הוא גילה, בדומה לדיסטרים והבדיל כמה דיסטרים הקיד מריק (כדי למנוע את הגלגל הלא). הגינו את אלגוריתם סתי גולמן שהקיבל הרי פירוק כמה דיסטרים לא אחד בורש ובאמצע את הדרך הכבדה יותר. השיטה השניה היא באמצעות משפט Top בו אנו הוכיח מלמטה למעלה ומוחרים תנאי את המסלול הארוך יותר.

לאחרון סתי גולמן אומר כי אם ב. תתו הגרף כדי שיש מספר זוגי דיסטרים מ שאנו נצרך $m+1$ דיסטרים. אחרת, אנו נקיים לכמות הדיסטרים כמו בתת הגף הקודם יותר.

אלגוריתם סתי גולמן מגיע לכ יש לנו מספר דיסטרים אינסוף. במציאות זה לא כך. ולכן נצטרך לסיים להגבלה תיכון של דיסטרים לפני כדון ותלה.

הוא הבדיל לא אלגוריתם לבעיות ארבעים שהשימוש בו להקצאת דיסטרים. השלב מאז כמו הוא הוא הינו השלב הראשון שבו אנו מקבלים תוכנת אמבלר ומתגרים מבנים סבו לגמרי (שיכון) הוא הרבה יותר קרוב מהתוכנית שהתקבלה בשלב הקודם. בנוסף, הקיד מתחשב בהקצאת פונקציית של הבעיה מלבד הקיד רק).

בהמשך מהאינטרנט הינו: בהינתן מ דיסטרים, הוא ניתן לבעיה על המקרה מתנאים שלנו ב- מ בבעיה ד שם לגיל אינמנו ובעיה שנצטרך לבצע "שפיכה" של תלת מהדיסטרים לאיכרון. במקרה כזה, עולה השאלה "איזו דיסטרים כדאי לפנות". בעיית בעיות הבעיה הינה NPC ולכן לא נמצא אלגוריתם מדויק ויעיל אינה פלינומיאלי לנו. (חישוב נויא ארוך). המקום מצאת, אנו נשתמש באלגוריתם אשר ומגיע פתרון מקורב. אם האלגוריתם ומגיע תשובה שהבעיה ניתן לבעיה ב-מ דיסטרים - הרשאה ובעיה, אם קיבלנו לא לפחות בעיה ניתן לבעיה אך אין בהכרחיות לחיות בשאים (כי האלגוריתם דומיסט).

ושנה מספר אלגוריתמים לבדיקת בעיות של בעיה. בעיקר כלי האלגוריתמים דומים (שקשה) תחילה אנו לבנות בעיה תלויות המתאים לתיכון שלנו. בהינתן בעיה התלויות, אנחנו מנסים לבדוק את הבעיה בהמשך הבדה למספר הדיסטרים שיש לנו. (תמיד ההבלה ש-ב שבנים לא יכול איתו בבעיה). אם אנחנו מגלים לבדוק את הבעיה - מגיין. אחרת ננסה לבדוק שמה כה של משתנה לאיכרון וננסה לבדוק שוב. נמשיך כך עד שנקבלת נגה חיוב לקבלת מתישהו).

<u>בעיה תלויות</u> - הבעיה אינו מכיון. הדמיונות בבעיה הוא המשתנים החיים בכל שלבי התוכנית (לכל משתנה יש רמות אקראי אם לא הינו חו במספר מקומות שונים). נמצא הקשר בין 2 דמיונות אם התוכן של המשתנים שהם מאיים ברוב לחיות בדיסטרים כי מנוגדים. הגדרה נוספת לקשר הינה: "2 משתנים נמשבים קשורים אם אחד מהם חו כשהשני מוגדר".

בשקפות 11 אנו חשב איך בדיקה בכל שורה איזו משתנים חיים בשקפות 12 אנו רואים סדר ובו הרשמה של משתנים חיים סביב כל שורה בקוד ואת הבעיה שנוצר. הנתון מקוקות מחברת בין דמיונות אשר לא מגיעה (לא בעיה). דמיון הבעיה בעיות בבעיה מיוצגים את הרעיון שהם יכולים לשבת בעיות הדיסטר

הינו הסיטואציה הקודמת שכבר לצרות משתנים חיים אנו לחלק את הקוד למשתנים חיים כלל היא סדרתית פעולות אשר לא מגיעה לשאת מרכזיות לבינקדיות. דומה בשקפות 11, reaching def הינה הקבוצה של המשתנים שלהם המשנים בקוד.

מאחר שמבצעים את הבעיה, אנו להחליט האם הוא ניתן לבעיה. אנחנו מנמיכים כי מספר הדיסטרים בשיטתנו פשוט להיות בטוח. נניח שיש לנו 3-3 כיוון שהבעיה של בעיה ב-3 בעיות הוא ב- P. בנוסף, נדרה מספר הבעיה כבר דרגתו של פעולת Move (מבוי שפיכה).

<u>מסקנה</u> אם יש בעיה בגודל מסוים הקשה מדי (לא מספר הבעיה), אין השפעה לא בעיה הבעיה

על משטח לא מתבסס האלגוריתם הראשון שנתנו ניקח את כל המתחים שרשמתם קודם מ-A ונורמל אותם מחדש. דומה שהורדנו וכל בורגלות לבדוק בדיוק שיש מהשפעים.

לאחר מכן נבדוק אוטרדות עד אשר "נתקע" או נגיע לשל ריק. אם "נתקענו" נחזור לשלב אחרת נחזור כך ו כל אוטרדה נורוד את כל המתחים שרשמתם קודם מ-A ואת כל הקשתות המחוברות אליהם.

האלגוריתם הוא וורסטי. ואינו מבוייק. (תשובה חיובית היא רגולרית).

אם "נתקענו" (בלומר אין נתון לבדוק ב-A ב בטוח), "נזרוק" אחד מהמתחים לגורמין ונסה שוב לבדוק את השל ללא הדומה שנתקענו.

השאלה הכבדה ביותר באלגוריתם שבשקופית 14 הינו שאלה בנוגע לשל התלוות (אפשר לצמצם רידת האלגוריתם).

כדי להתאים אלה דומה "לזרוק" (שתמש בהחלטה וורסטית הבוחרת מ"ב דרכת הדומה (נזרקים דרגה מבית). דרך נוספת היא לבדוק אלה משתנה מי יותר ואת הרבוסטר שחי זמן רב יותר לזרוק. דרך שלישית היא בדיקה כמה פעמים משתמשים במשתנה בתוך לולאה ומיון לולאה (ונזרקים את המשתנה עם המינה הגבוה ביותר).

לאחר שיש לנו מסודרים את כל המתחים מהשל (באלגוריתם נבחר מספר הדבקים שנרצה לבצע בשל) אנו בוחרים את הדבקים עם המבנים המתאימים עם יש המתפנים ומבצעים בכוח שם הדומה, אחרת, כל שלב נוסף דומה ונחזור לו בבנה השינה מהדבקים של שכנים בלתי השלב (ואת אין - נשתמש בבנה חדש).

יושם מבנות שלישית בעת "לזרוק" משתנים: עדיף כל משתנה אנו מגדירים את Priority לוקח שמתשב אותו רק (מספר הפעמים מחוץ-ללולאה + פ"א שימוש בתוך לולאה) תלוי הדרגה. (דומה כשקופית 30). אחרת נעשה קודם כל מישהו עם עדיפות נמוכה. התשובה הוא שכבר שמשתמשים בהשתנה יותר, תשלם יותר עדיפה הרבוסטה. הסיבה המנחה הוא שהמקבב של מספר השימושים בלולאה הוא גבוה. מישהו מחוץ ללולאה נמצא הנחה שלולאה רבה מספר פעמים ולכן לא משתנים בה הרבה פעמים יותר.

לאחר שהחזנו על "לזרוקה" של משתנה, אנו נוסיף לאמר כל חשלה של המשתנה הקודמת המשבשים את התנאי של המשתנים לזרוקים.

לאחר האלגוריתם, יתכן ונקבל קוד עם "שטויות". כלומר עם שורות אלו משמעות (כמו R1=R ו כ). לכן נוכל לחתוך שורות מיותרות ובסוף נקבל קוד מדומה כל הנכונה.