**בעיות שהתגברתי עליהן ואיך:**

1. צריך לסדר הרבה קירות באופן שישאיר את הלוח קשיר בלי לבדוק אם הוא קשיר בכל פעם כמה שרק ניתן (בדיקה כל פעם רצה על כל המשבצות שזה המון)

פתרון: נבנה עץ בBFS מנקודה אקראית בלוח

1.1 בניית עץ כזה תגרום בהכרח רק ללוח שלוקחים לו חלקים מהקצוות

פתרון: על מנת לאפשר לקירות להכנס יותר לעומק של הלוח, נכנס מכל עלה עמוק ככל הניתן, כלומר עד שנגיע לעלה שהסרתו לא תהפוך הורה שלו להיות עלה

2. וידוא כמות שכנים מינימלית עבור כל משבצת

שמנו לב כי לא ניתן לוודא כמות מינימלית גדולה מ-2 כיוון שתמיד יש פינה

פתרון: בחירת המקומות למשבצות הקיר נעשית דרך עותק של הלוח בצורתו הבסיסית בה רשום מספר השכנים של כל משבצת. בכל פעם שנבדוק את אפשרות השמת משבצת קיר בנקודה מסויימת נוריד 1 ממונה השכנים של השכנים בנקודה זו ונבדוק אם מספר השכנים יורד מתחת למינימום- אם כן המקום אסור.

נשים לב כי כל מקום שנבחר נאסר מהרשימה לבדיקות עתידיות בין הוא מיקום חוקי להשמת משבצת קיר או לא, שכן לא נרצה לבחור את אותו מקום פעמיים.

**הישגים תכונתיים:**

1. Bevy עובד בשאילתות על "ישויות" שקיימות בעולם של האפליקציה. לעיתים צריך שרשרת תגובה של מערכות במחלקות שונות על מנת לגרום למשהו להתבצע, כאשר כל מערכת צריכה משאבים שונים.

דוגמה: השחקן לוחץ על מקש או על העכבר על מנת להזיז משבצת, מנהל הלוח מחליף את המיקום הלוגי של המשבצות, מנהל הגרפיקה מחליף את המיקום הגרפי של המשבצות בלוח, מנהל הלוח בודק אם הלוח נפתר ואם כן נועל את הלוח ומדפיס הודעה.

פתרון נאיבי: הגישה הנאיבית תהיה לקשר לאפליקציה את מערכות האינפוט. אלו יקבלו גישה לכל הדברים שידרשו על ידי כל המערכות האחרות ויעבירו להן אותם- מהלוח עם המיקום הלוגי עד לישויות הגרפיות של המשבצות.

בעיה: שיטה זו בעייתית כיוון שהיא יוצרת תלות לא הכרחית בין הפעולות והעברות של הרבה משתנים לא הכרחיים לרוב הפונקציות (ובסופו של דבר, רשימת פרמטרים גדולה בהרבה מהדרוש למערכות האינפוט).

פתרון: נשתמש במערכת של אירועים על מנת להודיע למאזין במערכת הבאה שעליו לפעול. נעביר באירועים את המידע הדרוש ונשייך את המערכות לקבוצות מערכות על מנת לוודא שירוצו בסדר הנכון. בשיטה זו כל מערכת מקבלת רק את הפרמטרים שדרושים לעבודה שלה ישירות מהאפליקציה.

בעיה: כיוון שמאזיני האירועים עוברים על כל האירועים איטרטיבית, עלול להיווצר מצב בו נשלחים שני אירועי אינפוט בו זמנית. Bevy ו-Rust מטפלים אוטומטית במיקבול ונעילה של משאבים, אך עלול להיווצר מצב בו בקשות שהיו חוקיות בעת שליחתן אינן חוקיות יותר.

דוגמה למצב בעייתי: המשבצת הריקה יושבת בתא הראשון ויש לה משבצות איתן היא יכולה להתחלף מימין ומלמטה. אם, למשל, נלחץ בו זמנית על החצים המתאימים, נוכל להעביר הלאה למערכת הלוגית את המיקומים שלהם ואת מיקום המשבצת הריקה. התוצאה של זה תהיה שכאשר המערכת תחליף בין שני המיקומים שקיבלה בפעם השניה היא תנסה להחליף בין שתי משבצות שאינן ריקות, שאסור לשחקן לנסות לבצע.

פתרון: במקום להעביר דברים שעלולים להיות אסורים לנסות לבצע כאשר הם נשלחים במקביל, התוכנה שלי שולחת אירועים עם בקשות אותן תמיד ניתן לנסות לבצע, גם אם אסור (כלומר, הן עלולות לזרוק אזהרה אבל לא שגיאה). למשל- נעביר את הכיוון היחסי למשבצת הריקה ואת מספר המשבצת הריקה אליו הוא מתייחס, וכך תמיד יש משבצת ריקה כחלק מההחלפה. עוד דוגמה תהיה העברת המיקום החדש בו כל משבצת צריכה להיות לגרפיקה, במקום להחליף בין מיקומי משבצות, מאותה סיבה.

2. ישנו, פוטנציאלית, הצורך להשמיד(despawn, destroy) הרבה משבצות ולברוא (spawn, instantiate) הרבה משבצות באותו פריים (למשל, כאשר עוברים מלוח עם n=40 לוח עם n=42). זה מהלך מאוד יקר כיוון שהוא דורש טעינה ושחרור של משאבים רבים במסתכם דבר שעלול לתקוע את התוכנה לזמן רב עבור מספרים גדולים.

פתרון: נחזיק hashmap בו יוחזקו כל המשבצות לפי סוג ואינדקס המשבצת. כאשר ניצור לוח חדש, נתייג תחילה את כל המשבצות בהן אנחנו משתמשים בלוח החדש. כעת נשמיד את המשבצות בהן אנחנו לא משתמשים. היה והלוח החדש גדול יותר, ניצור משבצות חדשות בנוסף למשבצות הקיימות (כלומר, החל מהמספר הבא על מנת שלא יהיו התנגשויות או יווצרו משבצות מיותרות).

**טיוטה:**

נרצה ליצור לוח פתיר, כלומר שניתן לסדר בו את כל משבצות המספר לפי הסדר משמאל לימין ומלמטה למעלה, ובסופן את המשבצות הריקות (זאת תוך התעלמות ממשבצות הקיר (המשבצות החוסמות).

נחלק את הבעיה לשני חלקים:

* החלטת הצורה של הלוח באמצעות סידור משבצות "קירות" חוסמות
* החלטת סידור המספרים בתוך הצורה שקבענו בשלב הראשון

נתבונן בשתי גישות:

* Brute Force:

בגישה זו נתחיל מלוח פתיר ונבלגן אותו באמצעות רצף של פעולות הזזה. כיוון שלכל פעולת הזזה יש פעולת הזזה הפוכה לה, הלוח שיתקבל בהכרח יהיה פתיר (גם אם לא באופן אופטימלי) באמצעות רצף פעולות הפוך לזה בו הוא נוצר.

* פרמוטציה:

נבחר רצף אקראי של המספרים ונסדר אותם במשבצות החוקיות לפי סדר זה שהתקבל. שיטה זו יותר מהירה אך היא אינה מבטיחה לוח פתיר.

ראשית, נתבונן באופן סידור משבצות הקיר. נרצה לוודא:

* שהמיקומים החוקיים (הלא חסומים) יוצרים גרף קשיר.

נתבונן במצב בו יש בלוח אזור מבודד של מספרים אליו לא ניתן להגיע עם משבצות ריקות כלל. לוח זה עלול להיות פתיר אם כל המספרים המנותקים נמצאים במקומותיהם הנכונים, אך אין בהם עניין למשחק ולכן נוותר על הכנת לוחות במקרה זה.

* שהפתרון דורש החלפת סדר המספרים.

כלומר, שהמטרה אינה רק להביא את המשבצת הריקה למקומה אלא לשנות את הסדר בו המספרים מופיעים. לכן נרצה לוודא:

* שהסידור של המספרים הוא לא סידורם הנכון
* שהלוח בנוי כך שניתן לשנות את הסדר
* נכנה מקום בו המספרים מסודרים בסדר הנכון ורק צריך להזיז אותם "מקום מנוון"

נתבונן בנקודה 2b. ישנם שני מצבים בהם לא ניתן יהיה לשנות את סדר המספרים:

* "קו" – כלומר, מצב בו החל מנקודה מסוימת לכל משבצת יש שני שכנים בלבד ולאחרונה מבניהן יש שכן אחד בלבד. במצב כזה לא ניתן להחזיר משבצת ריקה אחורה בלי להשיב את המצב לקדמותו, ולכן על מנת לייצר לוח פתיר נדרש שבכל קו האיברים יהיו בסדר בו הם אמורים להיות בסופו של דבר, אלא אם יש משבצות ריקות כאורך הקו לפחות על מנת להביא את המשבצות שמקומן המקורי (בלוח המבולגן) הוא בתוך הקו על מנת לסדרן מחדש.
* "מעגל" – כלומר, מצב בו לכל המשבצות יש שני שכנים, פרט לנקודות המקשרות, "קצוות" המעגל, להן יש יותר (בהכרח יש לפחות אחת כיוון שהוא קשיר, וגרף שכולו מעגל אחד גדול לא מאפשר שינוי סדר ולכן מנוון).

במעגל לא ניתן לשנות את הסדר, אך הכנסה של משבצת ריקה לתוך המעגל לא תמצה אותה ולכן החוקים החלים על קו לא חלים עליו. גם כאן, על מנת שהמשבצות לא יהיו בסדרן היחסי בתוך המעגל, על המעגל לחפוף לאזור ברוחב שתיים.

נכנה קווים ומעגלים "אזורים מנוונים" ונסתכל עליהם כעל חוליות-על. אזי, על הקשתות, החלקים בלוח שמחברים בין האזורים המנוונים, להיות ברוחב של שתי משבצות לפחות על מנת שאזור מנוון יפסיק להיות מנוון. כיוון שנרצה לוח מעניין כמה שניתן, נשאף לצמצם את כמות ושטח האזורים המנוונים, בכך שנוודא רוחב של שתי משבצות היכן שניתן.