Artificial Intelligence Lab –

**CSP**

Submission date :11/5/2021

Shadi Halloun – 313552309

Noor Khamaisi – 207076076

קישור ל GitHub שלנו , Source קוד וקובץ הPDF מופיעים בצורה מסודרת.

<https://github.com/shadihalloun35/ArtificialIntelligenceLab.git>

**הקדמה:**

התרגיל לא היה קל בכלל , הוא דרש הרבה השקעה והרבה זמן ובמיוחד שהאלגוריתמים שדרושים בתרגיל הם לא קלים יחסית ואפילו אין עליהם הרבה חומר באינטרנט , אבל אחרי הרבה השקעה ואינסוף חיפושים ב GOOGLE הצלחנו סוף סוף להתגבר על התרגיל ולאתגר את עצמנו.

אנחנו מצפים אחרי הניסוי למצוא קשר בין דרגת צפיפות הגרף לבין הזמן שבו מוצאת התוכנית את צביעה מינימלית.

הבעיה היא בעיה NP קשה.

אין פתרון דטרמיניסטי ואופטימלי שפותר את הבעיה הזאת בזמן פחות מאקפוננציאלי מאורך הקלט.

ייצגנו את הגרף כמטריצת שכיוניות דו ממדית (Adjacency Matrix).

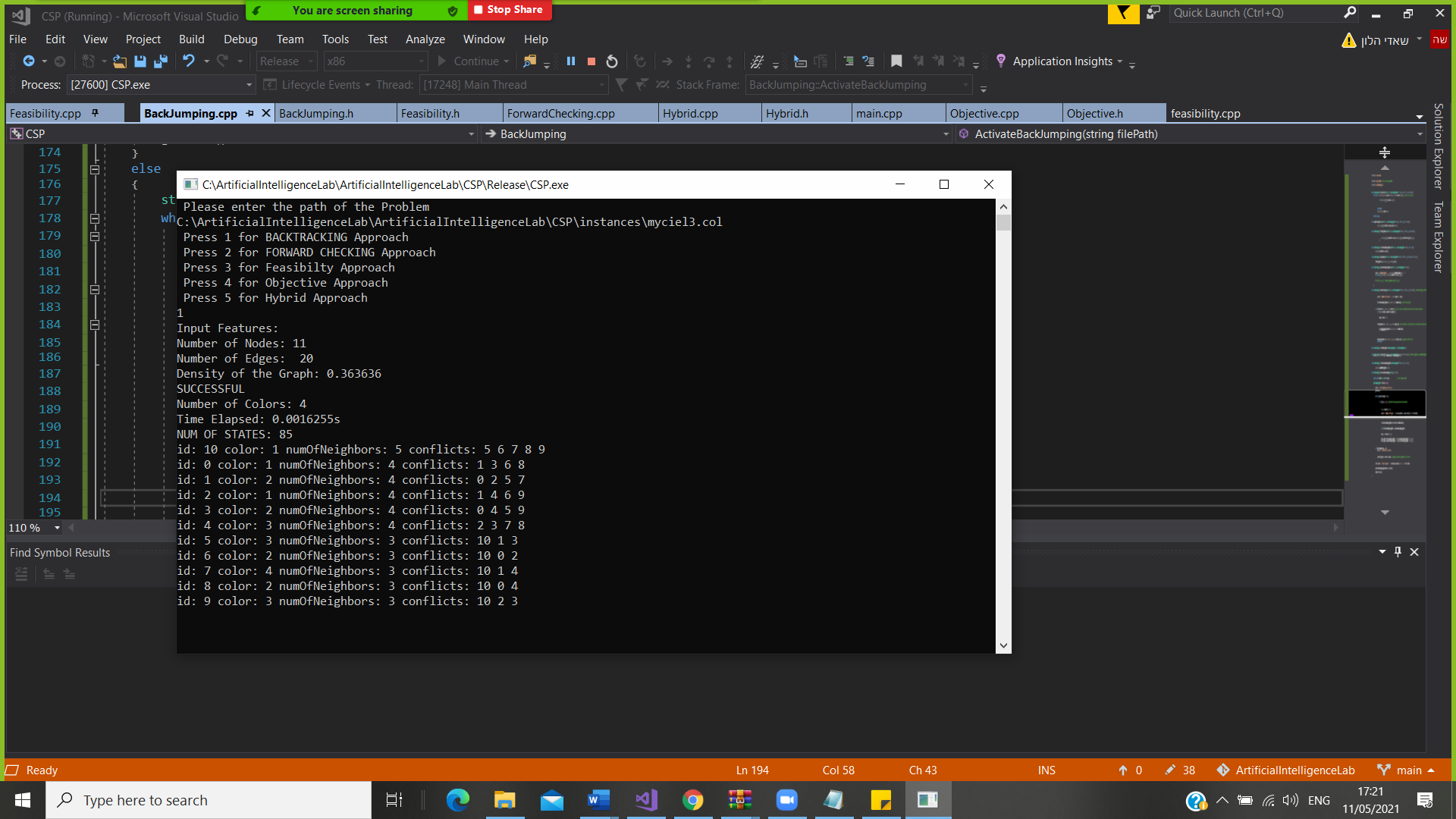
עבור החלק הראשון כדי להפעיל את התוכנה יש להריץ את ה EXE FILE , להכניס את ה-PATH של הקלט ואז לבחור את אחת הגישות שהיה צריך לממש.

**חלק א':**

1. עבור ה PATH הזה לדוגמא :

C:\ArtificialIntelligenceLab\ArtificialIntelligenceLab\CSP\instances\myciel3.col

המאפיינים מופיעים תחת הכותרת INPUT FEATURES :

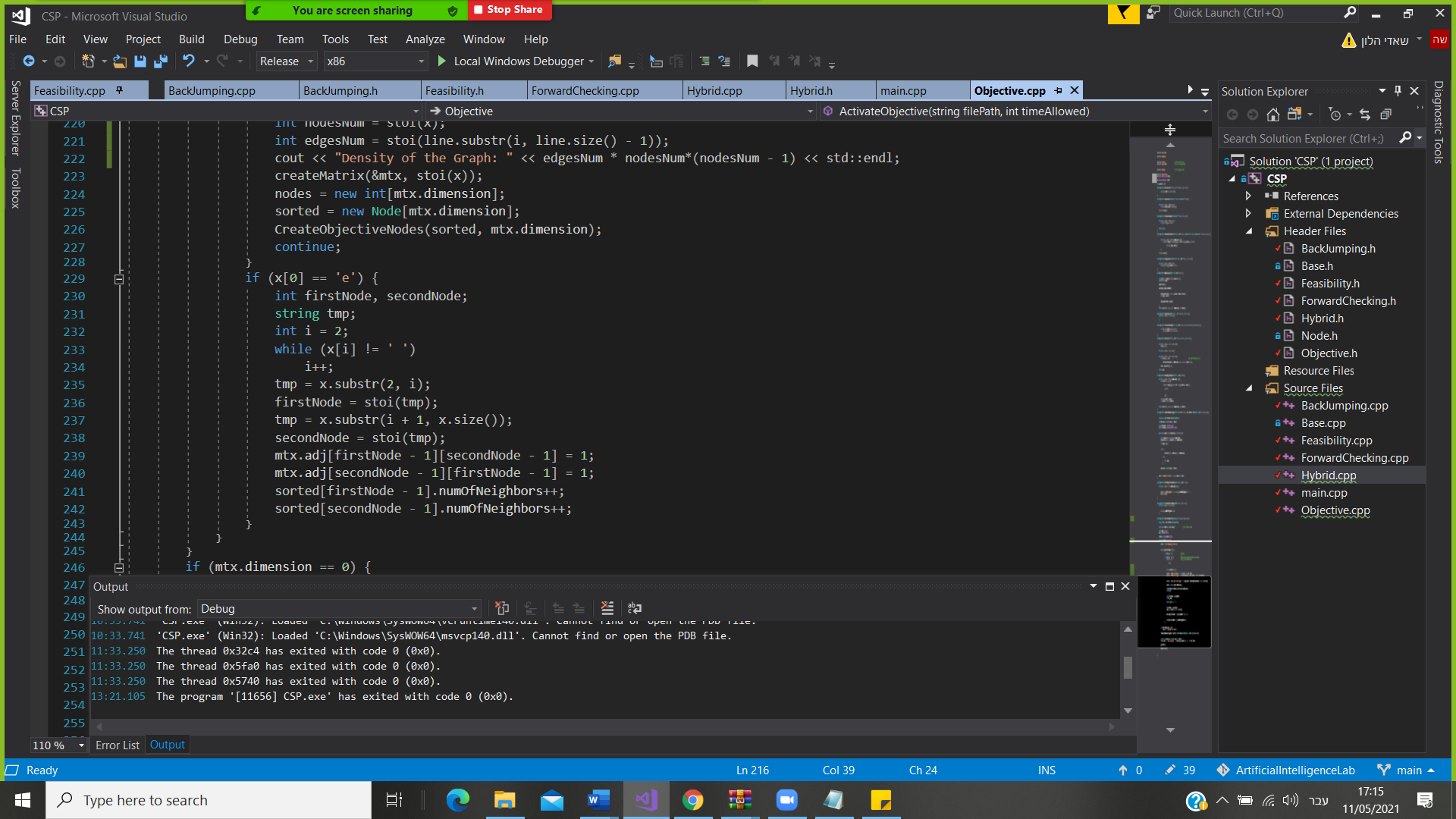


קריאת הקלט :

הקטע הזה מטפל בצמתים



הקטע הזה מטפל ב EDGES:



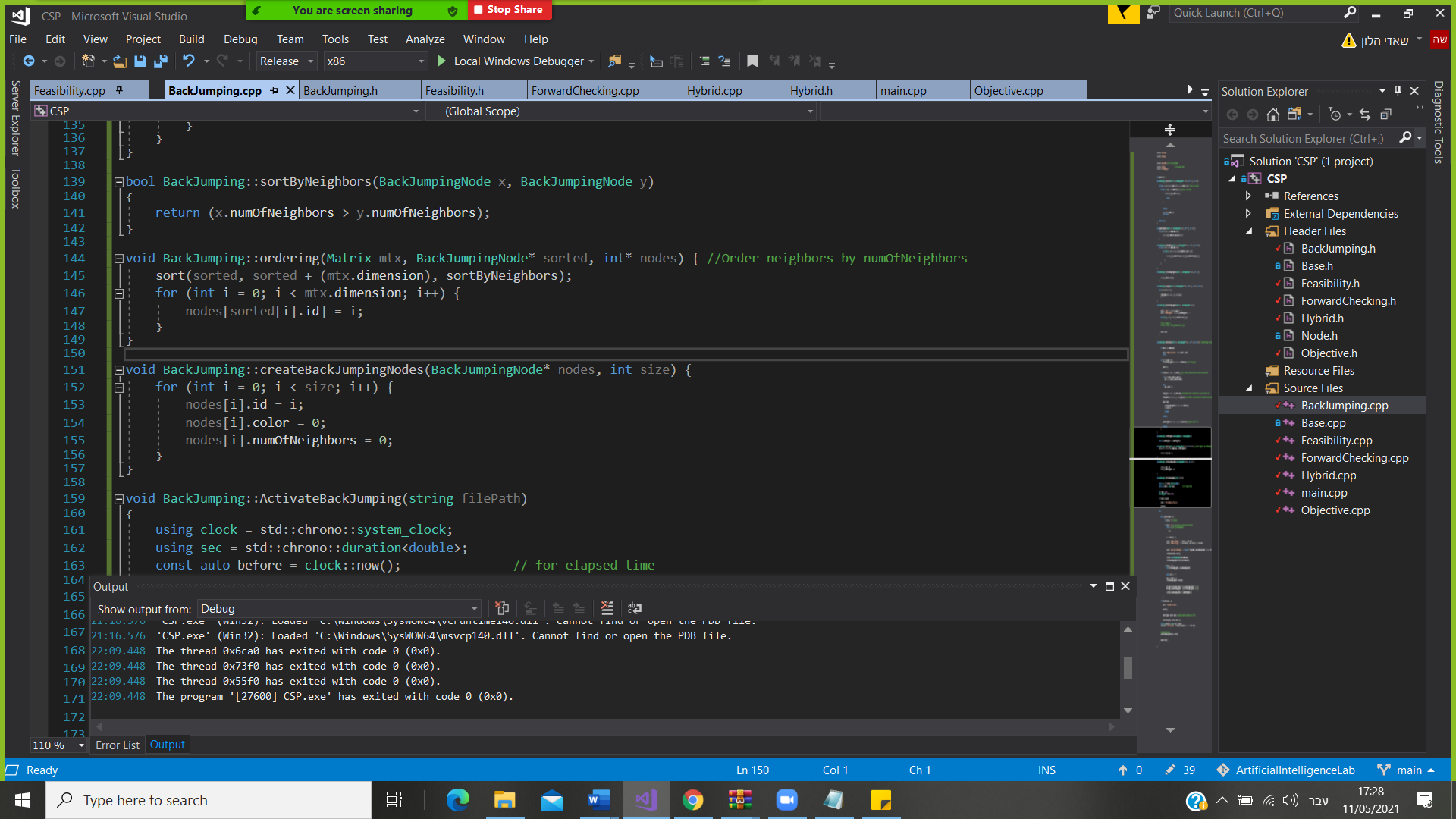
**חלק ב':**

**a.**

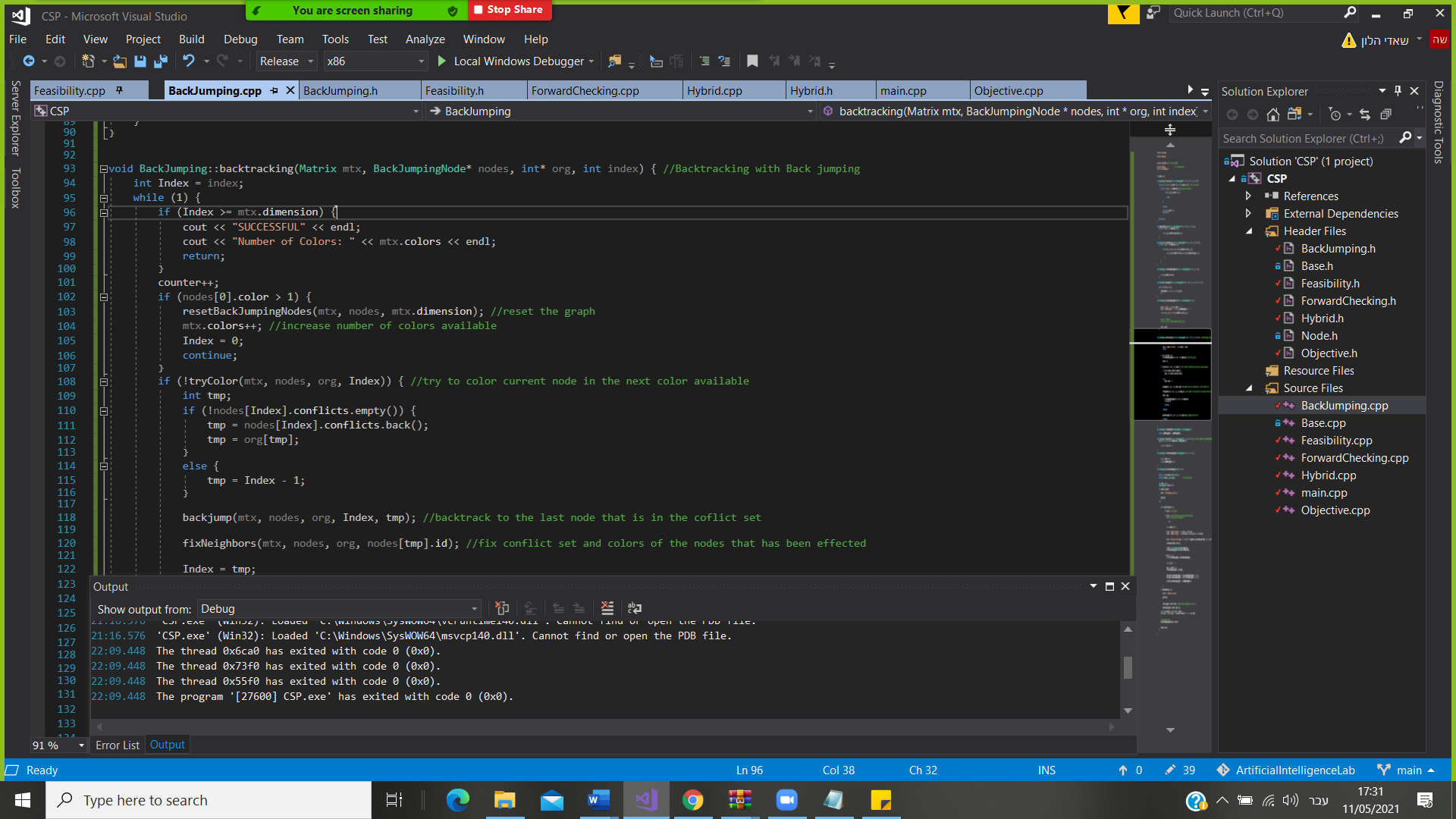
***Backtracking with Back jumping***

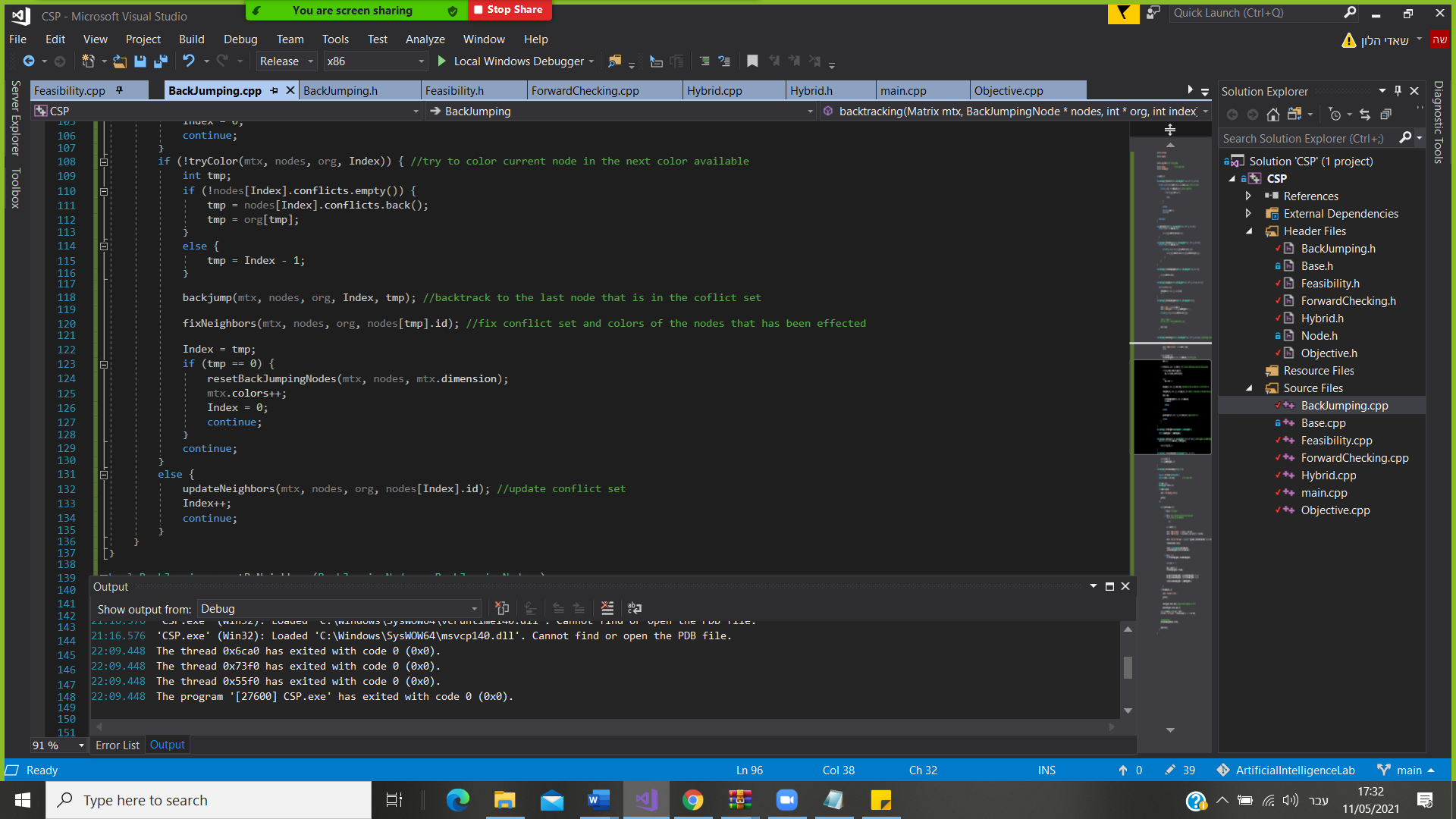
* התחלנו לצבוע צמתים לפי מספר השכנים (סדר יורד)

Largest Degree Ordering (LDO): It chooses a vertex with the highest number of neighbors

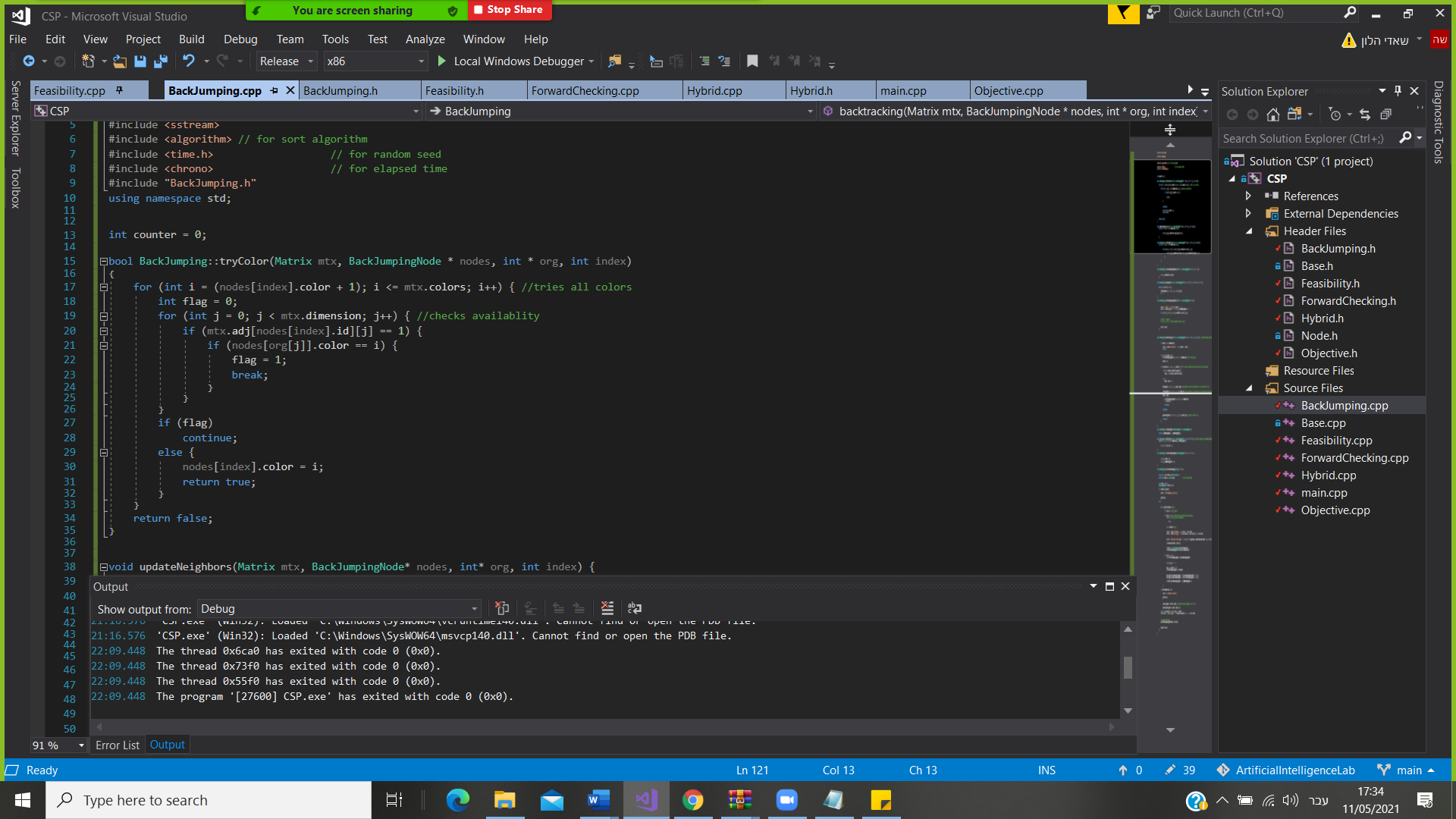


* האלגוריתם בכללי:

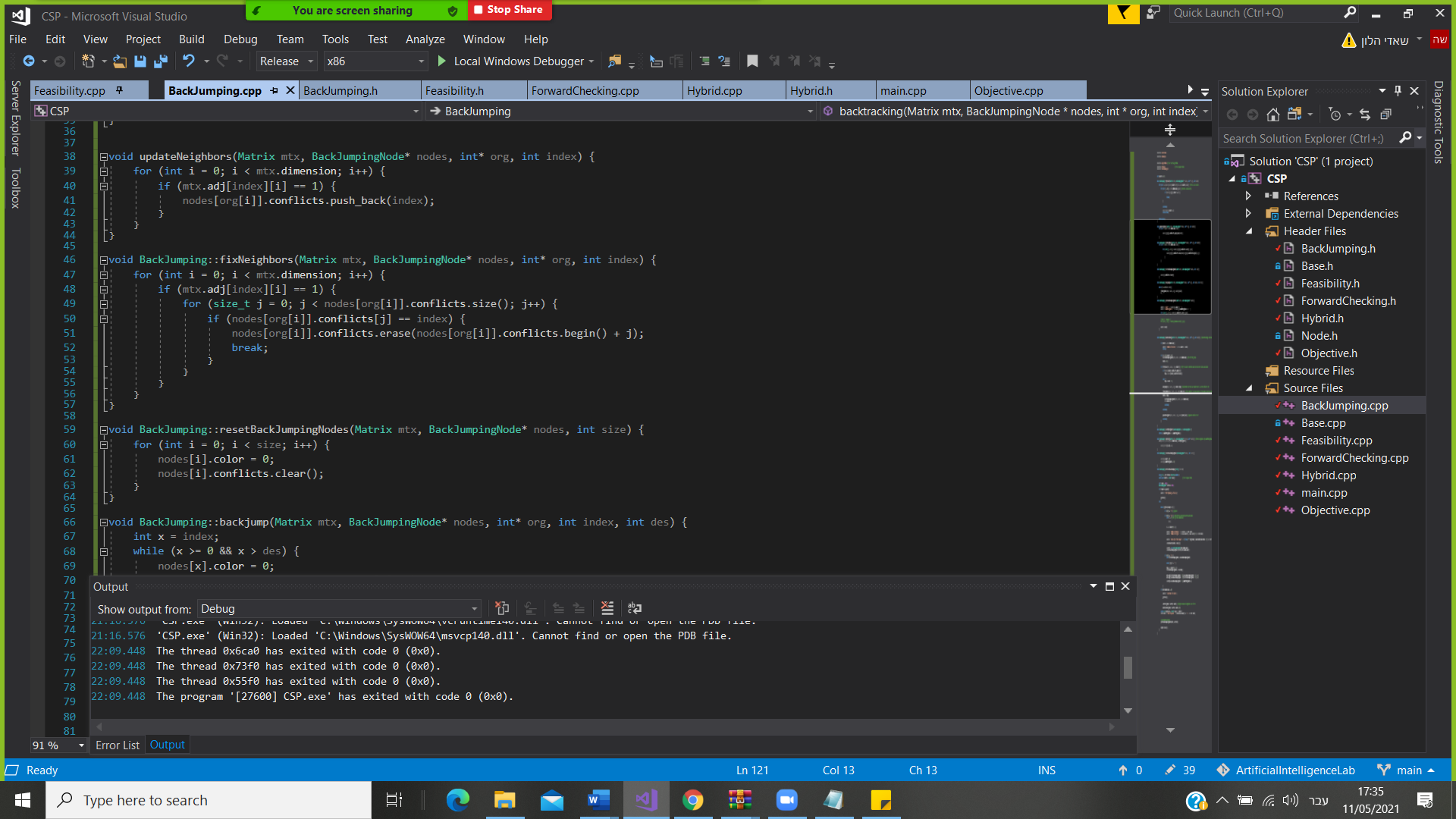




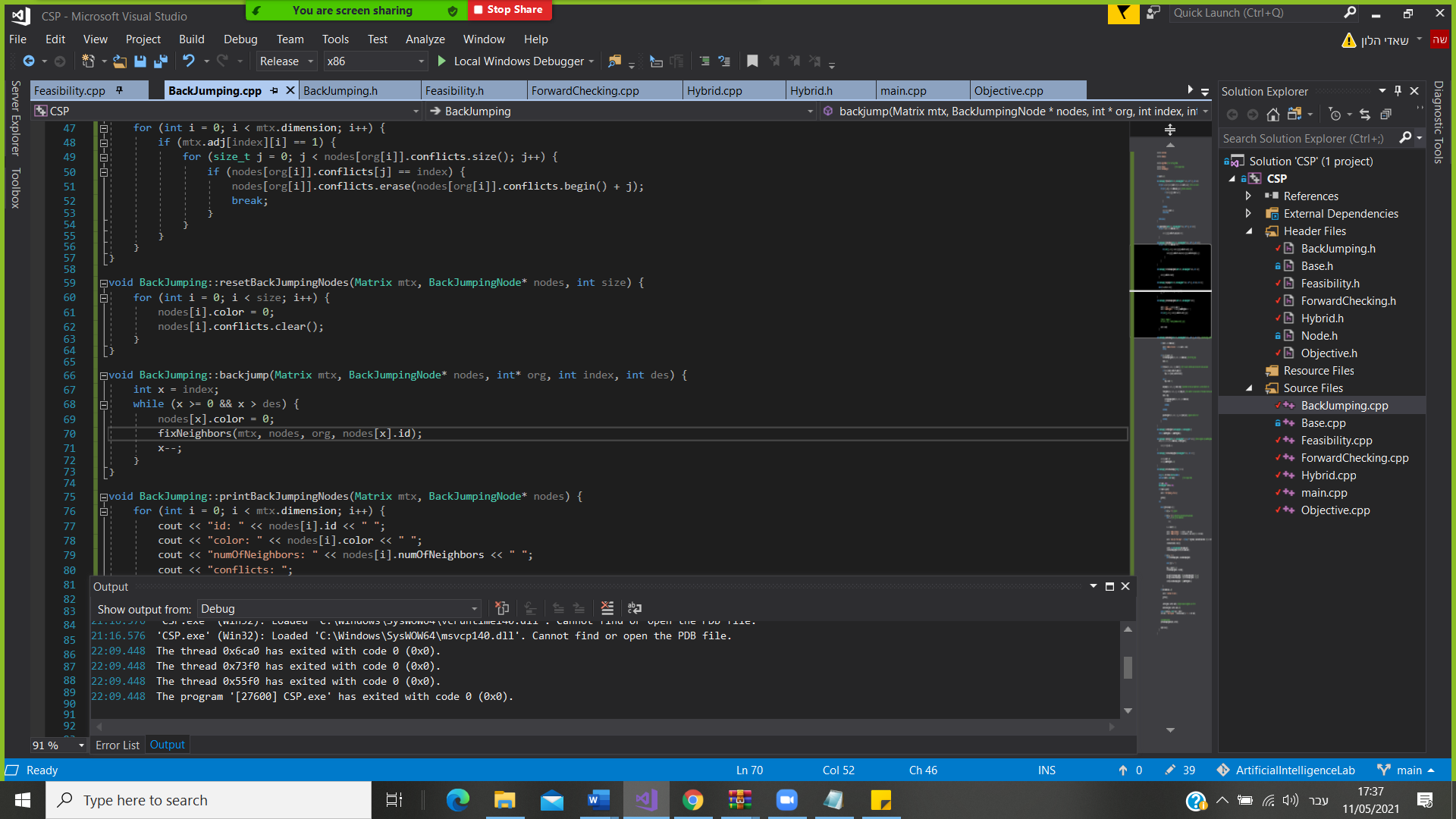
* הפונקציה המנסה לצבוע צומת בודד.



* fixNeighbors: בשלב ה-backtracking מוחקים את הצומת שאנחנו חוזרים ממנו מה-conflict set של שכניו.



* backjump: הפונקציה החוזרת למצב קודם וקוראת לfixNeighbor כדי לשחזר את מצב הגרף קודם.



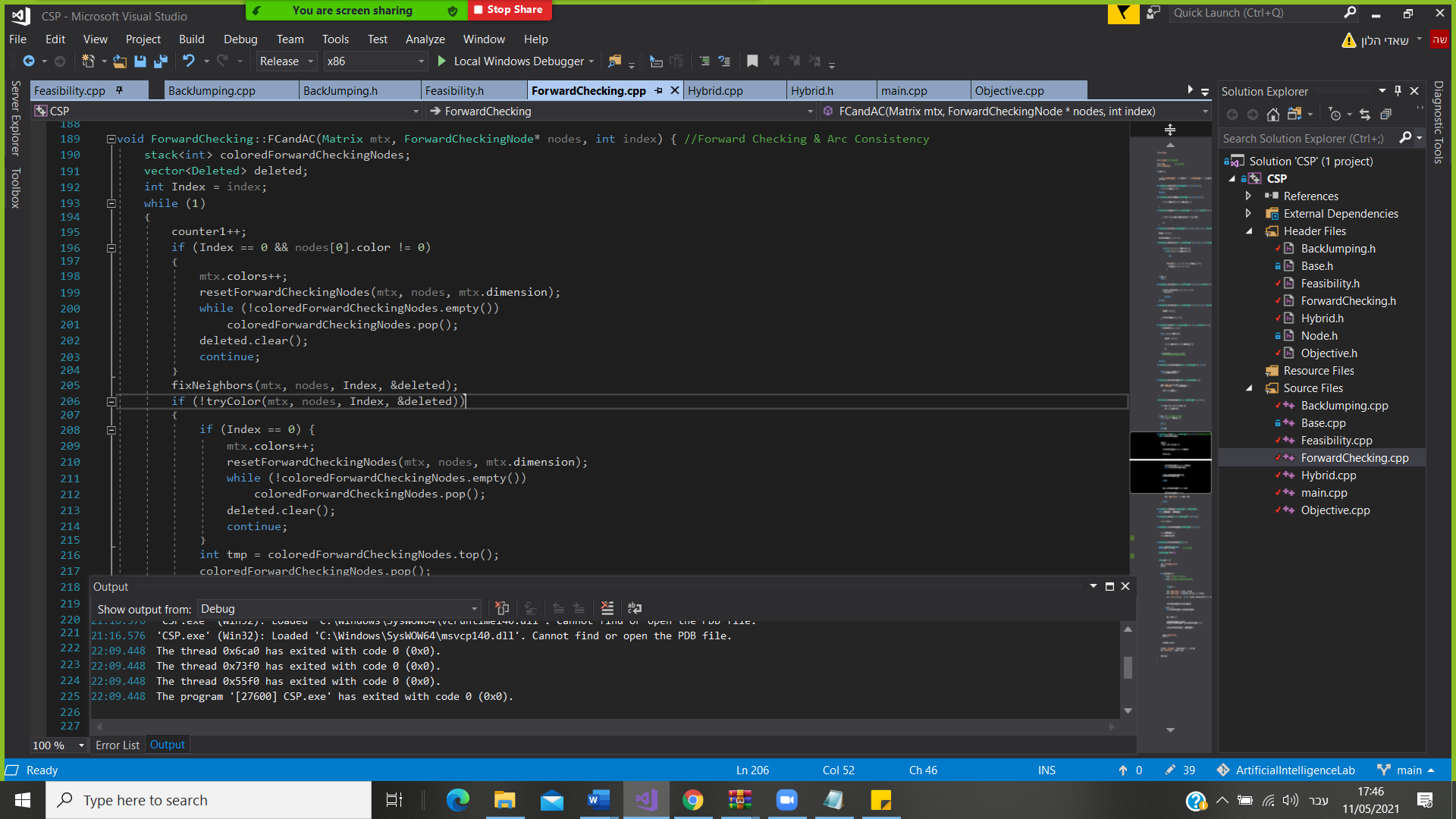
**b.**

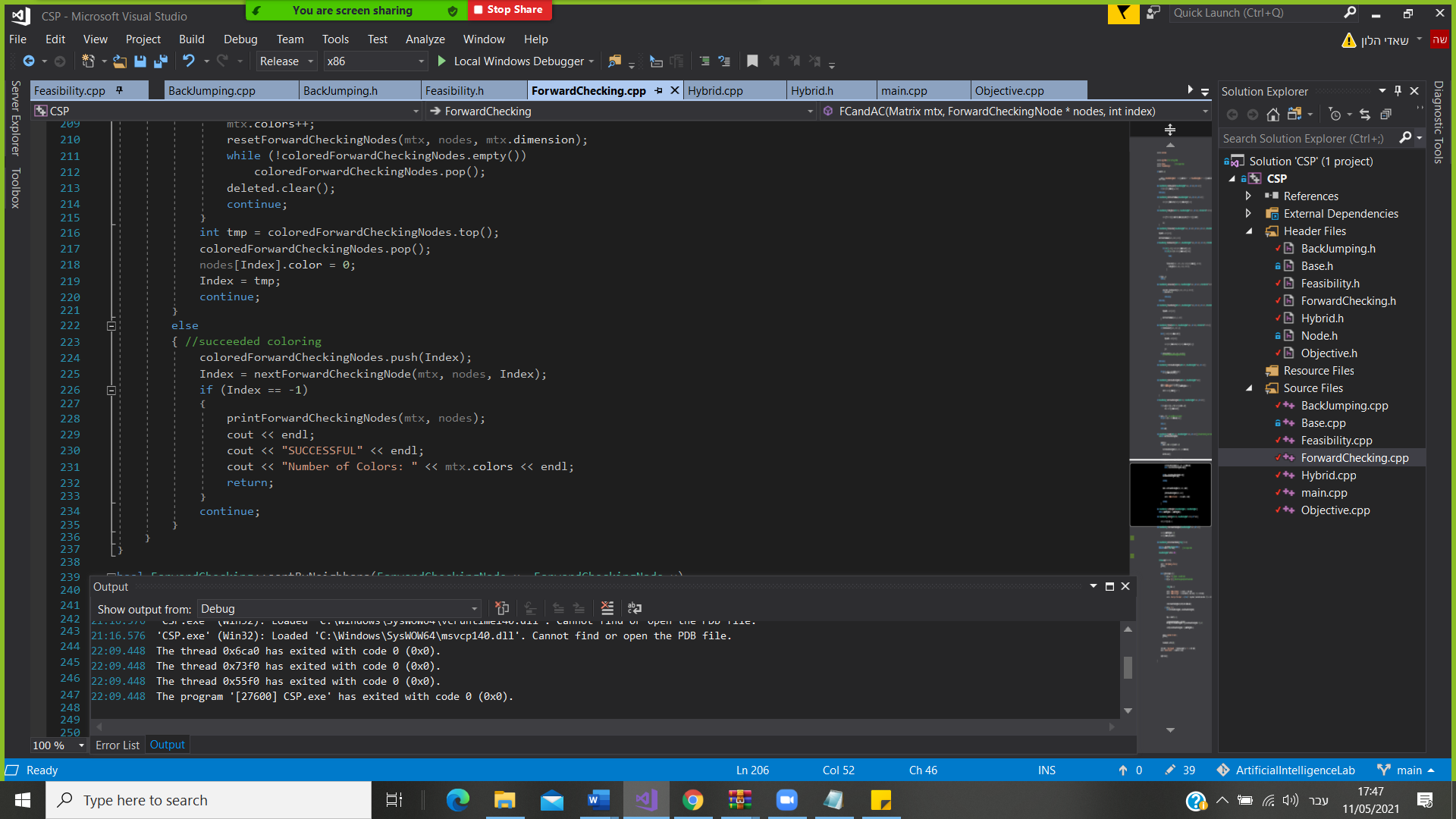
***Forward Checking & Arc Consistency***

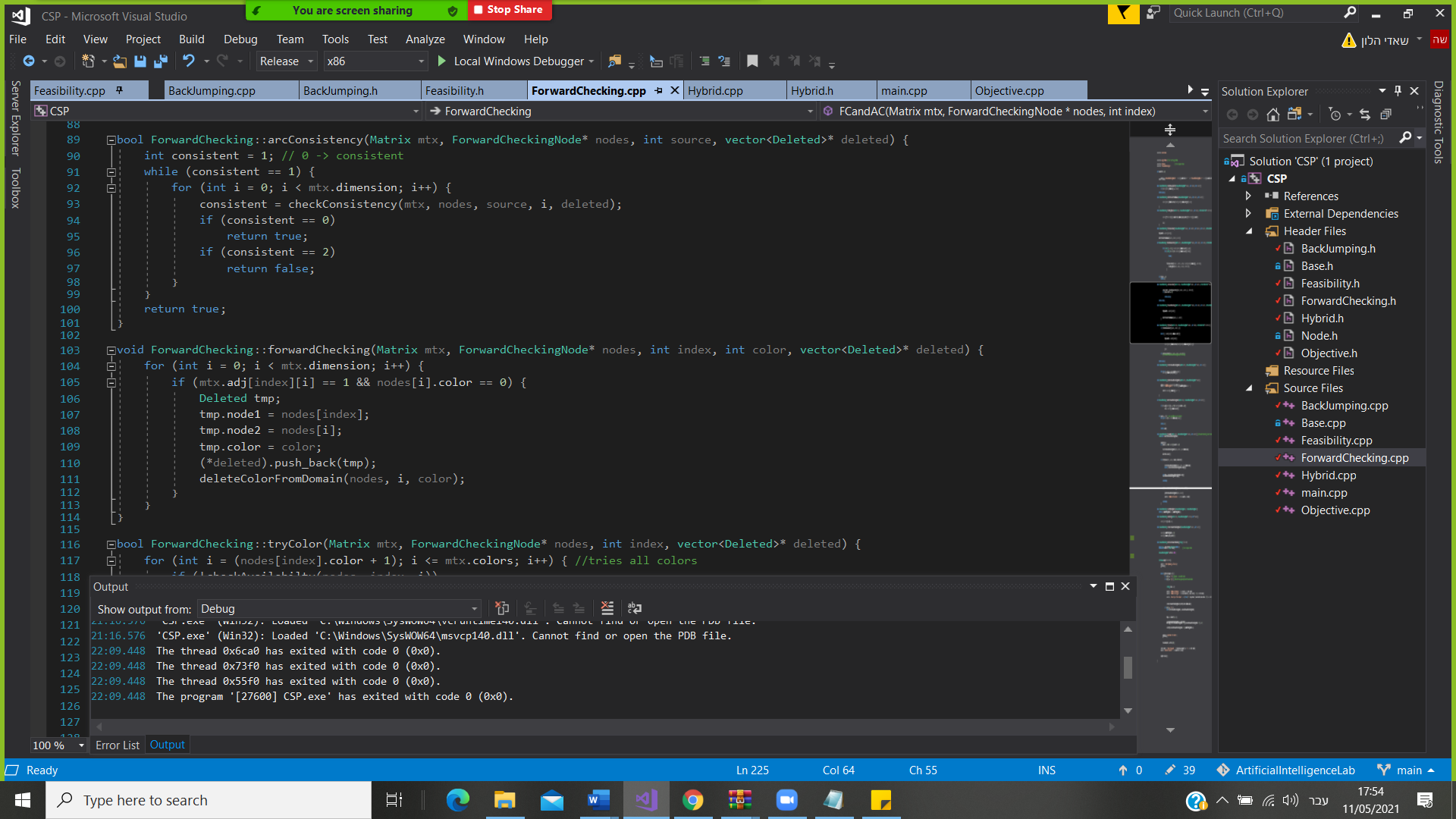
* האלגוריתם :

coloredNodes מכילה את כל הצמתים שכבר צבענו.  
deleted הוא vector שמכיל Deleted שמייצג את כל השינויים שנעשו

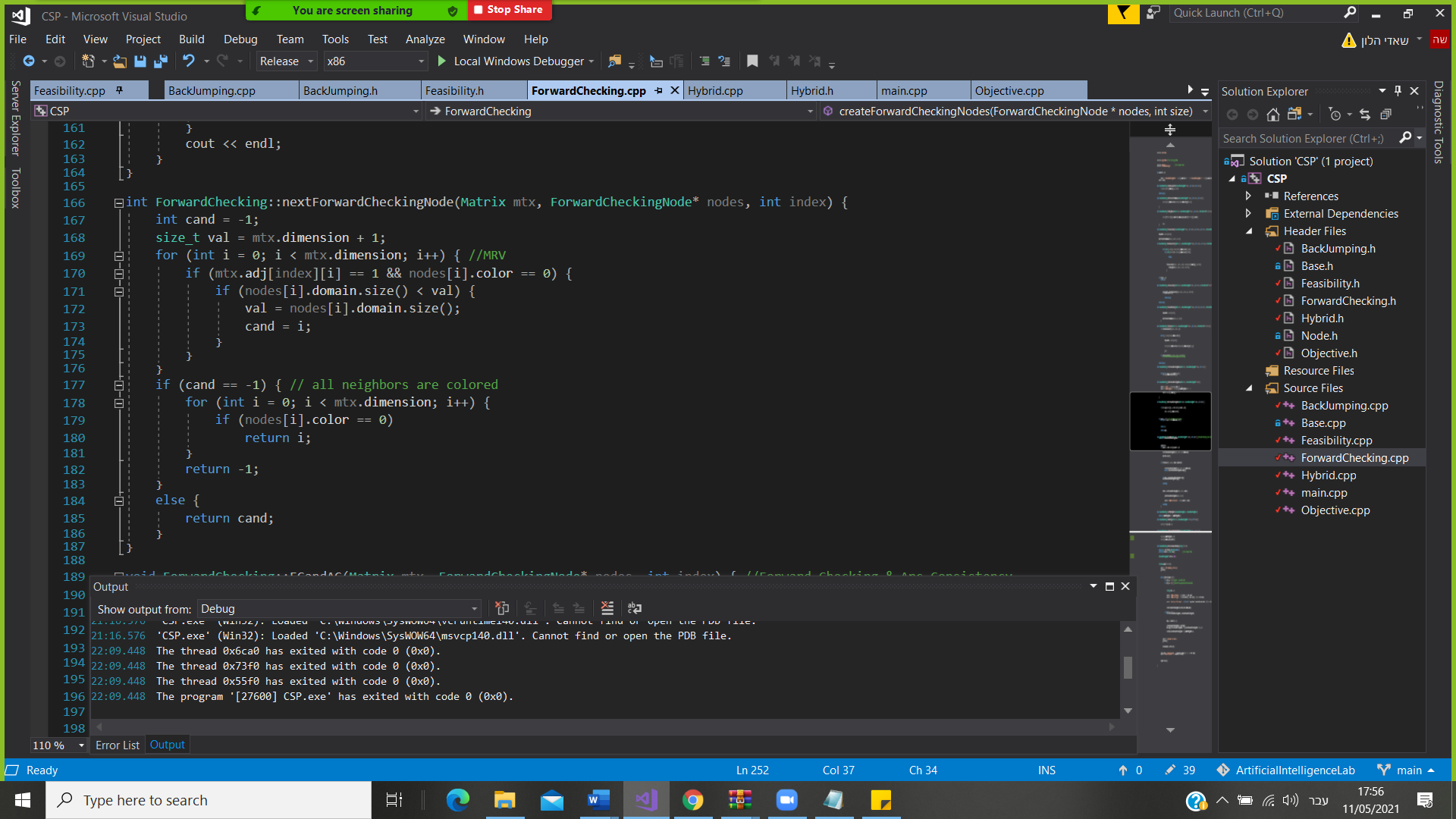
בגרף. Deleted מכל הצומת שצובעים אותו עכשיו, צומת שה-domain שלו השתנה ואת הצבע.







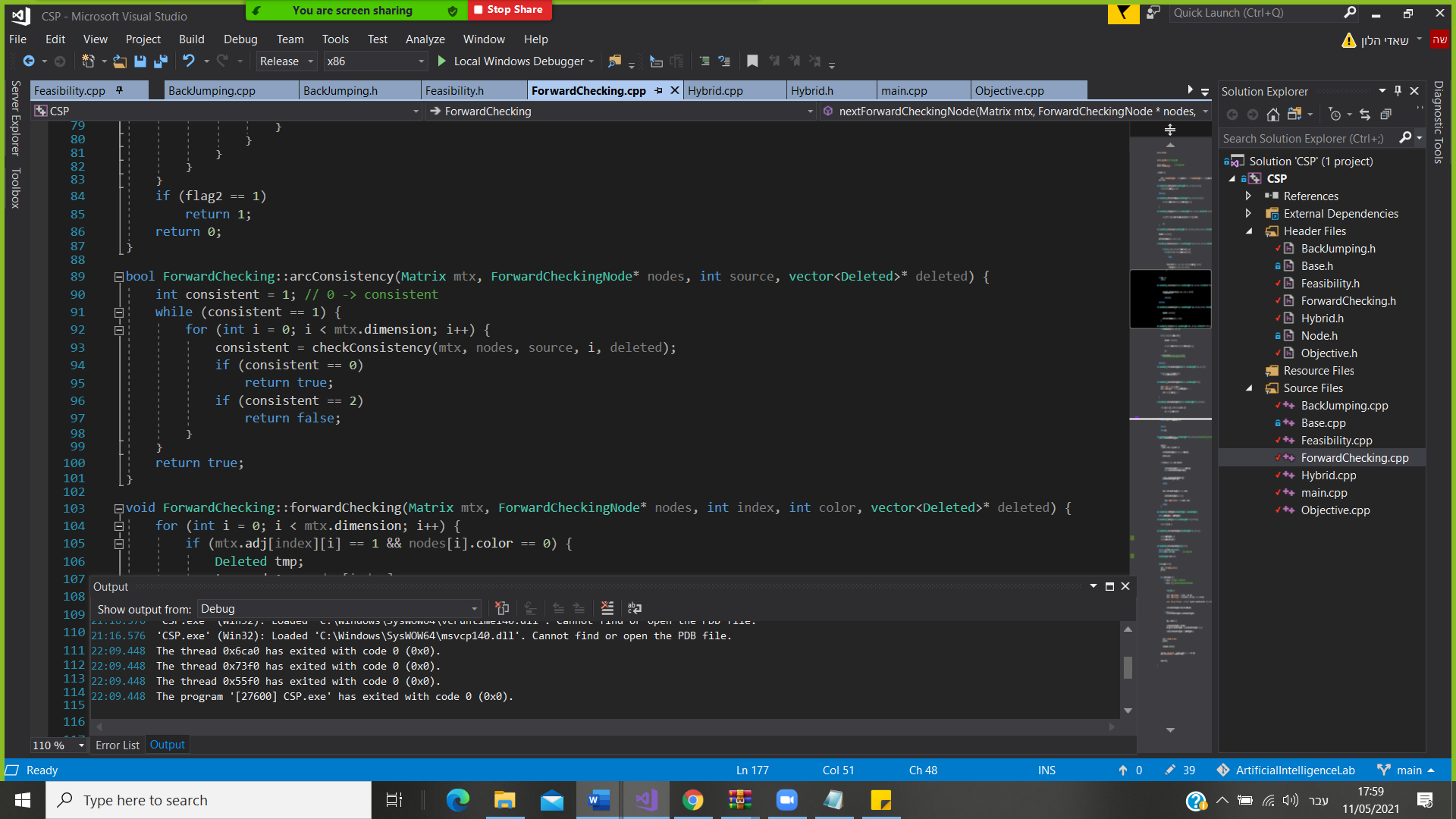
* בחירת הצומת הבא לצביעה:  
  השתמשנו בהיוריסטיקת הבחירה MRV (Most Restricted Variable)



* Arc Consistency:

אם consistent=0 אז קיבלנו גרף קונסיסטנטי

אם consistent=2 אז קיבלנו גרף שה-domain שלו ריק לכן לא קונסיסטנטי (וצריך לבצע backtracking)



אנחנו מצפים אחרי הניסוי למצוא קשר בין דרגת צפיפות הגרף לבין הזמן שבו מוצאת התוכנית את הפתרון.

ביצוע האלגוריתם Backtracking with Back jumping:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| #Colors | Elapsed time(s) | |States| | |V| | |E| | Test name |
| 4 | 0.00374 | 85 | 11 | 20 | myciel3.col |
| 5 | 0.0048218 | 1424 | 23 | 71 | myciel4.col |
| 6 | 0.218316 | 707525 | 47 | 236 | myciel5.col |
| 7 | 0.075347 | 35694 | 95 | 755 | myciel6.col |
| 8 | 0.0256729 | 12582 | 128 | 774 | miles250.col |
| 5 | 0.0748707 | 110 | 25 | 320 | queen5\_5.col |
| 7 | 0.700396 | 463420 | 36 | 580 | queen6\_6.col |
| 8 | 13.678 | 8373327 | 49 | 952 | queen7\_7.col |

ביצוע האלגוריתם Backtracking with Forward checking

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| #Colors | Elapsed Time(s) | |States| | |V| | |E| | Test name |
| 4 | 0.0216522 | 34 | 11 | 20 | myciel3.col |
| 5 | 0.064918 | 79 | 23 | 71 | myciel4.col |
| 6 | 0.242285 | 144 | 47 | 236 | myciel5.col |
| 7 | 0.194444 | 247 | 95 | 755 | myciel6.col |
| 8 | 0.597428 | 271 | 128 | 774 | miles250.col |
| 5 | 0.0565123 | 41 | 25 | 320 | queen5\_5.col |
| 9 | 0.0841219 | 214 | 36 | 580 | queen6\_6.col |
| 10 | 0.230837 | 282 | 49 | 952 | queen7\_7.col |

גרף השוואה בזמני ביצוע בין שני האלגורתמים:

אפשר לראות משני הגרפים הנ״ל שעבור קלטים קטנים Back Jumping מוציא פתרון בזמן פחות יותר מזמן מציאת הפתרון כשמשתמשים ב Forward Checking & Arc Consistency.

זה נובע מהסיבה שבגרפים קטנים יחסית מספר ה-Backtracking סביר והוא לא "מבזבז" זמן על בדיקות קדימה.

אך לעומת זאת בגרפים גדולים יש הפרש עצום בין זמן ביצוע אלגוריתם Back jumping לבין Forward Checking, וזה נובע מהעובדה   
ש-Backjumping עובד בלי בדיקות קדימה ובמקרה של תקלה חוזר ומנסה לסדר את הבעיה ובכך הוא מגדיל את זמן העבודה שלו, אלה ה Forward Checking & Arc Consistency פוסל וחוסך אפשרויות.

גרף השוואה במספר STATES שנסרקו בשני האלגורתמים:

אלגוריתם Back jumping מסתמך על זה שהוא בודק כל האפשריות ובמקרה של בעיה חוזר אחורה לעומת אלגוריתם  
 Forward checking & Arc Consistency בודק אחרי כל מצב איזה מצבים לא חוקיים נוספו ופוסל אותם ואז ממשיך בדרך הצביעה, ומפה רואים שפעולתו העיקרית לפני ההתקדמות היא לפסול מקרים לא חוקיים, כל הנ״ל מתבטא בגרף שיצרנו ובכך שבכל הטסטים שנבדקו מספר הצמתים שעוברים עליהם ב Back jumping הרבה יותר גדול ממספר הצמתים שעוברים עליהם ב Forward checking.

עבור החלק השני כדי להפעיל את התוכנה יש להריץ את ה EXE FILE , להכניס את ה-PATH של הקלט ,לבחור את אחת הגישות שהיה צריך לממש ואז להכניס את הזמן המוקצה לההרצה.

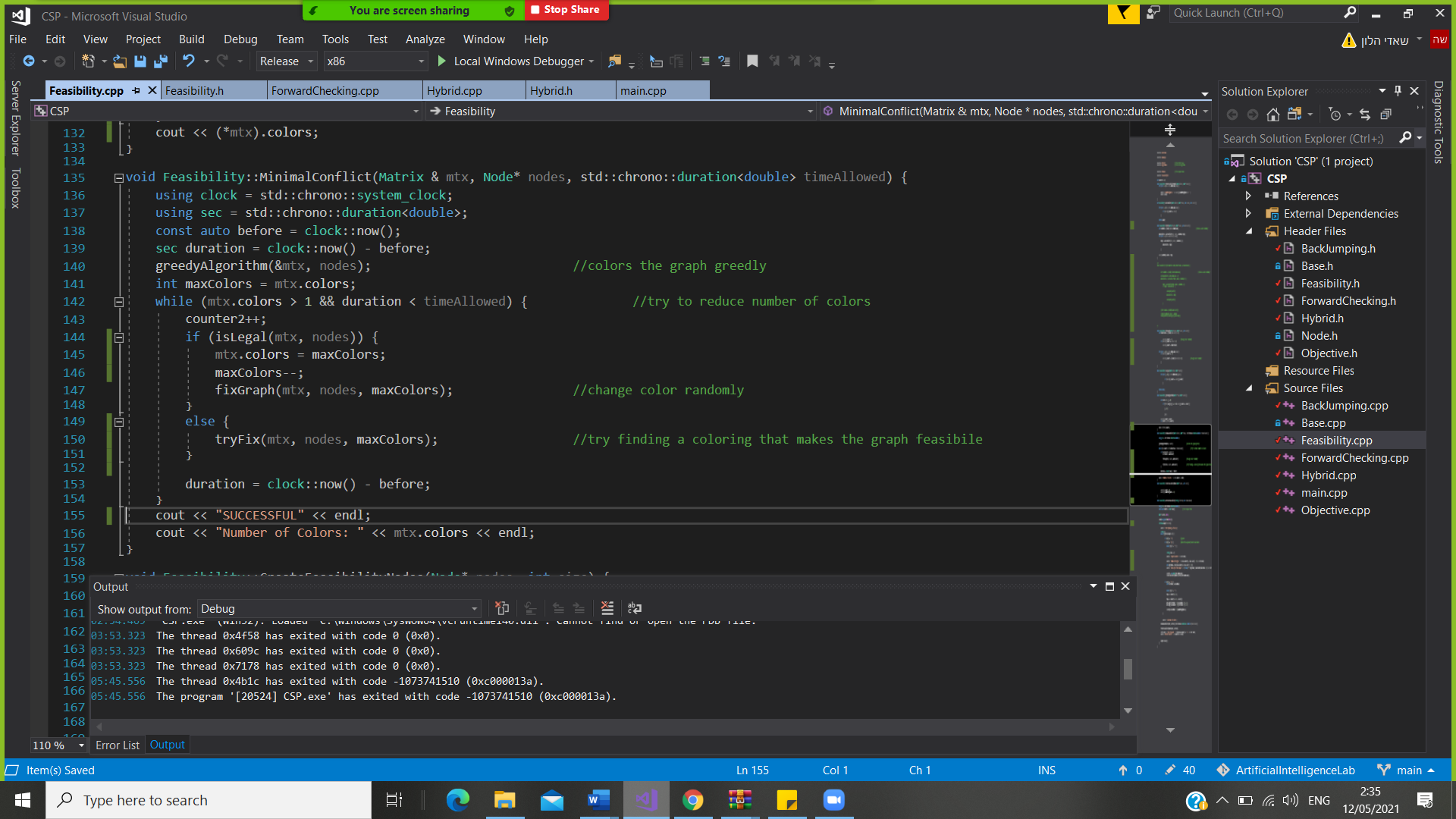
**חלק ג' + ד':**

בחלק זה נעבוד על השוואה בין ביצועי 3 גישות לפתירת בעיית צביעת גרף שלושת הגישות בחלק זה הן מתאימות לשיטת החיפוש הלוקאלי, הגישה הראשונה היא **הגישה הפיזבילית** שמימשנו בעזרת אלגוריתם החיפוש הלוקאלי**MIN CONFLICT** ,הגישה השנייה היא **פונקציית המטרה** שבעזרת אלגוריתם Simulated Annealing ו-Kempe Chain דאגנו למקסם כך ש הוא מספר הצבעים ו- הוא מספר הצמתים הצבועים ב- , הגישה השלישית היא **הגישה ההיברידית** שבעזרת אלגוריתם החיפוש **GENETIC ALGORITHM** דאגנו להקטין  
   
כך ש הוא מספר הצבעים, הוא מספר הצמתים הצבועים ב- ו-  
 הוא מספר ה-Bad Edges שקצותיו צבועים ב-.

a.

***Min Conflicts Algorithm***

* האלגוריתם:

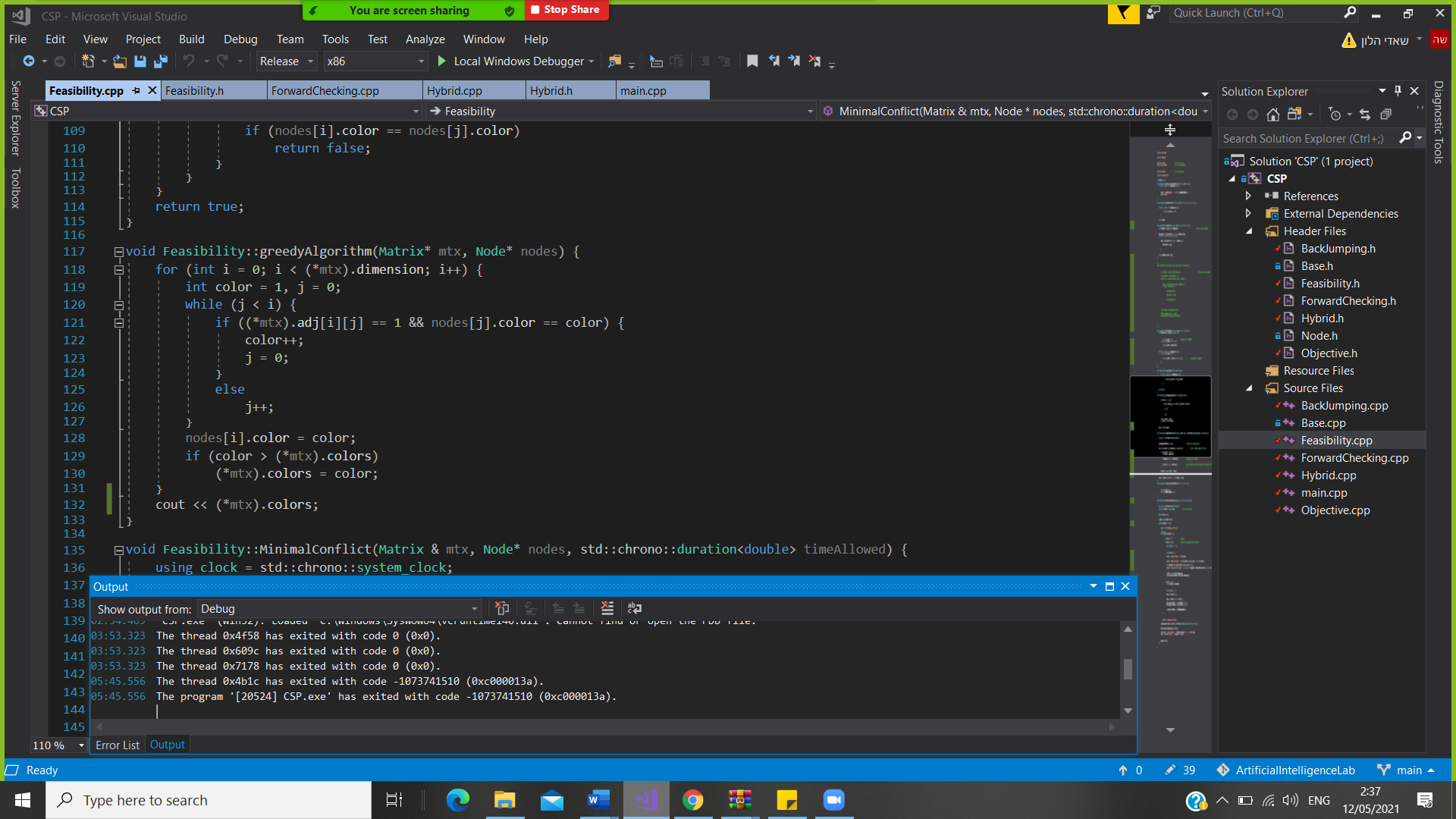


מתחילים בצביעת הגרף בצורה חמדנית (greedy) שזמן ריצתו פולינומי באורך הקלט וכמובן הגרף שנוצר הוא גרף חוקי.

בוחרים צבע אקראי שרוצים למחוק אותו וצובעים את כל הצמתים שצבועים בצבע זה בצבע אחר.

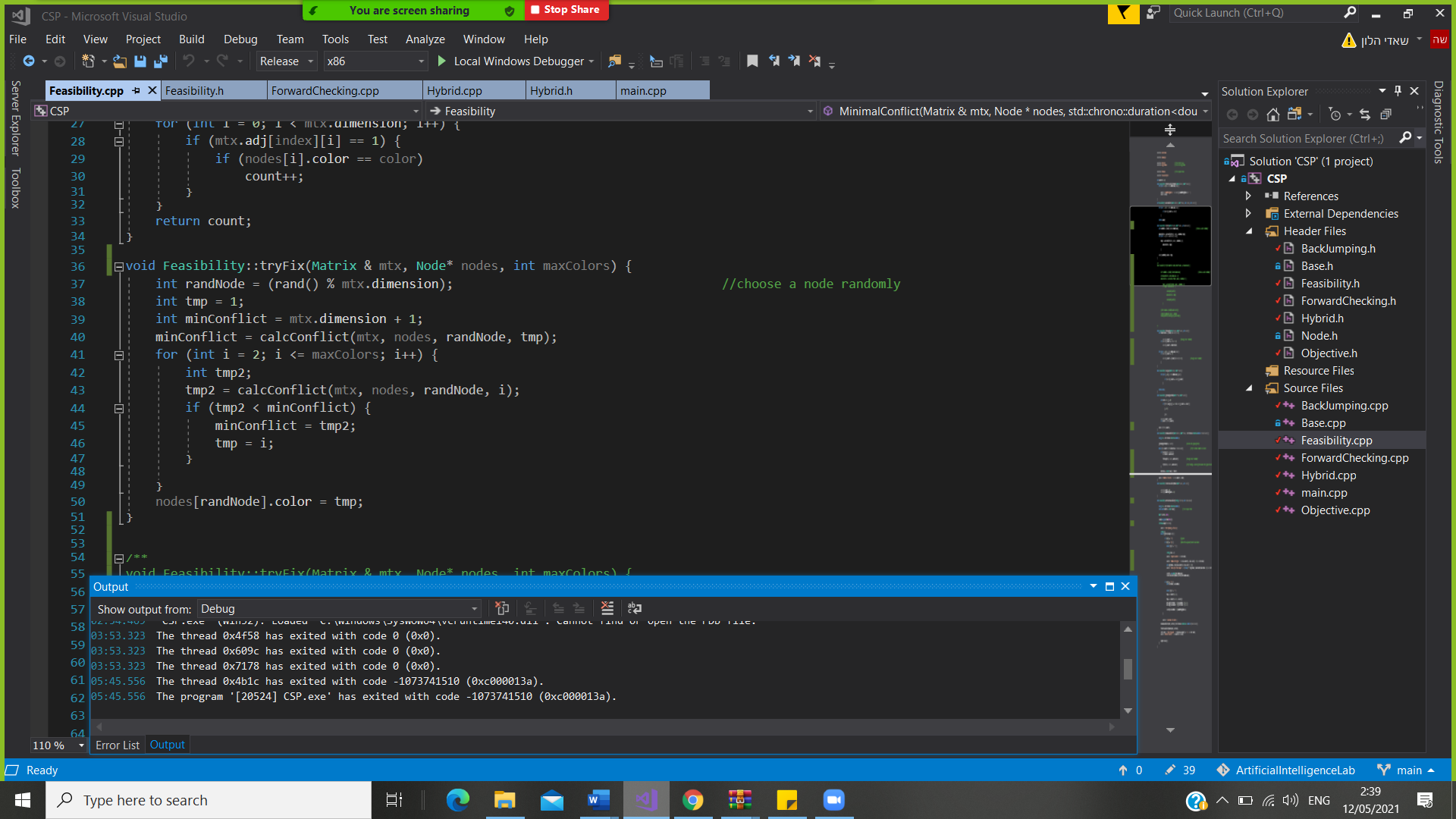
המטרה שלנו להפוך את הגרף שלנו לחוקי וזאת הגישה הפיזבילית.

* הצביעה החמדני:



* הפונקציה המנסה להפוך את הגרף לפיזבילי:

על פעם היא בוחרת צומת אקראי ובודקת מהו הצבע "הטוב" ביותר כלומר הצבע שעבורו מספר השכנים הצבועים באותו צבע הוא קטן ביותר  
 (min-conflict) וצובעת אותו בצבע זה.



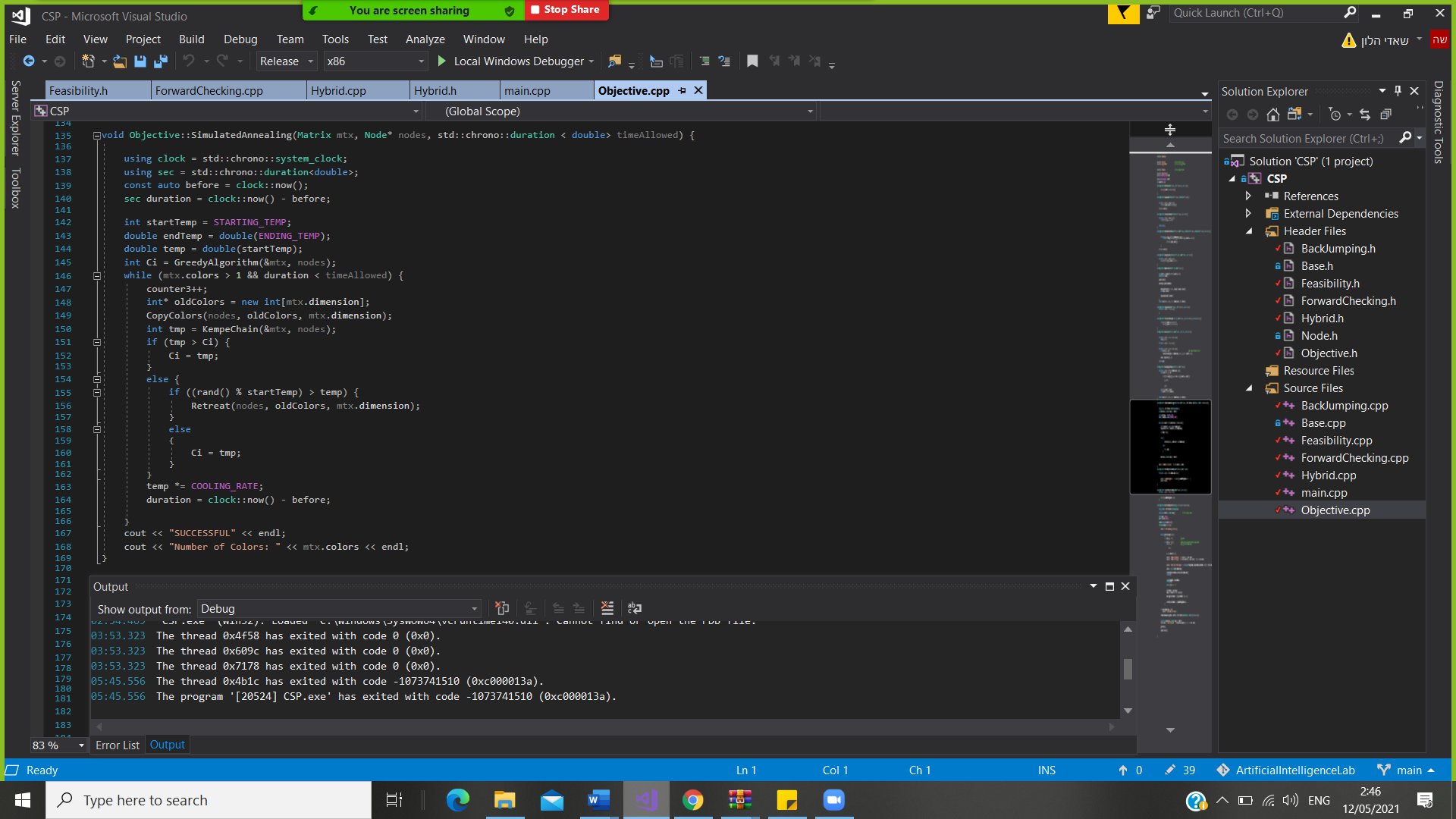
***Simulated Annealing***

***Kempe Chain***

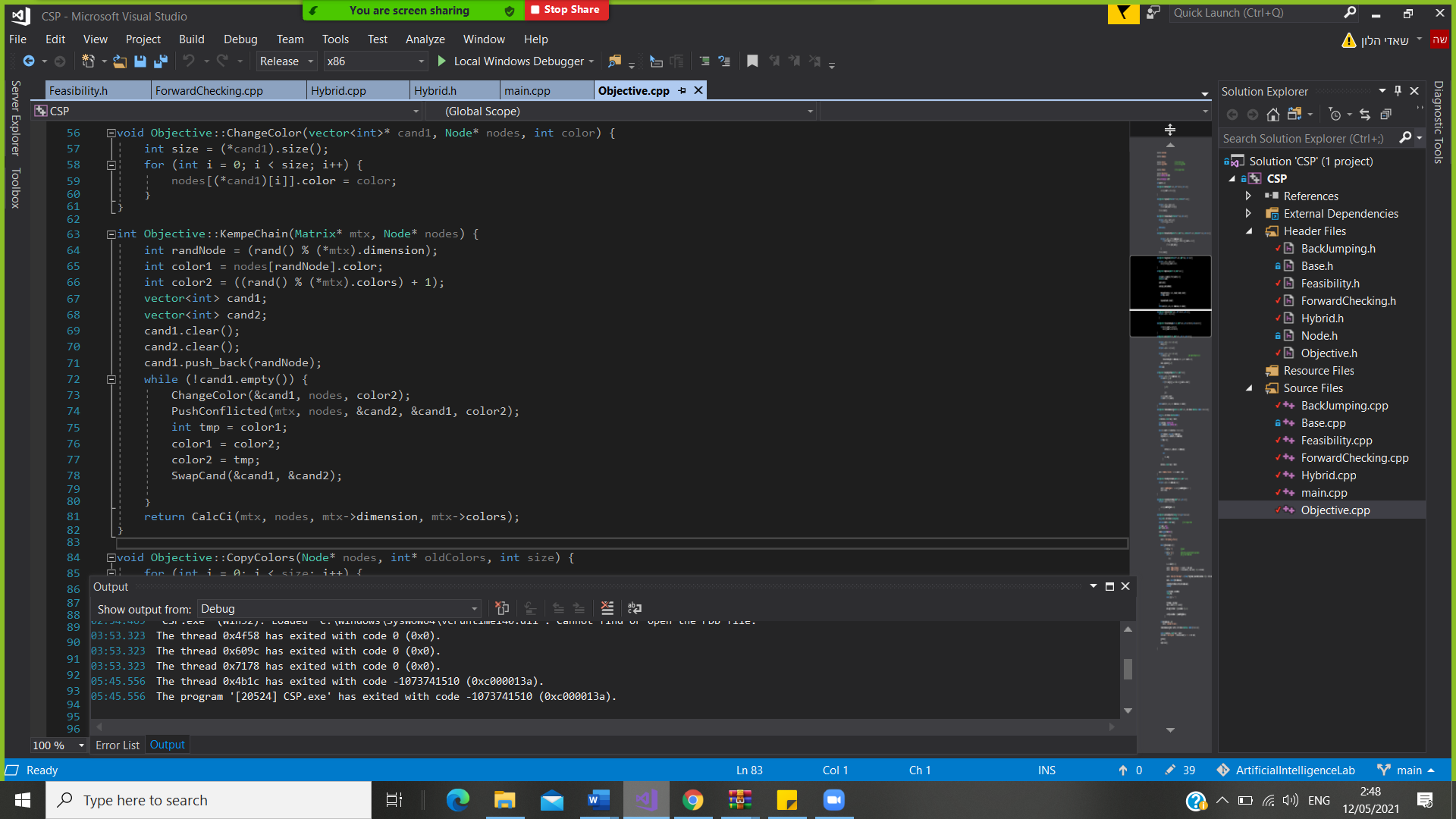
***הגישה ששמה פונקציית המטרה במרכזה***

הגישה הזאת מטרתה למקסם כך ש הוא מספר הצבעים ו- הוא מספר הצמתים הצבועים ב- .

* Simualted Annealing:  
  מתחילים עם טמפרטורה גבוהה שמוגדרת מראש ובכל שלב הטמפרטורה יורדת בהדרגה. כאשר הטמפרטורה גבוהה אז כל שינוי בגרף נקבל אותו. כאשר הטמפרטורה פוחתת הסיכוי שגרף "לא טוב" (כלומר קיבלנו יותר גרוע) שנקבל אותו הולכת ופוחתת. אלגוריתם זה עוזר לנו לא להיתקע ב-Local Maxima



* Kempe Chain: מנסים למקסם בעזרת Kempe Chain ע"י בחירת שתי מחלקות של צבעים והחליף צבעים בין שתי המחלקות אבל חייבים לשמור על הפיזביליות כלומר שיישאר הגרף חוקי.



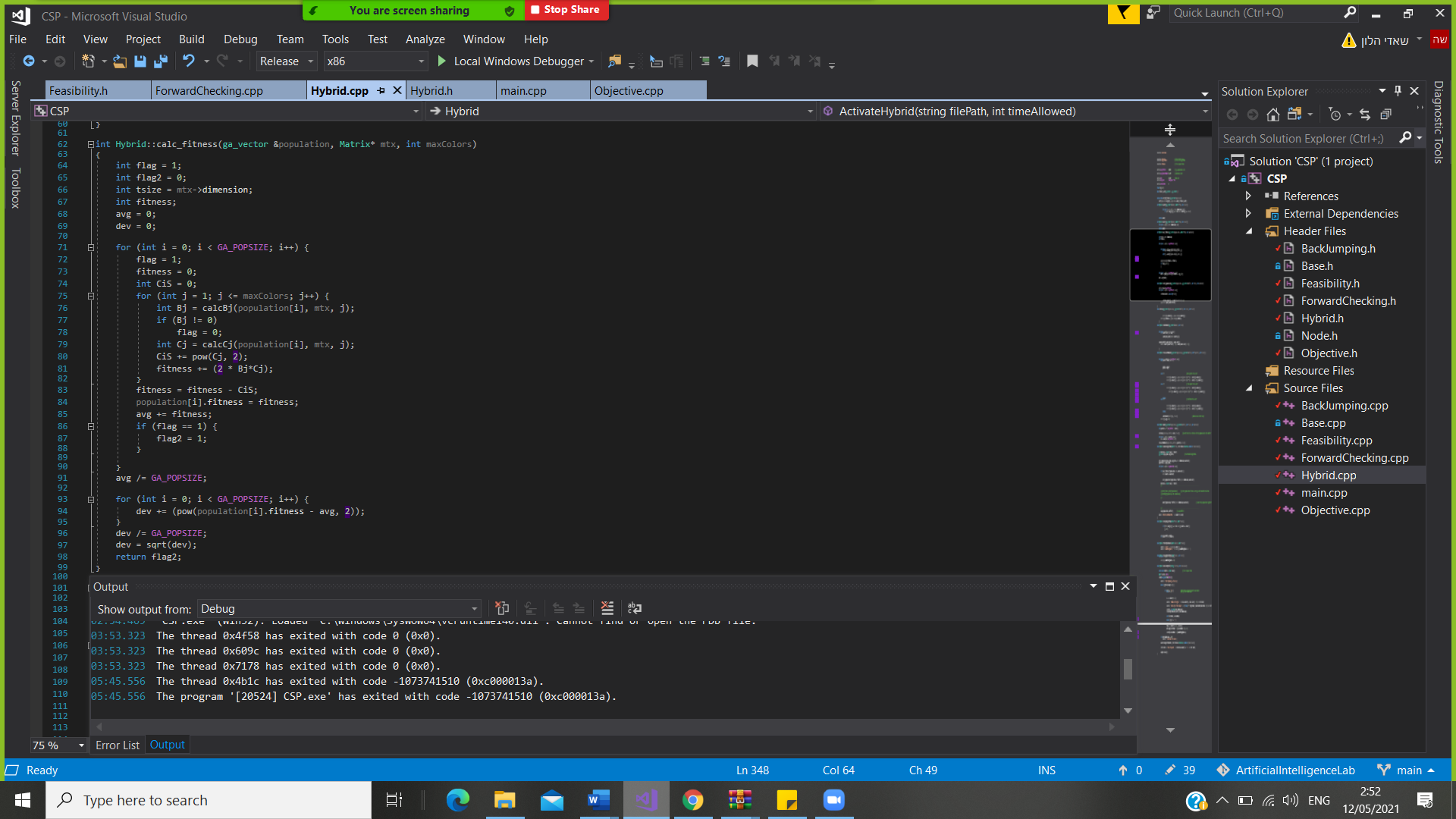
***Genetic Algorithm***

***הגישה ההיברידית***

בעזרת אלגוריתם החיפוש **GENETIC ALGORITHM** דואגים להקטין  
 (פונקציית המטרה)  
כך ש הוא מספר הצבעים, הוא מספר הצמתים הצבועים ב- ו-  
 הוא מספר ה-Bad Edges שקצותיו צבועים ב-.

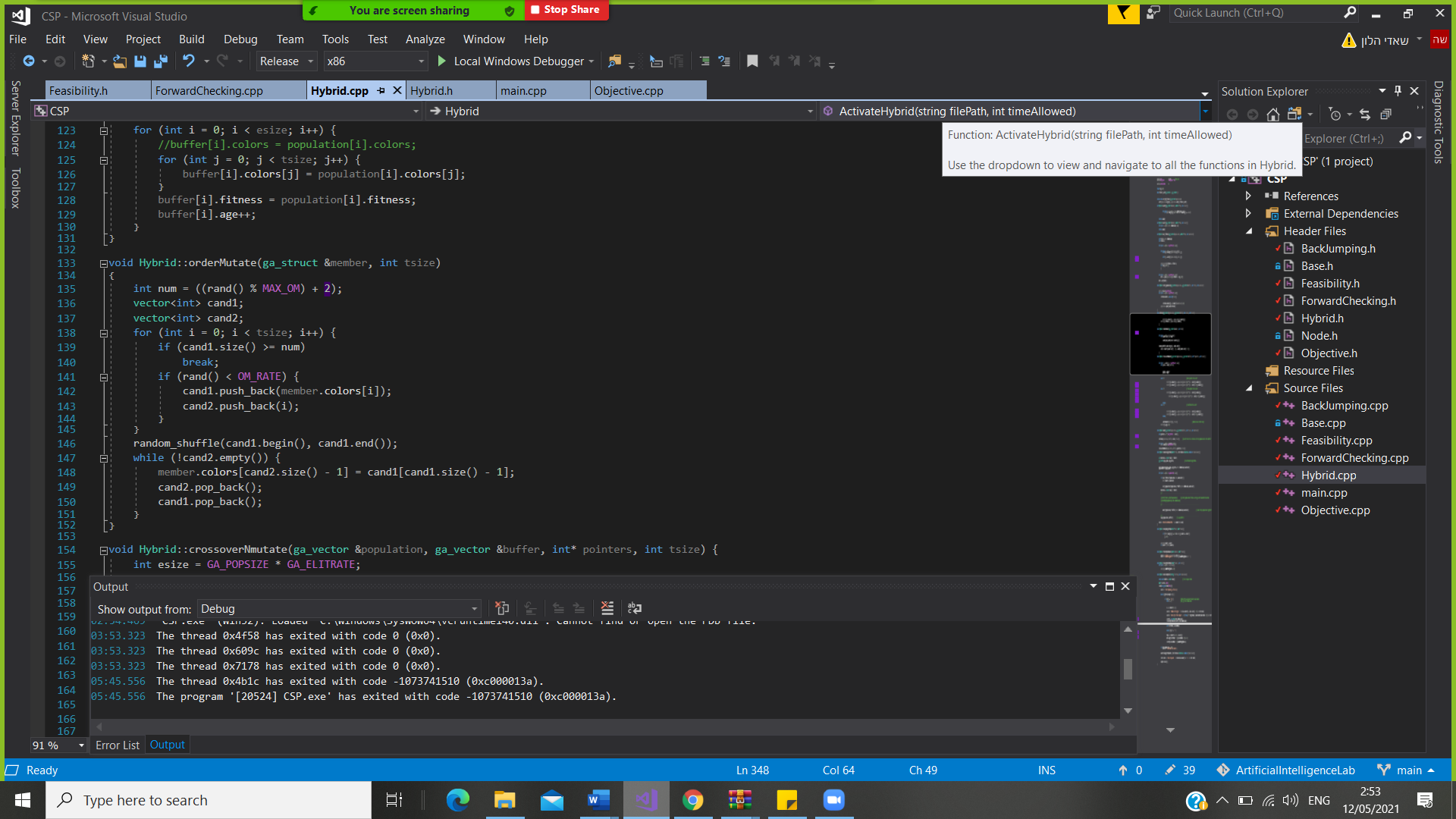
בעזרת אלגוריתם greedy נקבל מספר צביעות חוקי מסוים k ומתחילים כל פעם להקטין את k ולקבל גרף חוקי (הגישה הפיזבילית)

* חישוב ה-Fitness:



* שיטת מוטציה:

Order Mutation (OM) changing the order of some (few) vertices.



**חלק ה':**

**ביצועים הגישה הפיזבילית** **עם אלגוריתם MINCONFLICT:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| #Colors | Elapsed Time | |States| | |V| | |E| | Test name |
| 4 | 15.0545 | 28163480 | 11 | 20 | myciel3.col |
| 5 | 15.2131 | 9105431 | 23 | 71 | myciel4.col |
| 6 | 15.1385 | 5444599 | 47 | 236 | myciel5.col |
| 7 | 15.7799 | 3711061 | 95 | 755 | myciel6.col |
| 9 | 15.5869 | 1012471 | 128 | 774 | miles250.col |
| 7 | 15.0615 | 12713987 | 25 | 320 | queen5\_5.col |
| 9 | 15.0922 | 7898449 | 36 | 580 | queen6\_6.col |
| 9 | 15.1069 | 6883177 | 49 | 952 | queen7\_7.col |

אפשר לראות שהמימוש עם אלגוריתם MINIMAL CONFLICT, יכול למצוא צביעה מינימלית של גרפים צפופים בזמן סביר .

***Simulated Annealing***

***Kempe Chain***

***הגישה ששמה פונקציית המטרה במרכזה*:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| #Colors | Elapsed Time | |States| | |V| | |E| | Test name |
| 4 | 10.0104 | 8968243 | 11 | 20 | myciel3.col |
| 5 | 10.0276 | 3935773 | 23 | 71 | myciel4.col |
| 6 | 10.0592 | 1731896 | 47 | 236 | myciel5.col |
| 7 | 10.189 | 1033012 | 95 | 755 | myciel6.col |
| 8 | 10.2417 | 2956805 | 128 | 774 | miles250.col |
| 5 | 10.044 | 2844988 | 25 | 320 | queen5\_5.col |
| 7 | 10.0635 | 2346187 | 36 | 580 | queen6\_6.col |
| 7 | 10.0777 | 1811136 | 49 | 952 | queen7\_7.col |

אנחנו ניסינו לעבוד עם ה MINIMAL CONFLICT, וזה נתן לנו תוצאות רחוקות יחסית לכן ניסנו את ה SIMULATED וראינו שהמימוש עם אלגוריתםANNEALING SIMULATED, נותן לנו תוצאות יותר טובות .

**ביצועים הגישה ההיברידית עם אלגוריתם GENETIC:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| #Colors | Elapsed Time | |Generations| | |V| | |E| | Test name |
| 4 | 2.84535 | 1000 | 11 | 20 | myciel3.col |
| 5 | 6.33858 | 1000 | 23 | 71 | myciel4.col |
| 6 | 10.1301 | 482 | 47 | 236 | myciel5.col |
| 7 | 20.207 | 273 | 95 | 755 | myciel6.col |
| 9 | 20.3205 | 248 | 128 | 774 | miles250.col |
| 6 | 6.56133 | 1000 | 25 | 320 | queen5\_5.col |
| 9 | 12.5221 | 1000 | 36 | 580 | queen6\_6.col |
| 10 | 20.1067 | 940 | 49 | 952 | queen7\_7.col |

לא היה לנו ברירה רק לממש את ה GENETIC כי המרצה ביקש לממש את GENETIC ALGORITM עם אחת הגישות