***1)*** סיבוכיות היא כשורש של 2n. O(sqrt[2n]). נגדיר k שווה כsqrt(2n).  
זאת משום שנעשה דילוגים הולכים וקטנים עד לקומה אחרונה.

הפונקצייה השנייה שהתבקשתי לתת היא בסיבוכיות כ- O(k)=O(sqrt[2n]) כי שם חישבתי מה הוא הk. יש שם לולאה שכל פעם מגדילה את k באחד. והיא בתנאי שהסכום של כל המספרים עד k אינו עולה על n. לכן זה קורה k פעמים. (הסכום הזה שווה בערך לשורש 2n )  
  
כמות הדילוגים האפשרי הוא מקסימום k דילוגים.  
בפעם ראשונה נדלג k קומות ,עלות 1, אם ישבר נצטרך לעשות k-1 בדיקות עד אליו, סך הכל k בדיקות  
אם לא נשבר דילוג הראשון נבצע עוד דילוג של k-1 עד עכשיו שתי בדיקות, ואם פה נשבר נצטרך לבדוק מאיפה שלא היינו בו עד פה שזה k-2 בדיקות לכן סך הכל k  
וכן הלאה, ככל שביצענו יותר דילוגים, אם ישבר הכדור אז כמות הקומות שנשאר לבדוק קטן בהתאמה, לכן תמיד יתן k לכן תמיד במקרה הכי גרוע סיבוכיות היא k.  
(כמובן שכאשר נגיע לקצה יש לנו את השארית שעלולה להישאר, אך היא בבירור קטנה מk, אחרת k היה גדול ב1.  
  
את הבדיקה הזאת שתיארתי ביצעתי כבר בבנאי, כאשר הגעתי למסקנה איפה הכדור נשבר, שמרתי את זה במשתנה מחלקה שלי.  
וגם את משתנה המחלקה שלי שסופר כמה כדורים נשברו, עידכנתי תוך כדי הפעולות בבנאי.  
  
לכן מיידית בO(1) יכולתי לתת תשובה בפונקצייה הראשונה והשלישית שהתבקשתי ע"י שפשוט החזרתי מיד את המשתני מחלקה ששמרתי כבר.

***2)*** סיבוכיות שלושת הפונקציות שהתבקשתי לכתוב הוא n\*logn   
Bsearch חיפוש בינארי שיצרתי הוא בסיבוכיות logn משום שכל פעם מחפש רק בחצי מהטווח (אריה במדבר) ומחזיר את אינדקס האיבר הראשון של מספר הגדול\שווה מהמספר הנתון.  
פונקצייה שלי MYlis היא בסיבוכיות n\*logn משום שסך הכל הקציתי מערכים בגודל n שזה סיבוכיות n  
ואז גם ביצעתי לולאה בגודל n בה ביצעתי כל פעם חיפוש בינארי(logn) ועוד כמה פעולות בסיבוכיות של O(1). לכן סך הכל סיבוכיות MYlis היא O(n\*logn)  
  
פונ' lis רק קוראת לפונקצייה הפרטית MYlis ומחזירה איבר אחרון בה לכן סיבוכיות כמו שצריך O(n\*logn)  
פונ' lds יוצר מערך חדש הפוך (לולאה בגודל n ) לכן זה עולה סיבוכיות n ואז קוראת לlis על המערך החדש שעולה כאמור למעלה n\*logn וכמובן n+n\*logn = n\*logn בסיבוכיות. לכן גםlds כמו שצריך בסיבוכיות O(n\*logn)

MYlis(int[] arr){  
 int[] ans  
 int k <- 0

For i:=1 to n // O(n)  
 int spot <- Bsearch // O(logn)  
 if(spot == -1) Then k <- k+1 // O(1)  
 ans[i] <- k // O(1)  
return ans

פונקצייה lbs היא גם בסיבוכיות של O(n\*logn)  
קריאה לפונקצייה MYlis על מערך arr עולה O(n\*logn)  
יצירת מערך revArr הופכי לarr עולה O(n). מציאת MYlis שלו עולה n\*logn הפיכת התוצאה עולה O(n)  
חיבור בין שני המערכים שהתקבלו (באלכסון כדי לא לחזור על מישהו פעמיים) ושמירת תוצאה מקסימלית תוך כדי עולה O(n)  
ואז החזרת המקסימום הזה בתור תוצאה סופית  
סך הכל O(n\*logn)  
  
למה lis נכון:  
T מסמל אלכסון אפשרויות, אלכסון זה מייצג את הפוטנציאל של הסדר העולה, כל איבר נשאף להכניס במיקום הראשון שניתן שישמור על סדר באלכסון הזה(לכן נשתמש בחיפוש בינארי, תמיד האלכסון יהיה ממוין)  
גודל האלכסון הזה הוא התשובה שלנו, משום שהוא הפוטנציאל כפי שהגדרנו, כלומר ניתן ליצור תת סדרה עולה בגודל זה.  
בפונקצייה שלי MYlis בנוסף למציאת הגודל שמרתי אותו בכל איטרציה, כלומר קיבלתי מערך שמייצג את הפוטנציאל בתת מערך בגודל האינדקס המסויים. כמובן שהמיקום האחרון מכיל את הפוטנציאל הגדול ביותר(או ששוה לשאר הפוטנציאלים, בהכרח גדול או שווב לכל השאר) משום שמדובר בפוטנציאל של כל המערך(ולא תת מערך) לכן אותו החזרתי בפונקצייה lis.  
  
בlbs קיבלתי את INC שזה הפוטנציאלים של תתי המערכים מההתחלה עד האינדקס (גודל lis בתת סדרה זו)  
וDEC ייצג את הפוטנציאל lds של תת-המערך מהאינדקס עד הסוף(נותן לנו גודל סדרה יורדת הכי גדולה בתת מערך הזה)  
חיבור באלכסון יתן עלייה עד לאינדקס, ומשם והלאה ירידה, שזה בדיוק מה שרוצים, לכן נבדוק את כל אפשרויות החיבור ונחזיר את התשובה המקסימלית.  
  
DEC מייצג את מה שרצינו משום שעשינו MYlis על המערך ההפוך, קיבלנו פוטנציאלים של עלייה מהסוף להתחלה, שזה כמו ירידה במערך המקורי. אבל זה מההתחלה עד לאינדקס, לכן כשנהפוך את התוצאה קיבלנו את הפוטנציאל של הירידה מהאינקס עד לסוף.

***3)*** סיבוכיות של n\*m .  
 יש לולאה בגודל n שבתוכה לולאה פנימית בגודל m שמעתיק את הנתונים למשתני המחלקה. נותן O(n\*m)   
שתי לולאות אתחול, אחת בגודל n השנייה בגודל m . נותן O(n+m)  
לולאה בגודל n-1 ובתוכה לולאה בגודל m-1 שמפעילות בדיקות בסדרי גודל O(1) ומכניסות נתונים בO(1) לכן יתן סיבוכיות O(n\*m)   
  
סך הכל סיבוכיות של n\*m + n + m + n\*m יתן סיבוכיות O(n\*m)  
  
כל זה קורה כבר בתוך הבנאי. ובסוף אני מכניס את הערכים הרצויים בO(1) לתוך משתני המחלקה המתאימים (כמות המסלולים הקצרים ועלות המסלול הכי קצר)  
  
ושתי הפונקציות שהתבקשתי לכתוב פשוט מחזירות ישר בO(1) את המשתנה מחלקה המתאים.  
  
באתחול הלכתי על השורה והעמודה הראשונה, סתם ככה כמות המסלולים תמיד 1, והמחיר הוא המחיר של הקודם פלוס הצלע ממנו(אנחנו בקצוות, יש או עמודות נוספות לפני או שורות נוסופות לפני, לא יכול להיות שניהם)  
אבל בדקתי קודם שהנקודה לא בתחום מת, אם כן זה אומר שהמחיר 1- כי לא הגיוני, וכמות הדרכים היא 0. מעבר לכך גם בדקתי האם מחיר הקודם הוא 1- כי אם כן לא היה דרך להגיע אליו, ולכן גם אלי אין דרך, וזה כאילו אני בשטח מת (כי פה יש רק דרך יחידה ללכת והיא על הצלעות בקצה)  
  
מכאן לולאה כפולה שבעצם עוברת על כל המטריצה, ולכל נקודה שואלת אם היא בשטח מת, אם כן, אז כאמור מכניסה מינוס 1 בעלות ו0 במסלולים, אותו דבר כאשר שני הנקודות מהם אפשר להגיע אלי מהם הם בעלי עלות של מינוס 1 (כלומר לא ניתן להגיע לאף אחת מהם, ולכן גם אלי לא ניתן להגיע)  
אחרת ניתן להגיע אלי לפחות מכיוון אחד. אם רק מכיוון אחד אז העלות שלי היא עלות הקודם שניתן להגיע ממנו פלוס הצלע המתאימה ממנו, והמסלולים שלי זה בדיוק כמו של הקודם.  
אם משניהם ניתן להגיע, אז עלות שלי היא מינימום מבין עלות הקודמים+הצלע המתאים לכל אחד.  
וכמות המסלולים שלי הוא כמות המסלולים מהאחד שנתן עלות מינימאלית (או סכום המסלולים של שני הקודמים אם העלויות שלהם שווים)   
  
  
  
כל זה נכון משום ש:  
כדי להגיע לצומת נתונה C ניתן להגיע רק דרך שני צמתים A,B (או רק אחד אם מדובר בקצה כמובן)  
לכן עלות של C הוא מינימום מבין העלויות להגיע מA או מB.  
בנוסף כמות המסלולים הקצרים אל C בהכרח הוא כמות המסלולים הקצרים או מA או מB או משניהם יחד.   
כאשר העלות להגיע משניהם שווה, ניתן לבחור בכל אחד מהדרכים גם של A גם של B לכן כמות המסלולים של C שווה לסכום המסלולים.  
אחרת כמות המסלולים לC זה בדיוק כמות המסלולים של האחד ממנו עולה מינימאלי להגיע לC (או A או B )