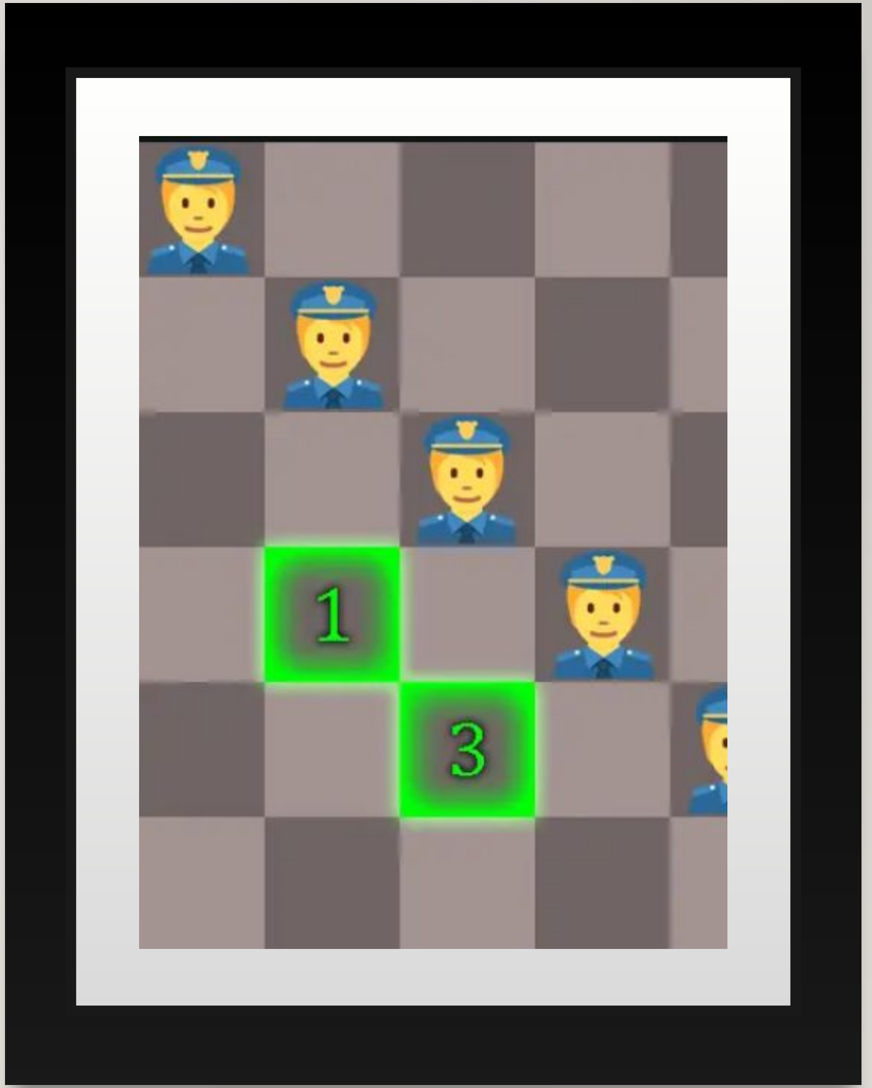
- בהינתן n מטבעות עם ערכים חיוביים, מה סכום הכסף הכי קטן שאתה לא יכול ליצור?
- סיבוכיות זמן – O(nlog(n))

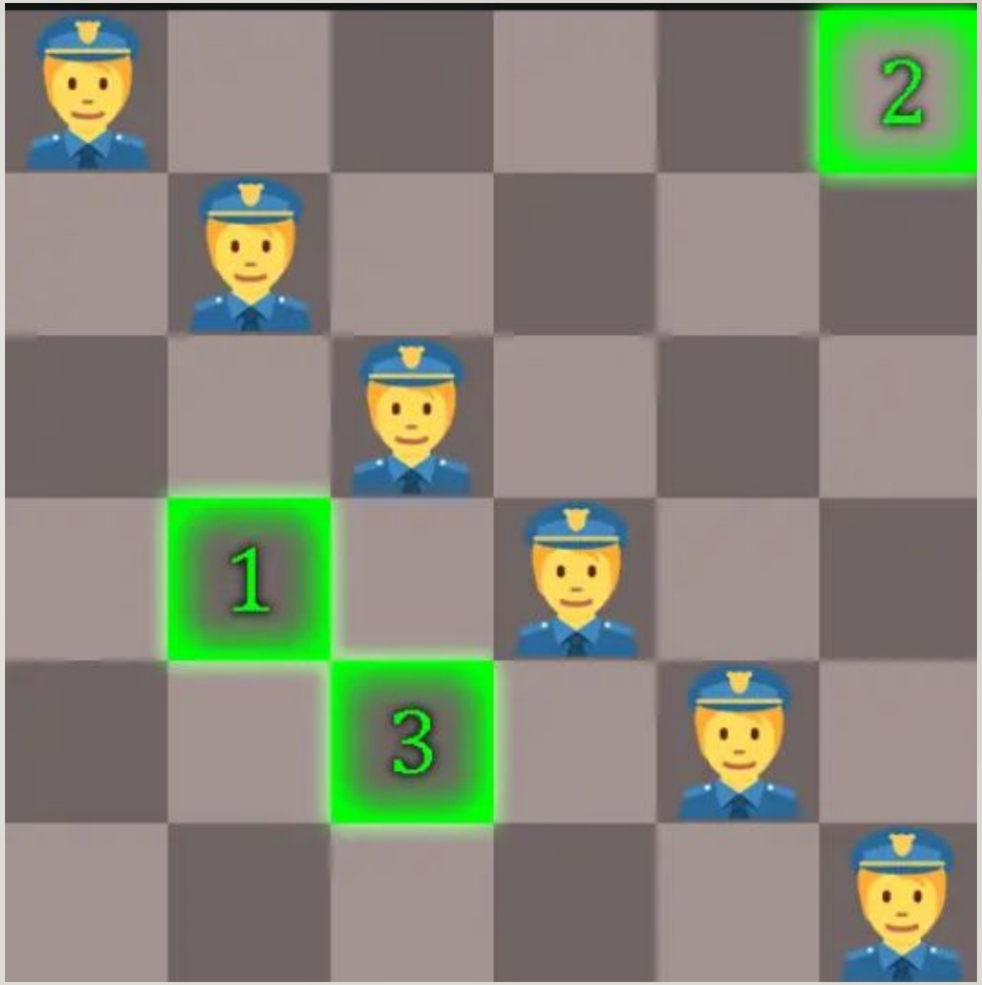
פתרון

- הבחנה- אם בעזרת המטבעות הנוכחיים אנחנו יכולים ליצור את כל הסכומים $1,2,\dots,k$, ונוסיף את המטבע x, אם x קטן או שווה ל-k, אז נוכל ליצור את כל הסכומים עד $k+x$, אבל אם $x>k+1$ עדיין לא נוכל ליצור את $k+1$. (מטבע לא עוזר ליצור סכומים שיותר קטנים ממנו).
- פתרון: נמיין את מערך המטבעות. נעבור עליו ונזכור את הסכום שלנו שמאותחל ב-0, נקרא לו sum. כשמגיעים למטבע מסוים נבדוק אם הוא יותר גדול מ-sum+1. אם הוא יותר גדול אז גם כל המטבעות הבאים יהיו יותר גדולים מ-sum+1, לכן לא נוכל ליצור את sum+1. אם הוא קטן או שווה אז נוכל ליצור את כל הסכומים עד sum+coin לכן נוסיף אותו לסכום שלנו ונמשיך הלאה (כל זה לפי ההבחנה).

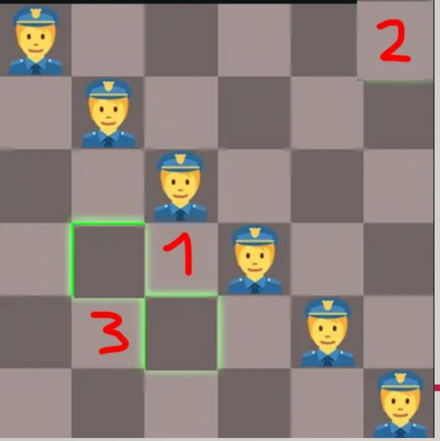
BASRENG

- אתה רוצה למכור את הסחורה הגנובה שלך בBasreng, עיר שיכולה להיות מיוצגת כמטריצה בגודל 2n על 2n.
- באלכסון הראשי של המטריצה יש שוטרים, מותר לך שכל שוטר יראה אותך פעם אחת, אם הוא יראה אותך פעמיים אז תיתפס.
- המטרה שלך היא לבחור נקודות כך שכל שוטר רואה אותך פעם אחת (נקודה אחת בעמודה או בשורה שלו), ומרחק ההליכה שלך אם תלך מנקודה 1 ל-2 ל-3 ל-4... הוא מקסימלי אפשרי. אם יש כמה פתרונות אפשריים תדפיס אחד. אי אפשר ללכת באלכסון.





- זאת דוגמה עבור $n=3$. אתה צריך לבחור 3 נקודות כך שמרחק ההליכה שלך יהיה כמה שיותר גדול, אם אתה הולך לנקודות לפי הסדר, אין מעברים באלכסון.
- בדוגמה הזאת מרחק ההליכה הוא 14, זאת אחת הדרכים האופטימליות עבור $n=3$.
- מצאו בזמן $O(n)$ דרך אופטימלית והדפיסו אותה.



פתרון

- ברור שאופטימלי לבחור נקודה בצד ימין למעלה של האלכסון הראשי, והבאה בצד ימין למטה שלו... כלומר לא לבחור פעמיים ברצף נקודה באותו צד של השוטרים.
- הבחנה I- קיים פתרון אופטימלי בו כל הנקודות שבחרת הם על האלכסון המשני. לא קשה לראות את זה אז לא נתעכב על ההוכחה, ונניח שכל הנקודות באלכסון המשני כי זה יעזור.
- הבחנה 2- ניתן להציג את התוספת של כל נקודה בהליכה בתור המרחק לבוא אליה מהחצי שלה של הלוח (של השוטרים), ועוד המרחק לחזור ממנה אל החצי השני של הלוח, מלבד הנקודה הראשונה והאחרונה כי אליהן לא הולכים או לא חוזרים.
- פתרון- נבחר את הנקודה הראשונה להיות נקודה קרובה לשוטרים באלכסון המשני, אז נבחר את הנקודה הכי רחוקה בצד השני של האלכסון המשני, אז נבחר את הנקודה הכי רחוקה שניתן באלכסון המשני... ונעשה ככה ל-n נקודות. זה כמו את הנקודה הראשונה לשים קרוב, ומאז לבחור כל פעם את הנקודה הכי רחוקה שאפשר בצד ההפוך של האלכסון המשני

MAKE A PALINDROME

- אתה מקבל מספרים n,k ומערך של מספרים. המטרה שלך הוא להפוך את המערך לפלינדרום. את יכול לבצע את הפעולה הבאה כל עוד גודל המערך גדול או שווה ל-k:
- תבחר תת מערך (רציף) בגודל לפחות k, ותוריד את האיבר ה-k-th הכי קטן בו.
- תדפיס "כן" אם אפשרי להפוך את המערך לפלינדרום, ו-"לא" אם זה לא אפשרי.
- סיבוכיות זמן O(nlogn).

פתרון

- הבחנה 1: אנחנו יכולים למחוק כל מספר שערכו גדול מהאיבר הk הכי קטן במערך, ולא יכולים למחוק אף איבר שערכו שווה לאיבר הk הכי קטן במערך.
- הבחנה 2: אין אף חיסרון בלמחוק איבר שערכו גדול מהאיבר הk הכי קטן במערך, אנחנו יכולים למחוק כל איבר שמקיים את זה.
- נוכל לספור כמה איברים יש שהערך שלהם שווה ל-k הכי קטן במערך, ואפשר בקלות לבדוק כמה מהם ניתן למחוק.
- פתרון: נלך עם שני מצביעים על המערך, אחד מתחיל בשמאל ואחד בימין. אם אחד המצביעים רואה איבר גדול מה-K הכי קטן הוא ימחק אותו, אם הוא רואה איבר קטן מה-K הכי קטן הוא יחכה עד שגם השני יראה אותו איבר, ואם הוא רואה איבר ששווה לk הכי קטן הוא יחכה בו וימחק רק אם חייב (השני רואה איבר קטן מ-k הכי קטן).

MAKE IT ZERO

- נתון מספר n ומערך חיובי a בגודל n, המטרה שלכם היא להפוך את המערך a למערך של אפסים, בעזרת כמה שפחות שימושים בפעולה הבאה:
- תבחר מערך b כך שקיים בו i שבו סכום האיברים עד i (כולל) שווה לסכום האיברים אחרי i.

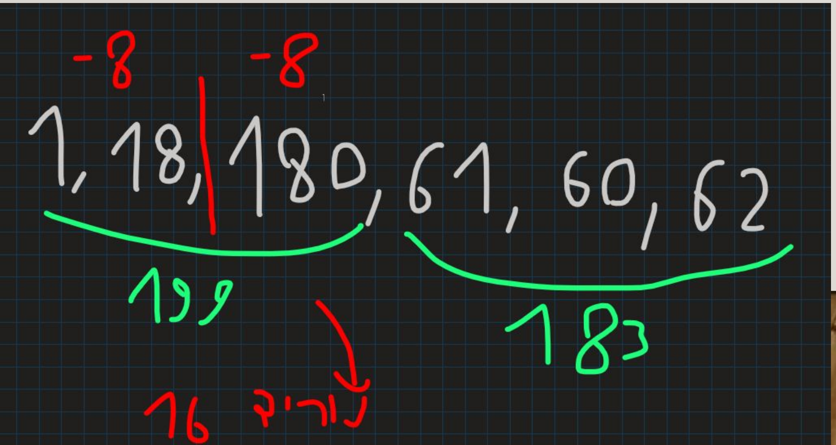
  תוריד מהאברים ב-a את האיבר המתאים ב-b.
- תדפיס את כמות הפעולות המינימליות לעשות זאת, ומה תעשה בכל פעולה, או שזה לא אפשרי.
- סיבוכיות זמן O(nlog(n)).
- דוגמה: עבור הקלט n=4, a = {5,3,1,5} פתרון אפשרי הוא

```
2
3 1 1 1
2 2 0 4
```

אין אף פתרון עם פעולה אחת.

פתרון

- נסתכל על המטרה: המטרה שלנו היא להביא את a למצב בו קיים i כך שהסכום עד i שווה לסכום אחרי i (כי אז נוכל בהכרח לבחור b שיאפס אותו).
- הבחנה- אם נבחר מלכתחילה את ה-i הזה, אז נוכל להסתכל על הבדל הסכומים ההתחלתי בין שני הצדדים של ה-i. אם נבחר מקום התחלתי טוב אז נוכל בפעולה אחת להוריד את ההבדל הזה מאחד הצדדים ולפתור תוך 2 פעולות. למה? כי בפעולה אנחנו יכולים להוריד מספר זוגי מאיזה צד שנרצה, נגיד אנחנו רוצים להוריד x מצד ימין של i, פשוט נבחר "חוצה" בצד השני ונוריד משני הצדדים שלו x/2, רק צריך לוודא שזה אפשרי.



סיום הפתרון

- הבחנות- אם סכום a אי זוגי זה לא אפשרי, כי אז לא ניתן לבחור כי לעולם לא נוכל לבחור מערך b שסכומו אי זוגי. בנוסף אם קיים איבר שגדול מכל שאר המערך זה כמובן לא אפשרי.
- אחרת זה יהיה אפשרי ב-2 פעולות או בפעולה אחת! פשוט נבחר את ה-i שלנו במקום שבו הסכום עד אליו הכי קרוב לסכום אחריו, אם אין הפרש ביניהם כמובן אפשר בפעולה אחת, ואם יש הפרש אז נאפס את ההפרש בכך שנוריד מאיברים בצד היותר גדול. זה בטוח אפשרי אם זה לא מפעיל את הבדיקה של הלא אפשרי שלנו, מכיוון שהסכום בטוח זוגי אז ניתן להוריד הפרש זוגי, ומכיוון שאין איבר אחד שגדול מכל שאר האיברים אז ניתן להוריד משני איברים שונים x/2, זה לא יהיה אפשרי רק במקרה שיש איבר יותר גדול מכל שאר האיברים.
- הקוד לא מסובך, פשוט כמה אפשרויות שונות לבדוק.

-8 -8

$\underbrace{1, 18, 180}_{199}, \underbrace{61, 60, 62}_{183}$

16 נוריד

RECOMMENDATIONS

- ניתנים לך n משתמשים של שירות מוזיקה עם מספרי רצועות מ-1 עד 10^9. כל משתמש אוהב קטע רציף של רצועות $[l_i, r_i]$. משתמש j הוא גורם מנבא עבור משתמש i אם $i \neq j$, ו-[l□, r□] מכיל במלואו את $[l_i, r_i]$. רצועה מומלצת מאוד עבור משתמש i היא רצועה שאינה ב-$[l_i, r_i]$ אלא בטווחי כל הגורמים המנבאים. חשב כמה רצועות מומלצות מאוד יש לכל משתמש (0 אם אין גורמים מנבאים).
- סיבוכיות זמן $O(n\log(n))$.

פתרון

- הבחנה I- ניתן לחלק את כמות ההמלצות עבור i לשני חלקים- ההמלצות מצד שמאל של הטווח וההמלצות מצד ימין. כמות ההמלצות מצד שמאל יהיו ההפרש בין Li לבין הטווח שמכיל את i בעל הL הכי גבוה, ואותו דבר הפוך עבור צד ימין.
- נתחיל בלחשב לכל אחד את כמות ההמלצות שיש לו משמאל, ואז נעשה אותו דבר אבל הפוך עבור ימין. זמן הריצה יהיה רק פי 2 שזה לא מפריע לנו, וזה כמובן יקל הרבה על החישוב.
- הבחנה 2- אם נלך על האיברים באופן ממוין לפי r ונתחיל מה-r הכי גדול ונרד, אז בהוספת איבר חדש אנחנו יודעים שכל האיברים עד עכשיו בעלי r גדול או שווה אליו וכל הבאים לא יהיו (נסדר במיוחד את המקרה של r שווה), אז רק צריך לדעת מה ה-L הכי גדול שראינו עד עכשיו אבל יותר קטן מ-Li, וזאת תהיה התשובה עבור ההמלצות של i משמאל.

המשך פתרון

- הסיבה היא שברור שהאיברים הבאים שנסתכל עליהם לא ישנו לנו כי ה-r שלהם יותר קטן, אז אנחנו בעצם מוצאים את האיבר שמכיל אותי עם L כמה שיותר גדול, כמו שרצינו.
- קל לעשות באמצעות עץ (set): נוסיף לעץ את ה-L של כל איבר כשנוסיף אותו, ונסתכל על ה-L אחד יותר קטן (נעשה – ל-iterator כדי למצוא בlogn).
- רק צריך לשים לב שאם יש שני איברים בעלי r שווה, אז נעבור קודם על האיבר בעל L יותר נמוך כדי שהתוצאות לא ישתנו בהוספת חדש. קל לעשייה בעזרת עריכת פונקציית המיון שלנו.
- נעשה את המעבר הזה פעמיים, עבור L ועבור R, ונסכום את התשובות.

סיכום

- במהלך המצגת עברנו על כמה דוגמאות וראינו איך Greedy יכול לעבוד, לפעמים בקלות מפתיעה, ולפעמים דורש קצת יותר מחשבה. אבל בסופו של דבר, אין דרך אחת ללמוד גריד, צריך לראות אותו בפעולה, להרגיש איפה הוא מצליח ואיפה לא.
- הרבה פעמים יהיו לנו פתרונות שחלקם גרידיים, אז חשוב לשים לב ולזהות מה גרידי.
- יתרון בגרידי הוא שהרבה פעמים הוא לא רק מקצר את זמן הריצה אלא גם את הקוד.